University of Niš
Faculty of Sciences and Mathematics
Department of Computer Science

Dejan Mančev

Training Structured Classifiers for Different Loss Functions with the Application to Sequence Labeling Problems

PhD thesis

Niš, 2015.

Univerzitet u Nišu
Prirodno-matematički fakultet
Departman za računarske nauke

Dejan Mančev

Treniranje strukturnih klasifikatora za različite funkcije gubitaka sa primenom na probleme klasifikovanja sekvenci

Doktorska disertacija

Niš, 2015.

COMMITTEE FOR THESIS DEFENCE

ADVISOR: (COMMITTEE MEMBER)	**Branimir Todorović** Associate professor Faculty of Sciences and Mathematics University of Niš
COMMITTEE CHAIR:	**Miroslav Ćirić** Full professor Faculty of Sciences and Mathematics University of Niš
COMMITTEE MEMBER:	**Predrag Stanimirović** Full professor Faculty of Sciences and Mathematics University of Niš
COMMITTEE MEMBER:	**Miomir Stanković** Full professor Faculty of Occupational Safety University of Niš
COMMITTEE MEMBER:	**Leonid Stoimenov** Full professor Faculty of Electronic Engineering University of Niš

DATE OF DEFENSE: _____

Прилог 4/1

ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ
НИШ

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		монографска
Тип записа, **ТЗ**:		текстуални / графички
Врста рада, **ВР**:		докторска дисертација
Аутор, **АУ**:		Дејан Манчев
Ментор, **МН**:		Бранимир Тодоровић
Наслов рада, **НР**:		Тренирање структурних класификатора за различите функције губитака са применом на проблеме класификовања секвенци
Језик публикације, **ЈП**:		енглески
Језик извода, **ЈИ**:		српски
Земља публиковања, **ЗП**:		Србија
Уже географско подручје, **УГП**:		Србија
Година, **ГО**:		2015
Издавач, **ИЗ**:		ауторски репринт
Место и адреса, **МА**:		Ниш, Вишеградска 33.
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		123 стр., граф. прикази
Научна област, **НО**:		Рачунарске науке
Научна дисциплина, **НД**:		Вештачка интелигенција
Предметна одредница/Кључне речи, **ПО**:		структурно учење, класификација секвенци
УДК		004.852:004.89(043.3)
Чува се, **ЧУ**:		библиотека
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У тези се разматрају алгоритми за обучавање структурних класификатора дефинисани са различитим функцијама губитака. Уведен је примарни субградијентни метод за оптимизацију структурне усредњене функције губитка, више екстензија класификатора који максимизирају маргину користећи к најбољих структура, секвенцијални дуални метод за оптимизацију двоструко преломљене функције губитка, разматрани су алгоритми за декодирање као и надовезивање класификатора у циљу унапређења резултата препознавања. Представљене су теоријске карактеристике уведених класификатора као и експериментални резултати на специфичним проблемима класификације секвенци који се јављају у обради говорног језика.
Датум прихватања теме, **ДП**:		24.11.2014.
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	
	Члан,ментор:	
	Члан:	
	Члан:	
	Члан:	

Прилог 4/2

ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ
НИШ

KEY WORDS DOCUMENTATION

Accession number, **ANO**:		
Identification number, **INO**:		
Document type, **DT**:		monograph
Type of record, **TR**:		textual / graphic
Contents code, **CC**:		doctoral dissertation
Author, **AU**:		Dejan Mančev
Mentor, **MN**:		Branimir Todorović
Title, **TI**:		Training Structured Classifiers for Different Loss Functions with the Application to Sequence Labeling Problems
Language of text, **LT**:		English
Language of abstract, **LA**:		Serbian
Country of publication, **CP**:		Serbia
Locality of publication, **LP**:		Serbia
Publication year, **PY**:		2015
Publisher, **PB**:		author's reprint
Publication place, **PP**:		Niš, Višegradska 33.
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		123 p. ; graphic representations
Scientific field, **SF**:		Computer Science
Scientific discipline, **SD**:		Artificial intelligence
Subject/Key words, **S/KW**:		structured learning, sequence labeling
UC		004.852:004.89(043.3)
Holding data, **HD**:		library
Note, **N**:		
Abstract, **AB**:		This thesis presents algorithms for training structured classifiers over different loss functions. It introduces a new primal subgradient method for the optimization of averaged sum loss, several extensions of max-margin classifiers to the k-best case, a sequential dual method for the structured ramp loss optimization. It considers different decoding algorithms over semirings and presents an organization of a two-structured-model committee in order to improve the results. It includes theoretical analysis of introduced algorithms, as well as experimental results on sequence labelling problems in natural language processing.
Accepted by the Scientific Board on, **ASB**:		11/24/14
Defended on, **DE**:		
Defended Board, **DB**:	President:	
	Member,Mentor:	
	Member:	
	Member:	
	Member:	

It gives me great pleasure to write this page. This means that this thesis and my schooling is nearing completion, so now I can think back and use this opportunity to express my gratitude to those who supported me throughout the course of my PhD studies and made the work on the thesis easier and more pleasant.

I would like to express my sincere gratitude to my advisor, Professor Branimir Todorović, first for introducing me to the field of machine learning and inspiring me to choose this area, and then for the excellent cooperation and great commitment during the preparation of the PhD thesis.

I would like to thank Professor Miroslav Ćirić, the leader of the project under which this thesis was realized, for financial assistance through project funds.

I am also very thankful to Dr. Velimir Ilić and Professor Miomir Stanković on the joint paper on the gradient computation of conditional random fields.

I would especially like to thank my lovely wife Ivana for the emotional support during my studies, and for the constant love and encouragement.

To all my friends whom I neglected due to the preparation of this thesis, my warmest thanks for your friendship, support and understanding.

Finally, I would like to thank my parents, who have been constantly directing me to this goal from an early age.

Table of contents

List of figures

List of tables

Nomenclature

Symbols

$\boldsymbol{\alpha}$	Changes in dual parameters
δ	Step size
η_t	Step size dependent on iteration t
$\boldsymbol{\lambda}$	Dual parameters
λ	Regularization parameter
ν_t	Confidence at position t
ξ	Slack variable for SVM
τ	KKT tolerance
a_t	Attribute at position t
$\boldsymbol{g}$	Gradient vector
$\mathbf{u}$	Arbitrary parameters
$\mathbf{w}$	Parameters in primal space
$\overline{\mathbf{w}}$	Averaged parameters
w_t	Observation at position t
$\boldsymbol{x}^n$	nth observation
$\boldsymbol{y}$	Arbitrary structure
y_t	Label at the tth position of sequence $\boldsymbol{y}$
$\boldsymbol{y}^n$	Original structure for the nth observation
$\widehat{\boldsymbol{y}}^n$	Predicted structure on the nth example without a cost function
$\widetilde{\boldsymbol{y}}^n$	Predicted structure on the nth example with a cost function
$\overline{\boldsymbol{y}}^n$	Predicted structure on the nth example with a negative cost function
C	Regularization parameter $1/\lambda$ divided by the no. of examples used in optimization
$\mathbf{K}$	Kernel matrix
$\mathcal{L}$	Lagrangian function
M	Dimension of parameters $\mathbf{w}$
N	Number of training examples
T	Sequence length

Sets

$\mathscr{A}_{\mathbf{w}}$	Violation set for the concave part of ramp loss with parameters $\mathbf{w}$
$\mathscr{B}_{\mathbf{w}}^{k,n}$	k-best structures on the nth example with parameter $\mathbf{w}$
$\mathscr{D}$	Training data
$\mathbb{R}$	Real numbers
S_n	Prediction violation structures
$\mathcal{Y}$	All possible labels for an element of the observation sequence
$\mathscr{Y}(\boldsymbol{x}^n)$	All structures for the nth observation
$\mathscr{Y}_{-n}$	All structures for the nth observation except the original structure
$\mathscr{W}$	Working set of active structures

Acronyms

ASL	Averaged sum loss
CCCP	Concave-convex procedure
CRF	Conditional random field
FB	Forward-backward
HMM	Hidden Markov model
KKT	Karush-Kuhn Tucker
MIRA	Margin Infused Relaxed Algorithm
MM	Max-margin
NER	Named entity recognition
PA	Passive-aggressive
POS	Part-of-speech
RPA	Restricted passive-aggressive
SDM	Sequential dual method
SMO	Sequential minimal optimization
SSVM	Structured support vector machine
SVM	Support vector machine
WP	Without projection

Functions

$\alpha_t(y)$	Forward vector at the tth position for label y
$\beta_t(y)$	Backward vector at the tth position for label y
$\boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)$	Feature vector for the transition from y_{t-1} to y_t on observation $\boldsymbol{x}$
$h_{\mathbf{w}}(\boldsymbol{x})$	Classification function that maps $\boldsymbol{x}$ to the corresponding structure
$l(c', c'')$	Cost between two labels
$\ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$	Loss function for the structure $\boldsymbol{y}$ on the nth example with par. $\mathbf{w}$
$\ell_n(\mathbf{w})$	Loss function for the nth example
$\ell_n^{\text{MM}}(\mathbf{w})$	Max-margin loss for the nth example i.e the structured hinge loss
$\ell_n^{\text{AS}}(\mathbf{w})$	Average sum loss for the nth example
$\ell_n^{\text{C}}(\mathbf{w})$	Concave part of the ramp loss for the nth example
$\ell_n^{k\text{best}}(\mathbf{w})$	k-best loss on the nth example
$\ell_n^{\text{Ramp}}(\mathbf{w})$	Ramp loss on the nth example
$\mathbf{F}(\boldsymbol{x}, \boldsymbol{y})$	Feature vector for the observation $\boldsymbol{x}$ and structure $\boldsymbol{y}$
$\Delta\mathbf{F}_n(\boldsymbol{y})$	Difference between feature vectors for structures $\boldsymbol{y}^n$ and $\boldsymbol{y}$
$\mathscr{L}(\mathbf{w})$	Likelihood function
$L(\boldsymbol{y}', \boldsymbol{y}'')$	Cost function between two structures e.g. the Hamming distance
$Q_{\boldsymbol{x},t}(c, c')$	Transition score from label c to c' at the tth position of the observation $\boldsymbol{x}$
$Z(\boldsymbol{x}; \mathbf{w})$	Normalization function for CRFs on observation $\boldsymbol{x}$

Chapter 1

Introduction

A year spent in artificial intelligence
is enough to make one believe in God.
Alan Jay Perlis (1922–1990)

The world's vast amount of information consisting of unclassified data is constantly growing and it requires to be processed intelligently. Classifying unstructured data such as news, articles, audio conversations, e-mails, scanned images, video information and many other that originate from a large number of sources has been the object of study of various research fields which include information extraction, spam detection, optical character recognition and speech recognition. They are all based on learning patterns with the ability to generalize the obtained knowledge and apply it to unseen data. Many of these problems are complex and involve learning with data which have their internal structure. For example, the internal structure of text data is represented by different grammatical and lexical features and relationships between elements. We will focus here on solutions for sequence labeling problems which belong to the information extraction area, where structured models will be used to learn complex structured patterns.

Sequence labeling assumes assigning a label to each element of a sequence. For example, if there is a sentence in natural language, we can define a task that will detect the grammatical category of each word, and say whether it is a verb, noun, adjective, adverb, determiner, preposition etc. This task is called part-of-speech (POS) tagging. At first glance, the task may look easy, we will use a dictionary to determine the word category. However, a problem rises as the same word can have different meanings and belong to different categories depending on the context in a particular sequence. One way to overcome this problem is to create rules which will determine the word class depending on the context of its use. There have been many attempts to create such rule-based systems with various success rates−the first ones being (Greene and Rubin, 1971; Klein and Simmons, 1963), followed by (Brill, 1992, 1994; Voutilainen, 1995). This requires excellent linguistic skills and requires adding a multitude of special cases and

exceptions to the system so that it can perform the classification task with high accuracy. This kind of a system also has a few drawbacks. First, covering all possible exceptions and special cases can be very time-consuming and may result in the system being too large and hard to organize. But most importantly, the rules are accurate for a specific task only, so if we change a language, for example, they cannot be applied, leaving us with a task of writing a completely new set of rules.

Another example of sequence labeling problems in natural language processing is the shallow parsing problem. Shallow parsing (Hammerton et al., 2002) identifies non-overlapping text segments which correspond to certain syntactic units. It is usually a step preceding full parsing and following POS tagging. The next example is named entity recognition (Nadeau and Sekine, 2007), where we need to recognize certain entities in a text, like the names of persons, organizations, locations, money, dates, etc. It is an important problem in information extraction systems. Sequence labeling problems also exist in other areas. For example, in bioinformatics the problem is finding DNA sequence patterns which can cause a specific disease (Xing et al., 2010), classifying protein sequences into categories corresponding to their role which helps to find a new protein (Deshpande and Karypis, 2002) or predicting new genes according to the known genes annotated on a related DNA sequence (Meyer and Durbin, 2004). In computer security, the problem is creating anomaly detection systems (Lane, 2000) and intrusion detection systems which distinguish between legitimate and illegitimate activities in system calls (Warrender et al., 1999).

To overcome problems in rule-based systems, one idea is to create a statistical model which will be able to learn the dependencies between the data and its classes. Once the learning model is defined, it can be trained on different domains (e.g. different languages) and also on different tasks. If we want to train such a model which will classify words in predefined classes, we will need a labeled sample of data, called a training corpus, to train our model and such approach belongs to *supervised machine learning*. The elementary task in supervised machine learning is binary classification which classifies each observation independently into one of two possible classes. Even though we can apply multiple binaries or a multiclass classifier to a sequence labeling problem, labeling each element of a sequence independently will not produce the highest results possible.

The traditional way of dealing with sequence labeling problems is using Hidden Markov models (HMMs) (Rabiner, 1990). These models have been successfully applied to different sequence labeling problems: POS tagging (Kupiec, 1992), name entity recognition (Zhou and Su, 2002), shallow parsing (Molina and Pla, 2002), the analysis of RNA structures (Durbin et al., 1998). They use the first order Markov property which says that the probability for the next state depends only on the current state and not on the ones preceding it. A HMM consists of hidden states, a set of observations which can be generated from each hidden state and three types of probabilities which it models, the probability of transition between two hidden states,

the probability that a hidden state generates an observation as its output and the probability of initial states. These probabilities are usually learned by maximizing the joint likelihood on training data. In order to make the problem tractable, HMMs must assume a conditional independence of observations given labels, and as generative models they are not able to include different dependencies, such as that the current transition may be dependent on the whole context around the current observation. These disadvantages have been overcome by introducing structured classifiers.

Structured classifiers deal with the classification problem where the output of a classifier is represented with a structure, not with a single label, as it was the case with binary and multiclass classifiers. The structures are composed of smaller atoms, which represent the output classes and have different dependencies among them. In the sequence labeling problem, the output structures are represented as chain structures, but in other problems they can be represented by a tree, lattice or graph in a general case. These classifiers are able to include various relationships between the input observation and output labels, which was one of the limitations in HMMs. As a consequence of their ability to deal with different relationships, they outperform standard binary and multiclass classifiers, but at the cost that they are more complex to train and require an inference during the training procedure. Over the last 15 years, various types of structured models and their training procedures have been proposed, including: the structured version of the Perceptron algorithm (Collins, 2002) as a simplest structured online algorithm, structured versions of support vector machines (Tsochantaridis et al., 2005), conditional random fields as probabilistic discriminative models (Lafferty et al., 2001). What connects all these models is that behind each of them is an optimization procedure with a corresponding loss function. In this thesis we consider training structured models with different loss functions and focus on their application in sequence labeling problems where the structures are in a chain form.

Outline of the thesis

The summary of the rest of the chapters follows below.

Chapter 2 defines a generalized learning problem as an optimization problem of a loss function and a regularization function. It gives an introduction to the structured classification problem and discusses the training of two structured models with the structured hinge and CRF loss function. As training a structured model requires a decoding algorithm both in a training and a testing phase, the last part of the chapter discusses different decoding algorithms. They are presented in a general form as forward algorithms - message passing schemes over semirings, where different choices lead to different inference procedures for a specific purpose. This part is based on the paper:

- Ilić, V., Mančev, D., Todorović, B., and Stanković, M. (2012). Gradient computation in linear-chain conditional random fields using the entropy message passing algorithm. *Pattern Recognition Letters*, 33(13):1776–1784.

where the forward-backward algorithm uses specific semirings for a numerically stable training of linear chain conditional random fields and their training on long observation sequences.

Chapter 3 considers the training of structured models using k sequences with the highest score. We introduce four k-best extensions of popular structured learning algorithms: the sequential dual method, LaRank, passive-aggressive and perceptron algorithm. We consider theoretical properties of k-best algorithms and also introduce a restricted version of the passive-aggressive algorithm for which we show that it satisfies a cumulative prediction loss bound similar to the single best case. Also, an experimental evaluation is conducted for the introduced algorithms and the influence of the parameter k. The chapter is based on the results from

- Mančev, D. and Todorović, B. (2015). k-best max-margin approaches for sequence labeling. *Computer Science and Information Systems (submitted)*.

Chapter 4 presents a committee of two structured models, the support vector machines and the conditional random fields. They are trained in a sequence where the second model uses the output from the first one. Beside this output, k-best sequences are used to extract confidence of the prediction which is included as an additional feature. The chapter discusses the organization of the training and testing phase and presents experimental results of the committee. The chapter is based on the paper:

- Mančev, D. and Todorović, B. (2012). Confidence based learning of a two-model committee for sequence labeling. In *11th Symposium on Neural Network Applications in Electrical Engineering (NEUREL)*, pages 167–170. IEEE.

Chapter 5 presents a sequential dual method for the optimization of a structured ramp loss. A concave-convex procedure is used to convert a non-convex problem into a series of convex ones, after which a convex optimization is performed in a dual space by using sequential minimal optimization. The chapter presents the results on sequence labeling problems as well as on the artificial data in which the method is exposed to outlayers. The chapter is based on the following paper:

- Mančev, D. (2015). A sequential dual method for the structured ramp loss minimization. *Facta Universitatis, Series: Mathematics and Informatics*, 30(1):13–28.

Chapter 6 introduces an averaged sum loss and a primal sub-gradient method for the optimization. The chapter includes a theoretical analysis of the proposed method presenting the bound of the cumulative prediction loss. Experimental results contain different testing scenarios as well as a comparison with other popular structured classification algorithms on sequence labeling tasks. The chapter is based on the paper:

- Mančev, D. and Todorović, B. (2014). A primal sub-gradient method for structured classification with the averaged sum loss. *International Journal of Applied Mathematics and Computer Science (AMCS)*, 24(4):917–930.

Chapter 2

Structured classification problems

This chapter introduces the learning problem in a general form as an optimization problem with a regularization and a loss function. It gives an introduction to structured classification problems and reviews two important structured models: structured max-margin classifiers and conditional random fields. It summarizes different loss functions used in the binary and the structured case. The chapter ends with a description of inference procedures used for training structured classifiers. They are presented as an inference over different semirings.

2.1 Generalized learning problem

Let $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$ be the training examples, where $\boldsymbol{x}^n \in \mathscr{X}$ represents an observation and $\boldsymbol{y}^n \in \mathscr{Y}(\boldsymbol{x}^n)$ its corresponding assignment. For example, in binary classification problems $\boldsymbol{y}^n$ represents a single label describing which of the two possible classes an observation $\boldsymbol{x}^n$ belongs to and the output set can be represented as $\mathscr{Y}(\boldsymbol{x}^n) = \{-1, 1\}$. In a multi-class classification problem, $\boldsymbol{y}^n$ represents one of many possible labels, while in the structured case it represents a structure. In supervised learning problems, we seek for a function $h_{\mathbf{w}}(\boldsymbol{x})$ over a training set $\mathscr{D}$, which will make a low error while mapping arbitrary elements $\boldsymbol{x}$ from an input set $\mathscr{X}$ into an assignment $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x})$. The standard way of finding such a function is to assume its form (such as a linear function over parameters) and then estimate parameters $\mathbf{w}$ by minimizing the regularized empirical risk on $\mathscr{D}$

$$\min_{\mathbf{w}} \frac{\lambda}{2}\Omega(\mathbf{w}) + R^{\mathscr{D}}_{emp}(\mathbf{w}), \tag{2.1}$$

where $\Omega(\mathbf{w})$ is a regularization function introduced in order to avoid overfitting and $\lambda \in \mathbb{R}^+$ is the regularization parameter. The function $R^{\mathscr{D}}_{emp}(\mathbf{w})$ represents the *empirical risk* on $\mathscr{D}$

$$R^{\mathscr{D}}_{emp}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ell_n(\mathbf{w}), \tag{2.2}$$

Table 2.1: Examples of loss functions for the binary and the structured case. In the binary case, the loss function is presented for an arbitrary observation $\boldsymbol{x}$, where y denotes its binary label. In the structured case, the loss function is presented over the nth observation $\boldsymbol{x}^n$, where $\boldsymbol{y}^n$ represents the true structure for $\boldsymbol{x}^n$, $\mathscr{Y}(\boldsymbol{x}^n)$ is the set of all structures, and $\mathscr{Y}_{-n} = \mathscr{Y}(\boldsymbol{x}^n) \setminus \boldsymbol{y}^n$.

Binary case	Loss function $\ell(\mathbf{w})$
Perceptron loss (Freund and Schapire, 1999; Rosenblatt, 1958)	$\max(0, -y\,\mathbf{w}^\mathsf{T}\boldsymbol{x})$
Hinge loss (Bennett and Mangasarian, 1992)	$\max(0, 1 - y\,\mathbf{w}^\mathsf{T}\boldsymbol{x})$
Squared hinge loss (Keerthi and DeCoste, 2005)	$\max(0, 1 - y\,\mathbf{w}^\mathsf{T}\boldsymbol{x})^2$
Logistic loss (Collins et al., 2002)	$\log(1 + \mathrm{e}^{-y\,\mathbf{w}^\mathsf{T}\boldsymbol{x}})$
Ramp loss (Collobert et al., 2006)	$\min(2, \max(0, 1 - y\,\mathbf{w}^\mathsf{T}\boldsymbol{x}))$

Structured case	Loss function $\ell_n(\mathbf{w})$
Perceptron loss (Collins, 2002)	$\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} -\mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y})$
Hinge loss (Tsochantaridis et al., 2005)	$\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} L(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y})$
Squared hinge loss (Chang and Yih, 2013)	$\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} (L(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y}))^2$
Scaled hinge loss (Tsochantaridis et al., 2005)	$\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} L(\boldsymbol{y}^n, \boldsymbol{y})(1 - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y}))$
k-best loss (Crammer et al., 2005; Mančev and Todorović, 2014)	$\frac{1}{k}\sum_{\boldsymbol{y}\in\mathscr{Y}^{k,n}_{\mathbf{w}}} \max(0, L(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y}))$
Averaged sum loss (Mančev and Todorović, 2014)	$\frac{1}{\lvert\mathscr{Y}_{-n}\rvert}\sum_{\boldsymbol{y}\in\mathscr{Y}_{-n}} \max(0, L(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y}))$
CRF loss (Lafferty et al., 2001)	$\log\big(\sum_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \mathrm{e}^{\mathbf{w}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y})}\big) - \mathbf{w}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n)$
Softmax-margin CRF loss (Gimpel and Smith, 2010)	$\log\big(\sum_{\boldsymbol{y}} \mathrm{e}^{\mathbf{w}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}) + L(\boldsymbol{y}^n, \boldsymbol{y})}\big) - \mathbf{w}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n)$
Ramp loss (Do et al., 2008; Gimpel, 2012)	$\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \big(L(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y})\big)$ $-\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \big(-L(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y})\big)$

where $\ell_n(\mathbf{w})$ is the loss function on the nth example, measuring how close the corresponding structure $\boldsymbol{y}^n$ is to the prediction $h_{\mathbf{w}}(\boldsymbol{x}^n)$. There are various possibilities for choosing a combination of the regularization and loss function yielding to different machine learning algorithms. An example of a regularization function is L_2 regularization, for which we have the following

problem

$$\min_{\mathbf{w}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{n=1}^{N} \ell_n(\mathbf{w}), \tag{2.3}$$

that we will use in this thesis, where $\|\cdot\|$ will represent L_2 norm. For the regularization function, L_1 regularization (Tibshirani, 1996) can also be used, which imposes sparsity among the parameters, or an elastic net (Zou and Hastie, 2005), which combines L_1 and L_2 regularization. There are various choices for choosing a loss function. Table 2.1 summarizes the examples of different loss functions used in classification in the binary and the structured case. After choosing a regularization and a loss function, the next step is to solve the appropriate optimization problem. If the loss function is differentiable (e.g. the CRF loss function), we can apply gradient based methods for the optimization. However, not all functions are differentiable. For example, hinge loss is a non-differentiable convex function where we can apply sub-gradient methods for optimization (Shalev-Shwartz et al., 2007), or deal with quadratic optimization (Nocedal and Wright, 2006) after transforming the problem into dual space.

Once we choose the regularization and loss function with the appropriate optimization method, finally we need to choose the regularization coefficient λ. This coefficient controls the importance we give to the regularization function against the loss function. Setting the value of λ too low will involve too much adjustment of parameters to the data, which can lead to the problem of overfitting and thus a bad generalization ability of the model, while setting the value too high can result that the wanted mapping function is not learned. A good choice can be found using a cross validation technique (Kohavi, 1995).

In the next two sections, we will review optimizations for the structured hinge and the CRF loss.

2.2 Structured max-margin classifiers

Let $\mathcal{X}$ be an input alphabet and $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$ a training set, where each $\boldsymbol{x}^n \in \mathcal{X}^{T_n}$ represents an input sequence of length T_n with the corresponding structure $\boldsymbol{y}^n$. The set of all possible structures over the sequence $\boldsymbol{x}^n$ is denoted by $\mathscr{Y}(\boldsymbol{x}^n)$ and $\mathscr{Y}_{-n} = \mathscr{Y}(\boldsymbol{x}^n) \setminus \boldsymbol{y}^n$. In further discussion, we focus on the sequence labeling problem where $\mathscr{Y}(\boldsymbol{x}^n) = \mathcal{Y}^{T_n}$ and $\mathcal{Y}$ represents a set of possible labels for an element of $\mathcal{X}$. The linear classification problem is to learn a function $h_{\mathbf{w}}$ from the data $\mathscr{D}$, which maps every sequence $\boldsymbol{x}$ over the alphabet $\mathcal{X}$ to an element of $\mathscr{Y}(\boldsymbol{x})$ in the following form:

$$h_{\mathbf{w}}(\boldsymbol{x}) = \operatorname*{arg\,max}_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x})} \mathbf{w}^{\mathsf{T}} \mathbf{F}(\boldsymbol{x}, \boldsymbol{y}), \tag{2.4}$$

with parameters $\mathbf{w} \in \mathbb{R}^M$, where $\mathbf{F}(\boldsymbol{x}, \boldsymbol{y})$ represents a *global feature vector* measuring the compatibility of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Features If output structures are in a chain form, then a global feature vector is the sum over feature functions from all positions t in a sequence, i.e.

$$\mathbf{F}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{t=1}^{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t). \tag{2.5}$$

Each feature vector $\boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t) \in \mathbb{R}^M$ usually presents binary-coded active features of the current transition and the current context around the t-th observation. For a sequence labeling problem, let us define O as a set of all combinations of words and attributes which can occur in a context. As an example, a context around the tth observation can contain combinations like: there is a specific word w at position t while the previous word starts in lower case; a word at position t represents a special character while the next word is a digit; there is a word w at position t and w' at position $t-1$, etc. For each pair of labels $c, c' \in \mathcal{Y}$ and $o \in O$, there will be one element in a feature vector $\boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)$ with a value $\mathbb{1}[y_t = c, y_{t-1} = c', o \in \boldsymbol{x}^{(t)}]$, where $\boldsymbol{x}^{(t)}$ denotes all combinations from O that appear in the tth context of observation $\boldsymbol{x}$, while $\mathbb{1}$ denotes the indicator function whose value is one if all the arguments are true and zero otherwise. Then, the length of feature vectors will be $M = |\mathcal{Y}|^2 |O|$, which can be a large number. However, these vectors have only a small number of elements different from zero, so they can be represented as sparse vectors.

Primal-dual form One way of finding function $h_{\mathbf{w}}(\boldsymbol{x})$ is to estimate $\mathbf{w}$ via the max-margin approach (Vapnik, 1998), which leads to the structured version of support vector machine (Tsochantaridis et al., 2005). This approach considers the regularized empirical risk minimization (2.3) on $\mathscr{D}$ with the structured hinge loss function and one way in which it can be presented is as a constraint minimization problem in the following form:

$$\min_{\mathbf{w}, \boldsymbol{\xi}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{n=1}^{N} \xi_n \tag{2.6}$$

$$\text{s.t. } \mathbf{w}^{\mathsf{T}} \Delta\mathbf{F}_n(\boldsymbol{y}) \geq L(\boldsymbol{y}^n, \boldsymbol{y}) - \xi_n, \quad \forall n, \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n), \tag{2.7}$$

where $\Delta\mathbf{F}_n(\boldsymbol{y}) = \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n) - \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y})$. According to the constraints, the original sequence $\boldsymbol{y}^n$ should produce a greater score $\mathbf{w}^{\mathsf{T}}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n)$ than any other sequence at least for the size of the margin for that sequence. The cost function $L(\boldsymbol{y}^n, \boldsymbol{y})$, used for margin scaling, represents the cost of labeling the sequence $\boldsymbol{x}^n$ with $\boldsymbol{y}$ instead of $\boldsymbol{y}^n$. To handle a non-separable case, for each example $\boldsymbol{x}^n$ we assign non-negative slack variables ξ_n which control the penalty for its misclassification and the whole problem refers to the N-slack formulation with margin scaling. Since $L(\boldsymbol{y}^n, \boldsymbol{y}^n)$ is equal to zero, the constraint for the sequence $\boldsymbol{y}^n$ in (2.7) produces $\xi_n \geq 0$. The parameter $\lambda \in \mathbb{R}^+$ controls the trade-off between the slack variable penalty and the size of the margin. As the regularization parameter will differ through algorithms depending on

the number of examples that an optimization problem takes into account, to simplify further analysis and synchronize its different meanings, let us define

$$C = \frac{1}{\lambda |D|} \tag{2.8}$$

as a regularization parameter for the number of examples $|D|$ included in the optimization problem, which in the case of the problem (2.6)-(2.7) is equal to $C = 1/(\lambda N)$. As the cardinality of $\mathscr{Y}(\boldsymbol{x}^n)$ is exponential, the direct optimization is untraceable. A straightforward transformation of the primal problem (2.6)-(2.7), by introducing Lagrange multipliers $\boldsymbol{\lambda} = [\lambda_{n,\boldsymbol{y}}]_{n,\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)}$, leads to the dual optimization problem

$$\min_{\boldsymbol{\lambda}} \ \frac{1}{2}\boldsymbol{\lambda}^\mathsf{T}\boldsymbol{K}\boldsymbol{\lambda} - \boldsymbol{\lambda}^\mathsf{T}\mathbf{L} \tag{2.9}$$

$$\text{s.t.} \sum_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \lambda_{n,\boldsymbol{y}} = C, \ \forall n, \quad \lambda_{n,\boldsymbol{y}} \geq 0, \ \forall n, \ \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n), \tag{2.10}$$

where $\boldsymbol{K}$ is a kernel matrix defined with elements $K_{n,\boldsymbol{y},m,\boldsymbol{y}'} = \Delta\mathbf{F}_n(\boldsymbol{y})^\mathsf{T}\Delta\mathbf{F}_m(\boldsymbol{y}')$ for every $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$, $\boldsymbol{y}' \in \mathscr{Y}(\boldsymbol{x}^m)$ and $\mathbf{L}$ is a vector with elements $L_{n,\boldsymbol{y}} = L(\boldsymbol{y}^n, \boldsymbol{y})$. The primal parameters are expressed in terms of dual ones as

$$\mathbf{w} = \sum_{n=1}^{N} \sum_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \lambda_{n,\boldsymbol{y}} \Delta\mathbf{F}_n(\boldsymbol{y}). \tag{2.11}$$

The constraints in (2.10) allow us to perform the optimization restricted to a single example at a time (Taskar, 2004). Let $\boldsymbol{\alpha}_n$ denote changes in $\boldsymbol{\lambda}_n$ during the processing of the nth example, i.e. let the change of parameters be made according to the update $\lambda'_{n,\boldsymbol{y}} \leftarrow \lambda_{n,\boldsymbol{y}} + \alpha_{n,\boldsymbol{y}}$. In order for new dual parameters $\boldsymbol{\lambda}'_n$ to be feasible, the sum of $\alpha_{n,\boldsymbol{y}}$ should be zero as well as $\lambda'_{n,\boldsymbol{y}} \geq 0$. By dropping all terms in (2.9)-(2.10) which do not depend on $\boldsymbol{\alpha}_n$, we can rewrite this optimization restricted to the nth example with respect to the parameter change as

$$\min_{\boldsymbol{\alpha}_n} \ \frac{1}{2}\boldsymbol{\alpha}_n^\mathsf{T}\boldsymbol{K}_n\boldsymbol{\alpha}_n - \boldsymbol{\alpha}_n^\mathsf{T}(\mathbf{L}_n - \Delta\mathbf{F}^\mathsf{T}\mathbf{w}) \tag{2.12}$$

$$\text{s.t.} \sum_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \alpha_{n,\boldsymbol{y}} = 0, \quad \lambda_{n,\boldsymbol{y}} + \alpha_{n,\boldsymbol{y}} \geq 0, \ \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n), \tag{2.13}$$

where $\mathbf{L}_n$ is a vector with elements $L_{n,\boldsymbol{y}} = L(\boldsymbol{y}^n, \boldsymbol{y})$ and $\boldsymbol{K}_n$ is the block of the kernel matrix $\boldsymbol{K}$ corresponding to the nth example. Note that function (2.12) depends on $\mathbf{w}$, which corresponds to parameters $\lambda_{n,\boldsymbol{y}}$ according to (2.11), and thus the parameters after the update will be

$$\mathbf{w}' = \mathbf{w} + \sum_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \alpha_{n,\boldsymbol{y}} \Delta\mathbf{F}_n(\boldsymbol{y}). \tag{2.14}$$

The gradient of the dual function (2.12) is $\boldsymbol{g} = [g_{n,\boldsymbol{y}}]_{n,\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)}$, with elements

$$g_{n,\boldsymbol{y}} = \sum_{\boldsymbol{z}\in\mathscr{Y}(\boldsymbol{x}^n)} \alpha_{n,\boldsymbol{z}} K_{n,\boldsymbol{y},\boldsymbol{z}} - L(\boldsymbol{y}^n, \boldsymbol{y}) + \mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y}), \tag{2.15}$$

which is used to express the violation of Karush-Kuhn-Tucker (KKT) conditions (Kuhn and Tucker, 1951) for the problem (2.12)-(2.13) as

$$\max_{\boldsymbol{y}\in I_0} g_{n,\boldsymbol{y}} > \min_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} g_{n,\boldsymbol{y}}, \tag{2.16}$$

where $I_0 = \{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n) : \alpha_{n,\boldsymbol{y}} > -\lambda_{n,\boldsymbol{y}}\}$.

Again, the dual problem has exponentially many constraints and a full optimization is not possible in real time. There are several approaches to dealing with such optimization described in the related work.

Related work For the special case of a linearly decomposable loss, Taskar et al. (2004) present this problem as an equivalent polynomial-size formulation in the maximum margin Markov network by introducing marginal variables to which we can apply sequential minimal optimization (SMO) (Platt, 1999). On the other hand, without the previous assumption, we can seek a small set of constraints that is sufficient to approximate a solution by increasing the working set of constraints through iterations. Joachims et al. (2009) use the cutting plane method on the equivalent formulation with one slack variable shared across all data and build the working set of constraints with a separation oracle. Even though the algorithm finds a solution where constraints are violated no more than ϵ after $\mathcal{O}(\frac{1}{\epsilon})$ iterations, each iteration assumes finding a separation oracle, which can be time consuming for a larger number of examples. Balamurugan et al. (2011) present the sequential dual method (SDM) for structural SVMs in which they sequentially add the constraint generated for the best structure and apply the sequential minimal optimization (SMO) (Platt, 1999) with additional heuristics to increase speed. Jaggi et al. (2013) propose a block-coordinate version of Frank-Wolfe optimization as an online algorithm with an efficiently computed optimal step size which has a duality gap guarantee. Chang and Yih (2013) suggest a dual coordinate descent applied to L_2 loss which allows updating only one dual variable at a time. Bordes et al. (2008) present the LaRank algorithm adapted to structured problems. After processing a new training example and adding the corresponding constraint, it goes back and reprocesses the old examples by optimizing them with the possibility of further increasing their working set of constraints in order to get training through the data in one pass. For large-scale problems there exist more suitable versions of online algorithms which simply sequentially perform parameter updates concerning only the most violated structure at a time, such as Perceptron (Collins, 2002) with a fixed step size, the passive-aggressive (PA) algorithm (Crammer et al., 2006) with the optimal step size analytically found in dual space by consid-

ering only one constraint corresponding to the 'best' structure, the primal sub-gradient descent method (Ratliff et al., 2006) with the predefined step size followed by a projection which transfers the parameter back into the feasible region. In Chapter 3 we will consider these algorithms in more detail and modify some of them in order to deal with the optimization over k structures with the highest score.

2.3 Conditional random fields

Conditional random fields (CRFs) (Lafferty et al., 2001) are probabilistic discriminative classifiers which can be applied for labeling and segmenting sequential data. When compared with more traditional sequence labeling tools like hidden Markov models (HMMs), the CRFs offer the advantage by relaxing the strong independence assumptions required by HMMs. Additionally, CRFs avoid the label bias problem (Lafferty et al., 2001) exhibited by the maximum entropy Markov models and other conditional Markov models based on directed graphical models.

The *CRF* parameter estimation is typically performed by some of the gradient methods, such as iterative scaling, conjugate gradient, or limited memory quasi-Newton methods (Gupta, 2006), (Lafferty et al., 2001), (Sha and Pereira, 2003), (Sutton, 2008), (Vishwanathan et al., 2006). All these methods require the computation of the likelihood gradient and the standard method for this computation (Lafferty et al., 2001) is based on the internal computation of CRF marginal probabilities by use of the forward-backward (FB) algorithm.

Conditional random fields (Lafferty et al., 2001) are defined by conditional probability

$$p(\boldsymbol{y}|\boldsymbol{x};\mathbf{w}) = \frac{1}{Z(\boldsymbol{x};\mathbf{w})}\mathrm{e}^{\mathbf{w}^{\mathsf{T}}\mathbf{F}(\boldsymbol{x},\boldsymbol{y})}, \tag{2.17}$$

where $\mathbf{F}(\boldsymbol{x},\boldsymbol{y})$ is a global feature vector measuring the compatibility of $\boldsymbol{x}$ and $\boldsymbol{y}$, $\boldsymbol{w} \in \mathbb{R}^M$ are the parameters of the model. The normalization factor,

$$Z(\boldsymbol{x};\mathbf{w}) = \sum_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x})} \mathrm{e}^{\mathbf{w}^{\mathsf{T}}\mathbf{F}(\boldsymbol{x},\boldsymbol{y})}, \tag{2.18}$$

where the sum is calculated over all $\boldsymbol{y}$ possible structures for the observation $\boldsymbol{x}$, is called the *partition function*.

The goal of the CRFs training is to build up the model (2.17) from the data set $((\boldsymbol{x}^n,\boldsymbol{y}^n))_{n=1}^N$. The standard method is to maximize the log likelihood of (2.17)

$$\mathscr{L}(\mathbf{w}) = \sum_{n=1}^{N} \log p(\boldsymbol{y}^n|\boldsymbol{x}^n;\mathbf{w}) \tag{2.19}$$

over the parameter vector $\mathbf{w}$. Instead of maximizing the log-likelihood, we can maximize a

negative log-posteriori probability with a Gaussian prior over parameters $p(\mathbf{w}) : \mathcal{N}(0, \sigma^2 I)$, which can be by comparison with (2.3) considered as a regularized empirical risk minimization

$$\min_{\mathbf{w}} \frac{\sigma^2}{2} \|\mathbf{w}\|^2 - \mathscr{L}(\mathbf{w}) = \min_{\mathbf{w}} \frac{\sigma^2}{2} \|\mathbf{w}\|^2 + \sum_{n=1}^{N} \Big(\log\Big(Z(\boldsymbol{x}^n; \mathbf{w})\Big) - \mathbf{w}^\mathsf{T} \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n) \Big), \tag{2.20}$$

with the regularization coefficient $\lambda = \sigma^2 / N$ and the *CRF loss function* defined as

$$\ell_n^{\mathrm{CRF}}(\mathbf{w}) = \log\Big(\sum_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x})} \mathrm{e}^{\mathbf{w}^\mathsf{T} \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y})} \Big) - \mathbf{w}^\mathsf{T} \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n). \tag{2.21}$$

The optimum can be found with several of the gradient methods (Gupta, 2006), (Lafferty et al., 2001), (Sha and Pereira, 2003), (Sutton and McCallum, 2006),(Sutton, 2008), (Vishwanathan et al., 2006), which requires the computation of the gradient $\nabla_{\mathbf{w}} \mathscr{L}(\mathbf{w})$. According to (2.17) and (2.19), the gradient can be expressed as

$$\nabla_{\mathbf{w}} \mathscr{L}(\mathbf{w}) = \sum_{n=1}^{N} \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n) - \sum_{n=1}^{N} \frac{\nabla_{\mathbf{w}} Z(\boldsymbol{x}^n; \mathbf{w})}{Z(\boldsymbol{x}^n; \mathbf{w})}. \tag{2.22}$$

Linear-Chain CRFs For linear-chain CRFs we will consider observation sequences $\boldsymbol{x} = (x_1, \ldots, x_T)$ and label sequences $\boldsymbol{y} = (y_1, \ldots, y_T)$. According to (2.5), a global feature vector is the sum over feature functions from all positions t in a sequence, so the conditional probability from (2.17) can be represented as

$$p(\boldsymbol{y} | \boldsymbol{x}; \mathbf{w}) = \frac{1}{Z(\boldsymbol{x}; \mathbf{w})} \prod_{t=1}^{T} \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)}, \tag{2.23}$$

and the normalization factor from (2.18) can also be rewritten considering all positions in a sequence as

$$Z(\boldsymbol{x}; \mathbf{w}) = \sum_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x})} \prod_{t=1}^{T} \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)}. \tag{2.24}$$

The main problem in the evaluation of the log likelihood gradient (2.22) is the computation of the quotient between the partition function gradient and the partition function. The partition function gradient can be represented as

$$\nabla_{\mathbf{w}} Z(\boldsymbol{x}; \mathbf{w}) = [\nabla_{\mathrm{w}_1} Z(\boldsymbol{x}; \mathbf{w}), \ldots, \nabla_{\mathrm{w}_M} Z(\boldsymbol{x}; \mathbf{w})], \tag{2.25}$$

where $\nabla_{\mathrm{w}_m} Z(\boldsymbol{x}; \mathbf{w})$ denotes the m-th partial derivative, and can be obtained from (2.24) after

the use of the Leibniz's product rule:

$$\nabla_{\mathbf{w}} Z(\boldsymbol{x}; \mathbf{w}) = \sum_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x})} \prod_{t=1}^{T} \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)} \sum_{k=1}^{T} \boldsymbol{f}(y_{k-1}, y_k, \boldsymbol{x}, k). \tag{2.26}$$

The standard method for the computation of the partition function and its gradient (Lafferty et al., 2001) is based on the forward-backward algorithm which is reviewed in the following section. The linear-chain CRF uses two types of inference: the first one is the *forward-backward algorithm* (Baum and Petrie, 1966) used during the training process, which is the standard method for the computation of the partition function and its gradient, and the second one is the *Viterbi decoding* (Viterbi, 1967) used during the classification to determine the best sequence of labels $\hat{\boldsymbol{y}}$ for the given observation sequence $\boldsymbol{x}$, i.e.

$$\hat{\boldsymbol{y}} = \operatorname*{arg\,max}_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x})} \mathbf{w}^\mathsf{T} \mathbf{F}(\boldsymbol{x}, \boldsymbol{y}). \tag{2.27}$$

These types of inferences will be reviewed in further text, the Viterbi algorithm in Section 2.4.2 and the forward-backward algorithm in Section 2.4.5.

2.4 Decoding algorithms for sequences

2.4.1 Inference over a semiring

Definition 1. *A* semiring *is a tuple* $\left(\mathbb{K}, \oplus, \otimes, 0, 1\right)$ *where* $\mathbb{K}$ *is a set with operations* $\oplus$ *and* $\otimes$ *such that both* $\oplus$ *and* $\otimes$ *are associative and have identity elements in* $\mathbb{K}$ *(*0 *and* 1 *respectively), 0 is the annihilating element for* $\otimes$*, the operation* $\oplus$ *is commutative, and the operation* $\otimes$ *is distributive over* $\oplus$*.*

Let $\left(\mathbb{K}, \oplus, \otimes, 0, 1\right)$ be a semiring and let $\boldsymbol{y} = (y_0, \ldots, y_T)$ be a sequence of variables of length T, where its elements belong to a set $\mathcal{Y}$. We define *local kernel* functions $u_t : \mathcal{Y}^2 \to \mathbb{K}$ for $t = 1, \ldots, T$, and the *global kernel* function $u : \mathcal{Y}^{T+1} \to \mathbb{K}$, assuming that the following decomposition

$$u(\boldsymbol{y}) = \bigotimes_{t=1}^{T} u_t(y_{t-1}, y_t) \tag{2.28}$$

holds for all $\boldsymbol{y} = (y_0, \ldots, y_T) \in \mathcal{Y}^{T+1}$.

