CHAPTER 1

GENERATIVE AND DISCRIMINATIVE CLASSIFIERS: NAIVE BAYES AND LOGISTIC REGRESSION

1 Learning Classifiers based on Bayes Rule

Here we consider the relationship between supervised learning, or function approximation problems, and Bayesian reasoning. We begin by considering how to design learning algorithms based on Bayes rule.

Consider a supervised learning problem in which we wish to approximate an unknown target function $f : X \rightarrow Y$, or equivalently $P(Y|X)$. To begin, we will assume Y is a boolean-valued random variable, and X is a vector containing n boolean attributes. In other words, $X = \langle X_1, X_2 \ldots, X_n \rangle$, where X_i is the random variable denoting the ith attribute of X.

Applying Bayes rule, we see that $P(Y = y_i|X)$ can be represented as

$$P(Y = y_i|X = x_k) = \frac{P(X = x_k|Y = y_i)P(Y = y_i)}{\sum_j P(X = x_k|Y = y_j)P(Y = y_j)}$$

where y_m represents the mth possible value for Y, and where the summation in the denominator is over all legal values of the random variable Y.

One way to learn $P(Y|X)$ is to use the training data to estimate $P(X|Y)$ and $P(Y)$. We can then use these estimates, together with Bayes rule above, to determine $P(Y|X = x_k)$ for any new instance x_k.

A NOTE ON NOTATION: We will consistently use upper case symbols (e.g., X) to refer to random variables, including both vector and non-vector variables. If X is a vector, then we use subscripts (e.g., X_i to refer to each random variable in X). We use lower case symbols to refer to *values* of random variables (e.g., $X_i = x_{ij}$ may refer to random variable X_i taking on its jth possible value). We will sometimes abbreviate by omitting variable names, for example abbreviating $P(X_i = x_{ij}|Y = y_k)$ to $P(x_{ij}|y_k)$. We will write $E[X]$ to refer to the expected value of X. We use superscripts to index training examples (e.g., X_i^j refers to the value of the random variable X_i in the jth training example.). We use $\delta(x)$ to denote an "indicator" function whose value is 1 if its logical argument x is true, and whose value is 0 otherwise. We use the $\#D\{x\}$ operator to denote the number of elements in the set D that satisfy property x. We use a "hat" to indicate estimates; for example, $\hat{\theta}$ indicates an estimated value of θ.

1.1 Unbiased Learning of Bayes Classifiers is Impractical

If we are going to train a Bayes classifier by estimating $P(X|Y)$ and $P(Y)$, then it is reasonable to ask how much training data will be required to obtain reliable estimates of these distributions. Let us assume training examples are generated by drawing instances at random from an unknown underlying distribution $P(X)$, then allowing a teacher to label this example with its Y value.

A hundred independently drawn training examples will usually suffice to obtain a maximum likelihood estimate of $P(Y)$ that is within a few percent of its correct value[1] when Y is a boolean variable. However, accurately estimating $P(X|Y)$ typically requires many more examples. To see why, consider the number of parameters we must estimate when Y is boolean and X is a vector of n boolean attributes. In this case, we need to estimate a set of parameters

$$\theta_{ij} \equiv P(X = x_i|Y = y_j)$$

where the index i takes on 2^n possible values (one for each of the possible vector values of X), and j takes on 2 possible values. Therefore, we will need to estimate approximately 2^{n+1} parameters. To calculate the exact number of required

[1] Why? See Chapter 5 of edition 1 of *Machine Learning*.

parameters, note for any fixed j, the sum over i of θ_{ij} must be one. Therefore, for any particular value y_j, and the 2^n possible values of x_i, we need compute only $2^n - 1$ independent parameters. Given the two possible values for Y, we must estimate a total of $2(2^n - 1)$ such θ_{ij} parameters. Unfortunately, this corresponds to two distinct parameters for *each* of the distinct instances in the instance space for X! Furthermore, to obtain reliable estimates of each of these parameters, we will need to observe each of these distinct instances multiple times! This is clearly unrealistic in most practical learning domains.

2 Naive Bayes Algorithm

Given the intractable sample complexity for learning Bayesian classifiers, we must look for ways to reduce this complexity. The Naive Bayes classifier does this by making a conditional independence assumption that dramatically reduces the number of parameters to be estimated when modeling $P(X|Y)$, from our original $2(2^n - 1)$ to just $2n$.

