Learning from Data 1
Naive Bayes

David Barber
dbarber@anc.ed.ac.uk
course page : http://anc.ed.ac.uk/∼dbarber/lfd1/lfd1.html

1 Why Naive Bayes?

Naive Bayes is one of the simplest density estimation methods from which we can form one of the standard classification methods in machine learning. Its fame is partly due to the following properties:

- Very easy to program and intuitive
- Fast to train and to use as a classifier
- Very easy to deal with missing attributes
- Very popular in certain fields such as computational linguistics/NLP

However, despite the simplicity of Naive Bayes, there are some pitfalls that need to be avoided, as we will describe. The pitfalls usually made are due to a poor understanding of the central assumption behind Naive Bayes, namely conditional independence. Before we explain how to use conditional independence to form a classifier, we concentrate on explaining the basic assumption of conditional independence.

2 Understanding Conditional Independence

Consider a general probability distribution of two variables, $p(x_1, x_2)$. Using Bayes rule, without loss of generality, we can write

$$p(x_1, x_2) = p(x_1|x_2)p(x_2) \tag{2.1}$$

Similarly, if we had another "class" variable, c, we can write, using Bayes rule :

$$p(x_1, x_2|c) = p(x_1|x_2, c)p(x_2|c) \tag{2.2}$$

In the above expression, we have not made any assumptions at all. Consider now the term $p(x_1|x_2, c)$. If knowledge of c is sufficient to determine how x_1 will be distributed, we don't need to know the state of x_2. That is, we may write $p(x_1|x_2, c) = p(x_1|c)$. For example, we may write the general statement:

$$p(cloudy, windy|storm) = p(cloudy|windy, storm)p(windy|storm) \tag{2.3}$$

where, for example, each of the variables can take the values 'yes' or 'no', and now further make the assumption $p(cloudy|windy, storm) = p(cloudy|storm)$ so that the distribution becomes

$$p(cloudy, windy|storm) = p(cloudy|storm)p(windy|storm) \tag{2.4}$$

We can generalise the situation of two variables to a conditional independence assumption for a set of variables $x_1, \ldots, x_N$, conditional on another variable c:

$$p(\mathbf{x}|c) = \prod_{i=1}^{N} p(x_i|c) \tag{2.5}$$

A further example may help to clarify the assumptions behind conditional independence. EasySell.com considers that its customers conveniently fall into two groups – the 'young' or 'old'. Based on only this information, they build general customer profiles for product preferences. Easysell.com *assumes* that, given the knowledge that a customer is either 'young' or 'old', this is *sufficient* to determine whether or not a customer will like a product, *independent* of their likes or dislikes for any other products. Thus, given that a customer is 'young', she has a 95% chance to like Radio1, a 5% chance to like Radio2, a 2% chance to like Radio3 and a 20% chance to like Radio4. Similarly, they model that an 'old' customer has a 3% chance to like Radio1, an 82% chance to like Radio2, a 34% chance to like Radio3 and a 92% chance to like Radio4. Mathematically, we would write

$$p(R1, R2, R3, R4|age) = p(R1|age)p(R2|age)p(R3|age)p(R4|age) \tag{2.6}$$

where each of the variables $R1, R2, R3, R4$ can take the values either 'like' or 'dislike', and the 'age' variable can take the value either 'young' or 'old'. Thus the information about the age of the customer is so powerful that this determines the individual product preferences without needing to know anything else. Clearly, this is a rather strong assumption, but a popular one, and sometimes leads to surprisingly good results.

In this chapter, we will take the conditioning variable to represent the class of the datapoint $\mathbf{x}$. Coupled then with a suitable choice for the conditional distribution $p(x_i|c)$, we can then use Bayes rule to form a classifier. In this chapter, we will consider two cases of different conditional distributions, one appropriate for discrete data and the other for continuous data. Furthermore, we will demonstrate how to learn any free parameters of these models.

