Chapter 1

Fundamental Statistical Techniques

The statistical approach to natural language processing (NLP) has become more and more important in recent years. This chapter gives an overview of some fundamental statistical techniques that have been widely used in different NLP tasks. Methods for statistical NLP mainly come from machine learning, which is a scientific discipline concerned with learning from data. That is, to extract information, discover patterns, predict missing information based on observed information, or more generally construct probabilistic models of the data. Machine learning techniques covered in this chapter can be divided into two types: supervised and unsupervised.

Supervised learning is mainly concerned with predicting missing information based on observed information. For example, predicting part-of-speech based on sentences. It employs statistical methods to construct a prediction rule from labeled training data. Supervised learning algorithms discussed in this chapter include Naive Bayes, support vector machines, and logistic regression. The goal of unsupervised learning is to group data into clusters. The main statistical techniques are mixture models and the expectation maximization algorithm. This chapter will also cover methods used in sequence analysis, such as hidden Markov model (HMM), conditional random field, and the Viterbi decoding algorithm.

1.1 Binary Linear Classification

The goal of binary classification is to predict an unobserved binary label $y \in \{-1, 1\}$, based on an observed input vector $x \in R^d$. A classifier $h(x)$ maps $x \in R^d$ to $\{-1, 1\}$. If it agrees with the label y, the error is zero. If it does not agree with label y, we suffer a loss of one. That is, the classification

error is defined as

$$\text{err}(h(x), y) = I(h(x) \neq y) = \begin{cases} 1 & h(x) \neq y \\ -1 & \text{otherwise} \end{cases},$$

where $I(\cdot)$ is the set indicator function.

A commonly used method for binary classification is to learn a real-valued scoring function $f(x) : R^d \to R$ that induces a classification rule

$$h(x) = \begin{cases} 1 & f(x) > 0 \\ -1 & \text{otherwise} \end{cases}. \tag{1.1}$$

A commonly used scoring function is linear: $f(x) = w^T x + b$, where $w \in R^d$ and $b \in R$. A binary classification method using linear scoring function is called a linear classifier.

In supervised learning, the classifier $h(x)$, or its scoring function $f(x)$, is learned from a set of labeled examples

$$\{(x_1, y_1), \ldots, (x_n, y_n)\},$$

referred to as training data. Its performance (average classification error) should be evaluated on a separate set of labeled data called test data.

A procedure that constructs a scoring function $f(x)$ from the training data is called a learning algorithm. For example, a standard learning algorithm for linear classification is linear least squares method, which finds a weight vector $\hat{w} \in R^d$ and bias $\hat{b} \in R$ for a linear scoring function $f(x) = \hat{w}^T x + \hat{b}$ by minimizing the squared error on the training set:

$$[\hat{w}, \hat{b}] = \arg\min_{w,b} \sum_{i=1}^{n} (w^T x_i + b - y_i)^2. \tag{1.2}$$

Using linear algebra, we may write the solution of this formula in closed form as

$$\begin{aligned} \hat{w} &= \left[\sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T \right]^{-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}), \\ \hat{b} &= \bar{y} - \hat{w}^T \bar{x}, \end{aligned}$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \qquad \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i.$$

One problem of the above formulation is that the matrix $\sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$ may be singular or ill-conditioned (this occurs, for example, when n is less than the dimension of x). A standard remedy is to use the ridge regression

method (Hoerl and Kennard 1970) that adds a regularization term $\lambda w^T w$ to (1.2). For convenient, we set $b = 0$:

$$\hat{w} = \arg\min_w \left[\frac{1}{n} \sum_{i=1}^{n} (w^T x_i y_i - 1)^2 + \lambda w^T w \right], \tag{1.3}$$

where $\lambda > 0$ is an appropriately chosen regularization parameter. The solution is given by

$$\hat{w} = \left(\sum_{i=1}^{n} x_i x_i^T + \lambda n I \right)^{-1} \left(\sum_{i=1}^{n} x_i y_i \right),$$

where I denotes the identity matrix. This method solves the ill-conditioning problem because $\sum_{i=1}^{n} x_i x_i^T + \lambda n I$ is always non-singular.

Note that taking $b = 0$ in (1.3) does not make the resulting scoring function $f(x) = \hat{w}^T x$ less general. To see this, one can embed all the data into a space with one more dimension with some constant A (normally, one takes $A = 1$). In this conversion, each vector $x_i = [x_{i,1}, \ldots, x_{i,d}]$ in the original space becomes the vector $x' = [x_{i,1}, \ldots, x_{i,d}, A]$ in the larger space. Therefore the linear classifier $w^T x + b = w'^T x'$, where $w' = [w, b]$ is a weight vector in $d + 1$ dimensional space. Due to this simple change of representation, the linear scoring function with b in the original space is equivalent to a linear scoring function without b in the larger space.

The introduction of the regularization term $\lambda w^T w$ in (1.3) makes the solution more stable. That is, a small perturbation of the observation does not significantly change the solution. This is a desirable property because the observations (both x_i and y_i) often contain noise. However, λ introduces a bias into the system because it pulls the solution $\hat{w}$ towards zero. When $\lambda \to \infty$, $\hat{w} \to 0$. Therefore it is necessary to balance the desirable stabilization effect and the undesirable bias effect, so that the optimal trade-off can be achieved. Figure 1.1 illustrates the training error versus test error when λ changes. As λ increases, due to the bias effect, the training error always increases. However, since the solution becomes more robust to noise as λ increases, the test error will decrease first. This is because the benefit of a more stable solution is larger than the bias effect. After the optimal trade-off (the lowest test error point) is achieved, the test error becomes larger when λ increases. This is because the benefit of more stability is smaller than the increased bias.

In practice, the optimal λ can be selected using cross-validation, where we randomly split the training data into two parts: a training part and a validation part. We use only the first (training) part to compute $\hat{w}$ with different λ, and then estimate its performance on the validation part. The λ with the smallest validation error is then chosen as the optimal regularization parameter.

