CMPT 825: Natural Language Processing **Spring 2008**

Lecture 10 — March 28 - April 7, 2008

Lecturer: Anoop Sarkar *Scribe: Louisa Harutyunyan*

10.1 An Introduction to Conditional Random Fields

This section is based [1].

Up until now in the course we have looked at log-linear models. We were interested in finding the probability given by

$$P(y|\overline{x}) = \frac{exp\left\{\sum_{\mathrm{k=1}}^{\mathrm{k}} \lambda_k f_k(\overline{x}, y)\right\}}{\sum_{\mathrm{y'}} exp\left\{\sum_{\mathrm{k=1}}^{\mathrm{k}} \lambda_k f_k(\overline{x}, y')\right\}}. \tag{10.1}$$

Now, we would like to know what happens when y itself is a sequence? (i.e want $P(\overline{y}|\overline{x})$). Traditionally, graphical models were used to represent the joint probability $P(\overline{y}, \overline{x})$. This however, can lead to difficulties. In the presence of rich local features in the relational data the distribution $P(\overline{x})$ needs to be modelled, which can include complex dependencies. A solution to this is to directly model the conditional distribution $P(\overline{y}|\overline{x})$. This is the approach taken by conditional random fields (CRF). A CRF is a conditional distribution $P(\overline{y}|\overline{x})$ with an associated graphical structure.

10.1.1 Graphical Models

Consider a probability distribution over the two sets of random variables X and Y. X is the set of input variables and Y is the set of output variables that we wish to predict. Every variable from these sets takes outcomes from their respective sets in a discrete manner. An assignment to X is denoted by x (i.e. $X = x$) and an assignment for a set $A \subset X$ is denoted by x_A. The same is done for the set Y. For the assignments $X_1 = x_1$, $X_2 = x_2$, the respective subset A and x_a are given by $A = \{X_1, X_2\}$ and $x_A = \langle x_1, x_2 \rangle$. We define an indicator function of x by $\delta(x, x')$. $\delta(x, x')$ takes the following values:

$$\delta(x, x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{otherwise.} \end{cases}$$

Undirected Graphical Model

A graphical model is a family of probability distributions that factorize according to an underlying graph. For a collection of the defined subsets A, an *undirected graphical model* is defined as the set of all distributions that can be written as

$$P(x, y) = \frac{1}{Z} \prod_{\text{A}} \Psi_A(x_A, y_A). \tag{10.2}$$

(x, y) variables are decomposed according to a grouping that is local. To represent the distribution over a large number of variables, the product of these local decompositions are used.

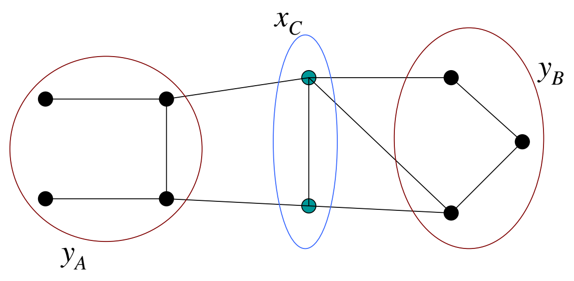

Figure 10.1. x_C is the observed data. Everything in y_A is independent of y_B given x_C .

Recall that until now we have been looking at directed models such as HMMs. With undirected graphical models we can model things similar to what is shown in Figure 10.1. It can be seen in Figure 10.1 that $y_A \perp y_B | x_C$ (i.e. y_A is connected to y_B through x_C). $y_A \perp y_B | x_C$ is easily defined in an undirected graphical model. However, it is much more difficult to do so in a directed model. Note that undirected graphical models are not conditional while HMMs (i.e. directed models) are.

A *model* is a family of distributions (i.e. any distribution that can be written as Equation 10.2). A particular single member of this family is referred to as a *random field*. Let $V = X \cup Y$. Equation 10.2 is defined for any choice of factors $F = \{\Psi\}$, where $\Psi_A : V^n \to \mathbb{R}^+$. Factors are used to make the structures simpler.

