Outline

- Introduction
- Multiclass classification
- Structured prediction: fixed-size structures
- Structured prediction: variable-size structures

1 Introduction

In many supervised learning settings in practice, the labels are not simply binary or one of a small number of classes, but rather, are more complex structured objects. For example, in part-of-speech tagging, the instances are sentences and the labels are sequences of part-of-speech tags. In natural language parsing, the instances are sentences and the labels are parse trees. In image segmentation, the instances are images and the labels are segment labelings of all the pixels. In each of these settings, there is a combinatorially large number of possible labels, and standard multiclass classification methods become infeasible. In this lecture we give a brief glimpse of structured prediction methods, which extend multiclass classification learning methods to structured settings.

We start by reviewing multiclass classification, and then discuss two approaches to structured prediction: conditional random fields (CRFs), which extend multiclass logistic regression to structured settings, and structured support vector machines (StructSVMs), which extend multiclass SVMs to structured settings. In many structured prediction applications, one must allow for structures of varying sizes (such as parse trees for sentences of varying lengths). Often in the literature, the descriptions of the basic algorithms underlying structured prediction and of the feature constructions/parameterizations needed to incorporate variable-size structures are wrapped up together, and it is hard to understand the precise role that each of these plays in the final learning methods. Here we present these aspects separately in order to clarify the role of the different pieces involved.

2 Multiclass Classification

Let us start by reviewing two discriminative methods for multiclass classification: multiclass logistic regression and multiclass SVMs. We will denote the set of labels by $\mathcal{Y} = \{1, \ldots, K\}$. For simplicity, in this section we will assume instances $x$ are $d$-dimensional feature vectors, with instance space $\mathcal{X} \subseteq \mathbb{R}^d$. 
2.1 Multiclass Logistic Regression

Recall that, similar to binary classification, multiclass logistic regression is a discriminative probabilistic model that models the conditional class probabilities \( p(y | x) \) for each class \( y \in \{1, \ldots, K\} \). In particular, just as binary logistic regression uses a real-valued function \( f(x) \) to model the class probability \( p(+1 | x) \) via the logistic function, similarly, multiclass logistic regression uses \( K \) real-valued functions \( f_y(x) \), one for each class \( y \), to model the class probabilities \( p(y | x) \) via the softmax function as follows:

\[
p(y | x) = \frac{\exp(f_y(x))}{\sum_{y'=1}^{K} \exp(f_{y'}(x))}.
\]

In linear multiclass logistic regression, one models each function \( f_y \) as \( f_y(x) = w_y^\top x \) for some weight vector \( w_y \in \mathbb{R}^d \); this amounts to assuming the following parametric model for the class probabilities:

\[
p(y | x; w) = \frac{\exp(w_y^\top x)}{\sum_{y'=1}^{K} \exp(w_{y'}^\top x)},
\]

where \( w = (w_1^\top, \ldots, w_K^\top)^\top \in \mathbb{R}^{Kd} \) denotes the collective parameter vector.

Given training data \( S = ((x_1, y_1), \ldots, (x_m, y_m)) \), where \( x_i \in \mathbb{R}^d \) and \( y_i \in \{1, \ldots, K\} \), one trains a multiclass logistic regression classifier by estimating the parameters \( w_1, \ldots, w_K \in \mathbb{R}^d \) via (conditional) maximum likelihood estimation. In particular, the conditional log-likelihood of the data under parameters \( w \) is

\[
\sum_{i=1}^{m} \ln p(y_i | x_i; w) = \sum_{i=1}^{m} \left( w_{y_i}^\top x_i - \ln \left( \sum_{y=1}^{K} \exp(w_y^\top x) \right) \right).
\]

This is a concave function of \( w \) and can be maximized using gradient-based methods to obtain maximum likelihood parameter estimates \( \hat{w} \). Often, one adds an \( L_2 \) regularization term \( \lambda \sum_{y=1}^{K} \| w_y \|_2^2 \) to the above objective; maximizing the resulting objective corresponds to computing MAP parameter estimates under independent Gaussian priors on the weight vectors.