Kschischang et al. (2001) describe a sum-product algorithm over the factor graphs, where the addition and multiplication can be taken from different semirings. We are interested here in this algorithm applied to chains. Using the forward-backward (FB) recursion, they solve two

Table 2.2: Examples of semirings used for different decoding algorithms with appropriate application. The k-$\max$ function returns a sorted list of k maximal elements.

Semiring	$u_t(y_{t-1}, y_t)$	Decoding algorithm	Problem description/Reference
$(\mathbb{R}, +, \cdot, 0, 1)$	$Q_t^e(y_{t-1}, y_t)$	FB algorithm	CRF training (Lafferty et al., 2001)
$([0,1], \max, \cdot, 0, 1)$	$p(y_t \vert y_{t-1}) \cdot p(x_t \vert y_t)$	Viterbi	Probability of the best derivation (Rabiner, 1990)
$(\mathbb{R}^*, \max, +, -\infty, 0)$	$Q_t(y_{t-1}, y_t)$	score-Viterbi	Score of the best derivation (Collins, 2002)
$(\mathbb{R}^*, \max, +, -\infty, 0)$	$\widetilde{Q}_t(y_{t-1}, y_t)$	cost-augmented score-Viterbi	Score of the best aug. derivation (Tsochantaridis et al., 2005)
$(\mathbb{R}^{*k}, \oplus^k, \otimes^k, -\boldsymbol{\infty}, \mathbf{0})$	$\widetilde{Q}_t(y_{t-1}, y_t)$	k-best cost-aug. score-Viterbi	Scores of k-best derivations (Crammer et al., 2005)
$(\mathbb{R}^*, \oplus, +, -\infty, 0)$	$Q_t(y_{t-1}, y_t)$	log-domain FB algorithm	Numerically stable CRF training (Ilić et al., 2012)
$(\mathbb{R} \times \mathbb{R}^M, \ominus, \odot, (0, \mathbf{0}), (1, \mathbf{0}))$	$(Q_t^e(y_{t-1}, y_t), Q_t^{ef}(y_{t-1}, y_t))$	log-domain EMP	CRF training on long sequences (Ilić et al., 2012)

Legend:

Symbol definitions

$Q_t(y_{t-1}, y_t) = \mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)$

$Q_t^e(y_{t-1}, y_t) = \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)}$

$\widetilde{Q}_t(y_{t-1}, y_t) = Q_t(y_{t-1}, y_t) + \nu\, l(y_t, y_t^n)$

$Q_t^{ef}(y_{t-1}, y_t) = Q_t^e(y_{t-1}, y_t) \cdot \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)$

$\mathbb{R}^* = \mathbb{R} \cup \{-\infty\}$

Operation definitions

$a \oplus b = \log(\mathrm{e}^a + \mathrm{e}^b)$

$(z_1, \boldsymbol{h}_1) \ominus (z_2, \boldsymbol{h}_2) = (z_1 + z_2, \boldsymbol{h}_1 + \boldsymbol{h}_2)$

$(z_1, \boldsymbol{h}_1) \odot (z_2, \boldsymbol{h}_2) = (z_1 z_2,\ z_1 \boldsymbol{h}_2 + z_2 \boldsymbol{h}_1)$

$\boldsymbol{a} \oplus^k \boldsymbol{b} = k\text{-}\max\{a_1, \ldots, a_k, b_1, \ldots, b_k\}$

$\boldsymbol{a} \otimes^k \boldsymbol{b} = k\text{-}\max\{a_i + b_j \vert 1 \leq i, j \leq k\}$

problems: the *marginalization problem*, which computes the sum

$$v_t(y_t, y_{t+1}) = \bigoplus_{y_{\{t-1,t\}^c}} u(\boldsymbol{y}) = \bigoplus_{y_{\{t-1,t\}^c}} \bigotimes_{i=1}^{T} u_i(y_{i-1}, y_i), \tag{2.29}$$

and the *normalization problem*, which computes the sum

$$Z = \bigoplus_{\boldsymbol{y}} u(\boldsymbol{y}) = \bigoplus_{\boldsymbol{y}} \bigotimes_{t=1}^{T} u_t(y_{t-1}, y_t). \tag{2.30}$$

The algorithm computes the *forward vector*

$$\alpha_i(y_i) = \bigoplus_{y_{0:i-1}} \bigotimes_{t=1}^{i} u_t(y_{t-1}, y_t), \tag{2.31}$$

by using the following recursion

$$\alpha_i(y_i) = \bigoplus_{y_{i-1}} u_{i-1}(y_{i-1}, y_i) \otimes \alpha_{i-1}(y_{i-1}), \tag{2.32}$$

with the initialization $\alpha_0(y_0) = 1$, and the *backward vector*

$$\beta_i(y_i) = \bigoplus_{y_{i+1:T}} \bigotimes_{t=i+1}^{T} u_t(y_{t-1}, y_t), \tag{2.33}$$

which is recursively computed using

$$\beta_i(y_i) = \bigoplus_{y_{i+1}} u_{i+1}(y_i, y_{i+1}) \otimes \beta_{i+1}(y_{i+1}) \tag{2.34}$$

and initialized to $\beta_T(y_T) = 1$. Once the forward α_{t-1} and backward β_t vectors are computed, we can solve the marginalization problem by use of the formula

$$\bigoplus_{y_{\{t-1,t\}^c}} \bigotimes_{i=1}^{T} u_i(y_{i-1}, y_i) = \alpha_{t-1}(y_{t-1}) \otimes u_t(y_{t-1}, y_t) \otimes \beta_t(y_t) \tag{2.35}$$

The normalization problem can be solved with the forward pass only according to

$$\bigoplus_{\boldsymbol{y}} \bigotimes_{t=1}^{T} u_t(y_{t-1}, y_t) = \bigoplus_{y_T} \alpha_T(y_T). \tag{2.36}$$

The previous recursions with specific semirings are used to solve different problems in structured classification problems. Table 2.2 shows examples of semirings used for different problems. In the following sections we will review some of them.

2.4.2 Viterbi algorithm

For sequence labeling we are looking for the cost function to be decomposable through positions in a sequence, i.e.

$$L(\boldsymbol{y}^n, \boldsymbol{y}) = \sum_{t=1}^{T} l(y_t^n, y_t), \tag{2.37}$$

where $l(y_t^n, y_t)$ represents a cost for labeling the observation x_t with y_t instead of y_t^n. The Viterbi algorithm (Viterbi, 1967) is used to find the optimal sequence $\hat{\boldsymbol{y}}$ from (2.27) and it can also be modified to find optimal sequences for a cost augmented problem. For the nth example $(\boldsymbol{x}^n, \boldsymbol{y})_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)}$, the transition score from a label c to a label c' on the observation x_t^n is defined as following

$$Q_{\boldsymbol{x}^n, t}(c, c') = \mathbf{w}^\mathsf{T} \boldsymbol{f}(c, c', \boldsymbol{x}^n, t). \tag{2.38}$$

The Viterbi score $V_t(c)$ for a label c at position t denotes the highest sum of transition scores over all paths starting at the beginning of the sequence and ending at the label c on position t.

Using the dynamic programming, the Viterbi scores can be recurrently computed as

$$V_t(c) = \max_{c' \in \mathcal{Y}} V_{t-1}(c') + Q_{\boldsymbol{x}^n,t}(c', c), \tag{2.39}$$

with initializing V_0 of a special label to zero. The previous formula can be easily modified for the cost augmented problem by simply adding the cost corresponding to the current label position on the decoding path

$$V_t(c) = \max_{c' \in \mathcal{Y}} V_{t-1}(c') + Q_{\boldsymbol{x}^n,t}(c', c) + \nu\, l(y_t^n, c), \tag{2.40}$$

with the same initialization as previously, where y_t^n denotes the original label at the tth position. The most commonly used cost function is the Hamming distance, where $l(y_t^n, y_t) = \mathbb{1}[y_t^n \neq y_t]$. The parameter ν is equal to one for the cost augmented problem (3.1), it is equal to minus one for the optimal sequence (5.4) used for the ramp loss, but it can also take non-constant values for the problems of direct loss minimization described in (McAllester et al., 2010). Finding all Viterbi scores during the forward pass through the sequence is followed by the path reconstruction during the backward pass. The time complexity of the Viterbi algorithm for a sequence of length T is $\mathcal{O}(|\mathcal{Y}|^2 T)$ and the memory complexity is $\mathcal{O}(|\mathcal{Y}|T)$.

2.4.3 k-best Viterbi algorithm

The Viterbi algorithm can be straightforwardly extended to the k-best variant by keeping at every position its k-best partial scores

$$\left(V_t^{(1)}(c), \ldots, V_t^{(k)}(c)\right) = k\text{-}\max_{\substack{c' \in \mathcal{Y} \\ i=1,\ldots,k}} V_{t-1}^{(i)}(c') + Q_{\boldsymbol{x}^n,t}(c', c) + \nu\, l(y_t^n, c). \tag{2.41}$$

Storing the k-best partial scores at each position can be done using the matrix, leading to the time complexity $\mathcal{O}(|\mathcal{Y}|^2 Tk)$ and the memory complexity $\mathcal{O}(|\mathcal{Y}|Tk)$. Brown and Golod (2010) present a method for storing scores using a tree structure, and then prune the nodes that are not involved in the k-best path in order to save memory, which can lead to significantly less memory in practice than with matrix implementation. By defining a local kernel as a vector of the following k identical elements

$$\boldsymbol{u}_t(c', c) = \left(Q_{\boldsymbol{x}^n,t}(c', c) + \nu\, l(y_t^n, c), \ldots, Q_{\boldsymbol{x}^n,t}(c', c) + \nu\, l(y_t^n, c)\right),$$

the k-best Viterbi recursion (2.41) can also be seen as an instance of the sum-product algorithm (Kschischang et al., 2001)

$$\boldsymbol{V}_t = \bigoplus_{c' \in \mathcal{Y}}{}^{k} \boldsymbol{V}_t(c') \otimes^k \boldsymbol{u}_t(c', c) \tag{2.42}$$

over the k-best Viterbi semiring (Goodman, 1999; Mohri, 2002), where $\oplus^k$ and $\otimes^k$ are defined in the legend of Table 2.2.

2.4.4 A* inference

The A* search can be used to generate k-best paths on trellis (Nagata, 1994; Soong and Huang, 1991). This algorithm can also be adopted to generate k-best paths with the involved cost function. It contains two steps: a *forward pass*, which is done by the Viterbi algorithm that finds the exact cost-augmented partial score to each node from the start position, and a *backward pass*, the A* search which uses Viterbi scores for the heuristic function to produce the k-best paths. After the forward pass defined by (2.40), the recursive formula at the position t for the backward pass is

$$(A_t(1), \ldots, A_t(k)) = k\text{-}\max_{\substack{c \in \mathcal{Y} \\ j=1,\ldots,k}} g_{\boldsymbol{x}^n,t}(c,j)$$

$$(B_t(1), \ldots, B_t(k)) = k\text{-}\arg\max_{\substack{c \in \mathcal{Y} \\ j=1,\ldots,k}} g_{\boldsymbol{x}^n,t}(c,j)$$

$$g_{\boldsymbol{x}^n,t}(c,j) = V_t(c) + Q_{\boldsymbol{x}^n,t+1}(c, B_{t+1}(j)) + A_{t+1}(j) + \nu\, l(y^n_{t+1}, B_{t+1}(j)).$$

Similarly as in the Viterbi algorithm, during the backward pass we must store backpointers, thus the path reconstruction is done in the last forward pass. The total time complexity is $\mathcal{O}(|\mathcal{Y}|^2 T + |\mathcal{Y}| T k \log k)$ and the total memory complexity is $\mathcal{O}(|\mathcal{Y}| T + kT)$ providing a better choice than k-best Viterbi algorithm.

One disadvantage of using A* decoding is the need to calculate the transition scores (2.38) at every position twice (once for the forward and the backward pass), which depending on the feature vector storing can be time consuming. This would not be a problem if we did not have long observation sequences, i.e. if memory is not a problem, in which case we can store all transition score matrices for the backward pass.

2.4.5 Forward-backward algorithm

The forward-backward (FB) algorithm first appeared in two independent publications (Baum and Petrie, 1966), (Chang and Hancock, 1966), but it is better known from subsequent papers (Bahl et al., 1974), (Baum, 1972). It makes use of dynamic programming, running with the asymptotical time complexity $\mathcal{O}(|\mathcal{Y}|^2 T)$ and with the memory complexity $\mathcal{O}(|\mathcal{Y}| T)$, where T denotes the sequence length and $|\mathcal{Y}|$ denotes the number of states classified. In spite of the time efficiency, it becomes spatially demanding when the sequence length is exceptionally large (Khreich et al., 2010) which appears in computer security (Lane, 2000), (Warrender et al., 1999), bioinformatics (Krogh et al., 1994), (Meyer and Durbin, 2004) and robot navigation

systems (Koenig and Simmons, 1996). In that case, the (log-domain) expectation semiring (Ilić et al., 2012) can be used to calculate the gradient of a linear-chain CRF only in the forward pass. Since a backward pass is not needed, this can be realized in a fixed memory space $\mathscr{O}(|\mathcal{Y}|^2 + |\mathcal{Y}|M)$ with the size independent of the sequence length T as forward vectors do not need to be stored for the backward pass, where M denotes the dimension of feature vectors. Otherwise, if memory is not a problem, the FB algorithm usually operates over the sum-product semiring, where for practical implementation the sum-product semiring should be replaced with its log-domain semiring (Ilić et al., 2012) for better numerical stability, which will be reviewed further in more detail.

Sum-product semiring forward-backward algorithm

The *sum-product semiring* is the tuple $\Big(\mathbb{R}, +, \cdot, 0, 1\Big)$, where $\mathbb{R}$ is the set of real numbers and the operations are defined in a standard way. The partition function (2.24) can be obtained as a solution of the normalization problem (2.36) using the factorization with local kernels

$$u_t(y_{t-1}, y_t) = \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)}, \tag{2.43}$$

and the gradient can be computed using the solution for the marginalization problem (2.35) in the sum-product semiring. First, we change the sum ordering in (2.26) and split the sum over $\boldsymbol{y}$ to $y_{\{k-1,k\}}$ and $y_{\{k-1,k\}^c}$ sums, transforming (2.26) to:

$$\nabla_{\mathbf{w}} Z(\boldsymbol{x}; \mathbf{w}) = \sum_{k=1}^{T} \sum_{y_{\{k-1,k\}}} \Bigg(\sum_{y_{\{k-1,k\}^c}} \prod_{t=1}^{T} \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)} \Bigg) \cdot \boldsymbol{f}(y_{k-1}, y_k, \boldsymbol{x}, k). \tag{2.44}$$

The marginal values, as the marginalization problem over the sum-product semiring

$$v_t(y_t, y_{t+1}) = \sum_{y_{\{k-1,k\}^c}} \prod_{t=1}^{T} \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)}, \tag{2.45}$$

can be found by recursive computation of forward vectors,

$$\alpha_i(y_i) = \sum_{y_{0:i-1}} \prod_{t=1}^{i} \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)}, \tag{2.46}$$

and backward vectors

$$\beta_i(y_i) = \sum_{y_{i+1:T}} \prod_{t=i+1}^{T} \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)}. \tag{2.47}$$

The FB algorithm over the sum-product semiring suffers from numerical instability since the exponential terms can fall out of the machine precision scope and it is usually replaced with a more stable FB algorithm over the log-domain sum-product semiring.

Log-domain sum-product semiring forward-backward algorithm

The log-domain sum-product semiring is the tuple $\Big(\mathbb{R}^*, \oplus, \otimes, -\infty, 0\Big)$, where $\mathbb{R}^*$ is the extended set of real numbers with minus infinity and the operations are defined by

$$a \oplus b = \log(\mathrm{e}^a + \mathrm{e}^b) \tag{2.48}$$

$$a \otimes b = a + b, \tag{2.49}$$

for all $a, b \in \mathbb{R}^*$. In log-domain, the local kernels have the form:

$$u_t(y_{t-1}, y_t) = \mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t), \tag{2.50}$$

for $t = 1, \ldots, T$. According to the following equalities

$$\log\Big(\sum_{i=1}^{T} a_i\Big) = \bigoplus_{i=1}^{T} \log a_i, \quad \log\Big(\prod_{i=1}^{T} a_i\Big) = \bigotimes_{i=1}^{T} \log a_i, \tag{2.51}$$

which hold straightforwardly from the definition of the log-domain sum-product semiring for every $a_i \in \mathbb{R}^*$, $1 \leq i \leq T$ and expression (2.31), the forward vector in the log-domain sum-product semiring is the logarithm of the forward vector in the sum-product semiring

$$\begin{aligned} \alpha_i(y_i) &= \bigoplus_{y_{0:i-1}} \bigotimes_{t=1}^{i} u_t(y_{t-1}, y_t) = \bigoplus_{y_{0:i-1}} \bigotimes_{t=1}^{i} \mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t) \\ &= \log\Big(\sum_{y_{0:i-1}} \prod_{t=1}^{i} \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)}\Big). \end{aligned} \tag{2.52}$$

The first forward vector is initialized to 0, which is the identity for $\otimes$, and it is recursively computed using

$$\alpha_i(y_i) = \bigoplus_{y_{i-1}} \Big(u_i(y_{i-1}, y_i) + \alpha_{i-1}(y_{i-1})\Big), \quad \alpha_0(y_0) = 0. \tag{2.53}$$

Similarly, the backward vector in the log-domain sum-product semiring is the logarithm of the backward vector in the sum-product semiring

$$\begin{aligned} \beta_i(y_i) &= \bigoplus_{y_{i+1:T}} \bigotimes_{t=i+1}^{T} u_t(y_{t-1}, y_t) = \bigoplus_{y_{i+1:T}} \bigotimes_{t=i+1}^{T} \mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t) \\ &= \log\Big(\sum_{y_{i+1:T}} \prod_{t=i+1}^{T} \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)}\Big), \end{aligned} \tag{2.54}$$

Algorithm 1: Log-domain FB algorithm

input : $\boldsymbol{x},\ \mathbf{w},\ \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t) \in \mathbb{R}^M;\ y_{t-1}, y_t \in \mathcal{Y},\ t = 1, \ldots, T;$
output: $\nabla_{\mathbf{w}} Z(\boldsymbol{x}; \mathbf{w}) / Z(\boldsymbol{x}; \mathbf{w});$

`/* Matrices initialization */`

for $t \leftarrow 1$ **to** T **do**
  foreach y_{t-1} *in* $\mathscr{Y}$ **do**
    foreach y_t *in* $\mathscr{Y}$ **do**
      $u_t(y_{t-1}, y_t) = \mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)$
    end
  end
end

`/* Forward phase */`

foreach y_0 *in* $\mathcal{Y}$ **do**
  $\alpha_0(y_0) \leftarrow 0;$
end
for $t \leftarrow 1$ **to** T **do**
  foreach y_t *in* $\mathcal{Y}$ **do**
    $\alpha_t(y_t) \leftarrow \bigoplus_{y_{t-1}} \left(u_t(y_{t-1}, y_t) + \alpha_{t-1}(y_{t-1}) \right)$
  end
end

`/* Backward phase */`

foreach y_T *in* $\mathcal{Y}$ **do**
  $\beta_T(y_T) \leftarrow 0;$
end
for $t \leftarrow T - 1$ **to** 0 **do**
  foreach y_t *in* $\mathcal{Y}$ **do**
    $\beta_t(y_t) \leftarrow \bigoplus_{y_{t+1}} \left(u_{t+1}(y_t, y_{t+1}) + \beta_{t+1}(y_{t+1}) \right);$
  end
end

`/* Termination */`

$\log Z(\boldsymbol{x}; \mathbf{w}) = \bigoplus_{y_T} \alpha_T(y_T)$
for $t \leftarrow 1$ **to** T **do**
  foreach y_{t-1} *in* $\mathcal{Y}$ **do**
    foreach y_t *in* $\mathcal{Y}$ **do**
      $v = \alpha_{t-1}(y_{t-1}) + u_t(y_{t-1}, y_t) + \beta_t(y_t)$
      for $m \leftarrow 1$ **to** M **do**
        $lnf \leftarrow \log f_m(y_{t-1}, y_t, \boldsymbol{x}, t);$
        $\log \nabla_m Z \leftarrow \log \nabla_m Z \oplus (v + lnf)$
      end
    end
  end
end
for $m \leftarrow 1$ **to** M **do**
  $\nabla_{\mathrm{w}_m} Z(\boldsymbol{x}; \mathbf{w}) / Z(\boldsymbol{x}; \mathbf{w}) \leftarrow \mathrm{e}^{\log \nabla_m Z - \log Z}$
end

which is recursively computed using

$$\beta_i(y_i) = \bigoplus_{y_{i+1}} \Big(u_{i+1}(y_i, y_{i+1}) + \beta_{i+1}(y_{i+1}) \Big), \quad \beta_T(y_T) = 0. \tag{2.55}$$

If the log-domain addition is performed using the definition $a \oplus b = \log(\mathrm{e}^a + \mathrm{e}^b)$, the numerical precision is being lost when computing e^a and e^b. But, as noted in (Sutton, 2008), $\oplus$ can be computed as

$$a \oplus b = a + \log\Big(1 + \mathrm{e}^{(b-a)}\Big) = b + \log\Big(1 + \mathrm{e}^{(a-b)}\Big), \tag{2.56}$$

which can be much more numerically stable, particularly if we pick a version of an identity with a smaller exponent.

The logarithm of the normalization function (2.24) according to (2.51) is

$$\log Z(\boldsymbol{x}; \mathbf{w}) = \log \sum_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x})} \prod_{t=1}^{T} \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)} = \bigoplus_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x})} \bigotimes_{t=1}^{T} u_t(y_{t-1}, y_t), \tag{2.57}$$

and it can be computed using the solution of the normalization problem in the log-domain sum-product semiring with the forward algorithm according to (2.36)

$$\log Z(\boldsymbol{x}; \mathbf{w}) = \bigoplus_{y_T} \alpha_T(y_T). \tag{2.58}$$

The marginal values (2.45) according to (2.51) in the log-domain sum-product semiring have the form

$$v_k(y_{k-1}, y_k) = \log \sum_{y_{\{k-1,k\}^c}} \prod_{t=1}^{T} \mathrm{e}^{\mathbf{w}^\mathsf{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t)} = \bigoplus_{y_{\{k-1,k\}^c}} \bigotimes_{t=1}^{T} u_t(y_{t-1}, y_t). \tag{2.59}$$

They can be efficiently computed according to the solution of the marginalization problems (2.35):

$$v_k(y_{k-1}, y_k) = \alpha_{k-1}(y_{k-1}) \otimes u_k(y_{k-1}, y_k) \otimes \beta_k(y_k), \tag{2.60}$$

where $\alpha_{k-1}(y_{k-1})$ and $\beta_k(y_k)$ are computed with the FB algorithm over the log-domain sum-product semiring using equations (2.53) and (2.55). Then, by taking the logarithm of the m-th component in gradient expression (2.44), we get

$$\log \nabla_{\mathrm{w}_m} Z(\boldsymbol{x}; \mathbf{w}) = \bigoplus_{k=1}^{T} \bigoplus_{y_{\{k-1,k\}}} v_k(y_{k-1}, y_k) \otimes \log f_m(y_{k-1}, y_k, \boldsymbol{x}, k). \tag{2.61}$$

Finally, the quotient between the partition function and its gradient can be computed according to

$$\frac{\nabla_{\mathbf{w}} Z(\boldsymbol{x}; \mathbf{w})}{Z(\boldsymbol{x}; \mathbf{w})} = \mathrm{e}^{\log \nabla_{\mathbf{w}} Z(\boldsymbol{x}; \mathbf{w}) - \log Z(\boldsymbol{x}; \mathbf{w})}. \tag{2.62}$$

The calculation of the previous quotient using the FB algorithm over the log-domain sum-product semiring is presented in Algorithm 1.

Chapter 3

Structured classification with k-best loss

Structured learning algorithms usually require an inference during the training procedure. Due to their exponential size of output space, the parameter update is performed only on a relatively small collection built from the "best" structures. The k-best MIRA is an example of an online algorithm which seeks optimal parameters by making updates on k structures with the highest score at a time. Following the idea of using k-best structures during the learning process, in this chapter we introduce four new k-best extensions of max-margin structured algorithms. We discuss their properties and connection, and evaluate all algorithms on two sequence labeling problems, the shallow parsing and named entity recognition. The experiments show how the proposed algorithms are affected by the changes of k in terms of the F-measure and computational time, and that the proposed algorithms can improve the results in comparison to those of the single best case. Moreover, the restriction to the single best case produces a comparison of the existing algorithms.

3.1 Introduction to k-best learning

In section 2.2 we discuss different techniques of dealing with exponentially many constraints. Most of them are based on the creation of a working set of constraints. While all these approaches make a single increment to their working set of constraints or just restrict the optimization to only one constraint, Crammer et al. (2005) first proposed the k-best version of the Margin Infused Relaxed Algorithm (MIRA) as an online algorithm which restricts optimization not only to one but to k-best constraints. The algorithm is designed to traverse through training examples, find the k-best structures inside the example with current parameters and then use these structures to update the parameters before moving to the next example. The idea is to adjust the parameters using new structures, but keep the changes as small as possible in order to hold previously obtained knowledge. At each example, after finding k outputs with the highest score, the algorithm minimizes the norm of parameter change while satisfying constraints on generated outputs. Even though the algorithm traverses online through examples, inside each example it performs full optimization subject to generated k constraints. The algorithm was tested on different problems. Crammer et al. (2005) present the results on handwriting recog-

nition, noun phrase chunking and named entity recognition (NER), showing that k-best MIRA performs as well as batch methods such as CRFs and M^{3}N, but converges more quickly after a few iterations. McDonald et al. (2005) use k-best MIRA for dependency parsing, testing different values of k, and state that even small values of k are sufficient to achieve close to best performance. Gimpel and Cohen (2007) also test k-best MIRA on NER and achieve results similar to the perceptron algorithm that were slightly better than the CRF, while MIRA gave better results on the part-of-speech (POS) tagging than perceptron for a reasonably small parameter k. Other related work includes using a primal sub-gradient method for the averaged sum loss (Mančev and Todorović, 2014) optimization, which is closely related to the k-best variant of the Pegasos algorithm (Shalev-Shwartz et al., 2011). The k-best approach is also applied in confidence weighted methods (Mejer and Crammer, 2010) and used as a confidence in the committee organization (Mančev and Todorović, 2012), where k-best paths are used to extract the confidence of a particular label which is included in the learning process to improve the results.

The general idea of the k-best approach is to restrict the learning process from the structured space to its subset constructed only of its k elements with the highest scores, since considering all elements from the original space would lead to an intractable learning process in real time. While the influence of the parameter k to the learning process is not yet explored in theory, empirically it is shown that small values of k contribute to the learning process, with slight degradation of performance for larger values. For k equal to one, different learning methods have been developed, while for higher values there is k-best Margin Infused Relaxed Algorithm (MIRA). Following the advantages of the k-best MIRA presented in (Crammer et al., 2005), (McDonald et al., 2005), (Gimpel and Cohen, 2007), our motivation is to investigate different possibilities of incorporating k-best structures into the learning process with the aim to get better results than the corresponding single best version, and to provide the theoretical analysis of the introduced algorithms. Thus, we introduce four new k-best versions (Mančev and Todorović, 2015) to train the structured SVM as extensions of the popular single best algorithms: the LaRank, SDM, passive-aggressive and perceptron algorithm.

3.2 k-best extensions

In this section we will describe extensions to the k-best case of popular algorithms. We start with reviewing the k-best MIRA from (Crammer et al., 2005) and then introduce the k-best version of the perceptron, LaRank algorithm, sequential dual method, and passive-aggressive algorithm. We also introduce the restricted version of the passive-aggressive algorithm for which the cumulative prediction loss is bound in a similar way as in the single best version.

3.2.1 k-best MIRA

Crammer and Singer (2003) propose the Margin Infused Relaxed Algorithm (MIRA) originally for multiclass problems, which was later extended to structured classification (Crammer et al., 2005). MIRA is defined in an online manner. After receiving a new example $(\boldsymbol{x}^n, \boldsymbol{y}^n)$, starting from parameters $\mathbf{w}_n$, the algorithm is searching for new parameters $\mathbf{w}$ in order to keep the norm of changing parameters as small as possible during the satisfaction of particular constraints on the received example. In a special case, the constraints on the example can be restricted to only one constraint corresponding to the predicted structure $\widetilde{\boldsymbol{y}}^n$ with the involved cost function. This is known as 1-best variant of MIRA, also called the online passive-aggressive (PA) algorithm (Crammer et al., 2006). The best sequence for $\boldsymbol{x}^n$ is found with respect to the structured hinge loss function as

$$\widetilde{\boldsymbol{y}}^n = \operatorname*{arg\,max}_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) = \operatorname*{arg\,max}_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \mathbf{w}^\mathsf{T} \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}) + L(\boldsymbol{y}^n, \boldsymbol{y}), \tag{3.1}$$

where the loss for an individual structure is defined with

$$\ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) = \max\left(0, L(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y})\right). \tag{3.2}$$

MIRA was primarily designed for multiclass classification. In that case it belongs to the family of ultraconservative online algorithms (Crammer and Singer, 2003), which consider only the updates of parameters corresponding to violated examples, i.e. to those examples that produce a higher score than the original one. Working with the ultraconservative property in the case of the online structured classification leads to problems in optimization, due to the possibility of a large number of violated constraints. This produces the need to consider the optimization restricted only to the constraints for the first k sequences with the highest score, leading to the following optimization problem

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_n\|^2 \tag{3.3}$$

$$\text{s.t. } \mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y}) \geq L(\boldsymbol{y}^n, \boldsymbol{y}),\ \forall \boldsymbol{y} \in \mathscr{B}^{k,n}_{\mathbf{w}_n}, \tag{3.4}$$

where $\mathscr{B}^{k,n}_{\mathbf{w}_n}$ represents the *k-best set*, formally defined as the set $\mathscr{B}^{k,n}_{\mathbf{w}_n} \subset \mathscr{Y}(\boldsymbol{x}^n)$ that satisfies $\forall \boldsymbol{y}' \in \mathscr{B}^{k,n}_{\mathbf{w}_n}, \forall \boldsymbol{y}'' \notin \mathscr{B}^{k,n}_{\mathbf{w}_n} \Longrightarrow \ell(\mathbf{w}_n; (\boldsymbol{x}^n, \boldsymbol{y}')) \geq \ell(\mathbf{w}_n; (\boldsymbol{x}^n, \boldsymbol{y}''))$ and $|\mathscr{B}^{k,n}_{\mathbf{w}_n}| = \min\{k, |\mathscr{Y}(\boldsymbol{x}^n)|\}$. Note that with this formulation the set is not uniquely defined. Crammer et al. (2005) define k-best MIRA for a linear separable problem, but we can consider its extension to the linear non-separable case. As it is an online algorithm, we can consider a general formulation where we perform optimization on the loss function for the nth example, which for the k-best case can be defined as an averaged sum of losses of k-best structures, i.e.

$$\min_{\mathbf{w}} \frac{\lambda}{2} \|\mathbf{w} - \mathbf{w}_n\|^2 + \ell_n^{k\text{best}}(\mathbf{w}), \qquad \text{where} \quad \ell_n^{k\text{best}}(\mathbf{w}) = \frac{1}{k} \sum_{\boldsymbol{y} \in \mathscr{B}^{k,n}_{\mathbf{w}}} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})). \tag{3.5}$$

Algorithm 2: k-best MIRA

Input : Training data: $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$, parameter $k \in \mathbb{N}$
Number of iterations: P
Output: Model parameters: $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{0}$;
for $p \leftarrow 1$ **to** P **do**
  foreach $(\boldsymbol{x}^n, \boldsymbol{y}^n)$ ***in*** $\mathscr{D}$ **do**
    $\mathscr{B}_{\mathbf{w}}^{k,n} \leftarrow k\text{-}\arg\max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$; `/* find k-best structures */`
    $\mathscr{W}_n \leftarrow \mathscr{B}_{\mathbf{w}}^{k,n}$;
    Find $\mathbf{v}$ as a solution of the optimization problem:
      $\min_{\mathbf{v}} \frac{1}{2} \|\mathbf{v} - \mathbf{w}\|^2$
      s.t. $\mathbf{v}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y}) \geq L(\boldsymbol{y}^n, \boldsymbol{y}), \quad \forall \boldsymbol{y} \in \mathscr{W}_n$
    $\mathbf{w} \leftarrow \mathbf{v}$;
  end
end

We can convert the previous formulation into a constraint optimization problem by introducing a slack variable $\xi_{n,\boldsymbol{y}}$ for each structure $\boldsymbol{y}$ belonging to the k-best set as

$$\min_{\mathbf{w}, \boldsymbol{\xi}} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_n\|^2 + C \sum_{\boldsymbol{y} \in \mathscr{B}_{\mathbf{w}}^{k,n}} \xi_{n,\boldsymbol{y}} \tag{3.6}$$

$$\text{s.t. } \mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y}) \geq L(\boldsymbol{y}^n, \boldsymbol{y}) - \xi_{n,\boldsymbol{y}}, \quad \xi_{n,\boldsymbol{y}} \geq 0, \qquad \forall \boldsymbol{y} \in \mathscr{B}_{\mathbf{w}_n}^{k,n}, \tag{3.7}$$

where $C = 1/(\lambda k)$. The k-best MIRA is a general method which incorporates the k-best search in an online manner through examples without any specification for solving the optimization problem (3.3)-(3.4) or (3.6)-(3.7) inside the example. The pseudocode for the linear separable version is presented in Algorithm 2.

3.2.2 k-best sequential dual method

In order to deal with a large dimensional problem (2.9)-(2.10), the sequential dual method (SDM) for structured SVMs (Balamurugan et al., 2011) makes an assumption that at optimum only a small number of constraints inside one example is active. The scaled version of the following sub-problem is considered

$$\min_{\boldsymbol{\alpha}_n} \frac{1}{2} \boldsymbol{\alpha}_n^\mathsf{T} \mathbf{K}_n \boldsymbol{\alpha}_n - \boldsymbol{\alpha}_n^\mathsf{T} (\mathbf{L}_n - \Delta\mathbf{F}^\mathsf{T} \mathbf{w}) \tag{3.8}$$

$$\text{s.t.} \sum_{\boldsymbol{y} \in \mathscr{W}_n} \alpha_{n,\boldsymbol{y}} = 0, \quad \alpha_{n,\boldsymbol{y}} \geq -\lambda_{n,\boldsymbol{y}}, \quad \forall \boldsymbol{y} \in \mathscr{W}_n, \tag{3.9}$$

where the set of examples $\mathscr{W}_n = \{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n) \,:\, \lambda_{n,\boldsymbol{y}} > 0\}$ is incrementally built by adding the sequence $\widetilde{\boldsymbol{y}}^n$ according to (3.1) that violates the margin the most on the current nth example and $\boldsymbol{K}_n$ denotes a block of matrix $\boldsymbol{K}$ which corresponds to the nth example. Specific heuristics are employed to control the growth of the set $\mathscr{W}_n$ in order to keep the number of optimization constraints at a reasonable size. After reaching the desired precision inside one example, the procedure continues to the next example to optimize, where the optimization is performed by the sequential minimal optimization (SMO) (Platt, 1999).

Now we will describe the SDM extension to the k-best case. The algorithm traverses through training examples and, at the nth example, its working set $\mathscr{W}_n$ is extended with current k-best sequences including the original sequence $\boldsymbol{y}^n$. Then the SMO is applied to optimize dual variables associated with sequences in $\mathscr{W}_n$. It selects a pair of sequences from $\mathscr{W}_n$ based on the 'maximum violating pair' strategy and performs the optimization until the KKT conditions become satisfied on $\mathscr{W}_n$. The test of the KKT condition violation, up to precision τ, is

$$g_{n,\boldsymbol{y}''} > g_{n,\boldsymbol{y}'} + \tau, \tag{3.10}$$

$$\boldsymbol{y}' = \arg\min_{\boldsymbol{y} \in \mathscr{W}_n} g_{n,\boldsymbol{y}}, \quad \boldsymbol{y}'' = \arg\max_{\boldsymbol{y} \in I_0'} g_{n,\boldsymbol{y}}, \tag{3.11}$$

where $I_0' = \{\boldsymbol{y} \in \mathscr{W}_n : \alpha_{n,\boldsymbol{y}} > -\lambda_{n,\boldsymbol{y}}\}$. The gradient of the dual function is expressed by (2.15), where the sum over all elements of $\mathscr{Y}(\boldsymbol{x}^n)$ can be reduced for the sub-problem (3.8)-(3.9) to the sum over the elements from $\mathscr{W}_n$ as

$$g_{n,\boldsymbol{y}} = \sum_{\boldsymbol{z} \in \mathscr{W}_n} \alpha_{n,\boldsymbol{z}} K_{n,\boldsymbol{y},\boldsymbol{z}} - L(\boldsymbol{y}^n, \boldsymbol{y}) + \mathbf{w}^{\mathsf{T}} \Delta\mathbf{F}_n(\boldsymbol{y}), \tag{3.12}$$

since $\alpha_{n,\boldsymbol{y}}$ is equal to zero if $\boldsymbol{y}$ does not belong to $\mathscr{W}_n$. After selecting sequences $\boldsymbol{y}'$ and $\boldsymbol{y}''$ which violate KKT conditions the most using (3.11), we consider the updates of the corresponding alpha parameters as $\alpha_{n,\boldsymbol{y}'} \leftarrow \alpha_{n,\boldsymbol{y}'} + \delta$ and $\alpha_{n,\boldsymbol{y}''} \leftarrow \alpha_{n,\boldsymbol{y}''} - \delta$. The optimization, now restricted to a single variable δ which must change the parameters in such way that they remain feasible, can be expressed in the analytical form as

$$\delta = \max\left(-\lambda_{n,\boldsymbol{y}'} - \alpha_{n,\boldsymbol{y}'}, \min\left(\lambda_{n,\boldsymbol{y}''} + \alpha_{n,\boldsymbol{y}''}, \frac{g_{n,\boldsymbol{y}''} - g_{n,\boldsymbol{y}'}}{\|\Delta\mathbf{F}_n(\boldsymbol{y}') - \Delta\mathbf{F}_n(\boldsymbol{y}'')\|^2}\right)\right). \tag{3.13}$$

Note that after satisfying the KKT conditions for the set $\mathscr{W}_n$ and leaving that example, unsatisfied conditions inside the example could still exist, since y' is found by a minimization over $\mathscr{W}_n$ instead of $\mathscr{Y}(\boldsymbol{x}^n)$. As a time consuming operation, the minimization over the entire $\mathscr{Y}(\boldsymbol{x}^n)$ is done only when we extend the current set $\mathscr{W}_n$ by k new sequences. These are the sequences corresponding to the lowest elements of the dual function gradient. Since $\mathscr{W}_n$ is extended when we start processing the nth example, all $\alpha_{n,\boldsymbol{y}}$ will be zero and thus these sequences are those with the highest $\ell(\mathbf{w}; (\boldsymbol{x}^n, y))$, e.i. the sequences from $\mathscr{B}^{k,n}_{\mathbf{w}}$.

Algorithm 3: k-best SDM

Input : Training data: $D = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$, parameter: $C \in \mathbb{R}^+$, $k \in \mathbb{N}$
Number of iterations: P
Output: Model parameters: $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{0}$; $\boldsymbol{\lambda} \leftarrow \mathbf{0}$;
$\mathscr{W}_n \leftarrow \{\boldsymbol{y}^n\}$, $\lambda_{n,\boldsymbol{y}^n} \leftarrow C$, $\forall n = 1, \ldots, N$;
for $p \leftarrow 1$ **to** P **do**
  for $n \leftarrow 1$ **to** N **do**
    kBV-PROCESSNEW(n, $\mathscr{W}_n$, $\mathbf{w}$, $\boldsymbol{\lambda}$);
  end
end

In the end of the section, let us consider the difference between k-best MIRA optimized in dual space with the SMO and k-best SDM. As a consequence of a different primal formulation, when it returns to an earlier example, k-best MIRA will create a new working set $\mathscr{W}_n$ while k-best SDM will update its previous set with new sequences (see Algorithm 3). Thus only the first epoch will be the same. However, if we consider the 1-best version of these algorithms, MIRA will be equivalent to the PA algorithm, while the SDM will have the first epoch through the data equivalent to MIRA and thus to the PA algorithm. These equivalents can also be observed because 1-best SDM in the first pass will choose sequences $\boldsymbol{y}' = \widetilde{\boldsymbol{y}}^n$ and $\boldsymbol{y}'' = \boldsymbol{y}^n$ on the nth example to optimize, and as all $\alpha_{n,\boldsymbol{y}}$ are zero at that moment and as all $\lambda_{n,\boldsymbol{y}}$ are zero, except for $\lambda_{n,\boldsymbol{y}^n} = C$, the step δ in (3.13) will be reduced to the step from (3.14). However, note that this equivalence considers setting τ to zero, but in practical application τ is set to be a small positive tolerance because of numerical errors and the runtime of the algorithm.

3.2.3 k-best LaRank algorithm

The LaRank algorithm (Bordes et al., 2007), introduced for multi-class problems, is a batch algorithm which uses the SMO with specific operations describing how to select a pair of coefficients for optimization. The algorithm has also been used for a structured output with a simpler scheduling for selecting a pair for optimization in an online manner and this version is called the OLaRank algorithm (Bordes et al., 2008). We adopt the following notation: if an example $[(\boldsymbol{x}^n, \boldsymbol{y})]_{\boldsymbol{y} \in \mathscr{Y}(x^n)}$ has all dual variables $\lambda_{n,\boldsymbol{y}} = 0$ equal to zero, we will call it a *new example*, otherwise we will refer to it as an *old example* or *support pattern*. The algorithm with the exact inference is used to solve the dual problem (2.9)-(2.10) by applying three basic operations:

PROCESSNEW Pick a new example $[(\boldsymbol{x}^n, \boldsymbol{y})]$ and choose the pair of coefficients $\lambda_{n,\boldsymbol{y}^n}$ and $\lambda_{n,\widetilde{\boldsymbol{y}}^n}$ to optimize, where $\widetilde{\boldsymbol{y}}^n$ is found by (3.1),

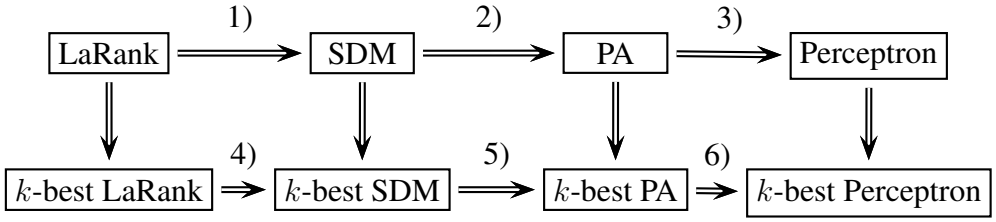

Figure 3.1: Connections between learning algorithms and their k-best extensions for the first epoch.
1) and 4) $n_R = 0$. 2) Described in the end of Section 3.2.2.
3) and 6) No cost ($\mathbf{L} = \mathbf{0}$), fixed step size ($\delta = 1$). The connection holds for more than one epoch.
5) Performs one SMO step on each k-best sequence inside an example in sequential order.

PROCESSOLD Random pick a support pattern $[(\boldsymbol{x}^n, \boldsymbol{y})]$ and choose the best pair of coefficients $\lambda_{n,\boldsymbol{z}}$ and $\lambda_{n,\widetilde{\boldsymbol{y}}^n}$ to optimize according to the gradient vector,

OPTIMIZE Random pick a support pattern and choose the best pair of non-zero coefficients to optimize according to the gradient vector.

OLaRank applies the PROCESSNEW step followed by a specific number of n_R REPROCESS operations, where REPROCESS means applying ten OPTIMIZE steps after one PROCESSOLD step. Note that $n_R = 0$ reduces OLaRank to the previously described SDM for the first epoch.

We introduced the k-best version of the previous algorithm, calling it k-best LaRank, which optimizes the same dual problem with the SMO, which in a special case reduces to the k-best SDM (see Fig. 3.1). k-best LaRank uses the following k-best variants (kBVs) of operations:

kBV-PROCESSNEW Pick a new example $[(\boldsymbol{x}^n, \boldsymbol{y})]$ and optimize coefficients which correspond to its k-best sequences together with the original sequence $\boldsymbol{y}^n$,

kBV-PROCESSOLD Random pick a support pattern and optimize non-zero coefficients together with new coefficients from k-best sequences,

kBV-OPTIMIZE Random pick a support pattern and perform the optimization inside this pattern among non-zero coefficients,

with the same scheduling as in OLaRank. Let the optimization be performed inside a pattern while the KKT conditions are violated more than τ, as in (3.10)-(3.11), by choosing the sequence $\boldsymbol{y}''$ among the $\boldsymbol{y}$s for which $\alpha_{n,\boldsymbol{y}} > -\lambda_{n,\boldsymbol{y}} + \kappa$, where τ and κ are small positive tolerances.

