Naive Bayes classifiers

Kevin P. Murphy

Last updated October 24, 2006

* Denotes advanced material that may be skipped on a first reading.

1 Classifiers

A **classifier** is a function f that maps input feature vectors $x \in \mathcal{X}$ to output class labels $y \in \{1, \ldots, C\}$, where $\mathcal{X}$ is the **feature space**. We will typically assume $\mathcal{X} = \mathbb{R}^D$ or $\mathcal{X} = \{0,1\}^D$, i.e., that the feature vector is a vector of D real numbers or D binary bits, but in general, we may mix discrete and continuous features. We assume the class labels are unordered (categorical) and mutually exclusive. (If we allow an input to belong to multiple classes, this is called a **multi-label** problem.) Our goal is to learn f from a **labeled training set** of N input-output pairs, (x_n, y_n), $n = 1 : N$; this is an example of **supervised learning**.

We will focus our attention on probabilistic classifiers, i.e., methods that return $p(y|x)$. (We will discuss the advantages of having probabilities below.) There are two main ways to do this. The first is to directly learn the function that computes the class posterior $p(y|x)$: this is called a **discriminative model**, since it discriminates between different classes *given* the input. The alternative is to learn the **class-conditional density** $p(x|y)$ for each value of y, and to learn the **class priors** $p(y)$; then one can apply Bayes rule to compute the posterior

$$p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(x|y)p(y)}{\sum_{y'=1}^{C} p(x|y')p(y')} \tag{1}$$

This is called a **generative model**, since it specifies a way to *generate* the feature vectors x for each possible class y. In this chapter, we will focus on generative models; we will consider discriminative models later.

A simpler alternative to generative and discriminative learning is to dispense with probabilities altogether, and to learn a function, called a **discriminant function**, that directly maps inputs to outputs e.g.,

$$f(x) = \hat{y}(x) = \arg\max_y p(y|x) \tag{2}$$

This is perhaps the most popular approach to classification, but below we list some advantages of having the full posterior probability over class labels instead of just an estimate of the most probable class.

2 Advantages of classifiers with probabilistic output *

Reject option With a probability distribution, we can tell if we are uncertain about our prediction, and if so, we can refuse to classify it, and pass the case on to a human expert. (This is important in some medical decision making tasks.) This is called choosing the "reject option"; we will study it later when we come to decision theory. With a discriminant function, we have no idea of the confidence of our answer.

Changing utility functions We can combine the probability distribution with a utility function to minimize our risk. Although we can learn a discriminant to directly minimize our risk, the advantage of

separately learning $p(y|x)$ and then adding on utilities is that, if our utility function changes, we do not need to re-learn $p(y|x)$.

Compensating for class imbalance Consider a binary classification task in which one class is much rarer than the other e.g., cancer vs normal, where only 1 in every 1,000 examples is cancerous. If we train a classifier directly on the imbalanced data, it can trivially get 99.9% accuracy by simply learning the rule $f(x) = normal$, i.e., saying "normal" no matter what the input is. To avoid this degenerate solution, we can train on a balanced training set. Call the resulting model $p_{bal}(y|x)$. At run time, we can modify this to work on the true data distribution as follows. By Bayes rule

$$p_{bal}(y|x) \propto p(x|y)p_{bal}(y) \tag{3}$$

Hence

$$p_{true}(y|x) \propto p(x|y)p_{true}(y) \propto \frac{p_{bal}(y|x)}{p_{bal}(y)}p_{true}(y) \tag{4}$$

Converting a posterior into a likelihood in this way is called the **scaled likelihood trick**.

Combining models Suppose we have two kinds of feature vectors, x_1 and x_2 (e.g., X-ray images and blood tests). Rather than building a large monolithic model, it might be more practical to learn two separate classifiers, $p(y|x_1)$ and $p(y|x_2)$ and then to combine them. The key to combining outputs from different informations sources (which is called **sensor fusion**) is to know how reliable each source is. This is precisely what the posterior probabilities tell us. A simple way to combine the systems is to assume the features are conditionally independent given the class labels:

$$p(x_1, x_2|y) = p(x_1|y)p(x_2|y) \tag{5}$$

This is the Naive Bayes assumption which we shall explain in more detail below. Given this assumption, we can use the scaled likelihood trick as follows

$$\begin{aligned} p(y|x_1, x_2) &\propto p(x_1, x_2|y)p(y) &(6)\\ &\propto p(x_1|y)p(x_2|y)p(y) &(7)\\ &\propto \frac{p(y|x_1)}{p(y)}\frac{p(y|x_2)}{p(y)}p(y) &(8)\\ &\propto \frac{p(y|x_1)p(y|x_2)}{p(y)} &(9)\end{aligned}$$