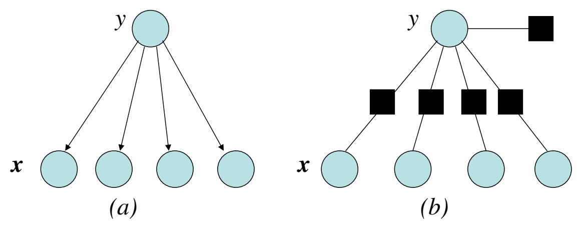

Figure 10.2. (a) Naive Bayes classifier as a directed model; (b) factor graph.

The constant Z from Equation 10.2 is a normalization factor defined as

$$Z = \sum_{\text{x, y}} \prod_{A} \Psi_A(x_A, y_A). \tag{10.3}$$

It ensures that the distribution sums to one. Z is a function of the set F of factors. Graphically, the factorization of Equation 10.2 is represented by a *factor graph*. A variable node $v_s \in V$ is connected to a factor node $\Psi_A \in F$ if v_s is an argument to Ψ_A. An example of a factor graph is given in Figure 10.2(b). The circles represent variable nodes and boxes represent factor nodes.

To see the relationship between the above mentioned model and the *log*-linear model, assume that each local function $\Psi_A(x_A, y_A)$ has the following form

$$\Psi_A(x_A, y_A) = exp\left\{\sum_{\text{k}} \theta_{Ak} f_{Ak}(x_A, y_A)\right\}, \tag{10.4}$$

for some real-valued parameter vector θ_A and for some set of feature functions $\{f_{Ak}\}$ defined over x_A, y_A. Taking the *log* on the left of Equation 10.4 will yield a *log*-linear model on the right.

Directed Graphical Models

A *directed graphical model* is based on a directed graph $G = (V, E)$. It is sometimes called Bayesian network. A directed model is a family of distributions that can be written as:

$$P(\overline{y}|\overline{x}) = \prod_{\mathrm{v}\in\mathrm{V}} P(v|\pi(v)), \tag{10.5}$$

where $\pi(v)$ are the parents of v in G. An example of Bayesian network can be seen in Figure 10.2(a), where each of the nodes of $\overline{x}$ is a condition on its parent y. Graphical models where outputs topologically precede the inputs (i.e. no $x \in X$ can be a parent of an output $y \in Y$) are referred to *generative models*.

It can be seen that the directed graphical model is similar to the *naive Bayes model*. Naive Bayes model is based on the joint probability that takes the following form:

$$P(y|\overline{x}) = p(y)\prod_{\mathrm{k}=1}^{K} p(x_k|y). \tag{10.6}$$

This model can be written as a factor graph by defining a factor $\Psi(y) = p(y)$ and a factor $\Psi_A(y, x_k)$ for each feature x_k. Both the directed model and the factor graph are shown in Figure 10.2.

By looking at Equation 10.6 in a slightly different way yields another graphical model known as *logistic regression*. This model is motivated by the assumption that the log probability, $log(y|\overline{x})$, of each class is a linear function of $\overline{x}$. This model is given by the following conditional distribution

$$P(y|\overline{x}) = \frac{1}{Z(\overline{x})} exp\left\{\lambda_y + \sum_{\mathrm{j}=1}^{k} \lambda_{y,j}x_j\right\}, \tag{10.7}$$

where $Z(\overline{x}) = \sum_{\mathrm{y}} exp\{\lambda_y + \sum_{\mathrm{j}=1}^{k} \lambda_{y,j}x_j\}$ is a normalizing constant and λ_y is a bias weight that acts like log $p(y)$ in Naive Bayes. Equation 10.7 uses one vector per class. To have a single set of weights shared among all classes a set of feature functions are defined the following way:

$$\begin{array}{ll} f_{y',j}(y,\overline{x}) = \delta(y,y')x_j & \text{for the feature weights} \\ f_{y'}(y,\overline{x}) = \delta(y,y') & \text{for the bias weights} \end{array}$$

where $\delta(x, x')$ is given by

$$\delta(x,x') = \begin{cases} 1 & \text{if } y = y' \\ 0 & \text{otherwise.} \end{cases}$$

Now, instead of $f_{y',j}$ we can write f_k to index each feature function $f_{y',j}$. Similarly, we can use λ_k to index its corresponding weight $\lambda_{y',j}$. The logistic regression now becomes

$$P(y|\overline{x}) = \frac{1}{Z(\overline{x})} exp \left\{ \sum_{\mathrm{k=1}}^{K} \lambda_k f_k(y, \overline{x}) \right\}. \tag{10.8}$$

From Equation 10.7 we see that

$$\lambda_y \to \log\, p(y)$$
$$\lambda_{y,j} \to \log\, p(x_j|y).$$

The notation switches from 2-dimensions into 1-dimension.

