# Topics in Structured Prediction: Problems and Approaches

DRAFT

Ankan Saha

June 9, 2010

## Abstract

We consider the task of structured data prediction. Over the last few years, there has been an abundance of data having inherent structure with strong correlation and complex dependencies between different parts of each input. Numerous applications across different disciplines like Part Of Speech tagging, Optical Character Recognition, Pitch accent prediction among others underline the structure in the data which needs to be captured by standard learning algorithms to perform better than standard multivariate classification/regression. In this paper, we survey the existing structured prediction approaches for both training and inference. We show how the different existing training algorithms (maximum margin methods and maximum log-likelihood methods) are extensions of Empirical Risk Minimization schemes to the structured prediction domain. We also review the standard graphical model formalism -which is used to inherently define the structure in most complex data- and the corresponding assumptions which lead to efficient training and prediction algorithms. Most of the existing structured prediction methods heavily depend on the use of joint kernels which do not easily allow them to learn from unlabeled data. Finally we provide a new scheme based on vector valued functions, which provides a rich framework for training and inference and can be seamlessly extended to perform semi-supervised structured learning as well. We formulate a couple of algorithms under the proposed setting and characterize the corresponding classifying functions.

## Acknowledgements

I would like to thank everyone who has helped me grasp the relevant concepts and engaged in insightful discussions over the last few years. I am greatly indebted to Partha for being a wonderful mentor and introducing me to the world of theoretical machine learning and also giving me the freedom to explore different related problems. I am thankful to Vishy and Ambuj for listening to many of the formulations I came up with and providing their suggestions. Thanks to all my friends, particularly Raun and Shalini for listening to my rants when things did not work. Finally and most importantly, I owe more than I can express to Mom and Dad. None of this would have been possible without them. I hope this makes Dad proud, as he watches me from his peaceful lair. Thanks a lot!

# Contents

# Chapter 1

# Introduction

The variety and complexity of tasks addressed by machine learning has been rapidly increasing. Applications as varied as medical diagnosis, speech and handwriting recognition, natural language processing, financial analysis, object detection, robot locomotion apply machine learning algorithms as key steps for enhanced performance. Over the last decade, the abundance of data from all such applications has ushered in a lot of opportunity to improve the existing classification and prediction methods that had been existing previously.

However, till a few years back some tasks like sentence parsing or part of speech prediction were performed accurately by human beings, but were considered as difficult for computers, which often resulted in erroneous prediction. Machine learning algorithms offer an alternative by capturing the intricate details and complexity of the data and utilizing to develop efficient training and inference models which learn from exposure to existing data.

Supervised learning refers to the task of learning from examples. The most common supervised learning task is prediction or classification. Given a set of labeled data inputs, the task is to learn a classifying function, based on that data, which predicts well on new, unseen data. The notion of *wellness* is quantified in terms of minimizing an error function which measures the number of mistakes made by the classifying function (hypothesis). A typical example of classification is optical character recognition where the input is a scanned image and the label (output) is the character ( English letters or numerals). This is a classification task, where given sufficiently labeled images, we need to come up with a classifier that assigns labels accurately not only on the observed images but also on unseen images of characters.

There is a vast amount of literature that explores various aspects of supervised learning ( See for example [Vap98, SS02, HTF09] and references therein). The two most important aspects in developing supervised learning algorithms are how to choose the function class $\mathcal{H}$ from which the classifying hypothesis $h$ is selected and given $\mathcal{H}$, how to choose the classifier $h$. Most machine learning algorithms including some of those described in this thesis restrict themselves to generalized linear models. In this thesis, we look at classes of functions called Reproducing Kernel Hilbert Spaces [Aro50].

Most of supervised learning generally focuses on binary classification (only two possible labels) or multiclass labels. However with increasing complexity of data, we now regularly face inputs in which the labels are multiple, complex and correlated.

## 1.1 Structured Prediction

This range of problems deal with complex inputs and outputs which are governed by an inherent structure with underlying complex dependencies and strong correlations between different parts of a single input and are of immense computational interest in the present day. Some examples and domains of applications of such problems include:

- **Natural language processing**: Part of speech (POS) tagging of the words in sentences, Named Entity recognition (NER) in a passage of text, parsing sentences.
- **Computer Vision**: Recognizing continuous optical stream of characters and words from scanned images.
- **Speech Recognition**: Recognition of words from continuous acoustic signals.
- **Computational Biology**: Completing sequences of genomes, protein structure prediction.

All of the above problems involve complicated input (generally in the form of a sequence). It is essential to draw information from this structure to perform better classification/prediction as opposed to treat the components of the structured inputs as independent entities. The existing literature [Tas04, SM06, TJHA05] generally focuses on models which try to create a score function combining both the inputs and the labels in the training data and predict the most likely label for a new given input as the label maximizing the score. However naive methods of inference run into intractability very fast as the space of labels is frequently exponential in size. As a result assumptions are made on the structure of the input and special dynamic programming methods are used to come up with efficient training and inference algorithms.

### 1.1.1 Graphical Models and Markov Networks

In most structured data algorithms, the data is assumed to conform to an underlying graphical model which represents the correlations and the dependencies between different parts of the structured output. Markov networks (also known as Markov Random Fields) are examples of undirected graphical models which are extensively used to model the interactions between the variables of the output. The nodes of the graph depict the labels of the components of the structured data (for example: the part of speech tag of words of a sentence) and the edges represent the dependencies between these nodes as well as the conditional independence. They are also used to depict the conditional distribution of the structured output given the input- this kind of probabilistic model is called the Conditional Random Field [LMP01] on which training and inference (based on maximizing log-likelihood of the data) can be performed efficiently if the graphical model satisfies certain properties like having low tree-width [BHS$^+$07].

Margin based models [Vap98] can also be set up based on underlying graphical model assumptions of the structured data [TGK04, TJHA05]. The corresponding training and inference algorithms run into intractability and efficient algorithms can be developed under appropriate assumptions on the graphical model. In particular, the training algorithms make use of convex duality and make use of a joint kernel formulation. They also assume that existing belief propagation techniques [Lau96, KFL01] can be used to compute the *marginals*( defined in chapter 6) of the dual variables over the graphical model efficiently.

Both of these techniques are specific examples of Regularized Risk Minimization schemes [Vap98] which is one of the most prevalent general schemes of developing supervised learning algorithms from labeled data.

## 1.2 Contribution

In this thesis, we introduce a new set of models based on **vector valued functions** [MP05] for performing structured prediction. We associate the space of structured labels $\mathcal{Y}$, with a Reproducing Kernel Hilbert Space (see section 2.2) $\mathcal{H}_\mathcal{Y}$ and then perform regularized risk minimization with functions that map structured inputs to functions(vectors) in $\mathcal{H}_\mathcal{Y}$. This results in defining operator valued kernels which have analogous reproducing properties [CVT06]. We set up vector valued regression and vector valued SVMs as analogous regularized risk minimization problems and characterize the classifying function $h^*$ in terms of the operator kernel function evaluated on the structured inputs.

Given a new structured input $\hat{\mathbf{x}}$, the inference algorithm is performed by selecting the label in $\mathcal{Y}$ whose mapping into the associated RKHS is closest to the image of $h^*(\hat{\mathbf{x}})$ under suitable notions of distance that vary according to the algorithm used. This inference algorithm is analogous to the existing ones in structured prediction literature and depend only on kernel evaluations and efficient belief propagation techniques. We leave the details of the inference procedure for future work.

It should be noted that our formulation carefully avoids the usage of joint kernels. As a result we are able to develop semi-supervised learning algorithms [BNS06] from labeled as well as unlabeled structured data under appropriate assumptions. We plan to do this by calculating the eigenfunctions of the Laplacian operator [BN04] on vector valued functions which form an orthogonal basis for the corresponding space of functions. These eigenfunctions efficiently represent the geometry of the unlabeled structured data. A suitable collection of the eigenfunctions can then be used to perform regularized risk minimization with the available labeled data, thus resulting in a semi-supervised classifier. Note that the existing structured learning algorithms define kernels jointly on the space of both inputs and the labels- as a result it is not possible to extend them to work with unlabeled data without padding them arbitrarily with labels, thus resulting in an enormous amount of noisy data [AMB06]. This is a big drawback since structured unlabeled data is now available in abundance while tagging them with labels manually is a time consuming, costly and possibly erroneous endeavor. Both these factors strongly motivate the need for efficient semi-supervised learning schemes for structured data which can produce reasonably accurate classifiers.

## 1.3 Outline

We review some of the basic definitions and the statistical framework of regularized risk minimization in chapter 2. We also look at some of the standard ERM algorithm schemes with different surrogate loss functions. The concepts of kernels and Reproducing Kernel Hilbert Spaces(RKHS) are also discussed in this chapter as well as the representer theorem (2.5) which helps characterize the minimizing hypothesis for regularized risk minimization problems.

Chapter 3 describes the structured data prediction setting in detail with associated examples and introduces the different algorithms in practice along with the kind of features they work with. We review concepts of graphical models and conditional independence in chapter 4 which describe notions of exponential families [WJ08] and the quantitative and qualitative aspects of an underlying graphical model structure for structured prediction.

A particular class of probabilistic structured learning algorithms called Conditional Random Fields [LMP01] are described in chapter 5. We describe the formulation of the training as well as the efficient dynamic programming schemes that come up in the training as well as the inference procedures. Chapter 6 look at another set of structured supervised algorithms based on margin maximization called Max-Margin Markov Networks ($\mathsf{M}^3\mathsf{N}$)[TGK04]. We develop the problem and describe both the primal and the convex dual formulations that are developed in the literature. We do not delve into the depth of the belief propagation schemes which are assumed to hold for efficient training of the model. The inference algorithm for $\mathsf{M}^3\mathsf{N}$s are similar to that of CRFs- hence we don't discuss them separately.

We give the basics of vector valued functions and operator valued kernels in chapter 7. The structured learning problem is set up in this framework and the corresponding formulation and approaches are discussed. Chapter 8 looks at ERM procedures corresponding to regression and vector valued SVMs which are developed analogous to ERM procedures in the existing literature. We look at the problem formulation closely and characterize the nature of the classifying function in terms of the operator valued kernels. We also have schemes for efficient algorithms when the vector valued functions admit a finite dimensional image space. However developing efficient implementable algorithms for arbitrary vector valued functions is a work in progress.

In chapter 9, we detail the semi-supervised learning scheme as one of the main projects currently in progress and also detail some of the exact optimization issues that need to be handled to develop implementable algorithms from the formulation of vector valued functions. Finally chapter 10 provides the conclusion.

# Chapter 2

# Supervised Learning

Supervised learning is the problem of learning from a set of given examples. Given the *training data*, $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)\}$ a set of independent and identically drawn examples from a fixed but unknown joint probability distribution $\mathcal{P}$ on $\mathcal{X} \times \mathbb{Y}$, the goal of the supervised learning algorithm $\mathcal{A}$ is to find a function (*hypothesis*) $h : \mathcal{X} \to \mathbb{Y}$ that maps inputs $\mathbf{x} \in \mathcal{X}$ to labels $y \in \mathbb{Y}$. The space of inputs $\mathcal{X}$ is arbitrary and problem dependent. In particular $\mathcal{X} \subseteq \mathbb{R}^d$, when the inputs are real valued vectors in $d$ dimensions, $\mathcal{X} \subseteq \Sigma^*$, when the inputs are strings of arbitrary length made up of letters from the alphabet $\Sigma$ or $\mathcal{X} \subseteq \mathcal{M}$, when the input points are drawn from a manifold $\mathcal{M}$. We work mostly with a discrete set of labels $\mathbb{Y} = \{y_1, \ldots y_k\}$. This type of supervised learning problems with labeled inputs when the number of labels is finite and discrete is generally referred to as **classification** whereas when the space of labels is continuous (for example $\mathbb{R}^d$) the associated learning problem is called **regression**. For example, in a weather prediction problem, the input $\mathbf{x}$ maybe a vector of attributes or $features$ signifying various environmental factors like relative humidity, temperature whereas $y$ might be $+1$ or $-1$ according to whether it is going to rain or not.

A good hypothesis $h$ should not only accurately classify the $\mathbf{x}_i$'s in the training data but also classify new samples $\mathbf{x}$ from the distribution $(\mathbf{x}, y) \sim \mathcal{P}$ reasonably well. There are different classes of learning algorithms depending upon the **hypothesis class** $\mathcal{H}$ from which the hypothesis $h$ is drawn. These include decision trees, neural networks, log linear models, kernel methods to name a few. For details on these models and further discussion, we refer the readers to [SS02, HTF09, Bis06]. In the subsequent chapters we look at examples of various functions based on log linear models and using different kind of kernels.

