

# Combining SVM with graphical models for supervised classification: an introduction to Max-Margin Markov Networks

Simon Lacoste-Julien<sup>†</sup>

<sup>†</sup>Department of EECS  
University of California, Berkeley  
Berkeley, CA 94720-1770  
*slacoste@eecs.berkeley.edu*

December 1, 2003

## Abstract

The goal of this paper is to present a survey of the concepts needed to understand the novel *Max-Margin Markov Networks* (M<sup>3</sup>-net) framework, a new formalism invented by Taskar, Guestrin and Koller [TGK03] which combines both the advantages of the graphical models and the Support Vector Machines (SVMs) to solve the problem of multi-label multi-class supervised classification. We will compare generative models, discriminative graphical models and SVMs for this task, introducing the basic concepts at the same time, leading at the end to a presentation of the M<sup>3</sup>-net paper.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Setting for multi-label multi-class classification</b>	<b>4</b>
2.1	Example: Part-of-speech tagging . . . . .	4
2.2	Loss function and risk . . . . .	5
2.3	Computational Learning Theory . . . . .	6
<b>3</b>	<b>Generative method</b>	<b>6</b>
3.1	HMM example . . . . .	7
3.2	Analysis . . . . .	8
<b>4</b>	<b>Discriminative Graphical Model</b>	<b>8</b>
4.1	Features in Graphical Models . . . . .	8
4.2	Conditional Random Fields (CRF) . . . . .	9
4.3	Analysis . . . . .	11
<b>5</b>	<b>Introduction to Support Vector Machines (SVM)</b>	<b>11</b>
5.1	Max-margin linear classifier . . . . .	11
5.2	Lagrangian duality . . . . .	13
5.3	SVM dual and the Kernel trick . . . . .	15
<b>6</b>	<b>Multi-class SVM</b>	<b>17</b>
6.1	Analysis . . . . .	19
<b>7</b>	<b>Max-Margin Markov Networks (<math>M^3</math> net)</b>	<b>19</b>
7.1	Margin-based Structured Classification . . . . .	19
7.2	Key insight to obtain the factored dual . . . . .	20
7.3	General graphs and other issues . . . . .	22
7.4	Analysis and other related papers . . . . .	22
<b>8</b>	<b>Conclusion</b>	<b>23</b>

## 1 Introduction

In supervised classification learning, the goal is to learn a function which assigns (discrete) labels to arbitrary objects, given a set of already assigned independent instances. This framework has a vast number of applications ranging from part-of-speech tagging to optical character recognition. In the past years, the state-of-the-art algorithm for this task has been the *Support Vector Machines* (SVMs) [CV95, Vap99], an algorithm based on the maximization of the margin of confidence of the classifier (where the margin is related to the minimal distance between the points and the classifier in some Euclidean space). Their success was due to their ability to use very high dimensional feature spaces (using the kernel trick) while

at the same time keeping strong generalization guarantees (from their max-margin property). One caveat though was that SVMs can only assign one label at a time and its running time is polynomial in the number of classes. This means that to jointly classify different objects, one either has to use a joint label with an exponential number of classes (exponential in the number of objects and thus intractable for SVM in general), or assume that the classification of each object is made independently. The latter option was typically used, though it meant that some precious information was lost in the case of structured data. For example, in the case of optical character recognition for words (in English say), each character was classified independently<sup>1</sup>. But by knowing the first and third character of the word (for example), one could distinguish more easily the second character (if the classifier was hesitating between ‘aaa’ and ‘aba’, then it could choose ‘aba’ confidently because of the non-existence of the pattern ‘aaa’ in English).

On the other hand, probabilistic graphical models [Jor98] provide a principled and efficient way to deal with structured data by taking advantage of the (potential) sparsity of the interactions in a factored way. But they didn’t have yet the same generalization guarantees as SVMs, neither could they support very high dimensional feature spaces. So one natural question to ask is whether or not there is a way to unify both approaches and get the best of both worlds. That is, is it possible to have a SVM-like approach which could deal also with structured data efficiently like graphical models do? The answer is yes, and it was shown recently by Taskar, Guestrin and Koller in a paper which is going to appear at NIPS 2003 [Tgk03]. The new framework that they propose, *Max-Margin Markov Networks* (M<sup>3</sup>-net), is a major breakthrough in the field of machine learning as it will open a whole new set of problems on which people could apply SVMs efficiently.

In this paper, we will give an overview of the concepts needed to understand the advantages of the M<sup>3</sup>-nets and how they are built. The background assumed is one provided by an introductory graduate course on graphical models taught at UC Berkeley<sup>2</sup>. The review won’t be exhaustive, neither comprehensive, but we hope that it can give the reader a somewhat self-contained overview of the different ways to do multi-label multi-class classification. The emphasis will be on the *theory* behind the M<sup>3</sup>-nets. We will provide concrete examples to help the understanding of how to apply the different ideas, but they are not claimed to be an efficient way to solve the given problem.

The paper is organized as follows. In section 2, we describe the setting of the problem that we will be interested in, namely, multi-label multi-class classification. We present a concrete example which will be used during the whole paper, and we describe a bit of the computational learning theory to give some more precision about the goal in supervised classification. In the next sections, we compare different approaches for multi-label classification. We present the generative approach in section 3 using a HMM example. In section 4, we illustrate the discriminative graphical model approach using a conditional random field. In section 5, we give a somewhat more detailed exposition of binary SVMs, by presenting the notion of margin, duality, kernel, and generalization bound. Section 6 describes how

---

<sup>1</sup>To be fair, we have to mention that the first benchmarks were done on digit recognition [SBV99] on US postal codes, in which the independence assumption is more realistic.

<sup>2</sup>CS281a: see <http://www.stat.berkeley.edu/%7Ebartlett/courses/241A-fall2003/index.html>.

to extend SVMs to the multi-class problem. Finally, we present the M<sup>3</sup>-net framework in section 7, which reunites all the ideas of the preceding sections.

## 2 Setting for multi-label multi-class classification

We are interested in multi-label multi-class classification, that is, to a classification problem where each label can take a finite number of values (multi-class), and where each (structured) object can take a finite number of labels (multi-label). Thus the goal is to learn a function (hypothesis)  $h : \mathcal{X} \rightarrow \mathcal{Y}$ , where  $\mathcal{X}$  is the set of (structured) objects (also called patterns) that we want to classify with  $l$  labels, and where  $\mathcal{Y} = \mathcal{Y}_1 \times \dots \times \mathcal{Y}_l$  is the set of (structured) labels (which is structured here as a vector of labels without loss of generality). In this paper, we will assume for simplicity that each  $\mathcal{Y}_i$  has the same form, namely,  $\mathcal{Y}_i = \{y_1, \dots, y_k\}$ , for some fixed  $k$ . The training of the classifier is made with a sample set  $\mathcal{S}$  of  $n$  iid labelled instances:

$$\mathcal{S} = \left\{ \left( \mathbf{x}^{(i)}, \mathbf{y}^{(i)} \triangleq \mathbf{t}(\mathbf{x}^{(i)}) \right) \right\}_{i=1}^n. \quad (1)$$

where  $\mathbf{x} \in \mathcal{X}$  and  $\mathbf{y} = (y_1, \dots, y_k) \in \mathcal{Y}$ , and where  $\mathcal{S}$  is assumed to be drawn from a fixed distribution  $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$ . Notice the use of the superscript to index the different training instances, and the subscript to index the *components* of the joint label. Also, we use the notation  $\mathbf{t}(\mathbf{x}^{(i)})$  for the *target* value of  $\mathbf{x}^{(i)}$  so that we can distinguish it with other (arbitrary) values for  $\mathbf{y}$  later. This is not necessary when doing binary classification, but will be a real notation convenience when dealing with M<sup>3</sup>-nets...

### 2.1 Example: Part-of-speech tagging

As a concrete example which will be used in the sequel, we look at the problem of labelling a FAQ database as was presented in [MFP00]. An example of a labelled FAQ instance was the following:

```
<head>X-NNTP-Poster: NewsHound v1.33
<head>
<head>
<head>Archive-name: acorn/faq/part2
<head>Frequency: monthly
<head>
<question>2.6) What configuration of serial cable should I use
<answer>
<answer> Here follows a diagram of the necessary connections
<answer>programs to work properly. They are as far as I know t
```

Here, each  $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_l)$  is a sequence of sentences; and each sentence  $\mathbf{x}_i$  is labelled by  $y_i \in \mathcal{Y} = \{\text{head, question, answer, tail}\}$ . In this case, the data is inherently sequential and it is clear that the label of a sentence doesn't only depend on its words but also on what

happened in the previous sentences. For example, the empty line is labelled sometimes as ‘head’ and other times as ‘question’ in this excerpt.