It is noteworthy to mention that there is an important difference between naive Bayes model and logistic regression. Naive Bayes model is generative (i.e. based on the model of joint distribution $P(y, \overline{x})$). Logistic regression is discriminative (i.e. based on a model of the conditional distribution $P(y|\overline{x})$). We can define $P(y, \overline{x})$ in a similar fashion as we did in obtaining Equation 10.8. The result is

$$P(y, \overline{x}) = \frac{exp \left\{ \sum_{\mathrm{k=1}}^{\mathrm{K}} \lambda_k f_k(\overline{x}, y) \right\}}{\sum_{\overline{x},y} exp \left\{ \sum_{\mathrm{k=1}}^{\mathrm{K}} \lambda_k f_k(\overline{x}, y) \right\}}. \tag{10.9}$$

From the generative interpretation of Equation 10.9 Naive Bayes model defines the same family of distributions as the logistic regression model. Naive Bayes model and logistic regression are called a *generative-discriminative pair*. The $argmax(y)$ in 10.8 will give the same answer as the $argmax(y, \overline{x})$. We can take a generative model and convert it to a discriminative model.

Suppose we have a generative model P with parameters θ. Using Bayes rule it is in the following form

$$P(\overline{y}, \overline{x}; \theta) = P(\overline{x}; \theta) P(\overline{y}|\overline{x}; \theta), \tag{10.10}$$

where $P(\overline{x}; \theta) = \sum_y P(\overline{y}, \overline{x}; \theta)$ and $P(\overline{y}|\overline{x}; \theta) = \frac{P(\overline{y}, \overline{x}; \theta)}{P(\overline{x}; \theta)}$. Now, compare this generative model with a discriminative model over the same family of joint distributions. To do this, we need to define a prior, which could have arisen from P with some parameter setting θ' and combine it with the conditional distribution that could have also arisen from P. This will result in the following distribution

$$P(\overline{y},\overline{x},\theta) = P(\overline{x};\theta')P(\overline{y}|\overline{x};\theta). \tag{10.11}$$

When we compare Equation 10.10 and Equation 10.11 we can see that the discriminative model has more freedom since it does not require that $\theta' = \theta$. The reason for this is the parameters θ in Equation 10.10 are used for the input distribution and the conditional, a good set of parameters must represent both well. The Equations 10.10 and 10.11 are the same, however, the assumptions for each are different.

10.1.2 From HMMs to CRFs

In the previous section the relationship between naive Bayes model and logistic regression was discussed. This relationship is similar to that of HMMs and CRFs. Consider the conditional distribution $P(\overline{y}|\overline{x})$ that follows from the joint distribution $P(\overline{y},\overline{x})$ of an HMM given in Equation 10.12

$$P(\overline{y},\overline{x}) = \prod_{\text{t}=1}^{T} P(y_t|y_{t-1})P(x_t|y_t). \tag{10.12}$$

We rewrite Equation 10.12 as

$$P(\overline{y},\overline{x}) = \frac{1}{Z} exp\left\{\sum_{\text{t}}\sum_{\text{i, j}} \lambda_{ij}\delta(y_t = i)\delta(y_{t-1} = j) + \sum_{\text{t}}\sum_{\text{i}}\sum_{\text{o}} \mu_{oi}\delta(y_t = i)\delta(x_t = o)\right\}, \tag{10.13}$$

where $\theta = \{\lambda_{ij}, \mu_{oi}\}$ are the parameters of the distribution. Every HMM can be written in this form by setting $\lambda_{ij} = \log P(y_t = i|y_{t-1} = j)$ and $\mu_{oi} = \log P(x_t = o|y_t = i)$.