Algorithm 4: k-best LaRank

Input : Training data: $D = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$, parameter: $C \in \mathbb{R}^+$, $k \in \mathbb{N}$
Output: Model parameters: $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{0}$; $\boldsymbol{\lambda} \leftarrow \mathbf{0}$;
$\mathscr{W}_n \leftarrow \{\boldsymbol{y}^n\}$, $\lambda_{n,\boldsymbol{y}^n} \leftarrow C$, $\forall n = 1, \ldots, N$;
for $n \leftarrow 1$ **to** N **do**
  kBV-PROCESSNEW(n, $\mathscr{W}_n$, $\mathbf{w}$, $\boldsymbol{\lambda}$);
  for $k \leftarrow 1$ **to** n_R **do**
    kBV-REPROCESS($(\mathscr{W}_1, \ldots, \mathscr{W}_n)$, $\mathbf{w}$, $\boldsymbol{\lambda}$);
  end
end

Procedure kBV-ProcessNew(n, $\mathscr{W}_n$, $\mathbf{w}$, $\boldsymbol{\lambda}$)

$\mathscr{W}_n \leftarrow \mathscr{W}_n \cup \mathscr{B}_{\mathbf{w}}^{k,n}$; `/* find k-best sequences of the nth example */`
SMO(n, $\mathscr{W}_n$, $\mathbf{w}$, $\boldsymbol{\lambda}$);

Procedure kBV-Reprocess($(\mathscr{W}_1, \ldots, \mathscr{W}_n)$, $\mathbf{w}$, $\boldsymbol{\lambda}$)

kBV-PROCESSOLD($(\mathscr{W}_1, \ldots, \mathscr{W}_n)$, $\mathbf{w}$, $\boldsymbol{\lambda}$);
for $i \leftarrow 1$ **to** $maxIter$ **do**
  kBV-OPTIMIZE($(\mathscr{W}_1, \ldots, \mathscr{W}_n)$, $\mathbf{w}$, $\boldsymbol{\lambda}$);
end

Procedure kBV-ProcessOld($(\mathscr{W}_1, \ldots, \mathscr{W}_n)$, $\mathbf{w}$, $\boldsymbol{\lambda}$)

$m \leftarrow$ Random$[1, n]$;
$\mathscr{W}_m \leftarrow \mathscr{W}_m \cup \mathscr{B}_{\mathbf{w}}^{k,n}$; `/* find k-best sequences of the mth example */`
SMO(m, $\mathscr{W}_m$, $\mathbf{w}$, $\boldsymbol{\lambda}$);

Procedure kBV-Optimize($(\mathscr{W}_1, \ldots, \mathscr{W}_n)$, $\mathbf{w}$, $\boldsymbol{\lambda}$)

$m \leftarrow$ Random$[1, n]$;
SMO(m, $\mathscr{W}_m$, $\mathbf{w}$, $\boldsymbol{\lambda}$);

Bordes et al. (Bordes et al., 2008) express the regret bound of the LaRank PROCESSNEW operation through the data with the same value as the bound for the passive-aggressive algorithm. As neither PROCESSOLD nor OPTIMIZE operation can decrease the dual function, the same bound is stated for the whole LaRank algorithm. Since the previous two operations are not considered, the true regret should be much smaller. Similar results can be stated for the k-best version. As the first SMO step in kBV-PROCESSNEW will increase the dual function by the same value as PROCESSNEW, and as other SMO steps cannot decrease the dual function, as well as kBV-PROCESSOLD and kBV-OPTIMIZE operations, the k-best LaRank holds the same regret bound as the passive-aggressive algorithm. The pseudo code is presented in Algorithm 4.

3.2.4 k-best passive-aggressive algorithms

The passive-aggressive (PA) algorithm is an online algorithm introduced by Crammer et al. (2006) and it solves the optimization problem (3.6)-(3.7) for $k = 1$, i.e. by using only the constraint generated from the best structure. We will refer to this algorithm as 1-best passive-aggressive or 1-best MIRA. Let us define a set of misclassified sequences with given parameters $Err_{\mathbf{w}_n} = \{\boldsymbol{y} \in \mathscr{Y}_{-n} : \ell(\mathbf{w}_n; (\boldsymbol{x}^n, \boldsymbol{y})) > 0\}$. This set will be used to decide weather or not the sequence should be used for parameter change. In order to calculate the step size, Crammer et al. (2006) used the method of Lagrange multipliers on the problem (3.6)-(3.7), which for the single best version has only two constraints: the constraint for the structure with the highest score $\widetilde{\boldsymbol{y}}^n$ and the one for the original structure $\boldsymbol{y}^n$. The optimal step size is then found in the closed form as

$$\delta = \min \left\{ \frac{\ell(\mathbf{w}_n; (\boldsymbol{x}^n, \widetilde{\boldsymbol{y}}^n))}{\left\| \Delta \mathbf{F}_n(\widetilde{\boldsymbol{y}}^n) \right\|^2}, C \right\}. \tag{3.14}$$

The 1-best passive-aggressive algorithm considers updating the parameters only for the best sequence with the following behaviors:

- *passive behavior* if the constraint in (3.7) for the best sequence of the current example is satisfied, the algorithm is passive, making no update;

- *aggressive behavior* if it is not a case, the algorithm makes an aggressive update

  $$\mathbf{w} = \mathbf{w}_n + \delta \Delta \mathbf{F}_n(\widetilde{\boldsymbol{y}}^n)$$

  on the nth example in order to satisfy the single constraint on the best sequence $\widetilde{\boldsymbol{y}}^n \in \mathscr{Y}(\boldsymbol{x}^n)$ found by (3.1).

Involving more than one constraint leads to k-best MIRA, where the optimization is performed over corresponding k-best constraints. We define the sequence of optimization problems inside one example in an online manner, where each of them is subject to only one of k-best constraints. Since only one constraint is considered at a time, each one can be solved analytically. Thus, we will sequentially traverse through k-best sequences, optimize the subproblem restricted to only one of the sequences

$$\min_{\xi, \mathbf{w}_n^{j+1}} \frac{1}{2} \left\| \mathbf{w}_n^{j+1} - \mathbf{w}_n^j \right\|^2 + C\xi \tag{3.15}$$

$$\text{s.t. } {\mathbf{w}_n^{j+1}}^{\mathsf{T}} \Delta \mathbf{F}_n(\boldsymbol{y}^{(n,j)}) \geq L(\boldsymbol{y}^n, \boldsymbol{y}^{(n,j)}) - \xi, \quad \xi \geq 0 \tag{3.16}$$

for $j = 1, \ldots, k$, and perform passive-aggressive updates

$$\mathbf{w}_n^{j+1} = \mathbf{w}_n^j + \delta_{n,j} \Delta \mathbf{F}_n(\boldsymbol{y}^{(n,j)}), \quad j = 1, \ldots, k, \tag{3.17}$$

where each step can be found in the closed form as

$$\delta_{n,j} = \min \left\{ \frac{\ell(\mathbf{w}_n^j; (\boldsymbol{x}^n, \boldsymbol{y}^{(n,j)}))}{\left\| \Delta \mathbf{F}_n(\boldsymbol{y}^{(n,j)}) \right\|^2}, C \right\}. \tag{3.18}$$

With previous two formulas, we define *k-best passive-aggressive updates*, supposing that we start processing the nth example with parameters $\mathbf{w}_n$, which are used to produce k-best sequences $(\boldsymbol{y}^{(n,1)}, \ldots, \boldsymbol{y}^{(n,k)}) = \mathscr{B}^{k,n}_{\mathbf{w}_n}$. The parameters $\mathbf{w}_n^j$ denote the intermediate states of parameters through the iterations between starting parameters $\mathbf{w}_n = \mathbf{w}_n^1$ and parameters after the optimization on the nth example $\mathbf{w}_{n+1} = \mathbf{w}_n^{k+1}$. While k-best MIRA can be seen as a semi-online algorithm (online through examples and batch inside a single example) the k-best passive-aggressive algorithm is defined in a complete online manner through the examples and also inside a single example.

Crammer et al. (2006) provide a cumulative prediction loss bound for the passive-aggressive algorithm for the cost-sensitive multiclass classification, which can also be applied to 1-best PA algorithm for a structured output. Here, we will provide a similar bound for the k-best case with slightly different assumptions, where updates are additionally restricted from k-best sequences. Let's first define the *prediction sequence* $\widehat{\boldsymbol{y}}_{\mathbf{w}_n}$ found by parameters $\mathbf{w}_n$ on the nth example as

$$\widehat{\boldsymbol{y}}_{\mathbf{w}_n} = \operatorname*{arg\,max}_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \mathbf{w}_n^{\mathsf{T}} \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}). \tag{3.19}$$

This sequence is also considered by Crammer et al. (2006) for use in the PA approach, where steps corresponding to $\widehat{\boldsymbol{y}}$ and $\widetilde{\boldsymbol{y}}$ are called the prediction-based and the max-loss step, respectively. We use this sequence to define the auxiliary set

$$A_{\mathbf{w}_n} = \{\boldsymbol{y} \in \mathscr{Y}_{-n} : \ell(\mathbf{w}_n; (\boldsymbol{x}^n, \boldsymbol{y})) \geq \ell(\mathbf{w}_n; (\boldsymbol{x}^n, \widehat{\boldsymbol{y}}_{\mathbf{w}_n}))\}, \tag{3.20}$$

which we need to define the *restricted k-best passive-aggressive* parameter updates as

$$\mathbf{w}_n^{j+1} = \begin{cases} \mathbf{w}_n^j + \delta_{n,j} \Delta \mathbf{F}_n(\boldsymbol{y}^{(n,j)}) & , \text{ if } \boldsymbol{y}^{(n,j)} \in A_{\mathbf{w}_n^j} \\ \mathbf{w}_n^j & , \text{ if } \boldsymbol{y}^{(n,j)} \notin A_{\mathbf{w}_n^j} \end{cases}, j = 1, \ldots, k, \tag{3.21}$$

and use it for the *k-best restricted passive aggressive (RPA) algorithm* (see Algorithm 5). Let all sequences from $\mathscr{B}^{k,n}_{\mathbf{w}_n}$ on which we make a non-zero update according to updates (3.21) be $\mathscr{W}_n = (\boldsymbol{y}^{(n,i_1)}, \ldots, \boldsymbol{y}^{(n,i_{|\mathscr{W}_n|})})$, for some indices $1 \leq i_1 < \ldots < i_{|\mathscr{W}_n|} \leq k$ where the length of vector $\mathscr{W}_n$ is denoted with $|\mathscr{W}_n|$. Further in this section, just for simplicity, we will refer to these sequences as $\mathscr{W}_n = (\boldsymbol{y}^{(n,1)}, \ldots, \boldsymbol{y}^{(n,|\mathscr{W}_n|)})$.

Lemma 1. *Let $(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$ be a sequence of examples and let $\mathbf{u}$ be any parameter*

vector. For the restricted k-best PA update defined by (3.21), it follows

$$\sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|}\delta_{n,j}\left(2\ell(\mathbf{w}_n^j;(\boldsymbol{x}^n,\boldsymbol{y}^{(n,j)}))-\delta_{n,j}\left\|\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right\|^2-2\ell(u;(\boldsymbol{x}^n,\boldsymbol{y}^{(n,j)}))\right)\le\|\mathbf{u}\|^2,$$

where $\mathscr{W}_n=(\boldsymbol{y}^{(n,1)},\ldots,\boldsymbol{y}^{(n,|\mathscr{W}_n|)})$ contains all sequences from $\mathscr{B}^{k,n}_{\mathbf{w}_n^1}$ on which we make a non-zero update according to (3.21).

Proof. Defining γ_n^j as $\left\|\mathbf{w}_n^j-\mathbf{u}\right\|^2-\left\|\mathbf{w}_n^{j+1}-\mathbf{u}\right\|^2$ and summing it over all n and j we get

$$\begin{aligned}\sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|}\gamma_n^j&=\sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|}\left(\left\|\mathbf{w}_n^j-\mathbf{u}\right\|^2-\left\|\mathbf{w}_n^{j+1}-\mathbf{u}\right\|^2\right)\\&=\left\|\mathbf{w}_1^1-\mathbf{u}\right\|^2-\left\|\mathbf{w}_N^{|\mathscr{W}_N|+1}-\mathbf{u}\right\|^2\le\left\|\mathbf{w}_1^1-\mathbf{u}\right\|^2=\|\mathbf{u}\|^2,\end{aligned}\tag{3.22}$$

because $\mathbf{w}_1^1=\mathbf{0}$ and $\mathbf{w}_{n+1}^1=\mathbf{w}_n^{|\mathscr{W}_n|+1}$. Further, the following is satisfied:

$$\delta_{n,j}\left(L(\boldsymbol{y}^{(n,j)},\boldsymbol{y}^n)-{\mathbf{w}_n^j}^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right)=\delta_{n,j}\ell(\mathbf{w}_n^j;(\boldsymbol{x}^n,\boldsymbol{y}^{(n,j)})),\tag{3.23}$$

$$\mathbf{u}^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})-L(\boldsymbol{y}^{(n,j)},\boldsymbol{y}^n)\ge-\ell(\mathbf{u};(\boldsymbol{x}^n,\boldsymbol{y}^{(n,j)})),\tag{3.24}$$

where the first equality holds since in case that the difference on its left hand side is less than zero, then the step $\delta_{n,j}$ is zero according to definition (3.18). According to the definition of γ_n^j, the previous two formulas and the parameter change (3.21), we get that

$$\begin{aligned}\gamma_n^j&=\left\|\mathbf{w}_n^j-\mathbf{u}\right\|^2-\left\|\mathbf{w}_n^{j+1}-\mathbf{u}\right\|^2\\&=\left\|\mathbf{w}_n^j-\mathbf{u}\right\|^2-\left\|\mathbf{w}_n^j-\mathbf{u}+\delta_{n,j}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right\|^2\\&=-2\delta_{n,j}(\mathbf{w}_n^j-\mathbf{u})^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})-\delta_{n,j}^2\left\|\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right\|^2\\&=-2\delta_{n,j}\left({\mathbf{w}_n^j}^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})-L(\boldsymbol{y}^{(n,j)},\boldsymbol{y}^n)\right)\\&\quad+2\delta_{n,j}\left(\mathbf{u}^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})-L(\boldsymbol{y}^{(n,j)},\boldsymbol{y}^n)\right)-\delta_{n,j}^2\left\|\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right\|^2\\&\ge\delta_{n,j}\left(2\ell(\mathbf{w}_n^j;(\boldsymbol{x}^n,\boldsymbol{y}^{(n,j)}))-\delta_{n,j}\left\|\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right\|^2-2\ell(\mathbf{u};(\boldsymbol{x}^n,\boldsymbol{y}^{(n,j)}))\right),\end{aligned}$$

which after summing and using (3.22) provides the desired inequality. □

Theorem 1. *Let $(\boldsymbol{x}^1,\boldsymbol{y}^1),\ldots,(\boldsymbol{x}^N,\boldsymbol{y}^N)$ be a sequence of examples where $\|\Delta\mathbf{F}_n(\boldsymbol{y})\|\le 1$ and $L(\boldsymbol{y}^n,\boldsymbol{y})\le C$ for all $\boldsymbol{y}\in\mathscr{W}_n$, $n=1,\ldots,N$. Then, for any parameter vector $\mathbf{u}$ and the restricted*

k-best update defined by (3.21), it follows

$$\sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}_{\mathbf{w}_n^j})^2 \leq \|\mathbf{u}\|^2 + 2C\sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|} \ell(\mathbf{u}; (\boldsymbol{x}^n, \boldsymbol{y}^{(n,j)})), \tag{3.25}$$

where $\mathscr{W}_n = (\boldsymbol{y}^{(n,1)}, \ldots, \boldsymbol{y}^{(n,|\mathscr{W}_n|)})$ contains all sequences from $\mathscr{B}^{k,n}_{\mathbf{w}_n^1}$ on which we make a non-zero update according to (3.21).

Proof. In the proof we use abbreviations $L_{n,j} = L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}_{\mathbf{w}_n^j})$, $\ell_{n,j} = \ell(\mathbf{w}_n^j; (\boldsymbol{x}^n, \boldsymbol{y}^{(n,j)}))$ and $\widehat{\boldsymbol{y}}_{n,j} = \widehat{\boldsymbol{y}}_{\mathbf{w}_n^j}$. According to the definition (3.19) of the prediction sequence, it follows

$$\mathbf{w}_n^{j\,\mathsf{T}}\mathbf{F}(\boldsymbol{x}^n, \widehat{\boldsymbol{y}}_{n,j}) \geq \mathbf{w}_n^{j\,\mathsf{T}}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}), \quad \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n),$$

which leads to inequality

$$L_{n,j} \leq \mathbf{w}_n^{j\,\mathsf{T}}\mathbf{F}(\boldsymbol{x}^n, \widehat{\boldsymbol{y}}_{n,j}) - \mathbf{w}_n^{j\,\mathsf{T}}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n) + L_{n,j} \leq \ell(\mathbf{w}_n^j; (\boldsymbol{x}^n, \widehat{\boldsymbol{y}}_{n,j})) \leq \ell_{n,j}, \tag{3.26}$$

where the last inequality in (3.26) comes from the definition of the set $\mathscr{W}_n$, i.e. because of $\boldsymbol{y}^{(n,j)} \in A_{\mathbf{w}_n^j}$. According to the Theorem condition $\left\|\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right\| \leq 1$, $j = 1, \ldots, |\mathscr{W}_n|$, we get an inequality for the step size

$$\delta_{n,j} = \min\left\{\frac{\ell_{n,j}}{\left\|\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right\|^2}, C\right\} \geq \min\{\ell_{n,j}, C\},$$

which leads to

$$\delta_{n,j}L_{n,j} \geq \min\{\ell_{n,j}L_{n,j}, CL_{n,j}\} \overset{(3.26)}{\geq} \min\{L_{n,j}^2, CL_{n,j}\} \geq L_{n,j}^2, \tag{3.27}$$

since $L_{n,j} \leq C$. Summing (3.27) for all examples (n, j) on which we made the update and using the inequality from (3.26) we get

$$\sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|} L_{n,j}^2 \leq \sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|} \delta_{n,j}L_{n,j} \leq \sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|} \delta_{n,j}\ell_{n,j}. \tag{3.28}$$

According to the definition of step size $\delta_{n,j}$, it is upper bound by parameter C, which allows us to rewrite the inequality from Lemma 1 as

$$\sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|} \delta_{n,j}\left(2\ell_{n,j} - \delta_{n,j}\left\|\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right\|^2\right) \leq \|\mathbf{u}\|^2 + 2C\sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|} \ell(u; (\boldsymbol{x}^n, \boldsymbol{y}^{(n,j)})).$$

Also, the definition of $\delta_{n,j}$ gives $\delta_{n,j}\left\|\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right\|^2 \leq \ell_{n,j}$ and thus the previous inequality

becomes

$$\sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|}\delta_{n,j}\ell_{n,j} \leq \|\mathbf{u}\|^2 + 2C\sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|}\ell(u;(\boldsymbol{x}^n,\boldsymbol{y}^{(n,j)})), \tag{3.29}$$

which in combination with (3.28) provides the desired bound. □

Corollary 1. *Let the conditions from the previous theorem be satisfied, then it follows*

$$\sum_{n=1}^{N}\sum_{j=1}^{|\mathscr{W}_n|}L(\boldsymbol{y}^n,\widehat{\boldsymbol{y}}_{\mathbf{w}_n^j})^2 \leq \|\mathbf{u}\|^2 + 2C\sum_{n=1}^{N}|\mathscr{W}_n|\ell_n(\mathbf{u}), \tag{3.30}$$

where $\ell_n(\mathbf{u}) = \max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)}\ell(\mathbf{u};(\boldsymbol{x}^n,\boldsymbol{y}))$.

In case of using only the best sequence, i.e. $\mathscr{W}_n = (\widetilde{\boldsymbol{y}})$, the bound from Corollary 1 reduces to the bound of 1-best case proved by Crammer et al. (2006). The proof of 1-best case uses a property $\ell(\mathbf{w}_n;(\boldsymbol{x}^n,\widetilde{\boldsymbol{y}})) \geq \ell(\mathbf{w}_n;(\boldsymbol{x}^n,\widehat{\boldsymbol{y}}))$, which is needed for inequality (3.27). However, this inequality does not hold if we change $\widetilde{\boldsymbol{y}}$ with an arbitrary sequence from k-best ones. In order to get a similar bound, updates must be restricted only to those examples which belong to the set $A_{\mathbf{w}_n}$. Nevertheless, checking if a sentence belongs to this set implies finding $\widehat{\boldsymbol{y}}$ every time we change the parameters (see Algorithm 5) which is computationally expensive. In the experiment section, we will consider both restricted k-best PA and k-best PA updates, even though for latter the previously proved prediction loss bound will not be satisfied.

3.2.5 k-best Perceptron

In case of not using a cost function, i.e. when $L(\boldsymbol{y}^n,\boldsymbol{y}) = 0$ for all $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$, the passive-aggressive algorithm reduces to the perceptron algorithm (Collins, 2002) with a fixed step size for the best sequence. To keep the spirit of the online manner of the single best perceptron, the k-best version should involve online traversing through k-best structures $(\boldsymbol{y}^{(n,1)},\ldots,\boldsymbol{y}^{(n,k)}) = \mathscr{B}^{k,n}_{\mathbf{w}_n}$ and sequential changes of parameters with the constant step size for each structure which belongs to the error set, i.e. for each structure which produces a higher score than the original structure. After finding the k-best structure, the following series of parameter change is applied

$$\mathbf{w}_n^{j+1} = \begin{cases} \mathbf{w}_n^j + \Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)}) & , \text{ if } \boldsymbol{y}^{(n,j)} \in Err_{\mathbf{w}_n^j} \\ \mathbf{w}_n^j & , \text{ if } \boldsymbol{y}^{(n,j)} \notin Err_{\mathbf{w}_n^j} \end{cases}, \quad j = 1,\ldots,k, \tag{3.31}$$

where $Err_{\mathbf{w}_n} = \{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n) : \mathbf{w}_n^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}) \leq 0\}$. The condition under which the k-best perceptron converges is the same as for 1-best case.

Theorem 2. *Let* $\mathscr{D} = ((\boldsymbol{x}^n,\boldsymbol{y}^n))_{n=1}^N$ *be a training set and let's suppose that there exist a vector* $\mathbf{u}$ *and* $\gamma > 0$ *such that* $\|\mathbf{u}\| = 1$ *and* $\mathbf{u}^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}) \geq \gamma$ *for all training examples* n *and for all*

$\boldsymbol{y} \in \mathscr{Y}_{-n}$. *For the* k*-best perceptron from Algorithm 5, it follows that*

$$\textit{Number of mistakes } \leq \frac{R^2}{\gamma^2},$$

where R *is a constant such that* $\forall n, \forall \boldsymbol{y} \in \mathscr{Y}_{-n}$, $\|\Delta\mathbf{F}_n(\boldsymbol{y})\| < R$.

Proof. For a sequence $\boldsymbol{y} \in \mathscr{Y}_{-n}$ a mistake is made with parameters $\mathbf{w}$ if $\mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y}) \leq 0$. Let $\overline{\mathbf{w}}^{(l)}$ be a vector before the l-th mistake. Suppose that the mistake is made on the n-th example, on the j-th sequence taken from k-best sequences generated from parameters $\mathbf{w}_n$, i.e. on the sequence $\boldsymbol{y}^{(n,j)} \in \mathscr{B}^{k,n}_{\mathbf{w}_n}$ where $\overline{\mathbf{w}}^{(l)} = \mathbf{w}_n^j$. According to the algorithm, it follows that $\overline{\mathbf{w}}^{(l+1)} = \overline{\mathbf{w}}^{(l)} + \Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})$ and taking the inner product of both sides with parameters $\mathbf{u}$ gives

$$\mathbf{u}^\mathsf{T}\overline{\mathbf{w}}^{(l+1)} = \mathbf{u}^\mathsf{T}\overline{\mathbf{w}}^{(l)} + \mathbf{u}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)}) \geq \mathbf{u}^\mathsf{T}\overline{\mathbf{w}}^{(l)} + \gamma.$$

Since $\overline{\mathbf{w}}^{(1)} = 0$ and $\mathbf{u}^\mathsf{T}\overline{\mathbf{w}}^{(1)} = 0$, it follows by induction that $\mathbf{u}^\mathsf{T}\overline{\mathbf{w}}^{(l+1)} \geq l\gamma$, and by the use of $\mathbf{u}^\mathsf{T}\overline{\mathbf{w}}^{(l+1)} \leq \|\mathbf{u}\| \left\|\overline{\mathbf{w}}^{(l+1)}\right\|$ we get $\left\|\overline{\mathbf{w}}^{(l+1)}\right\| \geq l\gamma$. Further,

$$\left\|\overline{\mathbf{w}}^{(l+1)}\right\|^2 = \left\|\overline{\mathbf{w}}^{(l)}\right\|^2 + 2\overline{\mathbf{w}}^{(l)\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)}) + \left\|\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right\|^2 \leq \left\|\overline{\mathbf{w}}^{(l)}\right\|^2 + R^2,$$

because parameters $\overline{\mathbf{w}}^{(l)}$ make a mistake on $\boldsymbol{y}^{(n,j)}$, i.e. $\overline{\mathbf{w}}^{(l)\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)}) \leq 0$. By induction, we get $\left\|\overline{\mathbf{w}}^{(l+1)}\right\|^2 \leq lR^2$. Combining the bounds $\left\|\overline{\mathbf{w}}^{(l+1)}\right\| \geq l\gamma$ and $\left\|\overline{\mathbf{w}}^{(l+1)}\right\|^2 \leq lR^2$, we get the upper bound for the number of mistakes

$$l^2\gamma^2 \leq \left\|\overline{\mathbf{w}}^{(l+1)}\right\|^2 \leq lR^2 \quad \Longrightarrow \quad l \leq \frac{R^2}{\gamma^2}.$$

□

The proof is very similar to the standard structured perceptron (Collins, 2002) since the property of the best sequence is not used in the proof. In the case of inseparable data, there is also a theorem that bounds the number of mistakes, proven in (Freund and Schapire, 1999) for the online perceptron and extended in (Collins, 2002) for the structured perceptron. The same bound holds for the k-best variant.

Theorem 3. *Let* $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$ *be a training set, let* $\mathbf{u}$ *be any vector with* $\|\mathbf{u}\| = 1$ *and* $\gamma > 0$. *Define*

$$m_n = \mathbf{u}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n) - \max_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \mathbf{u}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}), \quad \epsilon_n = \max\{0, \gamma - m_n\}$$

and $D = \sqrt{\sum_{n=1}^N \epsilon_n^2}$. *For the first pass over the training set of* k*-best perceptron from Algorithm*

5,

$$\textit{Number of mistakes} \leq \left(\frac{R+D}{\gamma}\right),$$

where R is a constant such that $\forall n, \forall \boldsymbol{y} \in \mathscr{Y}_{-n}$, $\|\Delta \mathbf{F}_n(\boldsymbol{y})\| < R$.

The idea of the proof is to transform an inseparable case into a separable one, then apply the theorem for a separable case to get the bound, and at the end show that the prediction with the original parameters is the same as with the transformed parameters. The proof is identical as in (Collins, 2002), where the only difference is that we apply Theorem 2 for k-best perceptron when we get a separable case.

Proof. First, we extend the feature vector $\mathbf{F}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d$ to $\overline{\mathbf{F}}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^{d+N}$ such that $\overline{\mathbf{F}}_i(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{F}_i(\boldsymbol{x}, \boldsymbol{y})$, $i = 1, \ldots, d$, and $\overline{\mathbf{F}}_{d+n}(\boldsymbol{x}, \boldsymbol{y})$ is equal to Z if $(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^n, \boldsymbol{y}^n)$ and equal to zero otherwise, for $n = 1, \ldots, N$. The vector $\mathbf{u}$ is extended to $\overline{\mathbf{u}} \in \mathbb{R}^{d+N}$ in a similar way: $\overline{\mathbf{u}}_i = \mathbf{u}_i$, $i = 1, \ldots, d$ and $\overline{\mathbf{u}}_{d+n} = \epsilon_n / Z$, $n = 1, \ldots, N$. Transformed vectors hold the following properties:

$$\begin{gathered}
\|\overline{\mathbf{u}}\|^2 = \|\mathbf{u}\|^2 + \textstyle\sum_{n=1}^{N} \epsilon_n^2 / Z^2 = 1 + D^2/Z^2 \\
\forall n, \forall \boldsymbol{y} \in \mathscr{Y}_{-n}, \overline{\mathbf{u}}^{\mathsf{T}} \overline{\mathbf{F}}(\boldsymbol{x}^n, \boldsymbol{y}^n) - \overline{\mathbf{u}}^{\mathsf{T}} \overline{\mathbf{F}}(\boldsymbol{x}^n, \boldsymbol{y}) \geq \gamma \\
\forall n, \forall \boldsymbol{y} \in \mathscr{Y}_{-n}, \left\| \overline{\mathbf{F}}(\boldsymbol{x}^n, \boldsymbol{y}^n) - \overline{\mathbf{F}}(\boldsymbol{x}^n, \boldsymbol{y}) \right\| < R^2 + Z^2.
\end{gathered}$$

From the first two properties, it follows that parameters $\overline{\mathbf{u}} / \|\overline{\mathbf{u}}\|$ separate data $\mathscr{D}$ with the margin $\gamma / \sqrt{1 + D^2/Z^2}$. Now, we can apply Theorem 2 and get that the number of mistakes k-best perceptron makes running on extended space is at most $\frac{1}{\gamma}(R^2 + Z^2)(1 + \frac{D^2}{Z^2})$. The value $Z = \sqrt{RD}$ minimizes the bound, giving us the statement of the theorem. Extended parameters generated from the first pass of the algorithm make the same prediction as the original parameters on test examples since the additional parameters affect only single training data. □

3.3 Results and discussion

We present experimental results on two sequence labeling tasks, the shallow parsing (Tjong Kim Sang and Buchholz, 2000) on CONLL-2000 corpus[1] and the named entity recognition in Spanish on CONLL-2002 corpus[2] (Tjong Kim Sang, 2002). In further text, we will address algorithms by their names removing the "k-best" prefix, and where needed, specifying the exact parameter.

[1] http://www.cnts.ua.ac.be/conll2000/chunking
[2] http://www.cnts.ua.ac.be/conll2002/ner

Algorithm 5: k-best perceptron, k-best PA and k-best restricted PA (RPA)

Input : Training data: $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$, parameter $C \in \mathbb{R}^+$, $k \in \mathbb{N}$
Number of iterations: P
Output: Model parameters: $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{0}$;
for $p \leftarrow 1$ **to** P **do**
  foreach $(\boldsymbol{x}^n, \boldsymbol{y}^n)$ ***in*** $\mathscr{D}$ **do**
    $\mathscr{W} \leftarrow \mathscr{B}_{\mathbf{w}}^{k,n}$; *update* $\leftarrow$ true;
    foreach $\boldsymbol{y}$ ***in*** $\mathscr{W}$ **do**
      $Err \leftarrow \begin{cases} \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y}) < L(\boldsymbol{y}^n, \boldsymbol{y}) & \text{(PA, RPA)} \\ \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y}) \leq 0 & \text{(Perceptron)} \end{cases}$
      if *update* **then**
        $\widehat{\boldsymbol{y}} \leftarrow \arg\max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \mathbf{w}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y})$; (RPA)
      end
      $Err \leftarrow Err$ and $\ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) \geq \ell(\mathbf{w}; (\boldsymbol{x}^n, \widehat{\boldsymbol{y}}))$; (RPA)
      if Err **then**
        $\delta \leftarrow \begin{cases} \min\left\{\ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))/ \ \|\Delta\mathbf{F}_n(\boldsymbol{y})\|^2, C\right\} & \text{(PA, RPA)} \\ 1 & \text{(Perceptron)} \end{cases}$
        $\mathbf{w} \leftarrow \mathbf{w} + \delta\,\Delta\mathbf{F}_n(\boldsymbol{y})$; *update* $\leftarrow$ true;
      else
        update $\leftarrow$ false;
      end
    end
  end
end

3.3.1 Problem description and features

Shallow parsing or *chunking* is a task of identifying non-overlapping text segments which correspond to certain syntactic units (chunks), such as noun phrases, verb phrases, prepositional phrases, etc. The CONLL-2000 corpus contains around a quarter of a million words already split for training and testing. Each word has a corresponding POS tag and a label. The labels are presented in BIO representation, where B stands for the beginning of a chunk, I for the interior, and O means that a word does not belong to any chunk. For each word, we first detect its characteristics which we use as local features. We extract standard features like the detection of special characters, the detection of numbers, a characteristic suffix of the word, belonging to a characteristic dictionary, whether the word is capitalized or all caps. All bigrams are constructed for a word and its local feature (including the POS tag) for the current and previous position, while unigrams are constructed only for the current position. These bigrams and unigrams with the combination of the current and previous label are used to create a feature vector at the current position. The results are presented in terms of F-measure, as a harmonic mean of

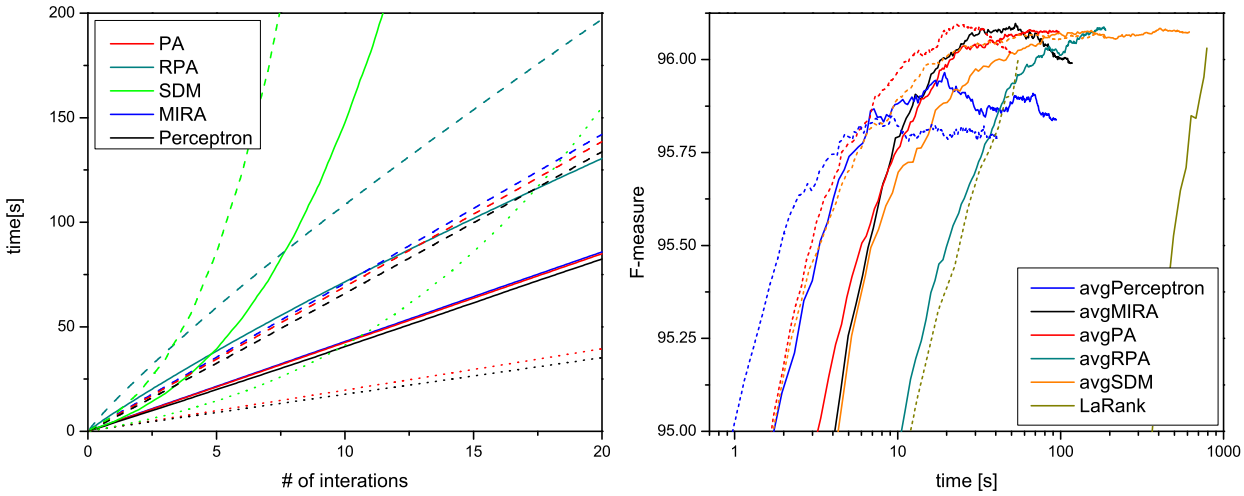

Figure 3.2: Training time comparison for different k-best algorithms on shallow parsing. The left panel shows time through iterations for different values of the parameter k, where the algorithms are denoted with colors, while k is represented with a line style: $k = 1$ with dots, $k = 5$ with a solid line, and $k = 10$ with a dashed line. The right panel shows the F-measure through time for $k = 1$ (dashed line) and $k = 5$ (solid line), where the LaRank trained in one pass uses parameter $n_R = 10$.

a precision and recall computed over tokens belonging to a chunk.

Named entity recognition (NER) is a task of detecting and classifying entities into specified categories, such as names of persons, organizations, locations, times, dates, etc. We use CoNLL-2002 data for Spanish NER. The corpus is divided into the training, test and development part containing four types of entities: person, organization, location and miscellaneous in standard BIO representation. The feature vectors are created in the same way as in the shallow parsing problem. We evaluate algorithms directly on the provided corpus without any additional external knowledge of the language.

3.3.2 Time and accuracy comparison

We have implemented all described algorithms in C++. The experiments are performed on a computer with Intel Core 2 Duo CPU 2.33 GHz and 8 GB RAM. We use A* decoding with Viterbi scores for the heuristic function to find the k-best paths (Nagata, 1994; Soong and Huang, 1991). To avoid oscillations during the learning process, we have applied parameter averaging as described in (Collins, 2002). Such algorithms we will denote using the prefix *avg*. Fig. 3.2 shows the speed comparison of the considered algorithms. All implemented algorithms share the same structures and operators when working with feature and weight vectors, thus the speed comparison shown in Fig. 3.2 can be considered reliable. The perceptron as the simplest method is slightly faster than the PA algorithm but on the right panel we can see that the other algorithm which incorporates the cost function provides better results. Recall that the RPA

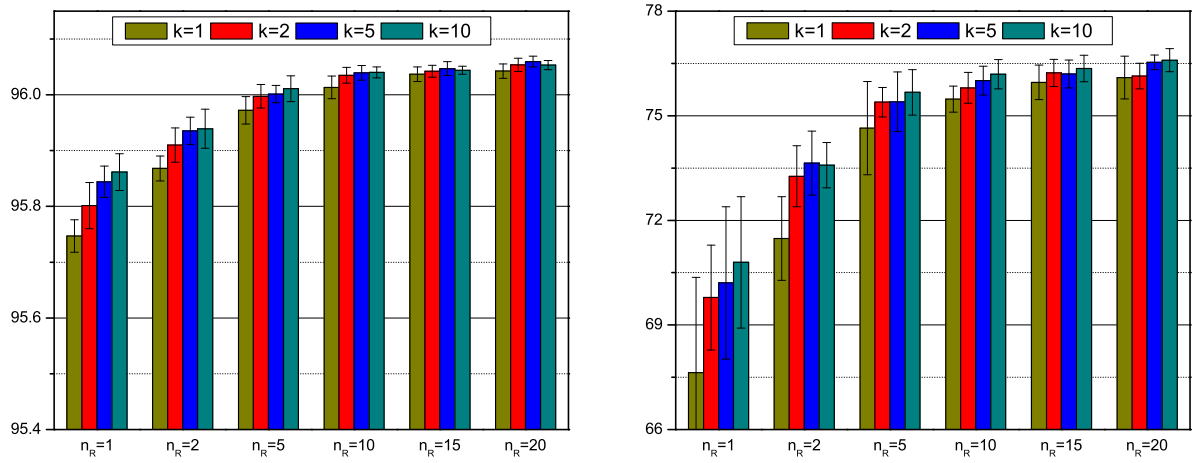

Figure 3.3: The results for k-best LaRank on shallow parsing (left) and NER in Spanish (right), for different k specified in legend and for different values of parameter n_R. Each column height represents the mean value of F-measure over 20 repetitions and the corresponding error bar represents the mean absolute error.

algorithm needs an additional 1-best decoding, and as a result of the necessity for additional decoding, for each k, its time deviates from the other algorithms with similar time consumption, the PA, perceptron and MIRA. The SDM requires significantly more time through iterations due to its continuously increasing active set of constraints. The heuristics from (Balamurugan et al., 2011), which control the set growth, can help to reduce the training time.

Next, we tested the LaRank algorithm trained in one pass. The results are presented in Fig. 3.3 for different values of parameters k and n_R. Since the selection of examples in REPROCESS operations is subject to random function, we presented the mean value of F-measure with the corresponding mean absolute error. In order to select regularization parameter C for the shallow parsing problem, we perform 5-fold cross-validation. We select the highest mean value of F-measure over 20 repetitions, and then we use this optimal parameter in a test scenario. For the NER problem, we use a development set to select the optimal parameter with the same scenario. The results on both tasks suggest the advantage of k-best versions over the single best version, especially with the lower values of parameter n_R. Also, we can see that a higher number of n_R has a positive influence to the F-measure.

Further, for different values of parameter k we present a single number for each algorithm where the other parameters (regularization parameter, the number of training iterations) are 5-fold cross-validated on the shallow parsing problem and estimated on the development set for NER. For shallow parsing, we select the best combination of the regularization parameter $C \in \{10^{-2}, 10^{-1}, 1, 10\}$ and the number of training epochs from the set $\{5, 10, 15, 20\}$. For

Table 3.1: Results for different algorithms and their corresponding parameters (regularization parameters, parameter k, and the number of training epochs) obtained from 5-fold cross-validation (shallow parsing) and from the development set (NER). For all algorithms, the results are presented with averaged parameters, so the *avg* prefix is omitted in the algorithm name. The number of training iterations is denoted with # and the tolerance for KKT conditions τ is set to 10^{-10}.

Shallow parsing												
	$k=1$			$k=2$			$k=5$			$k=10$		
Method	C	#	F-measure	C	#	F-measure	C	#	F-measure	C	#	F-measure
Perc.	–	5	95.821	–	5	95.892	–	5	**95.924**	–	5	95.875
PA	1	10	96.093	10^{-2}	15	**96.097**	10^{-2}	15	96.069	10^{-2}	15	96.056
RPA	1	10	96.093	10^{-2}	20	**96.099**	10^{-2}	20	96.079	10^{-2}	15	96.057
MIRA	1	10	96.066	10^{-1}	10	96.053	10^{-1}	5	**96.071**	10^{-2}	20	96.061
SDM	10^{-1}	20	96.057	10^{-1}	20	96.080	10^{-1}	20	96.075	10^{-1}	20	**96.081**
Named entity recognition (Spanish)												
	$k=1$			$k=2$			$k=5$			$k=10$		
Method	C	#	F-measure	C	#	F-measure	C	#	F-measure	C	#	F-measure
Perc.	–	30	75.886	–	30	76.019	–	30	**76.204**	–	30	76.122
PA	10^{-1}	30	76.349	10^{-1}	30	76.436	1	30	**76.640**	10^{-1}	30	76.549
RPA	10^{-1}	30	76.349	1	30	76.636	10^{-1}	30	**76.741**	10^{-1}	30	76.608
MIRA	1	30	76.262	1	30	**76.334**	1	30	76.312	10^{-1}	30	76.328
SDM	1	30	75.840	1	30	76.028	1	30	**76.194**	1	30	76.145

that problem, algorithms require less epochs to converge, while 30 epochs are also added to the previous set for NER, as the results were still improving after 20 iterations. Results are presented in Table 3.1. MIRA was optimized with the SMO with the practical check of KKT conditions (3.10)-(3.11) by setting tolerance $\tau = 10^{-10}$ for all values of k, and thus for $k = 1$ its results do not match the PA algorithm. For $k = 10$ there is usually a degradation of results, possibly because the inclusion of a lot of features into the training procedure via k-best sequences can raise the problem of overfitting. A problem with a higher k can rise in algorithms which are defined in an online manner inside an example, such as the PA, RPA and perceptron algorithm, as opposite to MIRA and SDM which perform full optimization inside an example. However, we can see that all k-best versions of algorithms make an improvement over the single best case and the best results are usually achieved with smaller values $k = 2$ and $k = 5$.

3.3.3 Statistical significance

In this section we discuss the statistical significance of the previously presented results. The improvements of the k-best version over the single best one are shown in Table 3.1 and now we want to test if these improvements are statistically significant. We tested the null hypothesis stating that there is no difference between the k-best and the single best version against the

Table 3.2: Contingency tables for the single best vs. the k-best version of different algorithms over different datasets. The column "Total" represents the contingency table across datasets and the last column represents its McNemar's test statistic.

	Shallow parsing		NER (Spanish)		Total		McNemar's test statistic
avgPerc	199265	1071	251189	688	450454	1759	37.230
	1180	10041	960	11754	2140	21795	
avgPA	200281	451	252092	131	452373	582	29.326
	548	10277	234	12134	782	22411	
avgRPA	200301	431	252088	135	452389	566	26.512
	537	10288	216	12152	753	22440	
avgMIRA	200223	477	252178	122	452401	599	1.559
	481	10376	162	12129	643	22505	
avgSDM	200655	85	251929	237	452584	322	1.086
	101	10716	248	12177	349	22893	

alternative hypothesis stating that there is a difference between these versions. The hypotheses are tested for each algorithm separately over multiple datasets. For this purpose, we used the recommended evaluation scenario for comparing classifiers described in (Salzberg, 1997). The results are generated with a 10-fold cross validation on the training corpora. We used the same sets as described in Section 3.3.2 to select optimal regularization parameters and the number of training epochs, while the set {2,5,10} is used for parameter k. In this way an algorithm with the best combination of the regularization parameter, the number of epochs and the value of k is trained on each of 9 folds and tested on the remaining fold. The collected results for each algorithm on each dataset are presented in Table 3.2 below, as a 2x2 contingency table with the following form:

	k-best positive	k-best negative
single best positive	PP	PN
single best negative	NP	NN

where PN, for example, represents the number of labels that a single best algorithm classifies correctly and its k-best version classifies incorrectly, while NP is the number of labels that the k-best version classifies correctly and the single best does not. The contingency table is used to

calculate McNemar's test statistic (McNemar, 1947)

$$\chi^2 = \frac{(PN - NP)^2}{PN + NP},$$

which has approximately chi-squared distribution with one degree of freedom under the null hypothesis. Note that values PP and NN from the table (the number of labels that both algorithms classify correctly and incorrectly, respectively) are not involved in the test statistic. With the significance level of 0.05, the critical value is 3.84, and we can reject the null hypothesis for all algorithms except the SDM and MIRA. We can conclude that with the significance level of 0.05, the improvements of the k-best version of the Perceptron, RPA and PA algorithm over the single best versions are statistically significant, while for the MIRA and SDM they are not.