## 2.1 Empirical Risk Minimization

Given a hypothesis class $\mathcal{H}$, one of the most commonly used schemes of selecting the best possible classifier $h$ is based on *empirical risk minimization*. In order to gauge the accuracy of a classifier $h$ we define a loss function $L : \mathcal{X} \times \mathbb{Y} \times \mathbb{Y} \rightarrow \mathbb{R}^+$ which is a measure of how well $h$ classifies $\mathbf{x}$ when the true label is $y$. Naturally we should have $L(\mathbf{x}, y, h(\mathbf{x})) = 0$ if $h(\mathbf{x}) = y$.

The most common loss considered in classification is the 0/1-loss:

$$L^{0/1}(\mathbf{x}, y, h(\mathbf{x})) = \mathbb{I}(y \neq h(\mathbf{x}))$$

where $\mathbb{I}(\cdot)$ denotes the indicator function *i.e.*$\mathbb{I}(g) = 1$ if $g$ is true and 0 otherwise.

The **risk** $\mathcal{R}_{L,\mathcal{P}}(h)$ of a classifier $h$ is the expected loss *w.r.t.* the distribution $\mathcal{P}$ from which the data is drawn. Thus

$$\mathcal{R}_{L,\mathcal{P}}(h) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{P}} \left[ L(\mathbf{x}, y, h(\mathbf{x})) \right] \tag{2.1}$$

A plausible way of doing classification is to choose the hypothesis $h^*$ with the minimum risk. Since we do not know the distribution $\mathcal{P}$, it is not possible to evaluate the actual risk. We estimate the risk using its empirical estimate $\mathcal{R}_{L,S}(h)$ given by the empirical average over the dataset $S$ *i.e.*

$$\mathcal{R}_{L,\mathcal{P}}(h) = \frac{1}{n}\sum_{i=1}^{n} L(\mathbf{x}_i, y_i, h(\mathbf{x}_i)) \tag{2.2}$$

For the 0/1- loss this actually refers to the proportion of the training examples which are misclassified.

It is not always a good idea to choose the hypothesis with the minimum empirical risk. In particular, if the hypothesis class is very large, it might **overfit** the training data, *i.e.*it might have zero empirical risk on the training data set $S$, but high actual risk, which occurs due to poor generalization performance of the classifier on unseen data. The key to selecting a good classifier is to trade off the empirical risk of the classifier with the complexity of the hypothesis class $\mathcal{H}$. This concept is called **regularization** [Vap98] and also helps in making the problem of empirical risk minimization $well - posed$ besides avoiding the problem of overfitting. We avoid the discussion of these topics due to lack of space and refer interested readers to [Vap98, DGL96] for details. Thus we solve a new problem called the *regularized empirical risk minimization* (ERM) problem:

$$h^* = \operatorname*{argmin}_{h \in \mathcal{H}} \mathcal{R}_{L,\mathcal{P}}(h) + \lambda\Omega(h) \tag{2.3}$$

where $\Omega(h)$ is generally a non-negative function called the **regularizer** which penalizes the inherent complexity of the classifier $h$ and $\lambda$ is a trade-off factor called the *regularization parameter.*

If the hypothesis class $\mathcal{H}$ is very large and complex, the search for the optimal $h^*$ in (2.3) can become a very challenging task. In particular, for structured learning, we will see that finding the optimal hypothesis frequently becomes intractable due to the exponential size of the label space. For these intractable classes, it becomes very important to choose an efficient searching procedure by the algorithm, for it to remain efficiently implementable. In the sequel, we consider the models for which the optimal hypothesis can be evaluated efficiently in polynomial time by existing methods in convex optimization.

## 2.2 Kernels and Reproducing Kernel Hilbert Spaces

For most machine learning problems, the hypothesis class $\mathcal{H}$ is chosen to be a Hilbert Space, more specifically a Reproducing Kernel Hilbert Space (RKHS) [Aro50]. Given a set $\mathbb{X}$ and a Hilbert space $\mathcal{H}$ of complex valued functions on $\mathbb{X}$, $\mathcal{H}$ is called an RKHS if $\forall \mathbf{x} \in \mathbb{X}$, the evaluational functional

$$ev_{\mathbf{x}} : f \mapsto f(\mathbf{x}) \qquad \forall f \in \mathcal{H}$$

from $\mathcal{H}$ to the complex numbers is continuous. This is equivalent to the evaluation functional $ev_{\mathbf{x}}$ being bounded $\forall \mathbf{x} \in \mathbb{X}$. Using the Riesz Representation Lemma [AG93], it can be shown that $\forall \mathbf{x} \in \mathbb{Z}$, there exists a function $K_{\mathbf{x}} : \mathbb{X} \to \mathbb{C}$ to the space of complex numbers such that

$$\forall f \in \mathcal{H} \qquad \langle K_{\mathbf{x}}, f \rangle_{\mathcal{H}} = f(\mathbf{x}) \tag{2.4}$$

This is called the *reproducing* property and $K_{\mathbf{x}}$ is called the point evaluation functional at $\mathbf{x}$. We define a function $K : \mathbb{X} \times \mathbb{X} \to \mathbb{C}$ given by

$$K(\mathbf{x}, \mathbf{x}') = K_{\mathbf{x}}(\mathbf{x}') = K_{\mathbf{x}'}(\mathbf{x})$$

which is called the reproducing kernel corresponding to $\mathcal{H}$ and the latter is called a RKHS associated with $K$.

Given an RKHS $\mathcal{H}$, the corresponding reproducing kernel is determined uniquely due to the Riesz Representation lemma. We have $\forall \mathbf{x} \in \mathbb{X}, \quad K(\mathbf{x}, \mathbf{x}) \geq 0$. Also the *gram matrix* of $K$, given by $K_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$ for all pairs $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{X}$ is positive semidefinite. Thus the kernel corresponding to the RKHS $\mathcal{H}$ is positive semidefinite.

Analogously, using a result known as the Moore-Aronszajn theorem [Aro50], it can be shown how to generate a unique RKHS corresponding to a positive semidefinite kernel $K$. Thus a kernel-RKHS pair is uniquely defined.

In an ERM problem (2.3), if the hypothesis space is an RKHS and the regularizer is chosen to be the norm of $h$ in $\mathcal{H}$ space, it can be shown using standard orthogonality arguments, that the solution $h^*$ lies in the span of the kernel function evaluated at the data points *i.e.*

$$h^*(\hat{\mathbf{x}}) = \sum_{i=1}^{n} \alpha_i K_{\mathbf{x}_i}(\hat{\mathbf{x}}) \tag{2.5}$$

This result is called the *Representer* theorem [KW71] and is one of the fundamental results in statistical machine learning.

## 2.3 Generalized Linear Models

In order to limit the complexity of the hypothesis class over which (2.3) is minimized, we generally work with generalized linear family of functions parametrized by some weight vector $\mathbf{w}$. Thus $h$ has the general form of $h_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$ where $\phi(\cdot)$ is some kind of *feature* map applied to the input.

As mentioned before the most obvious loss for classification problems is the 0/1-loss. But due to its non-convexity, it is very difficult to minimize this loss using standard methods from convex optimization. The standard alternative is to optimize a convex upper bound to the loss. Replacing different loss functions in (2.3) lead to different algorithms. In particular, the algorithms described in the subsequent sections - ridge regression, logistic regression and Support Vector Machines are different examples of empirical risk minimization algorithms (2.3) which differ in the choice of the loss term. The regularizers that are generally used for linear models are the $p$-norms of $\mathbf{w}$ where

$p \in [1, 2]$ which penalize the complexity of $\mathbf{w}$. The 2-norm is smooth and well behaved whereas the 1-norm induces sparsity in the solution which invokes its use despite the non-smoothness of the 1-norm. Other examples of regularization used include the entropy function on $\mathbf{w}$.

## 2.4 Algorithms for Empirical Risk Minimization

### 2.4.1 Ridge Regression

Ridge Regression, also known as the method of *least squares*, is an ERM algorithm in which the loss function is the square of the difference between $h(\mathbf{x})$ and $y$. Mathematically

$$h^* = \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y_i - h(\mathbf{x}_i))^2 + \lambda \|h\|_{\mathcal{H}}^2 \tag{2.6}$$

If $\mathcal{H}$ is an RKHS corresponding to a kernel $K$, using the Representer theorem (2.5) [KW71, SS02] obtained from standard orthogonality arguments, we know that the minimizer $h^*$ lies in the span of the kernel function evaluated at the data points of the sample $S$ *i.e.*

$$h^*(\cdot) = \sum_{i=1}^{n} \alpha_i K_{\mathbf{x}_i}(\cdot)$$

### 2.4.2 Logistic Regression

In this section, we consider the simplest example of logistic regression for binary classification where $y \in \{+1, -1\}$. The ratio between the conditional probabilities is parametrized by the estimating function $h$. Mathematically,

$$h(\mathbf{x}) = \log \frac{P(y = 1|\mathbf{x})}{P(y = -1|\mathbf{x})}$$

Simple algebra yields

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-h(\mathbf{x}))} \qquad \text{and} \qquad P(y = -1|\mathbf{x}) = \frac{1}{1 + \exp(h(\mathbf{x}))}$$

which can be combined to give

$$P(y|\mathbf{x}) = \frac{1}{1 + \exp(-yh(\mathbf{x}))} \tag{2.7}$$

Again, for simplicity, we assume the hypothesis class to be limited to generalized linear models parametrized by some weight vector $\mathbf{w}$. Thus

$$P(y|\mathbf{x}) = \frac{1}{1 + \exp(-y \langle \mathbf{w}, \phi(\mathbf{x}) \rangle)}$$

This is the expression for conditional likelihood of the data. Maximizing the **log-likelihood** is a standard statistical approach of fitting the model (parametrized by $\mathbf{w}$) to the data. This is equivalent to minimizing the negative of the log-likelihood of the data, which is actually the **logistic loss**. Minimization of the logistic loss with regularization leads to the algorithm called logistic regression.

$$\begin{aligned} \mathbf{w}^* &= \operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2} \|w\|^2 - \log \prod_{i=1}^{n} P(y_i|\mathbf{x}_i) \\ &= \operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y \langle \mathbf{w}, \phi(\mathbf{x}) \rangle)) \end{aligned} \tag{2.8}$$

The logistic loss is a convex upper bound to the scaled 0/1-loss. For multiclass losses the conditional log-likelihood is given in terms of a parametrized exponential family, details of which are described in chapter 5.

### 2.4.3 Support Vector Machines

Support Vector Machines [Vap98] have gained a lot of popularity over the last decade as an efficient method of binary and multi-class classification. It is based on the idea of *margin maximization*. In binary classification, the *margin* is defined as the distance of closest point from the separating hyperplane (assuming linear hypotheses) [SS02].

In the sequel we assume that the classifying function $h$ is linear *i.e.* $h(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}\rangle$ for some weight vector $\mathbf{w}$ and feature function $\phi(\cdot)$. In case of binary classification, given data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $y_i \in \{+1, -1\}$, the goal is to come up with a weight vector $\mathbf{w}^*$ such that $\langle \mathbf{w}^*, \phi(\mathbf{x}\rangle > 0$ when $y = 1$ and $\langle \mathbf{w}^*, \phi(\mathbf{x}\rangle < 0$ when $y = -1$ so that the classification made by predicting the sign of the dot product actually matches the sign of the label $y$. In other words,

$$\forall i \in [n] \qquad y_i \langle \mathbf{w}^*, \phi(\mathbf{x}_i) \rangle > 0$$

At the same time, the algorithm also aims to maximize the margin. For that purpose we want to ensure that $\langle \mathbf{w}^*, \phi(\mathbf{x}) \rangle$ is very positive by a large margin when $y = 1$ and very strongly negative otherwise. This leads us to modify the above set of constraints as

$$\forall i \in [n] \qquad y_i \langle \mathbf{w}^*, \phi(\mathbf{x}_i) \rangle > 1$$

Since the margin of a point $\mathbf{x}$ is defined by $\frac{\langle \mathbf{w}, \phi(\mathbf{x}) \rangle}{\|\mathbf{w}\|^2}$, maximizing the margin is equivalent to minimizing the norm of the weight vector. This leads us one of the formulation of the algorithm called Hard Margin- Support Vector Machines (SVMs)

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 \qquad \text{such that}$$
$$\forall i \in [n] \qquad y_i \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle > 1$$

An equivalent formulation can be described as an ERM problem with the loss function chosen to be the **hinge loss**. Given $(\mathbf{x}, y)$ and a function $f$, the hinge loss is given by

$$H(\mathbf{x}, y, f) = (1 - yf(\mathbf{x}))_+ = \max(1 - yf(\mathbf{x}), 0)$$

The hinge loss is another convex non-smooth upper bound to the 0/1-loss function. Given the above set of data points and restricting $f$ to the class of linear classifiers, the SVM optimization formulation can be set up as the following ERM problem

$$\min_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{n}(1 - y_i \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle)_+ + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

We can write the above formulation using slack variables $\xi_i$ and adding a set of constraints [SS02, SC04]. Borrowing concepts of convex duality, we can set up a function called the Lagrangian by adding dual variables $\alpha_i$ corresponding to each constraint leading to a vector $\boldsymbol{\alpha}_{n\times 1}$ of dual variables. We omit the details for brevity. Taking gradients of the Lagrangian yields the dual optimization problem

$$\max_{\boldsymbol{\alpha}} D(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2\lambda}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle \qquad \text{such that}$$
$$\forall i, \qquad 0 \le \alpha_i \le \frac{1}{n}$$

and the optimum weight vector $\mathbf{w}^* = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)$. The vectors $\mathbf{x}_i$ for which $\alpha_i > 0$ are the ones which lie on the margin and are sensitive for classification. These are called the **support vectors** and are essential for prediction.