Other examples of structured data classification are:

**collective webpage classification** where interlinked webpages are to be labelled collectively [TAK02];

**named entity recognition** where the goal is to find and identify named entities (like named organizations, persons, organizations, as well as temporal and monetary expressions) in a segmented text (and so there is a special ‘none’ label when a word is not a named entity) [BSW99];

**handwritten word recognition** where characters are individually labelled inside a word [TGM03];

Thomas Dietterich gives a review of sequential data classification in machine learning in [Die02].

## 2.2 Loss function and risk

We need some way to evaluate the performance of the classifier. A useful tool is the *loss function*<sup>3</sup>  $L : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \rightarrow [0, \infty[$ , where  $L(\mathbf{x}, \mathbf{y}, h(\mathbf{x}))$  gives the cost of assigning the label  $h(\mathbf{x})$  to the observation  $\mathbf{x}$  given that the correct label is  $\mathbf{y}$  [SS02, ch 3]. It is usually assumed that  $L(\mathbf{x}, \mathbf{y}, \mathbf{y}) = 0$  so that the minimal value is 0 and attained by the correct labelling. In a decision theoretic framework, the loss function is a negative utility function. A rational goal then would be to minimize the total loss on the labels to be predicted. In the case where one doesn’t know which objects it will have to classify in the future, it makes sense to minimize the *expected loss* on future data, called the *expected risk*  $R[h]$  which is a functional of the classifier  $h$ :

$$R[h] \triangleq \int_{\mathcal{X} \times \mathcal{Y}} L(\mathbf{x}, \mathbf{y}, h(\mathbf{x})) d\mathcal{D}(\mathbf{x}, \mathbf{y}) \quad (2)$$

Since we don’t know  $\mathcal{D}$ , it is not possible in general to compute the expected risk. One has to turn to approximations. A useful one, motivated by the Glivenko-Cantelli theorem<sup>4</sup>, is to compute the *empirical risk*  $R_{\text{emp}}[h]$ , evaluated on the sample  $\mathcal{S}$  of size  $n$ :

$$R_{\text{emp}}[h] \triangleq \frac{1}{n} \sum_{i=1}^n L(\mathbf{x}^{(i)}, \mathbf{t}(\mathbf{x}^{(i)}), h(\mathbf{x}^{(i)})) \quad (3)$$

Vapnik discusses in [Vap99] how we can generalize the law of large numbers in function space (with very strict conditions) to tell us when working with  $R_{\text{emp}}[h]$  will be a good approximation to  $R[h]$ .

---

<sup>3</sup>Also called *cost function* [Die02].

<sup>4</sup> which says that the sequence of empirical distribution functions for iid samples of size  $n$  converge uniformly to the true distribution function as the sample size  $n$  goes to infinity (see [Dur96]).

## 2.3 Computational Learning Theory

Now that the goal is explicit, we can discuss important issues about how to choose a classifier. The structure of the function space from  $\mathcal{X}$  to  $\mathcal{Y}$  can be quite unwieldy, so one usually restricts  $h$  amongst a parameterized family  $\mathcal{F} = \{h_{\mathbf{w}} : \mathbf{w} \in \mathcal{H}\}$ , where  $\mathcal{H}$  is some parameter space (usually a Hilbert space). The choice of  $\mathcal{F}$  is very important: if it is too much expressive, then there is a danger for the learner to *overfit* the training data (rote learn the data); if it is not expressive enough, then it won't be able to approximate the real function. This qualitative statement is made explicit with the following Probably Approximately Correct (PAC) bound; for any sample  $\mathcal{S}$  of size  $n$  and threshold  $\delta > 0$ , and for  $h \in \mathcal{A} \subseteq \mathcal{H}$ , the following bound holds with probability at least  $1 - \delta$ :

$$R[h] \leq R_{\text{emp}}[h] + \Omega(\mathcal{A}, n, \delta^{-1}) \quad (4)$$

where  $\Omega(\mathcal{A}, n, \delta^{-1})$  is a measure of the *capacity* of the parameter class  $\mathcal{A}$ , which is related to its expressiveness (we'll present an example of expression for  $\Omega$  in the case of Support Vector Machines in equation (17)). The more expressive the parameter class, the bigger the bound, and so this term characterizes the overfitting possibility<sup>5</sup>. The empirical risk, on the other hand, represents the training error, and so if the parameter class is not expressive enough, it won't be possible to make it small. The compromise between expressiveness and small training error is made by finding a parameter class  $\mathcal{A}$  with small capacity but which contains the parameter for a classifier  $h_{\mathbf{w}}$  which yields zero (or small) training error  $R_{\text{emp}}[\mathbf{w}]$ . The process of minimizing the RHS of equation (4) by iterating through different parameter classes  $\mathcal{A}_i$  of increasing capacity and finding the minimal training error achievable in each class at each step is called *structural risk minimization*, and was proposed by Vapnik [Vap99]. The bound is not necessarily tight, and thus it doesn't guarantee that a classifier with higher bound will necessarily have worse generalization performance than another classifier with lower bound. But it does provide of a more principled heuristic to choose classifiers. And it also gives a theoretical bound on the generalization error for a given classifier *which doesn't depend on specific assumptions about the (fixed) distribution  $\mathcal{D}$*  (the influence of  $\mathcal{D}$  is summarized in the empirical risk, which can be computed) and thus is quite robust about the lack of knowledge of the type of the distribution  $\mathcal{D}$ . We will see in section 5 how this has motivated a powerful learning algorithm with nice generalization guarantees, the support vector machine.

But before presenting it, we will have a quick look at different graphical model approaches to do multi-label multi-class classification so that we can compare later their advantages and drawbacks with the SVMs. We will start with the generative graphical model approach.

## 3 Generative method

In the generative approach, one builds a parameterized graphical model for the *joint* distribution  $P_{\mathbf{w}}(\mathbf{x}, \mathbf{y})$  using some conditional independence assumptions about the problem at

---

<sup>5</sup>And trying to minimize the model class complexity is a formal version of Occam's razor principle.

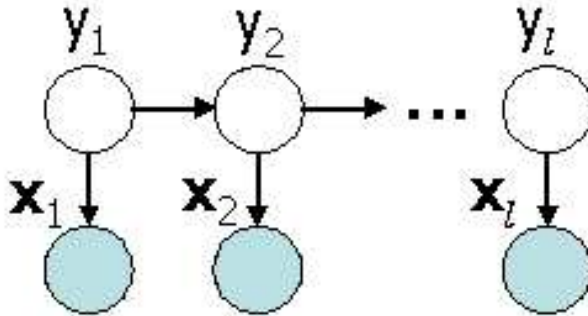


Figure 1: **(naive) HMM graphical model for the FAQ problem.** The definition of  $\mathbf{x}_i$  and  $y_i$  is given in section 2.1. The labels are hidden variables in the context of *classification*. For the training part, all the nodes are observed (and we will have  $n$  iid copies of this graph).

hand. Using the training data, the hope is that if the modelling assumptions are good, one will be able to find a parameter  $\mathbf{w}$  which yields a good approximation to the real distribution  $\mathcal{D}$ . Then to do classification, one can choose the most likely label by computing  $h_{\mathbf{w}}(\mathbf{x}) = \arg \max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}, \mathbf{w})$ , which can be efficiently computed using the Viterbi algorithm [Jor03] on the graphical model (even if there is an exponential number of labels - that is one of the great advantages of graphical models over SVMs).

### 3.1 HMM example

To illustrate the idea, we will come back to our FAQ database tagging example (see section 2.1) and use a Hidden Markov Model (HMM). We will restrict our attention in this paper to sequential data to simplify the notation, but we will discuss in section 7.3 how the ideas can also be applied to arbitrary graphs. HMMs have been widely used in part-of-speech tagging, amongst others, and a classical tutorial is given by Rabiner in [Rab89]. Figure 1 shows the HMM for the FAQ database problem. We can already see that we have made the incorrect assumption that the line content was conditionally independent of the other labels given its line label. This is wrong here since for example the blank line has a much higher probability of happening for a line labelled ‘answer’ which was preceded by a ‘question’ than for a line which was preceded by another ‘answer’. But we remind the reader that the point here is not to construct an accurate model but rather to illustrate the different concepts.

With this model, the joint is thus:

$$P(\mathbf{x}, \mathbf{y}) = P(y_1) \prod_{i=1}^{l-1} P(y_{i+1}|y_i) \prod_{i=1}^l P(\mathbf{x}_i|y_i) \quad (5)$$

where the implicit dependence on  $\mathbf{w}$  was not shown in the notation (and we will keep this convention for the remaining of the paper; each time there is a probability function or a potential, it is assumed that it depends implicitly on a parameter  $\mathbf{w}$ ). The parameterization

of  $P(y_{i+1}|y_i)$  is usually a matrix; and  $P(\mathbf{x}_i|y_i)$ <sup>6</sup> would normally (in the part-of-speech tagging community) be a multinomial over a finite vocabulary, though here it would be intractable since the number of possible sentences is pretty large... This indicates another disadvantage of using the generative approach (we’ll see in section 4.1 how a ‘feature’ approach can solve partly this problem). The training can be made (efficiently) using a maximum likelihood (ML) estimate which decomposes nicely here since all the nodes are observed (we don’t need to use EM). Once the model has been fitted on the training data, one can do classification by computing

$$h_{\mathbf{w}}(\mathbf{x}) = \arg \max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}, \mathbf{w}) \quad (6)$$

using the fact that  $P(\mathbf{y}|\mathbf{x}) \propto P(\mathbf{x}, \mathbf{y})$  for fixed  $\mathbf{x}$ .