2.1 Conditional Independence

> *Definition:* Given random variables X, Y and Z, we say X is **conditionally independent** of Y given Z, if and only if the probability distribution governing X is independent of the value of Y given Z; that is
>
> $$(\forall i, j, k) P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

As an example, consider three boolean random variables to describe the current weather: *Rain*, *Thunder* and *Lightning*. We might reasonably assert that *Thunder* is independent of *Rain* given *Lightning*. Because we know *Lightning* causes *Thunder*, once we know whether or not there is *Lightning*, no additional information about *Thunder* is provided by the value of *Rain*. Of course there is a clear dependence of *Thunder* on *Rain* in general, but there is no *conditional* dependence once we know the value of *Lightning*.

2.2 Derivation of Naive Bayes Algorithm

The Naive Bayes algorithm is a classification algorithm based on Bayes rule, that assumes the attributes $X_1 \ldots X_n$ are all conditionally independent of one another, given Y. The value of this assumption is that it dramatically simplifies the representation of $P(X|Y)$, and the problem of estimating it from the training data. Consider, for example, the case where $X = \langle X_1, X_2 \rangle$. In this case

$$P(X|Y) = P(X_1, X_2|Y)$$

$$\begin{aligned} &= P(X_1|X_2,Y)P(X_2|Y) \\ &= P(X_1|Y)P(X_2|Y) \end{aligned}$$

Where the second line follows from a general property of probabilities, and the third line follows directly from our above definition of conditional independence. More generally, when X contains n attributes which are conditionally independent of one another given Y, we have

$$P(X_1 \ldots X_n|Y) = \prod_{i=1}^{n} P(X_i|Y) \tag{1}$$

Notice that when Y and the X_i are boolean variables, we need only $2n$ parameters to define $P(X_i = x_{ik}|Y = y_j)$ for the necessary i, j, k. This is a dramatic reduction compared to the $2(2^n - 1)$ parameters needed to characterize $P(X|Y)$ if we make no conditional independence assumption.

Let us now derive the Naive Bayes algorithm, assuming in general that Y is any discrete-valued variable, and the attributes $X_1 \ldots X_n$ are any discrete or real-valued attributes. Our goal is to train a classifier that will output the probability distribution over possible values of Y, for each new instance X that we ask it to classify. The expression for the probability that Y will take on its kth possible value, according to Bayes rule, is

$$P(Y = y_k|X_1 \ldots X_n) = \frac{P(Y = y_k)P(X_1 \ldots X_n|Y = y_k)}{\sum_j P(Y = y_j)P(X_1 \ldots X_n|Y = y_j)}$$

where the sum is taken over all possible values y_j of Y. Now, assuming the X_i are conditionally independent given Y, we can use equation (1) to rewrite this as

$$P(Y = y_k|X_1 \ldots X_n) = \frac{P(Y = y_k)\prod_i P(X_i|Y = y_k)}{\sum_j P(Y = y_j)\prod_i P(X_i|Y = y_j)} \tag{2}$$

Equation (2) is the fundamental equation for the Naive Bayes classifier. Given a new instance $X^{new} = \langle X_1 \ldots X_n \rangle$, this equation shows how to calculate the probability that Y will take on any given value, given the observed attribute values of X^{new} and given the distributions $P(Y)$ and $P(X_i|Y)$ estimated from the training data. If we are interested only in the most probable value of Y, then we have the Naive Bayes classification rule:

$$Y \leftarrow \arg\max_{y_k} \frac{P(Y = y_k)\prod_i P(X_i|Y = y_k)}{\sum_j P(Y = y_j)\prod_i P(X_i|Y = y_j)}$$

which simplifies to the following (because the denominator does not depend on y_k).

$$Y \leftarrow \arg\max_{y_k} \; P(Y = y_k)\prod_i P(X_i|Y = y_k) \tag{3}$$

2.3 Naive Bayes for Discrete-Valued Inputs

To summarize, let us precisely define the Naive Bayes learning algorithm by describing the parameters that must be estimated, and how we may estimate them.

When the n input attributes X_i each take on J possible discrete values, and Y is a discrete variable taking on K possible values, then our learning task is to estimate two sets of parameters. The first is

$$\theta_{ijk} \equiv P(X_i = x_{ij} | Y = y_k) \tag{4}$$

for each input attribute X_i, each of its possible values x_{ij}, and each of the possible values y_k of Y. Note there will be nJK such parameters, and note also that only $n(J-1)K$ of these are independent, given that they must satisfy $1 = \sum_j \theta_{ijk}$ for each pair of i,k values.

In addition, we must estimate parameters that define the prior probability over Y:

$$\pi_k \equiv P(Y = y_k) \tag{5}$$

Note there are K of these parameters, $(K-1)$ of which are independent.