The decision rule (1.1) for a linear classifier $f(x) = w^T x + b$ is defined by a decision boundary $\{x : w^T x + b = 0\}$: on one side of this hyperplane, we predict $h(x) = 1$, and on the other side we predict $h(x) = -1$. If the hyperplane

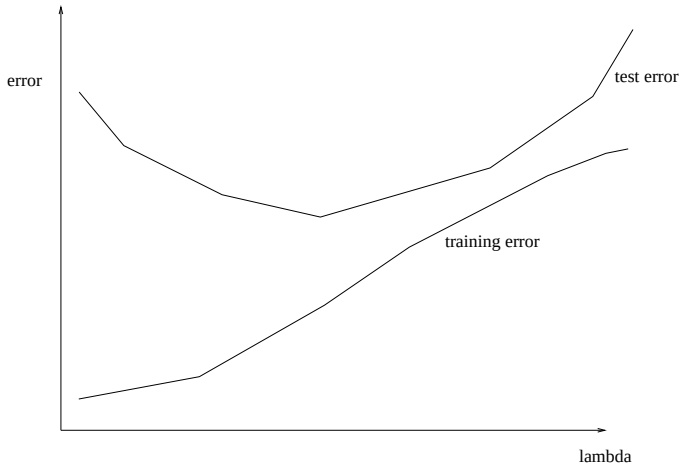

FIGURE 1.1: The Effect of Regularization

completely separates the positive data from the negative data without error, we call it a separating hyperplane. If the data is linearly separable, then there can be more than one possible separating hyperplanes, as shown in Figure 1.2. A natural question is: what is a better separating hyperplane? One possible measure to define the quality of a separating hyperplane is through the concept of margin, which is the distance of the nearest training example to the linear decision boundary. A separating hyperplane with a larger margin is more robust to noise because training data can still be separated after a small perturbation. In Figure 1.2, the boundary represented by the solid line has a larger margin than the boundary represented by the dashed line, and thus it is the preferred classifier.

The idea of finding an optimal separating hyperplane with largest margin leads to another popular linear classification method called support vector machine (SVM) (Cortes and Vapnik 1995; Joachims 1998). If the training data is linearly separable, the method finds a separating hyperplane with largest margin defined as

$$\min_{i=1,\ldots,n} (w^T x_i + b) y_i / \|w\|_2.$$

The equivalent formulation is to minimize $\|w\|_2$ under the constraint $\min_i (w^T x_i + b) y_i \geq 1$. That is, the optimal hyperplane is the solution to

$$[\hat{w}, \hat{b}] = \arg\min_{w,b} \|w\|_2^2$$
$$\text{subject to } (w^T x_i + b) y_i \geq 1 \ (i = 1, \ldots, n).$$

For training data that is not linearly separable, the idea of margin maximization cannot be directly applied. Instead, one considers the so-called

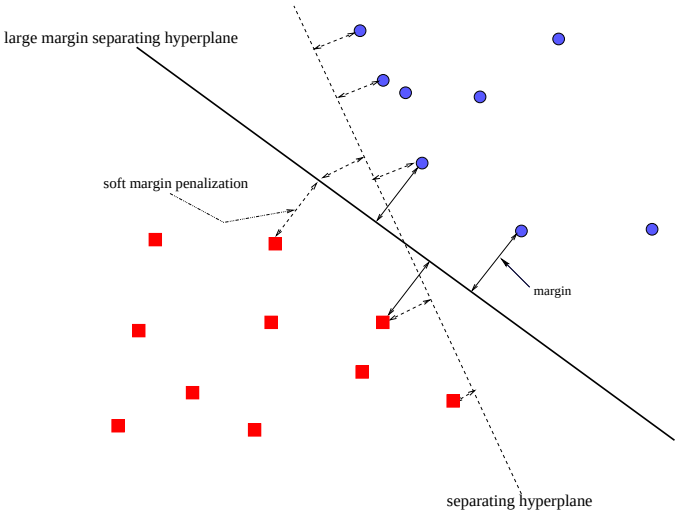

FIGURE 1.2: Margin and Linear Separating Hyperplane

soft-margin formulation as follows:

$$[\hat{w}, \hat{b}] = \arg\min_{w,b} \left[\|w\|_2^2 + C \sum_{i=1}^{n} \xi_i \right], \tag{1.4}$$
$$\text{subject to} \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \qquad (i = 1, \ldots, n).$$

In this method, we do not require that all training data can be separated with margin at least one. Instead, we introduce soft-margin slack variable $\xi_i \geq 0$ which penalize points with smaller margins. The parameter $C \geq 0$ balances the margin violation (when $\xi_i > 0$) and the regularization term $\|w\|_2^2$. When $C \to \infty$, we have $\xi_i \to 0$; therefore the margin condition $y_i(w^T x_i + b) \geq 1$ is enforced for all i. The resulting method becomes equivalent to the separable SVM formulation.

By eliminating ξ_i from (1.4), and let $\lambda = 1/(nC)$, we obtain the following equivalent formulation:

$$[\hat{w}, \hat{b}] = \arg\min_{w,b} \left[\frac{1}{n} \sum_{i=1}^{n} g((w^T x_i + b) y_i) + \lambda \|w\|_2^2 \right], \tag{1.5}$$

where

$$g(z) = \begin{cases} 1 - z & \text{if } z \leq 1, \\ 0 & \text{if } z > 0. \end{cases} \tag{1.6}$$

This method is rather similar to the ridge regression method (1.3). The main difference is a different loss function $g(\cdot)$ which is called hinge loss in the literature. Compared to the least squares loss, the hinge loss does not penalizes data points with large margin.

1.2 One-versus-all Method for Multi-category Classification

In practice, one often encounters multi-category classification, where the goal is to predict a label $y \in \{1, \ldots, k\}$ based on an observed input x.

If we have a binary classification algorithm that can learn a scoring function $f(x)$ from training data $\{(x_i, y_i)\}_{i=1,\ldots,n}$ with $y_i \in \{-1, 1\}$, then it can also be used for multi-category classification.