## 3.2 Analysis

As mentioned in [Jor03, ch 7], the generative approach can be efficient when we have accurate prior knowledge of the true data-generating process  $\mathcal{D}$ . On the other hand, it is *suboptimal* in some sense since it tries to estimate accurately the joint  $P(\mathbf{x}, \mathbf{y})$  when all we really need for classification is  $P(\mathbf{y}|\mathbf{x})$ , or rather its maximum argument. The discriminative approach, in comparison, models directly  $P(\mathbf{y}|\mathbf{x})$  without any assumptions of what are the interactions inside  $\mathbf{x}$ , which are not needed in classification since  $\mathbf{x}$  is given. This gives a more *robust* approach since it uses weaker assumptions about the distribution. It is also well suited for a *feature-based* approach, which we will present in the next section.

# 4 Discriminative Graphical Model

## 4.1 Features in Graphical Models

The usual definition of undirected graphical models uses arbitrary potentials on maximal cliques. But as mentioned in [Jor03, ch 19], “large, fully-parameterized cliques are problematic both for computational reasons (inference is exponential in the clique sizes) and for statistical reasons (the estimation of large numbers of parameters requires large amounts of data)”. In practice, one will thus want to use a reduced parameterization for the clique potentials. One way to do that is to use a *feature* approach with models in the exponential family. For a given clique  $C$  (using the notation from [Jor03]), we define a set of (fixed) features  $f_i(x_{C_i})$  where  $C_i \subseteq C$  is a subset of the variables that the feature  $f_i$  references and where  $f_i$  is an arbitrary real-valued function (though it is usually an indicator function). The parameterized clique potential with parameter  $\mathbf{w}$  is then written:

$$\psi_C(x_C) \triangleq \exp \left( \sum_{j \in \mathcal{I}_C} w_j f_j(x_{C_j}) \right) \quad (7)$$

where  $\mathcal{I}_C$  is the index set for the features for the clique  $C$ . We thus see that the features act as the sufficient statistics in the exponential family representation (see [Jor03, ch 8]).

---

<sup>6</sup> called the *class conditional* (see [Jor03, ch 5,7]).

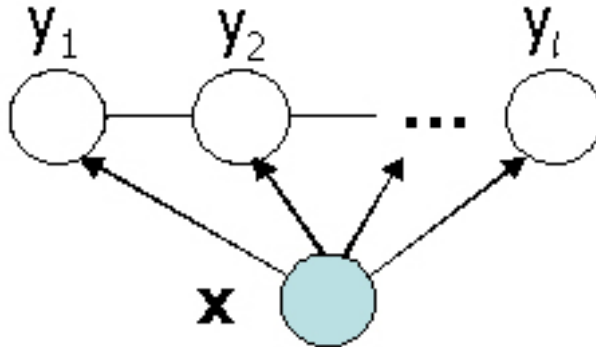


Figure 2: **Conditional Random field for the FAQ problem.** No assumption about the structure of  $\mathbf{x}$  is made (unlike in the generative framework), so that’s why  $\mathbf{x}$  is linked with all the label nodes. Compare with figure 1.

For a multinomial  $x$ , this parameterization is not restrictive since we could use one indicator feature (and thus one parameter) for each possible assignment to  $x$  and obtain any potential. But in general, we can use very sparse representations and this is advantageous both for modelling reasons (one can build a potential by using a ‘divide and conquer’ approach to identify individual (possibly overlapping) contributions and define a feature for each) and also for computational reasons (sparser potentials).

## 4.2 Conditional Random Fields (CRF)

We now present *conditional random fields* (CRF), the state of the art amongst graphical model discriminative methods for classification [LMP01]. CRFs model directly  $P(\mathbf{y}|\mathbf{x})$  by using feature-based parameterized potentials in an undirected graphical model. In the case of sequential data where the graph is a chain, the potentials are:

$$\psi(y_i, y_{i+1}, \mathbf{x}) \triangleq \exp \left( \sum_{j=1}^{d_1} \lambda_j f_j(y_i, y_{i+1}, \mathbf{x}) \right)$$

$$\psi(y_i, \mathbf{x}) \triangleq \exp \left( \sum_{j=1}^{d_2} \mu_j g_j(y_i, \mathbf{x}) \right)$$

with features  $f_j$  and  $g_j$  and parameters  $\lambda_j$  and  $\mu_j$  (and where we use the usual convention where the identity of the function can be found by the type of its arguments i.e.  $\psi(y_1, \mathbf{x}) \neq \psi(y_2, \mathbf{x})$  in general even if the *value* of  $y_2$  is the same as  $y_1$ ).

The CRF graphical model for the FAQ example is shown in figure 2. Here, no assumption about the structure of  $\mathbf{x}$  was made. For example,  $g_3(y_2 = \text{‘question’}, x)$  could be the indicator that “last line (1) was blank and this line (2) starts with a number”. Usually, the features will be local in the chain though they don’t have to. One can start to see that this feature

framework is well-suited to model the FAQ tagging problem, and that’s why in fact we chose this example<sup>7</sup>.

With this graphical model, the conditional  $P(\mathbf{y}|\mathbf{x})$  is then:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{i=1}^{l-1} \psi(y_i, y_{i+1}, \mathbf{x}) \prod_{i=1}^l \psi(y_i, \mathbf{x}) \quad (8)$$

where  $Z(\mathbf{x})$  is a normalization constant which depends on  $\mathbf{x}$ . It is now worthwhile to study the form of this conditional more in details. It was assumed in the example in [LMP01] that the potentials were homogeneous along the chain and similarly in the M<sup>3</sup>-net paper [TGK03], so we will use the same simplifying assumption. Now define the feature vector  $\mathbf{F}_i(\mathbf{x}, y_i, y_{i+1})$  as:

$$\mathbf{F}_i(\mathbf{x}, y_i, y_{i+1}) \triangleq (f_1(y_i, y_{i+1}, \mathbf{x}), \dots, f_{d_1}(y_i, y_{i+1}, \mathbf{x}), g_1(y_i, \mathbf{x}), \dots, g_{d_2}(y_i, \mathbf{x}))^T \in \mathbb{R}^{d_1+d_2} \quad (9)$$

and also (for notational convenience in the boundary case):

$$\mathbf{F}_l(\mathbf{x}, y_l, y_{l+1}) \triangleq (0, \dots, 0, g_1(y_l, \mathbf{x}), \dots, g_{d_2}(y_l, \mathbf{x}))^T \in \mathbb{R}^{d_1+d_2} \quad (10)$$

then we define the *joint feature* vector

$$\mathbf{F}(\mathbf{x}, \mathbf{y}) \triangleq \sum_{i=1}^l \mathbf{F}_i(\mathbf{x}, y_i, y_{i+1}) \quad (11)$$

With this notation, we can now easily see that the conditional is:

$$P(\mathbf{y}|\mathbf{x}) \propto \exp(\mathbf{w}^T \mathbf{F}(\mathbf{x}, \mathbf{y})) \quad (12)$$

with  $\mathbf{w} = (\lambda_1, \dots, \lambda_{d_1}, \mu_1, \dots, \mu_{d_2})^T$  and thus is a *softmax function*. It was mentioned in [Jor03, ch 7] that generative models with class-conditionals and marginals in the exponential family yielded the softmax function for  $P(\mathbf{y}|\mathbf{x})$  in general and thus we see that the form chosen for the CRF parameterization is very general.

The training of a CRF is made as for other completely observed graphical models with Maximum Likelihood. Since the graph is undirected, the ML optimization doesn’t separate in local ML problems like in the directed case because of the presence of the normalization term. The ML solution is found by using *Iterative Proportional Fitting* (IPF), which is exact in the case where the graph is triangulated (see [Jor03, ch 9]). For feature based graphical models, IPF is called *Iterative Scaling* and is described in [Jor03, ch 20]. An improved version is described for CRF in [LMP01]. Finally, the classification is made like in the generative case by computing (6).

---

<sup>7</sup>This example was taken from a paper on Maximum entropy Markov models (MEMMs) [MFP00], another form of discriminative graphical models which was the ancestor of CRFs.