If we define a kernel function $K$ as $K_{\mathbf{x}}(\cdot) = \langle y\phi(\mathbf{x}), \cdot\rangle$, the gram matrix of the kernel $K$ is a $n \times n$ matrix whose entries are given by $K_{i,j} = y_i y_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$. The optimum weight vector $\mathbf{w}^*$ can then be represented as $\mathbf{w}^* = \sum_i \alpha_i K_{\mathbf{x}_i}$ which is the representer theorem (2.5) in this scenario. Note that the dual optimization problem only depends on $\mathbf{x}$ via the inner products $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$. Hence one can employ the kernel trick: map $\mathbf{x}_i$ into a feature space via $\phi(\mathbf{x}_i)$ and compute the dot product in the feature space by using a kernel function $K(\mathbf{x}_i, \mathbf{x}_j) := \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$ [SS02]. The kernel trick makes SVM rather powerful because simple linear decision boundaries in feature space map into non-linear decision boundaries in the original space where the datapoints live.

# Chapter 3

# Learning with Structured Outputs

We start with a simple example: Part-of-speech(POS) tagging. Given a sentence in a particular language, the task is to tag the words in the sentence with the correct part of speech. There has been a significant amount of work in the natural language processing community addressing this problem ( See [Rat96, BPP96] and references therein). One of the natural ways of classification in this setting, is to consider the training data consisting of sentences as a bag of words(BOW) and do supervised classification on the words, without paying any attention to the structure of the sentence. While this task achieves reasonable accuracy in classification, it looks at the words in isolation and does not take the rich structure of the sentence into consideration. However, taking laws of the grammar into consideration (for *e.g.* in English, an adjective is generally followed by a noun or a verb), we can achieve significantly higher accuracy in POS classification.

In this chapter and most of the sequel, we assume that the output is not a single discrete value $y$, but has a richer structure, like a graph. For most practical purposes, we will be working with sequences, $\mathbf{y} = (y_1, y_2, \ldots, y_L)$, for example, the part of speech taggings of the words in a sentence. The output space $\mathcal{Y}$ is now given by $\mathcal{Y} \subseteq \bigcup_{k\geq 1} \mathbb{Y}^k$, which is a subset of the union of the product space of the individual variables for different lengths $k$. For concreteness, we assume that the length of the sequences $L$ is fixed. There generally exists a lot of correlation between adjacent variables in the sequence, *e.g.* consecutive words in a sentence for POS tagging problems. These spaces are therefore called **structured** and the corresponding prediction problems with discrete output spaces are referred to as **structured classification** or **structured prediction**.

Over the last decade, structured prediction has seen a vast expansion in application scenario. Problems as diverse as POS tagging, Named Entity Recognition (NER) [SM06], Parsing [Rat99, TKC+04] in computational linguistics, optical character recognition (OCR) [TGK04] in computer vision, pitch accent prediction in speech recognition [AMB06], identifying protein names in biology abstracts [Set05] all make use of structured prediction methods to classify data which has a sequence structure. As opposed to previous methods used for supervised classification, structured models work *jointly* over the input and output space to predict the outputs while respecting the inherent correlations and structure in the data.

Prediction problems take a two step approach. Given a set of input label pairs $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$ drawn from an unknown joint probability distribution $\mathcal{P}$ on $\mathcal{X} \times \mathcal{Y}$, we develop a supervised learning algorithm to generate a scoring function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ which measures how good a label $\mathbf{y}$ is for a given input $\mathbf{x}$. Given a new input $\hat{\mathbf{x}}$, the algorithm predicts the label which maximizes the scoring function *i.e.*

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\hat{\mathbf{x}}, \mathbf{y}) \ . \tag{3.1}$$

For computational feasibility, the scoring functions generally considered are linearized models operating on joint features. Thus

$$F(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}^*, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle \tag{3.2}$$

where the optimal weight vector $\mathbf{w}^*$ is generally evaluated by the supervised algorithm by solving an empirical risk minimization problem. It should be noted that in structured learning, the vector of features $\boldsymbol{\phi}$ is generally defined over both inputs as well as outputs. A typical example of a joint feature for a POS tagging task can be something like the following [Rat96]:

$$\phi(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if suffix}(x_i) = \texttt{"ing"} \text{ and } y_i = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

where the suffix $i$ refers to the $i^{\text{th}}$ position in the sequence.

A major concern for the implementation of most structured prediction algorithms is the issue of tractability. If each $y_i$ can take $k$ possible values *i.e.* $|\mathcal{Y}_i| = k$, the total number of possible labels for a sequence of length $L$ is $k^L$. Even though all these labels may not be feasible, the space of labels $\mathcal{Y}$ is likely to be exponential in the length of the input. Thus for many possible $\boldsymbol{\phi}$, finding the optimal $\mathbf{y}$ is intractable. We restrict our attention to models where the relevant optimization problems can be solved in polynomial time. To maintain tractability, it is customary to assume that $\mathcal{Y}$ is endowed with a graphical model structure [WJ08] and there exist conditional independence properties of the $\mathcal{Y}$ space. For an in-depth treatment of this topic, we refer the reader to [BHS$^+$07].

There are two main categories in which the existing literature of structured prediction models can be subdivided. These vary principally in the loss function used to solve the ERM. Probabilistic models like **Conditional random Fields (CRFs)** [LMP01] try to maximize the conditional log-likelihood of the data. This is equivalent to solving an ERM problem with the analog of logistic loss for structured prediction. Details of this approach are mentioned in chapter 5.

On the other hand max-margin methods try to maximize the margin while solving an ERM problem which uses the analog of hinge loss for structured prediction. This leads to SVM like algorithms *e.g.* **Max-Margin Markov Networks** (M$^3$N)[TGK04] and **SVM-Struct** [TJHA05].

Details of this approach are mentioned in chapter 6.

Each of these approaches have particular issues involving inference over an exponential space of labels and optimizing an ERM objective function over a feasibly exponential space which are handled by the corresponding algorithms appropriately.

# Chapter 4

# Graphical Models: Usage in Structured prediction

A lot of statistical dependencies exist between structured data entries. Graphical models are a natural formalism for exploiting the dependency structure between entities. They also make efficient use of features that can be extracted from the structured data to come up with rich models that perform much more accurately than standard multivariate regression.

Traditionally graphical models have been used to represent the joint probability distribution which assigns a normalized joint distribution $p(\mathbf{x}, \mathbf{y})$ on the joint space $\mathcal{X} \times \mathcal{Y}$. Such models which represent the joint distribution are called *generative* models. They are generally structured to allow for efficient maximum likelihood learning. However typically modelling the joint distribution is difficult since it requires modeling the marginal distribution $p(\mathbf{x})$ which may have complex dependencies. Thus training such a model would require a large set of simplifying assumptions (more precisely, *independence* assumptions) which reduces the performance of the model. In structured learning, *Hidden Markov Models* (HMMs) [Rab89] are examples of generative models which are used to model structured objects like sequences.

Another approach is to directly model the conditional distribution $p(\mathbf{y}|\mathbf{x})$. This model makes much fewer assumptions than the generative models and will learn the best possible conditional distribution and arguably the best possible prediction.This is the approach taken by *Conditional Random Fields* (CRFs) [LMP01] for structured learning. Before delving into the details of these models it is important to give a general idea of graphical models and conditional independence.

Graphical models [Lau96] combine probability and graph theory to address new problems that come up in designing and fitting probabilistic models to capture dependencies in complex structures. While defining such probabilistic models, *conditional independence* plays a key role. For notational convenience, we use capital letters $(X, Y, \mathbf{X}, \mathbf{Y})$ to denote random variables and lower case letters to denote a particular instantiation of the random variables.

A graphical model is a family of probability distributions that factorize according to a given underlying graphical structure. The quantitative aspects of the model are expressed in terms of potential functions on the nodes and the edges of the graph. Before explaining conditional independence formally, we motivate it using an example of POS prediction on a sequence. Consider a sentence where each word is a node of the graph which are connected by edges resulting in a chain. Each of the nodes is associated with an output variable $Y_i$ and the edges represent the correlations. The standard assumption made in most natural language processing applications is that $Y_j$ is conditionally independent of the other variables, given $Y_{j-1}$, $Y_{j+1}$ for all $j$. This is intuitive, since a word is supposed to have high correlation with its adjacent words. Mathematically this can be stated as

$$P(Y_i|Y_1,\ldots,Y_{i-1},Y_{i+1},\ldots,Y_L,\mathbf{x}) = P(Y_i|Y_{i-1},Y_{i+1},\mathbf{x}) \tag{4.1}$$

These conditional independence assumptions strongly aid in inference, since the independent problems can be looked at separately, thereby resulting in tractable inference algorithms. To maintain brevity, we don't go into the details but direct the readers to [BHS+07] and [Jor08] for an in-depth discussion of graphical models.

Depending on whether the underlying graph is directed or not the underlying graphical models are called Bayesian networks or Markov Networks [BHS+07]. As mentioned before, when these models are extended to structured prediction, we get Hidden Markov Models and Conditional Random Fields. In this thesis, we will focus mainly on the latter which are based on undirected graphical models.

## 4.1 Markov Networks

Markov Networks are a expressive family of models which are used for both discrete and continuous prediction [Pea88]. We treat the inputs and outputs as random variables and consider probability distributions over sets of random variables $\mathbf{X} \in \mathcal{X}$ and $\mathbf{Y} \in \mathcal{Y}$ and define a conditional probability distribution $p(\mathbf{Y}|\mathbf{X})$. The graphical structure of the framework provides several advantages: it exploits the correlations between outputs $\mathbf{Y}$ and also the complex conditional dependencies.

A Markov network is defined by the underlying undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the vertices refer to the random variables associated with the labels $\mathcal{V} = \{Y_1, Y_2, \dots Y_L\}$. We denote by $\mathcal{C}(\mathcal{G})$, the set of all the cliques in $\mathcal{G}$. In particular, when the graph is a sequence, as considered before, the cliques become the set of nodes and edges $\{Y_1, \dots Y_L, \{Y_1, Y_2\}, \{Y_2, Y_3\}, \dots, \{Y_{L-1}, Y_L\}\}$. We denote the set of variables associated with a clique $c$ as $\mathbf{Y}_c$, a particular assignment of variables associated with a clique as $\mathbf{y}_c$ (lower case) and the space of all possible assignments associated with the clique $c$ as $\mathcal{Y}_c$.

In the following, we use the concept of conditional independence: Given sets of random variables

$X$, $Y$ and $Z$, $X \perp Y|Z$ (read as: $X$ is independent of $Y$ given $Z$) iff $P(XY|Z) = P(X|Z)P(Y|Z)$.

We now formally define a Markov Network [Tas04].