For classification with 0-1 loss, having learned \( \hat{w} \), given a new instance \( x \) one predicts a class label \( \hat{y} \) according to the highest estimated class probability:

\[
\hat{y} \in \arg \max_{y \in \{1, \ldots, K\}} p(y | x; \hat{w})
\]

\[
= \arg \max_{y \in \{1, \ldots, K\}} \hat{w}_y^\top x.
\]

More generally, for cost-sensitive classification, one can predict a class label \( \hat{y} \) that minimizes expected cost under the estimated class probabilities.

2.2 Multiclass Support Vector Machines

There are many multiclass extensions of SVMs; here we discuss the Crammer-Singer variant.

Here again, one learns \( K \) real-valued functions \( f_y(x) \), one for each class \( y \). Here each function \( f_y(x) \) is viewed as producing a ‘score’ for class \( y \) given the instance \( x \). Again, with linear multiclass SVMs, these functions are parametrized as \( f_y(x) = w_y^\top x \) for some weight vectors \( w_1, \ldots, w_K \in \mathbb{R}^d \).

\(^1\)Strictly speaking, due to the normalization, we only need \( K - 1 \) functions \( f_1(x), \ldots, f_{K-1}(x) \); we can arbitrarily fix the \( K \)-th function to be \( f_K(x) = 0 \). For simplicity, however, we will use \( K \) functions; this will also match the formulation used in multiclass SVMs.
In this case, given training data $S$ as above, the learning algorithm aims to maximize the ‘margin’ between the score of the correct class and the score of any other class. In particular, for classification with 0-1 loss, one minimizes the following objective as a function of the parameters $w = (w_1^\top, \ldots, w_K^\top)^\top \in \mathbb{R}^{Kd}$:

$$
\sum_{i=1}^m \max \left( 0, 1 - (w_{y_i}^\top x_i - \max_{y \neq y_i} w_y^\top x_i) \right) + \lambda \sum_{y=1}^K \|w_y\|_2^2.
$$

This is a convex function of $w$ and can be minimized efficiently, either directly or via the dual, to obtain parameter estimates $\hat{w}$.

For classification with a cost-sensitive loss, a cost-sensitive variant minimizes the following objective instead, where $c_{yy'}$ denotes the cost of predicting label $y'$ when the true label is $y$:

$$
\sum_{i=1}^m \max \left( 0, \max_{y \neq y_i} (c_{y_i,y} - (w_{y_i}^\top x_i - w_{y_i}^\top x_i)) \right) + \lambda \sum_{y=1}^K \|w_y\|_2^2.
$$

This is again a convex function of $w$ and can be minimized efficiently to obtain parameter estimates $\hat{w}$.

In either case, having learned $\hat{w}$, given a new instance $x$ one predicts a class label $\hat{y}$ according to the highest scoring class under $\hat{w}$:

$$
\hat{y} \in \arg \max_{y \in \{1, \ldots, K\}} \hat{w}_y^\top x.
$$

### 2.3 An Alternative View

In both multiclass logistic regression and multiclass SVMs above, we viewed the score of a class $y$ given $x$ as being obtained by taking the dot product of a parameter vector $w_y$ with $x$. An alternative view is the following: given an instance $x$, for each label $y \in \{1, \ldots, K\}$, construct a $(Kd)$-dimensional joint instance-label feature vector $\phi(x, y)$ with $Kd$-dimensional blocks, all of which are zero except the $y$-th block which has the $d$-dimensional vector $x$ in it. Also recall that we collect the $K$ $d$-dimensional parameter vectors $w_y$ into a $(Kd)$-dimensional parameter vector $w$ that has $w_y$ in block $y$. Then clearly, we have

$$
w^\top \phi(x, y) = w_y^\top x,
$$

and so we can also view the score of a class $y$ as being obtained by taking the dot product of the full parameter vector $w$ with the joint instance-label feature vector $\phi(x, y)$. The parameterizations used in the structured prediction settings we discuss below bear more resemblance to this latter view (although with different instance-label feature constructions).