### 4.3 Analysis

CRF and other discriminative methods solve the problem of having to model  $\mathbf{x}$  when it is not needed, and so yield a more robust and efficient algorithm for classification than the generative approach when one doesn't have good prior knowledge about  $\mathbf{x}$ .

On the other hand, it still can't handle high dimensional features because the formalism manipulates them directly (not like kernel-based methods which will be presented in section 5.3). Moreover, the graphical model formalism doesn't include the risk formulation that we have presented in section 2.2. There are ways to relate the ML likelihood with the empirical risk by defining the correct loss function (see [SS02, ch 3]), but ML doesn't include a regularization term and so we can't obtain bounds on the expected risk like equation 4 (or rather, we haven't heard of). It was mentioned in [AH03] that sometimes a regularization term is added to the log-likelihood for CRF in order to avoid overfitting the data. We didn't have the time to study more this approach, but it was claimed in [TJK03] that discriminative methods don't have yet the same generalization guarantees as SVMs. This makes sense since the CRF framework doesn't take into account the loss function explicitly, which is what we want to optimize during learning. On the other hand, support vector machines make direct use of the bound given in equation (4), yielding nice generalization guarantees. We will now present a brief introduction to its concepts.

## 5 Introduction to Support Vector Machines (SVM)

### 5.1 Max-margin linear classifier

The Support Vector Machine (SVM) is a learning algorithm for a linear classifier which tries to maximize the margin of confidence of the classification on the training data set. It was described in [CV95] by Cortes and Vapnik. The classical tutorial paper usually quoted is the one by Burges [Bur98], though it is now starting to get old. A gentle (and modern) introduction is given by Cristianini and Schölkopf in [CS02]; and also in chapter 1 of [SS02]. The detailed story can be found in [Vap99], or in [CST00] at a more introductory level. We will now try to outline quickly the main ideas behind SVMs.

We first consider the case where we want to classify  $\mathbf{x}$  (which is assumed to lie in  $\mathbb{R}^d$ ) amongst two classes in  $\mathcal{Y} = \{-1, 1\}$ . A linear classifier will be an hyperplane in  $\mathbb{R}^d$  characterized by a normal  $\mathbf{w}$  and an offset  $b$ . The binary classifier can be:

$$h_{\mathbf{w}}(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x} + b). \quad (13)$$

where  $\text{sgn}$  is the sign function which will take value 0 at 0, say, so that the classification is ambiguous at this point. If the data is linearly separable, then it is possible to find an hyperplane which will correctly satisfy all the points, that is, a  $(\mathbf{w}, b)$  such that:

$$\begin{aligned} \mathbf{w}^T \mathbf{x}^{(i)} + b > 0 & \quad \text{if} \quad t(\mathbf{x}^{(i)}) = 1 \\ \mathbf{w}^T \mathbf{x}^{(i)} + b < 0 & \quad \text{if} \quad t(\mathbf{x}^{(i)}) = -1 \end{aligned}$$

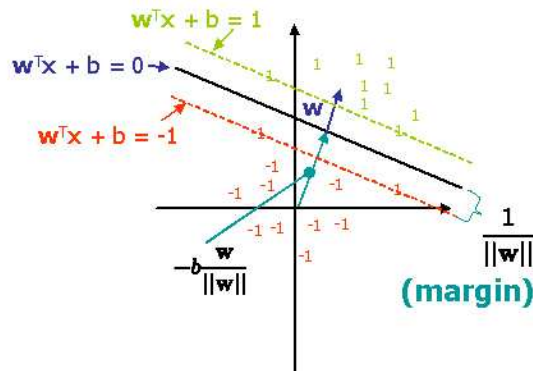


Figure 3: **SVM: linear binary classifier.** The separating hyperplane is in blue; the margin hyperplanes are dotted. The data points which lie on the margin are called *support vector*.

for all  $(\mathbf{x}^{(i)}, t(\mathbf{x}^{(i)}))$  in  $\mathcal{S}$ , which can be written more succinctly as

$$t(\mathbf{x}^{(i)}) \cdot (\mathbf{w}^T \mathbf{x}^{(i)} + b) > 0 \quad \forall i. \quad (14)$$

If this is the case, then we can rescale  $(\mathbf{w}, b)$  so that the closest points to the hyperplane (those which lie on the *margin*) satisfy  $|\mathbf{w}^T \mathbf{x}^{(i)} + b| = 1$ ; we then obtain a *canonical form* for  $(\mathbf{w}, b)$  which satisfies (tightly) the inequality:

$$t(\mathbf{x}^{(i)}) \cdot (\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1 \quad \forall i. \quad (15)$$

The geometry of this setting is shown in figure 3. It is not hard to see that given the above constraints, the margin (that is, the minimal distance between the data points and the hyperplane) has size  $1/\|\mathbf{w}\|$ . So if the goal is to maximize the margin, then we need to minimize  $\|\mathbf{w}\|$ . This can be cast in an equivalent constraint optimization problem:

$$\begin{aligned} & \text{minimize} && \frac{1}{2} \mathbf{w}^T \mathbf{w} && (16) \\ & (\mathbf{w}, b) \in \mathbb{R}^{d+1} && && \\ & \text{subject to} && t(\mathbf{x}^{(i)}) \cdot (\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1 && \forall i \end{aligned}$$

Before solving this problem, let's see why choosing a classifier by maximizing the margin is a rational thing to do. One informal reason is that if we expect the Euclidean distance to be representative of the dissimilarity between patterns (that is, similar patterns should be close), then the bigger the margin, the more confidence we have in our classification (there are no patterns of different classes which are close). Also, we expect the new patterns that we want to classify to stay close to the ones that we have already observed (since the training data should be representative of the real distribution), so the more room we have between the two different classes, the less chance we have to have ambiguity in future classification.

A more formal argument, though, is that by only choosing hyperplanes which have maximal margin, we restrict a lot the size of the class of classifiers that we are considering (it's capacity), and so we are really minimizing the RHS of equation (4), which gives an upper

bound on the expected risk (note that here the empirical risk is 0 since we assumed that the data was linearly separable). In fact, a measure of capacity which is defined in [Vap99] is the Vapnik Chervonenkis (VC) dimension for a class of classifiers  $\mathcal{H}$ . For the 0-1 loss (which is 1 when the label is incorrectly classified and 0 otherwise), the bound  $\Omega(\mathcal{A}, n, \delta^{-1})$  in equation (4) can be written:

$$\left( \frac{V \cdot (\log(2n/V) + 1) + \log(4/\delta)}{n} \right)^{1/2} \quad (17)$$

where  $V$  is the VC-dimension of the class of classifiers considered, and  $n$  is the size of the sample. This bound is called the *VC confidence* (see [Bur98]) and we are presenting it here just to give an idea of the kind of dependence on  $V$  and  $n$  that one can expect. Also, Vapnik derived in [Vap99] an expression for the VC dimension of the class of linear classifiers with specific margin which indicated that bigger margins yielded smaller error bound, thus giving considerable justification for maximizing the margin.

Finally, we have to deal with the case when the data is not linearly separable. Then some points  $\mathbf{x}^{(i)}$  can be at a distance  $\xi_i/\|\mathbf{w}\|$  on the wrong side of the margin hyperplane. We can thus transform the constraint (15) to:

$$t(\mathbf{x}^{(i)}) \cdot (\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i \quad (18)$$

If  $\mathbf{x}^{(i)}$  is misclassified, then the LHS of (18) is smaller than 0 and thus  $\xi_i$  has to be greater than 1. This means that  $\frac{C}{n} \sum_{i=1}^n \xi_i$  is an upper bound on the 0-C average loss on the training data (with a constant  $C > 0$ ), that is, of  $R_{\text{emp}}[h]$  for the 0-C loss (a cost of  $C$  for each error). Following our goal of structural risk minimization (minimizing the RHS of (4)), we want to minimize the sum of  $\frac{C}{n} \sum_{i=1}^n \xi_i$  (which is an upper bound on the empirical risk) and  $\|\mathbf{w}\|^2$ , which is related to  $\Omega(h)$ . The tradeoff between each contribution is encoded by the constant  $C$  that the user can choose, with  $C = \infty$  meaning that he can't tolerate any error (its cost is infinite). Hence, the learning algorithm will find the classifier  $(\mathbf{w}, b)$  which satisfies the following optimization problem on the training data:

$$\begin{aligned} & \text{minimize} && \frac{1}{2} \mathbf{w}^T \mathbf{w} + \frac{C}{n} \sum_i \xi_i && (19) \\ & (\mathbf{w}, b), \xi && && \\ & \text{subject to} && t(\mathbf{x}^{(i)}) \cdot (\mathbf{w}^T \mathbf{x}^{(i)} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i \end{aligned}$$

In order to solve this constrained optimization problem, we can use the method of Lagrange multipliers. As it is pretty crucial to the sequel, we now present a review of Lagrangian duality.