**Definition 1** *Given a set of random variables $\mathcal{V} = \{Y_1, Y_2, \ldots Y_L\}$ the associated* **Markov Network** *is defined by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a set of functions $\boldsymbol{\psi} = \{\psi_c\}_c$. The vertices of the graph are the variables in $\mathcal{V}$. Each clique $c \in \mathcal{C}(\mathcal{G})$ is associated with a* **potential** *function $\psi_c(\mathbf{x}, \mathbf{y}_c)$ where $\psi_c : \mathcal{X} \times \mathcal{Y}_c \to \mathbb{R}$ which associates a value with every assignment $\mathbf{y}_c$ to a clique $c$ with input $\mathbf{x}$. The random variables $\mathbf{Y}$ indexed by the vertices of the graph satisfy the Markovian property: $Y_i \perp Y_j|\mathcal{V}\backslash \{Y_i, Y_j\}$ if $\{Y_i, Y_j\} \notin \mathcal{E}$.*

The corresponding graph $\mathcal{G}$ is called the independence graph or the factor graph which governs the factorization of the Markov Network [KFL01]. The Markov Network $(\mathcal{G}, \boldsymbol{\psi})$ defines a conditional distribution which factorizes according to the *Hammersley-Clifford* factorization result [HC71]

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}(\mathcal{G})} \psi_c(\mathbf{x}, \mathbf{y}_c) \tag{4.2}$$

The constant $Z$ is a normalization factor parametrized only by $\mathbf{x}$ and given by

$$Z(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} \prod_{c \in \mathcal{C}(\mathcal{G})} \psi_c(\mathbf{x}, \mathbf{y}_c) \tag{4.3}$$

which ensures that the RHS in (4.2) sums up to 1. $Z$ is called the **partition function**. Computing $Z$ is intractable in general due to the exponential number of terms in the sum, but there are a lot of approaches to approximate it, including belief propagation [Lau96, KFL01].

In the case of sequences, since the cliques are only individual nodes and edges, we have the concept of node and edge potentials. For a particular position $i$ in the sequence, the node potential $\psi(\mathbf{x}, y_i)$ measure the correlation between the input $\mathbf{x}$ and the label of that node, while for an edge in the sequence $(y_i, y_{i+1})$, the corresponding edge potential $\psi(\mathbf{x}, \{y_i, y_{i+1}\})$ quantify the correlation between the adjacent pair of output variables as well as the input $\mathbf{x}$. Potential functions typically provide a high unnormalized value called the **score** to each assignment of the clique. In the POS tagging example, conditioned on the input sentence, the Markov network should give high score to the correct part of speech labels of the individual words based on the node potentials. On the other hand, the edge potentials should give high score to pairs of part of speech tags which have a higher tendency to occur adjacently. Combined together, these potential functions should provide the highest score to the true POS tagging of the sentence.

A Markov network can be expressed as a generalized log-linear model. In particular, we assume that the potential functions can be represented as

$$\psi_c(\mathbf{x}, \mathbf{y}_c) = \exp\left[\langle \mathbf{w}_c, \phi_c(\mathbf{x}, \mathbf{y}_c)\rangle\right] \tag{4.4}$$

for some real-valued parameter $\mathbf{w}$ (whose restriction to clique $c$ is defined by $\mathbf{w}_c$) and a set of *feature functions* $\{\phi_c\}$. The latter are also referred to as *sufficient statistics* if we view Markov network as an example of **exponential families** [WJ08]. As a result the log conditional probability can be expressed as

$$\log P(\mathbf{y}|\mathbf{x}; \mathbf{w}) = \sum_{c \in \mathcal{C}(\mathcal{G})} \langle \mathbf{w}_c, \phi_c(\mathbf{x}, \mathbf{y}_c)\rangle - \log Z(\mathbf{x}; \mathbf{w}) \tag{4.5}$$

The choice of features vary from one application to the other. For most real life applications including vision and natural language problems, the prediction accuracy of the classifier is strongly dependent on the quality of features constituting the model. One of the simplest node features used in POS tagging just constructs joint indicator function for the different possible word- POS tag combinations. For edge potentials, accordingly indicator functions are described for pairs of tags with some dependence on the input sentence as well. For a rich ensemble of features, we refer the readers to [Rat96]. The parameter weight vector $\mathbf{w}$ is formed by *stacking up* the weight vector over the cliques $\mathbf{w}_c$. Similarly the features over cliques $\phi_c(\mathbf{x}, \mathbf{y}_c)$ can be stacked to give a feature vector $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ over the entire input-label pair.

For creating a learning model with Markov Networks there are two main problems to be solved. Given a set of labeled structured data, the problem of supervised learning of the model is solved by using some ERM scheme (mentioned in section 2.1). On the other hand, given a model and a test input, the task of finding the most likely label assignment, referred to as maximum-a-posteriori (MAP) assignment is precisely the problem of **inference**. We explore the latter problem in the context of structured prediction.

## 4.2 Inference

According to (3.1) the inference problem in structured prediction is generally solved by choosing the $\hat{\mathbf{y}}$ which maximizes the score function that decides the compatibility of labels $\mathbf{y}$ for input $\mathbf{x}$. Clearly, searching over the $\mathbf{y}$ space naively results in tractability issues as the label space is exponential and in many cases possibly infinite. Standard dynamic programming algorithms like the **Viterbi** algorithm can be used to solve this problem for sequences in $O(L)$ time where $L$ is the length of the sequence. Other approaches like approximate or variational inference [WJ08] and the max-product algorithm, based on belief propagation based approaches [KFL01] are also common for finding the most probable label assignment.

# Chapter 5

# Conditional Random Fields

A Conditional Random Field (CRF) [LMP01, SM06] is simply a model for the conditional distribution $p(\mathbf{y}|\mathbf{x})$ endowed with an associated graphical structure. Since the model is conditional, complex dependencies of the input variables $\mathbf{x}$ do not need to be explicitly represented, which allows a lot of flexibility and usage of rich features in the input.

CRFs are examples of Markov Networks ( See Definition 1) in which the conditional probability $p(\mathbf{Y}|\mathbf{X})$ satisfy the Markovian property *w.r.t.* the underlying graph $\mathcal{G}$. In what follows, $\mathbf{X}, \mathbf{Y}$ are random variables and $Y_v$ is the label value associated with the vertex $v \in \mathcal{V}$.

**Definition 2** *Given a independence graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the random variable $\mathbf{Y} = \{Y_v\}_{v \in \mathcal{V}}$ are indexed by the vertices of $\mathcal{G}$, $(\mathbf{X}, \mathbf{Y})$ is called a Conditional Random Field if the random variables $Y_v$ satisfy the Markovian property with respect to the graph* i.e. $P(Y_v|\mathbf{X}, \mathbf{Y}_{\mathcal{V}\backslash v}) = P(Y_v|\mathbf{X}, \mathbf{Y}_{N(v)})$.

Here $\mathcal{V}\backslash v$ refers to the subset of vertices of $\mathcal{V}$ except $v$ and $N(v) = \{w : w \in \mathcal{V}, (v, w) \in \mathcal{E}\}$ is the neighborhood of the vertex $v$. Thus a CRF is a random field globally conditioned on the input $\mathbf{X}$ in which the label value on one vertex is conditionally independent of other label values given its neighbors. Note that in case of sequences(chains), the edges exist only from vertex $Y_i$ to $Y_{i+1}$

and the above independence condition reduces to

$$P(Y_i|\mathbf{X}, \mathbf{Y}_{-i}) = P(Y_i|\mathbf{X}, Y_{i-1}, Y_{i+1}) \tag{5.1}$$

which is analogous to (4.1).

## 5.1 Training a CRF model

Training a CRF involves maximizing the conditional log-likelihood of the data [SM06, SP03] and is analogous to solving an ERM logistic regression problem. The conditional probability distribution of a CRF factorizes according to (4.2). Given the training data $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$ and using (4.5), the log-likelihood parametrized by the model $\mathbf{w}$ is given by

$$\begin{aligned}\mathcal{L}_{\mathbf{w}} &= \sum_{i=1}^{n} \log P(\mathbf{y}_i|\mathbf{x}_i; \mathbf{w}) \\ &= \sum_{i=1}^{n} \{\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i)\rangle - \log Z(\mathbf{x}_i; \mathbf{w})\} \end{aligned} \tag{5.2}$$

Maximizing the log-likelihood is equivalent to minimizing the negative of this term. We add a regularizer to avoid overfitting and to penalize the complexity of the solution as discussed in section 2.1. This leads to the CRF primal objective function:

$$J(\mathbf{w}) = \sum_{i=1}^{n} \{- \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i)\rangle + \log Z(\mathbf{x}_i; \mathbf{w})\} + \frac{\lambda}{2} \|\mathbf{w}\|^2 \tag{5.3}$$

There have been various approaches to minimize the CRF objective: the *improved iterative scaling* (IIS) algorithm of [DPDPL97] was used by [LMP01] in their inaugural paper on CRFs, traditional convex optimization methods like *limited memory Quasi-Newton*(L-BFGS) were used by [SP03, SM06], while [CGK$^+$08] use exponentiated gradient based methods.

We do not delve into the details of the optimization algorithms, but point out that [CGK$^+$08] achieve the best known convergence rates in terms of the precision factor $\epsilon$. The exponentiated gradient method looks at the convex dual of the objective function [Roc70] and uses a gradient descent step in the exponent to update the dual variables. The algorithm converges $\epsilon$-close to an optimum solution in $O(\log(1/\epsilon))$ iterations.

## 5.2 Inference in CRFs

Given a model $\mathbf{w}$ and a new input $\hat{\mathbf{x}}$, the most probable label is the one which maximizes the log-likelihood.

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \log P(\mathbf{y}|\hat{\mathbf{x}};\mathbf{w}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{w}, \phi(\hat{\mathbf{x}},\mathbf{y})\rangle \tag{5.4}$$

We do not consider the log of the partition function, as it is independent of $\mathbf{y}$. Using the factorization of the features over the cliques of the underlying graph, we can adapt the standard Viterbi algorithm of [Rab89] used for inference in Hidden Markov Models to perform inference in CRFs.

Inference is a key step in most algorithms related to CRFs. During training, a variant of the forward-backward algorithm is used to calculate the partition function $Z(\mathbf{x})$. For most acyclic graphs, inference can be performed via efficient dynamic programming schemes. Here we describe the forward backward and Viterbi approach for sequences [LMP01, SM06].

For simplicity we assume that the label $y_i$ at each position $i$ in the sequence is drawn from the same alphabet $\mathbb{Y}$. We assume that the length of the sequence is $L$ and add dummy labels $y_0$ and $y_{L+1}$ at the beginning and end of the sequence which serve as **START** and **STOP** states in the dynamic programming. For each position $i \in \{1, 2, \ldots, L+1\}$ in the sequence, we define a $\mathbb{Y} \times \mathbb{Y}$ dimensional matrix $M_i(\mathbf{x}) = \{M_i(y', y|\mathbf{x})\}$ which has an element for all possible pairs of label interactions

$$M_i(y', y|\mathbf{x}) = \exp(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}|_i = \{y', y\})\rangle) \tag{5.5}$$

where $\mathbf{y}|_i$ is the sequence $\mathbf{y}$ restricted to the $i$th edge *i.e.* the clique $(y_{i-1}, y_i)$.