### 3 Structured Prediction: Fixed-Size Structures

Now consider settings where the labels are combinatorial structures. In particular, we will consider settings where each label consists of $n$ elements/sub-labels that may have some relationships or “dependencies” among them. For example, in a sequence labeling task such as part-of-speech (PoS) tagging, instances are sentences and labels are sequences of PoS tags with dependencies among them (e.g. it is unusual to have a determiner followed by a verb). Similarly, in image segmentation, instances are images and labels are assignments of segments to each pixel with dependencies among them (e.g. most pixels belong to the same segment as their neighbors). Formally, we will denote a label as $y = (y_1, \ldots, y_n)$, where each $y_i$ takes a value in $\{1, \ldots, K\}$. Thus there are $K^n$ possible labels $y$. 
When there is a combinatorial number of labels, treating each label as a separate class and directly applying a standard multiclass classification approach is infeasible, since that would require maintaining a combinatorial number of weight vectors/parameters. Below we discuss two widely used approaches to structured prediction: conditional random fields (CRFs), which can be viewed as generalizing multiclass logistic regression to structured settings, and structured SVMs (StructSVMs), which generalize multiclass SVMs to structured settings.

3.1 Conditional Random Fields

Conditional random fields (CRFs) are discriminative probabilistic models that generalize multiclass logistic regression to structured settings. As with logistic regression, they model the conditional label probabilities \( p(y \mid x) \). However, in a structured setting, modeling a general conditional distribution \( p(y \mid x) \) requires learning \( O(K^n) \) real-valued functions \( f_y(x) \), one for each labeling \( y \); for large \( n \), this is clearly intractable! Another extreme alternative would be to predict each sub-label \( y_t \) independently; this would amount to modeling the distribution \( p(y \mid x) \) as a product of individual distributions \( p(y_t \mid x) \), and would require learning only \( O(nK) \) real-valued functions \( f_t(y)(x) \) (one for each possible labeling \( y \in \{1, \ldots, K\} \) for each element \( t \in \{1, \ldots, n\} \)), but would ignore useful structure/dependencies between sub-labels.

CRFs allow models that lie somewhere in between the above two extremes by assuming a graphical model structure on \( y \) (conditioned on \( x \)). The graphical model captures the main dependencies between the \( n \) label elements \( y_1, \ldots, y_n \) via an undirected graph on \( n \) nodes. Denote the set of (maximal) cliques of the graph by \( C \); for each clique \( C \in C \), denote by \( y_C \) the vector of label elements in the clique \( C \); \( y_C = (y_t)_{t \in C} \). Then a CRF assumes a model of the form

\[
p(y \mid x) = \frac{1}{Z_x} \prod_{C \in C} \psi_C(x, y_C).
\]

For each clique \( C \), this effectively involves \( K^{|C|} \) real-valued functions of \( x \), one for each possible labeling \( y_C \) of the nodes in \( C \): \( f_{y_C}(x) = \ln \psi_C(x, y_C) \). Thus the total number of real-valued functions to be learned is \( \sum_{C \in C} K^{|C|} \). For example, if the graphical model is a tree (such as a chain), then there are \( n - 1 \) cliques (edges) of size 2, and one needs to learn \( (n - 1)K^2 \) functions.