3 Generative classifiers

3.1 Class prior

If we assume that the number of classes C is small, then we can easily estimate the class prior $p(y)$ by treating y as a multinomial random variable:

$$p(y = c|\pi) = \pi_c \tag{10}$$

where π is a vector of class probabilities. The MLE is simply

$$\hat{\pi}_c^{MLE} = \frac{\sum_{n=1}^{N} I(y_n = c)}{N} = \frac{N_c}{N} \tag{11}$$

where N_c is the number of training examples that have class label c. If we use a Dirichlet prior, the posterior mean estimate is

$$\hat{\pi}_c^{mean} = \frac{N_c + \alpha_c}{N + \alpha} \tag{12}$$

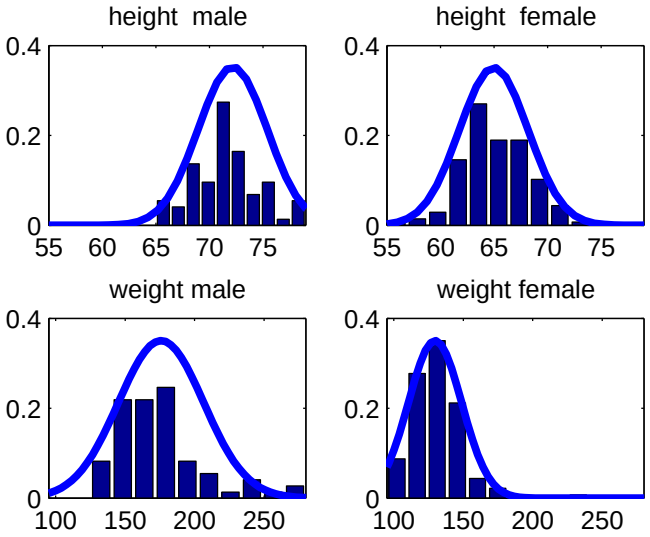

Figure 1: Univariate Gaussian class-conditional distributions for two features (height and weight) and two classes (male and female).

If the prior is uniform, $\alpha_c = 1$, we get

$$\hat{\pi}_c^{mean} = \frac{N_c + 1}{N + C} \tag{13}$$

Alternatively, if we use $\alpha_c = 2$, the MAP estimate is

$$\hat{\pi}_c^{MAP} = \frac{N_c + 1}{N + C} \tag{14}$$

As usual, this avoids the problems with zero counts that the MLE suffers from.

3.2 Class conditional densities

The main issue is how to specify the class conditional density $p(x|y)$. The usual assumption is that the parameters of each such conditional distribution are independent. Hence we can estimate the $p(x|y)$ separately for each value of y, so we just have so solve C separate density estimation problems.

If $x \in \mathbb{R}$ is a 1-dimensional scalar feature, such as height or weight, one possibility would be to use a Gaussian class-conditional density:

$$p(x|y = c, \theta_c) = \mathcal{N}(x|\mu_c, \Sigma_c) \tag{15}$$

where the mean and variance depend on the class c (see Figure 1). Similarly, if $x \in \{0, 1\}$ is a 1-dimensional binary feature, then we can use a Bernoulli class-conditional density:

$$p(x|y = c, \theta_c) = \theta_c^x (1 - \theta_c)^{1-x} \tag{16}$$

If $x \in \{1, \ldots, K\}$, we can use a multinomial class-conditional density

$$p(x|y = c, \theta_c) = \prod_{k=1}^{K} \theta_{ck}^{I(x=k)} \tag{17}$$

Below we discuss one way to extend these models to handle multiple features, x_i, for $i = 1 : D$.