Since cliques in a sequence graph are just the edges, therefore using (4.2) we know that the conditional probability of the label can be written down as the product of appropriate elements of the $(L+1)$ matrices

$$P(\mathbf{y}|\mathbf{x};\mathbf{w}) = \frac{1}{Z(\mathbf{x};\mathbf{w})} \prod_{c\in\mathcal{C}(\mathcal{G})} \psi_c(\mathbf{x}, \mathbf{y}_c) = \frac{1}{Z(\mathbf{x};\mathbf{w})} \prod_{i=1}^{L+1} M_i(y_{i-1}, y_i|\mathbf{x})$$

Writing down the partition function ((4.3)) using the distributive law [SM06], it can be shown that $Z(\mathbf{x};\mathbf{w})$ is equal to the (**START**,**STOP**) entry of the product of all the matrices *i.e.*

$$Z(\mathbf{x};\mathbf{w}) = \left(\prod_{i=1}^{L+1} M_i(\mathbf{x})\right)_{(\mathbf{START},\mathbf{STOP})} \tag{5.6}$$

Most training algorithms like IIS or gradient descent require the evaluation of $\sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{x};\mathbf{w})$ which decomposes as

$$\sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{Y}=\mathbf{y}|\mathbf{x};\mathbf{w}) = \sum_{i=1}^{L} \sum_{y',y\in\mathbb{Y}} P(Y_{i-1}=y', Y_i=y|\mathbf{x};\mathbf{w}) \tag{5.7}$$

Note that this is a polynomial summation over $L|\mathbb{Y}|^2$ terms which avoids intractability as a naive summation over the entire label space would have required summing over an exponential number of terms. A forward backward style algorithm [Rab89] can be used to evaluate each of the terms on the RHS of (5.7) [LMP01]. For each position in the sequence $i = 0, \ldots L+1$, we define corresponding *forward variables* $\boldsymbol{\alpha}_i(\mathbf{x})$, each of which is a vector $\{\alpha_i(y|xb)\}$ of size $|\mathbb{Y}|$ and has an entry for every $y \in \mathbb{Y}$. The forward variables are initialized as

$$\alpha_0(y|\mathbf{x}) = \begin{cases} 1 & \text{if } y = \mathbf{START} \\ 0 & \text{otherwise} \end{cases}$$

while the variable at the position $i$ is recursively updated as

$$\alpha_i(y|\mathbf{x}) = \sum_{y'\in\mathbb{Y}} \alpha_{i-1}(y'|\mathbf{x}) M_i(y', y|\mathbf{x})$$

The above recurrence relation can be expressed concisely as the following vector updates

$$\boldsymbol{\alpha}_i(\mathbf{x})^\top = \boldsymbol{\alpha}_{i-1}(\mathbf{x})^\top M_i(\mathbf{x}) \ . \tag{5.8}$$

Similarly we can define backward variables $\{\beta_i(y|\mathbf{x})\}$ for each position $i$ in the sequence which form the $|\mathbb{Y}|$ dimensional vectors $\boldsymbol{\beta}_i(\mathbf{x})$. These variables are initialized and updated in an analogous way

$$\beta_{L+1}(y|\mathbf{x}) = \begin{cases} 1 & \text{if } y = \textbf{STOP} \\ 0 & \text{otherwise} \end{cases} \tag{5.9a}$$

$$\boldsymbol{\beta}_i(\mathbf{x}) = M_{i+1}(\mathbf{x})\boldsymbol{\beta}_{i+1}(\mathbf{x}) \tag{5.9b}$$

Using the forward and backward recursions, each term in the RHS of (5.7) can be calculated as

$$P(Y_{i-1} = y', Y_i = y|\mathbf{x};\mathbf{w}) = \frac{1}{Z(\mathbf{x};\mathbf{w})}\boldsymbol{\alpha}_{i-1}(y'|\mathbf{x})M_i(y',y|\mathbf{x})\beta_i(y|\mathbf{x})$$

where the partition function is calculated efficiently using (5.6).

The other inference problem encountered is that of the most probable label assignment (5.4) which requires us to calculate $\operatorname{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x};\mathbf{w})$. In this case we can still perform analogous updates by replacing the summation by maximization terms. For this inference problem, we define Viterbi vectors $\boldsymbol{\gamma}_i(\mathbf{x})$ of length $|\mathbb{Y}|$ for each position $i$ in the sequence. These variables are initialized and updated according to the following rules

$$\gamma_0(y|\mathbf{x}) = \begin{cases} 1 & \text{if } y = \textbf{START} \\ 0 & \text{otherwise} \end{cases}$$

$$\forall i, \qquad \gamma_i(y|\mathbf{x}) = \max_{y' \in \mathbb{Y}} \gamma_{i-1}(y'|\mathbf{x})M_i(y',y|\mathbf{x})$$

The most probable label sequence assignment is obtained by finding the last label in the sequence $\operatorname{argmax}_{y\in\mathbb{Y}} \gamma_L(y|\mathbf{x})$ and then backtracking along the $\boldsymbol{\gamma}_i$ values to evaluate which intermediate labels generated the most probable label sequence. This is the same procedure as standard viterbi decoding.

# Chapter 6

# Max-Margin Structured Prediction

While Conditional Random Fields are probabilistic models which aim to train their parameters based on maximizing the log-likelihood of the data, methods based on maximizing the margin like Max-Margin Markov Networks ($\mathsf{M}^3\mathsf{N}$) [TGK04], Structured SVMs (SVMStruct) [TJHA05] are aimed at training a model which *maximizes* the score of the input with the correct label beyond the score with any other label by a certain *margin*.

These methods also employ an ERM scheme where the loss function used is the structured version of hinge loss. The concept of margin is defined as follows: Given a structured input $\hat{\mathbf{x}}$ and its correct label $\hat{\mathbf{y}}$, the score function (given by (3.2)) for this pair has to be higher than the score of $(\hat{\mathbf{x}}, \mathbf{y})$, $\mathbf{y} \neq \hat{\mathbf{y}}$, by a certain margin. This approach aligns the model with the features obtained from the correct input label pair and forces this alignment to be the best among all possible labels by a significant amount. Mathematically this can be depicted as

$$\forall \mathbf{y} \in \mathcal{Y} \qquad \langle \mathbf{w}, \boldsymbol{\phi}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \rangle \geq \langle \mathbf{w}, \boldsymbol{\phi}(\hat{\mathbf{x}}, \mathbf{y}) \rangle + \ell(\mathbf{y}, \hat{\mathbf{y}}) \tag{6.1}$$

The margin $\ell(\mathbf{y}, \hat{\mathbf{y}})$ is a measure of how much $\mathbf{y}$ is separated from $\hat{\mathbf{y}}$ and intuitively defines a distance metric on the $\mathcal{Y}$ space. In particular, $\ell(\hat{\mathbf{y}}, \hat{\mathbf{y}}) = 0$. While the margin gap should ideally be the 0/1 indicator distance between $\mathbf{y}$ and $\hat{\mathbf{y}}$, this concept of distance often does not convey enough information about the structured labels. A common choice for $\ell(\mathbf{y}, \hat{\mathbf{y}})$ for sequences is the Hamming distance between the two label sequences.

Given this definition of margin, the hinge loss is defined naturally for an input label pair $(\mathbf{x}_i, \mathbf{y}_i)$.

$$L(\mathbf{x}_i, \mathbf{y}_i, \mathbf{w}) = \max_{\mathbf{y} \in \mathcal{Y}} \ell(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) \rangle + \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) \rangle \tag{6.2}$$

The classifier is penalized by the maximum amount by which (6.1) is violated. Denoting $\boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) - \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}) = \boldsymbol{\psi}^i_{\mathbf{y}}$ and $\ell(\mathbf{y}_i, \mathbf{y})$ as $\ell^i_{\mathbf{y}}$, the corresponding primal regularized objective to be minimized can be written down as

$$J(\mathbf{w}) = C \sum_{i=1}^{n} \max_{\mathbf{y} \in \mathcal{Y}} \left\{ \ell^i_{\mathbf{y}} - \left\langle \mathbf{w}, \boldsymbol{\psi}^i_{\mathbf{y}} \right\rangle \right\} + \frac{\lambda}{2} \|\mathbf{w}\|^2 \tag{6.3}$$

where $C$ is a constant (For empirical average $C = \frac{1}{n}$) and $\lambda$ is the regularization parameter. This formulation has been the most common max margin structured framework [TGK04, TLJJ06, CGK$^+$08] and is referred to as the margin-rescaling approach in[TJHA05]. By defining slack variables $\xi_i$ for each $i$ corresponding to the loss term we can rewrite the primal problem as

$$\min_{\mathbf{w}, \xi_i} \quad J(\mathbf{w}) = \left\{ C \sum_i \xi_i + \frac{\lambda}{2} \|\mathbf{w}\|^2 \right\} \qquad \text{such that} \tag{6.4}$$

$$\forall i, \forall \mathbf{y} \in \mathcal{Y}, \qquad \xi_i \geq \ell^i_{\mathbf{y}} - \left\langle \mathbf{w}, \boldsymbol{\psi}^i_{\mathbf{y}} \right\rangle \qquad \text{and}$$

$$\forall i, \qquad \xi_i \geq 0$$

Borrowing concepts from convex duality and introducing Lagrangian variables $\alpha^i_{\mathbf{y}}$ corresponding to each constraint, we get the Lagrangian dual of (6.4) as:

$$\max_{\boldsymbol{\alpha}} \quad D(\boldsymbol{\alpha}) = \sum_i \sum_{\mathbf{y}\in\mathcal{Y}} \alpha^i_{\mathbf{y}} \ell^i_{\mathbf{y}} - \frac{1}{2\lambda} \sum_{i,\mathbf{y}\in\mathcal{Y}} \sum_{j,\mathbf{y}'\in\mathcal{Y}} \alpha^i_{\mathbf{y}} \alpha^j_{\mathbf{y}'} \left\langle \boldsymbol{\psi}^i_{\mathbf{y}}, \boldsymbol{\psi}^j_{\mathbf{y}'} \right\rangle \tag{6.5}$$

$$\text{such that} \qquad \forall i \qquad \sum_{\mathbf{y}\in\mathcal{Y}} \alpha^i_{\mathbf{y}} = C \qquad \text{and}$$

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \qquad \alpha^i_{\mathbf{y}} \geq 0$$

and the $\mathbf{w}$ optimizing (6.4) is given by

$$\mathbf{w}^* = \sum_i \sum_{\mathbf{y}\in\mathcal{Y}} {\alpha^*}^i_{\mathbf{y}} \boldsymbol{\psi}^i_y$$

where $\boldsymbol{\alpha}^*$ maximizes the dual problem $D(\boldsymbol{\alpha})$. Note that the second term in (6.5) can also be written as $\frac{1}{2\lambda}\boldsymbol{\alpha}^\top K \boldsymbol{\alpha}$ where $K$ is the Gram matrix of a *joint kernel* function over the $\mathcal{X} \times \mathcal{Y}$ space, *i.e.* $K : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \rightarrow \mathbb{R}$ given by

$$K((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_j, \mathbf{y}')) = \left\langle \boldsymbol{\psi}^i_{\mathbf{y}}, \boldsymbol{\psi}^j_{\mathbf{y}'} \right\rangle$$

A number of different approaches have been adopted to efficiently solve the optimization problems (6.4)-(6.5). Since the dual version requires summing over all possible $\mathbf{y}$'s (which might be possibly exponential in number), naive approaches do not lead to efficient algorithms. Most optimization problems attempt to optimize the dual objective since the primal is non-smooth due to the presence of a max over linear terms. In [TGK04], the authors try to adapt a SMO style approach (due to [Pla99]) to optimize (6.5) while [TJHA05] try to use *cutting plane* methods [Kel60] to search through the space of labels efficiently while optimizing the dual objective function. The authors of [TLJJ06] try to use dual extragradient techniques to optimize the dual, while the state of the art methods due to [CGK$^+$08] use exponentiated gradient methods to perform a gradient descent style step in the exponent. They converge to within $\epsilon$ approximation of the optimum in $O(1/\epsilon)$ iterations.

Most of the above approaches avoid the problem of tractability by computing marginals over the cliques of the graphical model, which are generally of polynomial order. The space of labels $\mathcal{Y}$ is endowed with an undirected graphical model structure $\mathcal{G}$. The margin function $\ell^i_{\mathbf{y}}$ and the features $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ decompose according to the cliques of $\mathcal{G}$ as

$$\ell^i_y = \ell(\mathbf{y}, \mathbf{y}_i) = \sum_{c \in \mathcal{C}(\mathcal{G})} \ell(\mathbf{y}_c, \mathbf{y}_{ic}), \qquad \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) = \bigoplus_{c \in \mathcal{C}(\mathcal{G})} \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}_c) \qquad \text{and} \qquad \boldsymbol{\psi}^i_{\mathbf{y}} = \bigoplus_{c \in \mathcal{C}(\mathcal{G})} \boldsymbol{\psi}^i_{\mathbf{y}_c}$$

The key advantage of the graphical model structure is that the marginals on the cliques can be efficiently computed. In particular, given the dual variables $\{\alpha^i_{\mathbf{y}}\}$, the marginals over a clique $c$ are given by

$$\alpha^i_{\mathbf{y}_c} = \sum_{\mathbf{y}' : \mathbf{y}'|_c = \mathbf{y}_c} \alpha^i_{\mathbf{y}} \tag{6.6}$$

where the summation is over all the labels $\mathbf{y}'$ whose restriction on the clique $c$ is the same as $\mathbf{y}_c$. Although $\mathcal{Y}$ can be exponentially large, efficient dynamic programming algorithms exist that exploit the factorized form (4.2), *e.g.* belief propagation [Lau96]. Efficient algorithms exist as long as the associated graphical model has **low-treewidth** [Tas04]. The computational cost is $O(s^\omega)$ where $s$ is the number of states of each node, and $\omega$ is the maximum size of the cliques. For example, a linear chain has $\omega = 2$. When $\omega$ is large, approximate algorithms also exist [WJ08, AdFDJ03, KFL01].