It is common to assume a log-linear form for the clique potentials. Here one constructs a feature vector \( \phi_C(x, y_C) \) for each clique \( C \), say of dimensionality \( d_C \), and then parametrizes the clique potentials as \( \psi_C(x, y_C) = \exp(\sum_{C \in C} w_C \phi_C(x, y_C)) \) for some weight vectors \( w_C \in \mathbb{R}^{d_C} \). We will collect the weight vectors \( w_C \) into a parameter vector \( w \in \mathbb{R}^{d} \), where \( d = \sum_{C \in C} d_C \); the log-linear conditional probability model under parameters \( w \) can then be written as

\[
p(y \mid x; w) = \frac{1}{Z_x(w)} \exp \left( \sum_{C \in C} w_C^T \phi_C(x, y_C) \right).
\]

Given labeled training data \( S = ((x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})) \), where each \( x^{(i)} \) is an instance and each \( y^{(i)} \) is a labeling in \( \{1, \ldots, K\}^n \), one trains a log-linear CRF model by estimating the parameters \( w \) via (conditional) maximum likelihood estimation. The (conditional) log-likelihood of the data under parameters \( w \) is

\[
\sum_{i=1}^{m} \ln p(y^{(i)} \mid x^{(i)}; w) = \sum_{i=1}^{m} \left( \sum_{C \in C} w_C^T \phi_C(x^{(i)}, y^{(i)}_C) - \ln \left( \sum_{y} \exp \left( \sum_{C \in C} w_C^T \phi_C(x^{(i)}, y_C) \right) \right) \right) \frac{1}{Z_x(w)}
\]

\( \sum_{C \in C} K^{|C|} \)

\^2Recall that clique potentials are non-negative; it is therefore easier to work with their logarithms, which are general real-valued functions.
This is a concave function of \( w \), and can be maximized via iterative numerical optimization methods. For models with small clique sizes, such as trees, the optimization can be performed efficiently.

Having learned \( \hat{w} \), given a new instance \( x \) one generally predicts a labeling \( \hat{y} \) with maximal conditional probability under \( \hat{w} \):

\[
\hat{y} \in \arg \max_y p(y | x; \hat{w}) = \arg \max_y \sum_{C \in C} \hat{w}_C^\top \phi_C(x, y_C) = \arg \max_y \hat{w}^\top \phi(x, y),
\]

where we have also collected the feature vectors \( \phi_C(x, y_C) \) into a single feature vector \( \phi(x, y) \) of dimensionality \( d = \sum_{C \in C} d_C \). Depending on the structure and sizes of the cliques \( C \) and the forms of the feature vectors \( \phi_C(x, y_C) \), the above maximization may or may not be possible to perform efficiently. When exact maximization is difficult, one often settles for an approximate maximization of some sort (usually related to approximate MAP inference in the associated graphical model).

### 3.1.1 Chain CRFs

A particular case of interest for which the above maximization can be performed efficiently is that of chain CRFs, in which the cliques are simply edges connecting consecutive sub-labels, \((y_t, y_{t+1})\). Chain CRFs are useful for modeling sequence labels. Here it’s common to also associate node potentials with each sub-label \( y_t \); the conditional probability model in this case is given by

\[
p(y | x; w) = \frac{1}{Z_x(w)} \exp \left( \sum_{t=1}^{n} w_t^\top \phi_t(x, y_t) + \sum_{t=1}^{n-1} w_{t,t+1}^\top \phi_{t,t+1}(x, y_t, y_{t+1}) \right),
\]

and the log-likelihood of the training data is then given by

\[
\sum_{i=1}^{m} \ln p(y^{(i)} | x^{(i)}; w) = \sum_{i=1}^{m} \left( \sum_{t=1}^{n} w_t^\top \phi_t(x^{(i)}, y_t^{(i)}) + \sum_{t=1}^{n-1} w_{t,t+1}^\top \phi_{t,t+1}(x^{(i)}, y_t^{(i)}, y_{t+1}^{(i)}) \right) - \ln \left( \sum_{y} \exp \left( \sum_{t=1}^{n} w_t^\top \phi_t(x^{(i)}, y_t) + \sum_{t=1}^{n-1} w_{t,t+1}^\top \phi_{t,t+1}(x^{(i)}, y_t, y_{t+1}) \right) \right) / Z_{x^{(i)}(w)}.
\]