## 5.2 Lagrangian duality

The general goal of duality in constrained optimization is twofold: to help to solve the original problem; and to provide a *certificate* which guarantees that we have found the optimal solution. The second goal is more of an issue in linear programming, though. Boyd

gives a nice treatment of Lagrangian duality in [BV04, ch 5]; and more details can be found in [Ber99]. We will consider here the following (primal) constrained optimization problem:

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && f_i(x) \geq 0, \quad i = 1, \dots, n \end{aligned} \quad (20)$$

where  $x \in \mathcal{X}$  for some arbitrary set  $\mathcal{X}$ . Call  $p^*$  the (global) solution to this problem.<sup>8</sup> We introduce the Lagrangian function  $L(x, \alpha)$  with non-negative Lagrangian multipliers  $\alpha_i$  (also called *dual variables*):

$$L(x, \alpha) \triangleq f(x) - \sum_{i=1}^n \alpha_i f_i(x) \quad (21)$$

For a feasible point  $x$ ,  $f_i(x) \geq 0$  and since  $\alpha_i \geq 0$  then we have that  $L(x, \alpha) \leq f(x) \quad \forall x$  feasible. Thus we have:

$$g(\alpha) \triangleq \inf_x L(x, \alpha) \leq p^* \quad (22)$$

$g(\alpha)$  is called the *dual function* and yields a lower bound for the (primal) optimal solution  $p^*$  for each value of  $\alpha \geq 0$ . To get the best bound, we look at the maximal value  $d^*$  for the dual function:

$$d^* \triangleq \sup_{\alpha \geq 0} g(\alpha) = \sup_{\alpha \geq 0} \inf_x L(x, \alpha) \quad (23)$$

The (*Lagrangian*) *dual* problem is to compute  $d^*$ . By equation (22), we have that  $d^* \leq p^*$  (always) which is called *weak duality*. When the primal problem satisfies certain regularity conditions (e.g. it is convex and has a strictly feasible solution - this is Slater's condition), then  $d^* = p^*$  and *strong duality* is said to hold. This means that we can find the minimum value for the primal by maximizing the dual. Moreover, if the dual function can be computed with certainty, then a (saddle) point  $(\alpha, x)$  which is feasible and satisfies  $g(\alpha) = f(x)$  gives the optimal value as well as a certificate.

There are several advantages to look at the dual problem. First of all,  $g(\alpha)$  can usually be computed easily since it is obtained from the *unconstrained* minimization of  $L(x, \alpha)$  over  $x$ . Moreover,  $g(\alpha)$  is *always* convex as is explained in [BV04], and so it doesn't have the problem of having many different local minima.

In the case where all the functions are differentiable, there is a nice geometrical interpretation for the optimal point. Figure 4 explains it (see [Ber99] for more details). If moreover strong duality holds, then the Karush-Kuhn-Tucker (KKT) conditions give sufficient and necessary conditions for  $(x^*, \alpha^*)$  to be a saddle point of the Lagrangian (i.e.  $x^*$  is primal optimal and  $\alpha^*$  is dual optimal). They are

- $(x^*, \alpha^*)$  is feasible (i.e.  $f_i(x^*) \geq 0$  and  $\alpha_i \geq 0$ );
- $\nabla_x L(x^*, \alpha^*) = \nabla f(x^*) - \sum_{i=1}^n \alpha_i \nabla f_i(x^*) = 0$  (this is a generalization of what figure 4 showed but with more than one constraint);

---

<sup>8</sup>By convention,  $p^* = -\infty$  if the feasible solutions are unbounded below and  $p^* = \infty$  if there is no feasible solution. But we won't go into those details.

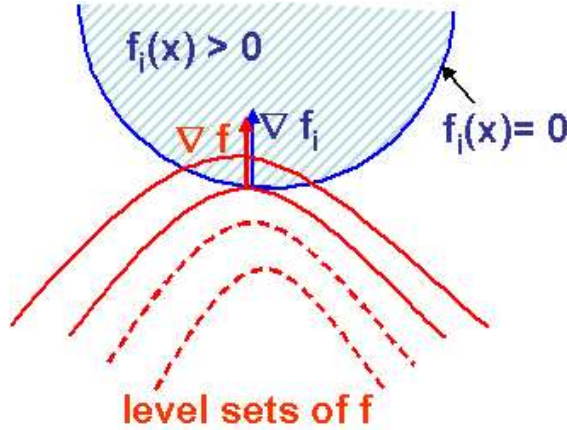


Figure 4: **Geometry of optimal point.** (for one constraint case)  $\nabla f$  has to be parallel **and** in the same direction to  $\nabla f_i$  so that there is no direction of decrease which stays in the feasible region; thus  $\nabla f(x) = \alpha \nabla f_i(x)$  at the optimum  $x$  for some  $\alpha \geq 0$ .

- $\alpha_i^* f_i(x^*) = 0 \quad \forall i.$

The last condition is called *complementary slackness* and indicates that the dual variables are only non-zero for the *active* constraints (that is, the constraints for which  $f_i(x^*) = 0$ ). In the one constraint case, this can be seen as follows. If  $f_i(x) > 0$ , then there is a neighborhood around  $x$  which is still feasible. So if there is a local minimum in this region, it is an *unconstrained* one and this implies  $\nabla f(x^*) = 0$ . If  $\nabla f_i(x^*) \neq 0$  (which is the case for non-degenerate constraints), then the second KKT condition implies that  $\alpha_i = 0$ .

### 5.3 SVM dual and the Kernel trick

We are now in position to do a very useful transformation on the primal problem (19). We first introduce the non-negative dual variables  $\alpha_i$  and  $\beta_i$  and construct the Lagrangian:

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{n} \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i (t(\mathbf{x}^{(i)}) \cdot (\mathbf{w}^T \mathbf{x}^{(i)} + b) - 1 + \xi_i) - \sum_{i=1}^n \beta_i \xi_i \quad (24)$$

We can now find the dual function  $g(\alpha, \beta)$  by minimizing (without constraint) the Lagrangian with respect to  $(\mathbf{w}, b, \xi)$ . From  $\partial L / \partial \mathbf{w} = \mathbf{0}$  we get

$$\mathbf{w} = \sum_{i=1}^n t(\mathbf{x}^{(i)}) \alpha_i \mathbf{x}^{(i)} \quad (25)$$

and thus we see that the maximal margin hyperplane  $\mathbf{w}$  can be written as a linear combination of the data points. The vectors for which  $\alpha_i \neq 0$  are called the *support vectors*, and by the KKT complementary condition  $\alpha_i (t(\mathbf{x}^{(i)}) \cdot (\mathbf{w}^T \mathbf{x} + b) - 1 + \xi_i) = 0$ , they will lie at a distance  $\xi_i / \|\mathbf{w}\|$  on the wrong side of the margin hyperplane.

The two other partials of  $L$  with respect to  $\xi$  and  $b$  yield the constraints<sup>9</sup>

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \alpha_i = \frac{C}{n} - \beta_i \quad (26)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^n t(\mathbf{x}^{(i)})\alpha_i = 0. \quad (27)$$

Substituting this solution back in (24), we get the dual function:

$$g(\alpha, \beta) = \sum_i^n \alpha_i - \frac{1}{2} \sum_i^n \sum_j^n t(\mathbf{x}^{(i)})t(\mathbf{x}^{(j)})\alpha_i\alpha_j(\mathbf{x}^{(i)} \bullet \mathbf{x}^{(j)}) \quad (28)$$

Since  $\beta_i$  doesn't appear in the objective function, we can combine the constraints (26) and (27) together (with  $\beta_i \geq 0$ ) to obtain the equivalent constraint  $C/n \geq \alpha_i \geq 0$ . Since the primal problem was convex and strictly feasible, we can find its solution by equivalently solving the dual problem (by strong duality):

$$\begin{aligned} \text{maximize}_{\alpha} \quad & \sum_i^n \alpha_i - \frac{1}{2} \sum_i^n \sum_j^n t(\mathbf{x}^{(i)})t(\mathbf{x}^{(j)})\alpha_i\alpha_j(\mathbf{x}^{(i)} \bullet \mathbf{x}^{(j)}) \\ \text{subject to} \quad & \sum_i^n t(\mathbf{x}^{(i)})\alpha_i = 0 \\ & \frac{C}{n} \geq \alpha_i \geq 0 \quad \forall i \end{aligned} \quad (29)$$

The optimal classifier is then given by:

$$h_{\alpha}(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^n t(\mathbf{x}^{(i)})\alpha_i(\mathbf{x}^{(i)} \bullet \mathbf{x}) \right) \quad (30)$$

The change of formulation from the primal to the dual has two advantages. First of all, we obtained a quadratic program (29) that generic packages can solve (the primal was also a QP though its structure was slightly more complicated). There is even a simple special purpose algorithm for this formulation called Sequential Minimal Optimization (SMO) which simply does coordinate ascent in the objective function by solving analytically the optimization problems with all variables fixed except a pair (see [CST00, ch 7]).