Equation 10.13 can be written more compactly by introducing the concept of feature functions. Recall that this same trick was used to obtain Equation 10.8. The feature functions are given by

$$f_k(y_t, y_{t-1}, x_t) = \begin{cases} f_{ij}(y, y', x) = \delta(y = i)\delta(y' = j) & \text{for each transition (i, j)} \\ f_{io}(y, y', x) = \delta(y = i)\delta(x = o) & \text{for each state-observation pair (i, o).} \end{cases}$$

Equation 10.13 can now be written as

$$P(\overline{y}, \overline{x}) = \frac{1}{Z} exp\left\{\sum_{\mathrm{k=1}}^{\mathrm{K}} \lambda_k f_k(y_t, y_{t-1}, x_t)\right\}, \tag{10.14}$$

where Z sums over $\overline{x}, \overline{y}$. Equation 10.14 defines the same family of distributions as that given in Equation 10.13 and therefore, as the original HMM given in Equation 10.12. Equation 10.14 makes sense only if we have an underlying graph. The underlying graph is similar to that of shown in Figure 10.3(b). The graph is decomposed into simple parts that the feature functions in Equation 10.14 go through.

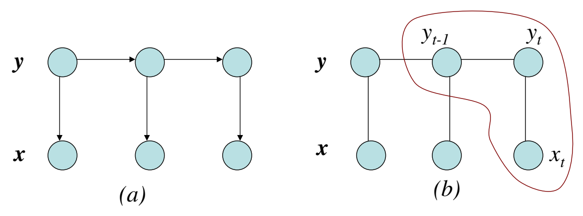

Figure 10.3. (a) Representation of an HMM; (b) representation of a CRF.

The conditional distribution $P(\overline{y}|\overline{x})$ that results from Equation 10.14 is

$$P(\overline{y}|\overline{x}) = \frac{1}{Z(\overline{x})} exp\left\{\sum_{\mathrm{k=1}}^{\mathrm{K}} \lambda_k f_k(y_t, y_{t-1}, x_t)\right\}, \tag{10.15}$$

where $Z(\overline{x}) = \sum_{\overline{y}} exp\left\{\sum_{\mathrm{k=1}}^{\mathrm{K}} \lambda_k f_k(y_t, y_{t-1}, x_t)\right\}$ (i.e. $Z(\overline{x})$ sums over all possible $\overline{y}$ for a given x). Equation 10.15 is a CRF, the discriminative equivalent to an HMM. Instead of representing x_t as one symbol we can represent it as a vector: $\overline{x}_t = \langle x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(n)}\rangle$. The size of $\overline{x}_t$ is the number of features that we have.

The graphical representation and relationship of all the models discussed thus far are shown in Figure 10.4.

10.1.3 Parameter Estimation

This section discusses how to estimate the parameters $\overline{\lambda}$ of a CRF. Parameter estimation is performed by penalized maximum likelihood. For a conditional

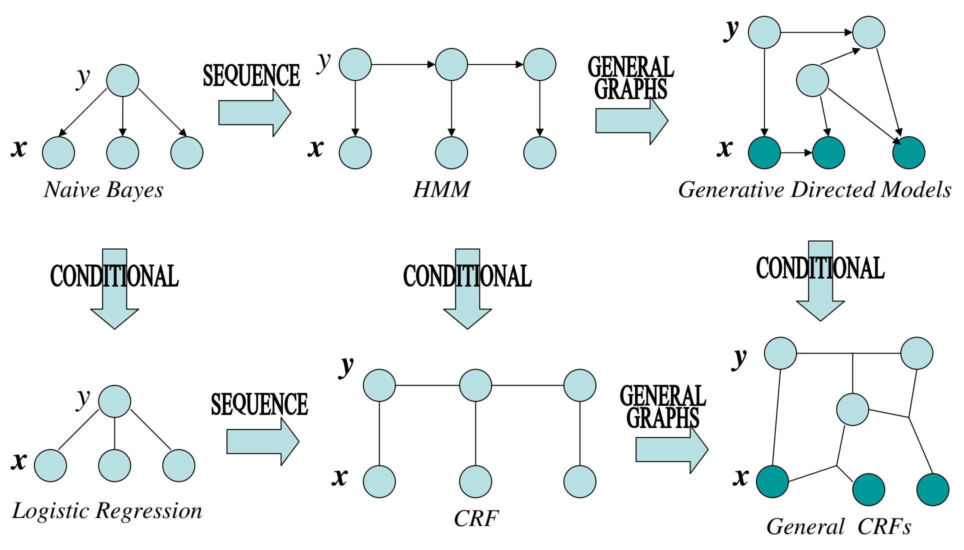

Figure 10.4. Diagraph of the relationship between naive Bayes, logistic regression, HMMs, CRFs, generative models and general CRFs
.