3 Are they Scottish?

Consider the following vector of attributes:

$$\text{(likes shortbread, likes lager, drinks whiskey, eats porridge, watched England play football)}^T \tag{3.1}$$

A vector $\mathbf{x} = (1, 0, 1, 1, 0)^T$ would describe that a person likes shortbread, does not like lager, drinks whiskey, eats porridge, and has not watched England play football. Together with each vector $\mathbf{x}^\mu$, there is a class label describing the nationality of the person: Scottish, or English. We wish to classify a new vector $\mathbf{x} = (1, 0, 1, 1, 0)^T$ as either Scottish or English. We can use Bayes rule to calculate the probability that $\mathbf{x}$ is Scottish or English:

$$p(S|\mathbf{x}) = \frac{p(\mathbf{x}|S)p(S)}{p(\mathbf{x})} \tag{3.2}$$

$$p(E|\mathbf{x}) = \frac{p(\mathbf{x}|E)p(E)}{p(\mathbf{x})} \tag{3.3}$$

Since we must have $p(S|\mathbf{x}) + p(E|\mathbf{x}) = 1$, we could also write

$$p(S|\mathbf{x}) = \frac{p(\mathbf{x}|S)p(S)}{p(\mathbf{x}|S)p(S) + p(\mathbf{x}|E)p(E)} \tag{3.4}$$

It is straightforward to show that the "prior" class probability $p(S)$ is simply given by the fraction of people in the database that are Scottish, and similarly $p(E)$ is given as the fraction of people in the database that are English. What about $p(\mathbf{x}|S)$? This is where our density model for $\mathbf{x}$ comes in. In the previous chapter, we looked at a using a Gaussian distribution. Here we will make a different, very strong *conditional independence* assumption:

$$p(\mathbf{x}|S) = p(x_1|S)p(x_2|S)\dots p(x_5|S) \tag{3.5}$$

What this assumption means is that knowing whether or not someone is Scottish, we don't need to know anything else to calculate the probability of their likes and dislikes.

Matlab code to implement Naive Bayes on a small dataset is written below, where each row of the datasets represents a (row) vector of attributes of the form equation (3.1).

```
% Naive Bayes using Bernoulli Distribution

xE=[0 1 1 1 0 0;  % english
    0 0 1 1 1 0;
    1 1 0 0 0 0;
    1 1 0 0 0 1;
    1 0 1 0 1 0];

xS=[1 1 1 1 1 1 1; % scottish
    0 1 1 1 1 0 0;
    0 0 1 0 0 1 1;
    1 0 1 1 1 1 0;
    1 1 0 0 1 0 0];

pE = size(xE,2)/(size(xE,2) + size(xS,2)); pS =1-pE; % ML class priors pE = p(c=E), pS=p(c=S)

mE = mean(xE')'; % ML estimates of p(x=1|c=E)
mS = mean(xS')'; % ML estimates of p(x=1|c=S)

x=[1 0 1 1 0]'; % test point

npE = pE*prod(mE.^x.*(1-mE).^(1-x)); % p(x,c=E)
npS = pS*prod(mS.^x.*(1-mS).^(1-x)); % p(x,c=S)

pxE = npE/(npE+npS) % probability that x is english
```

Based on the training data in the code above, we have the following : $p(x_1 = 1|E) = 1/2$,$p(x_2 = 1|E) = 1/2$,$p(x_3 = 1|E) = 1/3$,$p(x_4 = 1|E) = 1/2$,$p(x_5 = 1|E) = 1/2$, $p(x_1 = 1|S) = 1$,$p(x_2 = 1|S) = 4/7$,$p(x_3 = 1|S) = 3/7$,$p(x_4 = 1|S) = 5/7$,$p(x_5 = 1|S) = 3/7$ and the prior probabilities are $p(S) = 7/13$ and $p(E) = 6/13$.

For $\mathbf{x}^* = (1, 0, 1, 1, 0)^T$, we get

$$p(S|\mathbf{x^*}) = \frac{1 \times \frac{3}{7} \times \frac{3}{7} \times \frac{5}{7} \times \frac{4}{7} \times \frac{7}{13}}{1 \times \frac{3}{7} \times \frac{3}{7} \times \frac{5}{7} \times \frac{4}{7} \times \frac{7}{13} + \frac{1}{2} \times \frac{1}{2} \times \frac{1}{3} \times \frac{1}{2} \times \frac{1}{2} \times \frac{6}{13}} \tag{3.6}$$

which is 0.8076. Since this is greater than 0.5, we would classify this person as being Scottish.