We can estimate these parameters using either maximum likelihood estimates (based on calculating the relative frequencies of the different events in the data), or using Bayesian MAP estimates (augmenting this observed data with prior distributions over the values of these parameters).

Maximum likelihood estimates for θ_{ijk} given a set of training examples D are given by

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij} | Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\}}{\#D\{Y = y_k\}} \tag{6}$$

where the $\#D\{x\}$ operator returns the number of elements in the set D that satisfy property x.

One danger of this maximum likelihood estimate is that it can sometimes result in θ estimates of zero, if the data does not happen to contain any training examples satisfying the condition in the numerator. To avoid this, it is common to use a "smoothed" estimate which effectively adds in a number of additional "hallucinated" examples, and which assumes these hallucinated examples are spread evenly over the possible values of X_i. This smoothed estimate is given by

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij} | Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\} + l}{\#D\{Y = y_k\} + lJ} \tag{7}$$

where J is the number of distinct values X_i can take on, and l determines the strength of this smoothing (i.e., the number of hallucinated examples is lJ). This expression corresponds to a MAP estimate for θ_{ijk} if we assume a Dirichlet prior distribution over the θ_{ijk} parameters, with equal-valued parameters. If l is set to 1, this approach is called Laplace smoothing.

Maximum likelihood estimates for π_k are

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\}}{|D|} \tag{8}$$

where $|D|$ denotes the number of elements in the training set D.

Alternatively, we can obtain a smoothed estimate, or equivalently a MAP estimate based on a Dirichlet prior over the π_k parameters assuming equal priors on each π_k, by using the following expression

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\} + l}{|D| + lK} \tag{9}$$

where K is the number of distinct values Y can take on, and l again determines the strength of the prior assumptions relative to the observed data D.

2.4 Naive Bayes for Continuous Inputs

In the case of continuous inputs X_i, we can of course continue to use equations (2) and (3) as the basis for designing a Naive Bayes classifier. However, when the X_i are continuous we must choose some other way to represent the distributions $P(X_i|Y)$. One common approach is to assume that for each possible discrete value y_k of Y, the distribution of each continuous X_i is Gaussian, and is defined by a mean and standard deviation specific to X_i and y_k. In order to train such a Naive Bayes classifier we must therefore estimate the mean and standard deviation of each of these Gaussians:

$$\mu_{ik} = E[X_i|Y = y_k] \tag{10}$$

$$\sigma_{ik}^2 = E[(X_i - \mu_{ik})^2|Y = y_k] \tag{11}$$

for each attribute X_i and each possible value y_k of Y. Note there are $2nK$ of these parameters, all of which must be estimated independently.

Of course we must also estimate the priors on Y as well

$$\pi_k = P(Y = y_k) \tag{12}$$

The above model summarizes a Gaussian Naive Bayes classifier, which assumes that the data X is generated by a mixture of class-conditional (i.e., dependent on the value of the class variable Y) Gaussians. Furthermore, the Naive Bayes assumption introduces the additional constraint that the attribute values X_i are independent of one another within each of these mixture components. In particular problem settings where we have additional information, we might introduce additional assumptions to further restrict the number of parameters or the complexity of estimating them. For example, if we have reason to believe that noise in the observed X_i comes from a common source, then we might further assume that all of the σ_{ik} are identical, regardless of the attribute i or class k (see the homework exercise on this issue).

Again, we can use either maximum likelihood estimates (MLE) or maximum a posteriori (MAP) estimates for these parameters. The maximum likelihood estimator for μ_{ik} is

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k) \tag{13}$$

where the superscript j refers to the jth training example, and where $\delta(Y = y_k)$ is 1 if $Y = y_k$ and 0 otherwise. Note the role of δ here is to select only those training examples for which $Y = y_k$.

The maximum likelihood estimator for σ_{ik}^2 is

$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k) \tag{14}$$

This maximum likelihood estimator is biased, so the minimum variance unbiased estimator (MVUE) is sometimes used instead. It is

$$\hat{\sigma}_{ik}^2 = \frac{1}{(\sum_j \delta(Y^j = y_k)) - 1} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k) \tag{15}$$

3 Logistic Regression

Logistic Regression is an approach to learning functions of the form $f : X \rightarrow Y$, or $P(Y|X)$ in the case where Y is discrete-valued, $P(Y)$ is governed by a multinomial, and $X = \langle X_1 \ldots X_n \rangle$ is any vector containing discrete or continuous variables. In this section we will primarily consider the case where Y is a boolean variable, in order to simplify notation. In the final subsection we extend our treatment to the case where Y takes on any finite number of discrete values.