distribution and training data consisting of $\overline{x}$, a sequence of inputs, and $\overline{y}$, a sequence of desired predictions, the following log likelihood is used

$$L(\overline{\lambda}) = \sum_{\mathrm{i=1}}^{\mathrm{M}} \log P(\overline{y}|\overline{x}). \tag{10.16}$$

After substituting the CRF model into the likelihood, we obtain the following expression

$$L(\overline{\lambda}) = \sum_{\mathrm{i=1}}^{\mathrm{M}} \overline{\lambda}\ \overline{f}(y_t, y_{t-1}, \overline{x}_t) - \sum_{\mathrm{i=1}}^{\mathrm{M}} \log z - f(\overline{\lambda}). \tag{10.17}$$

Over large number of parameters to avoid overfitting, *regularization* is used to assign penalty on weight vectors whose norm is too large. Regularization in Equation 10.17 is given by the last term $f(\overline{\lambda})$.

10.1.4 Data Classification

This section discusses how to classify given datasets. Given two separate sets of input x_1 and x_2, as shown in Figure 10.5, we would like to find a discriminative boundary between the datasets. In Figure 10.5, the wanted discriminative boundary separating one class from the other is given by the line f.

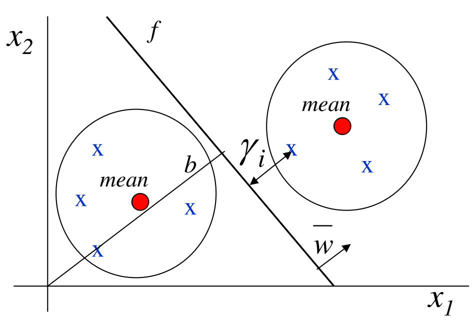

Figure 10.5. Classification of datasets. f is the boundary between the two sets.

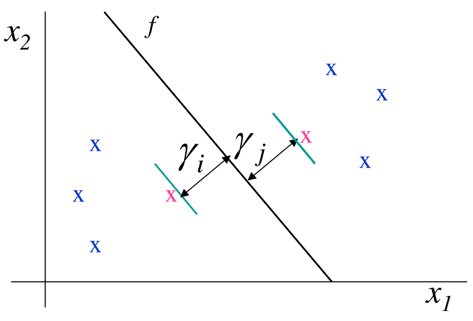

Figure 10.6. Classification of datasets with maximized minimum error measure γ_i.

We define f as $f : x \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^n$. Assume that f is linear $\Rightarrow$

$$f(\overline{x}) = \overline{w}\ \overline{x} + b = \sum_{\mathrm{i=1}}^{\mathrm{d}} w_i x_i + b.$$

We need to find f and use it to make a decision on the discriminative boundary between the datasets. How to use f to make a decision is defined by

$$h(\overline{x}) = sign(f(\overline{x})).$$

We define an error measure for the best fitting line. This error measure is given by

$$\gamma_i = y_i(\overline{w}\ \overline{x}_i + b), \tag{10.18}$$

where $y_i \in \{1, -1\}$. We would like $\gamma_i > 0\ \forall i$. For a positive value of γ determines how right the line classifies the data. However, if we are confident in the choice of f and make a mistake, the penalty in such cases is much higher. We can get many lines such that $\gamma_i > 0$. Out of all these lines we find a line such the minimum γ line is maximized (i.e. the minimum value of γ is the maximum it can be. Refer to Figure 10.6). Based on this maximized margin we can give a bound on the classification of unseen data. The larger the margin, better it does on unseen data.

10.2 Maximum Entropy Model for Part-of-Speech Tagging

This section is based on [2].