3.1 Further Issues

Consider trying to classify the vector $\mathbf{x} = (0, 1, 1, 1, 1)^T$. In the training data, all Scottish people say they like shortbread. This means that $p(\mathbf{x}, S) = 0$, and hence that $p(S|\mathbf{x}) = 0$. This demonstrates a difficulty with sparse data – very extreme class probabilities can be made. One way to ameliorate this situation is to smooth the probabilities in some way, for example by adding a certain small number M to the frequency counts of each class:

$$p(x_i = 1|c) = \frac{\text{number of times } x_i = 1 \text{ for class c} + M}{\text{number of times } x_i = 1 \text{ for class c} + M + \text{number of times } x_i = 0 \text{ for class c} + M} \tag{3.7}$$

This ensures that there are no zero probabilities in the model.

3.2 Gaussians

Fitting continuous data is also straightforward using Naive Bayes. For example, if we were to model each attributes distribution as a Gaussian, $p(x_i|c) = N(\mu_i, \sigma_i)$, this would be exactly equivalent to using the conditional Gaussian density estimator in the previous chapter by replacing the covariance matrix with all elements zero except for those on the diagonal.

3.3 Text Classification

Naive Bayes has been often applied to classify documents in classes. We will outline here how this is done. Refer to a computational linguistics course for details of how exactly to do this.

Consider a set of documents about politics, and a set about sport. We search through all documents to find the, say 100 most commonly occuring words. Each document is then represented by a 100 dimensional vector representing the number of times that each of the words occurs in that document – the so called 'bag of words' representation (this is clearly a very crude assumption since it does not take into account the order of the words). We then fit a Naive Bayes model by fitting a distribution of the number of occurrences of each word for all the documents of, first sport, and then politics. This then completes the model.

Bag of words

The reason Naive Bayes may be able to classify documents reasonably well in this way is that the conditional independence assumption is not so silly : if we know people are talking about politics, this perhaps is almost sufficient information to specify what kinds of other words they will be using – we don't need to know anything else. (Of course, if you want ultimately a more powerful text classifier, you need to relax this assumption).

4 Pitfalls with Naive Bayes

So far we have described how to implement Naive Bayes for the case of binary attributes and also for the case of Gaussian continuous attributes. However, very often, the software that people seem to commonly use requires that the data is in the form of binary attributes. It is in the transformation of non-binary data to a binary form that a common mistake occurs.

Consider the following attribute : age. In a survey, a person's age is marked down using the variable $a \in 1, 2, 3$. $a = 1$ means the person is between 0 and 10 years old, $a = 2$ means the person is between 10 and 20 years old, $a = 3$ means the person is older than 20. Perhaps there would be other attributes for the data, so that each data entry is a vector of two variables $(a, b)^T$.

1-of-M encoding

One way to transform the variable a into a binary representation would be to use three binary variables (a_1, a_2, a_3). Thus, $(1, 0, 0)$ represents $a = 1$, $(0, 1, 0)$ represents $a = 2$ and $(0, 0, 1)$ represents $a = 3$. This is called $1-of-M$ coding since only 1 of the binary variables is active in encoding the M states. The problem here is that this encoding, by construction, means that the variables a_1, a_2, a_3 are *dependent* – for example, if we know that $a_1 = 1$, we know that $a_2 = 0$ and $a_3 = 0$. Regardless of any possible conditioning, these variables will always remain completely dependent, contrary to the assumption of Naive Bayes. This mistake, however, is widespread – please help preserve a little of my sanity by not making the same error. The correct approach is to simply use variables with many states – the multinomial rather than binomial distribution. This is straightforward and left as an exercise for the interested reader.

5 Estimation using Maximum Likelihood : Bernoulli Process

In this section we formally derive how to learn the parameters in a Naive Bayes model from data. The results are intuitive, and indeed, we have already made use of them in the previous sections. However, it is instructive to carry out this procedure and some light can be cast also on the nature of the decision boundary (at least for the case of binary attributes).