A commonly used method is one-versus-all. Consider a multi-category classification problem with k classes: $y_i \in \{1, \ldots, k\}$. We may reduce it into k binary classification problems indexed by class label $\ell \in \{1, \ldots, k\}$. The ℓ-th problem has training data $(x_i, y_i^{(\ell)})$ $(i = 1, \ldots, n)$, where we define the binary label $y_i^{(\ell)} \in \{-1, +1\}$ as:

$$y_i^{(\ell)} = \begin{cases} 1 & \text{if } y_i = \ell \\ -1 & \text{otherwise} \end{cases}.$$

For each binary problem ℓ defined this way with training data $\{(x_i, y_i^{(\ell)})\}$, we may use a binary classification algorithm to learn a scoring function $f_\ell(x)$. For example, using linear SVM or linear least squares, we can learn a linear scoring function of the form $f_\ell(x) = w^{(\ell)} x + b^{(\ell)}$ for each ℓ. For a data point x, the higher the score $f_\ell(x)$, the more likely x belongs to class ℓ. Therefore the classification rule for the multi-class problem is

$$h(x) = \arg \max_{\ell \in \{1,\ldots,k\}} f_\ell(x).$$

Figure 1.3 shows the decision boundary for three classes with linear scoring functions $f_\ell(x)$ $(\ell = 1, 2, 3)$. The three dashed lines represent the decision boundary $f_\ell(x) = 0$ $(\ell = 1, 2, 3)$ for the three binary problems. The three solid lines represent the decision boundary of the multi-class problem, determined by the lines $f_1(x) = f_2(x)$, $f_1(x) = f_3(x)$, and $f_2(x) = f_3(x)$ respectively.

1.3 Maximum Likelihood Estimation

A very general approach to machine learning is to construct a probability model of each individual data point as $p(x, y|\theta)$, where θ is the model parameter that needs to be estimated from the data. If the training data are independent, then the probability of the training data is

$$\prod_{i=1}^{n} p(x_i, y_i|\theta).$$

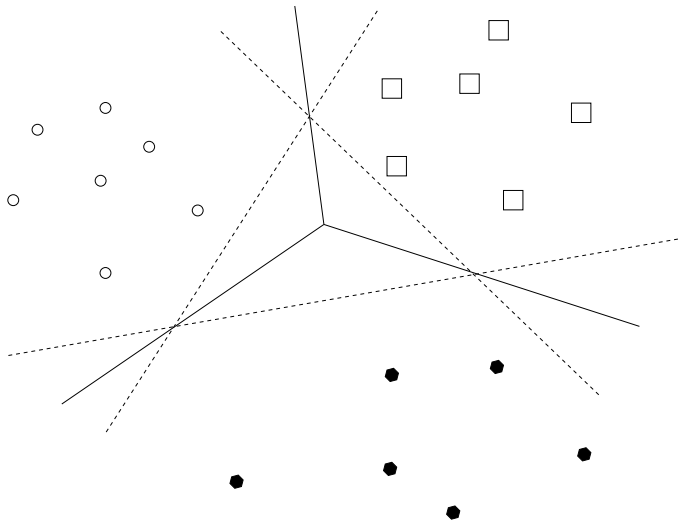

FIGURE 1.3: Multi-Class Linear Classifier Decision Boundary

A commonly used statistical technique for parameter estimation is the maximum-likelihood method, which finds a parameter $\hat{\theta}$ by maximizing the likelihood of the data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$:

$$\hat{\theta} = \arg\max_{\theta} \prod_{i=1}^{n} p(x_i, y_i|\theta).$$

More generally, we may impose a prior $p(\theta)$ on θ, and use the penalized maximum likelihood as follows:

$$\hat{\theta} = \arg\max_{\theta} \left[p(\theta) \prod_{i=1}^{n} p(x_i, y_i|\theta) \right].$$

In the Bayesian statistics literature, this method is also called MAP (maximum a posterior) estimator. A more common way to write the estimator is to take logarithm of the right hand side:

$$\hat{\theta} = \arg\max_{\theta} \left[\sum_{i=1}^{n} \ln p(x_i, y_i|\theta) + \ln p(\theta) \right].$$

For multi-classification problems with k classes $\{1, \ldots, k\}$, we obtain the following class conditional probability estimate

$$p(y|x) = \frac{p(x, y|\theta)}{\sum_{\ell=1}^{k} p(x, \ell|\theta)} \quad (y = 1, \ldots, k).$$

The class conditional probability function may be regarded as a scoring function, with the following classification rule that chooses the class with the largest conditional probability

$$h(x) = \arg \max_{y \in \{1,\ldots,k\}} p(y|x).$$

1.4 Generative and Discriminative Models

We shall give two concrete examples of maximum likelihood estimation for supervised learning. In the literature, there are two types of probability models called generative model and discriminative model. In a generative model, we model the conditional probability of input x given the label y; in a distriminative model, we directly model the condition probability $p(y|x)$. This section describes two methods: naive Bayes – a generative model, and logistic regression – a discriminative model. Both are commonly used linear classification methods.

1.4.1 Naive Bayes

The naive Bayes method starts with a generative model as in (1.7). Let $\theta = \{\theta^{(\ell)}\}_{\ell=1,\ldots,k}$ be the model parameter, where we use a different parameter $\theta^{(\ell)}$ for each class ℓ. Then we can model the data as:

$$p(x, y|\theta) = p(y)p(x|y, \theta), \quad p(x|y, \theta) = p(x|\theta^{(y)}). \tag{1.7}$$

This probability model can be visually represented using graphical model as in Figure 1.4, where the arrows indicate the conditional dependency structure among the variables.

The conditional class probability is

$$p(y|x) = \frac{p(y)p(x|\theta^{(y)})}{\sum_{\ell=1}^{k} p(y = \ell)p(x|\theta^{(\ell)})}.$$

In the following, we shall describe the multi-nomial naive Bayes model (McCallum and Nigam 1998) for $p(x|\theta^{(y)})$, which is important in many NLP problems. In this model, the observation x represents multiple (unordered) occurrences of d possible symbols. For example, x may represent the number of word occurrences in a text document by ignoring the word order information (such a representation is often referred to as "bag of words"). Each word in the document is one of d possible symbols from a dictionary.