The paper introduces a statistical model, which trains a corpus annotated with POS tags. The tags are assigned to previously unseen text. The POS tags are predicted by simultaneously using many contextual features. This model can be classified as a Maximum Entropy model. The given model as opposed to the CRF model is given as follows

$$P(y_t|\overline{x}) = \frac{exp\{\sum_{\mathrm{k}} \lambda_k f_k(\langle x, y_{t-2}, y_{t-1}\rangle)\}}{\sum_{\mathrm{y_t}} exp\{\sum_{\mathrm{k}} \lambda_k f_k(\overline{x}, y_t)\}} = \frac{1}{Z}\lambda_o \prod_{\mathrm{k}} \alpha_k f_k(h, y_k) \qquad (10.19)$$

where h = history = $\langle \overline{x}, y_{t-2}, y_{t-1}\rangle$. So for a sequence of words and tags as training data, h_i is the history available to predict each tag i. The features used for this model are given in Table 10.1.

The features generated for tagging unknown words rely on the distribution that *rare* words in the training set are similar to unknown words in the test data, in terms of how their spellings help predict their tags. Words are considered to be *rare* if they occur less than 5 times in the training set. The rare word features in Table 10.1, which look at word spellings, apply to *rare* words, as well as unknown words in the test data. Each of the features of Table 10.1 is connected to a tag.

Experiments use $40K$ sentences in the training data set with no unknown words. About $5.5K$ sentences were used for the Test data set with about $3.5K$ unknown words. The model performs at 96.43% on the Development Set. The current best known accuracy is 97.3%. This is an improvement because the model used is exactly the same except the next and previous tags are included.

Condition	Features
w_i is not rare	$w_i = X$
w_i is rare	X is prefix of w_i, $\lvert X\rvert \leq 4$
	X is suffix of w_i, $\lvert X\rvert \leq 4$
	w_i contains number
	w_i contains uppercase character
	w_i contains hyphen
$\forall w_i$	$t_{i-1} = X$
	$t_{i-2}t_{i-1} = XY$
	$w_{i-1} = X$
	$w_{i-2} = X$
	$w_{i+1} = X$
	$w_{i+2} = X$

Table 10.1. Features on the current history h_i.

10.3 Maximum Entropy Markov Models

This section is based on [3].

This paper presents a new Markovian sequence model. The model allows observations to be represented as arbitrary overlapping features, such as word capitalization, formatting, part-of-speech. It defines the conditional probability of state sequences given observation sequences. To do this a maximum entropy framework is used to fit a set of exponential models that represent the probability of a state given an observation and the previous state.

Previous models have two problems. Frist, they do not have a richer representation of observations in terms of overlapping features. Second, the HMM parameters are set to maximize the likelihood of the observation sequence. However, in most cases we would like to predict the state sequence given the observation sequence. The traditional approach uses a generative joint model to solve a conditional problem where the observations are given.

The introduced Markovian sequence model, called *maximum entropy Markov model* (MEMM) replaces the transition and observations functions with a single function to provide the probability of current state given a pervious state and a current observation, denoted by $P(y_t|y_{t-1}, x_t)$. For M states in the model, construct a probability $P_{y_{t-1}}(y_t|x_t)$ for each state given the current input. This gives rise to the following expression

$$P_{y_{t-1}}(y_t = i | x_t = o) = \frac{1}{Z(y_t)} exp \left\{ \sum_{\text{k}} \lambda_k f_k(y_t, x_t) \right\}. \tag{10.20}$$

Experiments are done on extracting the question-answer pairs in lists of frequently asked question (FAQs). Collection of 38 files are used. All documents in the collection have the same structure. Each contain a header, question/answer pairs, and a tail. Header may include things such as table of contents and tails are generally includes items such as copyright notices and acknowledgments. For this experiment, 24 Boolean features of lines are defined shown in Table 10.2.

begins-with-number	contains-question-mark
begins-with-ordinal	contains-question-word
begins-with-punctuation	ends-with-question-mark
begins-with-question-word	first-alpha-is-capitalized
begins-with-subject	indented
blank	indented-1-to-4
contains-alphanum	indented-5-to-10
contains-bracketed-number	more-than-one-third-space
contains-http	only-punctuation
contains-non-space	prev-is-blank
contains-number	prev-begins-with-ordinal
contains-pipe	shorter-than-30

Table 10.2. Line-based features used in experiments.