Logistic Regression assumes a parametric form for the distribution $P(Y|X)$, then directly estimates its parameters from the training data. The parametric model assumed by Logistic Regression in the case where Y is boolean is:

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)} \tag{16}$$

and

$$P(Y = 0|X) = \frac{\exp(w_0 + \sum_{i=1}^n w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)} \tag{17}$$

Notice that equation (17) follows directly from equation (16), because the sum of these two probabilities must equal 1.

One highly convenient property of this form for $P(Y|X)$ is that it leads to a simple linear expression for classification. To classify any given X we generally want to assign the value y_k that maximizes $P(Y = y_k|X)$. Put another way, we assign the label $Y = 0$ if the following condition holds:

$$1 < \frac{P(Y = 0|X)}{P(Y = 1|X)}$$

substituting from equations (16) and (17), this becomes

$$1 < \exp(w_0 + \sum_{i=1}^n w_i X_i)$$

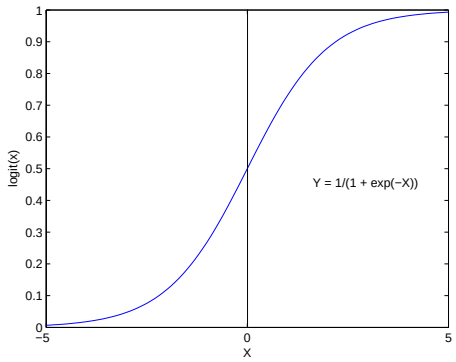

Figure 1: Form of the logistic function. In Logistic Regression, $P(Y|X)$ is assumed to follow this form.

and taking the natural log of both sides we have a linear classification rule that assigns label $Y = 0$ if X satisfies

$$0 < w_0 + \sum_{i=1}^{n} w_i X_i \tag{18}$$

and assigns $Y = 1$ otherwise.

Interestingly, the parametric form of $P(Y|X)$ used by Logistic Regression is precisely the form implied by the assumptions of a Gaussian Naive Bayes classifier. Therefore, we can view Logistic Regression as a closely related alternative to GNB, though the two can produce different results in many cases.

3.1 Form of $P(Y|X)$ for Gaussian Naive Bayes Classifier

Here we derive the form of $P(Y|X)$ entailed by the assumptions of a Gaussian Naive Bayes (GNB) classifier, showing that it is precisely the form used by Logistic Regression and summarized in equations (16) and (17). In particular, consider a GNB based on the following modeling assumptions:

- Y is boolean, governed by a Bernoulli distribution, with parameter $\pi = P(Y = 1)$
- $X = \langle X_1 \ldots X_n \rangle$, where each X_i is a continuous random variable
- For each X_i, $P(X_i|Y = y_k)$ is a Gaussian distribution of the form $N(\mu_{ik}, \sigma_i)$
- For all i and $j \neq i$, X_i and X_j are conditionally independent given Y

Note here we are assuming the standard deviations σ_i vary from attribute to attribute, but do not depend on Y.

We now derive the parametric form of $P(Y|X)$ that follows from this set of GNB assumptions. In general, Bayes rule allows us to write

$$P(Y=1|X) = \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1)+P(Y=0)P(X|Y=0)}$$

Dividing both the numerator and denominator by the numerator yields:

$$P(Y=1|X) = \frac{1}{1+\frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}}$$

or equivalently

$$P(Y=1|X) = \frac{1}{1+\exp(\ln\frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)})}$$

Because of our conditional independence assumption we can write this

$$\begin{aligned} P(Y=1|X) &= \frac{1}{1+\exp(\ln\frac{P(Y=0)}{P(Y=1)}+\sum_i \ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)})} \\ &= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\sum_i \ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)})} \end{aligned} \tag{19}$$

Note the final step expresses $P(Y=0)$ and $P(Y=1)$ in terms of the binomial parameter π.