Chapter 4

Confidence based learning of a two-model committee

This chapter presents the use of a two structured model committee, where the output of the first model together with its confidence is set as the input of the second model. The confidence for the given context of predictions in the sequence is extracted from the alternative hypotheses generated from the first model. We present experiments on shallow parsing, comparing the performance of the proposed method to separate models.

4.1 Introduction to committee based methods

As we have seen in two previous chapters, over the last decade, various structured learning algorithms have been developed with the goal of predicting a complex structure of labels over the input, instead of just assigning a single label to it. In order to improve the results in this area further, it has been considered to combine structured models in a committee (Nguyen and Guo, 2007; Song and Sarkar, 2008). The idea of combining multiple models rises from the question whether we can create a combination of these models so that they obtain better prediction performance than the individual models separately. In this chapter we have adopted this approach and suggested the combination of two well-known models, conditional random fields and structured support vector machines. The combination of these learning models is based on their training in a sequence, with the predictions and additional context confidence feature passed from one classifier to another.

The *committee*-based methods assume combining multiple models in order to produce a final prediction with an improved accuracy. An example of a committee is boosting (Kearns and Valiant, 1989; Schapire, 1990), which assumes to combine multiple "weak models", i.e. models which perform only slightly better than a random guessing, to get a "strong" model, i.e. the one which has good correlation with the true classification. The committee of two models can be used for extracting additional global features for the second model. Collins (2000) uses this for the re-ranking approach, where the second model attempts to improve the initial rank-

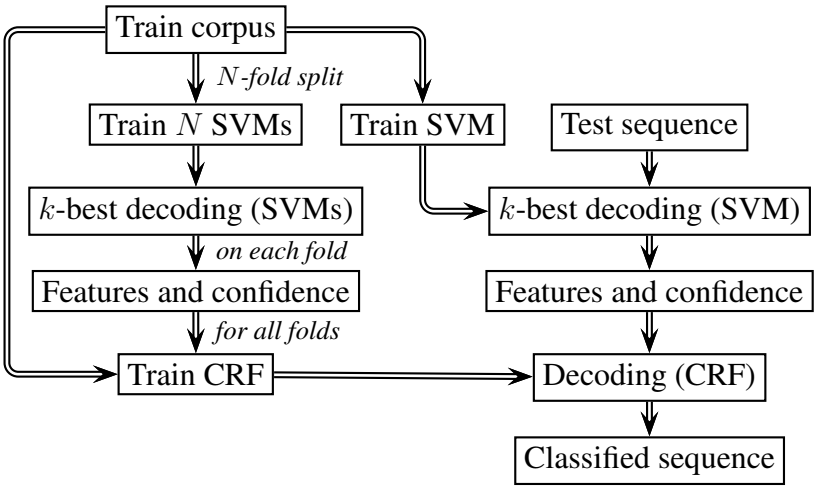

Figure 4.1: An organization of the training and classification phase of a two-model committee with a confidence. In this example, the first model is the SVM, which is used to extract confidence using k-best sequences for the second model, the CRF. The training corpus for the second model is filled with predictions and the corresponding confidence from the first model. They are generated from first model on each fold after its training on the remaining N-1 folds.

ing with these additional features. Similarly, Song and Sarkar (2008) used the combination of two structured classifiers, where the first model, the CRF, is used to produce global features for the second one, the perceptron (Collins, 2002), as well as a restrict search space for the second model. Hoefel and Elkan (2008) present the results for sequence labeling by combining the classical (non-structured) SVM with the CRF model. They state that the intuition for choosing these two models was that both learning approaches are somewhat orthogonal in their advantages, and their combination can achieve high accuracy. Nguyen and Guo (2007) used an ensemble of structured classifiers and obtained more accurate solutions compared with the best results of the individual structured models.

In the following text, we will first describe the organization of the training and classification phase in a two-model committee in Section 4.2, and then introduce the concept of context confidence of predictions in Section 4.3. The last section compares the results of committee methods to the results of individual models on a shallow parsing task.

4.2 Committee organization

We use a committee of two classifiers, where the first one is the structured SVM and the second one is the CRF. The structured SVM is trained using the k-best sequential dual method described in Section 3.2.2 without the parameter averaging. After the optimal parameters are found, the model is used to generate n-best candidates (sequences) for the second classifier both for training and test data. As the second model should use the output from the first one on both

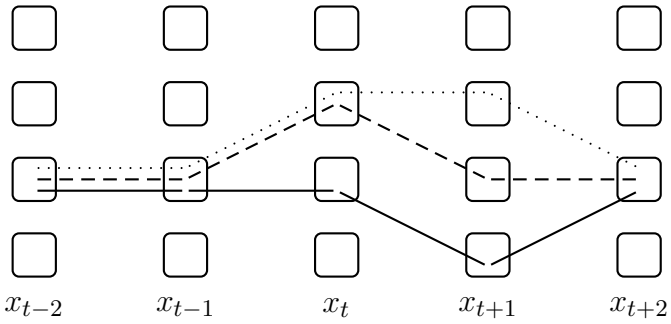

Figure 4.2: An illustration of predicted labels around the tth observation in a sequence. Let the dashed line connect the predicted labels, and solid and dotted lines represent alternative paths with the highest score. At observations x_{t-2}, x_{t-1} and x_{t+2}, we see that all paths pass through the same label and the confidence for these labels is equal to one, which is not the case with observations x_t and x_{t+1} where the confidence is $\nu_t = 2/3$ and $\nu_{t+1} = 1/3$.

training and test data, we cannot simply generate these outputs after training one SSVM, since the confidence of the output will be different on seen and unseen data. To do this, we use a 5-fold split of the training data, where similarly to the cross-validation technique, each time the training is performed at all folds except one and the outputs are generated for the excluded one. The outputs are used to extract confidence, explained in the next section, and then these outputs with the assigned confidence are set as features for the second model. The CRF is trained by the limited-memory quasi-Newton method L-BFGS (Nocedal and Wright, 2006). Once the CRF model is trained, for the classification phase we need one additional trained SVM model on the whole training corpus, which will pass its outputs to the CRF model. At the end, the CRF produces the final decision on test data. Figure 4.1 describes the flow of training and test phase.

4.3 Confidence feature

The *confidence* score of predicted labels can be used for committee methods. Schapire and Singer (1999) introduced the boosting algorithm using confidence rated predictions, where each weak model assigns confidences to each of its predictions. Culotta and McCallum (2004) studied field confidence using the constrained forward-backward algorithm for CRFs, which they used for creating a structural database from text documents. In the recent years, Mejer and Crammer (2010) have been using confidence-weighted learning combined with different types of confidence. Following these ideas, we defined a confidence for each context of predictions, which estimates the reliability of consecutive predictions from the first classifier in the committee. These confidences together with the corresponding predictions are set as additional features of the next classifier.

Let $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n)}$ be the n-best sequences generated from the first classifier for the observation sequence $\boldsymbol{x}$. To define a confidence of a predicted label, one of the approaches from

Mejer and Crammer (2010) was to assign uniform importance to each of these n labellings and measure their agreement with the predicted labels. If $\hat{\boldsymbol{y}}$ is the predicted sequence of labels for the observation $\boldsymbol{x}$, found by (2.27), the confidence ν_t of the t-th label will be

$$\nu_t = \frac{|\{i : y_t^{(i)} = \hat{y}_t\}|}{n}. \tag{4.1}$$

Figure 4.2 illustrates an example of three best paths around the tth position in sequence from which we can calculate the confidence of prediction. Let C_t be a set of positions for the context around the t-th prediction. We consider a confidence of the predictions assigned to a context as a product of the confidences of the individual labels defined by (4.1), i.e.

$$c_t = \prod_{i \in \mathsf{C}_t} \nu_i. \tag{4.2}$$

As the context at position t we used three consecutive predictions: the previous, the current and the next prediction, i.e. $\mathsf{C}_t = \{t-1, t, t+1\}$. Then, the context confidence is discretized into intervals and the features are created which determine belonging to an interval. One of the intervals should contain only the value 1 and represent the maximal confidence of predictions. This interval should be a good indicator of the correct prediction of the first model. The statistics of the maximal confidence of the first model on the CONLL-2000 corpus[1] shows that 97.7% of its predictions are correct. On the other hand, when a confidence does not belong to this interval, 23.6% of predictions are not correct, in which case that second model should try to improve the result.

4.4 Experiments

We present experiments for shallow parsing (Tjong Kim Sang and Buchholz, 2000) on the CONLL-2000 corpus. Shallow parsing identifies non-overlapping text segments which correspond to certain syntactic units. It is a step preceding full parsing and it belongs to the sequence labeling problem where each token has to be labeled by one of the specified tags.

Experimental setup

For training a structured SVM, we used the k-best SDM algorithm with $k = 5$. Also, the number of output sequences from the SVM is $n = 5$. The model is trained with a regularization parameter $C = 0.1$, while the precision for violating the KKT conditions is set to $\tau = 10^{-3}$. Local characteristics for an observation (a word) will be called attributes of a word. As the corpus already contains a part-of-speech (POS) tag for each observation, we used the POS tag in com-

[1] http://www.cnts.ua.ac.be/conll2000/chunking

Table 4.1: Templates used for generating features at position t in a sequence, where w_t denotes a current word, a_t denotes attributes for the current word, p_t is the prediction of the previous model for the current word and ca_t is a discretized attribute for the context confidence of the current word.

Base templates (BT)	
(y_t, y_{t-1})	
(y_t, a_t)	(y_t, w_t)
(y_t, y_{t-1}, w_t)	(y_t, y_{t-1}, a_t)
$(y_t, y_{t-1}, w_t, w_{t-1})$	$(y_t, y_{t-1}, w_t, a_{t-1})$
$(y_t, y_{t-1}, a_t, w_{t-1})$	$(y_t, y_{t-1}, a_t, a_{t-1})$
Prediction templates (PT)	**Confidence templates (CT)**
(p_t)	(ca_t)
(p_t, w_t)	(ca_t, w_t)
(p_t, a_t)	(ca_t, a_t)

bination with other lexical characteristics to create attributes of a word. These characteristics include: if a token is written in all caps, the initial cap or lowercase; if a token contains digits, if it represents a special character, a punctuation mark or an abbreviation, the belonging to specific external linguistic dictionaries. At each position in a sentence, Table 4.1 shows base templates used for generating features for the SVM as different combinations of words, attributes and their labels. It also shows additional prediction templates and confidence templates which can be used in addition to base templates. From the output of the SVM, the confidence for context C_t is extracted and discretized into intervals [0,1) and [1,1]. After training the SVM, we use the *CRFsuite* (Okazaki, 2007) implementation for the CRF with default settings.

Results

We compare individual models with base templates, the SVM[BT] and the CRF[BT], to the committee with predicted labels from the previous model (without confidence) SVM[BT]-CRF[BT+PT], and the committee with the included confidence SVM[BT]-CRF[BT+PT+CT]. The Table 4.2 shows the precision, recall and F-measure as the harmonic mean of precision

Table 4.2: Performances for shallow parsing on the CONLL-2000 test data set. Templates from TABLE 4.1 used for the training of the corresponding model are specified inside square brackets.

Model	Precision	Recall	F-measure
SVM[BT]	94.94	96.10	95.52
CRF[BT]	95.23	96.91	96.06
SVM[BT]-*CRF*[BT+PT]	95.43	96.96	96.19
SVM[BT]-*CRF*[BT+PT+CT]	95.48	96.96	96.21

and recall. Tokens which do not belong to any chunk are labeled by a special tag which is not included in the calculation of total measures. We see that the committee of the SVM-CRF performs better than the individual models both for precision and recall. Additionally, the included confidence of predicted labels contributes to improving the total precision of the committee with no influence on its recall.

Chapter 5

Structured classification with the ramp loss

The chapter presents a sequential dual method for the non-convex structured ramp loss minimization. The method uses the concave-convex procedure which transforms a non-convex problem iterativelly into a series of convex ones. The sequential minimal optimization is used to deal with the convex optimization by sequentially traversing through the data and optimizing parameters associated with the incrementally built set of active structures inside each of the training examples. The chapter includes the results on two sequence labeling problems, shallow parsing and part-of-speech tagging, and also presents the results on artificial data when the method is exposed to outlayers. The comparison with a primal sub-gradient method with the structured ramp and hinge loss is also presented.

5.1 Non-convex SVMs

Support vector machines (SVMs) (Vapnik, 1998) usually assume a convex loss during the optimization. The convexity contributes to an easier optimization which ends up in the global optimum. The binary version of a SVM is extended to the structured version (Tsochantaridis et al., 2004) by optimizing the structured hinge loss and retaining its convex property. Structured classifiers allow better incorporation of features into the learning procedure since they can deal with the connections between parts of the structure, which results in better recognition results in comparison to standard binary and multiclass classifiers. Over the last decade, many algorithms adapted to the structured version have been written, such as the Perceptron algorithm (Collins, 2002), the passive-aggressive algorithm (Crammer et al., 2006), the sequential dual method (Balamurugan et al., 2011), the Pegasos algorithm (Shalev-Shwartz et al., 2011), etc.

In parallel, there has also been work on non-convex losses. The non-convex ramp loss is defined in (Collobert et al., 2006) for binary classification with the aim to avoid that examples with a higher hinge loss become support vectors. In this way, a sparser model is created which has computational benefits. They use the concave-convex procedure (CCCP) (Yuille and Ran-

garajan, 2003) to optimize the non-convex loss. In order to improve the training time, an online version of SVM with the ramp loss is presented in (Wang and Vucetic, 2009), which also uses the CCCP with a strategy of an efficient working set selection. With the similar aim, Ertekin et al. (2011) use the CCCP in combination with procedures from the LaSVM solver (Bordes et al., 2005) to get an online solver with a significantly lower training and classification time. This area continues to be developed with other approaches and move in different directions such as introducing a smooth ramp loss (Wang et al., 2008), using linear programming for ramp loss SVMs (Brooks, 2011), introducing heuristics to handle a mixed integer non-linear optimization for the ramp loss on larger datasets (Carrizosa et al., 2014), etc.

In addition to a growing interest in the ramp loss in binary classification, there has been research, in parallel, in the corresponding methods in the structured case. The paper (Do et al., 2008) introduces a structured version of the non-convex loss. In difference to the binary case, in the structured case we do not have a direct reduction of support vectors and a significant decrease in runtime. The presented advantage of the non-convex structured loss in Do et al. (2008) happens in the scenario in which the labels are noisy and when for each example there is a large set of labels that are (almost) as good as the label in the training set. This better performance of the structured ramp loss is due to the fact that it is upper bounded and that the noisy examples can be discarded when the error is too large, as opposite to the unbounded structured hinge loss. Next, three different types of structured ramp losses are presented in Gimpel and Smith (2012), which are successfully applied to problems in machine translation. The algorithm uses a combination of the CCCP and the stochastic sub-gradient descent for the parameter optimization.

In this chapter, we present a sequential method for the optimization of one of the formulations of the structured ramp loss in dual space presented in (Mančev, 2015). After the application of the CCCP, the convex problem is optimized in dual space by sequentially traversing through examples one at a time, in a similar fashion as the sequential dual method presented in (Balamurugan et al., 2011). We present experimental results on two sequence labeling problems and also check the behavior of the structured ramp loss on noisy data.

5.2 Definition and characteristics of the structured ramp loss

We will start from minimizing the regularized empirical risk (2.3) over the training set $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$ as

$$\min_{\mathbf{w}} J(\mathbf{w}) = \min_{\mathbf{w}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{n=1}^{N} \ell_n(\mathbf{w}), \tag{5.1}$$

where $\ell_n(\mathbf{w})$ represents a loss function on the nth example with parameters $\mathbf{w}$, and let us denote $C = 1/(\lambda N)$ as in (2.8). In the case of the structured ramp loss, $\ell_n(\mathbf{w})$ is defined as

$$\begin{aligned}\ell_n^{\text{Ramp}}(\mathbf{w}) &= \ell_n^{\text{MM}}(\mathbf{w}) + \ell_n^{\text{C}}(\mathbf{w}) \\ &= \max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) - \max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) \\ &= \ell(\mathbf{w};(\boldsymbol{x}^n,\widetilde{\boldsymbol{y}}^n_{\mathbf{w}})) - \ell_C(\mathbf{w};(\boldsymbol{x}^n,\overline{\boldsymbol{y}}^n_{\mathbf{w}}))\,,\end{aligned} \tag{5.2}$$

where[1]

$$\ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) = \max\{0, -L(\boldsymbol{y}^n,\boldsymbol{y}) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y})\},\ \ell_n^{\text{C}}(\mathbf{w}) = -\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y}))\,,$$
$$\ell(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) = \max\{0, L(\boldsymbol{y}^n,\boldsymbol{y}) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y})\},\ \ell_n^{\text{MM}}(\mathbf{w}) = \max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})).$$

The optimal structures $\widetilde{\boldsymbol{y}}^n_{\mathbf{w}}$ and $\overline{\boldsymbol{y}}^n_{\mathbf{w}}$, in which the parameters $\mathbf{w}$ can be omitted when there is no confusion, are defined as

$$\widetilde{\boldsymbol{y}}^n \equiv \widetilde{\boldsymbol{y}}^n_{\mathbf{w}} = \arg\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})), \tag{5.3}$$

$$\overline{\boldsymbol{y}}^n \equiv \overline{\boldsymbol{y}}^n_{\mathbf{w}} = \arg\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y}))\,. \tag{5.4}$$

When introducing the structured ramp loss, Do et al. (2008) had the aim to improve the results on the data where the labels are noisy, and in the case when there are a lot of structures which are as good as the original one. This relies on some important characteristics which can be written for the ramp loss. First, from the definition of the ramp loss, we can see its relation to the hinge loss

$$\ell_n^{\text{Ramp}}(\mathbf{w}) \le \ell_n^{\text{MM}}(\mathbf{w}),$$

because $\ell_n^{\text{C}}(\mathbf{w}) \le 0$. Next, the ramp loss cannot be increased without the bound, as it is case with the hinge loss, because the following inequality holds

$$2L(\boldsymbol{y}^n,\overline{\boldsymbol{y}}^n) \le \ell_n^{\text{Ramp}}(\mathbf{w}) \le 2L(\boldsymbol{y}^n,\widetilde{\boldsymbol{y}}^n). \tag{5.5}$$

This can be easily seen by embedding structures $\overline{\boldsymbol{y}}^n$ and $\widetilde{\boldsymbol{y}}^n$ into the ramp loss definition

$$\begin{aligned}\ell_n^{\text{Ramp}}(\mathbf{w}) &= L(\boldsymbol{y}^n,\widetilde{\boldsymbol{y}}^n) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\widetilde{\boldsymbol{y}}^n) + \min_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \{L(\boldsymbol{y}^n,\boldsymbol{y}) + \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y})\} \\ &\le L(\boldsymbol{y}^n,\widetilde{\boldsymbol{y}}^n) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\widetilde{\boldsymbol{y}}^n) + L(\boldsymbol{y}^n,\widetilde{\boldsymbol{y}}^n) + \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\widetilde{\boldsymbol{y}}^n) \\ &= 2L(\boldsymbol{y}^n,\widetilde{\boldsymbol{y}}^n)\end{aligned}$$

[1]Note that this is only one of possible definition of function $\ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y}))$ and the ramp loss, which is discussed in (Gimpel, 2012). In the Do et al. (2008) version we have $\ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) = \max\{0, -\mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y})\}$.

and, similarly,

$$\begin{aligned}\ell_n^{\text{Ramp}}(\mathbf{w}) &= \max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)}\{L(\boldsymbol{y}^n,\boldsymbol{y}) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y})\} + L(\boldsymbol{y}^n,\overline{\boldsymbol{y}}^n) + \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\overline{\boldsymbol{y}}^n)\\ &\geq L(\boldsymbol{y}^n,\overline{\boldsymbol{y}}^n) - \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\overline{\boldsymbol{y}}^n) + L(\boldsymbol{y}^n,\overline{\boldsymbol{y}}^n) + \mathbf{w}^\mathsf{T}\Delta\mathbf{F}_n(\overline{\boldsymbol{y}}^n)\\ &= 2L(\boldsymbol{y}^n,\overline{\boldsymbol{y}}^n).\end{aligned}$$

Figure 5.1 presents an illustration of the ramp loss in a binary and a structured case. In the binary case, when the absolute value of classifier's score $m(\mathbf{w})$ is greater than one, these examples do not become support vectors, which allows us to create a more sparse and robust model as it was done in (Wang and Vucetic, 2009). On the other hand, in the structured case we have many structures inside one example, so for each $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$, we have a corresponding convex and concave function. However, in difference to the binary case, the structured ramp loss is obtained by adding convex and concave losses of different structures found during the inference with current parameters $\mathbf{w}$. So, the structured ramp loss is dependent on two structures defined with current parameters and we cannot easily apply a removal of structures similar to Wang and Vucetic (2009) for the binary case in order to create a sparse model. However, every point for the ramp loss will satisfy (5.5), which can help the model to become resistant to structures with noisy labels.

5.3 Primal-dual problem after the CCCP application on the ramp loss

With the ramp loss, we have the following optimization problem:

$$\min_{\mathbf{w}}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\ell_n^{\text{MM}}(\mathbf{w}) + C\sum_{n=1}^{N}\ell_n^{\text{C}}(\mathbf{w})\right\}, \tag{5.6}$$

which we can optimize using the *concave-convex procedure* (CCCP) introduced in (Yuille and Rangarajan, 2003). The CCCP is an iterative optimization procedure which allows us to find the optimum of a function which can be represented as a sum of the convex and the concave part. At each iteration, it approximates the concave function with its first order Taylor expansion at current parameters, and sets the parameters for the next iteration as a solution of the sum of the convex and the approximate concave part.

Let us suppose that we have a loss function $J(\mathbf{w})$, which can be represented as a sum of the convex $J_{\text{vex}}(\mathbf{w})$ and the concave part $J_{\text{cav}}(\mathbf{w})$. Using the CCCP, we get series of following optimization problems

$$\mathbf{w}^{t+1} = \arg\min_{\mathbf{w}} J_t(\mathbf{w},\mathbf{w}^t) = \arg\min_{\mathbf{w}}\left\{J_{\text{vex}}(\mathbf{w}) + \mathbf{w}^\mathsf{T}J'_{\text{cav}}(\mathbf{w}^t)\right\}.$$

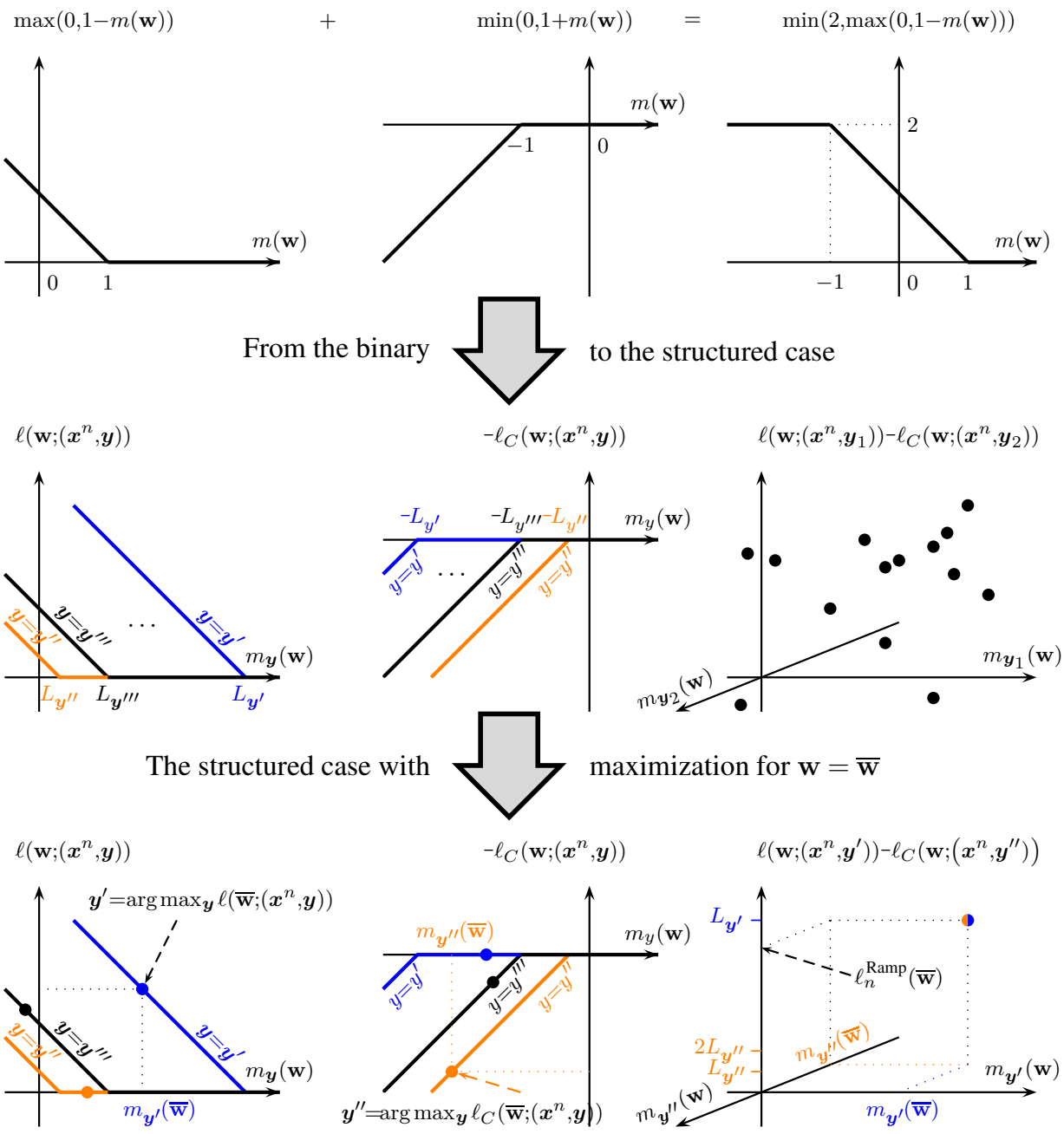

Figure 5.1: The ramp loss in the binary and the structured case. In the binary case $m(\mathbf{w}) = t(\mathbf{w}^T\boldsymbol{\phi}(\boldsymbol{x})+b)$, where $t \in \{-1,+1\}$ denotes a binary label, b represents a bias threshold, while $\boldsymbol{\phi}$ is a nonlinear mapping from the input space to the feature space. In the structured case we denote $m_{\boldsymbol{y}}(\mathbf{w}) = \mathbf{w}^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y})$ and $L_{\boldsymbol{y}} = L(\boldsymbol{y}^n,\boldsymbol{y})$. The first two panels in the middle represent the dependence of the hinge and the concave loss on the scores $m_{\boldsymbol{y}}(\mathbf{w})$ for different structures $\boldsymbol{y}$ on the nth example, respectively, followed by the representation of the sum of these losses for two arbitrarily chosen structures $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$. The three bottom panels show the losses with certain parameters $\overline{\mathbf{w}}$. In that case, the ramp loss is created by the addition of the hinge and the concave loss for structures $\boldsymbol{y}'$ and $\boldsymbol{y}''$, respectively, which are found through the maximization with parameters $\overline{\mathbf{w}}$. Then, in the bottom right panel, the ramp loss is between $2L_{\boldsymbol{y}''}$ and $2L_{\boldsymbol{y}'}$, i.e. the inequality $2L_{\boldsymbol{y}''} \leq \ell_n^{\text{Ramp}}(\overline{\mathbf{w}}) \leq 2L_{\boldsymbol{y}'}$ holds.

Let us define the *concave violation set* as

$$\mathcal{A}_{\mathbf{w}^t} = \{n : \ell_C(\mathbf{w}^t; (\boldsymbol{x}^n, \overline{\boldsymbol{y}}^n_{\mathbf{w}^t})) > 0\}.$$

In the case of the ramp loss, the previous CCCP iterations become

$$\mathbf{w}^{t+1} = \arg\min_{\mathbf{w}} \Bigg\{ \underbrace{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \ell_n^{\text{MM}}(\mathbf{w})}_{J_{\text{vex}}(\mathbf{w})} + \mathbf{w}^{\mathsf{T}} \underbrace{C\sum_{n=1}^{N} \partial_{\mathbf{w}} \ell_n^{\text{C}}(\mathbf{w}^t)}_{J'_{\text{cav}}(\mathbf{w}^t)} \Bigg\}, \tag{5.7}$$

where $\partial_{\mathbf{w}} \ell_n^{\text{C}}(\mathbf{w}^t) = \Delta\mathbf{F}_n(\overline{\boldsymbol{y}}^n_{\mathbf{w}^t}) \mathbb{1}[n \in \mathcal{A}_{\mathbf{w}^t}]$, $\mathbb{1}$ is the indicator function with values zero (one) if its argument is false (true) and $\overline{\boldsymbol{y}}^n_{\mathbf{w}^t}$ is calculated using parameters $\mathbf{w}^t$. Each minimization problem (5.7) can be optimized using primal sub-gradient methods, such as the structured Pegasos algorithm (Shalev-Shwartz et al., 2011), or we can transform it into a constraint optimization problem

$$\mathbf{w}^{t+1} = \arg\min_{\mathbf{w}} J_t(\mathbf{w}, \mathbf{w}^t) = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \xi_n + C \sum_{n \in \mathcal{A}_{\mathbf{w}^t}} \mathbf{w}^{\mathsf{T}} \Delta\mathbf{F}_n(\overline{\boldsymbol{y}}^n_{\mathbf{w}^t}) \tag{5.8}$$

$$\text{s.t.} \quad \mathbf{w}^{\mathsf{T}} \Delta\mathbf{F}_n(\boldsymbol{y}) \geq L(\boldsymbol{y}^n, \boldsymbol{y}) - \xi_n, \quad \forall n, \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n). \tag{5.9}$$

By introducing Lagrange multipliers $\lambda_{n,\boldsymbol{y}} \geq 0$ for each structure, we get the Lagrange function

$$\mathcal{L}(\boldsymbol{\lambda}, \mathbf{w}, \mathbf{w}^t) = J_t(\mathbf{w}, \mathbf{w}^t) - \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \lambda_{n,\boldsymbol{y}} \left(\mathbf{w}^{\mathsf{T}} \Delta\mathbf{F}_n(\boldsymbol{y}) - L(\boldsymbol{y}^n, \boldsymbol{y}) + \xi_n \right).$$

From the KKT conditions (Kuhn and Tucker, 1951), we get that at optimum the following must be satisfied

$$\mathbf{w} = \mathbf{u} - \mathbf{v}, \quad \mathbf{u} = \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \lambda_{n,\boldsymbol{y}} \Delta\mathbf{F}_n(\boldsymbol{y}), \quad \mathbf{v} = C \sum_{n \in \mathcal{A}_{\mathbf{w}^t}} \Delta\mathbf{F}_n(\overline{\boldsymbol{y}}^n_{\mathbf{w}^t});$$

$$\sum_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \lambda_{n,\boldsymbol{y}} = C, \ \forall n; \quad \lambda_{n,\boldsymbol{y}} \left(\mathbf{w}^{\mathsf{T}} \Delta\mathbf{F}_n(\boldsymbol{y}) - L(\boldsymbol{y}^n, \boldsymbol{y}) + \xi_n \right) = 0, \ \forall n, \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n).$$

Transforming the primal optimization problem into an equivalent dual one, we get

$$\min_{\boldsymbol{\lambda}} \frac{1}{2} \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{K} \boldsymbol{\lambda} - \boldsymbol{\lambda}^{\mathsf{T}} \mathbf{L} - \boldsymbol{\lambda}^{\mathsf{T}} \Delta\mathbf{F}^{\mathsf{T}} \mathbf{v} \tag{5.10}$$

$$\text{s.t.} \sum_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \lambda_{n,\boldsymbol{y}} = C, \ \forall n, \ \lambda_{n,\boldsymbol{y}} \geq 0, \forall n, \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n), \tag{5.11}$$

where $\boldsymbol{K}$ is a kernel matrix defined with elements $K_{n,\boldsymbol{y},m,\boldsymbol{y}'} = \Delta\mathbf{F}_n(\boldsymbol{y})^{\mathsf{T}} \Delta\mathbf{F}_m(\boldsymbol{y}')$ for every $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$, $\boldsymbol{y}' \in \mathscr{Y}(\boldsymbol{x}^m)$ and $\mathbf{L}$ is a vector with elements $L_{n,\boldsymbol{y}} = L(\boldsymbol{y}^n, \boldsymbol{y})$, where $n, m \in$

$\{1,\ldots,N\}$.

According to the constraints (5.11), we can restrict the optimization to only one example. Using a similar technique as described in Taskar (2004), we define $\boldsymbol{\alpha}_n$ which will represent the changes in parameters $\boldsymbol{\lambda}$ on the nth example. Thus we will have changes on the nth example described as $\lambda'_{n,\boldsymbol{y}} \leftarrow \lambda_{n,\boldsymbol{y}} + \alpha_{n,\boldsymbol{y}}$. After these changes, the new parameters should be feasible, so they must satisfy $\lambda'_{n,\boldsymbol{y}} \geq 0$, while the sum of $\alpha_{n,\boldsymbol{y}}$ should be zero. Transforming the problem (5.10)-(5.11) by dropping all terms that do not depend on $\boldsymbol{\alpha}_n$, we get the optimization restricted to the nth example

$$\min_{\boldsymbol{\alpha}_n} D_{\boldsymbol{\lambda}_n}(\boldsymbol{\alpha}_n) = \min_{\boldsymbol{\alpha}_n} \frac{1}{2}\boldsymbol{\alpha}_n^{\mathsf{T}}\boldsymbol{K}_n\boldsymbol{\alpha}_n - \boldsymbol{\alpha}_n^{\mathsf{T}}(\mathbf{L}_n - \Delta\mathbf{F}_n^{\mathsf{T}}\mathbf{w}) \tag{5.12}$$

$$\text{s.t.} \sum_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \alpha_{n,\boldsymbol{y}} = 0, \quad \lambda_{n,\boldsymbol{y}} + \alpha_{n,\boldsymbol{y}} \geq 0,\ \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n), \tag{5.13}$$

where $\Delta\mathbf{F}_n = [\Delta\mathbf{F}_n(\boldsymbol{y})]_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)}$, and $\boldsymbol{K}_n = \Delta\mathbf{F}_n^{\mathsf{T}}\Delta\mathbf{F}_n$ is a kernel matrix for the nth example. The parameter updates on the nth example can be represented as

$$\mathbf{w}' = \mathbf{w} + \Delta\mathbf{F}_n\boldsymbol{\alpha}_n.$$

The gradient of the dual function $D_{\boldsymbol{\lambda}_n}(\boldsymbol{\alpha}_n)$ with respect to $\boldsymbol{\alpha}_n$ is

$$\boldsymbol{g}_n = \mathbf{K}_n\boldsymbol{\alpha}_n - \mathbf{L}_n + \Delta\mathbf{F}_n^{\mathsf{T}}\mathbf{w} \tag{5.14}$$

with elements $g_{n,\boldsymbol{y}}$, which is used to express the violation of Karush-Kuhn-Tucker (KKT) conditions (Kuhn and Tucker, 1951) for the problem (5.12)-(5.13) as

$$\max_{\boldsymbol{y}\in I_0} g_{n,\boldsymbol{y}} > \min_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} g_{n,\boldsymbol{y}}, \tag{5.15}$$

where $I_0 = \{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n) : \alpha_{n,\boldsymbol{y}} > -\lambda_{n,\boldsymbol{y}}\}$. Since all elements of $\boldsymbol{\alpha}_n$ outside the working set are equal to zero, i.e. $\alpha_{n,\boldsymbol{y}} = 0$, $\forall \boldsymbol{y} \in \mathscr{W}_n$, the elements of the gradient can be represented in the following form

$$g_{n,\boldsymbol{y}} = \sum_{\boldsymbol{z}\in\mathscr{W}_n} \alpha_{n,\boldsymbol{z}} K_{n,\boldsymbol{y},\boldsymbol{z}} - L(\boldsymbol{y}^n,\boldsymbol{y}) + \mathbf{w}^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}), \forall \boldsymbol{y} \in \mathscr{W}_n. \tag{5.16}$$

First, we need to find a sequence which minimizes the gradient of dual function (5.14). When we start processing the nth example, all parameters $\alpha_{n,\boldsymbol{y}}$ are equal to zero and we have that

$$\arg\min_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} -L(\boldsymbol{y}^n,\boldsymbol{y}) + \mathbf{w}^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}) = \arg\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) = \widetilde{\boldsymbol{y}}^n. \tag{5.17}$$

From the previous formula, we see that structure $\widetilde{\boldsymbol{y}}^n$ can be found by applying the standard augmented Viterbi decoding using the parameters $\mathbf{w} = \mathbf{u} - \mathbf{v}$. Once the structure is found, the working set of the nth example $\mathscr{W}_n = \{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n) \,:\, \lambda_{n,\boldsymbol{y}} > 0\}$ is incrementally built by adding the structure $\widetilde{\boldsymbol{y}}^n$ according to (5.17), in a similar fashion as described in (Balamurugan et al., 2011). After that, we can rewrite the KKT condition violation (5.15) with precision τ restricted to the set $\mathscr{W}_n$

$$g_{n,\boldsymbol{y}''} > g_{n,\boldsymbol{y}'} + \tau, \tag{5.18}$$

$$\boldsymbol{y}' = \arg\min_{\boldsymbol{y} \in \mathscr{W}_n} g_{n,\boldsymbol{y}}, \quad \boldsymbol{y}'' = \arg\max_{\boldsymbol{y} \in I_0'} g_{n,\boldsymbol{y}}, \tag{5.19}$$

where $I_0' = \{\boldsymbol{y} \in \mathscr{W}_n : \alpha_{n,\boldsymbol{y}} > -\lambda_{n,\boldsymbol{y}}\}$. Note that when we start processing the nth example, $\boldsymbol{y}'$ would be equal to $\widetilde{\boldsymbol{y}}^n$, which will not be true for further iterations on the nth example, since we extend the working set $\mathscr{W}_n$ only when we start processing the nth example. The parameters inside each set $\mathscr{W}_n$ will be optimized using the sequential minimal optimization (Platt, 1999), where the step size is derived further.

Step size Let $\boldsymbol{h}$ be a vector where $h_{\boldsymbol{y}'} = 1$, $h_{\boldsymbol{y}''} = -1$ and all other elements are equal to zero. We can write

$$D_{\boldsymbol{\lambda}_n}(\boldsymbol{\alpha}_n + \tilde{\delta}\boldsymbol{h}) = D_{\boldsymbol{\lambda}_n}(\alpha_n) + \boldsymbol{h}^\mathsf{T}\boldsymbol{g}_n\tilde{\delta} + \frac{\tilde{\delta}^2}{2}\boldsymbol{h}^\mathsf{T}\nabla\boldsymbol{g}_n\boldsymbol{h},$$

which is reduced to

$$D_{\boldsymbol{\lambda}_n}(\boldsymbol{\alpha}_n + \tilde{\delta}\boldsymbol{h}) = D_{\boldsymbol{\lambda}_n}(\boldsymbol{\alpha}_n) + \tilde{\delta}(g_{n,\boldsymbol{y}'} - g_{n,\boldsymbol{y}''}) + \frac{\tilde{\delta}^2}{2}\left\|\Delta\mathbf{F}_n(\boldsymbol{y}') - \Delta\mathbf{F}_n(\boldsymbol{y}'')\right\|^2,$$

because of the definition of $\boldsymbol{h}$. We seek $\tilde{\delta}$ which minimizes $D_{\boldsymbol{\lambda}_n}(\boldsymbol{\alpha}_n + \tilde{\delta}\boldsymbol{h})$ and thus we set

$$0 = \frac{\partial}{\partial\tilde{\delta}} D_{\boldsymbol{\lambda}_n}(\boldsymbol{\alpha}_n + \tilde{\delta}\boldsymbol{h}) = g_{n,\boldsymbol{y}'} - g_{n,\boldsymbol{y}''} + \tilde{\delta}\left\|\Delta\mathbf{F}_n(\boldsymbol{y}') - \Delta\mathbf{F}_n(\boldsymbol{y}'')\right\|^2$$

and get the unbounded step as

$$\tilde{\delta} = \frac{g_{n,\boldsymbol{y}''} - g_{n,\boldsymbol{y}'}}{\left\|\Delta\mathbf{F}_n(\boldsymbol{y}') - \Delta\mathbf{F}_n(\boldsymbol{y}'')\right\|^2}.$$

Now, we want $\tilde{\delta}$ to be bounded to δ in order for the change in parameters to be feasible. At that time all parameters are feasible, and we want to make a change

$$\alpha_{n,\boldsymbol{y}'}^{new} = \alpha_{n,\boldsymbol{y}'} + \delta, \qquad \alpha_{n,\boldsymbol{y}''}^{new} = \alpha_{n,\boldsymbol{y}''} - \delta, \tag{5.20}$$

that must satisfy that new $\boldsymbol{\lambda}_n$ is also feasible, which means that

$$\lambda_{n,\boldsymbol{y}} + \alpha_{n,\boldsymbol{y}}^{new} \geq 0, \quad \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n). \tag{5.21}$$

From (5.20) and (5.21), we get a step which must satisfy the following conditions

$$\lambda_{n,\boldsymbol{y}'} + \alpha_{n,\boldsymbol{y}'} + \delta \geq 0, \quad \lambda_{n,\boldsymbol{y}''} + \alpha_{n,\boldsymbol{y}''} - \delta \geq 0,$$

which gets us the upper and lower limit for the step size

$$-\lambda_{n,\boldsymbol{y}'} - \alpha_{n,\boldsymbol{y}'} \leq \delta \leq \lambda_{n,\boldsymbol{y}''} + \alpha_{n,\boldsymbol{y}''},$$

and the final step as

$$\delta = \max\left(-\lambda_{n,\boldsymbol{y}'} - \alpha_{n,\boldsymbol{y}'}, \min\left(\lambda_{n,\boldsymbol{y}''} + \alpha_{n,\boldsymbol{y}''}, \frac{g_{n,\boldsymbol{y}''} - g_{n,\boldsymbol{y}'}}{\|\Delta\mathbf{F}_n(\boldsymbol{y}') - \Delta\mathbf{F}_n(\boldsymbol{y}'')\|^2}\right)\right).$$

$$\overline{\boldsymbol{y}}_{\mathbf{0}}^n = \arg\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell_C(\mathbf{0};(\boldsymbol{x}^n,\boldsymbol{y})) = \arg\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} -L(\boldsymbol{y}^n,\boldsymbol{y}) = \boldsymbol{y}^n$$

and, thus, the first CCCP iteration will represent the optimization of the hinge loss. In this way, during the first epoch the parameters have been initialized with the hinge loss for the next epochs. Note that this initialization is due to the definition of the ramp loss, and with a different formulation, as presented by Do et al. (2008), the previous will not hold and in that case the parameters must be explicitly initialized.

5.4 Experimental results

In this section, we present a comparison of the ramp loss and the hinge loss. First, we will make a comparison of two real sequence labeling problems, the shallow parsing (Tjong Kim Sang and Buchholz, 2000) on the CONLL-2000 corpus[2] and the part-of-speech (POS) tagging on the Brown corpus[3]. After that, we will present the results obtained on artificial data, where the data is created by modifying the corpus for sequence labeling by adding outlayers in different percentages.

Compared algorithms and notation We will compare the sequential dual method (SDM) with the ramp loss and the SDM with the hinge loss (Balamurugan et al., 2011). In both versions, we do not use any additional heuristics to control the growth of the working set. We also include a structured Pegasos algorithm (Shalev-Shwartz et al., 2011) for comparison.