3.3 The naive Bayes assumption

The **naive Bayes** or **idiot Bayes** assumption is that all the features are conditionally independent given the class label:

$$p(x|y=c) = \prod_{i=1}^{D} p(x_i|y=c) \tag{18}$$

Even though this is usually false (since features are usually dependent), the resulting model is easy to fit and works surprisingly well. In the case of Gaussian data, we get

$$p(x|y=c,\theta_c) = \prod_{i=1}^{D} \mathcal{N}(x_i|\mu_{ic},\sigma_{ic}) \tag{19}$$

so we just have to estimate $C \times D$ separate Gaussian parameters, μ_{ic}, σ_{ic}. In the case of binary data, we get

$$p(x|y=c,\theta_c) = \prod_{i=1}^{D} Be(x_i|\theta_{ic}) \tag{20}$$

so we just have to estimate $C \times D$ separate Bernoulli parameters, θ_{ic}. We discuss the case of binary (and K-ary) data in more detail below.

3.4 Plug-in classifier

The full Bayesian posterior predictive density on the class label Y given an input X and the training data $\mathcal{D}$ is given by

$$p(y=c|x,D) \propto \int\int p(x|y=c,\theta_c)p(y=c|\pi)p(\theta_c)p(\pi)d\theta_c d\pi \tag{21}$$

This is often approximated using a plug-in approximation as follows

$$p(y=c|x,D) \approx p(y=c|x,\hat{\theta},\hat{\pi}) \propto p(x|y=c,\hat{\theta}_c)p(y=c|\hat{\pi}) \tag{22}$$

In the case where all variables have multinomial distributions with Dirichlet priors, if we use the posterior mean as our parameter estimate, then the plug-in approach will be exact.

4 Naive Bayes for document classifcation

Suppose we want to classify a document into one of C classes (e.g., $C = 2$ and the classes are "spam" and "non-spam"). A very simple representation, called the **bag of words** model, is to ignore word ordering and to just count the number of times each word occurs. Suppose there are D words in the language (we can always enforce this by mapping all words outside the vocabularly into a special symbol such as **unk**, for unknown word). Then a document can be represented as a p-vector of counts (a **word frequency histogram**). Let $X = k$ mean the word occurs exactly k times, for $k = 0 : K-1$; for simplicity, we will say this word has count k. (If the word occurs more than $K-1$ times in a document, we will just treat it as if it occured $K-1$ times; here K is a user chosen upper bound.) In this case, we can represent the class-conditional density as a product of multinomials:

$$p(x|Y=c,\theta) = \prod_{i=1}^{D}\prod_{k=1}^{K} \theta_{ick}^{I(x_i=k)} \tag{23}$$

where $\theta_{ick} = p(X_i = k|Y=c)$ is the probability of observing the i'th word having count k given that the class is c. (Obviously $\sum_{k=1}^{K} \theta_{ick} = 1$ for all i, c.) The intuition behind this is that the number of times a word occurs in a document may provide some information abour what type of document it is. (A better

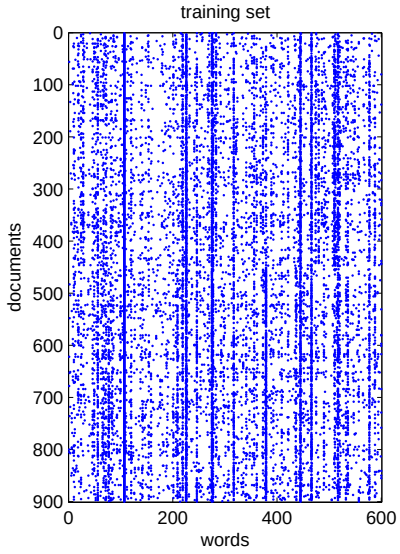

Figure 2: Training data for document classification visualized using matlab's `spy` command, which is useful for sparse matrices. A black dot in row i column j means document i contains word j at least once.

representation is to use a Poisson distribution on the word counts, but this is beyond the scope of this chapter.)

An even simpler representation is to just to represent whether the word occurs or not, and not worry about how many times. Let us call this binary feature vector x. In this case, we can represent the class-conditional densities as a product of Bernoulli distributions:

$$p(x|Y=c,\theta) = \prod_{i=1}^{D} \theta_{ic}^{x_i}(1-\theta_{ic})^{1-x_i} \tag{24}$$

where θ_{ic} is the probability word i occurs in class c, $x_i = 1$ means word i is present, and $x_i = 0$ otherwise. See for example Figure 2 for binary count data, and Figure 3 for the class-conditional densities. Despite the simplicity of this model, it works surprisingly well, and is the basis of most email **spam filters**.