Now consider just the summation in the denominator of equation (19). Given our assumption that $P(X_i|Y=y_k)$ is Gaussian, we can expand this term as follows:

$$\begin{aligned} \sum_i \ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)} &= \sum_i \ln\frac{\frac{1}{\sqrt{2\pi\sigma_i^2}}\exp\left(\frac{-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_i^2}}\exp\left(\frac{-(X_i-\mu_{i1})^2}{2\sigma_i^2}\right)} \\ &= \sum_i \ln\exp\left(\frac{(X_i-\mu_{i1})^2-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right) \\ &= \sum_i \left(\frac{(X_i-\mu_{i1})^2-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right) \\ &= \sum_i \left(\frac{(X_i^2-2X_i\mu_{i1}+\mu_{i1}^2)-(X_i^2-2X_i\mu_{i0}+\mu_{i0}^2)}{2\sigma_i^2}\right) \\ &= \sum_i \left(\frac{2X_i(\mu_{i0}-\mu_{i1})+\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right) \\ &= \sum_i \left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i+\frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right) \end{aligned} \tag{20}$$

Note this expression is a linear weighted sum of the X_i's. Substituting expression (20) back into equation (19), we have

$$P(Y=1|X) = \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi} + \sum_i \left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2-\mu_{i0}^2)}{2\sigma_i^2}\right))} \tag{21}$$

Or equivalently,

$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_{i=1}^n w_i X_i)} \tag{22}$$

where the weights $w_1 \ldots w_n$ are given by

$$w_i = \frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}$$

and where

$$w_0 = \ln\frac{1-\pi}{\pi} + \sum_i \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}$$

Also we have

$$P(Y=0|X) = 1 - P(Y=1|X) = \frac{\exp(w_0 + \sum_{i=1}^n w_i X_i)}{1+\exp(w_0 + \sum_{i=1}^n w_i X_i)} \tag{23}$$

3.2 Estimating Parameters for Logistic Regression

The above subsection proves that $P(Y|X)$ can be expressed in the parametric form given by equations (16) and (17), under the Gaussian Naive Bayes assumptions detailed there. It also provides the value of the weights w_i in terms of the parameters estimated by the GNB classifier. Here we describe an alternative method for estimating these weights. We are interested in this alternative for two reasons. First, the form of $P(Y|X)$ assumed by Logistic Regression holds in many problem settings beyond the GNB problem detailed in the above section, and we wish to have a general method for estimating it in a more broad range of cases. Second, in many cases we may suspect the GNB assumptions are not perfectly satisfied. In this case we may wish to estimate the w_i parameters directly from the data, rather than going through the intermediate step of estimating the GNB parameters which forces us to adopt its more stringent modeling assumptions.

One reasonable approach to training Logistic Regression is to choose parameter values that maximize the conditional data likelihood. The conditional data likelihood is the probability of the observed Y values in the training data, conditioned on their corresponding X values. We choose parameters W that satisfy

$$W \leftarrow \arg\max_W \prod_l P(Y^l|X^l, W)$$

where $W = \langle w_0, w_1 \ldots w_n \rangle$ is the vector of parameters to be estimated, Y^l denotes the observed value of Y in the lth training example, and X^l denotes the observed

value of X in the lth training example. The expression to the right of the arg max is the conditional data likelihood. Here we include W in the conditional, to emphasize that the expression is a function of the W we are attempting to maximize.

Equivalently, we can work with the log of the conditional likelihood:

$$W \leftarrow \arg\max_W \sum_l \ln P(Y^l|X^l,W)$$

This conditional data log likelihood, which we will denote $l(W)$ can be written as

$$l(W) = \sum_l Y^l \ln P(Y^l = 1|X^l,W) + (1-Y^l) \ln P(Y^l = 0|X^l,W)$$

Note here we are utilizing the fact that Y can take only values 0 or 1, so only one of the two terms in the expression will be non-zero for any given Y^l.

To keep our derivation consistent with common usage, we will in this section flip the assignment of the boolean variable Y so that we assign

$$P(Y = 0|X) = \frac{1}{1+\exp(w_0 + \sum_{i=1}^n w_i X_i)} \tag{24}$$

and

$$P(Y = 1|X) = \frac{\exp(w_0 + \sum_{i=1}^n w_i X_i)}{1+\exp(w_0 + \sum_{i=1}^n w_i X_i)} \tag{25}$$

In this case, we can reexpress the log of the conditional likelihood as:

$$\begin{aligned} l(W) &= \sum_l Y^l \ln P(Y^l = 1|X^l,W) + (1-Y^l) \ln P(Y^l = 0|X^l,W) \\ &= \sum_l Y^l \ln \frac{P(Y^l = 1|X^l,W)}{P(Y^l = 0|X^l,W)} + \ln P(Y^l = 0|X^l,W) \\ &= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1+\exp(w_0 + \sum_i^n w_i X_i^l)) \end{aligned}$$

where X_i^l denotes the value of X_i for the lth training example. Note the superscript l is not related to the log likelihood function $l(W)$.