[2] http://www.cnts.ua.ac.be/conll2000/chunking

[3] http://khnt.aksis.uib.no/icame/manuals/brown/

Algorithm 6: Sequential dual method for structured ramp loss

Input : Training data: $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$, parameter $C \in \mathbb{R}^+$
Number of epochs: P, Number of CCCP iterations: T
Output: Model parameters: $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{0}$; $\boldsymbol{\lambda} \leftarrow \mathbf{0}$;
$\mathscr{W}_n \leftarrow \{\boldsymbol{y}^n\}$, $\lambda_{n,\boldsymbol{y}^n} \leftarrow C$, $\forall n = 1, \ldots, N$;
for $p \leftarrow 1$ **to** P **do**
  $\mathcal{A} \leftarrow \emptyset$;
  for $n \leftarrow 1$ **to** N **do**
    $\overline{\boldsymbol{y}}^n \leftarrow \arg\max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \ell_C(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$;
    if $\ell_C(\mathbf{w}; (\boldsymbol{x}^n, \overline{\boldsymbol{y}}^n)) > 0$ **then**
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{n\}$;
    end
  end
  $\mathbf{v} \leftarrow C \sum_{n \in \mathcal{A}} \Delta\mathbf{F}_n(\overline{\boldsymbol{y}}^n)$;
  $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{v}$;
  for $t \leftarrow 1$ **to** T **do**
    for $n \leftarrow 1$ **to** N **do**
      $\widetilde{\boldsymbol{y}}^n \leftarrow \arg\max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$;
      $\mathscr{W}_n \leftarrow \mathscr{W}_n \cup \{\widetilde{\boldsymbol{y}}^n\}$;
      SMO(n, $\mathscr{W}_n$, $\mathbf{w}$, $\boldsymbol{\lambda}$);
    end
  end
end

Procedure SMO(n, $\mathscr{W}_n$, $\mathbf{w}$, $\boldsymbol{\lambda}$)

Local constant: KKT tolerance τ

$\boldsymbol{\alpha} \leftarrow \mathbf{0}$;
repeat /* SMO over the set $\mathscr{W}_n$ */
  kkt_satisfied $\leftarrow$ false;
  $K_{n,\boldsymbol{y}',\boldsymbol{y}''} \leftarrow \Delta\mathbf{F}_n(y')^\mathsf{T} \Delta\mathbf{F}_n(y''), \forall \boldsymbol{y}', \boldsymbol{y}'' \in \mathscr{W}_n$; /* kernel matrix */
  $g_{n,\boldsymbol{y}} \leftarrow \sum_{\boldsymbol{z} \in \mathscr{W}_n} \alpha_{n,\boldsymbol{z}} K_{n,\boldsymbol{y},\boldsymbol{z}} - L(\boldsymbol{y}^n, \boldsymbol{y}) + \mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y}), \forall \boldsymbol{y} \in \mathscr{W}_n$; /* gradient */
  $\boldsymbol{y}' \leftarrow \arg\min_{\boldsymbol{y} \in \mathscr{W}_n} g_{n,\boldsymbol{y}}$;
  $\boldsymbol{y}'' \leftarrow \arg\max_{\boldsymbol{y} \in \mathscr{W}_n : \alpha_{n,\boldsymbol{y}} > -\lambda_{n,\boldsymbol{y}}} g_{n,\boldsymbol{y}}$;
  if $g_{n,\boldsymbol{y}''} > g_{n,\boldsymbol{y}'} + \tau$ **then** /* if KKT cond. is not satisfied */
    $\delta_{n,\boldsymbol{y}',\boldsymbol{y}''} \leftarrow (g_{n,\boldsymbol{y}''} - g_{n,\boldsymbol{y}'}) / (K_{n,\boldsymbol{y}',\boldsymbol{y}'} - 2K_{n,\boldsymbol{y}',\boldsymbol{y}''} + K_{n,\boldsymbol{y}'',\boldsymbol{y}''})$;
    $\delta \leftarrow \max\left(-\lambda_{n,\boldsymbol{y}'} - \alpha_{n,\boldsymbol{y}'}, \min\left(\lambda_{n,\boldsymbol{y}''} + \alpha_{n,\boldsymbol{y}''}, \delta_{n,\boldsymbol{y}',\boldsymbol{y}''}\right)\right)$; /* bounded step */
    $\alpha_{n,\boldsymbol{y}'} \leftarrow \alpha_{n,\boldsymbol{y}'} + \delta$; $\alpha_{n,\boldsymbol{y}''} \leftarrow \alpha_{n,\boldsymbol{y}''} - \delta$
  else
    kkt_satisfied $\leftarrow$ true;
  end
until not kkt_satisfied;
$\lambda_{n,\boldsymbol{y}} \leftarrow \lambda_{n,\boldsymbol{y}} + \alpha_{n,\boldsymbol{y}}, \forall \boldsymbol{y} \in \mathscr{W}_n$; /* change dual parameters */
$\mathbf{w} \leftarrow \mathbf{w} + \sum_{\boldsymbol{y} \in \mathscr{W}_n} \alpha_{n,\boldsymbol{y}} \Delta\mathbf{F}_n(\boldsymbol{y})$; /* change primal parameters */
$\mathscr{W}_n \leftarrow \mathscr{W}_n \setminus \{\boldsymbol{y} : \lambda_{n,\boldsymbol{y}} = 0\}$; /* remove inactive structures */

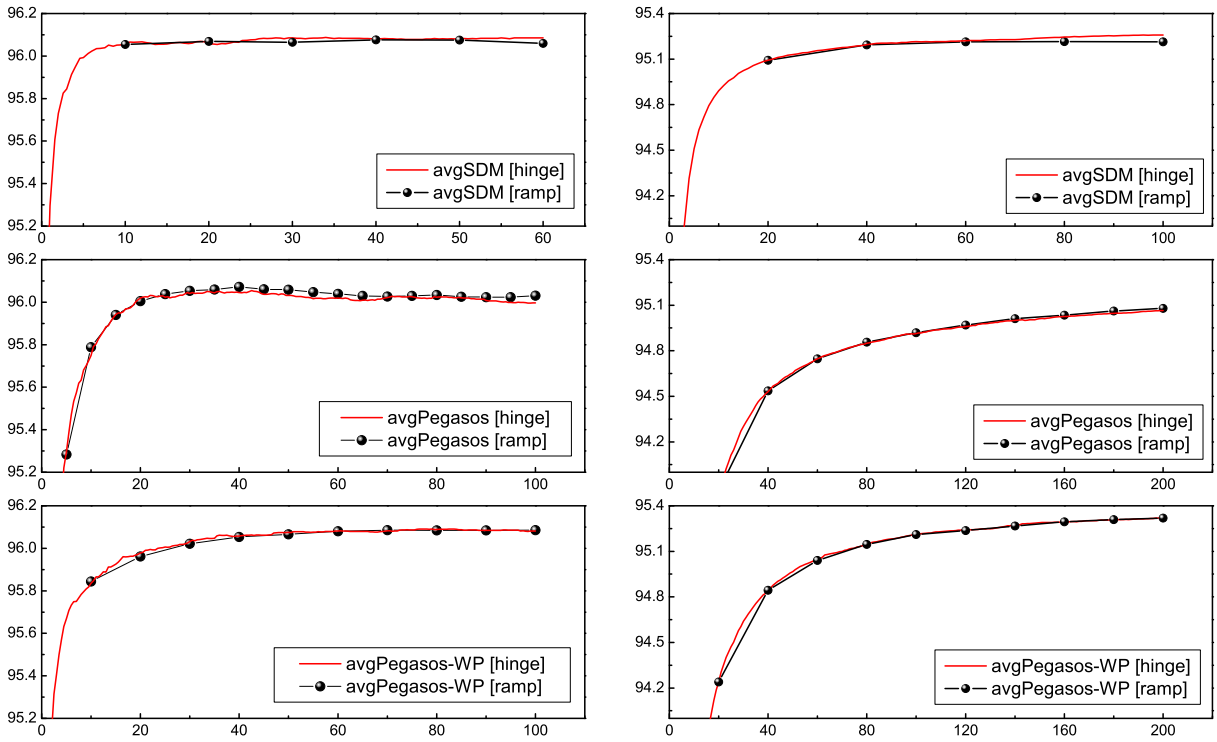

Figure 5.2: The results for the hinge and the ramp loss for the SDM and the stochastic Pegasos algorithm. The three left panels represent the results for shallow parsing, while the three panels on the right represent the results for POS tagging. The x axis represents the number of epochs for the hinge loss, and for the ramp loss it represents the total number of epochs through data (the number of outer epochs P multiplied by the number of the inner CCCP iterations T), while the y axis represents the F-measure and accuracy for the left and right panels, respectively.

To avoid oscillations during the learning process, we have applied parameter averaging for all algorithms and we will denote such algorithms using the prefix *avg*. When the learning process assumes only the addition of feature vectors multiplied by an argument, parameter averaging can be easily implemented as presented by (Collins, 2002), while in case we need to scale the feature vector, which is the case with the Pegasos algorithm, parameter averaging can be implemented using linear transformation as described by Xu (2011). With the suffix -WP, we will denote that the Pegasos algorithm is used without the optional projection step.

In order to select the regularization parameter for further experiments, we perform a cross-validation. We use a 5-fold cross-validation to find the optimal parameter for each method separately, and then we employ this optimal parameter in a test scenario. Table 5.1 provides the results on both datasets with the corresponding parameters selected via cross-validation. For the hinge loss during the cross validation, we selected the best pair of the regularization parameter and the number of training epochs up to 100 (200) for shallow parsing (Pos tagging), choosing between {10, 20, 30, 50, 100} epochs (including 200 for POS tagging). During the cross-validation for the ramp loss, we selected the number of CCCP iterations between {5, 10, 15, 20}. The number of epochs is selected in order that the product with the CCCP iterations

Table 5.1: The results for the hinge vs. the ramp loss and their corresponding parameters (regularization parameters, the number of training epochs, and the number of CCCP iterations for the ramp loss) obtained from a 5-fold cross-validation for each dataset. For all algorithms, the results are presented with averaged parameters, so the *avg* prefix is omitted in the algorithm name. The parameter $C = 1/(\lambda N)$ denotes the regularization parameter for the SDM, and λ is the regularization parameter used for Pegasos algorithms in (Shalev-Shwartz et al., 2011).

Shallow parsing					
Method	Loss	Reg.	T	# epoch	F-measure
SDM	Hinge	$C = 10^{-1}$	–	100	96.084
SDM	Ramp	$C = 10^{-1}$	10	4	96.076
Stohastic Peg	Hinge	$\lambda = 2 \cdot 10^{-3}$	–	30	96.041
Stohastic Peg	Ramp	$\lambda = 2 \cdot 10^{-3}$	5	7	96.072
Stohastic Peg-WP	Hinge	$\lambda = 10^{-3}$	–	100	96.082
Stohastic Peg-WP	Ramp	$\lambda = 10^{-3}$	10	10	96.086
Pos tagging					
Method	Loss	Reg.	T	# epoch	Accuracy
SDM	Hinge	$C = 10^{-1}$	–	200	95.292
SDM	Ramp	$C = 10^{-1}$	20	4	95.216
Stohastic Peg	Hinge	$\lambda = 10^{-4}$	–	200	95.065
Stohastic Peg	Ramp	$\lambda = 10^{-4}$	20	10	95.079
Stohastic Peg-WP	Hinge	$\lambda = 2 \cdot 10^{-3}$	–	200	95.320
Stohastic Peg-WP	Ramp	$\lambda = 2 \cdot 10^{-3}$	20	10	95.319

be up to 100 for shallow parsing (200 for POS tagging). In parallel, the dependence of results through epochs with optimal parameters is presented in Figure 5.2. Both in Table 5.1 and in Figure 5.2, we can notice that the results for the ramp and the hinge loss are similar for each algorithm on both datasets. The SDM reaches its highest results faster with a fewer numbers of epochs, while Pegasos without the projection step provides higher results on both datasets. The results through epochs are very close for the hinge and the ramp loss and we cannot see the advantage of the ramp loss on these corpora. Only the Pegasos algorithm without the projection performs a little bit better with the ramp loss on the shallow parsing problem.

Results on artificial data Since the ramp loss should be helpful with noisy examples, we will test the previous algorithms on this kind of data next. For this purpose, we modified the CONLL-2000 corpus by changing the labels of randomly chosen examples. We chose a portion of examples and changed their labels randomly. With this approach, such portions of examples became outlayers. Four artificial datasets were created, where randomly chosen portions of 5, 10, 15 and 20 percent of examples were affected by changing their labels. Figure 5.3 shows the results for the hinge versus the ramp loss for the SDM and the Pegasos algorithm depending on the percentage of outlayers added to corpus. We can see that all algorithms with the ramp

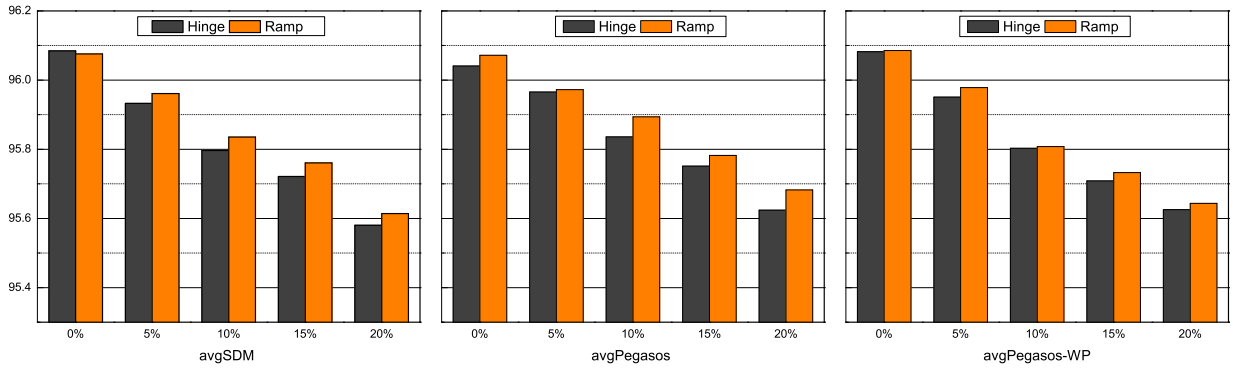

Figure 5.3: The hinge loss vs. the ramp loss for shallow parsing on a dataset with artificially generated outlayers. The y axis represents the F-measure, while the x axis represents the percentage of examples from the CONLL-2000 corpus which are changed to outlayers. The bars corresponding to 0% on the x axis indicate the results for chunking on the corpus without modification.

loss perform better than the corresponding ones with the hinge loss when the learning process includes noisy examples. This is expected as the ramp loss is upper bounded, and outlayers do not make such big changes in parameters as the hinge loss can.

The results on artificial data show us that both the SDM and the Pegasos algorithm with the ramp loss clearly show an advantage over the hinge loss, which can be useful in real problems with noisy data. On the other hand, the performance on two sequence labeling problems indicates that the algorithms perform similarly with the ramp and the hinge loss. In that case, the hinge loss has the advantage since it does not need additional decoding and it is easier for optimization.

Chapter 6

A primal sub-gradient method with the averaged sum loss

This chapter presents a primal sub-gradient method for the structured SVM optimization defined with the averaged sum of hinge losses inside each example. Compared to the mini-batch version of the Pegasos algorithm for the structured case, which deals with a single structure from each of multiple examples, our algorithm considers multiple structures from a single example in one update. This approach should increase the amount of information learned from the example. We show that the proposed version with the averaged sum loss has at least the same guarantees in terms of prediction loss as the stochastic version. Experiments have been conducted on two sequence labeling problems, shallow parsing and part-of-speech tagging, and also include a comparison to other popular sequential structured learning algorithms.

6.1 Optimization with a sub-gradient method

Through previous chapters, we consider the optimization problem (2.3) with different loss functions. If a loss function is differentiable, like it is the case with the CRF loss, we can apply the gradient decent method for the optimization. However, some loss functions are not differentiable, e.g. the structured hinge loss, where we cannot simply apply the gradient descent method. It that case, we consider an equivalent constraint optimization problem leading to structured support vector machines which were trained in a dual space by sequential minimal optimization or by using an online formulation with one constraint where the solution can be found in an analytical form as in passive-aggressive algorithms. The other option is to to compute a sub-gradient for the structured hinge loss. A *sub-gradient* $\boldsymbol{g}$ of a function $J(\mathbf{w})$ at a point $\mathbf{w}_0$ satisfies the following

$$J(\mathbf{w}) - J(\mathbf{w}_0) \geq \boldsymbol{g}^{\mathsf{T}}(\mathbf{w} - \mathbf{w}_0),$$

for every $\mathbf{w}$. If we have a convex non-differentiable function, we can apply a sub-gradient method for its optimization. The method is presented by Shor et al. (1985) and it works in a

similar fashion as a gradient descent method, which means that it changes parameters

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \eta_t \boldsymbol{g}^{(t)}$$

with a step size η_t at the tth iteration by moving them in any sub-gradient direction $\boldsymbol{g}^{(t)}$ of function J at point $\mathbf{w}^{(t)}$. However, the method does not share the property of descent, since the objective function value does not necessarily decrease with an update. If the function is differentiable at a point $\mathbf{w}^{(t)}$, then the only choice for $\boldsymbol{g}^{(t)}$ is the gradient of function J at that point, and thus in that case the sub-gradient method reduces to the standard gradient descent method with the step size η_t. Another difference between the sub-gradient and the gradient descent method is that the step size is fixed ahead of time and it is not chosen via line search as it can be the case with gradient descent methods. The usual choice is to choose a constant step size $\eta_t = c$, or a diminishing step size like $\eta_t = c/t$, $\eta_t = c/\sqrt{t}$, where $c > 0$. This method can be applied for optimization (2.3) with the structured hinge loss.

Shalev-Shwartz et al. (2011) proposed the Pegasos algorithm which takes a sub-gradient step with a predetermined step size and which can work in the mini-batch variant by choosing a set of examples and performing a sub-gradient step on it. Its structured version was successfully applied to different problems: dependency parsing (Martins et al., 2011), semantic role labeling (Lim et al., 2013), part-of-speech tagging (Ni et al., 2010), optical character recognition (Jaggi et al., 2013), named entity recognition (Lee et al., 2011). The empirical performance indicated fast convergence with the results comparable to other structured algorithms, while Ratliff et al. (2006) show that the cumulative prediction loss for the structured sub-gradient method grows only sublinearly in time.

In the following text, we shall consider the averaged sum of hinge losses over the structures inside one example and an approximate primal objective function to which the sub-gradient method is applied. Such changes in the loss function result in the fact that the algorithm can consider multiple structures inside one example (similar to the k-best variant of MIRA). For this version we provide a cumulative bound of prediction losses and perform experiments with other popular sequential structured learning algorithms.

The chapter is organized as follows: In section 6.2, we introduce the averaged sum loss for structured classifiers. After reviewing the existing version of Pegasos for the structured case, in Section 6.3 we introduce the Pegasos algorithm with the averaged sum loss. Next, we provide a theoretical analysis of the introduced algorithm, followed by implementation concerns for sparse updates and the calculation of averaged parameters. In section 6.6, we present the experiments on sequence labeling problems, testing different properties of the method and presenting a comparison with other popular structured algorithms.

6.2 Averaged sum loss

Again, we begin by considering the problem of minimizing the regularized empirical risk (2.3) over the set $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$

$$\min_{\mathbf{w}} f(\mathbf{w}) = \min_{\mathbf{w}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{n=1}^{N} \ell_n(\mathbf{w}). \tag{6.1}$$

As there are many output structures inside each example, the loss function can be defined for each one separately. Let $\ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$ represent a loss for the structure $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$ with parameters $\mathbf{w}$. Similarly as in (3.2), we consider a *hinge loss for structure* $\boldsymbol{y}$ as

$$\ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) = \max\left(0, L(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y})\right). \tag{6.2}$$

Since inside each example there are many output structures, usually we deal only with those which provide the maximum loss on the current example. As we have seen before, $\ell(\mathbf{w}; (\boldsymbol{x}^n, \widetilde{\boldsymbol{y}}^n))$ is the *max margin* (MM) *loss*[1], where $\widetilde{\boldsymbol{y}}^n$ is the 'best' structure for $\boldsymbol{x}^n$ with respect to the loss function found by (3.1).

In this chapter we will consider the *average sum* (AS) *loss* $\ell_n^{\text{AS}}(\mathbf{w})$ defined as

$$\ell_n^{\text{AS}}(\mathbf{w}) = \frac{1}{|\mathscr{Y}_{-n}|} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})), \tag{6.3}$$

which represents the expected hinge loss for structures inside the nth example. If the AS loss is used in problem (6.1), it leads to the corresponding constraint optimization problem

$$\min_{\mathbf{w}, \boldsymbol{\xi}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{n=1}^{N} \frac{1}{|\mathscr{Y}_{-n}|} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \xi_{n,\boldsymbol{y}} \tag{6.4}$$

$$\text{s.t. } \forall n, \forall \boldsymbol{y} \in \mathscr{Y}_{-n} : \mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y}) \geq L(\boldsymbol{y}^n, \boldsymbol{y}) - \xi_{n,\boldsymbol{y}}, \quad \xi_{n,\boldsymbol{y}} \geq 0, \tag{6.5}$$

where now one non-negative slack variable is assigned to each output structure. Using one slack variable per output structure inside one example can be seen as a structural generalization of the Weston and Watkins (1998) multi-class SVM, where slack variables are assigned to possible classes inside an example.

[1] In literature this loss is called the structured hinge loss (Taskar et al., 2004) or the max margin loss for the structured case (Collins et al., 2008). Even though the former name is more common, we will prefer the latter one in this chapter to avoid confusion with the hinge loss for a structure that is already defined in (6.2).

6.3 Structured Pegasos algorithms

Pegasos with the max margin loss

Pegasos is a sub-gradient method introduced in (Shalev-Shwartz et al., 2007). The algorithm on each iteration t chooses a set $A_t \subseteq \{1, \ldots, N\}$ of cardinality k. Then the objective function (6.1) is approximated with

$$f^{\text{MM}}(\mathbf{w}, A_t) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{k} \sum_{n \in A_t} \ell(\mathbf{w}; (\boldsymbol{x}^n, \widetilde{\boldsymbol{y}}^n)) \tag{6.6}$$

and optimized using the sub-gradient descent $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_t^{\text{MM}}$, with the value of the approximate objective sub-gradient

$$\nabla_t^{\text{MM}} = \lambda \mathbf{w}_t - \frac{1}{k} \sum_{n \in A_t^+} \Delta\mathbf{F}_n(\widetilde{\boldsymbol{y}}^n), \tag{6.7}$$

$A_t^+ = \{n \in A_t : \ell(\mathbf{w}_t; (\boldsymbol{x}^n, \widetilde{\boldsymbol{y}}^n)) > 0\}$, where the step size is set to $\eta_t = 1/(\lambda t)$. After each sub-gradient step, the parameters can be optionally projected on the ball of radius $1/\sqrt{\lambda}$ with the update

$$\mathbf{w}_{t+1} \leftarrow \min\left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}_{t+1}\|}\right\} \mathbf{w}_{t+1}. \tag{6.8}$$

The pseudocode is presented in Algorithm 7. In the case of $k = 1$, the update corresponds to the stochastic version, for $k = N$ this is the standard (batch) version and for $1 < k < N$ it is called the mini-batch version.

Pegasos with the averaged sum loss

Let us consider using the AS loss and approximate the objective function (6.1) with

$$\frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{|A_t|} \sum_{n \in A_t} \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})), \tag{6.9}$$

where $B_n \subseteq \mathscr{Y}_{-n}$ and $A_t \subseteq \{1, \ldots, N\}$ contains the set of examples on which the approximation is made. Further on, we will consider the previous approximation restricted only to the nth example, i.e., where we choose $A_t = \{n\}$ and define

$$f^{\text{AS}}(\mathbf{w}, B_n) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})). \tag{6.10}$$

This restriction allows us to obtain an online algorithm through examples with a mini-batch optimization inside each example according to set B_n, while the selection of B_n allows us

to choose which structures we will consider in the optimization process. We consider a subgradient of the approximate objective (6.10) given by

$$\nabla^{\text{AS}} = \boldsymbol{\lambda}\mathbf{w} - \frac{1}{|B_n|}\sum_{\boldsymbol{y}\in B_n^+}\Delta\mathbf{F}_n(\boldsymbol{y}), \tag{6.11}$$

where $B_n^+ = \{\boldsymbol{y} \in B_n : \ell(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) > 0\}$. Thus the parameter update for the structured Pegasos with the AS loss in the tth iteration on the nth example is

$$\mathbf{w}_{t+1} = (1-\eta_t\boldsymbol{\lambda})\mathbf{w}_t + \frac{\eta_t}{|B_n|}\sum_{\boldsymbol{y}\in B_n^+}\Delta\mathbf{F}_n(\boldsymbol{y}). \tag{6.12}$$

Let us define the *prediction violation set of structures*, S_n, as

$$S_n = \{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n) : \ell(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) \geq \ell(\mathbf{w};(\boldsymbol{x}^n,\widehat{\boldsymbol{y}}^n))\},$$

where the *prediction structure* $\widehat{\boldsymbol{y}}^n$ is given by

$$\widehat{\boldsymbol{y}}^n = \arg\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \mathbf{w}^{\mathsf{T}}\mathbf{F}(\boldsymbol{x}^n,\boldsymbol{y}). \tag{6.13}$$

In theoretical analysis, we will consider the version of Pegasos with the AS loss where the selection of the set B_n is not from all $\mathscr{Y}_{-n}$ structures, but only from S_n, and to such a restriction we will refer as the *restricted Pegasos* algorithm. The way we choose the set B_n from S_n of size k is not important for further analysis. Note that by choosing $B_n = \{\widetilde{\boldsymbol{y}}^n\}$ the algorithm is reduced to a stochastic Pegasos with the MM loss. Also note that it is possible to select A_t with a cardinality greater than one, and the algorithm will operate over multiple structures inside each of selected examples in one update.

Pegasos with the k-best loss Let $\mathscr{B}_{\mathbf{w}}^{k,n}$ denote a set of k structures with the highest score on the nth example, i.e., the structures which maximize the value of $\ell(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y}))$. Further, we can define the *k-best loss* as

$$\ell_n^{k\text{best}}(\mathbf{w}) = \frac{1}{k}\sum_{\boldsymbol{y}\in\mathscr{B}_{\mathbf{w}}^{k,n}}\ell(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})),$$

and the corresponding objective function restricted to the nth example as

$$f^{Best_n^k}(\mathbf{w}) = f^{\text{AS}}(\mathbf{w},\mathscr{B}_{\mathbf{w}}^{k,n}). \tag{6.14}$$

According to (6.14), we can observe the k-best objective as a special case of the AS objective approximation (6.10) which is made on the $\mathscr{B}_{\mathbf{w}}^{k,n}$ set. Also, we can see that the k-best loss lies between the MM loss and the AS loss, i.e., $\ell_n^{\text{AS}}(\mathbf{w}) \leq \ell_n^{k\text{best}}(\mathbf{w}) \leq \ell_n^{\text{MM}}(\mathbf{w})$. The k-best loss is convex (Boyd and Vandenberghe, 2004) and we can apply the Pegasos algorithm for

optimization with the sub-gradient and parameter update defined with (6.11) and (6.12) by setting $B_n = \mathscr{B}^{k,n}_{\mathbf{w}}$. If we choose B_n to be $\mathscr{B}^{k,n}_{\mathbf{w}}$, not a subset from S_n, such a version can also be called k-best Pegasos, as it works in a similar framework as k-best MIRA by Crammer et al. (2005) and it will directly optimize the k-best loss. Moreover, we can use k-best decoding to find structures from the prediction violation set. Since we need k output structures (if they exist) with the loss greater than the loss for the prediction structure, we do this by finding $\mathscr{B}^{k,n}_{\mathbf{w}}$ and removing structures which do not belong to S_n. The pseudocode is presented in Algorithm 8.

6.4 Theoretical analysis

In the structured case we care about the cumulative bound of prediction losses through the iterations between the prediction structure $\widehat{\boldsymbol{y}}^n$ and the true structure $\boldsymbol{y}^n$, i.e., the sum over $L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}^n)$. This bound for the stochastic sub-gradient method with the MM loss is given by Ratliff et al. (2006), and we will provide a bound for the restricted Pegasos with the AS loss. First we need the following lemma from (Shalev-Shwartz et al., 2011). Recall that a function f is λ-strongly convex if $f(\mathbf{w}) - \frac{\lambda}{2}\|\mathbf{w}\|^2$ is a convex function.

Lemma 2. *(Shalev-Shwartz et al., 2011) Let $f_1, \ldots, f_T$ be a sequence of λ-convex functions and D be a closed convex set. Define $\Pi_D(\mathbf{w}) = \arg\min_{\mathbf{w}' \in D} \|\mathbf{w} - \mathbf{w}'\|$. Let $\mathbf{w}_1, \ldots, \mathbf{w}_{T+1}$ be a sequence of vectors such that $\mathbf{w}_1 \in D$ and, for $t \geq 1$, $\mathbf{w}_{t+1} = \Pi_D(\mathbf{w}_t - \eta_t \nabla_t)$, where ∇_t belongs to the sub-gradient set of f_t at $\mathbf{w}_t$ and $\eta_t = \frac{1}{\lambda t}$. Assume that for all t, $\|\nabla_t\| \leq G$. Then, for all $\mathbf{u} \in D$ it follows that*

$$\frac{1}{T}\sum_{t=1}^{T} f_t(\mathbf{w}_t) \leq \frac{1}{T}\sum_{t=1}^{T} f_t(\mathbf{u}) + \frac{G^2(1+\ln T)}{2\lambda T}.$$

Theorem 4. *Let $(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$ be a sequence of examples where $\|\Delta\mathbf{F}_n(\boldsymbol{y})\| \leq R$ and $L(\boldsymbol{y}^n, \boldsymbol{y}) \leq 1$ for all $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$, $n = 1, \ldots, N$ and $\mathbf{w}^* = \arg\min_{\mathbf{w}} f(\mathbf{w})$, where $f(\mathbf{w})$ is defined with loss function $\ell^{AS}_n(\mathbf{w})$. Then, for the update (6.12) with the optional projection step (6.8) it follows that*

$$\frac{1}{N}\sum_{n=1}^{N} f^{AS}(\mathbf{w}_n, B_n) \leq \frac{1}{N}\sum_{n=1}^{N} f^{AS}(\mathbf{w}^*, B_n) + \frac{c(1+\ln N)}{2\lambda N},$$

where $c = (\sqrt{\lambda} + R)^2$ if we perform the projection step and $c = 4R^2$ otherwise.

Proof. We first show that the conditions of Lemma 2 are satisfied. Function $f^{\text{AS}}(\mathbf{w}_n, B_n)$ is a λ-convex function by definition. Further, if we use the projection step, then it follows that $\|\mathbf{w}_n\| \leq 1/\sqrt{\lambda}$ and $\|\nabla_n\| \leq \lambda + R$. In case we do not use it, with a similar technique as employed by Shalev-Shwartz et al. (2011), we get $\|\mathbf{w}_n\| \leq R/\sqrt{\lambda}$ and $\|\nabla_n\| \leq 2R$.

Next, we want to show that $\mathbf{w}^* \in D$, which is obvious if we do not use the projection. In

case we use it, then for the primal problem (6.4)-(6.5) we have the corresponding dual problem

$$\max_{\boldsymbol{\alpha}} \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \alpha_{n,\boldsymbol{y}} L_{n,\boldsymbol{y}} - \frac{1}{2} \left\| \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \alpha_{n,\boldsymbol{y}} \Delta \mathbf{F}_n(\boldsymbol{y}) \right\|^2 \tag{6.15}$$

$$\text{s.t.} \quad 0 \leq \alpha_{n,\boldsymbol{y}} \leq \frac{C}{|\mathscr{Y}_{-n}|}, \ \forall n, \forall \boldsymbol{y} \in \mathscr{Y}_{-n}, \tag{6.16}$$

where $C = 1/(\lambda N)$, $L_{n,\boldsymbol{y}}$ is an abbreviation for $L(\boldsymbol{y}^n, \boldsymbol{y})$, and the connection between primal and dual parameters is $\mathbf{w} = \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \alpha_{n,\boldsymbol{y}} \Delta \mathbf{F}_n(\boldsymbol{y})$. If $(\mathbf{w}^*, \xi^*)$ is the optimal point for the primal problem and $\boldsymbol{\alpha}^*$ is the optimum for the dual one, then from the strong duality at the optimum there is an equality between the primal and dual objective value,

$$\frac{1}{2} \|\mathbf{w}^*\|^2 \leq \frac{1}{2} \|\mathbf{w}^*\|^2 + C \sum_{n=1}^{N} \frac{1}{|\mathscr{Y}_{-n}|} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \xi^*_{n,\boldsymbol{y}} = \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \alpha^*_{n,\boldsymbol{y}} L_{n,\boldsymbol{y}} - \frac{1}{2} \|\mathbf{w}^*\|^2,$$

where the first inequality is due to $\xi^*_{n,\boldsymbol{y}} \geq 0$. Now, we obtain

$$\|\mathbf{w}^*\|^2 \leq \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \alpha^*_{n,\boldsymbol{y}} L_{n,\boldsymbol{y}} \leq \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \alpha^*_{n,\boldsymbol{y}} \leq \frac{1}{\lambda}$$

and $\|\mathbf{w}^*\| \leq \sqrt{1/\lambda}$. We can now apply Lemma 2 and get the desired bound. □

Theorem 5. *Let the conditions from the previous theorem be satisfied and let B_n be chosen as $B_n \subseteq S_n$. Then it follows that*

$$\sum_{n=1}^{N} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}^n) \leq \frac{\lambda}{2} N \|\mathbf{w}^*\|^2 + \frac{c(1+\ln N)}{2\lambda} + \sum_{n=1}^{N} \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}^*; (\boldsymbol{x}^n, \boldsymbol{y})). \tag{6.17}$$

Proof. According to the definition of $f^{\text{AS}}(\mathbf{w}_n, B_n)$, from the previous theorem we have

$$\frac{\lambda}{2N} \sum_{n=1}^{N} \|\mathbf{w}_n\|^2 + \frac{1}{N} \sum_{n=1}^{N} \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}_n; (\boldsymbol{x}^n, \boldsymbol{y})) \leq$$

$$\frac{\lambda}{2} \|\mathbf{w}^*\|^2 + \frac{1}{N} \sum_{n=1}^{N} \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}^*; (\boldsymbol{x}^n, \boldsymbol{y})) + \frac{c(1+\ln N)}{2\lambda N}. \tag{6.18}$$

Using the definition of $\widehat{\boldsymbol{y}}^n$ it follows that

$$\mathbf{w}^{\mathsf{T}} \mathbf{F}(\boldsymbol{x}^n, \widehat{\boldsymbol{y}}^n) \geq \mathbf{w}^{\mathsf{T}} \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}), \quad \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n), \tag{6.19}$$

which leads to $\mathbf{w}^{\mathsf{T}}\Delta\mathbf{F}_n(\widehat{\boldsymbol{y}}_n) \leq 0$. Therefore,

$$L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}^n) \leq \ell(\mathbf{w}; (\boldsymbol{x}^n, \widehat{\boldsymbol{y}}^n)) \leq \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})), \quad \forall \boldsymbol{y} \in B_n,$$

where the last inequality follows since B_n is a subset from S_n. Now, we have

$$\sum_{n=1}^{N} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}^n) \leq \sum_{n=1}^{N} \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}_n; (\boldsymbol{x}^n, \boldsymbol{y})) \leq \frac{\lambda}{2} \sum_{n=1}^{N} \|\mathbf{w}_n\|^2 + \sum_{n=1}^{N} \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}_n; (\boldsymbol{x}^n, \boldsymbol{y})),$$

which in combination with (6.18) provides the desired bound. □

From the previous theorem and using the inequality

$$\frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}^*; (\boldsymbol{x}^n, \boldsymbol{y})) \leq \ell_n^{\text{MM}}(\mathbf{w}^*), \quad \forall B_n \subseteq \mathscr{Y}_{-n}, \tag{6.20}$$

we get the following corollary.

Corollary 2. *Let the conditions from the previous theorem be satisfied. Then it follows that*

$$\sum_{n=1}^{N} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}^n) \leq \frac{\lambda}{2} N \|\mathbf{w}^*\|^2 + \sum_{n=1}^{N} \ell_n^{MM}(\mathbf{w}^*) + \frac{c(1+\ln N)}{2\lambda}, \tag{6.21}$$

as well as

$$\sum_{n=1}^{N} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}^n) \leq \sqrt{\frac{c(1+\ln N)}{N}}\, \|\mathbf{w}^*\| + \sum_{n=1}^{N} \ell_n^{MM}(\mathbf{w}^*),$$

by choosing

$$\lambda = \sqrt{\frac{c(1+\ln N)}{N \|\mathbf{w}^*\|^2}}.$$

If we set $B_n = \{\widetilde{\boldsymbol{y}}^n\}$ for each n, then the equality holds in (6.20) and the previous bound reduces to the bound of the stochastic version provided by Ratliff et al. (2006). For the other selection of B_n, according to the inequality (6.20), the right-hand side of (6.17) is at most the right-hand side of (6.21), so the Corollary 2 states that Pegasos with the AS loss has at most the same bound of cumulative prediction losses as the stochastic Pegasos algorithm.

The Pegasos algorithm of Shalev-Shwartz et al. (2011) picks examples uniformly at random. Even though the uniform sampling is not used in the previous theorems, it can improve the convergence rate of the method. Also, picking examples uniformly at random can be very helpful to eliminate problems in a dataset when the examples are grouped by some criteria in parts of the corpus and come in a particular order.

Algorithm 7: Structured Pegasos with the MM loss (Shalev-Shwartz et al., 2011).

Input : Training data: $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$, parameter $\boldsymbol{\lambda} \in \mathbb{R}^+$, $k \in \mathbb{N}$
Number of iterations: T
Output: Model parameters: $\mathbf{w}$

$\mathbf{w} := \mathbf{0}$;
for $t := 1$ **to** T **do**
  Choose $A_t \subseteq \{1, \dots N\}$ so that $|A_t| = k$;
  foreach $n \in A_t$ **do**
    Find $\widetilde{\boldsymbol{y}}^n = \arg\max_{\boldsymbol{y}} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$; `/* single best decoding */`
  end
  $A_t^+ := \{n \in A_t : \ell(\mathbf{w}; (\boldsymbol{x}^n, \widetilde{\boldsymbol{y}}^n)) > 0\}$;
  $\eta_t := 1/(\boldsymbol{\lambda} t)$;
  $\mathbf{w} := (1 - \eta_t \boldsymbol{\lambda})\mathbf{w} + \frac{\eta_t}{k} \sum_{n \in A_t^+} \Delta\mathbf{F}_n(\widetilde{\boldsymbol{y}}^n)$;
  [Optional: $\mathbf{w} := \min\left\{1, \frac{1/\sqrt{\boldsymbol{\lambda}}}{\|\mathbf{w}\|}\right\} \mathbf{w}$]; `/* projection */`
end

6.5 Implementation concerns

Regarding implementation, there are two main operations that are performed over the parameter vector: *scaling*, when we first scale $\mathbf{w}_t$ by factor $(1 - \eta_t \boldsymbol{\lambda})$ and optionally once again in the projection step, and the operation *add*, where we add scaled feature vectors to current parameters multiple times. Shalev-Shwartz et al. (2011) present a sparse implementation where *scaling* can be done in $\mathcal{O}(1)$ and *add* a new feature vector in $\mathcal{O}(d)$, where d is the number of non-zero elements in the feature vector. This is done by representing the parameter vector as $\mathbf{w} = a\mathbf{v}$. They also consider averaged parameters

$$\overline{\mathbf{w}}_T = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t, \tag{6.22}$$

and state that, in practice, the final hypothesis $\mathbf{w}_T$ often provides better results. We do not have a theoretical analysis for averaged parameters, since we do not bound the overall objective $f(\overline{\mathbf{w}}_T)$ in the structured case with the AS loss. However, we provide an experimental analysis for averaged and non-averaged parameters in the next section. In order to calculate averaged parameters we should not simply apply formula (6.22), because we will not get sparse updates. Xu (2011) presents an efficient procedure to find averaged parameters using linear transformation, where the addition of a new feature vector is also done in $\mathcal{O}(d)$. In practical implementations both averaged and non-averaged parameters require rescaling from time to time since the variables can go out of range. In a non-averaged implementation it can be easily done by rescaling a to one, while rescaling for averaged parameters can be found in the implementation of Bottou

Algorithm 8: (Restricted) Structured Pegasos with the AS loss.

Input : Training data: $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$, parameter $\boldsymbol{\lambda} \in \mathbb{R}^+$, $k \in \mathbb{N}$
Number of iterations: T
Output: Model parameters: $\mathbf{w}$

$\mathbf{w} := \mathbf{0}$;
for $t := 1$ **to** T **do**
  Choose n from $\{1, \dots N\}$;
  Select $B_n \subset \mathscr{Y}(\boldsymbol{x}^n)$ of size k e.g. $B_n := k\text{-}\arg\max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$;
  `/* k-best decoding */`
  $\widehat{\boldsymbol{y}}^n := \arg\max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \mathbf{w}^\mathsf{T} \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y})$; `/* prediction sequence */`
  $B_n := \{\boldsymbol{y} \in B_n : \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) \geq \ell(\mathbf{w}; (\boldsymbol{x}^n, \widehat{\boldsymbol{y}}^n))\}$; `/* for restricted version */`
  $B_n^+ := \{\boldsymbol{y} \in B_n : L(\boldsymbol{y}^n, \boldsymbol{y}) > \mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y})\}$;
  $\eta_t := 1/(\boldsymbol{\lambda} t)$;
  $\mathbf{w} := (1 - \eta_t \boldsymbol{\lambda})\mathbf{w} + \frac{\eta_t}{|B_n|} \sum_{\boldsymbol{y} \in B_n^+} \Delta\mathbf{F}_n(\boldsymbol{y})$;
  [Optional: $\mathbf{w} := \min\left\{1, \frac{1/\sqrt{\boldsymbol{\lambda}}}{\|\mathbf{w}\|}\right\} \mathbf{w}$]; `/* projection */`
end

(2008). Note that rescaling is not a sparse operation, but this is usually not a problem since it does not need to be called very often.

The algorithm also requires selecting a set B_n from the prediction violation set S_n. Checking if the structure belongs to the set S_n is an easy task, however building such a set can be a problem as we need to collect all structures with the score greater than the score for the prediction structure. Fortunately, the algorithm does not require the calculation of the whole set as we need only an arbitrary portion of its elements to approximate the objective function. Since we need structures with the highest score, we can use a k-best inference to create the $\mathscr{B}_{\mathbf{w}}^{k,n}$ set with the top k structures in a descending order, and then we can easily remove structures which do not belong to S_n to get the required set B_n from $S_n \cap \mathscr{B}_{\mathbf{w}}^{k,n}$.

6.6 Experimental results

We present experimental results on shallow parsing on the CONLL-2000 corpus[2] and part-of-speech (POS) tagging on the Brown corpus[3]. The results are presented in terms of F-measure, as a harmonic mean of precision and recall computed over tokens belonging to a chunk, while for POS tagging they are presented as accuracy, i.e. the proportion of correctly classified labels over all tokens in a sentence. To create feature vectors we used base templates from Table 4.1

[2] http://www.cnts.ua.ac.be/conll2000/chunking
[3] http://khnt.aksis.uib.no/icame/manuals/brown/

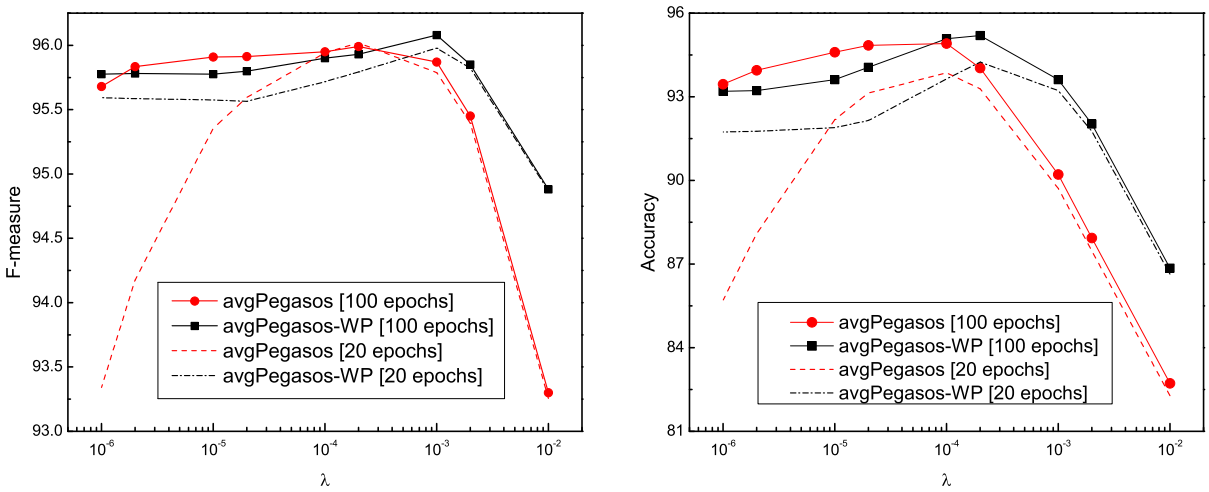

Figure 6.1: Dependence on the regularization parameter λ for the stochastic Pegasos algorithm. The results are presented with averaged parameters after 20 and 100 training epochs. Shallow parsing (left) and POS tagging (right).

and for finding the k-best path we applied A* inference described in Section 2.4.4.

Choosing an example for update in the Pegasos algorithm with the average sum loss is done sequentially, while for the MM loss the training set is partitioned into parts of size k on which the updates are performed sequentially. One pass through all training examples will be referred to as an *epoch*. In the forthcoming discussion we use the following abbreviations to specify the case we tested: the prefix *avg* before the algorithm name means that the test results are produced with averaged parameters (6.22), the Pegasos algorithm will be abbreviated with Peg, its restricted version with resPeg, the suffix MML and ASL after the algorithm name will respectively refer to the max-margin loss and the average sum loss, and -WP at the end will denote that the Pegasos algorithm is used without the projection step.