4.1 Parameter estimation

The MLE estimate of the parameters is gotten by normalizing the counts

$$\hat{\theta}_{ick}^{MLE} = \frac{N_{ick}}{\sum_{k'} N_{ick'}} \tag{25}$$

where N_{ick} is the number of times word i occurs exactly k times in documents of class c:

$$N_{ick} = \sum_{n:y_n=c} I(\#_{n,i} = k) \tag{26}$$

where $\#_{n,i}$ is the number of times word i occurs in document n:

$$\#_{n,i} = \sum_{w \in n} I(w = i) \tag{27}$$

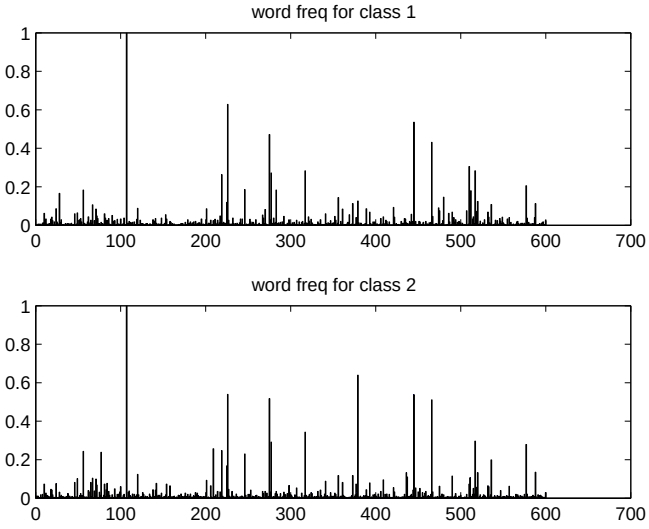

Figure 3: Class conditional densities $p(x_i = 1|c)$ for two document classes.

So $\#_{ni} \in \{0, \ldots, K-1\}$ (we truncate the count at $K-1$). To avoid zero counts (e.g., of rare words), we can use a Dirichlet prior; then the posterior mean becomes

$$\hat{\theta}_{ick}^{mean} = \frac{N_{ick} + \alpha_k}{\sum_{k'} N_{ick'} + \alpha_{k'}} \tag{28}$$

Typically we use $\alpha_k = 1$.

In the simpler binary model, this becomes

$$\hat{\theta}_{ic} = \frac{N_{ic} + 1}{N_c + 2} \tag{29}$$

where N_{ic} is the number of times word i occurs (one or more times) in documents of class c and N_c is the number of documents of class c:

$$N_{ic} = \sum_{n:y_n=c} I(\#_{ni} > 0) \tag{30}$$

$$N_c = \sum_n I(y_n = c) \tag{31}$$

4.2 Form of the class posterior

We can derive an analytic expression for the posterior $p(y|x)$ by using Bayes rule. Consider the multinomial model. Let $X_{ik} = 1$ if $X_i = k$, a 1-of-K encoding. Then the class conditional density can be written

$$p(x|Y = c, \theta) = \prod_{i=1}^{D} \prod_{k=1}^{K} \theta_{ick}^{I(x_i=k)} = \prod_{i=1}^{D} \prod_{k=1}^{K} \theta_{ick}^{x_{ik}} \tag{32}$$

Hence the class posterior becomes

$$p(Y=c|x,\theta,\pi) = \frac{p(x|y=c)p(y=c)}{\sum_{c'} p(x|y=c')p(y=c')} \tag{33}$$

$$= \frac{\exp[\log p(x|y=c) + \log p(y=c)]}{\sum_{c'} \exp[\log p(x|y=c') + \log p(y=c')]} \tag{34}$$

$$= \frac{\exp\left[\log \pi_c + \sum_i \sum_k x_{ik} \log \theta_{ick}\right]}{\sum_{c'} \exp\left[\log \pi_{c'} + \sum_i \sum_k x_{ik} \log \theta_{i,c',k}\right]} \tag{35}$$

If we define

$$x' = [1, x_{11}, \ldots, x_{1K}, \ldots, x_{D1}, \ldots, x_{DK}] \tag{36}$$

$$\beta_c = [\log \pi_c, \log \theta_{1c1}, \ldots, \log \theta_{1cK}, \ldots, \log \theta_{Dc1}, \ldots, \log \theta_{DcK}] \tag{37}$$

(so the state index k varies more rapidly than the dimension i), then

$$p(Y=c|x,\beta) = \frac{\exp[\beta_c^T x']}{\sum_{c'} \exp[\beta_{c'}^T x']} \tag{38}$$

This is called a **softmax** distribution. We will discuss this more later.