Unfortunately, there is no closed form solution to maximizing $l(W)$ with respect to W. Therefore, one common approach is to use gradient ascent, in which we work with the gradient, which is the vector of partial derivatives. The ith component of the vector gradient has the form

$$\frac{\partial l(W)}{\partial w_i} = \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l,W))$$

where $\hat{P}(Y^l|X^l,W)$ is the Logistic Regression prediction using equations (24) and (25) and the weights W. To accommodate weight w_0, we assume an illusory $X_0 = 1$ for all l. This expression for the derivative has an intuitive interpretation: the term inside the parentheses is simply the prediction error; that is, the difference

between the observed Y^l and its predicted probability! Note if $Y^l = 1$ then we wish for $\hat{P}(Y^l = 1|X^l, W)$ to be 1, whereas if $Y^l = 0$ then we prefer that $\hat{P}(Y^l = 1|X^l, W)$ be 0 (which makes $\hat{P}(Y^l = 0|X^l, W)$ equal to 1). This error term is multiplied by the value of X_i^l, which accounts for the magnitude of the $w_i X_i^l$ term in making this prediction.

Given this formula for the derivative of each w_i, we can use standard gradient ascent to optimize the weights W. Beginning with initial weights of zero, we repeatedly update the weights in the direction of the gradient, changing the ith weight according to

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W))$$

where η is a small constant (e.g., 0.01) which determines the step size. Because the conditional log likelihood $l(W)$ is a concave function in W, this gradient ascent procedure will converge to a global maximum. Gradient ascent is described in greater detail, for example, in Chapter 4 of Mitchell (1997). In many cases where computational efficiency is important it is common to use a variant of gradient ascent called conjugate gradient ascent, which often converges more quickly.

3.3 Regularization in Logistic Regression

Overfitting the training data is a problem that can arise in Logistic Regression, especially when data is very high dimensional and training data is sparse. One approach to reducing overfitting is *regularization*, in which we create a modified "penalized log likelihood function," which penalizes large values of W. One approach is to use the penalized log likelihood function

$$W \leftarrow \arg\max_W \sum_l \ln P(Y^l|X^l, W) - \frac{\lambda}{2}||W||^2$$

which adds a penalty proportional to the magnitude of W. Here λ is a constant that determines the strength of this penalty term. The penalty term can be interpreted as the result of imposing a Normal prior on W, with zero mean, and whose variance is related to $1/\lambda$. Note when $P(W)$ is normal with mean zero, then $\ln P(W)$ will yield a term proportional to $||W||^2$.

Given this penalized log likelihood function, it is easy to rederive the gradient descent rule. The derivative of this penalized log likelihood function is similar to our earlier derivative, with one additional penalty term

$$\frac{\partial l(W)}{\partial w_i} = \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W)) - \lambda w_i$$

which gives us the modified gradient descent rule

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W)) - \eta\lambda w_i \tag{26}$$

In cases where we have prior knowledge about likely values for specific w_i, it is possible to derive a similar penalty term by using a Normal prior on W with a non-zero mean.

3.4 Logistic Regression for Functions with Many Discrete Values

Above we considered using Logistic Regression to learn $P(Y|X)$ only for the case where Y is a boolean variable. More generally, if Y can take on any of the discrete values $\{y_1, \ldots y_K\}$, then the form of $P(Y = y_k|X)$ for $Y = y_1, Y = y_2, \ldots Y = y_{K-1}$ is:

$$P(Y = y_k|X) = \frac{\exp(w_{k0} + \sum_{i=1}^{n} w_{ki} X_i)}{1 + \sum_{j=1}^{K-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji} X_i)} \tag{27}$$

When $Y = y_K$, it is

$$P(Y = y_K|X) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji} X_i)} \tag{28}$$

Here w_{ji} denotes the weight associated with the jth class $Y = y_j$ and with input X_i. It is easy to see that our earlier expressions for the case where Y is boolean (equations (16) and (17)) are a special case of the above expressions. Note also that the form of the expression for $P(Y = y_K|X)$ assures that $[\sum_{k=1}^{K} P(Y = y_k|X)] = 1$.

The primary difference between these expressions and those for boolean Y is that when Y takes on K possible values, we construct $K-1$ different linear expressions to capture the distributions for the different values of Y. The distribution for the final, Kth, value of Y is simply one minus the probabilities of the first $K-1$ values.