6.6.1 Restricted vs. non-restricted version

We first compare the results for Pegasos with the AS loss and the corresponding restricted version. Recall that the restriction is made by selecting the set B_n as the subset of S_n, and that it was needed for the theoretical analysis. Also note that using the restricted version implies the calculation of the prediction sequence (6.13), see Algorithm 8, which increases the training time since the additional Viterbi decoding must be performed. In Fig. 6.2 we see a minor difference in results whenever Pegasos with the AS loss is used with or without restriction. The specific parameter k is presented in caption, and there were similar small differences with other parameters we tested. Since the results are so similar, with the only difference in training time, in further analysis we only include the restricted Pegasos in time comparison with other algorithms and in the last table with the results of all methods.

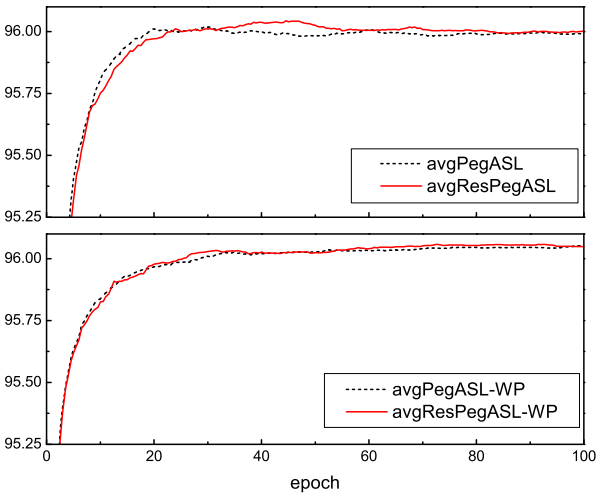

Figure 6.2: Results in terms of F-measure through epochs for the restricted and non-restricted version of the Pegasos algorithm with the averaged sum loss on shallow parsing. The curves are drawn with $k = 10$.

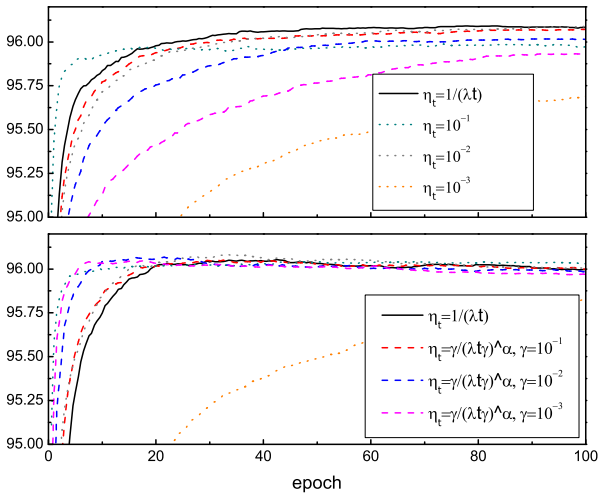

Figure 6.3: Results in terms of F-measure through epochs with different step sizes for the stochastic Pegasos algorithm without a projection step (top panel) and with a projection step (bottom panel). The specified step sizes in legends are common for both panels with $\alpha = 0.75$, but shown in two legends for better clarity.

6.6.2 Dependence of the regularization parameter

Figure 6.1 presents the influence of the regularization parameter λ on the results for shallow parsing and POS tagging. Small values of regularization parameters need more iterations i.e., long runtimes, which is mentioned in (Shalev-Shwartz et al., 2011) and can be clearly seen from Fig. 6.1 for both shallow parsing and POS tagging. However, large regularization parameters produce almost no difference in results after 20 and 100 epochs, but the outcomes are not satisfying. Interestingly, the best results on both datasets are achieved when the projection is not used, even if theoretical analysis provides similar bounds for both versions.

In order to select the regularization parameter in further experiments, we perform cross-validation. We use a 5-fold cross-validation to find the optimal parameter for each method separately, and then we employ this optimal parameter in a test scenario in all figures. When we present curves as the dependence of results through epochs, we use the optimal parameter provided after 100 training epochs in cross-validation. Further, in experiments we will include a case when we present only final results, and then the optimal number of training epochs will also be selected and included in cross-validation, which will be described later (see Section 6.6.7).

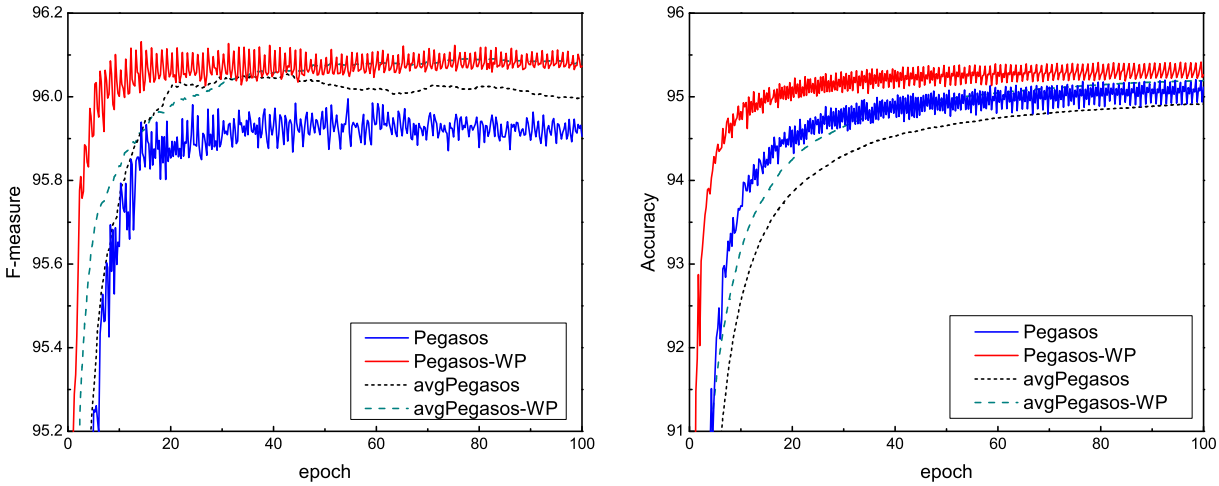

Figure 6.4: Results for shallow parsing (left) and POS tagging (right) through the iterations for the stochastic Pegasos algorithm dependent on the use of projection and averaged parameters.

6.6.3 Different step sizes

The Pegasos algorithm in the tth iteration changes parameters with the step size $\eta_t = 1/(\lambda t)$. This step can be generalized as $\eta_t = 1/(\lambda t)^\alpha$ with $\alpha = 1$. However, other values of $\alpha \in (1/2, 1]$ can be used, as suggested in Moulines and Bach (2011). The step can be further generalized as $\eta_t = \gamma/(\lambda\gamma t)^\alpha$ with a constant γ which is used in the ASGD implementation of (Bottou, 2008) with $\alpha = 0.75$. Another approach is to employ a constant small step size in each iteration (Ratliff et al., 2006).

In Fig. 6.3 we present an experiment where we tested different step sizes. As noticed by Nemirovski et al. (2009), with $\alpha = 1$ the choice of the regularization parameter is critical. However, as we described before, we perform cross-validation to choose the optimal regularization parameter, and the results using the Pegasos step are very similar to those with a constant step size equal to 10^{-2} or using $\eta_t = \gamma/(\lambda\gamma t)^\alpha$ with $\gamma = 10^{-1}$ when the projection step is not performed (top panel). A larger constant step size $\eta_t = 10^{-1}$ can provide faster convergence in the first few epochs, but in the end it does not achieve very good results. As opposite to the results without projection, where other step sizes do not seem to be beneficial, we can see in the bottom panel the improvement of other step sizes in first iterations when the projection step is included. After all epochs in that case, all step sizes provide very similar results, except a very small constant size $\eta_t = 10^{-3}$, with the best results achieved with $\eta_t = 10^{-1}$.

6.6.4 Projection and averaged parameters

Using averaged parameters helps to avoid oscillations in the test results. Figure 6.4 shows the oscillations when averaged parameters are not used. The Pegasos algorithm with averaged parameters slowly converges, but after 100 epochs it usually provides similar results to the

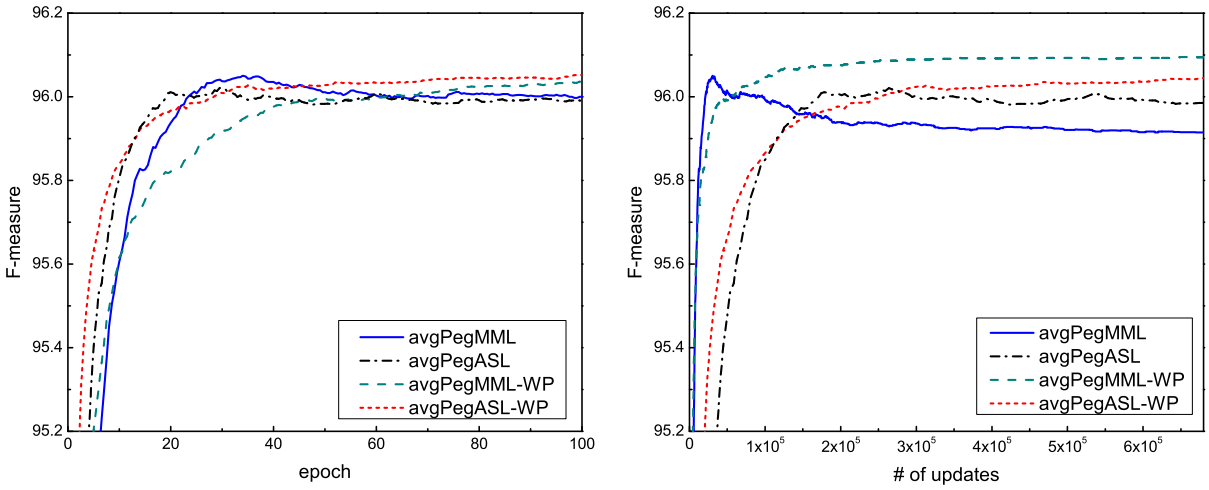

Figure 6.5: Comparison between Pegasos algorithms with the MM and AS loss. The dependence of the F-measure through epochs (left) and the dependence on the number of parameter updates (right). All curves are presented with the averaged parameters and with $k = 10$ for the shallow parsing problem.

non-averaged case. It is also possible to use a mixed approach that combines both cases by starting the averaging process after some portion of training iterations, as suggested by Rakhlin et al. (2012), which should result in faster convergence. An open question is when or if to start averaging (Shamir, 2012). Rakhlin et al. (2012) use averaging after $T/2$ iterations, Xu (2011) uses a comparison of the moving average of the empirical loss of non-averaged parameters and exponential moving average parameters to determine when to start averaging, while in the Bottou (2008) implementation is by default set to averaging after the first epoch. Additionally, Shamir and Zhang (2012) propose a simple averaging scheme which can be performed with other stopping criteria on-the-fly, with a number of training iterations unknown in advance.

6.6.5 Max margin loss vs. averaged sum loss

Figure 6.5 presents the results for the Pegasos algorithm with the MM and AS loss. In the left figure we see that both algorithms converge to the same results irrespective of whether or not we use the projection. As expected, the convergence in terms of the number of epochs through the training set is faster for Pegasos with the AS loss, since it considers many more structures for the update. On the other hand, when comparing convergence in terms of the number of updates, the convergence is faster when the MM loss is used, since it considers structures from k different training examples in one update. Even if they converge to same values, the choice of the algorithm will depend on whether we want to obtain the maximum results with a fewer number of updates or with a fewer number of examples presented to the algorithm. However, as we will see in further analysis, if we disregard the number of epochs/updates performed by the algorithm, and consider only the training time, then the stochastic version will be more suitable

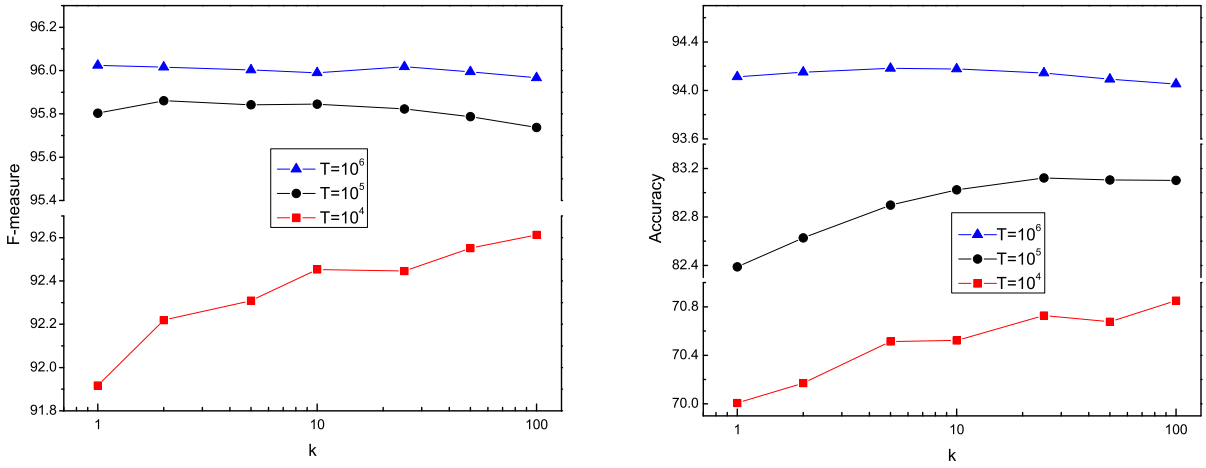

Figure 6.6: Results for the Pegasos algorithm for different values of parameter k after a fixed number of iterations specified in the legend. Results on shallow parsing (left) and POS tagging (right).

since the decoding time plays an important role in the overall runtime.

6.6.6 Dependence of parameter k

The influence of parameter k is shown in Fig. 6.6 with similar behavior on the given corpora. The optimal regularization parameter is found during the cross-validation separately for each point in the figure, which is defined with k provided on the x axis and the number of training iterations provided in legend. With a smaller number of iterations, the results get better as we increase k for both shallow parsing and POS tagging. This is expected since we extract more information from each example and learn based on k-best structures.

Also when the number of iterations is large, i.e., when the algorithm converges to its best result, all values of parameter k provide similar results with slight degradation for a very large k, e.g. 25, 50, 100. The reason is probably that including a lot of structures into the training procedure over numerous iterations creates many active features and may lead to overfitting. Thus, in this scenario increasing k is not useful and the single best version is enough to get very good results. However, achieving better results by increasing k with a smaller number of iterations shows us where the k-best version can be useful. If we want to train sequentially in one pass, or if we have a model which should be online corrected when a new example is presented, dealing with k higher than one should increase results as it includes more information from the new example.

6.6.7 Comparison with other algorithms

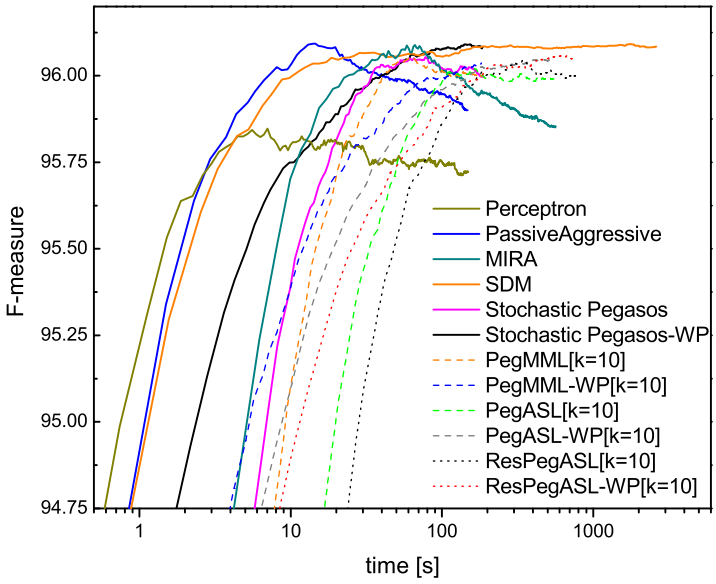

Figure 6.7: Training time comparison for different algorithms. For all algorithms, the results are presented with averaged parameters. The horizontal axis represents the training time, and the curves show the F-measure on the test corpus after various time spent on training. All curves represent the training time for 100 epochs on the shallow parsing problem and $k = 10$ is set for MIRA. Best seen in color.

Finally, we compare the Pegasos algorithm with other popular sequential structured algorithms: the Perceptron, MIRA, SDM and passive-aggressive algorithm, which we described in Chapter 3 in more detail. We have implemented all algorithms for comparison in C++. The experiments are performed on a computer with an Intel Core i7-3612QM CPU 2.10 GHz and 8 GB RAM. Figure 6.7 shows the dependence of F-measure through the time spent on training for previously mentioned algorithms together with different versions of the Pegasos algorithm. All implemented algorithms share the same structures and operators when working with feature and parameter vectors, thus the time comparison shown in Fig. 6.7 can be considered reliable.

Figure 6.7 shows the amount of time that is needed with $k = 10$ for the AS and MM loss in comparison to the stochastic Pegasos. Also the restricted version is included, which is one of the slowest since it needs additional decoding. For other online algorithms, we can see that after achieving the best results there is a degradation of the F-measure on the test corpus. They all need a few epochs to converge, so they need not be trained for 100 epochs since it can raise the problem of overfitting. This is especially evident with MIRA, because at each iteration it must satisfy all k new generated constraints in an online manner, which can notably make changes from the previous parameters. On the other hand, Pegasos algorithms need more iterations to achieve the highest results with averaged parameters, as we have seen before. Speaking of the highest results, we can see that all algorithms, except for the Perceptron, obtain very similar highest F-measures with slight differences among them. However, they mostly differ in

Table 6.1: Results for different algorithms and their corresponding parameters (regularization parameters, parameter k, and number of training epochs) obtained from a 5-fold cross-validation for each dataset. For all algorithms, the results are presented with averaged parameters, so the *avg* prefix is omitted in the algorithm name. The parameter C denotes the regularization parameter, where in the PA algorithm it means that the dual parameters are upper bound with C, while in MIRA and SDM it means that the sum of dual parameters is equal to C inside one example.

		Shallow parsing		
Method	Reg.	k	# epoch	F-measure
Perceptron	–	–	20	95.798
PassiveAggressive	$C = 1$	–	10	96.093
MIRA	$C = 10^{-2}$	2	30	96.095
SDM	$C = 10^{-1}$	–	100	96.084
Stohastic Peg	$\lambda = 2 \cdot 10^{-3}$	–	30	96.041
Stohastic Peg-WP	$\lambda = 10^{-3}$	–	100	96.082
PegMML	$\lambda = 2 \cdot 10^{-3}$	5	30	96.044
PegMML-WP	$\lambda = 10^{-3}$	2	100	96.094
PegASL	$\lambda = 2 \cdot 10^{-3}$	2	30	96.047
PegASL-WP	$\lambda = 10^{-3}$	2	50	96.076
ResPegASL	$\lambda = 2 \cdot 10^{-3}$	5	30	96.034
ResPegASL-WP	$\lambda = 10^{-3}$	2	50	96.078
		Pos tagging		
Method	Reg.	k	# epoch	Accuracy
Perceptron	–	–	30	94.829
PassiveAggressive	$C = 10^{-2}$	–	100	95.189
MIRA	$C = 10^{-2}$	5	50	95.214
SDM	$C = 10^{-1}$	–	200	95.292
Stohastic Peg	$\lambda = 10^{-4}$	–	200	95.065
Stohastic Peg-WP	$\lambda = 2 \cdot 10^{-3}$	–	200	95.320
PegMML	$\lambda = 10^{-4}$	2	200	95.068
PegMML-WP	$\lambda = 2 \cdot 10^{-3}$	2	200	95.302
PegASL	$\lambda = 10^{-4}$	5	200	95.070
PegASL-WP	$\lambda = 2 \cdot 10^{-3}$	5	200	95.317
ResPegASL	$\lambda = 10^{-4}$	2	200	95.066
ResPegASL-WP	$\lambda = 2 \cdot 10^{-3}$	5	200	95.298

terms of training time, the F-measure dependence on the number of training epochs and on the regularization parameter.

We can reach the same conclusions from Table 6.1. It provides the final results for each method on both datasets with the corresponding parameters selected via cross-validation. In order to select the stopping criteria during the cross-validation up to 100 epochs for shallow parsing (200 epochs for POS tagging), we selected the best results between {10, 20, 30, 50, 100} epochs for shallow parsing (including 200 for POS tagging). Therefore, the combination

of the best pair (regularization, number of epochs) is presented, and then used in a test scenario. If a method has a hyperparameter k, then it is also cross-validated from the set $\{2, 5, 10\}$. As mentioned before, we can see that different versions of Pegasos with the projection consistently provide lower results than the corresponding versions without the projection, even though the theoretical analysis provides similar bounds for both versions. Also, the optimal values of parameter k in Table 6.1 are 2 or 5 for all algorithms, which shows that the problems do not need a lot of additional structures to achieve the best results with enough training epochs.

From the previous experiments, we can see that the results are consistent on both problems. The dependence of the regularization parameter, parameter k, whether we use the averaging option, the restricted version or the projection, shows us similar behavior on both corpora. We have presented the results for sequence labeling problems. However, the presented method is applicable to other domains wherever we can define a structured classification problem and define a prediction violation set by k-best inference exactly or by some approximation.

In contrast to the existent mini-batch version with the max-margin loss, which can be suitable for parallel implementation, the advantage of the version with the average sum loss is extracting more information from each example. It should be useful, for instance, when the existing model should be corrected online as a new example is presented. However, dealing with multiple structures is not quite useful in case we train the model with many iterations and already include enough structures, where the stochastic version provides quite satisfying results with a simpler and faster training. Moreover, the proposed algorithm can be used in combination with mini-batch iterations taking the advantages from both approaches, i.e., extracting more information from each training example, and can also be suitable for a parallel processing of multiple examples.

Chapter 7

Conclusion

Structured models have numerous applications in real world problems, and currently present a state of the art approach in the classification of data whose outputs are represented with a structure. We have focused particularly on their training and their application in sequence labeling problems which belong to natural language processing. Here we summarize and discuss most important contributions.

We have introduced the averaged sum loss (ASL) and a primal sub-gradient method used for its optimization. Unlike the standard structured hinge loss which deals only with the structure with the highest score, the primal sub-gradient method with the ASL can deal with multiple structures inside one example. It relates with the k-best approach, but the choice of structures is not necessarily from the k-best set.

In the theoretical analysis, we have shown that the bound of cumulative prediction losses for the sub-gradient method with the ASL is at most as the bound for the stochastic version, while the empirical evaluation suggests that with a smaller number of iterations increasing the number of structures contributes to improving the results.

We have introduced k-best extensions of structural max-margin classifiers: k-best (restricted) passive-aggressive - (R)PA and k-best perceptron, which are defined completely in an online manner, easy to implement and, except for the RPA algorithm, very fast and suitable for large scale problems. The k-best RPA algorithm is presented for theoretical concerns, in order to satisfy a cumulative prediction loss bound similar to the one in the single best PA algorithm.

The introduced k-best versions of the SDM and LaRank algorithm perform full optimization inside each example making these algorithms highly computationally consuming. The k-best version of LaRank provides notable improvements in comparison to the single best case and the algorithm is suitable for training in one pass through the data.

We have presented a two structured model committee trained in a sequence, in which the latter model uses the prediction with its confidence of the former. In this organization, the confidence of a prediction or a context of predictions is estimated using alternative paths, which

helps to improve the results of separate models.

We have presented a sequential dual method for the structured ramp loss. The presented results on sequence labeling problems and the comparison with sub-gradient methods indicate that the ramp loss provides similar results to the corresponding method with the hinge loss. On the other hand, when the methods are exposed to outlayers during the learning procedure, the ramp loss consistently provides better results.

Bibliography

Bahl, L., Cocke, J., Jelinek, F., and Raviv, J. (1974). Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *IEEE Transactions on Information Theory*, 20(2):284–287.

Balamurugan, P., Shevade, S., Sundararajan, S., and Keerthi, S. S. (2011). A Sequential Dual Method for Structural SVMs. In *SDM 2011 - Proceedings of the Eleventh SIAM International Conference on Data Mining*.

Baum, L. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8.

Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563.

Bennett, K. P. and Mangasarian, O. L. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optimization methods and software*, 1(1):23–34.

Bordes, A., Bottou, L., Gallinari, P., and Weston, J. (2007). Solving multiclass support vector machines with LaRank. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 89–96, New York, NY, USA. ACM.

Bordes, A., Ertekin, S., Weston, J., and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619.

Bordes, A., Usunier, N., and Bottou, L. (2008). Sequence labelling svms trained in one pass. In *Machine Learning and Knowledge Discovery in Databases*, volume 5211, pages 146–161. Springer.

Bottou, L. (2008). SGD implementation available at. http://leon.bottou.org/projects/sgd.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.

Brill, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language*, pages 112–116. Association for Computational Linguistics.

Brill, E. (1994). Some advances in transformation-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, AAAI '94, pages 722–727, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Brooks, J. P. (2011). Support vector machines with the ramp loss and the hard margin loss. *Operations research*, 59(2):467–479.

Brown, D. G. and Golod, D. (2010). Decoding hmms using the k best paths: algorithms and applications. *BMC Bioinformatics*, 11(S-1).

Carrizosa, E., Nogales-Gómez, A., and Romero Morales, D. (2014). Heuristic approaches for support vector machines with the ramp loss. *Optimization Letters*, 8(3):1125–1135.

Chang, M.-W. and Yih, W.-t. (2013). Dual coordinate descent algorithms for efficient large margin structured prediction. *Transactions of the Association for Computational Linguistics (TACL)*, 1:207–218.

Chang, R. W. and Hancock, J. C. (1966). On receiver structures for channels having memory. *IEEE Transactions on Information Theory*, 12(4):463–468.

Collins, M. (2000). Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 175–182. Stanford University.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, volume 10 of *EMNLP '02*, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Collins, M., Globerson, A., Koo, T., Carreras, X., and Bartlett, P. L. (2008). Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9:1775–1822.

Collins, M., Schapire, R. E., and Singer, Y. (2002). Logistic regression, adaboost and bregman distances. *Mach. Learn.*, 48(1-3):253–285.

Collobert, R., Sinz, F., Weston, J., and Bottou, L. (2006). Trading convexity for scalability. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 201–208, New York, NY, USA. ACM.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

Crammer, K., McDonald, R., and Pereira, F. (2005). Scalable large-margin online learning for structured classification. In *NIPS Workshop on Learning With Structured Outputs*.

Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Culotta, A. and McCallum, A. (2004). Confidence estimation for information extraction. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 109–112, Stroudsburg, PA, USA. Association for Computational Linguistics.

Deshpande, M. and Karypis, G. (2002). Evaluation of techniques for classifying biological sequences. In *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '02, pages 417–431, London, UK, UK. Springer-Verlag.

Do, C. B., Le, Q. V., Teo, C. H., Chapelle, O., and Smola, A. J. (2008). Tighter bounds for structured estimation. In Koller, D., editor, *Advances in neural information processing systems*, pages 281–288, Red Hook. Curran Associates, Inc.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

Ertekin, S., Bottou, L., and Giles, C. L. (2011). Nonconvex online support vector machines. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(2):368–381.

Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Gimpel, K. (2012). *Discriminative Feature-Rich Modeling for Syntax-Based Machine Translation*. PhD thesis, Carnegie Mellon University.

Gimpel, K. and Cohen, S. (2007). Discriminative online algorithms for sequence labeling- a comparative study.

Gimpel, K. and Smith, N. A. (2010). Softmax-margin crfs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 733–736, Los Angeles, California. Association for Computational Linguistics.

Gimpel, K. and Smith, N. A. (2012). Structured ramp loss minimization for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 221–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Goodman, J. (1999). Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Greene, B. and Rubin, G. (1971). *Automatic Grammatical Tagging of English*. Department of Linguistics, Brown University.

Gupta, R. (2006). Conditional random fields. Technical report, The Kanwal Rekhi School of Information Technology (KReSIT), Indian Institute of Technology Bombay.

Hammerton, J., Osborne, M., Armstrong, S., and Daelemans, W. (2002). Introduction to special issue on machine learning approaches to shallow parsing. *J. Mach. Learn. Res.*, 2:551–558.

Hoefel, G. and Elkan, C. (2008). Learning a two-stage svm/crf sequence classifier. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 271–278, New York, NY, USA. ACM.

Ilić, V. M., Mančev, D., Todorović, B. T., and Stanković, M. S. (2012). Gradient computation in linear-chain conditional random fields using the entropy message passing algorithm. *Pattern Recognition Letters*, 33(13):1776–1784.

Jaggi, M., Lacoste-Julien, S., Schmidt, M., and Pletscher, P. (2013). Block-Coordinate Frank-Wolfe Optimization for Structural SVMs. In Dasgupta, S. and Mcallester, D., editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 53–61. JMLR Workshop and Conference Proceedings.

Joachims, T., Finley, T., and Yu, C.-N. J. (2009). Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59.

Kearns, M. and Valiant, L. (1989). Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, volume 21, pages 433–444, Seattle, Washington.

Keerthi, S. S. and DeCoste, D. (2005). A modified finite newton method for fast solution of large scale linear svms. *The Journal of Machine Learning Research*, 6:341–361.

Khreich, W., Granger, E., Miri, A., and Sabourin, R. (2010). On the memory complexity of the forward-backward algorithm. *Pattern Recognition Letters*, 31(2):91–99.

Klein, S. and Simmons, R. F. (1963). A computational approach to grammatical coding of english words. *Joutnal of the ACM*, 10(3):334–347.

Koenig, S. and Simmons, R. G. (1996). Unsupervised learning of probabilistic models for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2301–2308.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Krogh, A., Brown, M., Mian, I. S., Sjolander, K., and Haussler, D. (1994). Hidden markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531.

Kschischang, F. R., Frey, B. J., and Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

Kuhn, H. W. and Tucker, A. W. (1951). Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif. University of California Press.

Kupiec, J. (1992). Robust part-of-speech tagging using a hidden markov model. *Computer Speech & Language*, 6(3):225–242.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lane, T. D. (2000). *Machine Learning Techniques for the Computer Security Domain of Anomaly Detection*. PhD thesis.

Lee, C., Ryu, P.-M., and Kim, H. (2011). Named entity recognition using a modified pegasos algorithm. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2337–2340, New York, USA. ACM.

Lim, S., Lee, C., and Ra, D. (2013). Dependency-based semantic role labeling using sequence labeling with a structural svm. *Pattern Recognition Letters*, 34(6):696–702.

Mančev, D. and Todorović, B. (2015). k-best max-margin approaches for sequence labeling. *Computer Science and Information Systems (subbmitted)*.

Mančev, D. (2015). A sequential dual method for the structured ramp loss minimization. *Facta Universitatis, Series: Mathematics and Informatics*, 30(1):13–27.

Mančev, D. and Todorović, B. (2012). Confidence based learning of a two-model committee for sequence labeling. In *11th Symposium on Neural Network Applications in Electrical Engineering (NEUREL)*, pages 167–170. IEEE.

Mančev, D. and Todorović, B. (2014). A primal sub-gradient method for structured classification with the averaged sum loss. *International Journal of Applied Mathematics and Computer Science (AMCS)*, 24(4):917–930.

Martins, A. F. T., Smith, N. A., Xing, E. P., Aguiar, P. M. Q., and Figueiredo, M. A. T. (2011). Online learning of structured predictors with multiple kernels. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pages 507–515.

McAllester, D., Hazan, T., and Keshet, J. (2010). Direct loss minimization for structured prediction. In Lafferty, J., Williams, C. K. I., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 1594–1602.

McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 91–98, Stroudsburg, PA, USA. Association for Computational Linguistics.

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Mejer, A. and Crammer, K. (2010). Confidence in structured-prediction using confidence-weighted models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 971–981, Stroudsburg, PA, USA. Association for Computational Linguistics.

Meyer, I. M. and Durbin, R. (2004). Gene structure conservation aids similarity based gene prediction. *Nucleic acids research*, 32(2):776–783.

Mohri, M. (2002). Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350.

Molina, A. and Pla, F. (2002). Shallow parsing using specialized HMMs. *The Journal of Machine Learning Research*, 2:595–613.

Moulines, E. and Bach, F. R. (2011). Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 24*, pages 451–459. Curran Associates, Inc.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Nagata, M. (1994). A stochastic japanese morphological analyzer using a forward-DP backward-A* n-best search algorithm. In *Proceedings of the 15th Conference on Computational Linguistics*, volume 1 of *COLING '94*, pages 201–207, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. (2009). Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609.

Nguyen, N. and Guo, Y. (2007). Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 681–688, New York, NY, USA. ACM.

Ni, Y., Saunders, C., Szedmak, S., and Niranjan, M. (2010). The application of structured learning in natural language processing. *Machine Translation*, 24(2):71–85.

Nocedal, J. and Wright, J. S. (2006). *Numerical optimization*. Springer, 2nd edition.

Okazaki, N. (2007). Crfsuite: a fast implementation of conditional random fields (crfs).

Platt, J. C. (1999). Advances in kernel methods. chapter Fast training of support vector machines using sequential minimal optimization, pages 185–208. MIT Press, Cambridge, MA, USA.

Rabiner, L. R. (1990). Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rakhlin, A., Shamir, O., and Sridharan, K. (2012). Making gradient descent optimal for strongly convex stochastic optimization. In Langford, J. and Pineau, J., editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 449–456, New York, NY, USA. Omnipress.

Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. (2006). Subgradient methods for maximum margin structured learning. In *In ICML Workshop on Learning in Structured Output Spaces*, volume 46.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–407.

Salzberg, S. L. (1997). On comparing classifiers: Pitfalls toavoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3):317–328.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

Sha, F. and Pereira, F. (2003). Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 134–141, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 807–814, New York, NY, USA. ACM.

Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30.

Shamir, O. (2012). Open problem: Is averaging needed for strongly convex stochastic gradient descent? *Journal of Machine Learning Research-Proceedings Track*, 23:47–1.

Shamir, O. and Zhang, T. (2012). Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *arXiv preprint arXiv:1212.1824*.

Shor, N. Z., Kiwiel, K. C., and Ruszcayǹski, A. (1985). *Minimization Methods for Non-differentiable Functions*. Springer-Verlag New York, Inc., New York, NY, USA.

Song, D. and Sarkar, A. (2008). Training a perceptron with global and local features for chinese word segmentation. In *Third International Joint Conference on Natural Language Processing, IJCNLP 2008*, pages 143–146.

Soong, F. K. and Huang, E.-F. (1991). A tree-trellis based fast search for finding the n-best sentence hypotheses in continuous speech recognition. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 705–708, Toronto.

Sutton, C. A. (2008). *Efficient training methods for conditional random fields*. PhD thesis, University of Massachusetts Amhers.

Sutton, C. A. and McCallum, A. (2006). *Introduction to Conditional Random Fields for Relational Learning*. MIT Press.

Taskar, B. (2004). *Learning Structured Prediction Models: A Large Margin Approach*. PhD thesis, Stanford University, CA.

Taskar, B., Guestrin, C., and Koller, D. (2004). Max-margin markov networks. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 25–32, Cambridge, MA. MIT Press.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58:267–288.

Tjong Kim Sang, E. F. (2002). Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–4, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, ConLL '00, pages 127–132, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In Greiner, R. and Schuurmans, D., editors, *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, New York, NY, USA. ACM.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Vapnik, V. (1998). *Statistical learning theory*. Wiley.

Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., and Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 969–976, New York, NY, USA. ACM.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Voutilainen, A. (1995). A syntax-based part-of-speech analyser. In *Proceedings of the Seventh Conference on European Chapter of the Association for Computational Linguistics*, EACL '95, pages 157–164, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Wang, L., Jia, H., and Li, J. (2008). Letters: Training robust support vector machine with smooth ramp loss in the primal space. *Neurocomputing*, 71(13-15):3020–3025.

Wang, Z. and Vucetic, S. (2009). Fast online training of ramp loss support vector machines. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 569–577.

Warrender, C., Forrest, S., and Pearlmutter, B. (1999). Detecting intrusions using system calls: Alternative data models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pages 133–145. IEEE.

Weston, J. and Watkins, C. (1998). Multi-class support vector machines. Technical report, Department of Computer Science, Royal Holloway, University of London.

Xing, Z., Pei, J., and Keogh, E. (2010). A brief survey on sequence classification. *SIGKDD Explor. Newsl.*, 12(1):40–48.

Xu, W. (2011). Towards optimal one pass large scale learning with averaged stochastic gradient descent. *arXiv preprint arXiv:1107.2490*.

Yuille, A. L. and Rangarajan, A. (2003). The concave-convex procedure. *Neural Computation*, 15(4):915–936.

Zhou, G. and Su, J. (2002). Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 473–480, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

Appendix A

Treniranje strukturnih klasifikatora za različite funkcije gubitaka sa primenom na probleme klasifikovanja sekvenci

Klasifikacija sekvenci (engl. sequence labeling) predstavlja postupak dodeljivanja odgovarajuće oznake (klase) svakom elementu sekvence. Na primer, kada imamo sekvencu koja predstavlja rečenicu iz govornog jezika, možemo definisati problem koji će detektovati gramatičku kategoriju reči u njoj, da li se radi o imenici, glagolu, pridevu, prilogu, itd. Takav problem naziva se prepoznavanje vrsta reči (engl. part-of-speech tagging). Na prvi pogled, problem može delovati jednostavno, potrebno je samo pronaći odgovarajuću kategoriju u rečniku. Međutim, teškoće nastaju zbog mogućnosti da jedna reč pripada različitim kategorijama u zavisnosti od konteksta u kome je upotrebljena u rečenici. Drugi problem klasifikovanja sekvekci je plitko parsiranje (engl. shallow parsing). On se bavi prepoznavanjem sintagmi u rečenici kao što su: imeničke, glagolske, pridevske, priloške, itd. Ovaj problem najčešće dolazi nakon prepoznavanja vrsta reči, a prethodi mu problem sintaksnog parsiranja. Sledeći problem je prepoznavanje entiteta (engl. named-entity recognition), gde je potrebno prepoznati određene entitete u tekstu kao što su imena osoba, organizacija, lokacija, itd. Problemi klasifikovanja sekvenci takođe postoje i u drugim oblastima, kao što je pronalaženje obrazaca u DNK lancima koji odgovaraju specifičnoj bolesti ili klasifikovanje proteina u oblasti biologije, zatim detektovanje upada u sistem gde je potrebno razlikovati legitimne i neligitimne aktivnosti u sistemskim pozivima u oblasti bezbednosti računara .

Jedan od mogućih pristupa rešavanju ovog problema je kreiranje pravila koja bi određivala klasu reči u zavisnosti od njenog konteksta. Iako je kreiranje takvog sistema moguće, on ima svoje nedostatke. Pisanje pravila može biti naporno i zahteva odlično poznavanje lingvistike, zatim u govornom jeziku postoji dosta izuzetaka a pokušajem da se većina njih uzme u obzir sistem pravila bi postao prilično glomazan i težak za organizovanje. Ali možda najveća mana

ovakvog sistema je to da su pravila upotrebljiva samo za konkretan problem, pa ukoliko promenimo jezik, na primer, moraćemo pisati novi skup pravila.

Drugi način za rešavanje problema klasifikovanja sekvenci je kreiranje statističkog modela. Prednost ovakvih modela iskazuje se u tome da jednom kreiran model može biti treniran (obučavan) na različitim domenima (npr. na različitim jezicima) kao i na različitim problemima. Da bismo obučili takav model, potreban nam je označeni skup podataka koji ćemo zvati korpusom za obučavanje (engl. training corpus), a pristup pronalaženja funkcije koja modeluje preslikavnje ulaznih podataka u odgovarajuće unapred definisane oznake nazivaćemo nadgledano učenje (engl. supervised learning). Elementarni zadatak nadgledanog učenja je binarna klasifikacija koja klasifikuje podatke u jednu od dve moguće klase. Iako možemo upotrebiti više binarnih ili jedan višeklasni klasifikator da bismo rešili problem klasifikovanja sekvenci, predviđanje svakog elementa sekvence nezavisno neće dati maksimalne rezultate prepoznavanja (Nguyen and Guo, 2007).

Prvobitan pristup klasifikovanju sekvenci podrazumevao je korišćenje generativnog skrivenog Markovljevog modela (engl. Hidden Markov model) (Rabiner, 1990). Ovaj model koristi Markovljevu pretpostavku prvog reda, koja kaže da verovatnoća sledećeg stanja zavisi samo od prethodnog stanja, a ne od onih stanja koje prethode prethodnom. Skriveni Markovljev model se sastoji od skrivenih stanja, skupa opservacija koje se mogu generisati iz svakog skrivenog stanja i tri tipa verovatnoća koje se procenjuju: verovatnoće prelaska između dva skrivana stanja, verovatnoće da skriveno stanje generiše opservaciju kao izlaz i verovatnoće inicijalnih stanja. Ove verovatnoće se obično procenjuju maksimiziranjem združene verodostojnosti nad podacima za treniranje. Da bi se problem mogao rešiti, skriveni Markovljevi modeli moraju da pretpostave uslovnu nezavisnost opservacija pri datim klasama. Takođe, kao generativni modeli oni ne mogu da uključe različite vrste zavisnosti, na primer da trenutni prelaz između skrivenih stanja može da zavisi od konteksta oko trenutne opservacije. Ovi nedostaci su otklonjeni uvođenjem strukturnih klasifikatora.

Strukturni klasifikatori (engl. structured classifiers) se bave problemom klasifikacije gde izlaz klasifikatora predstavlja strukturu, a ne jednu oznaku kao što je bio slučaj sa binarnim i višeklasnim klasifikatorima. Strukture se sastoje iz manjih jedinica koje predstavljaju izlazne klase i među kojima važe različite zavisnosti. U problemima pridruživanja oznaka svakom elementu sekvence, izlazne strukture su predstavljene u obliku lanaca, ali u drugim problemima strukture mogu biti predstavljene u vidu stabala, mreža ili grafova u opštem slučaju. Ovi klasifikatori mogu da uključe različite veze između ulaznih opservacija i izlaznih klasa, što je bio jedan od nedostataka generativnih skrivenih Markovljevih modela. Kao posledica mogućnosti da uključe različite veze, oni mogu da daju bolje rezultate u odnosu na standardne binarne i višeklasne klasifikatore, ali po ceni složenijeg obučavanja koje pritom zahteva i odgovarajuće dekodiranje. U proteklih petnaestak godina, predloženo je više modela

strukturnih klasifikatora i algoritama za njihovo obučavanje kao što su: strukturna verzija Perceptron algoritma kao najjednostavniji strukturni onlajn algoritam, strukturne verzije mašina sa vektorima podrške (engl. support vector machines), uslovna slučajna polja (engl. conditional random fields) kao probabilistički diskriminativni modeli. Ono što povezuje sve ove modele je da svaki od njih podrazumeva optimizaciju sa odgovarajućom funkcijom gubitka. U ovoj tezi razmatramo obučavanje strukturnih modela sa različitim funkcijama gubitaka koncentrišući se na njihovu primenu u problemima klasifikovanja sekvenci, tj. kada su strukture u vidu lanaca.

U drugom poglavlju ove teze posmatra se problem učenja kao problem optimizacije regularizacione funkcije i funkcije gubitka, kao i algoritmi za dekodiranje koji su potrebni u procesu obučavanja strukturnih klasifikatora. U trećem poglavlju uvodimo četiri algoritma za obučavanje strukturnih modela korišćenjem k najboljih struktura i razmatramo njihove teorijske osobine i eksperimentalne rezultate. U četvrtom poglavlju je predstavljeno kombinovanje dva strukturna modela, gde u procesu obučavanja drugi model koristi previđanja iz prvog modela zajedno sa njegovim poverenjem. Peto poglavlje prezentuje sekvencijalni dualni metod za optimizaciju strukturne dvostruko prelomljene funkcije gubitka (engl. ramp loss) kao i rezultate na realnom i veštački generisanom problemu klasifikovanja sekvence. Šesto poglavlje uvodi usrednjenu funkciju gubitka i primarni subgradijentni metod za optimizaciju, uz teorijsku analizu i eksperimentalne rezultate uvedenog metoda. U nastavku slede ključni rezultati pomenutih poglavlja.

A.1 Strukturni klasifikatori

U ovom odeljku uvodimo osnovne pojmove u vezi sa strukturnim klasifikatorima, čije ćemo obučavanje posmatrati kao problem optimizacije sa različitim funkcijama gubitaka. Razmatraćemo takođe i algoritme za dekodiranje koji se koriste u procesu obučavanja i klasifikacije.