The inference procedure for $\mathsf{M}^3\mathsf{N}$ also follow the same approach as that of CRFs since the most probable label assignment is evaluated using (5.4). Thus the Viterbi algorithm can be used to efficiently predict the label with the highest score for a given input $\mathbf{x}$.

# Chapter 7

# Vector Valued Functions

While there is a rich literature on approaches to structured prediction, they focus excessively on *joint* features extracted from the data. This leads to the use of joint kernels which are defined on the $\mathcal{X} \times \mathcal{Y}$ space. Although this allows for a richer structure, it also restricts development in a number of different directions, particularly semi-supervised learning- which involves learning from a small number of labeled and a very large number of unlabeled data. Also, the complexity of the structure of the $\mathcal{X}$ and $\mathcal{Y}$ are not captured thoroughly in the above approaches.

We introduce a new formulation for structured prediction framework. Given the space of outputs $\mathcal{Y}$, we define a kernel $K_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ which analogously gives us an RKHS $\mathcal{H}_{\mathcal{Y}}$ of real valued functions on the $\mathcal{Y}$ space endowed with dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}_{\mathcal{Y}}}$. Every element $\mathbf{y} \in \mathcal{Y}$ can be mapped to $\tilde{\mathbf{y}} = K_{\mathcal{Y}}(\mathbf{y}, \cdot) \in \mathcal{H}_{\mathcal{Y}}$. We now define a Hilbert space of **vector-valued** functions [CVT06]

**Definition 3** *A Hilbert space $\mathcal{H} = \{f : \mathcal{X} \to \mathcal{H}_{\mathcal{Y}}\}$ endowed with a dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is a Reproducing Kernel Hilbert Space (RKHS) if $\forall \mathbf{x} \in \mathcal{X}$, there exists a positive constant $C_{\mathbf{x}}$ such that*

$$\forall f \in \mathcal{H} \qquad \|f(\mathbf{x})\|_{\mathcal{H}_{\mathcal{Y}}} \leq C_{\mathbf{x}} \|f\|_{\mathcal{H}}$$

An equivalent definition is that $\mathcal{H}$ is an RKHS when $\forall \mathbf{x} \in \mathcal{X}, \forall \tilde{\mathbf{y}} \in \mathcal{H}_\mathcal{Y}$ the linear functional mapping $f \in \mathcal{H}$ to $\langle \tilde{\mathbf{y}}, f(\mathbf{x}) \rangle_{\mathcal{H}_\mathcal{Y}}$ is continuous. Using the standard Riesz Representation Lemma [AG93], we know that $\forall \mathbf{x} \in \mathcal{X}, \tilde{\mathbf{y}} \in \mathcal{H}$ there is a function of the form $\mathcal{K}_\mathbf{x}\tilde{\mathbf{y}} \in \mathcal{H}$ such that

$$\forall f \in \mathcal{H} \qquad \langle \tilde{\mathbf{y}}, f(\mathbf{x}) \rangle_{\mathcal{H}_\mathcal{Y}} = \langle \mathcal{K}_\mathbf{x}\tilde{\mathbf{y}}, f \rangle_\mathcal{H} \tag{7.1}$$

This helps us in defining an operator valued kernel $\mathcal{K}$ on vector valued function spaces. (7.1) is the equivalent of the reproducing property in vector valued spaces. The function $\mathcal{K}_\mathbf{x}\tilde{\mathbf{y}}$ can be used to define the following linear operator. Given $\mathbf{x}, \mathbf{x}' \in \mathcal{X}, \quad \mathcal{K}(\mathbf{x}, \mathbf{x}') : \mathcal{H}_\mathcal{Y} \to \mathcal{H}_\mathcal{Y}$ is defined as

$$\mathcal{K}(\mathbf{x}, \mathbf{x}')\tilde{\mathbf{y}} := \mathcal{K}_{\mathbf{x}'}\tilde{\mathbf{y}}(\mathbf{x}) \in \mathcal{H}_\mathcal{Y}$$

Denoting $\mathcal{L}(\mathcal{H}_\mathcal{Y})$ as the space of linear operators on $\mathcal{H}_\mathcal{Y}$, we can now define an operator valued kernel $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{H}_\mathcal{Y})$. For more details about vector valued kernels, we direct the readers to [MP05, CVT06]. In particular, $\mathcal{K}$ satisfies the positive semi-definiteness property and $\mathcal{K}(\mathbf{x}, \mathbf{x})$ is a positive definite operator. It can be proved that there is an RKHS $\mathcal{H}_\mathcal{K}$ of functions $f : \mathcal{X} \to \mathcal{H}_\mathcal{Y}$ which admits $\mathcal{K}$ as its reproducing kernel.

Our approach is to attempt structured prediction with the above machinery of vector valued functions. Given the space of labels $\mathcal{Y}$ we generate a Hilbert space $\mathcal{H}_\mathcal{Y}$ of functions (that can be considered as vectors) as the RKHS of a kernel $\mathcal{K}_\mathcal{Y}$. The definition of this kernel function will ideally vary across particular structured classification tasks like POS tagging, Optical Character Recognition, Named Entity Recognition. Depending on the context, we can define string or graph kernels [LSST$^+$02, VBSK08] on $\mathcal{Y}$ which give us a rich function space $\mathcal{H}_\mathcal{Y}$.

We define an operator valued kernel $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{H}_\mathcal{Y})$ which in turn specifies an RKHS $\mathcal{H}_\mathcal{K}$ of vector valued functions. For standard structured learning, our formulation is two-fold. Given a set of $n$ labeled examples $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$, we first map the labels $\mathbf{y}_i$'s to the corresponding vectors(or functions, depending upon the setup) $\tilde{\mathbf{y}}$ in $\mathcal{H}_{\mathcal{Y}}$ using the kernel function $\mathcal{K}_{\mathcal{Y}}$. The training step consists of learning a vector valued function $g^\star : \mathcal{X} \to \mathcal{H}_{\mathcal{Y}}$ by solving an ERM to minimize a regularized risk functional

$$g^\star = \operatorname*{argmin}_{g \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{x}_i, \tilde{\mathbf{y}}_i, g) + \lambda \|g\|_{\mathcal{H}}^2 \tag{7.2}$$

Intuitively $g^\star$ is the vector valued function which best fits the data and generalizes to unknown structured data. Once the classifier $g^\star$ is obtained, given a new $\hat{\mathbf{x}}$, we solve a **pre-image** problem [WCE+03, CMW05] to obtain the best possible label in $\mathcal{Y}$.

$$\mathbf{y}^\star = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} d(g^\star(\hat{\mathbf{x}}), \tilde{\mathbf{y}}) \tag{7.3}$$

The pre-image problem finds the $\mathbf{y} \in \mathcal{Y}$ whose mapping in the $\mathcal{H}_{\mathcal{Y}}$ space is closest to the image of $\hat{\mathbf{x}}$ under $g^\star$. Note that the concept of alignment $d(\cdot)$ in $\mathcal{H}_{\mathcal{Y}}$ to solve the pre-image problem is flexible and varies according to the structured prediction applications. A general example would be the squared loss which is most prevalent in the literature.

$$d(g^\star(\hat{\mathbf{x}}), \tilde{\mathbf{y}}) = \|g^\star(\hat{\mathbf{x}}) - \tilde{\mathbf{y}}\|_{\mathcal{H}_{\mathcal{Y}}}^2$$

Another example of the alignment function would be most natural to use in case of vector-valued SVMs ( See section 8.2). In that case it is natural to consider the most probable $\mathbf{y}$ as the one which is best aligned with the classifier evaluated on $\hat{\mathbf{x}}$. In other words,

$$d(g^\star(\hat{\mathbf{x}}), \tilde{\mathbf{y}}) = -\left\langle g^\star(\hat{\mathbf{x}}), \tilde{\mathbf{y}} \right\rangle_{\mathcal{H}_{\mathcal{Y}}}$$

It is interesting to note the analogies that can be drawn to connect our approach to the other prevalent margin based structured learning approaches. The pre-image problem (7.3), is equivalent to the maximization performed in (5.4) whereas (7.2) is analogous to the minimization performed (5.3),(6.3) while training the model $\mathbf{w}$.

Depending on the loss function used in (7.2), we get analogous versions of Vector Valued Regression and Vector valued SVMs which we explore in the next chapter. We will also show that using a variant of the famous Representer theorem (2.5) the minimizer $g^\star$ of (7.2) can be expressed in the span of the operator kernel $\mathcal{K}$ evaluated on the input data $\{\mathbf{x}_i\}$. Although there can be several possible candidates for $\mathcal{K}$ [MP05], we choose the heat kernel [BGV04, BN04]

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \sum_i \exp(\lambda_i t) e_i(\mathbf{x}) e_i(\mathbf{x}') \tag{7.4}$$

which is now an operator on the $\mathcal{H}_\mathcal{Y}$ space. Here $\{\lambda_i, e_i\}$ form an eigenvalue-eigenvector system of the Laplacian operator [BN04] applied on functions in $\mathcal{H}_\mathcal{K}$ and $t$ is a parameter. Thus $e_i : \mathcal{X} \to \mathcal{H}_\mathcal{Y}$. The operator in (7.4) acts on a particular $\tilde{\mathbf{y}} \in \mathcal{H}_\mathcal{Y}$ in the following way.

$$\mathcal{K}(\mathbf{x}, \mathbf{x}')(\tilde{\mathbf{y}}) = \sum_i \exp(\lambda_i t) e_i(\mathbf{x}) \left\langle e_i(\mathbf{x}'), \tilde{\mathbf{y}} \right\rangle \quad \in \mathcal{H}_\mathcal{Y}$$

# Chapter 8

# Formulation for different Loss Functions

In this chapter, we look at different loss functions and the corresponding formulations for vector valued functions.

## 8.1 Ridge Regression on Vector valued functions

We follow the standard supervised learning scenario. Given a data set $\{(\mathbf{x}_1, \mathbf{y}_1) \ldots (\mathbf{x}_n, \mathbf{y}_n)\}$ with $\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}$, we want to learn a hypothesis that best fits the data for classification. As described in the last chapter, we define a kernel $K_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$, which defines a Hilbert space $\mathcal{H}_{\mathcal{Y}}$ of functions on $\mathcal{Y}$. Using this kernel we map each $\mathbf{y} \in \mathcal{Y}$ to corresponding functions(vectors) $\tilde{\mathbf{y}} = K_{\mathcal{Y}}(\mathbf{y}, \cdot) \in \mathcal{H}_{\mathcal{Y}}$. We now have a data set of the form $\{(\mathbf{x}_1, \tilde{\mathbf{y}}_1) \ldots (\mathbf{x}_n, \tilde{\mathbf{y}}_n)\}$ on which we run the ERM problem with squared loss.

Using the notations of the last chapter, we specify the Hilbert Space of functions $\mathcal{H}_{\mathcal{K}} =$

$\{f : \mathcal{X} \to \mathcal{H}_\mathcal{Y}\}$ which is an RKHS corresponding to the kernel $\mathcal{K}$. The corresponding minimization problem turns out to be

$$\min_{f \in \mathcal{H}_\mathcal{K}} J(f) = \frac{1}{n} \sum_{i=1}^{n} \|f(\mathbf{x}_i) - \tilde{\mathbf{y}}_i\|^2_{\mathcal{H}_\mathcal{Y}} + \lambda \|f\|^2_{\mathcal{H}_\mathcal{K}} \tag{8.1}$$

The following theorem proves a variant of the Representer theorem (2.5) and characterizes the nature of the solution [MP05].

**Theorem 4** *The minimizer $f^*$ of $J(f)$ in (8.1) is unique and can be represented as*

$$f^* = \sum_{i=1}^{n} \mathcal{K}_{\mathbf{x}_i} \mathbf{c}_i \tag{8.2}$$

*where the function coefficients $\mathbf{c}_i \in \mathcal{H}_\mathcal{Y}$ are the unique solutions of the linear equations*

$$\forall i \in [n] \qquad \sum_{j=1}^{n} \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)\mathbf{c}_j + \lambda \mathbf{c}_i = \tilde{\mathbf{y}}_i \tag{8.3}$$

**Proof** Consider $\mathcal{H}_* = \{\mathcal{K}_{\mathbf{x}_i}\mathbf{c} | i \in [n], \mathbf{c} \in \mathcal{H}_\mathcal{Y}\}$ be the space spanned by all possible functions which can be expressed by (8.2). Given any function $f \in \mathcal{H}_\mathcal{K}$, it can be expressed as the sum of two orthogonal components - one in the span of $\mathcal{K}_{\mathbf{x}_i}$ (and thus lying in $\mathcal{H}_*$) and the other one orthogonal to $\mathcal{H}_*$. Thus

$$f = \sum_{j=1}^{n} \mathcal{K}_{\mathbf{x}_j} \mathbf{c}_j + f^\perp \tag{8.4}$$

where $\mathbf{c}_i \in \mathcal{H}_\mathcal{Y}$ are functional coefficients and $f^\perp$ is orthogonal to $\mathcal{H}_\mathcal{Y}$. Using (7.1)

$$\forall j \in [n], \mathbf{c} \in \mathcal{H}_\mathcal{Y} \qquad \left\langle \mathcal{K}_{\mathbf{x}_j}\mathbf{c}, f^\perp \right\rangle_{\mathcal{H}_\mathcal{K}} = \left\langle \mathbf{c}, f^\perp(\mathbf{x}_j) \right\rangle_{\mathcal{H}_\mathcal{Y}} = 0$$

Since the above relation holds for all $\mathbf{c} \in \mathcal{H}_\mathcal{Y}$ it is easy to deduce that $f^\perp(\mathbf{x}_j) = 0$ for all $j \in [n]$.