In this case, the gradient descent rule with regularization becomes:

$$w_{ji} \leftarrow w_{ji} + \eta \sum_{l} X_i^l (\delta(Y^l = y_j) - \hat{P}(Y^l = y_j|X^l, W)) - \eta \lambda w_{ji} \tag{29}$$

where $\delta(Y^l = y_j) = 1$ if the lth training value, Y^l, is equal to y_j, and $\delta(Y^l = y_j) = 0$ otherwise. Note our earlier learning rule, equation (26), is a special case of this new learning rule, when $K = 2$. As in the case for $K = 2$, the quantity inside the parentheses can be viewed as an error term which goes to zero if the estimated conditional probability $\hat{P}(Y^l = y_j|X^l, W))$ perfectly matches the observed value of Y^l.

4 Relationship Between Naive Bayes Classifiers and Logistic Regression

To summarize, Logistic Regression directly estimates the parameters of $P(Y|X)$, whereas Naive Bayes directly estimates parameters for $P(Y)$ and $P(X|Y)$. We often call the former a discriminative classifier, and the latter a generative classifier.

We showed above that the assumptions of one variant of a Gaussian Naive Bayes classifier imply the parametric form of $P(Y|X)$ used in Logistic Regression. Furthermore, we showed that the parameters w_i in Logistic Regression can

be expressed in terms of the Gaussian Naive Bayes parameters. In fact, if the GNB assumptions hold, then asymptotically (as the number of training examples grows toward infinity) the GNB and Logistic Regression converge toward identical classifiers.

The two algorithms also differ in interesting ways:

- When the GNB modeling assumptions do not hold, Logistic Regression and GNB typically learn different classifier functions. In this case, the asymptotic (as the number of training examples approach infinity) classification accuracy for Logistic Regression is often better than the asymptotic accuracy of GNB. Although Logistic Regression is consistent with the Naive Bayes assumption that the input features X_i are conditionally independent given Y, it is not rigidly tied to this assumption as is Naive Bayes. Given data that disobeys this assumption, the conditional likelihood maximization algorithm for Logistic Regression will adjust its parameters to maximize the fit to (the conditional likelihood of) the data, even if the resulting parameters are inconsistent with the Naive Bayes parameter estimates.

- GNB and Logistic Regression converge toward their asymptotic accuracies at different rates. As Ng & Jordan (2002) show, GNB parameter estimates converge toward their asymptotic values in order $\log n$ examples, where n is the dimension of X. In contrast, Logistic Regression parameter estimates converge more slowly, requiring order n examples. The authors also show that in several data sets Logistic Regression outperforms GNB when many training examples are available, but GNB outperforms Logistic Regression when training data is scarce.

5 What You Should Know

The main points of this chapter include:

- We can use Bayes rule as the basis for designing learning algorithms (function approximators), as follows: Given that we wish to learn some target function $f : X \rightarrow Y$, or equivalently, $P(Y|X)$, we use the training data to learn estimates of $P(X|Y)$ and $P(Y)$. New X examples can then be classified using these estimated probability distributions, plus Bayes rule. This type of classifier is called a *generative* classifier, because we can view the distribution $P(X|Y)$ as describing how to generate random instances X conditioned on the target attribute Y.

- Learning Bayes classifiers typically requires an unrealistic number of training examples (i.e., more than $|X|$ training examples where X is the instance space) unless some form of prior assumption is made. The *Naive Bayes* classifier assumes all attributes describing X are conditionally independent given Y. This assumption dramatically reduces the number of parameters

that must be estimated to learn the classifier. Naive Bayes is a widely used learning algorithm, for both discrete and continuous X.

- When X is a vector of discrete-valued attributes, Naive Bayes learning algorithms can be viewed as linear classifiers; that is, every such Naive Bayes classifier corresponds to a hyperplane decision surface in X. The same statement holds for Gaussian Naive Bayes classifiers if the variance of each feature is assumed to be independent of the class (i.e., if $\sigma_{ik} = \sigma_i$).

- Logistic Regression is a function approximation algorithm that uses training data to directly estimate $P(Y|X)$, in contrast to Naive Bayes. In this sense, Logistic Regression is often referred to as a *discriminative* classifier because we can view the distribution $P(Y|X)$ as directly discriminating the value of the target value Y for any given instance X.

- Logistic Regression is a linear classifier over X. The linear classifiers produced by Logistic Regression and Gaussian Naive Bayes are identical in the limit as the number of training examples approaches infinity, *provided* the Naive Bayes assumptions hold. However, if these assumptions do not hold, the Naive Bayes bias will cause it to perform less accurately than Logistic Regression, in the limit. Put another way, Naive Bayes is a learning algorithm with greater bias, but lower variance, than Logistic Regression. If this bias is appropriate given the actual data, Naive Bayes will be preferred. Otherwise, Logistic Regression will be preferred.