Neka je $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$ skup podataka za obučavanje, gde $\boldsymbol{x}^n \in \mathscr{X}$ predstavlja opservaciju, a $\boldsymbol{y}^n \in \mathscr{Y}(\boldsymbol{x}^n)$ odgovarajući izlaz. Na primer, u binarnoj klasifikaciji $\boldsymbol{y}^n$ je oznaka koja opisuje kojoj od dve moguće klase pripada opservacija $\boldsymbol{x}^n$, tako da se izlazni skup može prikazati kao $\mathscr{Y}(\boldsymbol{x}^n) = \{-1, 1\}$. U višeklasnoj klasifikaciji, $\boldsymbol{y}^n$ predstavlja jednu od više mogućih oznaka, dok u strukturnom slučaju označava strukturu. U problemima nadgledanog učenja, zadatak je pronaći funkciju $h_{\mathbf{w}}(\boldsymbol{x})$ na osnovu skupa podataka za obučavanje $\mathscr{D}$ koja će praviti što je moguće manju grešku pri pridruživanju izlaza $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x})$ proizvoljnom ulaznom elementu $\boldsymbol{x} \in \mathscr{X}$. Standardni način pronalaženja ovakve funkcije je pretpostavljanje njenog oblika (npr. linearna funkcija po nepoznatim parametrima), a zatim određivanje parametara $\mathbf{w}$

minimiziranjem regularizovanog empirijskog rizika na $\mathscr{D}$

$$\min_{\mathbf{w}} \frac{\lambda}{2}\Omega(\mathbf{w}) + R^{\mathscr{D}}_{emp}(\mathbf{w}), \tag{A.1}$$

gde je $\Omega(\mathbf{w})$ regularizaciona funkcija uvedena u cilju izbegavanja prevelikog prilagođavanja podacima za obučavanje (engl. overfitting) što kao posledicu ima smanjivanje tačnosti prepoznavanja na primerima koji nisu uključeni u toku obučavanja, dok je $\lambda \in \mathbb{R}^+$ regularizacioni parametar. Funkcija $R^{\mathscr{D}}_{emp}(\mathbf{w})$ predstavlja *empirijski rizik* na $\mathscr{D}$

$$R^{\mathscr{D}}_{emp}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ell_n(\mathbf{w}), \tag{A.2}$$

gde je $\ell_n(\mathbf{w})$ funkcija greške na n-tom podatku koja opisuje koliko je odgovarajuća struktura $\boldsymbol{y}^n$ daleko od predviđene strukture $h_{\mathbf{w}}(\boldsymbol{x}^n)$. Postoje različite kombinacije izbora regularizacione funkcije i funkcije greške koje dovode do različitih algoritama mašinskog učenja. Primer regularizacione funkcije je L_2 regularizacija za koju dobijamo sledeći problem

$$\min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{n=1}^{N} \ell_n(\mathbf{w}), \tag{A.3}$$

koji ćemo razmatrati u ovoj tezi, gde $\|\cdot\|$ predstavlja L_2 normu. Pored ove funkcije, koristi se i L_1 regularizacija (Tibshirani, 1996) koja omogućava dobijanje sparsnog modela, kao i elastična mreža (Zou and Hastie, 2005) koja kombinuje L_1 i L_2 regularizaciju. Takođe, postoje različite mogućnosti za izbor funkcije greške. Tabela 2.1 u odeljku 2 daje pregled ovih funkcija koje se koriste u klasifikaciji u binarnom i strukturnom slučaju. Nakon izbora regularizacione i funkcije greške, sledeći korak je nalaženje rešenja optimizacionog problema. Ukoliko je funkcija gubitka diferencijabilna (npr. funkcija greške uslovnih slučajnih polja), onda možemo primeniti metod gradijentog spusta za optimizaciju. Međutim, nisu sve funkcije diferencijabilne. Na primer, zglobna funkcija (engl. hinge loss) je nediferencijabilna konveksna funkcija za čiju optimizaciju možemo primeniti subgradijente metode (Shalev-Shwartz et al., 2007), ili kvadratnu optimizaciju u dualnom prostoru (Nocedal and Wright, 2006).

Kada izaberemo regularizacionu i funkciju greške kao i odgovarajući metod za optimizaciju, poslednji korak je izbor regularizacionog koeficijenta λ. Postavljanje suviše male vrednosti za parametar λ dovodi do prevelikog prilagođavanja modela podacima za obučavanje, a samim tim i do loše sposobnosti generalizacije, dok suviše velike vrednosti dovode do toga da se željena funkcija $h_{\mathbf{w}}(\boldsymbol{x})$ ne pronađe. Jedan od načina izbora λ je metod unakrsne validacije (engl. cross validation) (Kohavi, 1995).

Posmatrajmo sada strukturni slučaj. Neka je $\mathcal{X}$ ulazni alfabet i $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$ skup

podataka za obučavanje gde svako $\boldsymbol{x}^n \in \mathcal{X}^{T_n}$ predstavlja ulaznu sekvencu dužine T_n kojoj odgovara izlazna struktura $\boldsymbol{y}^n$. Skup svih mogućih struktura za sekvencu $\boldsymbol{x}^n$ označićemo sa $\mathscr{Y}(\boldsymbol{x}^n)$ i neka je $\mathscr{Y}_{-n} = \mathscr{Y}(\boldsymbol{x}^n) \setminus \boldsymbol{y}^n$. U daljem izlaganju, fokusiraćemo se na probleme klasifikovanja sekvenci gde je $\mathscr{Y}(\boldsymbol{x}^n) = \mathcal{Y}^{T_n}$, dok $\mathcal{Y}$ predstavlja skup mogućih klasa za element iz $\mathcal{X}$. Linearni klasifikatori pronalaze funkciju $h_{\mathbf{w}}$ na osnovu podataka $\mathscr{D}$, koja preslikava svaku ulaznu sekvencu $\boldsymbol{x}$ nad alfabetom $\mathcal{X}$ u jedan element skupa $\mathscr{Y}(\boldsymbol{x})$ u sledećem obliku:

$$h_{\mathbf{w}}(\boldsymbol{x}) = \underset{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x})}{\arg\max}\, \mathbf{w}^{\mathsf{T}} \mathbf{F}(\boldsymbol{x}, \boldsymbol{y}) = \hat{\boldsymbol{y}}, \tag{A.4}$$

sa parametrima $\mathbf{w} \in \mathbb{R}^M$, gde $\mathbf{F}(\boldsymbol{x}, \boldsymbol{y})$ predstavlja globalni vektor karakteristika (engl. global feature vector) koji meri kompatibilnost ulaza $\boldsymbol{x}$ sa strukturom $\boldsymbol{y}$. U zavisnosti od izbora funkcije gubitka definisani su različiti strukturni modeli.

Strukturni klasifikatori sa maksimalnom marginom (Tsochantaridis et al., 2005) koriste strukturnu zglobnu funkciju gubitka (engl. hingle loss) $\ell_n^{\text{MM}}(\mathbf{w})$ definisanu sa

$$\ell_n^{\text{MM}}(\mathbf{w}) = \max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})), \quad \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) = \max(0, L(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{w}^{\mathsf{T}} \Delta\mathbf{F}_n(\boldsymbol{y})),$$

pri čemu je $\ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y}))$ funkcija gubitka za strukturu $\boldsymbol{y}$. Optimizacioni problem bez ograničenja (A.3) sa prethodno definisanom strukturnom zglobnom funkcijom možemo predstaviti kao problem sa ograničenjima u sledećem obliku

$$\min_{\mathbf{w}, \boldsymbol{\xi}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{n=1}^{N} \xi_n, \quad \text{s.t. } \mathbf{w}^{\mathsf{T}} \Delta\mathbf{F}_n(\boldsymbol{y}) \geq L(\boldsymbol{y}^n, \boldsymbol{y}) - \xi_n, \ \ \forall n, \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n), \tag{A.5}$$

gde je $\Delta\mathbf{F}_n(\boldsymbol{y}) = \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n) - \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y})$, a funkcija cene $L(\boldsymbol{y}^n, \boldsymbol{y})$ (engl. cost function) predstavlja cenu označavanja sekvence $\boldsymbol{x}^n$ strukturom $\boldsymbol{y}$ umesto strukturom $\boldsymbol{y}^n$. U problemima sa ograničenjima umesto parametra λ najčešće ćemo koristi $C = 1/(\lambda|D|)$ kao regularizacioni parametar, gde je $|D|$ broj primera uključenih u optimizaciju, što se u slučaju optimizacije (A.5) svodi na $C = 1/(\lambda N)$.

Uslovna slučajna polja (engl. conditional random fields) (Lafferty et al., 2001) su definisana uslovnom verovatnoćom

$$p(\boldsymbol{y}|\boldsymbol{x}; \mathbf{w}) = \frac{1}{Z(\boldsymbol{x}; \mathbf{w})} \mathrm{e}^{\mathbf{w}^{\mathsf{T}} \mathbf{F}(\boldsymbol{x}, \boldsymbol{y})}, \quad \text{gde je} \quad Z(\boldsymbol{x}; \mathbf{w}) = \sum_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x})} \mathrm{e}^{\mathbf{w}^{\mathsf{T}} \mathbf{F}(\boldsymbol{x}, \boldsymbol{y})}, \tag{A.6}$$

pri čemu je $Z(\boldsymbol{x}; \mathbf{w})$ normalizaciona funkcija. Ukoliko posmatramo maksimiziranje logaritma verodostojnosti na podacima za obučavanje $\mathscr{D}$ sa dodatom kvadratnom regularizacionom

funkcijom, problem možemo predstaviti kao optimizaciju (A.3), pri čemu je funkcija greške:

$$\ell_n^{\text{CRF}}(\mathbf{w}) = \log\Big(\sum_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x})} \mathrm{e}^{\mathbf{w}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n,\boldsymbol{y})}\Big) - \mathbf{w}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n,\boldsymbol{y}^n). \tag{A.7}$$

Za optimizaciju ove funkcije gubitka potrebno je zaključivanje (dekodiranje) tokom obučavanja, ali i nakon završenog obučavanja potreban je algoritam za dekodiranje koji će određivati strukture iz (A.4).

A.1.1 Algoritmi za dekodiranje sekvenci

Neka je $\big(\mathbb{K},\oplus,\otimes,0,1\big)$ poluprsten i neka je $\boldsymbol{y}=(y_0,\ldots,y_T)$ sekvenca promenljivih dužine T, čiji elementi uzimaju vrednosti iz skupa $\mathcal{Y}$. Definišimo funkcije $u_t:\mathcal{Y}^2\to\mathbb{K}$ za $t=1,\ldots,T$, kao i funkciju $u:\mathcal{Y}^{T+1}\to\mathbb{K}$, pretpostavljajući pritom da sledeće razlaganje

$$u(\boldsymbol{y}) = \bigotimes_{t=1}^{T} u_t(y_{t-1},y_t) \tag{A.8}$$

važi za sve $\boldsymbol{y}=(y_0,\ldots,y_T)\in\mathcal{Y}^{T+1}$. U radu (Kschischang et al., 2001) opisan je zbir-proizvod (engl. sum-product) algoritam nad faktor grafovima, pri čemu se za operacije sabiranja i množenja mogu koristiti različiti poluprstenovi. Mi razmatramo primenu ovog algoritma na strukturu lanca. Algoritam se sastoji od računanja sledećih vektora rekurzivno ka napred

$$\alpha_i(y_i) = \bigoplus_{y_{0:i-1}} \bigotimes_{t=1}^{i} u_t(y_{t-1},y_t) = \bigoplus_{y_{i-1}} u_{i-1}(y_{i-1},y_i)\otimes\alpha_{i-1}(y_{i-1}), \tag{A.9}$$

sa inicijalizacijom $\alpha_0(y_0)=1$, kao i računanja sledećih vektora rekurzivno ka nazad

$$\beta_i(y_i) = \bigoplus_{y_{i+1:T}} \bigotimes_{t=i+1}^{T} u_t(y_{t-1},y_t) = \bigoplus_{y_{i+1}} u_{i+1}(y_i,y_{i+1})\otimes\beta_{i+1}(y_{i+1}), \tag{A.10}$$

sa inicijalizacijom $\beta_T(y_T)=1$. Nakon izračunavanja α_{t-1} i β_t vektora, moguće je rešiti sledeći marginalizacioni problem

$$v_t(y_t,y_{t+1}) = \bigoplus_{y_{\{t-1,t\}^c}} u(\boldsymbol{y}) = \bigoplus_{y_{\{t-1,t\}^c}} \bigotimes_{i=1}^{T} u_i(y_{i-1},y_i) = \alpha_{t-1}(y_{t-1})\otimes u_t(y_{t-1},y_t)\otimes\beta_t(y_t), \tag{A.11}$$

kao i normalizacioni problem

$$Z = \bigoplus_{\boldsymbol{y}} u(\boldsymbol{y}) = \bigoplus_{\boldsymbol{y}} \bigotimes_{t=1}^{T} u_t(y_{t-1},y_t) = \bigoplus_{y_T} \alpha_T(y_T). \tag{A.12}$$

Navedene rekurzije sa specifičnim poluprstenovima se koriste za rešavanje različitih problema

strukturne klasifikacije što je predstavljeno u Tabeli 2.2. U nastavku ćemo opisati neke od njih.

Viterbijev algoritam Viterbijev algoritam (Viterbi, 1967) se koristi za nalaženje optimalne sekvence $\hat{\boldsymbol{y}}$ iz (A.4), ali se takođe može modifikovati za nalaženje optimalne sekvence sa dodatom funkcijom cene pod pretpostavkom da se i funkcija cene kao i globalni vektor karakteristika razlažu na sledeći način:

$$\mathbf{F}(\boldsymbol{x},\boldsymbol{y}) = \sum_{t=1}^{T} \boldsymbol{f}(y_{t-1}, y_t, \boldsymbol{x}, t), \qquad L(\boldsymbol{y}^n,\boldsymbol{y}) = \sum_{t=1}^{T} l(y_t^n, y_t), \tag{A.13}$$

gde $l(y_t^n, y_t)$ predstavlja cenu označavanja opservacije x_t sa y_t umesto y_t^n. Za n-ti primer $(\boldsymbol{x}^n, \boldsymbol{y})_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)}$, težina prelaza iz labele c u labelu c' na opservaciji x_t^n kao i funkcija u_t sa uključenom funkcijom cene definišu se kao

$$Q_{\boldsymbol{x}^n,t}(c,c') = \mathbf{w}^\mathsf{T}\boldsymbol{f}(c,c',\boldsymbol{x}^n,t), \qquad u_t(c,c') = Q_{\boldsymbol{x}^n,t}(c,c') + \nu\, l(y_t^n, c)$$

odakle se koristećenjem rekurzivne formule (A.9) za poluprsten $\left(\mathbb{R}^*, \max, +, -\infty, 0\right)$ dobija težina optimalne sekvence, gde je $\mathbb{R}^* = \mathbb{R}\cup\{-\infty\}$.

Viterbijev algoritam nalaženja k-najboljih sekvenci Viterbijev algoritam može se modifikovati da nalazi težine k najboljih sekvenci. Definisanjem funkcije $\boldsymbol{u}_t$ kao vektora od k identičnih elemenata

$$\boldsymbol{u}_t(c',c) = \left(Q_{\boldsymbol{x}^n,t}(c',c) + \nu\, l(y_t^n,c), \ldots, Q_{\boldsymbol{x}^n,t}(c',c) + \nu\, l(y_t^n,c)\right),$$

k-najbolji Viterbijev algoritam se može posmatrati kako algoritam definisan rekurzijama (A.9) za k-najbolji poluprsten $\left(\mathbb{R}^{*k}, \oplus^k, \otimes^k, -\boldsymbol{\infty}, \mathbf{0}\right)$ (Goodman, 1999; Mohri, 2002), pri čemu su operacije $\oplus^k$ i $\otimes^k$ definisane na sledeći način

$$\boldsymbol{a}\otimes^k\boldsymbol{b} = k\text{-}\max\{a_i + b_j | 1\le i,j\le k\}, \quad \boldsymbol{a}\oplus^k\boldsymbol{b} = k\text{-}\max\{a_1,\ldots,a_k,b_1,\ldots,b_k\}.$$

Pamćenje k najboljih parcijalnih težina na svakoj poziciji vodi do vremenske kompleksnosti $\mathcal{O}(|\mathcal{Y}|^2Tk)$ i memorijske kompleksnosti $\mathcal{O}(|\mathcal{Y}|Tk)$.

A* dekodiranje Drugi način nalaženja k najboljih sekvenci je korišćenje A* algoritma za generisanje k najboljih putanja (Nagata, 1994; Soong and Huang, 1991). Ovaj algoritam se takođe može prilagoditi za nalaženje k najboljih sekvenci sa uključenom funkcijom cene. Sastoji se iz dva koraka: korak napred za nalaženje parcijalnih dužina od početka do svake pozicije koji se obavlja Viterbijevim algoritmom sa uključenom funkcijom cene, i korak nazad, gde A* algoritam koristi prethodne dužine za generisanje k najboljih putanja pomoću

sledećih rekurzivnih formula:

$$(A_t(1),\dots,A_t(k)) = k\text{-}\max_{\substack{c\in\mathcal{Y}\\ j=1,\dots,k}} g_{\boldsymbol{x}^n,t}(c,j), \quad (B_t(1),\dots,B_t(k)) = k\text{-}\arg\max_{\substack{c\in\mathcal{Y}\\ j=1,\dots,k}} g_{\boldsymbol{x}^n,t}(c,j)$$
$$g_{\boldsymbol{x}^n,t}(c,j) = V_t(c) + Q_{\boldsymbol{x}^n,t+1}(c, B_{t+1}(j)) + A_{t+1}(j) + \nu\, l(y^n_{t+1}, B_{t+1}(j)).$$

Vremenska kompleksnost algoritma je $\mathcal{O}(|\mathcal{Y}|^2T + |\mathcal{Y}|Tk\log k)$, dok je memorijska kompleksnost $\mathcal{O}(|\mathcal{Y}|T + kT)$. Jedna od mana ovakvog pristupa je potreba za računanjem težina prelaza na svakoj poziciji dva puta, što u zavisnosti od načina čuvanja vektora karakteristika može biti vremenski zahtevna operacija. U slučaju da se algoritam ne primenjuje na vrlo dugačke sekvence, ovaj problem moguće je rešiti čuvanjem svih matrica težina za drugi prolazak.

Dekodiranje napred-nazad Algoritam *napred-nazad* (engl. forward-backward) uveden je u (Baum and Petrie, 1966; Chang and Hancock, 1966) i može se koristiti za određivanje gradijenta uslovnih slučajnih polja. Algoritam koristi rekurzije (A.9)-(A.10) sa odgovarajućim poluprstenom. U slučaju poluprstena sa standardnim operacijama množenja i sabiranja $\left(\mathbb{R},+,\cdot,0,1\right)$, definisanjem sledećih funkcija $u_t(y_{t-1},y_t) = \mathrm{e}^{\mathbf{w}^\mathsf{T}\boldsymbol{f}(y_{t-1},y_t,\boldsymbol{x},t)}$, gradijent uslovnih slučajnih polja može se izračunati kao

$$\nabla_{\mathbf{w}} Z(\boldsymbol{x};\mathbf{w}) = \sum_{k=1}^{T}\sum_{y_{\{k-1,k\}}}\left(\sum_{y_{\{k-1,k\}^c}}\prod_{t=1}^{T}\mathrm{e}^{\mathbf{w}^\mathsf{T}\boldsymbol{f}(y_{t-1},y_t,\boldsymbol{x},t)}\right)\cdot\boldsymbol{f}(y_{k-1},y_k,\boldsymbol{x},k). \tag{A.14}$$

Algoritam napred-nazad nad zbir-proizvod poluprstenom ima problema sa numerčkom stabilnošću, jer delovi sa eksponentima mogu ispasti van opsega mašinske preciznosti što se može rešiti uvođenjem sledećeg poluprstena.

Zbir-poizvod poluprsten na log-domenu definiše se kao $\left(\mathbb{R}^*,\oplus,\otimes,-\infty,0\right)$, pri čemu je $\mathbb{R}^* = \mathbb{R}\cup -\infty$, a operacije su definisane sa $a\oplus b = \log(\mathrm{e}^a+\mathrm{e}^b)$ i $a\otimes b = a+b$ za svako $a,b\in\mathbb{R}^*$. U log-domenu, funkcije u_t imaju sledeći oblik:

$$u_t(y_{t-1},y_t) = \mathbf{w}^\mathsf{T}\boldsymbol{f}(y_{t-1},y_t,\boldsymbol{x},t), \quad t=1,\dots,T.$$

Logaritam normalizacione funkcije (A.6) može se izračunati kao

$$\log Z(\boldsymbol{x};\mathbf{w}) = \log\sum_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x})}\prod_{t=1}^{T}\mathrm{e}^{\mathbf{w}^\mathsf{T}\boldsymbol{f}(y_{t-1},y_t,\boldsymbol{x},t)} = \bigoplus_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x})}\bigotimes_{t=1}^{T}u_t(y_{t-1},y_t) = \bigoplus_{y_T}\alpha_T(y_T), \tag{A.15}$$

dok je logaritam m-tog parcijalnog izvoda jednak

$$\log \nabla_{\mathrm{w}_m} Z(\boldsymbol{x}; \mathbf{w}) = \bigoplus_{k=1}^{T} \bigoplus_{y_{\{k-1,k\}}} v_k(y_{k-1}, y_k) \otimes \log f_m(y_{k-1}, y_k, \boldsymbol{x}, k), \tag{A.16}$$

pri čemu se marginalne vrednosti $v_k(y_{k-1}, y_k)$ računaju prema formuli (A.11). Računanje gradijenta uslovnih slučajnih polja nad log domenom predstavljeno je u algoritmu 1 u odeljku 2.

A.2 Treniranje strukturnih klasifikatora na primerima sa najvećim gubicima

Opšta ideja korišćenja pristupa obučavanja koje uključuje k najvećih gubitaka je ograničavanje procesa učenja sa izlaznog prostora na njegov podskup sastavljen samo od k elemenata sa najvećim gubitkom, iz razloga što uključenje celog izlaznog skupa nije moguće zbog njegove eksponencijalne veličine. Ovde ćemo prezentovati četiri nove verzije (Mančev and Todorović, 2015) algoritama za treniranje strukturnih klasifikatora bazirane na ovom pristupu.

Sekvencijalni dualni metod sa k najboljih struktura Sekvencijalni dualni metod za strukturne klasifikatore originalno je uveden u radu (Balamurugan et al., 2011) i on koristi samo najbolju strukturu za dopunu trenutnog skupa ograničenja. Ovaj metod možemo modifikovati da radi sa više struktura. Ukoliko transformišemo optimizacioni problem za mašine sa vektorima podrške (A.5) iz primarnog u dualni prostor, dobijamo sledeći optimizacioni problem

$$\min_{\boldsymbol{\lambda}} \frac{1}{2}\boldsymbol{\lambda}^{\mathsf{T}}\boldsymbol{K}\boldsymbol{\lambda} - \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{L} \qquad \text{s.t.} \sum_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \lambda_{n,\boldsymbol{y}} = C,\ \forall n, \quad \lambda_{n,\boldsymbol{y}} \geq 0,\ \forall n,\ \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n), \tag{A.17}$$

gde je $\boldsymbol{K}$ matrica definisana elementima $K_{n,\boldsymbol{y},m,\boldsymbol{y}'} = \Delta\mathbf{F}_n(\boldsymbol{y})^{\mathsf{T}}\Delta\mathbf{F}_m(\boldsymbol{y}')$ za svako $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$, $\boldsymbol{y}' \in \mathscr{Y}(\boldsymbol{x}^m)$ i $\mathbf{L}$ je vektor sa elementima $L_{n,\boldsymbol{y}} = L(\boldsymbol{y}^n, \boldsymbol{y})$. S obzirom na veliki broj ograničenja u problemu (A.17), sekvencijalni dualni metod zasniva se na ideji da je moguće sekvencijalno dodavati ograničenja i razmatra sledeći potproblem na n-tom primeru

$$\min_{\boldsymbol{\alpha}_n} \frac{1}{2}\boldsymbol{\alpha}_n^{\mathsf{T}}\mathbf{K}_n\boldsymbol{\alpha}_n - \boldsymbol{\alpha}_n^{\mathsf{T}}(\mathbf{L}_n - \Delta\mathbf{F}^{\mathsf{T}}\mathbf{w}) \quad \text{s.t.} \sum_{\boldsymbol{y}\in\mathscr{W}_n} \alpha_{n,\boldsymbol{y}} = 0,\ \alpha_{n,\boldsymbol{y}} \geq -\lambda_{n,\boldsymbol{y}},\ \forall \boldsymbol{y} \in \mathscr{W}_n, \tag{A.18}$$

gde se se skup struktura $\mathscr{W}_n = \{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n) \ : \ \lambda_{n,\boldsymbol{y}} > 0\}$ inkrementalno kreira dodajući strukturu $\widetilde{\boldsymbol{y}}^n$ prema (A.4) na n-tom primeru, a $\boldsymbol{K}_n$ predstavlja blok matrice $\boldsymbol{K}$ koji odgovara n-tom primeru. Možemo posmatrati proširenje ovog metoda za rad sa k najboljih struktura

koju ćemo objasniti u nastavku. Algoritam prolazi kroz primere i na n-tom primeru dopunjuje svoj skup ograničenja sa k struktura sa trenutno najvećim gubitkom. Nakon toga, sekvencijalna minimalna optimizacija (SMO) (Platt, 1999) se primenjuje za optimizaciju dualnih promenljivih koje odgovaraju strukturama iz $\mathscr{W}_n$. Metod bira par struktura iz skupa $\mathscr{W}_n$ na osnovu strategije para koji najviše narušava ograničenja, nakon čega primenjuje optimizaciju sve dok Karush-Kuhn-Tuckerovi (KKT) uslovi (Kuhn and Tucker, 1951) ne budu zadovoljeni na skupu $\mathscr{W}_n$. Pseudokod je predstavljen u algoritmu 3.

LaRank algoritam sa k najboljih struktura U ovom paragrafu predstavićemo proširenje LaRank algoritma (Bordes et al., 2008) za slučaj sa k najvećih gubitaka (Mančev and Todorović, 2015). Ovaj algoritam je dizajniran za treniranje u jednom prolasku kroz podatke. Primer $[(\boldsymbol{x}^n, \boldsymbol{y})]_{\boldsymbol{y} \in \mathscr{Y}(x^n)}$ koji ima sve dualne parametre $\lambda_{n,\boldsymbol{y}} = 0$ jednake nuli zvaćemo *novi primer*, u suprotnom koristićemo naziv *viđen primer*. Verzija LaRank algoritma sa k najboljih struktura koristi sledeće operacije:

kBV-PROCESSNEW Izaberemo novi primer $[(\boldsymbol{x}^n, \boldsymbol{y})]$ i optimizujemo koeficijente koji odgovaraju k-najboljim strukturama zajedno sa pravom strukturom $\boldsymbol{y}^n$,

kBV-PROCESSOLD Slučajnim izborom odaberemo viđen primer i optimizujemo njegove koeficijente koji su različiti od nule zajedno sa k struktura sa trenutno najvećim gubitkom

kBV-OPTIMIZE Slučajnim izborom izaberem viđen primer i optimizujemo njegove koeficijente koji su različiti od nule.

Za optimizaciju algoritam koristi SMO i primenjuje prethodne operacije naizmenično po unapred definisanom redosledu. Ovaj redosled može biti isti kao i u verziji predstavljenoj u (Bordes et al., 2008), gde posle primene kBV-PROCESSNEW koraka sledi odgovarajući broj od n_R REPROCESS operacija, pri čemu REPROCESS operacija označava primenu deset kBV-OPTIMIZE nakon jednog kBV-PROCESSOLD koraka. Pseudokod je dat u algoritmu 4.

Pasivno-agresivni algoritam nad k najboljih struktura Pasivno-agresivni algoritam je onlajn algoritam uveden u (Crammer et al., 2006) koji rešava optimizacioni problem (3.6)-(3.7) za $k = 1$, tj. koristeći samo ograničenje za strukturu sa najvećim gubitkom. S obzirom na to da postoji samo jedno ograničenje pored ograničenja za originalnu strukturu $\boldsymbol{y}^n$, koristeći metod Lagranžovih multiplikatora moguće je odrediti veličinu koraka u analitičkom obliku. U radu (Mančev and Todorović, 2015) je predstavljena verzija ovog algoritma koja uključuje k struktura sa najvećim gubitkom i koja optimizuje odgovarajuće parametre sekvencijalno unutar primera. Označimo sa $Err_{\mathbf{w}_n} = \{\boldsymbol{y} \in \mathscr{Y}_{-n} : \ell(\mathbf{w}_n; (\boldsymbol{x}^n, \boldsymbol{y})) > 0\}$ skup

struktura na kojima je funkcija gubitka sa datim parametrima veća od nule. Ovaj skup ćemo koristiti za utvrđivanje da li strukturu treba koristiti za promenu parametara.

Ovde ćemo definisati niz optimizacionih problema unutar jednog primera, pri čemu svaki od njih koristi jednu od k struktura sa najvećim gubitkom. S obzirom na to da u svakom trenutku razmatramo samo jednu strukturu, svaki optimizacioni problem se može rešiti analitički. Dakle, mi ćemo sekvencijalno prolaziti kroz strukture sa najvećim gubitkom, optimizovati potproblem ograničen samo na jednu strukturu

$$\min_{\xi,\mathbf{w}_n^{j+1}} \frac{1}{2}\left\|\mathbf{w}_n^{j+1}-\mathbf{w}_n^j\right\|^2+C\xi \quad \text{s.t. } {\mathbf{w}_n^{j+1}}^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)}) \geq L(\boldsymbol{y}^n,\boldsymbol{y}^{(n,j)})-\xi,\ \xi\geq 0 \tag{A.19}$$

za $j=1,\ldots,k$, i upotrebiti pasivno-agresivnu promenu parametara

$$\mathbf{w}_n^{j+1}=\mathbf{w}_n^j+\delta_{n,j}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)}), \quad j=1,\ldots,k, \tag{A.20}$$

pri čemu se svaki korak može naći kao

$$\delta_{n,j}=\min\left\{\frac{\ell(\mathbf{w}_n^j;(\boldsymbol{x}^n,\boldsymbol{y}^{(n,j)}))}{\left\|\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)})\right\|^2},C\right\}. \tag{A.21}$$

Prethodnim dvema formulama definisana je promena parametara za k-najbolji pasivno-agresivni algoritam pretpostavljajući da smo počeli obrađivati n-ti primer sa parametrima $\mathbf{w}_n$, koji su korišćeni za dobijanje k sekvenci sa najvećim gubitkom $\mathscr{B}_{\mathbf{w}_n}^{k,n}=(\boldsymbol{y}^{(n,1)},\ldots,\boldsymbol{y}^{(n,k)})$. Parametri $\mathbf{w}_n^j$ označavaju međustanja između početnih parametara $\mathbf{w}_n=\mathbf{w}_n^1$ i parametara nakon završene optimizacije na n-tom primeru $\mathbf{w}_{n+1}=\mathbf{w}_n^{k+1}$.

Da bismo odredili kumulativnu grešku predviđanja za uvedeni algoritam, moraćemo da primenimo dodatno ograničenje za promenu parametara. Definišimo prvo pomoćni skup

$$A_{\mathbf{w}_n}=\{\boldsymbol{y}\in\mathscr{Y}_{-n}:\ell(\mathbf{w}_n;(\boldsymbol{x}^n,\boldsymbol{y}))\geq\ell(\mathbf{w}_n;(\boldsymbol{x}^n,\widehat{\boldsymbol{y}}_{\mathbf{w}_n}))\}, \tag{A.22}$$

pri čemu je $\widehat{\boldsymbol{y}}_{\mathbf{w}_n}=\arg\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)}\ \mathbf{w}_n^{\mathsf{T}}\mathbf{F}(\boldsymbol{x}^n,\boldsymbol{y})$, koji ćemo iskoristiti za definisanje *k-najbolje ograničene pasivno-agresivne* promene parametara

$$\mathbf{w}_n^{j+1}=\begin{cases}\mathbf{w}_n^j+\delta_{n,j}\Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)}) & ,\ \text{if } \boldsymbol{y}^{(n,j)}\in A_{\mathbf{w}_n^j}\\ \mathbf{w}_n^j & ,\ \text{if } \boldsymbol{y}^{(n,j)}\notin A_{\mathbf{w}_n^j}\end{cases},\ j=1,\ldots,k, \tag{A.23}$$

da bismo definisali ograničenu verziju pasivno-agresivnog algoritma nad k najvećih gubitaka (algoritam 5). Označimo sa $\mathscr{W}_n=(\boldsymbol{y}^{(n,i_1)},\ldots,\boldsymbol{y}^{(n,i_{|\mathscr{W}_n|})})$ niz struktura iz skupa k najboljih struktura $\mathscr{B}_{\mathbf{w}_n}^{k,n}$ na kojima pravimo promenu parametara različitu od nule, za neke indekse $1\leq i_1<\ldots<i_{|\mathscr{W}_n|}\leq k$. U daljem tekstu, zbog jednostavnosti, označavaćemo ove sekvence sa

$\mathscr{W}_n = (\boldsymbol{y}^{(n,1)}, \ldots, \boldsymbol{y}^{(n,|\mathscr{W}_n|)})$.

Lema 1. *Neka je* $(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$ *niz primera i neka je* $\mathbf{u}$ *proizvoljan vektor. Za* k*-najbolju ograničenu promenu parametara definisanu sa (A.23) važi*

$$\sum_{n=1}^{N} \sum_{j=1}^{|\mathscr{W}_n|} \delta_{n,j} \left(2\ell(\mathbf{w}_n^j; (\boldsymbol{x}^n, \boldsymbol{y}^{(n,j)})) - \delta_{n,j} \left\| \Delta \mathbf{F}_n(\boldsymbol{y}^{(n,j)}) \right\|^2 - 2\ell(u; (\boldsymbol{x}^n, \boldsymbol{y}^{(n,j)})) \right) \leq \|\mathbf{u}\|^2,$$

pri čemu skup $\mathscr{W}_n = (\boldsymbol{y}^{(n,1)}, \ldots, \boldsymbol{y}^{(n,|\mathscr{W}_n|)})$ *sadrži sve sekvence iz* $\mathscr{B}^{k,n}_{\mathbf{w}_n^1}$ *za koje je promena parametara definisana sa (A.23) različita od nule.*

Teorema 1. *Neka je* $(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$ *niz primera, pri čemu je* $\|\Delta \mathbf{F}_n(\boldsymbol{y})\| \leq 1$ *i* $L(\boldsymbol{y}^n, \boldsymbol{y}) \leq C$ *za svako* $\boldsymbol{y} \in \mathscr{W}_n$*,* $n = 1, \ldots, N$*. Tada za proizvoljan vektor* $\mathbf{u}$ *i* k*-najbolju ograničenu promenu definisanu sa (A.23) sledi*

$$\sum_{n=1}^{N} \sum_{j=1}^{|\mathscr{W}_n|} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}_{\mathbf{w}_n^j})^2 \leq \|\mathbf{u}\|^2 + 2C \sum_{n=1}^{N} \sum_{j=1}^{|\mathscr{W}_n|} \ell(\mathbf{u}; (\boldsymbol{x}^n, \boldsymbol{y}^{(n,j)})), \tag{A.24}$$

pri čemu skup $\mathscr{W}_n = (\boldsymbol{y}^{(n,1)}, \ldots, \boldsymbol{y}^{(n,|\mathscr{W}_n|)})$ *sadrži sve sekvence iz* $\mathscr{B}^{k,n}_{\mathbf{w}_n^1}$ *za koje je promena parametara definisana sa (A.23) različita od nule.*

Dokaz leme 1 i teoreme 1 dat je u odeljku 3.2.4.

Posledica 1. *Neka su uslovi prethodne teoreme zadovoljeni, tada važi*

$$\sum_{n=1}^{N} \sum_{j=1}^{|\mathscr{W}_n|} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}_{\mathbf{w}_n^j})^2 \leq \|\mathbf{u}\|^2 + 2C \sum_{n=1}^{N} |\mathscr{W}_n| \ell_n(\mathbf{u}), \tag{A.25}$$

gde je $\ell_n(\mathbf{u}) = \max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{u}; (\boldsymbol{x}^n, \boldsymbol{y}))$.

Provera da li struktura pripada skupu $A_{\mathbf{w}_n}$ zahteva nalaženje $\widehat{\boldsymbol{y}}$ prilikom svake promene parametara što može biti vremenski zahtevno. U eksperimentalnom delu, mi ćemo razmatrati verziju sa ograničenim korakom kao i verziju bez ograničenja, iako za nju neće važiti prethodno pokazana granica za kumulativni gubitak predviđanja.

k-najbolji perceptron Kada se ne koristi funkcija cene, tj. kada je $L(\boldsymbol{y}^n, \boldsymbol{y}) = 0$ za svako $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$, pasivno-agresivni algoritam se svodi na perceptron algoritam uveden u radu (Collins, 2002) koji koristi konstantni korak u optimizaciji. Mi uvodimo k-najbolju verziju ovog algoritma koja će sekvencijalno prolaziti kroz k struktura sa najvećim gubitkom $(\boldsymbol{y}^{(n,1)}, \ldots, \boldsymbol{y}^{(n,k)}) = \mathscr{B}^{k,n}_{\mathbf{w}_n}$ i sekvencijalno menjati parametre konstantnim korakom za svaku strukturu koja pripada skupu grešaka, tj. za svaku strukturu koja ima veću težinu od

originalne. Nakon nalaženja k struktura sa najvećim gubitkom, primenjuje se sledeća serija promena (Mančev and Todorović, 2015)

$$\mathbf{w}_n^{j+1} = \begin{cases} \mathbf{w}_n^j + \Delta\mathbf{F}_n(\boldsymbol{y}^{(n,j)}) & , \text{ if } \boldsymbol{y}^{(n,j)} \in Err_{\mathbf{w}_n^j} \\ \mathbf{w}_n^j & , \text{ if } \boldsymbol{y}^{(n,j)} \notin Err_{\mathbf{w}_n^j} \end{cases} , \quad j = 1, \ldots, k, \tag{A.26}$$

gde je $Err_{\mathbf{w}_n} = \{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n) : \mathbf{w}_n^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y}) \leq 0\}$. Uslovi pod kojima verzija perceptrona sa k najboljih struktura konvergira su isti kao u slučaju standardne verzije.

Teorema 2. *Neka je $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$ skup podataka za obučavanje i neka postoji vektor $\mathbf{u}$ i broj $\gamma > 0$ tako da je $\|\mathbf{u}\| = 1$ i $\mathbf{u}^\mathsf{T}\Delta\mathbf{F}_n(\boldsymbol{y}) \geq \gamma$ za svaki podatak n i sve strukture $\boldsymbol{y} \in \mathscr{Y}_{-n}$. Za verziju perceptron algoritma sa k najboljih struktura iz algoritma 5, sledi da je*

$$\textit{Broj grešaka } \leq \frac{R^2}{\gamma^2},$$

gde je R broj za koji važi $\|\Delta\mathbf{F}_n(\boldsymbol{y})\| < R$, $\forall n, \forall \boldsymbol{y} \in \mathscr{Y}_{-n}$.

Teorema 3. *Neka je $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$ skup podataka za obučavanje, $\mathbf{u}$ proizvoljan vektor kod koga je $\|\mathbf{u}\| = 1$ i neka je $\gamma > 0$. Definišimo*

$$m_n = \mathbf{u}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}^n) - \max_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \mathbf{u}^\mathsf{T}\mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}), \quad \epsilon_n = \max\{0, \gamma - m_n\},$$

i $D = \sqrt{\sum_{n=1}^N \epsilon_n^2}$. Za prvi prolazak perceptron algoritma sa k najboljih struktura iz algoritma 5 kroz podatke za obučavanje važi da je

$$\textit{Broj grešaka } \leq \left(\frac{R+D}{\gamma}\right),$$

gde je R broj za koji važi $\|\Delta\mathbf{F}_n(\boldsymbol{y})\| < R$, $\forall n, \forall \boldsymbol{y} \in \mathscr{Y}_{-n}$.

Dokaz teoreme 2 i teoreme 3 dat je u odeljku 3.2.5.

Eksperimentalni rezultati Prezentovali smo rezultate uvedenih algoritama na dva problema klasifikovanja sekvenci: plitko parsiranje i prepoznavanje entiteta. Algoritme smo implementirali u jeziku C++, a za testiranje smo koristili kompjuter sa procesorom Intel Core 2 Duo CPU 2.33 GHz i 8 GB RAM. Da bismo izbegli oscilacije rezultata u procesu učenja, primenili smo usrednjavanje parametara opisano u (Collins, 2002). Slika 3.2 u odeljku 3 prezentuje vremensko poređenje predstavljenih algoritama. Perceptron kao najjednostavniji metod je malo brži od pasivno-agresivnog algoritma, ali na slici vidimo da drugi algoritmi koji uključuju funkciju cene daju bolje rezultate. Sekvencijalni dualni metod zahteva značajno više vremena za obučavanje zbog stalnog dodavanja novih ograničenja.

Sledeće testiranje je urađeno za LaRank algoritam treniran u jednom prolasku. Rezultate smo prikazali na slici 3.3 za različite vrednosti parametara k i n_R. Kako su primeri u REPROCESS operaciji dobijeni slučajnim izborom, prikazali smo rezultate kao srednju vrednost F-mere sa odgovarajućom srednjom apsolutnom greškom. Za selekciju regularizacionog parametara C, primenili smo tehniku petostruke unakrsne validacije za problem plitkog parsiranja, dok smo za problem prepoznavanja entiteta koristili poseban skup podataka namenjen podešavanju modela. Dobijeni rezultati sugerišu prednost verzije sa k najboljih struktura, naročito u slučaju nižih vrednosti parametra n_R. Takođe, možemo videti da povećanje parametra n_R ima pozitivan uticaj na F-meru.

U tabeli 3.1 u odeljku 3 prikazali smo rezultate za različite vrednosti parametra k, pri čemu su ostali parametri (regularizacioni parametar, broj trening iteracija) selektovani unakrsnom validacijom za problem plitkog parsiranja, a određeni na pomoćom skupu podataka za problem prepoznavanja entiteta. Postavili smo toleranciju $\tau = 10^{-10}$ za ispitivanje KKT uslova. Za slučaj kada je $k = 10$, obično su dobijene niže vrednosti F-mere u odnosu na ostale vrednosti, najverovatnije zato što uključenje velikog broja karakteristika iz k najboljih struktura može da dovede do problema prevelikog prilagođavanja modela podacima za obučavanje (engl. overfitting). Isto tako, možemo da vidimo da su sve verzije sa k najboljih struktura dale bolje rezultate prepoznavanja u odnosu na verzije sa jednom najboljom strukturom, pri čemu su obično poboljšanja dobijena za parametre $k = 2$ i $k = 5$.

Razmatraćemo i statističku značajnost dobijenih poboljšanja. Testirali smo nultu hipotezu da ne postoje razlike između verzija sa k najboljih i jednom najboljom strukturom sa alternativnom hipotezom da postoje razlike između ovih verzija. Za tu svrhu primenili smo pristup za poređenje klasifikatora opisan u (Salzberg, 1997). U tabeli 3.2 predstavljene su matrice kontingencije dobijene primenom McNemarovog testa (McNemar, 1947). Sa pragom značajnosti od 0.05, iz tabele možemo da odbacimo nultu hipotezu za sve algoritme, osim za sekvencijalni dualni metod i MIRA. Zaključujemo da su poboljšanja za k-najbolji perceptron, pasivno-agresivni i ograničeni pasivno-agresivni algoritam u odnosu na verzije sa jednom najboljom strukturom statistički značajna sa datim pragom značajnosti.

A.3 Treniranje kombinacije dva modela zasnovano na poverenju

U prethodna dva odeljka razmatrali smo različite strukturne modele i njihove rezultate prepoznavanja. U cilju daljeg poboljšanja rezultata, mogu se posmatrati kombinacije strukturnih modela kao u radovima (Nguyen and Guo, 2007; Song and Sarkar, 2008). U ovom odeljku predložićemo kombinaciju dva strukturna modela koje ćemo trenirati u nizu, pri čemu

se predviđanja zajedno sa dodatnim poverenjem prosleđuju sa prvog modela na drugi.

Koristićemo kombinaciju dva strukturna modela, gde je prvi model strukturna mašina sa vektorima podrške, a drugi pripada uslovnim slučajnim poljima. Mašina sa vektorima podrške je trenirana korišćenjem sekvencijalnog dualnog metoda sa k najboljih struktura. Nakon nalaženja optimalnih parametara, model se koristi za generisanje n sekvenci sa najvećim težinama za drugi model na trening i test podacima. Pošto drugi model treba da koristi izlaz iz prvog modela, kako na trening tako i na test podacima, ne možemo jednostavno generisati ove izlaze nakon obučavanja jedne mašine sa vektorima podrške, s obzirom na to da će poverenje za izlazne podatke biti različito na viđenim i neviđenim primerima. Da bismo to rešili, podelili smo skup za obučavanje na 5 približno jednakih delova, slično kao u tehnici unakrsne validacije, i svaki put trenirali metod na svim delovima osim na jednom koji je korišćen za generisanje izlaza na neviđenim primerima. Izlazi su korišćeni za generisanje poverenja koje drugi model koristi kao dodatni ulaz. Uslovna slučajna polja su trenirana kvazi-Njutnovom metodom L-BFGS (Nocedal and Wright, 2006). Nakon treniranja uslovnih slučajnih polja, u fazi klasifikacije potrebno nam je još jedno dodatno treniranje prvog modela na celom skupu za obučavanje, koji će se koristiti za generisanje izlaza koji se prosleđuju drugom modelu. Na kraju, drugi model daje konačnu klasifikaciju test podataka. Slika 4.1 opisuje faze obučavanja i klasifikacije.