Plugging this back in (8.1) and denoting $\sum_{j=1}^{n} \mathcal{K}_{\mathbf{x}_j}\mathbf{c}_j$ as $f^{\|}$ we get

$$\begin{aligned} J(f) &= \frac{1}{n}\sum_{i=1}^{n}\left\|\sum_{j=1}^{n}\mathcal{K}(\mathbf{x}_i,\mathbf{x}_j)\mathbf{c}_j + f^{\perp}(\mathbf{x}_i) - \tilde{\mathbf{y}}_i\right\|_{\mathcal{H}_\mathcal{Y}}^2 + \lambda\left\|f^{\|} + f^{\perp}\right\|_{\mathcal{H}_\mathcal{K}}^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\left\|\sum_{j=1}^{n}\mathcal{K}(\mathbf{x}_i\mathbf{x}_j)\mathbf{c}_j - \tilde{\mathbf{y}}_i\right\|_{\mathcal{H}_\mathcal{Y}}^2 + \lambda(\left\|f^{\|}\right\|_{\mathcal{H}_\mathcal{K}}^2 + \left\|f^{\perp}\right\|_{\mathcal{H}_\mathcal{K}}^2) \end{aligned}$$

Since $f^{\|}$ and $f^{\perp}$ lie in orthogonal spaces, the dot product terms in the second expression all yield zero. Thus we see that $f^{\perp}$ just increases the regularizer term while not contributing at all to the loss term. Thus the objective can be minimized further by setting $f^{\perp} = 0$. Thus the optimum $f^*$ lies in $\mathcal{H}_*$, *i.e.* there exist $\{\mathbf{c}_i\}$ such that

$$f^* = \sum_{i=1}^{n}\mathcal{K}_{\mathbf{x}_i}\mathbf{c}_i$$

Next, we prove that the coefficients $\{\mathbf{c}_i\}$ satisfy the linear system (8.3). In the sequel we omit the space with respect to which the norm or inner products are evaluated when its obvious. The objective function (8.1) can be expressed as

$$\|f(\mathbf{x}_i)\|^2 - 2\langle\tilde{\mathbf{y}}_i, f(\mathbf{x}_i)\rangle + \|\tilde{\mathbf{y}}\|^2 + \lambda\|f\|_{\mathcal{H}_\mathcal{K}}^2$$

To find the optimum $f^*$, we take gradient of the objective with respect to $f$ and set it equal to zero. Using the reproducing property (7.1), the gradient is given by

$$2\sum_{i=1}^{n}\left[\mathcal{K}_{\mathbf{x}_j}f^*(\mathbf{x}_j) - 2\mathcal{K}_{\mathbf{x}_j}\tilde{\mathbf{y}}_j\right] + 2\lambda f^* = 0$$

Plugging in the value of the optimum $f^*$ (8.2) and using the linearity of the $\mathcal{K}_\mathbf{x}$ operator , we get

$$2\sum_{i=1}^{n}\mathcal{K}_{\mathbf{x}_i}\left(\sum_{j=1}^{n}\mathcal{K}(\mathbf{x}_i,\mathbf{x}_j)\mathbf{c}_j - \tilde{\mathbf{y}}_i + \lambda\mathbf{c}_i\right) = 0$$

The above equality can be satisfied if each of the inner terms is 0 which gives (8.3).

Finally, we prove that this representation is unique. Suppose there exist two sets of coefficients $\{\mathbf{c}_j\}$ and $\{\mathbf{d}_j\}$, not all same, such that

$$\forall i \in [n] \qquad \sum_{j=1}^{n} \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)\mathbf{c}_j + \lambda \mathbf{c}_i = \tilde{\mathbf{y}}_i$$
$$\forall i \in [n] \qquad \sum_{j=1}^{n} \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)\mathbf{d}_j + \lambda \mathbf{d}_i = \tilde{\mathbf{y}}_i$$

Setting $\mathbf{g}_i = \mathbf{c}_i - \mathbf{d}_i$ we have

$$\forall i \in [n] \qquad \sum_{j=1}^{n} \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)\mathbf{g}_j + \lambda \mathbf{g}_i = 0$$

Taking dot products on both sides with $\mathbf{g}_i$ yields $\forall i \in [n]$,

$$\sum_{j=1}^{n} \langle \mathbf{g}_i, \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)\mathbf{g}_j \rangle_{\mathcal{H}_\mathcal{Y}} + \lambda \left\| \mathbf{g}_i \right\|^2_{\mathcal{H}_\mathcal{Y}} = 0$$

Summing over $i$ and using (7.1) gives

$$\begin{aligned} \sum_{i=1}^{n}\sum_{j=1}^{n} \langle \mathbf{g}_i, \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)\mathbf{g}_j \rangle_{\mathcal{H}_\mathcal{Y}} + \lambda \sum_{i=1}^{n} \left\| \mathbf{g}_i \right\|^2_{\mathcal{H}_\mathcal{Y}} &= 0 \\ \implies \quad \left\langle \sum_i \mathcal{K}_{\mathbf{x}_i}\mathbf{g}_i, \sum_j \mathcal{K}_{\mathbf{x}_j}\mathbf{g}_j \right\rangle_{\mathcal{H}_\mathcal{K}} + \lambda \sum_{i=1}^{n} \left\| \mathbf{g}_i \right\|^2_{\mathcal{H}_\mathcal{Y}} &= 0 \\ \implies \left\| \sum_i \mathcal{K}_{\mathbf{x}_i}\mathbf{g}_i \right\|^2_{\mathcal{H}_\mathcal{K}} + \lambda \sum_{i=1}^{n} \left\| \mathbf{g}_i \right\|^2_{\mathcal{H}_\mathcal{Y}} &= 0 \end{aligned}$$

Since the right hand side is a sum of squares which adds up to 0, each individual term is 0, thus giving $\mathbf{g}_i = 0$, for all $i$ which in turn yields that $\mathbf{c}_i = \mathbf{d}_i$ and the representation is unique. ■

When the functions $\mathbf{c}_i$ are vectors of finite dimension $d$, (8.3) results in $nd$ equations in all with $nd$ constraints which should have a non-trivial solution that can be solved to obtain the vectors $\mathbf{c}_i$. Exact state-of-the art algorithms for solving this system take $O((nd)^{2.376})$ time [CW87]. However an approximate solution by solving the quadratic program (8.1) should have a lower dependence on $d$ and $n$. Handling arbitrary functions $\mathbf{c}_i$ (which might be infinite dimensional) remains open for future work.

## 8.2 Hinge Loss: Vector Valued SVMs

For the hinge loss formulation, we define a multiclass-hinge loss analogous to (6.3). Given a dataset $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{n}$, we define a concept of margin $\ell(\tilde{\mathbf{y}}, \tilde{\mathbf{y}}')$ or in the vein of chapter 6. Thus $\ell(\tilde{\mathbf{y}}, \tilde{\mathbf{y}}') = 0$ if $\tilde{\mathbf{y}}' = \tilde{\mathbf{y}}$ and it is a measure of the distance by which the two functions are separated in the $\mathcal{H}_\mathcal{Y}$ space. Note that since $\tilde{\mathbf{y}}$ are modeled as functions( or vectors) in $\mathcal{H}_\mathcal{Y}$ space, we can use more expressive measures of distance as compared to Hamming loss which is generally used for standard margin based structured prediction [TGK04]. In particular, we can use norm distance between $\mathbf{y}$ and $\tilde{\mathbf{y}}$ which depends on how the RKHS $\mathcal{H}_\mathcal{Y}$ is. As in chapter 6, we denote $\ell^i_{\tilde{\mathbf{y}}}$ to denote $\ell(\tilde{\mathbf{y}}_i, \tilde{\mathbf{y}})$. Using slack variables, the primal optimization function can be written down as

$$\min_{f,\xi_i} C \sum_i \xi_i + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}_\mathcal{K}} \qquad \text{such that} \tag{8.5}$$

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \qquad \xi_i \geq \ell^i_{\tilde{\mathbf{y}}} - \langle f(\mathbf{x}_i), \tilde{\mathbf{y}}_i - \tilde{\mathbf{y}} \rangle \qquad \text{and}$$

$$\forall i, \xi_i \geq 0 \tag{8.6}$$

Note that we define the set of constraints only for the $\tilde{\mathbf{y}} \in \mathcal{H}_\mathcal{Y}$ obtained from the $\mathbf{y} \in \mathcal{Y}$. Thus we reduce the set of constraints from possibly infinite( for all $\tilde{\mathbf{y}} \in \mathcal{H}_\mathcal{Y}$) to possibly exponential, since they are only defined for each possible label in $\mathcal{Y}$. Also, the constraints define the concept of margin based loss, since this is equivalent to enforcing that the alignment between $f(\mathbf{x}_i)$ and $\tilde{\mathbf{y}}_i$ is better than that with arbitrary $\tilde{\mathbf{y}}$ by a certain margin. In other words, this loss formulation tries to enforce for every example $(\mathbf{x}_i, \mathbf{y}_i)$

$$\forall y \in \mathcal{Y}, \langle \tilde{\mathbf{y}}_i, f(\mathbf{x}_i) \rangle \geq \langle \tilde{\mathbf{y}}, f(\mathbf{x}_i\rangle + \ell(\tilde{\mathbf{y}}_i, \tilde{\mathbf{y}})$$

Defining dual variables $\alpha^i_{\mathbf{y}}$ corresponding to each $i \in [n]$ and $\mathbf{y} \in \mathcal{Y}$, we can express the Lagrangian as

$$\mathcal{L}(f, \boldsymbol{\xi}, \boldsymbol{\alpha}) = C \sum_i \xi_i + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}_\mathcal{K}} + \sum_{i, \mathbf{y} \in \mathcal{Y}} \alpha^i_{\mathbf{y}} \left( \ell^i_{\tilde{\mathbf{y}}} - \langle f(\mathbf{x}_i), \tilde{\mathbf{y}}_i - \tilde{\mathbf{y}} \rangle - \xi_i \right)$$

Taking gradients with respect to the primal variables $f, \xi_i$ and setting it to 0 gives

$$f^* = \sum_{i=1}^n \sum_{\mathbf{y} \in \mathcal{Y}} \alpha^i_{\mathbf{y}} \mathcal{K}_{\mathbf{x}_i} (\tilde{\mathbf{y}}_i - \tilde{\mathbf{y}}) = \sum_{i=1}^n \mathcal{K}_{\mathbf{x}_i} \mathbf{c}_i$$

where $\mathbf{c}_i = \sum_{\mathbf{y} \in \mathcal{Y}} \alpha^i_{\mathbf{y}} (\tilde{\mathbf{y}}_i - \tilde{\mathbf{y}})$. Thus the minimizer of (8.5) can also be expressed in the span of the kernel function evaluated at the data points. Replacing this expression in the Lagrangian, we can derive the dual optimization problem analogous to (6.5).

$$\max_{\boldsymbol{\alpha}} D(\boldsymbol{\alpha}) = \sum_i \sum_{\mathbf{y} \in \mathcal{Y}} \alpha^i_{\mathbf{y}} \ell^i_{\mathbf{y}} - \frac{1}{2} \sum_j \left\| \mathcal{K}_{\mathbf{x}_j} \mathbf{c}_j \right\|^2_{\mathcal{H}_\mathcal{K}} \tag{8.7}$$

Note that the dual is a smooth function in $f$ and thus standard smooth optimization procedures can be adopted to optimize $D(\boldsymbol{\alpha})$ provided that the exponential sums to evaluate $\mathbf{c}_i$ can be calculated efficiently using procedures akin to belief propagation.