- We can view function approximation learning algorithms as statistical estimators of functions, or of conditional distributions $P(Y|X)$. They estimate $P(Y|X)$ from a sample of training data. As with other statistical estimators, it can be useful to characterize learning algorithms by their bias and expected variance, taken over different samples of training data.

6 Further Reading

Wasserman (2004) describes a Reweighted Least Squares method for Logistic Regression. Ng and Jordan (2002) provide a theoretical and experimental comparison of the Naive Bayes classifier and Logistic Regression.

EXERCISES

1. At the beginning of the chapter we remarked that "A hundred training examples will usually suffice to obtain an estimate of $P(Y)$ that is within a few percent of the correct value." Describe conditions under which the 95% confidence interval for our estimate of $P(Y)$ will be ± 0.02.

2. Consider learning a function $X \rightarrow Y$ where Y is boolean, where $X = \langle X_1, X_2 \rangle$, and where X_1 is a boolean variable and X_2 a continuous variable. State the parameters that must be estimated to define a Naive Bayes classifier in this case. Give the formula for computing $P(Y|X)$, in terms of these parameters and the feature values X_1 and X_2.

3. In section 3 we showed that when Y is Boolean and $X = \langle X_1 \ldots X_n \rangle$ is a vector of continuous variables, then the assumptions of the Gaussian Naive Bayes classifier imply that $P(Y|X)$ is given by the logistic function with appropriate parameters W. In particular:

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

and

$$P(Y = 0|X) = \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

Consider instead the case where Y is Boolean and $X = \langle X_1 \ldots X_n \rangle$ is a vector of *Boolean* variables. Prove for this case also that $P(Y|X)$ follows this same form (and hence that Logistic Regression is also the discriminative counterpart to a Naive Bayes generative classifier over Boolean features).

Hints:

- Simple notation will help. Since the X_i are Boolean variables, you need only one parameter to define $P(X_i|Y = y_k)$. Define $\theta_{i1} \equiv P(X_i = 1|Y = 1)$, in which case $P(X_i = 0|Y = 1) = (1 - \theta_{i1})$. Similarly, use θ_{i0} to denote $P(X_i = 1|Y = 0)$.
- Notice with the above notation you can represent $P(X_i|Y = 1)$ as follows

$$P(X_i|Y = 1) = \theta_{i1}^{X_i} \ (1 - \theta_{i1})^{(1 - X_i)}$$

  Note when $X_i = 1$ the second term is equal to 1 because its exponent is zero. Similarly, when $X_i = 0$ the first term is equal to 1 because its exponent is zero.

4. (based on a suggestion from Sandra Zilles). This question asks you to consider the relationship between the MAP hypothesis and the Bayes optimal hypothesis. Consider a hypothesis space H defined over the set of instances X, and containing just two hypotheses, $h1$ and $h2$ with equal prior probabilities $P(h1) = P(h2) = 0.5$. Suppose we are given an arbitrary set of training data D which we use to calculate the posterior probabilities $P(h1|D)$ and $P(h2|D)$. Based on this we choose the MAP hypothesis, and calculate the Bayes optimal hypothesis. Suppose we find that the Bayes optimal classifier is not equal to either $h1$ or to $h2$, which is generally the case because the Bayes optimal hypothesis corresponds to "averaging over" all hypotheses in H. Now we create a new hypothesis $h3$ which is equal to the Bayes

optimal classifier with respect to H, X and D; that is, $h3$ classifies each instance in X exactly the same as the Bayes optimal classifier for H and D. We now create a new hypothesis space $H' = \{h1, h2, h3\}$. If we train using the same training data, D, will the MAP hypothesis from H' be $h3$? Will the Bayes optimal classifier with respect to H' be equivalent to $h3$? (Hint: the answer depends on the priors we assign to the hypotheses in H'. Can you give constraints on these priors that assure the answers will be yes or no?)

7 Acknowledgements

I very much appreciate receiving helpful comments on earlier drafts of this chapter from the following: Nathaniel Fairfield, Vineet Kumar, Andrew McCallum, Anand Prahlad, Wei Wang, Geoff Webb, and Sandra Zilles.

REFERENCES

Mitchell, T (1997). *Machine Learning*, McGraw Hill.

Ng, A.Y. & Jordan, M. I. (2002). On Discriminative vs. Generative Classifiers: A comparison of Logistic Regression and Naive Bayes, *Neural Information Processing Systems*, Ng, A.Y., and Jordan, M. (2002).

Wasserman, L. (2004). *All of Statistics*, Springer-Verlag.