The statistical dependencies between features was not controlled in these experiments. There are some features that are mutually disjoint. Note that the usefulness of particular feature, such as `indented`, depends on the formatting of a particular FAQ. Each group of documents that belong to the same FAQ are treated as a separate dataset. In a given group, the model is trained on one document and tested on the remaining documents in the group.

By the training the model on one document and testing it on the remaining documents in the group gives rise to the label bias problem. Take the example in Figure 10.7 where all paths go through y_{t-1}. The route taken will go through the state `IN` since it has the highest probability based on the trained data for the word *that*. However, this in many cases may not be

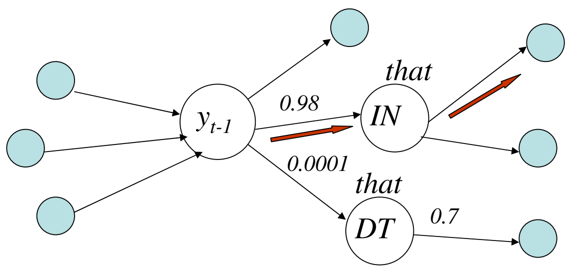

Figure 10.7. Model MEMM inaccuracy for the word *that*.

accurate, since in many cases the probability that the word *that* would be tagged with `DT` is higher. This model makes a choice for a transition based on the expense of the accuracy of a given sentence. It is defined by state and forces the model to be very confident when it shouldn't be.

10.4 Perceptron Algorithms

This section is based on [4].

This section discusses parameter estimation algorithms, alternatives to CRFs. These algorithms are based on the Perceptron algorithm. The Perceptron algorithm is an online algorithm looking for a classifier which essentially learns a line. The line is given by $f(\overline{x}) = \overline{w}\ \overline{x} + b$ where $\overline{w} \in \mathbb{R}^d$. Each input example is split into these d features. The Perceptron algorithm is given in Algorithm 1 where η is the learning rate.

Theorem 10.1. *(Novikoff) Z is a separable* non-trivial *training set. Suppose there exists $\overline{w}_{opt}$ such that $\|w_{opt}\| = 1$ and $y_i(\overline{w}_{opt}\ \overline{x}_i + b_{opt}) \geq \gamma$ for $1 \leq i \leq l$. Then the number of mistakes made by Algorithm 1 is $\left(\frac{2R}{\gamma}\right)^2$.*

Recall γ from Data Classification. The bigger γ is, the larger is the margin that separates the data. For a large γ the number of mistakes decreases. Mistakes are dependent on the distance between the vectors as seen in Figure 10.6. A strong assumption made in Theorem 10.1 is $y_i(\overline{w}_{opt}\ \overline{x}_i + b_{opt}) \geq \gamma$ (i.e. the number of mistakes is related to the distance of the margin) and $\|w_{opt}\| = 1$ helps show that $\left(\frac{2R}{\gamma}\right)^2$ is true.

Algorithm 1 Perceptron Algorithm

```
Initialize: $\overline{w}_o$; $\overline{b}_o$; $k = 0$
Initialize: $R = max_{1 \le i \le l} \|x_i\|$
while no mistakes do
  for all $i = 1 \dots l$ do
    if $y_i(\overline{w}_k\ \overline{x}_i + b_k) \le 0$ then
      $\overline{w}_{k+1} = \overline{w}_k + \eta\ y_i\ \overline{x}_i$
      $b_{k+1} = b_k + \eta\ y_i\ R^2$
      $k = k + 1$
    end if
  end for
end while
```

We could have a case where the given data is not clean (i.e. not separable). In such a case, for example, when the data is not separable in 2-dimensions, we map it to a higher dimension. The reason for doing this is that the data may become separable in a higher dimension.