Neka su $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n)}$ sekvence sa najvećim težinama generisane prvim modelom na opservacionoj sekvenci $\boldsymbol{x}$. Da bismo definisali poverenje za predviđenu klasu, jedan pristup iz rada (Mejer and Crammer, 2010) je da dodelimo uniformnu značajnost svakom od datih n označavanja i da merimo njihovo slaganje sa predviđenom sekvencom. Ako je $\hat{\boldsymbol{y}}$ predviđena sekvenca klasa za opservaciju $\boldsymbol{x}$, poverenje ν_t klase na poziciji t će biti

$$\nu_t = \frac{|\{i : y_t^{(i)} = \hat{y}_t\}|}{n}. \tag{A.27}$$

Slika 4.2 ilustruje primer računanja poverenja predviđanja oko opservacije na poziciji t koristeći tri sekvence sa najvećim težinama. Neka C_t bude skup svih pozicija u kontekstu oko predviđanja na poziciji t. Mi ćemo razmatrati (Mančev and Todorović, 2012) poverenje predviđanja za određeni kontekst kao proizvod poverenja za individualne klase

$$c_t = \prod_{i \in \mathsf{C}_t} \nu_i. \tag{A.28}$$

Za kontekst na poziciji t koristili smo tri uzastopna predviđanja: trenutno, prethodno i naredno predviđanje, tj. $\mathsf{C}_t = \{t-1, t, t+1\}$. Nakon toga, poverenje predviđanja za određeni kontekst je diskretizovano u intervale, a zatim su kreirane karakteristike koje predstavljaju pripadanje nekom od intervala. Jedan od intervala treba da sadrži samo broj jedan koji će predstavljati

maksimalno poverenje za dati kontekst predikcija. Ovaj interval je dobar indikator da je predviđanje prvog modela korektno. Statistika pokazuje da je 97.7% predviđenja prvog modela tačno na korpusu CONLL-2000[1] kada imamo maksimalno poverenje za dati kontekst. Sa druge strane, ukoliko poverenje nije maksimalno, 23.6% predviđanja nije tačno i u tom slučaju drugi model treba da pokuša da popravi prepoznavanje prvog modela.

Rezultate smo predstavili na problemu plitkog parsiranja. Koristili smo vrednost $k = 5$ za treniranje sekvencijalnog dualnog metoda sa k najboljih sekvenci, a broj izlaznih sekvenci iz prvog modela je postavljen na vrednost $n = 5$. Model je treniran korišćenjem regularizacionog parametra $C = 0.1$ i tačnosti $\tau = 10^{-3}$ za narušavanje KKT uslova. Za svaku poziciju u sekvenci, tabela 4.1 prikazuje osnovne šablone korišćene za generisanje vektora karakteristika koji predstavljaju kombinaciju reči, njihovih atributa i klasa. U tabeli su takođe prikazani šabloni koji uključuju predviđanje i poverenje koji se mogu dodati osnovnim šablonima. Kao ulazna karakteristika drugog modela, poverenje za kontekst je izraženo preko pripadnosti intervalima [0,1) i [1,1], a za treniranje uslovnih slučajnih polja korišćena je implementacija *CRFsuite* (Okazaki, 2007) sa podrazumevanim podešavanjima. Izvršili smo poređenje individualnih modela sa osnovnim šablonima sa kombinacijom modela koja sadrži predviđanje prvog modela (bez poverenja) i kombinacijom modela sa uključenim predviđanjem i poverenjem. Tabela 4.2 u odeljku 4 prikazuje preciznost, odziv i F-meru. U tabeli vidimo da komitet dva modela daje bolje rezultate nego individualni modeli u pogledu preciznosti i odziva, dok je uključivanje poverenja uticalo na povećanje preciznosti bez efekta na odziv.

A.4 Strukturna klasifikacija sa dvostruko prelomljenom funkcijom gubitka

Mašine sa vektorima podrške (Vapnik, 1998) najčešće podrazumevaju konveksnu funkciju gubitka koja doprinosi jednostavnijoj optimizaciji i garantuje završetak u globalnom optimumu. Moguće je koristiti i nekonvenksne funkcije gubitka. Nekonveksna dvostruko prelomljena funkcija gubitka (engl. ramp loss) definisana je u (Collobert et al., 2006) za binarnu klasifikaciju, sa ciljem da se izbegne da primeri sa velikim zglobnim gubitkom postanu vektori podrške. Oni su koristili konkavno-konveksnu metodu (Yuille and Rangarajan, 2003) za optimizaciju nekonveksnog gubitka. Pored binarne klasifikacije, u radu (Do et al., 2008) je uveden nekonveksni gubitak za stukturnu klasifikaciju, gde je navedena prednost ovakvih modela u slučaju kada su klase pridružene sa šumom ili kada za jedan primer postoji veliki skup klasa koje su odgovarajuće isto koliko i originalna klasa. Do ove prednosti dolazi zbog činjenice da je dvostruko prelomljena funkcija gubitka ograničena sa gornje strane i da se

[1] http://www.cnts.ua.ac.be/conll2000/chunking

podaci sa šumom mogu odbaciti kada je gubitak suviše veliki, što nije slučaj sa neograničenom zglobnom funkcijom gubitka.

U ovom odeljku, uvodimo sekvencijalni dualni metod za optimizaciju dvostruko prelomljene funkcije gubitka (Mančev, 2015). Nakon primene konkavno-konveksne metode, konveksni problem je optimizovan u dualnom prostoru prolazeći sekvencijalno kroz primere za obučavanje. Počinjemo sa optimizacijom regularizovanog empirijskog rizika (A.3) nad skupom podataka $\mathscr{D}$, pri čemu ćemo za $\ell_n(\mathbf{w})$ posmatrati dvostruko prelomljenu funkciju gubitka definisanu sa

$$\begin{aligned}\ell_n^{\text{Ramp}}(\mathbf{w}) &= \ell_n^{\text{MM}}(\mathbf{w}) + \ell_n^{\text{C}}(\mathbf{w}) = \max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) - \max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) \\ &= \ell(\mathbf{w};(\boldsymbol{x}^n,\widetilde{\boldsymbol{y}}_{\mathbf{w}}^n)) - \ell_C(\mathbf{w};(\boldsymbol{x}^n,\overline{\boldsymbol{y}}_{\mathbf{w}}^n))\,,\end{aligned}$$

pri čemu je[2]

$$\ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) = \max\{0, -L(\boldsymbol{y}^n,\boldsymbol{y}) - \mathbf{w}^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y})\},\ \ell_n^{\text{C}}(\mathbf{w}) = -\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y}))\,.$$

Strukture $\widetilde{\boldsymbol{y}}_{\mathbf{w}}^n$ i $\overline{\boldsymbol{y}}_{\mathbf{w}}^n$, u kojima indeks $\mathbf{w}$ može da se izostavi kada ne postoji opasnost da dođe do konfuzije, definisane su kao

$$\widetilde{\boldsymbol{y}}^n \equiv \widetilde{\boldsymbol{y}}_{\mathbf{w}}^n = \arg\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})), \quad \overline{\boldsymbol{y}}^n \equiv \overline{\boldsymbol{y}}_{\mathbf{w}}^n = \arg\max_{\boldsymbol{y}\in\mathscr{Y}(\boldsymbol{x}^n)} \ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y}))\,. \tag{A.29}$$

Važne karakteristike dvostruko prelomljene funkcije greške pokazane u (Do et al., 2008) su da je ona odozgo ograničena zglobnom funkcijom i da leži između cena za optimalne strukture iz (A.29)

$$\ell_n^{\text{Ramp}}(\mathbf{w}) \le \ell_n^{\text{MM}}(\mathbf{w}), \quad 2L(\boldsymbol{y}^n,\overline{\boldsymbol{y}}^n) \le \ell_n^{\text{Ramp}}(\mathbf{w}) \le 2L(\boldsymbol{y}^n,\widetilde{\boldsymbol{y}}^n). \tag{A.30}$$

Na slici 5.1 u odeljku 5 ilustrovali smo dvostruko prelomljenu funkciju gubitka u binarnom i strukturnom slučaju. S obzirom na njenu nekonveksnost, za optimizaciju se može koristiti konveksno-konkavna metoda (Yuille and Rangarajan, 2003), koja omogućava optimizuju funkcije koja se može predstaviti kao suma konveksnog i konkavnog dela. U svakoj iteraciji ona aproksimira konkavnu komponentu koristeći Tejlorov razvoj do prvog stepena sa trenutnim parametrima, a zatim nalazi sledeće parametre kao optimalno rešenje za sumu konveksnog i linearizovanog konkavnog dela funkcije. Ukoliko imamo funkciju cilja $J(\mathbf{w})$, koja se može predstaviti kao suma konveksne $J_{\text{vex}}(\mathbf{w})$ i konkavne funkcije $J_{\text{cav}}(\mathbf{w})$,

[2]Ovo je samo jedna od mogućih definicija funkcije $\ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y}))$ koja je predstavljena u (Gimpel, 2012). U radu (Do et al., 2008) koristi se verzija gde je $\ell_C(\mathbf{w};(\boldsymbol{x}^n,\boldsymbol{y})) = \max\{0, -\mathbf{w}^{\mathsf{T}}\Delta\mathbf{F}_n(\boldsymbol{y})\}$.

korišćenjem ovog metoda dobijamo sledeći niz optimizacionih problema:

$$\mathbf{w}^{t+1} = \arg\min_{\mathbf{w}} J_t(\mathbf{w}, \mathbf{w}^t) = \arg\min_{\mathbf{w}} \left\{ J_{\text{vex}}(\mathbf{w}) + \mathbf{w}^\mathsf{T} J'_{\text{cav}}(\mathbf{w}^t) \right\}.$$

U slučaju dvostruke prelomljene funkcije gubitka sa dodatom regularizacionom funkcijom, prethodni niz iteracija svodi se na

$$\mathbf{w}^{t+1} = \arg\min_{\mathbf{w}} \left\{ \underbrace{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \ell_n^{\text{MM}}(\mathbf{w})}_{J_{\text{vex}}(\mathbf{w})} + \mathbf{w}^\mathsf{T} \underbrace{C\sum_{n=1}^{N} \partial_{\mathbf{w}} \ell_n^{\mathsf{C}}(\mathbf{w}^t)}_{J'_{\text{cav}}(\mathbf{w}^t)} \right\}, \tag{A.31}$$

gde je $\partial_{\mathbf{w}} \ell_n^{\mathsf{C}}(\mathbf{w}^t) = \Delta\mathbf{F}_n(\overline{\boldsymbol{y}}^n_{\mathbf{w}^t}) \mathbb{1}[n \in \mathcal{A}_{\mathbf{w}^t}]$, $\mathcal{A}_{\mathbf{w}^t} = \{n : \ell_C(\mathbf{w}^t; \left(\boldsymbol{x}^n, \overline{\boldsymbol{y}}^n_{\mathbf{w}^t}\right)) > 0\}$, $\mathbb{1}$ predstavlja indikatorsku funkciju čija je vrednost nula (jedan) ukoliko njen argument ima netačnu (tačnu) vrednost, $C = 1/(\lambda N)$, a struktura $\overline{\boldsymbol{y}}^n_{\mathbf{w}^t}$ je dobijena korišćenjem parametara $\mathbf{w}^t$. Svaki minimizacioni problem (A.31) može se transformisati u optimizacioni problem sa ograničenjima

$$\mathbf{w}^{t+1} = \arg\min_{\mathbf{w}} J_t(\mathbf{w}, \mathbf{w}^t) = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \xi_n + C \sum_{n \in \mathcal{A}_{\mathbf{w}^t}} \mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\overline{\boldsymbol{y}}^n_{\mathbf{w}^t}) \tag{A.32}$$

$$\text{s.t.} \quad \mathbf{w}^\mathsf{T} \Delta\mathbf{F}_n(\boldsymbol{y}) \geq L(\boldsymbol{y}^n, \boldsymbol{y}) - \xi_n, \quad \forall n, \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n). \tag{A.33}$$

Uvođenjem Lagranžovih multiplikatora $\lambda_{n,\boldsymbol{y}} \geq 0$ za svaku strukturu unutar n-tog primera, dobijamo ekvivalentni dualni problem

$$\min_{\boldsymbol{\lambda}} \frac{1}{2}\boldsymbol{\lambda}^\mathsf{T} \boldsymbol{K} \boldsymbol{\lambda} - \boldsymbol{\lambda}^\mathsf{T} \mathbf{L} - \boldsymbol{\lambda}^\mathsf{T} \Delta\mathbf{F}^\mathsf{T} \mathbf{v}, \quad \text{s.t.} \sum_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \lambda_{n,\boldsymbol{y}} = C,\ \forall n,\ \lambda_{n,\boldsymbol{y}} \geq 0, \forall n, \forall \boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n),$$

gde je $\boldsymbol{K}$ matrica definisana elementima $K_{n,\boldsymbol{y},m,\boldsymbol{y}'} = \Delta\mathbf{F}_n(\boldsymbol{y})^\mathsf{T} \Delta\mathbf{F}_m(\boldsymbol{y}')$ za svako $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$, $\boldsymbol{y}' \in \mathscr{Y}(\boldsymbol{x}^m)$, vektor $\mathbf{L}$ je definisan elementima $L_{n,\boldsymbol{y}} = L(\boldsymbol{y}^n, \boldsymbol{y})$, pri čemu je $n, m \in \{1, \ldots, N\}$, dok je veza između primarnih i dualnih parametara definisana sa

$$\mathbf{w} = \mathbf{u} - \mathbf{v}, \quad \mathbf{u} = \sum_{n=1}^{N} \sum_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \lambda_{n,\boldsymbol{y}} \Delta\mathbf{F}_n(\boldsymbol{y}), \quad \mathbf{v} = C \sum_{n \in \mathcal{A}_{\mathbf{w}^t}} \Delta\mathbf{F}_n(\overline{\boldsymbol{y}}^n_{\mathbf{w}^t}).$$

Kako je prethodni dualni problem konveksan, možemo primeniti sekvencijalni dualni metod (Balamurugan et al., 2011) za njegovu optimizaciju.

Eksperimentalni rezultati U ovom odeljku prezentujemo poređenje klasifikatora sa dvostruko prelomljenom i zglobnom funkcijom greške na problemu plitkog parsiranja i prepoznavanja vrsta reči. Takođe, prikazaćemo rezultate na veštački generisanim podacima sa

dodatim šumom. Pored prethodno opisanog sekvencijalnog dualnog metoda za dvostruko prelomljenu funkciju greške, u poređenje ćemo uključiti i Pegasos algoritam (Shalev-Shwartz et al., 2011). Da bismo izbegli oscilacije tokom procesa učenja, primenili smo usrednjavanje parametara opisano u (Xu, 2011). Za izbor regularizacionog parametra koristili smo k-struku unakrsnu validaciju sa parametrom $k = 5$. Tabela 5.1 opisuje rezultate oba metoda na datim problemima zajedno sa izabranim parametrima tokom unakrsne validacije. Zavisnost rezultata od broja epoha u procesu obučavanja predstavljena je na slici 5.2. Iz tabele 5.1 i slike 5.2 možemo da zaključimo da su rezultati za obe funkcije gubitka na oba problema veoma slični. Sekvencijalni dualni metod brže dostiže maksimalnu F-meru, nakon samo nekoliko epoha, dok Pegasos algoritam bez projekcije daje bolje rezultate na oba problema. Rezultati kroz epohe su veoma slični za obe funkcije gubitka i na datim problemima ne možemo da vidimo prednost dvostruko prelomljene funkcije gubitka. Jedino Pegasos algoritam bez projekcije daje za nijansu bolje rezultate sa dvostruko prelomljenom funkcijom gubitka na problemu plitkog parsiranja.

S obzirom na to da dvostruko prelomljena funkcija gubitka treba da bude korisna u slučaju podataka sa šumom, testiraćemo metode na takvim podacima. Za tu svrhu, modifikovali smo originalni skup podataka za obučavanje za problem plitkog parsiranja, menjajući klase slučajno izabranih primera. Na taj način kreirali smo četiri skupa, gde smo slučajnim izborom izabrali 5, 10, 15 i 20 procenata od ukupnog broja primera i slučajnim izborom promenili odgovarajuće klase u tim primerima (sekvencama).

Slika 5.3 prikazuje rezultate zglobne i dvostruko prelomljene funkcije gubitka na generisanim skupovima kada se za optimizaciju koristi sekvencijalni dualni metod i Pegasos algoritam. Možemo primetiti da svi algoritmi sa dvostruko prelomljenom funkcijom gubitka daju bolje rezultate nego u slučaju korišćenja zglobne funkcije gubitka kada se u proces učenja uključe podaci sa šumom. Ovo su očekivani rezultati, jer je dvostruko prelomljena funkcija ograničena sa gornje strane, i podaci sa šumom ne mogu napraviti velike promene parametara kao što to može biti slučaj sa zglobnom funkcijom gubitka.

Možemo zaključiti da rezultati na podacima sa veštački dodatim šumom pokazuju jasnu prednost sekvencijalnog dualnog metoda i Pegasos algoritma sa dvostruko prelomljenom funkcijom naspram zglobne funkcije gubitka. Sa druge strane, rezultati na dva realna problema klasifikovanja sekvenci su veoma slični za obe funkcije gubitka, pa je u tom slučaju zglobna funkcija gubitka pogodnija za korišćenje jer za nju nije potrebno dodatno dekodiranje i zato što je njena optimizacije jednostavnija.

A.5 Primarni subgradijenti metod za optimizaciju usrednjene funkcije gubitka

U ovom odeljku uvodimo usrednjenu funkciju gubitka i primarni subgradijenti metod za njenu optimizaciju (Mančev and Todorović, 2014). Pokazaćemo da uvedeni algoritam ima najmanje istu granicu za kumulativni gubitak predviđanja u odnosu na stohastičku verziju algoritma sa zglobnom funkcijom gubitka. Eksperimenti su izvršeni na problemu plitkog parsiranja i prepoznavanja vrsta reči.

U prethodnim odeljcima razmatrali smo optimizacioni problem (A.3) sa različitim funkcijama gubitka. Ukoliko je funkcija gubitka diferencijabilna, kao što je to slučaj u uslovnim slučajnim poljima, možemo primeniti gradijentne metode za optimizaciju. Međutim, neke funkcije nisu diferencijabilne, npr. strukturna verzija zglobne funkcije, gde ne možemo da primenimo gradijentne metode. U tom slučaju, jedna opcija je primena subgradijentnih metoda. Subgradijent $\boldsymbol{g}$ funkcije $J(\mathbf{w})$ u tački $\mathbf{w}_0$ zadovoljava

$$J(\mathbf{w}) - J(\mathbf{w}_0) \geq \boldsymbol{g}^{\mathsf{T}}(\mathbf{w} - \mathbf{w}_0),$$

za svako $\mathbf{w}$. Ukoliko imamo konveksnu nediferencijabilnu funkciju, možemo primeniti subgradijentni metod (Shor et al., 1985) za njenu optimizaciju, na sličan način kao i gradijenti metod

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \eta_t \boldsymbol{g}^{(t)}$$

sa korakom η_t u t-toj iteraciji, pomerajući pritom parametre u pravcu subgradijenta $\boldsymbol{g}^{(t)}$ funkcije J u tački $\mathbf{w}^{(t)}$. Ukoliko je funkcija diferencijabilna u tački $\mathbf{w}^{(t)}$, onda je jedini izbor za $\boldsymbol{g}^{(t)}$ gradijent funkcije J u toj tački, pa se u tom slučaju subgradijentni metod svodi na gradijentni spust sa korakom η_t. Jedna od razlika subgradijentnog i gradijentnog metoda je u tome što subgradijentni metod koristi korak zadat unapred, dok gradijentni metodi mogu optimizovati korak u svakoj iteraciji. Uobičajene veličine koraka su $\eta_t = c$, $\eta_t = c/t$ i $\eta_t = c/\sqrt{t}$, gde je $c > 0$.

Ponovo počinjemo od minimizacije regularizovanog empirijskog rizika (A.3) na skupu $\mathscr{D} = ((\boldsymbol{x}^n, \boldsymbol{y}^n))_{n=1}^N$. Kako postoji puno izlaznih struktura za svaki primer, movžemo definisati funkciju gubitka za svaku strukturu posebno kao

$$\ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) = \max\left(0, L(\boldsymbol{y}^n, \boldsymbol{y}) - \mathbf{w}^{\mathsf{T}} \Delta\mathbf{F}_n(\boldsymbol{y})\right). \tag{A.34}$$

U ovom odeljku uvodimo usrednjenu funkciju gubitka (engl. average sum loss) (Mančev and

Todorović, 2014) $\ell_n^{\text{AS}}(\mathbf{w})$ definisanu sa

$$\ell_n^{\text{AS}}(\mathbf{w}) = \frac{1}{|\mathscr{Y}_{-n}|} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})), \tag{A.35}$$

koja predstavlja prosečni gubitak za strukturu n-tog primera. U slučaju da se ova funkcija koristi u formulaciji (A.3), imamo ekvivalentni optimizacioni problem sa ograničenjima

$$\min_{\mathbf{w}, \boldsymbol{\xi}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_{n=1}^{N} \frac{1}{|\mathscr{Y}_{-n}|} \sum_{\boldsymbol{y} \in \mathscr{Y}_{-n}} \xi_{n,\boldsymbol{y}} \tag{A.36}$$

$$\text{s.t. } \forall n, \forall \boldsymbol{y} \in \mathscr{Y}_{-n} : \mathbf{w}^{\mathsf{T}} \Delta \mathbf{F}_n(\boldsymbol{y}) \geq L(\boldsymbol{y}^n, \boldsymbol{y}) - \xi_{n,\boldsymbol{y}}, \quad \xi_{n,\boldsymbol{y}} \geq 0, \tag{A.37}$$

gde je jedna nenegativna pomoćna promenljiva (engl. slack variable) pridružena svakoj izlaznoj strukturi. Korišćenje jedne pomoćne promenljive po strukturi unutar jednog primera može se posmatrati kao generalizacija (Weston and Watkins, 1998) multiklasne mašine sa vektorima podrške u kojoj je po jedna pomoćna promenljiva dodeljena mogućim klasama unutar jednog primera.

Strukturni Pegasos algoritam sa usrednjenom funkcijom gubitka Posmatrajmo korišćenje usrednjene funkcije gubitka (A.35) u problemu (A.3) koji je aproksimiran sa

$$\frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{|A_t|} \sum_{n \in A_t} \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})), \tag{A.38}$$

gde je $B_n \subseteq \mathscr{Y}_{-n}$ i gde $A_t \subseteq \{1, \ldots, N\}$ sadrži primere koji su korišćeni za aproksimaciju. Nadalje, mi ćemo razmatrati prethodnu aproksimaciju ograničenu samo na n-ti primer, tj. gde je $A_t = \{n\}$, i definisati

$$f^{\text{AS}}(\mathbf{w}, B_n) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})). \tag{A.39}$$

Ovo ograničenje nam omogućava da dobijemo sekvencijalni algoritam kroz primere sa mini-grupnom (engl. mini-batch) optimizacijom unutar jednog primera prema izboru skupa B_n. Izbor ovog skupa dozvoljava da odaberemo koje strukture ćemo razmatrati u optimizacionom procesu. Posmatraćemo subgradijent aproksimirane kriterijumske funkcije (6.10) dat sa

$$\nabla^{\text{AS}} = \lambda \mathbf{w} - \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n^+} \Delta \mathbf{F}_n(\boldsymbol{y}), \tag{A.40}$$

gde je $B_n^+ = \{\boldsymbol{y} \in B_n : \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) > 0\}$. Prema tome, promena parametara u t-toj iteraciji na n-tom primeru biće

$$\mathbf{w}_{t+1} = (1 - \eta_t \boldsymbol{\lambda})\mathbf{w}_t + \frac{\eta_t}{|B_n|} \sum_{\boldsymbol{y} \in B_n^+} \Delta \mathbf{F}_n(\boldsymbol{y}), \tag{A.41}$$

nakon koje parametre možemo opciono projektovati na loptu radijusa $1/\sqrt{\boldsymbol{\lambda}}$ sledećim ažuriranjem

$$\mathbf{w}_{t+1} \leftarrow \min\left\{1, \frac{1/\sqrt{\boldsymbol{\lambda}}}{\|\mathbf{w}_{t+1}\|}\right\} \mathbf{w}_{t+1}. \tag{A.42}$$

Definišimo sledeći skup struktura

$$S_n = \{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n) : \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})) \geq \ell(\mathbf{w}; (\boldsymbol{x}^n, \widehat{\boldsymbol{y}}^n))\},$$

pri čemu je $\widehat{\boldsymbol{y}}^n$ dato sa

$$\widehat{\boldsymbol{y}}^n = \arg \max_{\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)} \mathbf{w}^\mathsf{T} \mathbf{F}(\boldsymbol{x}^n, \boldsymbol{y}). \tag{A.43}$$

Posmatraćemo takođe verziju Perceptron algoritma sa usrednjenom funkcijom gubitka gde izbor skupa B_n neće biti iz skupa svih struktura $\mathscr{Y}_{-n}$, već samo iz skupa S_n i metod sa takvim ograničenjem nazivaćemo ograničeni Pegasos algoritam. Način izbora skupa B_n iz S_n kardinalnosti k nije bitan za dalju analizu. Zapazimo da se izborom $B_n = \{\widetilde{\boldsymbol{y}}^n\}$ ovaj algoritam svodi na stohastičku verziju Perceptron algoritma (Shalev-Shwartz et al., 2007). Takođe, zapazimo da je moguće izabrati skup A_t kardinalnosti veće od jedan i u tom slučaju bi algoritam koristio više primera sa više struktura unutar svakog primera za jednu promenu parametara.

Razmotrimo sličnosti i razlike između Pegasos algoritma koji radi sa k struktura sa najvećim gubitkom i prethodno predstavljenog algoritma. Neka $\mathscr{B}_{\mathbf{w}}^{k,n}$ označava k struktura sa najvećim gubitkom na n-tom primeru. Možemo definisati usrednjeni gubitak ovih struktura kao

$$\ell_n^{k\text{best}}(\mathbf{w}) = \frac{1}{k} \sum_{\boldsymbol{y} \in \mathscr{B}_{\mathbf{w}}^{k,n}} \ell(\mathbf{w}; (\boldsymbol{x}^n, \boldsymbol{y})), \tag{A.44}$$

i odgovarajuću kriterijumsku funkciju ograničenu na n-ti primer kao

$$f^{Best_n^k}(\mathbf{w}) = f^{\text{AS}}(\mathbf{w}, \mathscr{B}_{\mathbf{w}}^{k,n}). \tag{A.45}$$

Na osnovu (A.45), možemo zaključiti da je prethodna kriterijumska funkcija specijalni slučaj aproksimacije kriterijumske funkcije (A.39) koja je napravljena na skupu struktura iz $\mathscr{B}_{\mathbf{w}}^{k,n}$. Takođe, vidimo da je usrednjeni gubitak k struktura sa najvećim gubitkom između usrednjene funkcije gubitka i zglobnog gubitka, tj. $\ell_n^{\text{AS}}(\mathbf{w}) \leq \ell_n^{k\text{best}}(\mathbf{w}) \leq \ell_n^{\text{MM}}(\mathbf{w})$. Ukoliko bismo

izabrali da B_n bude skup $\mathscr{B}_{\mathbf{w}}^{k,n}$, a ne proizvoljan podskup skupa S_n, takvu verziju mogli bismo da nazovemo k-najbolji Pegasos algoritam, jer će ona direktno optimizovati gubitak (A.44). Takođe, mi smo koristili k-najbolje dekodiranje da odredimo skup S_n. S obzirom na to da nam treba k izlaznih struktura (ukoliko postoje) sa gubitkom većim od gubitka za sekvencu $\widehat{\boldsymbol{y}}^n$, to možemo uraditi nalaženjem skupa $\mathscr{B}_{\mathbf{w}}^{k,n}$ i brisanjem svih struktura koje ne pripadaju skupu S_n. Pseudokod je dat u algoritmu 8.

U teorijskoj analizi posmatraćemo granicu kumulativnog gubitka predviđanja kroz iteracije između predviđene strukture $\widehat{\boldsymbol{y}}^n$ i prave strukture $\boldsymbol{y}^n$, tj. sumu $L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}^n)$.

Teorema 4. *Neka je $(\boldsymbol{x}^1, \boldsymbol{y}^1), \ldots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$ niz primera gde je $\|\Delta \mathbf{F}_n(\boldsymbol{y})\| \leq R$ i $L(\boldsymbol{y}^n, \boldsymbol{y}) \leq 1$ za svako $\boldsymbol{y} \in \mathscr{Y}(\boldsymbol{x}^n)$, $n = 1, \ldots, N$ i neka je $\mathbf{w}^* = \arg\min_{\mathbf{w}} f(\mathbf{w})$, gde je $f(\mathbf{w})$ definisana pomoću funkcije gubitka $\ell_n^{AS}(\mathbf{w})$. U tom slučaju, za promenu parametara (A.41) sa opcionim projekcionim korakom (A.42) sledi da je*

$$\frac{1}{N}\sum_{n=1}^{N} f^{AS}(\mathbf{w}_n, B_n) \leq \frac{1}{N}\sum_{n=1}^{N} f^{AS}(\mathbf{w}^*, B_n) + \frac{c(1+\ln N)}{2\lambda N},$$

gde je $c = (\sqrt{\lambda} + R)^2$ ukoliko koristimo projekcioni korak i $c = 4R^2$ u suprotnom slučaju.

Teorema 5. *Neka su zadovoljeni uslovi prethodne teoreme i neka je B_n izabran kao $B_n \subseteq S_n$. Tada sledi:*

$$\sum_{n=1}^{N} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}^n) \leq \frac{\lambda}{2} N \|\mathbf{w}^*\|^2 + \frac{c(1+\ln N)}{2\lambda} + \sum_{n=1}^{N} \frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}^*; (\boldsymbol{x}^n, \boldsymbol{y})). \tag{A.46}$$

Iz prethodne teoreme, korišćenjem nejednakosti

$$\frac{1}{|B_n|} \sum_{\boldsymbol{y} \in B_n} \ell(\mathbf{w}^*; (\boldsymbol{x}^n, \boldsymbol{y})) \leq \ell_n^{\text{MM}}(\mathbf{w}^*), \quad \forall B_n \subseteq \mathscr{Y}_{-n}, \tag{A.47}$$

dobijamo sledeću posledicu.

Posledica 2. *Neka su uslovi prethodne teoreme zadovoljeni. Tada sledi*

$$\sum_{n=1}^{N} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}^n) \leq \frac{\lambda}{2} N \|\mathbf{w}^*\|^2 + \sum_{n=1}^{N} \ell_n^{MM}(\mathbf{w}^*) + \frac{c(1+\ln N)}{2\lambda},$$

kao i

$$\sum_{n=1}^{N} L(\boldsymbol{y}^n, \widehat{\boldsymbol{y}}^n) \leq \sqrt{\frac{c(1+\ln N)}{N}} \|\mathbf{w}^*\| + \sum_{n=1}^{N} \ell_n^{MM}(\mathbf{w}^*), \quad \textit{birajući} \quad \lambda = \sqrt{\frac{c(1+\ln N)}{N\|\mathbf{w}^*\|^2}}.$$

Eksperimentalni rezultati U ovom odeljku prezentujemo rezultate uvedenog metoda na problemima plitkog parsiranja i prepoznavanja vrsta reči.

Na slici 6.1 prezentovan je uticaj regularizacionog parametra λ na rezultate prepoznavanja za oba problema. Male vrednosti regularizacionog parametra zahtevaju više iteracija, tj. duže treniranje, što je pomenuto i u radu (Shalev-Shwartz et al., 2011). Sa druge strane, visoke vrednosti regularizacionog parametra dovode do neprimetnih razlika između rezultata dobijenih posle obučavanja modela kroz 20 i 100 epoha, ali konačni rezultati u tom slučaju nisu zadovoljavajući. Najbolji rezultati na oba problema su postignuti u slučaju kada projekcioni korak nije korišćen u algoritmu. Kako bismo izabrali regularizacioni parametar za dalja testiranja, kao i ranije, upotrebili smo tehniku unakrsne validacije.

Sledeći rezultati prikazuju uticaj korišćenja ograničenja, gde B_n mora biti podskup skupa S_n, u odnosu na verziju bez korišćenja navedenog ograničenja. Na slici 6.2 vidimo da korišćenje ograničenja ne utiče značajno na rezultate prepoznavanja. Dalje, algoritam u t-toj iteraciji koristi korak $\eta_t = 1/(\lambda t)$ za promenu parametara. Postoje i druge opcije za izbor veličine koraka predložene u (Moulines and Bach, 2011), (Bottou, 2008) i (Ratliff et al., 2006). Na slici 6.3 predstavili smo rezultate u zavisnosti od izbora veličine koraka. Rezultati Pegasos algoritma su veoma slični rezultatima dobijenim za konstantnu veličinu koraka jednaku $\eta_t = 10^{-2}$ ili $\eta_t = \gamma/(\lambda\gamma t)^{\alpha}$ sa $\gamma = 10^{-1}$, kada se ne koristi projekcioni korak. Nasuprot rezultatima bez projekcionog koraka, kada se on koristi vidimo poboljšanje u prvih nekoliko epoha pri korišćenju drugih veličina koraka. U tom slučaju, nakon svih epoha, sve veličine koraka daju slične rezultate, osim kod veoma malog koraka $\eta_t = 10^{-3}$, pri čemu su najbolji rezultati dobijeni za $\eta_t = 10^{-1}$.

Usrednjavanje parametara doprinosi smanjenju oscilacija u rezultatima prepoznavanja. Na slici 6.4 vidimo oscilovanje rezultata kada usrednjavanje parametara nije uključeno. Slika 6.5 prikazuje rezultate Pegasos algoritma sa strukturnom zglobnom funkcijom i usrednjenom funkcijom gubitka. Na levoj slici možemo videti da oba algoritma konvergiraju ka istim vrednostima nezavisno od toga da li se koristi projekcioni korak ili ne. Kao što je očekivano, Pegasos algoritam sa usrednjenom funkcijom greške brže konvergira maksimalnim rezultatima u odnosu na broj epoha, s obzirom na to da koristi više struktura pri jednom ažuriranju parametara. Sa druge strane, kada poredimo rezultate u zavisnosti od broja promena parametara verzija sa zglobnom funkcijom gubitka konvergira brže jer ona uključuje više različitih primera u jednoj promeni parametara. Iako konvergiraju ka istim vrednostima, izbor funkcije gubitka zavisiće od toga da li želimo da dobijemo maksimalne rezultate nakon što je moguće manjeg broja epoha ili posle manjeg broja viđenih primera. Međutim, ukoliko zanemarimo broj primera i epoha i razmatramo samo vreme treniranja, stohastička verzija algoritma biće najpogodnija za upotrebu, s obzirom na to da vreme dekodiranja igra značajnu ulogu u ukupnom vremenu obučavanja.

Zavisnost u odnosu na parametar k prikazana je na slici 6.6. Sa manjim brojem iteracija, uvećanje parametra k utiče na poboljšanje rezultata u prepoznavanju za oba predstavljena problema. Ovo su očekivani rezultati jer se uključuju dodatne informacije kroz k struktura sa najvećim gubitkom. Kada je broj iteracija veliki, sve vrednosti parametra k daju slične rezultate sa blagim pogoršanjem za veoma velike vrednosti parametra k, kao što su 25, 50, 100. Razlog je najverovatnije taj da uključenje velikog broja struktura u proces obučavanja sa velikim brojem iteracija može uticati na veliko prilagođavanje parametara podacima za obučavanje. U tom slučaju, uvećanje parametra k nije korisno i stohastička verzija daje zadovoljavajuće rezultate. Međutim, dobijanje boljih rezultata sa povećanjem parametra k u slučaju manjeg broja iteracija pokazuje kada je verzija sa k najboljih struktura korisna. Ukoliko želimo da obučavamo metod u jednom prolasku kroz podatke, ili ukoliko imamo model koji treba da koriguje svoje parametre onlajn kada se prosledi novi podatak, korišćenje k struktura sa najvećim gubitkom treba da poboljša rezultate prepoznavanja.

Na kraju ćemo porediti Pegasos algoritam sa ostalim popularnim strukturnim algoritmima za klasifikovanje sekvenci kao što su: perceptron, MIRA, sekvencijalni dualni metod i pasivno-agresivni algoritam koji su bolje objašnjeni u odeljku A.2. Slika 6.7 pokazuje zavisnost F-mere od vremena obučavanja. Pegasos algoritmima je potrebno više iteracija da bi dostigli maksimalne rezultate prepoznavanja kada se koristi usrednjavanje parametara. Kada pričamo o najboljim rezultatima, možemo videti da svi algoritmi osim perceptrona dostižu veoma slične maksimalne vrednosti F-mere. Sličan zaključak možemo da donesemo i iz tabele 6.1. Ona prikazuje konačne rezultate za svaki metod zajedno sa odgovarajućim parametrima dobijenim iz unakrsne validacije.

A.6 Zaključak

Strukturni modeli imaju široku primenu u realnim problemima i trenutno predstavljaju najmoderniji pristup klasifikaciji podataka čiji su izlazi predstavljeni nekom strukturom. U tezi smo se skoncentrisali na problem njihovog obučavanja kao i na njihove primene na probleme klasifikovanja sekvenci. U nastavku dajemo najznačajniji doprinos teze.

Uveli smo primarni subgradijentni metod za optimizaciju strukturne usrednjene funkcije gubitka. Nasuprot standardnoj strukturnoj zglobnoj funkciji gubitka koja koristi samo jednu strukturu sa najvećim gubitkom, uvedeni metod može da koristi više struktura unutar jednog primera u procesu optimizacije.

U teorijskoj analizi, pokazali smo da je granica kumulativnog gubitka predviđanja za uvedeni metod maksimalno jednaka granici za subgradijentni metod sa zglobnom funkcijom gubitka, dok eksperimentalni rezultati sugerišu da pri manjem broju iteracija povećanje broja struktura uključenih u optimizaciju unutar jednog primera doprinosi boljim rezultatima.

Uveli smo proširenje strukturnih klasifikatora za rad sa k struktura sa najvećim gubitkom: k-najbolji (ograničeni) pasivno-agresivni algoritam i k-najbolji perceptron, koji su kompletno definisani u onlajn režimu, jednostavni za implementaciju, i osim ograničene verzije pasivno-agresivnog algoritma, veoma brzi i pogodni za primenu na probleme velikih razmera. Ograničena verzija je uvedena zbog teorijskih garancija, jer je njena granica za kumulativni gubitak predviđanja slična granici verzije sa zglobnom funkcijom gubitka.

Uveli smo proširenje sekvencijalog dualnog metoda i LaRank algoritma za slučaj sa k najboljih struktura unutar jednog primera, ali su te verzije postale vremenski zahtevne za obučavanje. U eksperimentalnim rezultatima, verzija LaRank algoritma sa k najboljih struktura dala je značajna poboljšanja u odnosu na osnovnu verziju algoritma, a metod je pogodan za obučavanje jednim prolaskom kroz podatke.

Predstavili smo organizaciju komiteta sa dva modela koji se treniraju u nizu, pri čemu drugi model koristi predviđenja prvog metoda zajedno sa poverenjem u njih. U ovoj organizaciji, poverenje u predviđanje ili kontekst predviđanja procenjuje se korišćenjem alternativnih struktura i omogućava poboljšanje rezultata u odnosu na pojedinačne modele.

Uveli smo sekvencijalni dualni metod za treniranje strukturnih nekonveksnih mašina sa vektorima podrške koje koriste dvostruko prelomljenu funkciju gubitka. Predstavljeni rezultati na realnim problemima klasifikovanja sekvenci ukazuju na sličnost prepoznavanja ovih metoda i metoda sa strukturnim zglobnim funkcijama. Sa druge strane, kada uvedeni metod radi sa podacima sa šumom, prepoznavanje je primetno poboljšano u odnosu na metode sa zglobnim funkcijama gubitka.

Appendix B

Biography

Dejan Mančev was born on May 1st, 1985, in Niš, Serbia. He completed Ratko Vukićević Elementary School in Niš as the best pupil of the generation, and Svetozar Marković Grammar School in a special class for talented mathematicians.

In the school year 2004/2005, he entered the Faculty of Sciences and Mathematics, University of Niš, at the Department of Mathematics and Informatics, and graduated in 2008 with a grade point average of 9.89/10 and grade 10/10 on his diploma thesis *Application of Kernel Methods for Relation Extraction*.

In 2008/2009, he enrolled in PhD studies at the Department of Computer Science, the Faculty of Sciences and Mathematics, University of Niš, and passed all exams with a grade point average of 10/10. During the studies, he spent a month at the Technische Universität in Dresden, Germany, as a part of the internship program *Natural Language Processing and Automata*. He published four papers in international journals with IF, one paper at an international conference, one paper in a domestic journal, and has one paper under review.

Since September 2009, Dejan has been working as a teaching assistant at the Faculty of Sciences and Mathematics in Nis in the Department of Computer Science on the following courses: Intelligent Systems, Computer Networks, Introduction to Operating Systems, Advanced Course in Computer Architecture, Programming Languages, Introduction to Programming, Fundamentals of Computing and Application of Computers in Biology.

Since 2010, he has participated in the project *Development of methods of computation and information processing: theory and applications* supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

List of papers:

- Mančev, D. and Todorović, B. (2015). k-best max-margin approaches for sequence labeling. *Computer Science and Information Systems (submitted)*.
- Mančev, D. (2015). A sequential dual method for the structured ramp loss minimization. *Facta Universitatis, Series: Mathematics and Informatics*, 30(1):13–28.
- Mančev, D. and Todorović, B. (2014). A primal sub-gradient method for structured classification with the averaged sum loss. *International Journal of Applied Mathematics and Computer Science (AMCS)*, 24(4):917–930.
- Stojković, M., Simić, V., Milosević, D., Mančev, D., and Penczak, T. (2013). Visualization of fish community distribution patterns using the self-organizing map: A case study of the Great Morava river system (Serbia). *Ecological Modelling*, 248:20–29.
- Milošević, D., Simić, V., Stojković, M., Čerba, D., Mančev, D., Petrović, A., and Paunović, M. (2013). Spatio-temporal pattern of the Chironomidae community: toward the use of non-biting midges in bioassessment programs. *Aquatic Ecology*, 47(1):37–55.
- Ilić, V., Mančev, D., Todorović, B., and Stanković, M. (2012). Gradient computation in linear-chain conditional random fields using the entropy message passing algorithm. *Pattern Recognition Letters*, 33(13):1776–1784.
- Mančev, D. and Todorović, B. (2012). Confidence based learning of a two-model committee for sequence labeling. In *11th Symposium on Neural Network Applications in Electrical Engineering (NEUREL)*, pages 167–170. IEEE.

Appendix C

Thesis documentation

Универзитет у Нишу

ИЗЈАВА О АУТОРСТВУ

Изјављујем да је докторска дисертација, под насловом

Тренирање стуктурних класификатора за различите функције губитака са применом на проблеме класификовања секвенци

која је одбрањена на Природно-математичком факултету Универзитета у Нишу:

- резултат сопственог истраживачког рада;
- да ову дисертацију, ни у целини, нити у деловима, нисам пријављивао/ла на другим факултетима, нити универзитетима;
- да нисам повредио/ла ауторска права, нити злоупотребио/ла интелектуалну својину других лица.

Дозвољавам да се објаве моји лични подаци, који су у вези са ауторством и добијањем академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада, и то у каталогу Библиотеке, Дигиталном репозиторијуму Универзитета у Нишу, као и у публикацијама Универзитета у Нишу.

У Нишу, _____

Аутор дисертације: Дејан Манчев

Потпис аутора дисертације:

_____

Универзитет у Нишу

ИЗЈАВА О ИСТОВЕТНОСТИ ШТАМПАНОГ И ЕЛЕКТРОНСКОГ ОБЛИКА ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Име и презиме аутора: Дејан Манчев

Наслов дисертације: *Тренирање стуктурних класификатора за различите функције губитака са применом на проблеме класификовања секвенци*

Ментор: др. Бранимир Тодоровић

Изјављујем да је штампани облик моје докторске дисертације истоветан електронском облику, који сам предао/ла за уношење у **Дигитални репозиторијум Универзитета у Нишу.**

У Нишу, _____

Потпис аутора дисертације:

_____

Универзитет у Нишу

ИЗЈАВА О КОРИШЋЕЊУ

Овлашћујем Универзитетску библиотеку „Никола Тесла“ да, у Дигитални репозиторијум Универзитета у Нишу, унесе моју докторску дисертацију, под насловом:

Тренирање стуктурних класификатора за различите функције губитака са применом на проблеме класификовања секвенци.

Дисертацију са свим прилозима предао/ла сам у електронском облику, погодном за трајно архивирање.

Моју докторску дисертацију, унету у Дигитални репозиторијум Универзитета у Нишу, могу користити сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons), за коју сам се одлучио/ла.

1. Ауторство **(CC BY)**

2. Ауторство – некомерцијално **(CC BY-NC)**

3. Ауторство – некомерцијално – без прераде **(CC BY-NC-ND)**

4. Ауторство – некомерцијално – делити под истим условима **(CC BY-NC-SA)**

5. Ауторство – без прераде **(CC BY-ND)**

6. Ауторство – делити под истим условима **(CC BY-SA)**

(Молимо да подвучете само једну од шест понуђених лиценци; опис лиценци дат је у Упутству).

У Нишу, _____

Аутор дисертације: Дејан Манчев

Потпис аутора дисертације:

_____