Specifically, each data point x_i is a d-dimensional vector $x_i = [x_{i,1}, \ldots, x_{i,d}]$ representing the number of occurrences of these d symbols: for each symbol j in the dictionary, $x_{i,j}$ is the number of occurrences of symbol j. For each

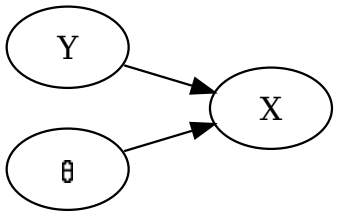

FIGURE 1.4: Graphical Representation of Generative Model

class ℓ, we assume that words are independently drawn from the dictionary according to a probability distribution $\theta^{(\ell)} = [\theta_1^{(\ell)}, \ldots, \theta_d^{(\ell)}]$: that is, the symbol j occurs with probability $\theta^{(\ell)}$. Now, for a data point x_i with label $y_i = \ell$, x_i comes from a multi-nomial distribution:

$$p(x_i|\theta^{(\ell)}) = \prod_{j=1}^{d} (\theta_j^{(\ell)})^{x_{i,j}} p\left(\sum_{j=1}^{d} x_{i,j}\right),$$

where we make the assumption that the total number of occurrences $\sum_{j=1}^{d} x_j$ is independent of the label $y_i = \ell$. For each $y \in \{1, \ldots, k\}$, we consider the so-called Dirichlet prior for $\theta^{(y)}$ as

$$p(\theta^{(y)}) \propto \prod_{j=1}^{d} (\theta_j^{(y)})^{\lambda},$$

where $\lambda > 0$ is a tuning parameter. We may use the MAP estimator to compute $\theta^{(\ell)}$ separately for each class ℓ:

$$\hat{\theta}^{(\ell)} = \arg\max_{\theta \in R^d} \left[\left(\prod_{i:y_i=\ell} \prod_{j=1}^{d} (\theta_j)^{x_{i,j}} \right) \prod_{j=1}^{d} (\theta_j)^{\lambda} \right]$$
$$\text{subject to } \sum_{j=1}^{d} \theta_j = 1, \text{ and } \theta_j \geq 0 \ (j = 1, \ldots, d).$$

The solution is given by

$$\hat{\theta}_j^{(\ell)} = \frac{n_j^{(\ell)}}{\sum_{j'=1}^{d} n_{j'}^{(\ell)}},$$

where

$$n_j^{(\ell)} = \lambda + \sum_{i:y_i=\ell} x_{i,j}.$$

Let $n^{(\ell)} = \sum_{i:y_i=\ell} 1$ be the number of training data with class label ℓ for each $\ell = 1, \ldots, k$, then we may estimate

$$p(y) = n^{(y)}/n.$$

With the above estimates, we obtain a scoring function

$$f_\ell(x) = \ln p(x|\theta^{(y)}) + \ln p(y) = (\hat{w}^{(\ell)})^T x + \hat{b}^{(\ell)},$$

where

$$\hat{w}^{(\ell)} = [\ln \hat{\theta}_j^{(\ell)}]_{j=1,\ldots,d}, \quad \hat{b}^{(\ell)} = \ln(n^{(\ell)}/n).$$

The conditional class probability is given by the Bayes rule:

$$p(y|x) = \frac{p(x|y)p(y)}{\sum_{\ell=1}^k p(x|\ell)p(\ell)} = \frac{e^{f_y(x)}}{\sum_{\ell=1}^k e^{f_\ell(x)}}, \tag{1.8}$$

and the corresponding classification rule is:

$$h(x) = \arg \max_{\ell \in \{1,\ldots,k\}} f_\ell(x).$$

1.4.2 Logistic Regression

Naive Bayes is a generative model in which we model the conditional probability of input x given the label y. After estimating the model parameter, we may then obtain the desired class conditional probability $p(y|x)$ using the Bayes rule. A different approach is to directly model the conditional probability $p(y|x)$. Such model is often called discriminative model. The dependency structure is given by Figure 1.5.

Ridge regression can be interpreted as the MAP estimator for a discriminative model with Gaussian noise (note that although ridge regression can be applied to classification problems, the underlying Gaussian noise assumption is only suitable for real valued output) and a Gaussian prior. The probability model is (with parameter $\theta = w$)

$$p(y|w, x) = N(w^T x, \tau^2),$$

with prior on parameter

$$p(w) = N(0, \sigma^2).$$

Here, we shall simply assume that σ^2 and τ^2 are known variance parameters, and the only unknown parameter is w. The MAP estimator is

$$\hat{w} = \arg\min_w \left[\frac{1}{\tau^2} \sum_{i=1}^n (w^T x_i - y_i)^2 + \frac{w^T w}{\sigma^2} \right],$$

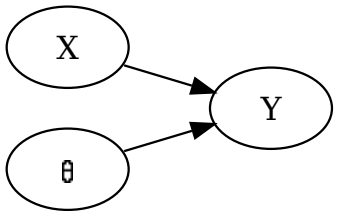

FIGURE 1.5: Graphical Representation of Discriminative Model

which is equivalent to the ridge regression method in (1.3) with $\lambda = \tau^2/\sigma^2$.

However, for binary classification, since $y_i \in \{-1, 1\}$ is discrete, the noise $w^T x_i - y_i$ cannot be a Gaussian distribution. The standard remedy to this problem is logistic regression, which models the conditional class probability as

$$p(y = 1|w, x) = \frac{1}{\exp(-w^T x) + 1}.$$

This means that both for $y = 1$ and $y = -1$, the likelihood is:

$$p(y|w, x) = \frac{1}{\exp(-w^T xy) + 1}. \tag{1.9}$$

If we again assume a Gaussian prior $p(w) = N(0, \sigma^2)$, then the penalized maximum likelihood estimate is

$$\hat{w} = \arg\min_w \left[\sum_{i=1}^{n} \ln(1 + \exp(-w^T x_i y_i)) + \lambda w^T w \right], \tag{1.10}$$

where $\lambda = 1/(2\sigma^2)$. Its use in text categorization as well as numerical algorithms for solving the problem can be found in (Zhang and Oles 2001).