Having learned parameters \( \hat{w} \), given a new instance \( x \) one generally predicts a labeling \( \hat{y} \) according to

\[
\hat{y} \in \arg \max_y p(y | x; \hat{w}) = \arg \max_y \sum_{t=1}^{n} w_t^\top \phi_t(x, y_t) + \sum_{t=1}^{n-1} w_{t,t+1}^\top \phi_{t,t+1}(x, y_t, y_{t+1}).
\]

This maximization can generally be performed efficiently (with \( O(K^2 n) \) computations) using a dynamic programming approach similar to that used in the Viterbi algorithm for finding the most likely hidden state sequence in an HMM.

### 3.2 Structured Support Vector Machines

Just as CRFs generalize multiclass logistic regression to structured settings, structured support vector machines (StructSVMs) generalize multiclass SVMs to structured settings.
Here again one starts with an undirected graph on \( n \) nodes with clique set \( C \). However, rather than learning a probabilistic model, here for each (maximal) clique \( C \in C \) and labeling \( y_C \) of the nodes in \( C \), one learns a real-valued function \( f^C(y_C, x) \) that produces a local ‘clique score’ for \( y_C \) given the instance \( x \). Again, it is common to construct a feature vector \( \phi_C(x, y_C) \) for each clique \( C \), say of dimensionality \( d_C \), and to parametrize the functions \( f^C(y_C, x) \) as

\[
f^C(y_C, x) = w^\top_C \phi_C(x, y_C),
\]

where \( w_C \in \mathbb{R}^{d_C} \) is a parameter vector associated with clique \( C \). Typically, the feature vector \( \phi_C(x, y_C) \) is sparse and selects certain components corresponding to the labeling \( y_C \); the weight vector \( w_C \) also contains components corresponding to each labeling \( y_C \), similar to the setting described in Section 2.3 (effectively, each clique \( C \) is treated as a separate multiclass problem, with \( K^{|C|} \) classes).

The ‘score’ for a complete labeling \( y \) for an instance \( x \) is then obtained by adding the local clique scores associated with \( y \):

\[
f_y(x) = \sum_{C \in C} f^C(y_C, x) = \sum_{C \in C} w^\top_C \phi_C(x, y_C) = w^\top \phi(x, y),
\]

where again we have collected the parameter vectors \( w_C \in \mathbb{R}^{d_C} \) into a single parameter vector \( w \in \mathbb{R}^d \), with \( d = \sum_C d_C \), and the feature vectors \( \phi_C(x, y_C) \in \mathbb{R}^{d_C} \) into a single feature vector \( \phi(x, y) \in \mathbb{R}^d \).

Given training data \( S = ((x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})) \) as above, the StructSVM algorithm finds parameters \( w \) that maximize (a cost-sensitive version of) the margin between the score of the correct labeling \( y^{(i)} \) for instance \( x^{(i)} \) and the score of any other labeling \( y \). Specifically, let \( c_{y, y'} \) denote the cost of predicting a labeling \( y' \) when the true labeling is \( y \). Then the StructSVM algorithm (based on the Crammer-Singer multiclass SVM) aims to minimize the following objective as a function of the parameters \( w \):

\[
\sum_{i=1}^m \max \left( 0, \max_{y \neq y^{(i)}} \left( c_{y^{(i)}, y} - (w^\top \phi(x^{(i)}, y^{(i)}) - w^\top \phi(x^{(i)}, y^{(i)})) \right) \right) + \lambda \|w\|_2^2.
\]

While this is a convex function of the parameters \( w \), on introducing slack variables to rewrite the maximum in the above objective, in general, the primal problem has an exponential number of constraints (one for each possible labeling \( y \) for each instance \( x^{(i)} \), so a total of \( mK^n \)), and the dual problem correspondingly has an exponential number of dual variables. In some special cases (such as when the underlying graph is a chain and the feature vectors and loss function have a suitable structure), the dual problem can be rewritten in a compact form involving a polynomial number of variables and can be solved efficiently. An alternative approach involves solving the primal problem using a cutting-plane approach that makes explicit use of only a polynomial number of constraints; as a subroutine, this requires a certain optimization problem (related to MAP inference in the underlying graphical model) to be solved efficiently. Again, this can be done in some special cases (such as chain graphs for sequence labeling tasks).