The other advantage, which is even more remarkable, is that the data only enters in the formulation as dot products. This is a tremendous advantage since this means that we don't need to keep track of the individual components of the vectors; all we need is how they dot themselves with the other vectors. The crucial point is that dot product in submanifolds of space of very high dimensions can be carried out efficiently (if the submanifold has small enough intrinsic dimension). We can thus map our input vector  $\mathbf{x}$  to a feature space  $\mathcal{H}$  with a feature map  $\mathbf{x} \mapsto \mathbf{F}(\mathbf{x}) \in \mathcal{H}$  where  $\mathcal{H}$  can be very high dimensional (even infinite dimensional). This mapping is called the *kernel trick*. The dot product in the new space is

---

<sup>9</sup>Strictly speaking, the Lagrangian is linear in  $\xi$  and  $b$  and so is unbounded unless its derivative with respect to those variables is zero. So formally, the dual function takes value  $-\infty$  for values of  $\xi$  and  $b$  for which the slope is not zero.

given by a *kernel function*  $k(\mathbf{x}, \mathbf{x}') = \mathbf{F}(\mathbf{x}) \bullet \mathbf{F}(\mathbf{x}')$  which can be computed in the original space.

As an example of kernel, consider the kernel

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= (\mathbf{x} \bullet \mathbf{x}')^2 = \left( \sum_{i=1}^n x_i x'_i \right)^2 \\ &= \sum_{i=1}^n \sum_{j=1}^n x_i x_j x'_i x'_j \\ &= \sum_{(i,j)=(1,1)}^{(n,n)} (x_i x_j) (x'_i x'_j) = \mathbf{F}(\mathbf{x}) \bullet \mathbf{F}(\mathbf{x}') \end{aligned}$$

if  $\mathbf{F}(\mathbf{x}) = (x_i x_j)_{(i,j)=(1,1)}^{(n,n)}$  is the feature map, which amounts to map the original vector  $\mathbf{x}$  to all its monomials of degree 2 (and thus a space of dimension  $\binom{n+1}{2}$ ). The kernel can be computed in linear time, even if the feature space has a quadratic number of coordinates. Similarly, we can show that the kernel  $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \bullet \mathbf{x}')^d$  amounts to map a vector to all its monomials of degree  $d$  (there are  $\binom{n+d-1}{d}$  of them!).

The advantage of mapping the data to high dimensional feature space is that a linear boundary in the feature space can correspond to an arbitrary non-linear boundary in the original space. SVM with the kernel trick can thus learn a very rich class of functions. But because of the strict max-margin criterion for the choice of classifier, it still doesn't suffer from the curse of dimensionality.

The kernel computes the distance in the feature space and is thus a measure of dissimilarity. The design of kernels which represent well the underlying interactions has become an art. They can be created by starting first from a feature map; or by defining the kernel directly; or by constructing it from other kernels.

## 6 Multi-class SVM

Now that we have seen how SVMs can be used to do binary classification, we have to consider how they are adapted to do multiway classification. The approach which is often used (and works well in practice) is to simply train  $k$  binary classifier (independently) which will each learn to recognize one class vs. the rest (and is thus called the *one versus the rest* approach [SS02, ch 7]). Then the winner class can be chosen with

$$h(\mathbf{x}) = \arg \max_{j \in \{1, \dots, k\}} \hat{w}_j^T \mathbf{x} + b_j \quad (31)$$

where each  $(\hat{w}_j^T, b_j)$  is the hyperplane for the class  $j$  and  $\hat{w}_j$  is normalized. This classifier amounts to choose the class of which its binary hyperplane is the farthest (in signed Euclidean distance) from the point that one wants to classify. This makes sense since the farther from the plane it is, the more confidence you have in the classification. Figure 5 gives an example of multiway classification using one vs. rest hyperplanes.

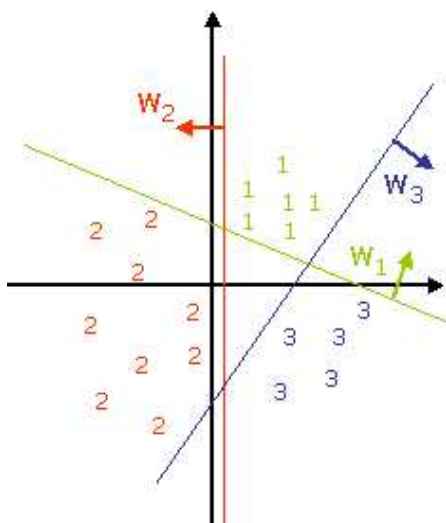


Figure 5: **Multi-class SVM**. Each plane separates one class with the rest with an optimal margin. The class of a point is finally given by the farthest plane to it (if it is on the negative side of all the planes, it is assigned to the closest plane (since this is negative distance) - or sometimes the classification is just rejected to improve accuracy).

A theoretical problem with this approach is that the binary classifiers are trained independently and thus can't 'cooperate'. For example, it could be advantageous to sacrifice the accuracy on a really bad class if the resulting global classifier is still more accurate by the ways of interaction of the data. This is not possible with the one vs. rest approach.

Other approaches are to use pairwise classification; or to train jointly the binary classifiers with a joint objective function. It was mentioned in [SS02, ch 7] that pairwise or joint classification didn't improve much the results of one vs. rest in practice. But to generalize the setting to multi-label classification, one has to consider the joint objective function. This was presented first in [WW99] and then more thoroughly in [CS01]. We will present the approach with the notation of  $M^3$ -net instead of repeating it twice (once for the single-label case; once for the multi-label case). Before moving to the  $M^3$ -net, we will analyze how multi-label multi-class classification was done with SVM before the creation of  $M^3$ -net.

The one vs. rest approach learns a binary classifier for each class. In the case of a joint label, the number of classes is exponential in the number of labels, and so it is intractable in general to do the labelling jointly. It was normally assumed that each labelling was independent. Figure 6 shows the corresponding graphical model for the FAQ problem. So in this case the classifier would be:

$$y_i = h_i(\mathbf{x}_i) = \arg \max_{j \in \{1, \dots, k\}} \hat{w}_{ij}^T \mathbf{x}_i + b_{ij} \quad (32)$$

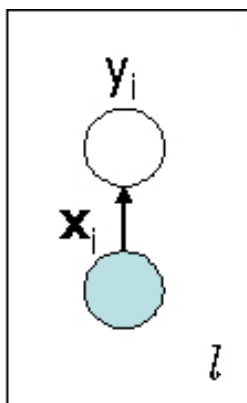


Figure 6: SVM model for the FAQ problem. The labels are assumed to be independent.

## 6.1 Analysis

The advantages of the SVM approach is the possibility to use high dimensional feature spaces (using the kernel trick). For the FAQ tagging problem, one could use very rich feature functions of the sentences, richer than in the CRF model. Also, the max-margin approach gives theoretical bound on the generalization error. On the other hand, the SVM here doesn't exploit the structure of the labels, which was found to be crucial in this case as explained in section 3.1. This last part is what the M<sup>3</sup>-net formalism will solve.

# 7 Max-Margin Markov Networks (M<sup>3</sup> net)

## 7.1 Margin-based Structured Classification

The M<sup>3</sup>-net uses a CRF to parameterize in a factored way the conditional distribution  $P(\mathbf{y}|\mathbf{x})$ , i.e.

$$P(\mathbf{y}|\mathbf{x}) \propto \exp(\mathbf{w}^T \mathbf{F}(\mathbf{x}, \mathbf{y})) \quad (33)$$

exactly like in (12) and with the feature vectors defined in (11).  $\mathbf{w}^T \mathbf{F}(\mathbf{x}, \mathbf{y})$  is called the *discriminant function*, and can be seen as the measure of compatibility between the observation sequence  $\mathbf{x}$  and the joint label  $\mathbf{y}$  [AH03]. The corresponding classifier is thus

$$h_{\mathbf{w}}(\mathbf{x}) = \arg \max_{\mathbf{y}} \mathbf{w}^T \mathbf{F}(\mathbf{x}, \mathbf{y}) \quad (34)$$

We now want to include some kind of max-margin argument in the training part and thus we don't use the usual Maximum Likelihood estimate for CRF. Like we had discussed before, we are not interested in the exact form of  $P(\mathbf{y}|\mathbf{x})$ , we are rather interested in the classifier  $\arg \max_{\mathbf{y}} \mathbf{w}^T \mathbf{F}(\mathbf{x}, \mathbf{y})$  and with what confidence the classification is done. Note that in the single label case, we can recover equation (32) from (34) by defining  $\mathbf{w} = (w_1^T, \dots, w_k^T)^T$  and  $\mathbf{F}(\mathbf{x}, j) = (0, 0, 0, \dots, \mathbf{x}_j^T, 0, 0, \dots, 0)^T$  and augmenting  $\mathbf{x}$  by the constant coordinate one to obtain homogeneous coordinates.