Consider a dataset $X = \{\mathbf{x}^\mu, \mu = 1, \ldots, P\}$ of binary attributes. That is $x_i^\mu \in \{0, 1\}$. Each datapoint $\mathbf{x}^\mu$ has an associated class label c^μ. Based upon the class label, we can split the inputs into those that belong to each class : $X^c = \{\mathbf{x}|\mathbf{x} \text{ is in class } c\}$. We will consider here only the case of two classes (this is called a Bernoulli process – the case of more classes is also straightforward and called the multinomial process). Let the number of datapoints from class $c = 0$ be n_0 and the number from class $c = 1$ be n_1.

For each class of the two classes, we then need to estimate the values $p(x_i = 1|c) \equiv \theta_i^c$. (The other probability, $p(x_i = 0|c)$ is simply given from the normalisation requirement, $p(x_i = 0|c) = 1 - p(x_i = 1|c) = 1 - \theta_i^c$). Using the standard assumption that the data is generated identically and independently, the likelihood of the model generating the dataset X^c (the data X belonging to class c) is

$$p(X^c) = \prod_{\mu \text{ from class c}} p(\mathbf{x}^\mu|c) \tag{5.1}$$

Using our conditional independence assumption

$$p(\mathbf{x}|c) = \prod_i p(x_i|c) = \prod_i (\theta_i^c)^{x_i}(1-\theta_i^c)^{1-x_i} \tag{5.2}$$

(remember that in each term in the above expression, x_i is either 0 or 1 and hence, for each i term in the product, only one of the two factors will contribute, contributing a factor θ_i^c if $x_i = 1$ and $1 - \theta_i^c$ if $x_i = 0$). Putting this all together, we can find the log likelihood

$$L(\boldsymbol{\theta}^c) = \sum_{i,\mu} x_i^\mu \log \theta_i^c + (1 - x_i^\mu)\log(1-\theta_i^c) \tag{5.3}$$

Optimising with respect to $\theta_i^c \equiv p(x_i = 1|c)$ (differentiate with respect to θ_i^c and equate to zero) gives

$$p(x_i = 1|c) = \frac{\text{number of times } x_i = 1 \text{ for class c}}{(\text{number of times } x_i = 1 \text{ for class c}) + (\text{number of times } x_i = 0 \text{ for class c})} \tag{5.4}$$

A similar Maximum Likelihood argument gives the intuitive result:

$$p(c) = \frac{\text{number of times class c occurs}}{\text{total number of data points}} \tag{5.5}$$

5.1 Classification Boundary

If we just wish to find the most likely class for a new point $\mathbf{x}$, we can compare the log probabilities, classifying $\mathbf{x}^*$ as class 1 if

$$\log p(c = 1|\mathbf{x}^*) > \log p(c = 0|\mathbf{x}^*) \tag{5.6}$$

Using the definition of the classifier, this is equivalent to (since the normalisation constant $-\log p(\mathbf{x}^*)$ can be dropped from both sides)

$$\sum_i \log p(x_i^*|c = 1) + \log p(c = 1) > \sum_i \log p(x_i^*|c = 0) + \log p(c = 0) \tag{5.7}$$

Using the binary encoding $x_i \in \{0, 1\}$, this is : classify $\mathbf{x}^*$ as class 1 if

$$\sum_i \left\{x_i^* \log \theta_i^1 + (1 - x_i^*)\log(1 - \theta_i^1)\right\} + \log p(c = 1) > \sum_i \left\{x_i^* \log \theta_i^0 + (1 - x_i^*)\log(1 - \theta_i^0)\right\} + \log p(c = 0) \tag{5.8}$$

Note that this decision rule can be expressed in the form : classify $\mathbf{x}^*$ as class 1 if $\sum_i w_i x_i^* + a > 0$ for some suitable choice of weights w_i and constant a (the reader is invited to find the explicit values of these weights). The interpretation of this is that $\mathbf{w}$ specifies a hyperplane in the $\mathbf{x}$ space and $\mathbf{x}^*$ is classified as a 1 if it lies on one side of the hyperplane. We shall talk about other such “linear” classifiers in a later chapter.