For graphs where the above optimization problem can be solved efficiently, having learned parameters \( \hat{w} \), given a new instance \( x \) one predicts a labeling \( \hat{y} \) that has maximal score under \( \hat{w} \):

\[
\hat{y} \in \arg\max_y f_y(x)
= \arg\max_y \hat{w}^\top \phi(x, y).
\]

Again, depending on the structure and sizes of the cliques \( C \) and the forms of the feature vectors \( \phi_C(x, y_C) \), the above maximization may or may not be possible to perform efficiently. For sequence labeling tasks with chain graphs, as discussed above, the maximization can be done using \( O(K^2n) \) computations using a Viterbi-like approach; efficient algorithms also exist for certain other graph structures. Again, when exact maximization is difficult, one often settles for an approximate maximization of some sort (usually related to approximate MAP inference in the associated graphical model).
3.3 Example: Part-of-Speech Tagging

We briefly discuss an example to clarify the types of features often used in practice.

Consider a part-of-speech (PoS) tagging task, in which instances \( x \) are sentences and labels \( y \) are sequences of PoS tags (such as noun (N), verb (V), determiner (D), etc). For now, consider only sentences of a fixed length \( n \), say \( n = 5 \); we will relax this in the next section.

Suppose we model the label sequence \( y \) for any instance \( x \) as a chain graph:

\[
x: \quad \text{The dog chased the cat}
\]

There are \( n - 1 = 4 \) cliques here: \{1, 2\}, \{2, 3\}, \{3, 4\}, and \{4, 5\}. For each of these cliques \{\( t, t + 1 \}\}, we construct a joint instance-label feature vector \( \phi_{t,t+1}(x, y_t, y_{t+1}) \). For example, the feature vector might have an indicator component for each possible pair of words \( (u, u') \) and each possible pair of PoS tags \( (k, k') \), which for any \( (x, y) \) pair, takes value 1 if the words and tags at positions \( (t, t + 1) \) are equal to \( (u, u') \) and \( (k, k') \), respectively, and 0 otherwise:

\[
\phi_{t,t+1;u,u',k,k'}(x, y_t, y_{t+1}) = \mathbf{1}(x_t = u, x_{t+1} = u', y_t = k, y_{t+1} = k').
\]

If there are \( M \) words in the vocabulary and \( K \) PoS tags, then for each \( t \), the feature vector \( \phi_{t,t+1}(x, y_t, y_{t+1}) \) has \( K^2M^2 \) such components. (Note that we can also include features that involve any other parts of the sentence \( x \); they do not need to refer only to words at positions \( t \) and \( t + 1 \).) As an example, for the \( (x, y) \) pair

\[
\begin{align*}
\mathbf{y}: & \quad \mathbf{D} \quad \mathbf{N} \quad \mathbf{V} \quad \mathbf{D} \quad \mathbf{N} \\
\mathbf{x}: & \quad \text{The dog chased the cat}
\end{align*}
\]

the feature \( \phi_{1,2;\text{the},\text{dog},\mathbf{D},\mathbf{N}}(x, y_1, y_2) \) takes the value 1; the feature \( \phi_{1,2;\text{my},\text{cat},\mathbf{D},\mathbf{N}}(x, y_1, y_2) \) takes the value 0; and the feature \( \phi_{4,5;\text{the},\text{dog},\mathbf{D},\mathbf{N}}(x, y_1, y_2) \) takes the value 0.