Although binary logistic regression can be used to solve multi-class problems using the one-versus-all method described earlier, there is a direct formulation of multi-class logistic regression, which we shall describe next. Assume we have k classes, the naive Bayes method induces a probability of the form (1.8), where each function $f_\ell(x)$ is linear. Therefore as a direct generalization of the binary logistic model in (1.9), we may consider multi-category logistic model:

$$p(y|\{w^{(\ell)}\}, x) = \frac{e^{(w^{(y)})^T x}}{\sum_{\ell=1}^{k} e^{(w^{(\ell)})^T x}}. \tag{1.11}$$

The binary logistic model is a special case of (1.11) with $w^{(1)} = w$ and $w^{(-1)} = 0$. If we further assume Gaussian priors for each $w^{(\ell)}$:

$$P(w^{(\ell)}) = N(0, \sigma^2) \quad (\ell = 1, \ldots, k),$$

then we have the following MAP estimator:

$$\{\hat{w}^{(\ell)}\} = \arg\min_{\{w^{(\ell)}\}} \left[\sum_{i=1}^{n} \left(-(w^{(y_i)})^T x_i + \ln \sum_{\ell=1}^{k} e^{(w^{(\ell)})^T x_i} \right) + \lambda \sum_{\ell=1}^{k} (w^{(\ell)})^T w^{(\ell)} \right],$$

where $\lambda = 1/(2\sigma^2)$.

Multi-class logistic regression is also referred to as the maximum entropy method (MaxEnt) (Berger, Pietra, and Pietra 1996) under following more general form:

$$P(y|w, x) = \frac{\exp(w^T z(x, y))}{\sum_{\ell=1}^{k} \exp(w^T z(x, y))}, \tag{1.12}$$

where $z(x, y)$ is a human constructed vector called feature vector that depends both on x and on y. Let $w = [w^{(1)}, \ldots, w^{(k)}] \in R^{kd}$, and $z(x, y) = [0, \ldots, 0, x, 0, \ldots, 0]$, where x appears only in the positions $(y-1)d+1$ to yd that corresponds to $w^{(y)}$. With this representation, we have $w^T z(x, y) = (w^{(y)})^T x$, and (1.11) becomes a special case of (1.12).

Although logistic regression and naive Bayes share the same conditional class probability model, a major advantage of the logistic regression method is that it does not make any assumption on how x is generated. In contrast, naive Bayes assumes that x is generated in a specific way, and uses such information to estimate the model parameters. The logistic regression approach shows that even without any assumptions on x, the conditional probability can still be reliably estimated using discriminative maximum likelihood estimation.

1.5 Mixture Model and EM

Clustering is a common unsupervised learning problem. Its goal is to group unlabeled data into clusters so that data in the same cluster are similar, while data in different clusters are dissimilar. In clustering, we only observe the input data vector x, but do not observe its cluster label y. Therefore it is called unsupervised learning.

Clustering can also be viewed from a probabilistic modeling point of view. Assume that the data belong to k clusters. Each data point is a vector x_i, with $y_i \in \{1, \ldots, k\}$ being its corresponding (unobserved) cluster label. Each y_i takes value $\ell \in \{1, \ldots, k\}$ with probability $p(y_i = \ell | x_i)$.

The goal of clustering is to estimate $p(y_i = \ell | x_i)$. Similar to the naive Bayes approach, we start with a generative model of the following form:

$$p(x|\theta, y) = p(x|\theta^{(y)}).$$

Since y is not observed, we integrate out y to obtain:

$$p(x|\theta) = \sum_{c=1}^{k} \mu_\ell p(x|\theta^{(\ell)}), \tag{1.13}$$

where $\mu_\ell = p(y = \ell)$ $(\ell = 1, \ldots, k)$ are k parameters to be estimated from the data. The model in (1.13), with missing data (in this case, y) integrated out, is called a mixture model. A cluster $\ell \in \{1, \ldots, k\}$ is referred to as a mixture component.

We can interpret the data generation process in (1.13) as follows. First we pick a cluster ℓ (mixture component) from $\{1, \ldots, k\}$ with a fixed probability μ_ℓ as y_i; then we generate data points x_i according to the probability distribution $p(x_i|\theta^{(\ell)})$.

In order to obtain the cluster conditional probability $p(y = \ell|x)$, we can simple apply Bayes rule:

$$p(y|x) = \frac{\mu_y p(x|\theta^{(y)})}{\sum_{\ell=1}^{k} \mu_\ell p(x|\theta^{(\ell)})}. \tag{1.14}$$

Next we show how to estimate the model parameters $\{\theta^{(\ell)}, \mu_\ell\}$ from the data. This can be achieved using the penalized maximum likelihood method:

$$\{\hat{\theta}^{(\ell)}, \hat{\mu}_\ell\} = \arg \max_{\{\theta^{(\ell)}, \mu_\ell\}} \left[\sum_{i=1}^{n} \ln \sum_{\ell=1}^{k} \mu_\ell p(x_i|\theta^{(\ell)}) + \sum_{\ell=1}^{k} \ln p(\theta^{(\ell)}) \right]. \tag{1.15}$$

A direct optimization of (1.15) is usually difficult because the sum over the mixture components ℓ is inside the logarithm for each data point. However, for many simple models such as the naive Bayes example considered earlier, if we know the label y_i for each x_i, then the estimation becomes easier: we simply estimate the parameters using the equation

$$\{\hat{\theta}^{(\ell)}, \hat{\mu}_\ell\} = \arg \max_{\{\theta^{(\ell)}, \mu_\ell\}} \left[\sum_{i=1}^{n} \ln \mu_{y_i} p(x_i|\theta^{(y_i)}) + \sum_{\ell=1}^{k} \ln p(\theta^{(\ell)}) \right],$$

which does not have the sum inside the logarithm. For example, in the naive Bayes model, both μ_ℓ and $\theta^{(\ell)}$ can be estimated using simple counting.