A suitable notion of margin in this case is the difference between the value of the discriminant for the true label with the value of the best runner up. If the data is linearly separable, we can rescale  $\mathbf{w}$  so that the minimal difference is 1 (similar to equation (15)). The respected margin constraint is thus:

$$\mathbf{w}^T \mathbf{F}(\mathbf{x}, \mathbf{t}(\mathbf{x})) \geq 1 + \mathbf{w}^T \mathbf{F}(\mathbf{x}, \mathbf{y}), \quad \forall \mathbf{y} \neq \mathbf{t}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{S} \quad (35)$$

and we see here the convenience of having defined the target function  $\mathbf{t}(\mathbf{x})$ . Again, for the case of non-separable data, we introduce the non-negative slack variables  $\xi_{\mathbf{x}}$  (one for each element of the sample). We also define  $\Delta \mathbf{F}_{\mathbf{x}}(\mathbf{y}) = \mathbf{F}(\mathbf{x}, \mathbf{t}(\mathbf{x})) - \mathbf{F}(\mathbf{x}, \mathbf{y})$  for brevity of notation. The constraint (36) then becomes:

$$\mathbf{w}^T \Delta \mathbf{F}_{\mathbf{x}}(\mathbf{y}) \geq 1 - \xi_{\mathbf{x}}, \quad \forall \mathbf{y} \neq \mathbf{t}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{S} \quad (36)$$

Finally, Taskar et al. in [TGK03] suggests the use of a margin which should be proportional to the number of individual errors  $\Delta \mathbf{t}_{\mathbf{x}}(\mathbf{y}) \triangleq \sum_{i=1}^l I\{y_i \neq (\mathbf{t}(\mathbf{x}))_i\}$  made inside a label. This amounts to define a loss function which is proportional to the number of individual label errors (with constant of proportionality  $C$ ). Similarly as for the binary SVM case, we will then have that  $\frac{C}{n} \sum_{\mathbf{x} \in \mathcal{S}} \xi_{\mathbf{x}}$  will be an upper bound for the empirical risk according to this loss function.

With all those modifications, the max-margin framework for  $M^3$ -net thus becomes:

$$\begin{aligned} & \text{minimize} && \frac{\mathbf{w}^T \mathbf{w}}{2} + \frac{C}{n} \sum_{\mathbf{x} \in \mathcal{S}} \xi_{\mathbf{x}} && (37) \\ & \mathbf{w}, \xi \\ & \text{subject to} && \mathbf{w}^T \Delta \mathbf{F}_{\mathbf{x}}(\mathbf{y}) \geq \Delta \mathbf{t}_{\mathbf{x}}(\mathbf{y}) - \xi_{\mathbf{x}}, \quad \forall \mathbf{x} \in \mathcal{S}, \quad \forall \mathbf{y} \end{aligned}$$

Note that the constraint  $\xi_{\mathbf{x}} \geq 0$  is included by the case when  $\mathbf{y} = \mathbf{t}(\mathbf{x})$ . As in the usual SVM approach that we have described in section 5.3, we can derive the dual formulation of this problem by minimizing the Lagrangian with respect to  $\mathbf{w}$  and  $\xi_{\mathbf{x}}$  and then substituting back the solution to get the dual objective function to maximize. Here, there was one constraint for each  $\mathbf{x} \in \mathcal{S}$  and  $\mathbf{y}$  and so we introduce the dual variables  $\alpha_{\mathbf{x}}(\mathbf{y})$ . The derivations are pretty similar to the standard case, and so we only quote the result:

$$\begin{aligned} & \text{maximize} && \sum_{\mathbf{x}, \mathbf{y}} \alpha_{\mathbf{x}}(\mathbf{y}) \Delta \mathbf{t}_{\mathbf{x}}(\mathbf{y}) - \frac{1}{2} \sum_{\mathbf{x}, \mathbf{y}} \sum_{\mathbf{x}', \mathbf{y}'} \alpha_{\mathbf{x}}(\mathbf{y}) \alpha_{\mathbf{x}'}(\mathbf{y}') (\Delta \mathbf{F}_{\mathbf{x}}(\mathbf{y}) \bullet \Delta \mathbf{F}_{\mathbf{x}'}(\mathbf{y}')) && (38) \\ & \alpha_{\mathbf{x}}(\mathbf{y}) \\ & \text{subject to} && \sum_{\mathbf{y}} \alpha_{\mathbf{x}}(\mathbf{y}) = n^{-1} C, \quad \forall \mathbf{x} \in \mathcal{S}; \quad \alpha_{\mathbf{x}}(\mathbf{y}) \geq 0 \quad \forall \mathbf{x} \in \mathcal{S}, \quad \forall \mathbf{y} \end{aligned}$$

Up to this point, there was not much anything new. This derivation was an almost exact adaptation of the formulation of Crammer and Singer given in [CS01], but for the multi-label case. The result here is an optimization problem with an exponential number of variables and constraints, and thus doesn't seem to be practical.

## 7.2 Key insight to obtain the factored dual

But we haven't used the fact yet that we had sparse correlations in the feature representation. How can we do that? The key insight was to notice, like was already mentioned in [CS01] for

the single-label multi-class formulation, that the dual variables  $\alpha_{\mathbf{x}}(\mathbf{y})$  satisfy the constraints of an *unnormalized joint probability distribution* over  $\mathbf{y}$ . We can thus try to *marginalize* them to obtain only the relevant clique marginals respecting the structure of the graph. Indeed, because of the structure of the objective function, **the dual variables will ‘factorize’ in exactly the same way as the feature functions**. For our example with sequential labels, using equation (11) for the definition of the joint feature vectors, we obtain that

$$\Delta \mathbf{F}_{\mathbf{x}}(\mathbf{y}) = \sum_{i=1}^l \Delta \mathbf{F}_i(\mathbf{x}, y_i, y_{i+1}) \quad (39)$$

and thus by pushing the sum inward (like in variable elimination in graphical models), we obtain that

$$\sum_{\mathbf{x}, \mathbf{y}} \sum_{\mathbf{x}', \mathbf{y}'} \alpha_{\mathbf{x}}(\mathbf{y}) \alpha_{\mathbf{x}'}(\mathbf{y}') (\Delta \mathbf{F}_{\mathbf{x}}(\mathbf{y}) \bullet \Delta \mathbf{F}_{\mathbf{x}'}(\mathbf{y}')) =$$

$$\sum_{\mathbf{x}, \mathbf{x}'} \sum_{i=1}^l \sum_{i'=1}^l \sum_{y_i, y_{i+1}} \sum_{y_{i'}, y_{i'+1}} (\Delta \mathbf{F}_i(\mathbf{x}, y_i, y_{i+1}) \bullet \Delta \mathbf{F}_{i'}(\mathbf{x}', y_{i'}, y_{i'+1})) \left( \sum_{\mathbf{y} \sim [y_i, y_{i+1}]} \alpha_{\mathbf{x}}(\mathbf{y}) \right) \left( \sum_{\mathbf{y}' \sim [y_{i'}, y_{i'+1}]} \alpha_{\mathbf{x}'}(\mathbf{y}') \right)$$

where  $\mathbf{y} \sim [y_i, y_{i+1}]$  denotes a full assignment  $\mathbf{y}$  consistent with the partial assignment  $y_i, y_{i+1}$ . We see that most of the (exponential) sum is simply doing a marginalization over the dual variable. So it makes sense to define the marginal dual variables

$$\mu_{\mathbf{x}}(y_i, y_{i+1}) \triangleq \sum_{\mathbf{y} \sim [y_i, y_{i+1}]} \alpha_{\mathbf{x}}(\mathbf{y})$$

$$\mu_{\mathbf{x}}(y_i) \triangleq \sum_{\mathbf{y} \sim [y_i]} \alpha_{\mathbf{x}}(\mathbf{y})$$

. Then the objective function depends only on (a polynomial number of) those, because of the factorization of the features. We have thus transformed an exponential sum into a polynomial sum! But it is not clear that we could obtain an equivalent formulation using the dual marginals. For each dual marginal assignment, we want to be able to construct a joint distribution. In the case of a tree, this can always be done if the marginals satisfy the consistency condition:

$$\sum_{y_i} \mu_{\mathbf{x}}(y_i, y_{i+1}) = \mu_{\mathbf{x}}(y_{i+1}) \quad (40)$$

In the case of general triangulated graphs, we can define consistency conditions for the clique marginals at their separator set from a junction tree which will also be sufficient to define a joint (see [CDLS99]). For non triangulated graphs, though, there is no clear way to define consistency conditions to ensure the existence of a joint.