In the case of binary features, $K=2$, so things simplify as follows. Let $\theta_{ic1} = \theta_{ic} = p(X_i = 1|Y=c)$, so $\theta_{ic2} = 1 - \theta_{ic1}$. Also, let $x'_{i1} = 1$ iff $x_i = 1$ and $x'_{i2} = 1$ iff $x_i = 0$. Then we have

$$x' = [1, I(x_1=1), I(x_1=0), \ldots, I(x_D=1), I(x_D=0)] \tag{39}$$

$$\beta_c = [\log \pi_c, \log \theta_{1c}, \log(1-\theta_{1c}), \ldots, \log \theta_{Dc}, \log(1-\theta_{Dc})] \tag{40}$$

In the case of two classes ($Y \in \{1,2\}$), the expressions simplify as follows

$$p(Y=1|x,\theta) = \frac{e^{\beta_1^T x'}}{e^{\beta_1^T x'} + e^{\beta_2^T x'}} \tag{41}$$

$$= \frac{1}{1 + e^{(\beta_2 - \beta_1)^T x'}} \tag{42}$$

$$= \frac{1}{1 + e^{w^T x'}} \tag{43}$$

$$= \sigma(w^T x') \tag{44}$$

where we have defined $w = \beta_1 - \beta_2$ and $\sigma(\cdot)$ is the **logistic** or **sigmoid** function

$$\sigma(u) = \frac{1}{1+e^{-u}} \tag{45}$$

We will study this more later.

5 Log-sum-exp trick

One practical issue that arises when applying generative classifiers is that $p(\vec{x}|Y)$ is often a very small number: the probability of observing any particular high-dimensional vector is small, since we have $\sum_{\vec{x}} p(\vec{x}|Y) = 1$. This can lead to **numerical underflow**. When trying to compute

$$P(Y=c|\vec{x}) = \frac{P(\vec{x}|Y=c)P(Y=c)}{\sum_{c'=1}^{C} P(\vec{x}|Y=c')P(Y=c')} \tag{46}$$

```
function s = logsumexp(b, dim)
% s = logsumexp(b) by Tom Minka
% Returns s(i) = log(sum(exp(b(:,i))))  while avoiding numerical underflow.
% s = logsumexp(b, dim) sums over dimension 'dim' instead of summing over rows

if nargin < 2 % if 2nd argument is missing
  dim = 1;
end

[B, junk] = max(b,[],dim);
dims = ones(1,ndims(b));
dims(dim) = size(b,dim);
b = b - repmat(B, dims);
s = B + log(sum(exp(b),dim));
i = find(~finite(B));
if ~isempty(i)
  s(i) = B(i);
end
```

Figure 4: Matlab code for `logsumexp.m`.

one could take logs as follows:

$$b_c \stackrel{\text{def}}{=} \log[P(\vec{x}|Y=c)P(Y=c)] \tag{47}$$

$$\log P(Y=c|\vec{x}) = b_c - \log \sum_{c'=1}^{C} e^{b_{c'}} \tag{48}$$

However, $\log \sum_c e^{b_c}$ will underflow, because $e^{b_c} = P(\vec{x}, Y=c)$ is very small, by assumption. One should therefore use the **log-sum-exp** trick:

$$\log \sum_c e^{b_c} = \log \left[(\sum_c e^{b_c}) e^{-B} e^{B} \right] \tag{49}$$

$$= \log \left[(\sum_c e^{b_c - B}) e^{B} \right] \tag{50}$$

$$= \left[\log (\sum_c e^{b_c - B}) \right] + B \tag{51}$$

where $B = \max_c b_c$. For example,

$$\log(e^{-120} + e^{-121}) = \log\left(e^{-120}(e^0 + e^{-1})\right) = \log(e^0 + e^{-1}) - 120 \tag{52}$$

See Figure 4 for some matlab code.