As discussed in Section 3.1.1 above, in chain graphs, it is common to also include feature vectors \( \phi_t(x, y_t) \) for the individual nodes \( t \) \( (t = 1, \ldots, n) \). Here we might have an indicator component for each possible word \( u \) and PoS tag \( k \), which takes value 1 if the word and tag at position \( t \) are equal to \( u \) and \( k \), respectively, and 0 otherwise:

\[
\phi_{t;u,k}(x, y_t) = \mathbf{1}(x_t = u, y_t = k).
\]

Then for each \( t \), the feature vector \( \phi_t(x, y_t) \) has \( KM \) components. For example, for the sentence-label pair \( (x, y) \) given above, the features \( \phi_{1;\text{the},\mathbf{D}}(x, y_1) \), \( \phi_{2;\text{dog},\mathbf{N}}(x, y_2) \), and \( \phi_{4;\text{the},\mathbf{D}}(x, y_4) \) all take the value 1; the features \( \phi_{1;\text{the},\mathbf{N}}(x, y_1) \) and \( \phi_{2;\text{cat},\mathbf{N}}(x, y_2) \) take the value 0.

Under the above feature construction, the full joint feature vector \( \phi(x, y) \) has \((n-1)K^2M^2 + nKM\) components (of which only a small number are non-zero for any given sentence-label pair \( (x, y) \)). The parameter vector \( w \) would therefore also have \((n-1)K^2M^2 + nKM\) components, with \((n-1)\) component vectors \( w_{t,t+1} \in \mathbb{R}^{K^2M^2} \) corresponding to the 2-node cliques, and \( n \) component vectors \( w_t \in \mathbb{R}^{KM} \) corresponding to the 1-node cliques. The score for a PoS tag sequence \( y \) given a sentence \( x \) is then given by

\[
w^\top \phi(x, y) = \sum_{t=1}^{n-1} w_{t,t+1}^\top \phi_{t,t+1}(x, y_t, y_{t+1}) + \sum_{t=1}^{n} w_t^\top \phi_t(x, y_t).
\]
4 Structured Prediction: Variable-Size Structures

In practice, in many applications where structured prediction methods are needed, the structures can be of variable size. For example, in PoS tagging, sentences and therefore PoS tag sequences can have varying lengths; in natural language parsing, sentences and therefore parse trees can similarly have varying sizes; in image segmentation, images and therefore segment labelings can be of varying sizes.

In such applications, one usually identifies repeating substructures in the label graph, constructs similar features for these substructures, and then implements parameter-sharing across the substructures; in this way, the number of parameters does not depend on the number of times the substructure occurs in the graph. (This is similar to how stationary/homogeneous HMMs use the same transition and emission probability parameters at different time steps $t$, and can therefore be used to model sequences of varying lengths.)

For example, in the PoS tagging task discussed above, rather than learn a different set of parameters $w_{t,t+1}$ for each 2-node clique $\{t, t+1\}$ and a different set of parameters $w_t$ for each 1-node clique $\{t\}$, one simply learns a single set of parameters $w_{\text{pair}} \in \mathbb{R}^{KxM^2}$ for 2-node cliques and a single set of parameters $w_{\text{single}} \in \mathbb{R}^{KM}$ for 1-node cliques. Then the score for a PoS tag sequence $y$ given a sentence $x$, for any length $n'$, is given by

$$w^T \phi(x, y) = w_{\text{pair}}^{T} \sum_{t=1}^{n'-1} \phi_{t,t+1}(x, y_t, y_{t+1}) + w_{\text{single}}^{T} \sum_{t=1}^{n'} \phi_{t}(x, y_t),$$

where the feature vectors $\phi_{\text{pair}}(x, y)$ and $\phi_{\text{single}}(x, y)$ now contain counts of how many times particular word-tag combinations appear in the $(x, y)$ pair, regardless of the length $n'$.

5 Further Reading

For CRFs, see [4, 5, 1, 6]. For StructSVMs, see [7, 1, 2, 3].

References