The Expectation Maximization (EM) algorithm (Dempster, Laird, and Rubin 1977) simplifies the mixture model estimation problem by removing the sum over ℓ inside the logarithm in (1.15). Although we do not know the true value of y_i, we can estimate the conditional probability of $y_i = \ell$ for

Initialize $\theta^{(\ell)}$ and let $\mu_\ell = 1/k$ $(\ell = 1, \ldots, k)$
iterate
  // the E-step
  for $i = 1, \ldots, n$
    $q_{i,y} = \mu_y p(x_i|\theta^{(y)}) / \sum_{\ell=1}^{k} \mu_\ell p(x_i|\theta^{(\ell)})$ $\qquad (y = 1, \ldots, k)$
  end for
  // the M-step
  for $y = 1, \ldots, k$
    $\theta^{(y)} = \arg\max_{\tilde{\theta}} [\sum_{i=1}^{n} q_{i,y} \ln p(x_i|\tilde{\theta}) + \ln p(\tilde{\theta})]$
    $\mu_y = \sum_{i=1}^{n} q_{i,y}/n$
  end for
until convergence

FIGURE 1.6: The EM Algorithm

$\ell = 1, \ldots, k$ using (1.14). This can then be used to move the sum over ℓ inside the logarithm to a sum over ℓ outside the logarithm: for each data point i, we weight each mixture component ℓ by the estimated conditional class probability $p(y_i = \ell|x_i)$. That is, we repeatedly solve the following optimization problem

$$[\hat{\theta}^{(\ell)}_{new}, \hat{\mu}^{new}_\ell] = \arg \max_{\theta^{(\ell)}, \mu_\ell} \left[\sum_{i=1}^{n} p(y_i = \ell|x_i, \hat{\theta}_{old}, \hat{\mu}^{old}) \ln[\mu_\ell p(x_i|\theta^{(\ell)})] + \ln p(\theta^{(\ell)}) \right]$$

for $\ell = 1, \ldots, k$. Each time, we start with $[\hat{\theta}_{old}, \hat{\mu}^{old}]$ and update its value to $[\hat{\theta}_{new}, \hat{\mu}^{new}]$. Note that the solution of $\hat{\mu}^{new}_\ell$ is

$$\hat{\mu}^{new}_\ell = \frac{\sum_{i=1}^{n} p(y_i = \ell|x_i, \hat{\theta}_{old}, \hat{\mu}^{old})}{\sum_{i=1}^{n} \sum_{\ell'=1}^{k} p(y_i = \ell'|x_i, \hat{\theta}_{old}, \hat{\mu}^{old})}.$$

The algorithmic description of EM is given in Figure 1.6.

In practice, a few dozen iterations of EM often gives a satisfactory result. It is also necessary to start EM with different random initial parameters. This is to improve local optimal solutions found by the algorithm with each specific initial parameter configuration.

The EM algorithm can be used with any generative probability model including the naive Bayes model discussed earlier. Another commonly used model is Gaussian, where we assume $p(x|\theta^{(\ell)}) \propto \exp(-\frac{(\theta^{(\ell)}-x)^2}{2\sigma^2})$. Figure 1.7 shows a two dimensional Gaussian mixture model with two mixture components represented by the dotted circles. For simplicity, we may assume that σ^2 is known. Under this assumption, Figure 1.6 can be used to compute the mean vectors $\theta^{(\ell)}$ for the Gaussian mixture model, where the E and M steps are given by

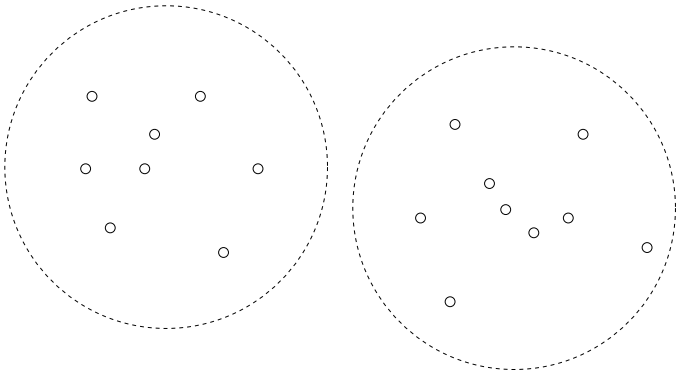

FIGURE 1.7: Gaussian Mixture Model with Two Mixture Components

- (E step) $q_{i,y} = \mu_y \exp\left(-\frac{(x_i-\theta^{(y)})^2}{2\sigma^2}\right) / \sum_{\ell=1}^{k} \mu_\ell \exp\left(-\frac{(x_i-\theta^{(\ell)})^2}{2\sigma^2}\right)$.
- (M step) $\mu_\ell = \sum_{i=1}^{n} q_{i,y}/n$ and $\theta^{(\ell)} = \sum_{i=1}^{n} q_{i,\ell} x_i / \sum_{i=1}^{n} q_{i,\ell}$.

1.6 Sequence Prediction Models

NLP problems involve sentences that can be regarded as sequences. For example, a sentence of n words can be represented as a sequence of n observations $\{x_1, \ldots, x_n\}$. We are often interested in predicting a sequence of hidden labels $\{y_1, \ldots, y_n\}$, one for each word. For example, in part-of-speech (POS) tagging, y_i is the POS of the word x_i.

The problem of predicting hidden labels $\{y_i\}$ given observations $\{x_i\}$ is often referred to as sequence prediction. Although this task may be regarded as a supervised learning problem, it has an extra complexity that data (x_i, y_i) in the sequence are dependent. For example, label y_i may depend on the previous label y_{i-1}. In the probabilistic modeling approach, one may construct a probability model of the whole sequence $\{(x_i, y_i)\}$, and then estimate the model parameters.

Similar to the standard supervised learning setting with independent observations, we have two types of models for sequence prediction: generative and discriminative. We shall describe both approaches in this section. For simplicity, we only consider first order dependency where y_i only depends on y_{i-1}. Higher order dependency (e.g. y_i may depend on y_{i-2}, y_{i-3} and so on) can be easily incorporated but requires more complicated notations. Also for simplicity, we shall ignore sentence boundaries, and just assume that

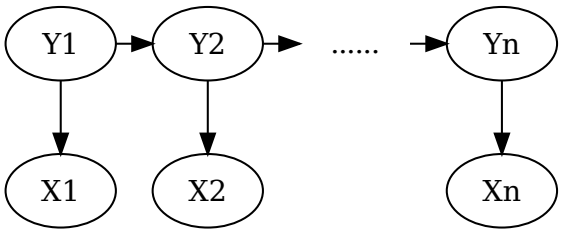

FIGURE 1.8: Graphical Representation of Hidden Markov Model

the training data contain n sequential observations. In the following, we will assume that each y_i takes one of the k values in $\{1, \ldots, k\}$.