# Chapter 9

# Future Work

A lot of the work on vector valued functions for structured learning is still work in progress. As a result, there are a number of topics in which there is scope of new developments and algorithms.

## 9.1 Semi-Supervised Structured Prediction

Semi-supervised learning refers to learning from labeled as well as unlabeled examples. Given a set of labeled data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{l}$ and unlabeled data $\{\mathbf{x}_{l+j}\}_{j=1}^{u}$, we want to develop a hypothesis function that predicts well on new (and the unlabeled) input data. With the increasing abundance of raw data in almost all applications over the last few years and the high costs, difficulty and error involved in manually labeling it, the need for developing good learning algorithms which exploit the geometry of the unlabeled examples is constantly on the rise. There has been a significant amount of work in semi-supervised learning for unstructured data in the last few years (See [BNS06] and references therein) that make use of graph based methods to learn a good classifier by making use of geometric information from the data.

Unfortunately there has been very limited work on semi-supervised learning for structured data.

[AMB06] try to adapt manifold regularization approaches by adding an *intrinsic* regularizer- which depends on the geometry of the data- to the objective function. This is based on the assumption that the structured data lie on a low dimensional manifold and labels of two *similar* inputs are close to each other based on some notion of distance. While both the assumptions are standard in the realm of semi-supervised learning and make sense for structured data, [AMB06] try to build a graph on cliques of the input instead of the inputs themselves. Also since standard approaches in structured learning depend heavily on joint kernels [BHS+07], the authors are forced to pad up the unlabeled data with all possible labels in the label space to generate labeled examples. As a result, for any given input $\hat{\mathbf{x}}$, the correct label $\hat{\mathbf{y}}$ gets padded with the same weight as a label which does not match the correct label at any of the positions in the input. Other approaches [Bre08] try to adapt standard semi-supervised algorithms like co-learning into structured domains and furthers it analogously to co-regression and co-SVMs. Recently there have been some work involving latent variables where the labels corresponding to the unlabeled data are treated as latent variables and are marginalized over [YJ09]. The lack of approaches towards this problem is startling, since structured data is in general more complicated to annotate manually whereas input data is abundant and easy to obtain, which is a further incentive for using semi-supervised learning.

We propose a geometry based approach on the lines of [BN04] using vector valued functions. Given a set of input data $\{\mathbf{x}_i\}_{i=1}^n$, we generate a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ where the vertices of $\mathbf{V}$ are the input data and the edges $\mathbf{W}_{i,j} \in \mathbf{E}$ represent weights defined on the edge joining $\mathbf{x}_i$ and $\mathbf{x}_j$. We define a notion of distance on the input space. In general, vertices which are closer are considered to be similar and the corresponding edges are assigned more weights as opposed to edges between vertices which are very different from each other. Note that this graph is different from the underlying graphical model governing the structure of the data defined in chapter 4.

For this purpose, we can define a new kernel $K_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which defines an RKHS $\mathcal{H}_{\mathcal{X}}$ of functions on the input space $\mathcal{X}$ such that every $\mathbf{x}_i \mapsto K_{\mathbf{x}_i} = \tilde{\mathbf{x}}_i \in \mathcal{H}_{\mathcal{X}}$. The weight vectors on edge $(i, j)$ can then be given by $\mathbf{W}_{i,j} = \exp(-\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|^2_{\mathcal{H}_{\mathcal{X}}}/4t)$ for some parameter $t$. We define a graph Laplacian operator $L$ on functions in the space $\mathcal{H}_{\mathcal{K}} = \{f : \mathcal{X} \to \mathcal{H}_{\mathcal{Y}}\}$. The Laplacian operator can be defined in the standard way as for scalar functions

$$Lf(\mathbf{x}_i) = \mathbf{D}_{i,i} f(\mathbf{x}_i) - \sum_j \mathbf{W}_{i,j} f(\mathbf{x}_j)$$

where $\mathbf{D}_{i,i} = \sum_j \mathbf{W}_{i,j}$. It can be shown easily [BNS06] that $L$ is a self-adjoint, positive semidefinite operator. Thus its eigenfunctions form an orthonormal basis of the space $\mathcal{H}_{\mathcal{K}}$. After solving an eigenvalue-eigenfunction system for $L$, we can choose the $k$ most significant eigenfunctions (corresponding to the largest eigenvalues) to represent a function $f$ in $\mathcal{H}_{\mathcal{K}}$. Given the labeled data, we can solve an ERM problem to get the minimizing function $f$ which conforms to the geometry of the unlabeled data and also minimizes the error on the labeled data. This gives us a formulation for semi-supervised learning.

The standard formulation for the laplacian $L$ requires the entries $\mathbf{D}_{i,i}$ and the weights $\mathbf{W}_{i,j}$ to be scalars [BNS06]. However we can model each $\mathbf{D}_{i,i}$ and $\mathbf{W}_{i,j}$ as operators in themselves so as to give a greater degree of freedom to the Laplacian operator and allow larger number of distinct eigenvalues and eigenspaces.

## 9.2 Optimization

Although the formulations for ERM approaches using vector-valued functions are well defined, optimizing each of the objective functions efficiently is significantly non-trivial. Most of the standard algorithms in structured prediction rely on efficient belief propagation techniques to evaluate the marginals over the cliques in the graphical model. For solving (8.7), we can use similar approaches to sum over the exponential number of coefficients $\left\{\alpha_{\mathbf{y}}^{i}\right\}$. This is straightforward when the functions $\mathbf{c}_i$ are finite fixed dimensional vectors. However for arbitrary functions, we can decompose $\mathbf{c}_i$ and the kernel operator $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ over a system of basis functions in $\mathcal{H}_{\mathcal{Y}}$ and optimize over the coefficients of the basis vectors.

Solving the pre-image problem (7.3) is a big step which leads to the final prediction. Due to the exponential size of the search space $\mathcal{Y}$, we need to use a dynamic programming procedure analogous to the viterbi algorithm- which is straightforward if the functions are considered to be finite dimensional vectors. For arbitrary functions, a representation in terms of the basis functions of $\mathcal{H}_{\mathcal{Y}}$ is required to be able to perform inference efficiently.

# Chapter 10

# Conclusions

We have presented a new approach to structured object prediction that uses the concept of vector valued functions. Our approach captures richness of structure by allowing multiple kernels to be defined on the $\mathcal{Y}$, $\mathcal{X}$ as well as the operator valued kernel $\mathcal{K}$, each of which gives us the freedom to work with an RKHS over which training and inference algorithms can be performed efficiently.

We present the formulations for two different loss functions and set up the corresponding ERM problems and characterize the minimizing function using a variant of the Representer theorem for the vector valued function setting. We also give possible examples of operator valued kernels as well as kernels on $\mathcal{X}$ and $\mathcal{Y}$ that can be defined to capture the structure.

Our contribution is significant in the fact that we are not constrained to use joint kernels as in the standard literature- thus ensuring a seamless transition to semi-supervised learning. This is of fundamental importance, as structured data is becoming more and more complex with time, thus leading in difficulties in annotating it. On the other hand, getting access to large amounts of unlabeled data is gradually becoming easier thus naturally implying the learning of a classifier from both labeled and unlabeled data.

It is possible to use different approaches to perform optimization faster for training the learning algorithm on given labeled data. The exponentiated gradient is the state of the art method achieving the current best rates at training both CRFs and M$^3$Ns. However newer algorithms that exploit the excessive gap scheme [Nes05] have provably better rates of convergence ($O(1/\sqrt{\epsilon})$) for training M$^3$Ns. It would be interesting to note whether these new methods can be adapted to training vector valued SVMs.

Finally we are currently trying to run this model on certain real life applications to see how they perform compared to other approaches. It would be interesting to observe what trade-off can be bought using the power of different kernels over the standard usage of joint kernels on $\mathcal{X} \times \mathcal{Y}$ space.

# Bibliography

[AdFDJ03] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

[AG93] N.I. Akheizer and I.M. Glazman. *Theory of linear operators in Hilbert Spaces*, volume 1. Dover Reprint, 1993.

[AMB06] Yasemin Altun, David McAllester, and Mikhail Belkin. Maximum margin semi-supervised learning for structured variables. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 33–40. MIT Press, Cambridge, MA, 2006.

[Aro50] N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

[BGV04] Nicole Berline, E. Getzler, and Michle Vergne. *Heat Kernels and Dirac Operators.* Springer-Verlag, 2004.

[BHS$^+$07] G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data.* MIT Press, Cambridge, Massachusetts, 2007.

[Bis06] Christopher Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[BN04] Mikhail Belkin and Partha Niyogi. Semi-supervised learning on riemannian manifolds. *Mach. Learn.*, 56(1-3):209–239, 2004.

[BNS06] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, November 2006.

[BPP96] A. Berger, S. A. Della Pietra, and V.J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[Bre08] Ulf Brefeld. *Semi-supervised Structured Prediction Models.* PhD thesis, Humboldt-Universitt zu Berlin, 2008.

[CGK$^+$08] Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *JMLR*, 9:1775–1822, 2008.

[CMW05] Corinna Cortes, Mehryar Mohri, and Jason Weston. A general regression technique for learning transductions. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 153–160, New York, NY, USA, 2005. ACM.

[CVT06] C. Carmeli, E. De Vito, and A. Toigo. Vector valued Reproducing Kernel Hilbert spaces of integrable functions and Mercer theorem. *Analysis and Applications*, 4:377–408, 2006.

[CW87] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6, New York, NY, USA, 1987. ACM.

[DGL96] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Applications of mathematics*. Springer, New York, 1996.

[DPDPL97] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393, 1997.

[HC71] J. M. Hammersley and P. E. Clifford. Markov fields on finite graphs and lattices. unpublished manuscript, 1971.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, New York, 2 edition, 2009.

[Jor08] M. I. Jordan. *An Introduction to Probabilistic Graphical Models*. MIT Press, 2008. To Appear.

[Kel60] J. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial Applied Mathematics*, 8:703–712, 1960.

[KFL01] Frank Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[KW71] G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, 33:82–95, 1971.

[Lau96] S. L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, UK, 1996.

[LMP01] J. D. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *Proceedings of International*

*Conference on Machine Learning*, volume 18, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

[LSST$^+$02] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, February 2002.

[MP05] Charles A. Micchelli and Massimiliano A. Pontil. On learning vector-valued functions. *Neural Comput.*, 17(1):177–204, 2005.

[Nes05] Yurii Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM J. on Optimization*, 16(1):235–249, 2005.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, 1988.

[Pla99] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

[Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, February 1989.

[Rat96] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. 1996.

[Rat99] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–175, 1999.

[Roc70] R. T. Rockafellar. *Convex Analysis*, volume 28 of *Princeton Mathematics Series*. Princeton University Press, Princeton, NJ, 1970.

[SC04] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.

[Set05] Burr Settles. ABNER: an open source tool for automatically tagging genes, protiens, and other entity names in text. *Bioinformatics*, 21(14):3191–3192, April 2005.

[SM06] C. Sutton and A. McCallum. *An Introduction to Conditional Random Fields for Relational Learning*. MIT Press, 2006.

[SP03] Fei Sha and Fernando C. N. Pereira. Shallow parsing with conditional random fields. In *HLT-NAACL*, 2003.

[SS02] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[Tas04] Ben Taskar. *Learning Structured Prediction Models: A Large Margin Approach*. PhD thesis, Stanford University, 2004.

[TGK04] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32, Cambridge, MA, 2004. MIT Press.

[TJHA05] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005.

[TKC$^+$04] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Empirical Methods in Natural Language Processing*, pages 1–8, Barcelona, Spain, 2004. Association for Computational Linguistics.

[TLJJ06] Ben Taskar, Simon Lacoste-Julien, and Michael Jordan. Structured prediction, dual extragradient and bregman projections. *Journal of Machine Learning Research*, 7:1627–1653, 2006.

[Vap98] V. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, New York, 1998.

[VBSK08] S. V. N. Vishwanathan, Karsten Borgwardt, Nicol N. Schraudolph, and Imre Risi Kondor. On graph kernels. *J. Mach. Learn. Res.*, 2008. submitted.

[WCE$^+$03] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 873–880. MIT Press, Cambridge, MA, 2003.

[WJ08] M. J. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*, volume 1 of *Foundations and Trends in Machine Learning.* 2008.

[YJ09] Chun-Nam John Yu and T. Joachims. Learning structural svms with latent variables. In *International Conference on Machine Learning (ICML)*, 2009.