The approach taken here is different. In the case of inseparable data we define the notion of *Slack Variables* given in Algorithm 2. The same proof that Novikoff gave for perceptron can be extended to that of slack variables. For the inseparable case we have the following theorem

Theorem 10.2. *(Freund & Schapice) Define $D = \sqrt{\sum_{i=1}^{l} \xi_i^2}$. Then the amount by which the training data fails to meet the margin γ is at most $\left(\frac{2(R+D)}{\gamma}\right)^2$.*

Recall for the separable case from Theorem 10.1 this upper bound was $\left(\frac{2R}{\gamma}\right)^2$.

As mentioned above Perceptron algorithm learns a line given by $F(\overline{x}) = sign(\overline{w}\ \overline{x} + b)$. This is known as the primal learning of w directly. We obtain the dual of this by substituting into $F(\overline{x})$ $\overline{w} = \sum_{\mathrm{i}=1}^{\mathrm{l}} \alpha_i y_i \overline{x}_i$. α_i are the local weights for each example and are called the Dual variables. The examples of importance are those with $\alpha_i > 0$. We would like to extend this so the output is a sequence. So $F(\overline{x})$ becomes

$$F(\overline{x}) = argmax_{\overline{y} \in GEN(\overline{x})} \Phi(\overline{x}, \overline{y}) \overline{w}. \tag{10.21}$$

Algorithm 2 Slack Variables

```
margin $\gamma > 0$
margin slack $\xi$
for do
    each example $(\overline{x}_i, y_i)$
    if $\xi_i > \gamma\ \overline{x}_i$ then
        $\overline{x}_i$ is misclassified by $(\overline{w}, b)$
    end if
    $\xi\left((\overline{x}_i, y_i), (\overline{w}, b), \gamma\right) = max(0, \gamma - y_i(\overline{w}\ \overline{x}_i + b))$
end for
```

$\Phi(\overline{x}, \overline{y})$ represents a feature over combination of input and output. GEN $(\overline{x})$ takes x and gives a set of candidate y's. A variation of Viterbi could be used for this. Here, we do not limit to local features but expand to global features. Global features are defined as the sum of local features. Functions of local representation are given by

$$\Phi_s(\overline{w}, \overline{t}) = \sum_{\mathrm{i=1}}^{\mathrm{n}} \Phi_s(h_i, t_i), \tag{10.22}$$

where $h_i = \langle t_{i-1}, t_{i-2}, \overline{w}, i\rangle$. The training algorithm for the training data $(\overline{x}, \overline{y}) \ldots (\overline{x}_m, \overline{y}_m)$ is given by Algorithm 3.

Algorithm 3 Training Algorithm

```
for $t = 1 \ldots T$ do
    for $i = 1 \ldots m$ do
        $y'_i = F(\overline{x}_i)$
        if $y_i \neq y'_i$ then
            $\overline{w} = \overline{w} + \Phi(\overline{x}_i, \overline{y}_i) - \Phi(\overline{x}_i, \overline{y'}_i)$
        end if
    end for
end for
```

Taking the perceptron algorithm given in Algorithm 1 and replacing the inner loop with

```
if $y_i(\overline{w}_k\ \overline{x}_i + b_k) > 0$ then
    $c_k += 1$
else
```

$c_{k+1} = 1$
will keep all of the weight vectors as well as how good the weight vectors are. This modification to the perceptron algorithm is known as the Voted Perceptron. The original perceptron algorithm only kept the last weight vector. So, instead of making a prediction like $F(\overline{x})$, it makes one like $\sum_{\mathrm{i=1}}^{\mathrm{k}} c_i((\overline{w}_i\ \overline{x}) + b_i)$. However, this has a drawback. It is expensive to implement in terms of memory. In practice, the average perceptron is used instead.

Bibliography

[1] C. Sutton and A. McCallum, *An Introduction to Conditional Random Fields for Relational Learning*. In Lise Getoor and Ben Taskar, editors. Introduction to Statistical Relational Learning. MIT Press. 2007.

[2] A. Ratnaparkhi, *A Maximum Entropy Model for Part-Of-Speech Tagging*. Proc. Conference on Empirical Methods in Natural Language Processing. EMNLP 1996.

[3] A. McCallum, D. Freitag and F. Pereira, *Maximum Entropy Markov Models for Information Extraction and Segmentation*. Proc. 17th ICML. 2000.

[4] M. Collins, *Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms*. EMNLP 2002.