Given those additional consistency conditions, we can thus define the *factored dual formulation*:

$$\text{maximize}_{\mu_{\mathbf{x}}(y_i, y_{i+1}), \mu_{\mathbf{x}}(y_i), 1 \leq i \leq l} \sum_{\mathbf{x}} \sum_{i=1}^l \sum_{y_i} \mu_{\mathbf{x}}(y_i) \Delta t_{\mathbf{x}}(\mathbf{y}) \quad (41)$$

$$-\frac{1}{2} \sum_{\mathbf{x}, \mathbf{x}'} \sum_{i, i'} \sum_{(y_i, y_{i+1}), (y_{i'}, y_{i'+1})} \mu_{\mathbf{x}}(y_i, y_{i+1}) \mu_{\mathbf{x}'}(y_{i'}, y_{i'+1}) (\Delta F_{\mathbf{x}}(y_i, y_{i+1}) \bullet \Delta F_{\mathbf{x}'}(y_{i'}, y_{i'+1}))$$

subject to  $\sum_{y_i} \mu_{\mathbf{x}}(y_i, y_{i+1}) = \mu_{\mathbf{x}}(y_{i+1}), \quad \sum_{y_i} \mu_{\mathbf{x}}(y_i) = n^{-1}C, \quad \mu_{\mathbf{x}}(y_i, y_{i+1}) \geq 0$

which is now a simple quadratic program with a polynomial number of variables! Because each feasible point of the factored dual problem is also a feasible point for the dual (we can define a joint from the consistent marginals), and vice versa; and that the objective functions are the same; then we see that the two problems are equivalent. Finally, we can find back the primal solution from the dual variables as usual:

$$\mathbf{w} = \sum_{\mathbf{x}} \sum_{i=1}^l \sum_{y_i, y_{i+1}} \mu_{\mathbf{x}}(y_i, y_{i+1}) \Delta F_{\mathbf{x}}(y_i, y_{i+1}) \quad (42)$$

and this formulation can also make use of the kernel trick as everything is expressed in terms of dot products.

### 7.3 General graphs and other issues

We have presented all the models in this framework by using a sequential example (for simplicity), but all the given frameworks apply also for arbitrary graph structure on the labels (the notation just becomes a bit more messy). In the case of the  $M^3$ -net, we have said that if the graph is triangulated, then we can easily build the factor dual by defining the clique marginals with the consistency conditions on the separator sets. If the graph is non-triangulated, then [TGK03] presents an approach to triangulate it and still obtain an equivalent optimization problem (but with a number of new constraints exponential in the size of the cliques). They mention at the same time that trying to optimize (41) with only local consistency conditions (not the triangulated version) amounts to optimize the objective function on an approximation of the marginal polytope.

Taskar et al. also presents in the paper [TGK03] an adapted version of SMO to solve the optimization problem more efficiently than generic QP solvers. Finally, they prove a generalization bound on the expected risk in a similar fashion than in the SVM case, using their max-margin approach.

### 7.4 Analysis and other related papers

We have seen that  $M^3$ -nets can model joint labels efficiently using the CRF formalism; can use the kernel trick in the optimization problem arising from their max-margin criteria; and have generalization guarantees. Taskar et al. presented in [TGK03] some experimental comparisons between different classification approaches for handwritten word recognition with their  $M^3$ -net approach.  $M^3$ -net outperforms them consistently (of course); but some people have criticized their implementation of the other algorithms (CRF, etc.), and so more test would be warranted just as a consistency check.

This year, Altun et al. have developed a similar setting for multi-label, multi-class classification (in the sequential case) called *Hidden Markov Support Vector Machines* (HM-SVM) (see [ATH03] and [AH03]). They also used CRF and presented a parallel development to the one of the M<sup>3</sup>-net paper until the dual formulation of their max-margin problem (equation (38)). They haven't cast it exactly in the same form and so haven't noticed that the dual variables could be seen as joint distributions. Their approach was then to try to use heuristics to solve this problem with an exponential number of constraints, with hope that in practice only tiny portion of the constraints will be relevant. Their results were only on very small data set, though, and so nothing can be inferred for now. A SVM approach to multi-label classification was also presented in [EW01], though we haven't had the time to look at it thoroughly.

## 8 Conclusion

We have given a survey of different methods to do multi-label, multi-class classification. Generative models can be efficient if one has very good prior knowledge about the problem, though they are in general suboptimal and non-robust because of their modelling assumptions. Discriminative graphical models like CRFs can model efficiently the structure of the labels, and are not suboptimal like the generative models, but they don't have the generalization guarantees of SVMs and the possibility to use the kernel trick. We have presented SVMs as a max-margin linear classifier, using the notion of duality to obtain a formulation which can be kernelized. SVMs have good generalization guarantees and can make use of the high dimensional feature spaces, but don't handle interactions between the label in the multi-label case. Finally, M<sup>3</sup>-net provides both the advantages of CRFs and SVMs, by using a key transformation in the standard max-margin framework of a multi-label SVM inspired by CRFs. The crucial point was that the dual variables would factor as the feature vectors. Taskar et al. also made sure to adapt the standard results for SVMs to their new framework by presenting an efficient algorithm for the learning part (SMO) and by giving a bound on the expected risk (generalization error). There was also a discussion of how to use them for arbitrary graphical structure. This framework will open the max-margin based methods to a whole new set of problems with structured data. Interesting research directions for M<sup>3</sup>-net in the future could be, apart more experimental results, to look at how they could apply it to learn the network structure, or to deal with hidden variables.

## Acknowledgements

This paper is submitted as a class project for Peter's Bartlett Statistical Learning Theory class at UC Berkeley in Fall 2003. I would like to thank Peter Bartlett for the suggestion of topic as well as some interesting discussions about computational learning theory. I also thank Ben Taskar and Carlos Guestrin for having kindly accepted to answer my questions about their paper.

## References

- [AH03] Yasemin Altun and Thomas Hofmann. Large margin methods for label sequence learning. In *8th European Conference on Speech Communication and Technology (EuroSpeech)*, 2003. 4.3, 7.1, 7.4
- [ATH03] Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden Markov support vector machine. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, Washington DC, 2003. 7.4
- [Ber99] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 2nd edition, 1999. 5.2, 5.2
- [BSW99] Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. An algorithm that learns what’s in a name. *Machine Learning Journal Special Issue on Natural Language Learning*, 34:211–232, 1999. 2.1
- [Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. 5.1, 5.1
- [BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. Note published yet; available from <http://www.stanford.edu/~boyd/cvxbook.html>. 5.2, 5.2
- [CDLS99] Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer, New York, NY, 1999. 7.2
- [CS01] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001. 6, 7.1, 7.2
- [CS02] Nello Cristianini and Bernhard Schölkopf. Support vector machines and kernel methods: The new generation of learning machines. *AI Magazine*, 23(3):31–41, 2002. 5.1
- [CST00] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000. 5.1, 5.3
- [CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. 1, 5.1
- [Die02] Thomas G. Dietterich. Machine learning for sequential data: A review. In *Lecture Notes in Computer Science*. Springer-Verlag, 2002. 2.1, 3
- [Dur96] R. Durrett. *Probability: theory and examples*. Duxbury Press, Belmont, CA, second edition, 1996. 4

- [EW01] Andre Elisseeff and Jason Weston. Kernel methods for multi-labelled classification and categorical regression problems. Technical report, Biowulf Technologies, 2001. [7.4](#)
- [Jor98] Michael I. Jordan, editor. *Learning in Graphical Models*. MIT press, Cambridge, MA, 1998. [1](#)
- [Jor03] Michael I. Jordan. An introduction to probabilistic graphical models. Book in preparation, 2003. [3](#), [3.2](#), [4.1](#), [4.1](#), [6](#), [4.2](#)
- [LMP01] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001. [4.2](#), [4.2](#), [4.2](#)
- [MFP00] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. 17th International Conf. on Machine Learning*, pages 591–598. Morgan Kaufmann, San Francisco, CA, 2000. [2.1](#), [7](#)
- [Rab89] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. [3.1](#)
- [SBV99] Bernhard Schölkopf, Christopher Burges, and Vladimir Vapnik. Incorporating invariances in support vector learning machines. In *Proceedings, International Conference on Artificial Neural Networks*, Berlin, 1999. Springer-Verlag. [1](#)
- [SS02] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002. [2.2](#), [4.3](#), [5.1](#), [6](#), [6](#)
- [TAK02] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, 2002. Morgan Kaufmann. [2.1](#)
- [TGK03] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. To appear in Neural Information Processing Systems Conference (NIPS03), Vancouver, Canada, December 2003. ([document](#)), [1](#), [2.1](#), [4.2](#), [4.3](#), [7.1](#), [7.3](#), [7.4](#)
- [Vap99] V. Vapnik. *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer, New York, NY, 1999. [1](#), [2.2](#), [2.3](#), [5.1](#), [5.1](#), [5.1](#)
- [WW99] J. Weston and C. Watkins. Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, 4 1999. [6](#)