1.6.1 Hidden Markov Model

The standard generative model for sequence prediction is the hidden Markov model (HMM), illustrated in Figure 1.8. It has been used in various NLP problems, such as POS tagging (Kupiec 1992). This model assumes that each y_i depends on the previous label y_{i-1}, and x_i only depends on y_i. Since x_i depends only on y_i, if the labels are observed on the training data, we may write the likelihood mathematically as:

$$p(x_i|y_i\theta) = p(x_i|\theta^{(y_i)}),$$

which is identical to (1.7).

One often uses the naive Bayes model for $p(x|\theta^{(y)})$. Because the observations x_i are independent conditioned on y_i, the parameter θ can be estimated from the training data using exactly the same method described in Section 1.4.1. Using the Bayes rule, the conditional probability of the label sequence $\{y_i\}$ is given by

$$p(\{y_i\}|\{x_i\}, \theta) \propto p(\{y_i\}) \prod_{i=1}^{n} p(x_i|\theta^{(y_i)}).$$

That is,

$$p(\{y_i\}|\{x_i\}, \theta) \propto \prod_{i=1}^{n} [p(x_i|\theta^{(y_i)}) p(y_i|y_{i-1})]. \tag{1.16}$$

Similar to Section 1.4.1, the probability $p(y_i = a|y_{i-1} = b) = p(y_i = a, y_{i-1} = b)/p(y_{i-1} = b)$ can be estimated using counting. Let n_b be the number of

training data with label b, and $n_{a,b}$ be the number of consecutive label pairs (y_i, y_{i-1}) with value (a, b). We can estimate the conditional probability as

$$\begin{aligned} p(y_{i-1} = b) =& \frac{n_b}{n}, \\ p(y_i = a, y_{i-1} = b) =& \frac{n_{a,b}}{n}, \\ p(y_i = a | y_{i-1} = b) =& \frac{n_{a,b}}{n_b}. \end{aligned}$$

The process of estimating the sequence $\{y_i\}$ from observation $\{x_i\}$ is often called decoding. A standard method is the maximum-likelihood decoding, which finds the most likely sequence $\{\hat{y}_i\}$ based on the conditional probability model (1.16). That is,

$$[\{\hat{y}_i\}] = \arg\max_{\{y_i\}} \sum_{i=1}^{n} f(y_i, y_{i-1}), \tag{1.17}$$

where

$$f(y_i, y_{i-1}) = \ln p(x_i | \theta^{(y_i)}) + \ln p(y_i | y_{i-1}).$$

It is not possible to enumerate all possible sequences $\{y_i\}$ and pick the largest score in (1.17) because the number of possible label sequences is k^n. However, an efficient procedure called the Viterbi decoding algorithm can be used to solve (1.17). The algorithm uses dynamic programming to track the best score up to a position j, and update the score recursively for $j = 1, \ldots, n$. Let

$$s_j(y_j) = \max_{\{y_i\}_{i=1,\ldots,j-1}} \sum_{i=1}^{j} f(y_i, y_{i-1}),$$

then it is easy to check that we have the following recursive identity:

$$s_{j+1}(y_{j+1}) = \max_{y_j \in \{1,\ldots,k\}} [s_j(y_j) + f(y_{j+1}, y_j)].$$

Therefore $s_j(y_j)$ can be computed recursively for $j = 1, \ldots, n$. After computing $s_j(y_j)$, we may trace back $j = n, n-1, \ldots, 1$ to find the optimal sequence $\{\hat{y}_j\}$. The Viterbi algorithm that solves (1.17) is presented in Figure 1.9.

1.6.2 Local Discriminative Model for Sequence Prediction

HMM is a generative model for sequence prediction. Similar to the standard supervised learning, one can also construct discriminative models for sequence prediction. In a discriminative model, in addition to the Markov dependency of y_i on y_{i-1}, we also allow an arbitrary dependency of y_i on $x_1^n = \{x_i\}_{i=1,\ldots,n}$. That is, we consider a model of the form

$$p(\{y_i\} | x_1^n, \theta) = \prod_{i=1}^{n} p(y_i | y_{i-1}, x_1^n, \theta). \tag{1.18}$$

Initialize $s_0(y_0) = 0$ $(y_0 = 1, \ldots, k)$
for $j = 0, \ldots, n-1$
  $s_{j+1}(y_{j+1}) = \max_{y_j \in \{1,\ldots,k\}}[s_j(y_j) + f(y_{j+1}, y_j)] \qquad (y_{j+1} = 1, \ldots, k)$
end for
$\hat{y}_n = \arg\max_{y_n \in \{1,\ldots,k\}} s_n(y_n)$
for $j = n-1, \ldots, 1$
  $\hat{y}_j = \arg\max_{y_j \in \{1,\ldots,k\}}[s_j(y_j) + f(\hat{y}_{j+1}, y_j)]$
end for

FIGURE 1.9: The Viterbi Algorithm

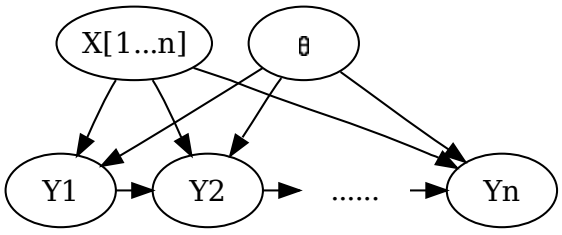

FIGURE 1.10: Graphical Representation of Discriminative Local Sequence Prediction Model

The graphical model representation is given in Figure 1.10.

One may use logistic regression (MaxEnt) to model the conditional probability in (1.18). That is, we let $\theta = w$ and

$$p(y_i|y_{i-1}, x_1^n, \theta) = \frac{\exp(w^T z_i(y_i, y_{i-1}, x_1^n))}{\sum_{\ell=1}^{k} \exp(w^T z_i(y_i = \ell, y_{i-1}, x_1^n))}. \tag{1.19}$$

The vector $z_i(y_i, y_{i-1}, x_1^n)$ is a human constructed vector called feature vector. This model has identical form as the maximum entropy model (1.12). Therefore supervised training algorithm for logistic regression can be directly applied to train the model parameter θ. On the test data, given a sequence x_1^n, one can use the Viterbi algorithm to decode $\{y_i\}$ using the scoring function $f_\ell(y_i, y_{i-1}) = \ln p(y_i|y_{i-1}, x_1^n, \theta)$. This method has been widely used in natural language processing, for example, part-of-speech tagging (Ratnaparkhi 1996).

More generally, one may reduce sequence prediction into a standard prediction problem, where we simply predict the next label y_i given the previous label y_{i-1} and the observation x_1^n. One may use any classification algorithm such as SVM to solve this problem. The scoring function returned by the underlying classifier can then be used as the scoring function for the Viterbi decoding algorithm. An example of this approach is (Zhang, Damerau, and Johnson 2002).

1.6.3 Global Discriminative Model for Sequence Prediction

In (1.18), we decompose the conditional model of the label sequence $\{y_i\}$ using local model of the form $p(y_i|y_{i-1}, x_1^n, \theta)$ at each position i. Another approach is to treat the label sequence $y_1^n = \{y_i\}$ directly as a multi-class classification problem with k^n possible values. We can then directly apply the MaxEnt model (1.12) to this k^n-class multi-category classification problem using the following representation:

$$p(y_1^n|w, x_1^n) = \frac{e^{f(w, x_1^n, y_1^n)}}{\sum_{y_1^n} e^{f(w, x_1^n, y_1^n)}}, \tag{1.20}$$

where

$$f(w, x_1^n, y_1^n) = \sum_{i=1}^{n} w^T z_i(y_i, y_{i-1}, x_1^n),$$

where $z_i(y_i, y_{i-1}, x_1^n)$ is a feature vector just like (1.19). While in (1.19), we model the local conditional probability $p(y_i|y_{i-1})$ which is a small fragment of the total label sequence $\{y_i\}$, in (1.20), we directly model the global label sequence.

The probability model (1.20) is called conditional random field (CRF) (Lafferty, McCallum, and Pereira 2001). The graphical model representation is

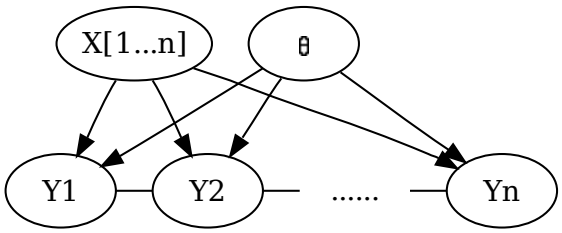

FIGURE 1.11: Graphical Representation of Discriminative Global Sequence Prediction Model

given in Figure 1.11. Unlike Figure 1.10, the dependency between each y_i and y_{i-1} in Figure 1.11. is undirectional. This means that we do not directly model the conditional dependency $p(y_i|y_{i-1})$, and do not normalize the conditional probability at each point i in the maximum entropy representation of the label sequence probability.

The CRF model is more difficult to train because the normalization factor in the denominator of (1.20) has to be computed in the training phase. Although the summation is over k^n possible values of the label sequence y_1^n, similar to the Viterbi decoding algorithm, the computation can be arranged efficiently using dynamic programming. In decoding, the denominator can be ignored in the maximum-likelihood solution. That is, the most likely sequence $\{\hat{y}_i\}$ is the solution of

$$\{\hat{y}_i\} = \arg\max_{y_1^n} \sum_{i=1}^{n} w^T z_i(y_i, y_{i-1}, x_1^n). \tag{1.21}$$

The solution of this problem can be efficiently computed using the Viterbi algorithm.

More generally, global discriminative learning refers to the idea of treating sequence prediction as a multi-category classification problem with k^n classes, and a classification rule of the form (1.21). This approach can be used with some other learning algorithms such as Perceptron (Collins 2002) and large margin classifiers (Taskar, Guestrin, and Koller 2004; Tsochantaridis, Joachims, Hofmann, and Altun 2005; Tillmann and Zhang 2008).

Bibliography

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

Collins, M. (2002, July). Discriminative Training Methods for Hidden Markov Models: Theory and Experi ments with Perceptron Algorithms. In *Proceedings of the Conference on Emperical Methods in Natural Language Modeli ng (EMNLP'02)*, Philadelphia,PA, pp. 1–8.

Cortes, C. and V. Vapnik (1995). Support vector networks. *Machine Learning 20*, 273–297.

Dempster, A., N. Laird, and D. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B 39*(1), 1–38.

Hoerl, A. E. and R. W. Kennard (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics 12*(1), 55–67.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. In *European Conference on Machine Learing, ECML-98*, pp. 137–142.

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

Lafferty, J., A. McCallum, and F. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling s equence data. In *Proceedings of ICML-01*, San Francisco, CA, pp. 282–289. Morgan Kaufmann.

McCallum, A. and K. Nigam (1998). A comparison of event models for naive Bayes text classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pp. 41–48.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 133–142.

Taskar, B., C. Guestrin, and D. Koller (2004). Max-Margin Markov Networks. In S. Thrun, L. Saul, and B. Schölkopf (Eds.), *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press.

Tillmann, C. and T. Zhang (2008). An online relevant set algorithm for statistical machine translation. *IEEE Transactions on Audio, Speech, and Language processing 16*(7), 1274–1286.

Tsochantaridis, I., T. Joachims, T. Hofmann, and Y. Altun (2005). Large Margin Methods for Structured and Interdependent Output Vari ables. *JMLR 6*, 1453–1484.

Zhang, T., F. Damerau, and D. E. Johnson (2002). Text chunking based on a generalization of Winnow. *Journal of Machine Learning Research 2*, 615–637.

Zhang, T. and F. J. Oles (2001). Text categorization based on regularized linear classification methods. *Information Retrieval 4*, 5–31.