Chapter 4
Bayes classifier and naive Bayes classifier

I) The Bayes classifier

We are given a set of training points from C classes. Our aim is to be able to compute the predictive probabilities for each of C potential classes. These probabilities can then form the basis of a decision-making process or be used to compute an expectation.

From Bayes' rule, we obtain the expression for the predictive probability:

$$\mathbb{P}(y_{new} = c \mid x_{new}, X, y) = \frac{\mathbb{P}(x_{new} \mid y_{new} = c, X, y)\,\mathbb{P}(y_{new} = c \mid X, y)}{\mathbb{P}(x_{new} \mid X, y)}$$

We expand $\mathbb{P}(x_{new} \mid X, y)$ by summing over all the C possible classes and we obtain:

$$\mathbb{P}(y_{new} = c \mid x_{new}, X, y) = \frac{\mathbb{P}(x_{new} \mid y_{new} = c, X, y)\,\mathbb{P}(y_{new} = c \mid X, y)}{\sum_{c'=1}^{C} \mathbb{P}(x_{new} \mid y_{new} = c', X, y)\,\mathbb{P}(y_{new} = c' \mid X, y)}$$

We are left with the task of defining $\mathbb{P}(x_{new} \mid y_{new} = c, X, y)$ and $\mathbb{P}(y_{new} = c \mid X, y)$

1) Likelihood (class conditional distribution)

The likelihood term $\mathbb{P}(x_{new} | y_{new} = c, X, y)$ is a distribution specific to the c-th class (and cond. on $y_{new} = c$), evaluated at x_{new}. To create a Bayes classifiers, we ned to define C of these class-conditional distributions. We could use the same type of distribution for each class (or not)

Once we have chosen the distribution for the c-th class, we need the choose its parameters. For ex, if we choose a Gaussian distribution, we need to choose the mean and covariance. This will be done using the training data (ex: using a MLE).

2) Prior class distribution

the other quantity of interest is $\mathbb{P}(y_{new} = c | X, y)$. this is the probability that the object belongs to class c conditioned on just the training data X, y. It enables us to specify any prior beliefs in the class before we see it. the technical restrictions are of course:

$$\begin{cases} \mathbb{P}(y_{new} = c | X, y) > 0 \\ \sum_{c'} \mathbb{P}(y_{new} = c' | X, y) = 1 \end{cases}$$

Two popular choices are

$$\begin{cases} i) \text{ uniform prior} : \mathbb{P}(y_{new} = c | X, y) = \frac{1}{C} \\ ii) \text{ class size prior} : \mathbb{P}(y_{new} = c | X, y) = \frac{N_c}{N} \end{cases}$$

(N_c = number of objects of class c in the training set
N = number of objects in the training set)

II) Naive Bayes classifier

In this section, we discuss how to classify vectors of discrete-valued features, $x \in \{1, \dots, K\}^D$. We have to specify the class conditional distribution $(\mathbb{P}(x_{new} | y_{new} = c, X, y))$. The simplest approach is to assume the features are conditionally independant given the class label. This allows us to write:

$$\mathbb{P}(x_{new} | y_{new} = c, X, y) = \prod_{j=1}^{D} \mathbb{P}(x_j^{new} | y^{new} = c, X, y)$$

The resulting model is called a naive Bayes classifier (NBC). The model is called "naive" because we do not expect the features to be independ, even conditioned on the class label. However it often works well in practice. One reason is that the model is quite simple (it has only $O(CD)$ parameters, for C classes and D features) and hence is relatively immune to overfitting.

Examples:

* In the case of real-valued features, we can use a Gaussian distribution: $\mathbb{P}(x_{new} | \underbrace{y_{new}}_{c}, X, y)$

$$\prod_{j=1}^{D} \mathcal{N}(x_j^{new} | \mu_{jc}, \sigma_{jc}^2).$$

* In the case of binary features ($x_j \in \{0, 1\}$), we can use a Bernoulli distribution: $\mathbb{P}(x_{new} | y_{new} = c, X, y)$

$$\prod_{j=1}^{D} \text{Ber}(x_j | \mu_{jc})$$

(This is sometimes called a multivariate Bernoulli naive Bayes.)

* In the case of categorical features ($x_j \in \{1, \dots, K\}$), we can use the multinoulli distribution:

$$\mathbb{P}(x_{new} \mid y_{new} = c, X, y) = \prod_{j=1}^{D} \text{Cat}(x_j^{new} \mid \mu_{jc})$$

$$\hookrightarrow \mathbb{P}(x_j^{new} = k) = \mu_{jc}(k)$$

$$\left(\text{with } \sum_{k=1}^{K} \mu_{jc}(k) = 1\right)$$

1) MLE for NBC

Remember: $\mathcal{D} = (X, y)$

Suppose that $\begin{cases} \mathbb{P}(y=c) = \pi_c \quad (\pi_1 + \dots + \pi_C = 1), \\ \mathbb{P}(x \mid y=c) \text{ is a distribution parametrized by } \theta_c. \end{cases}$

We then have: $(\forall i)$

$$\mathbb{P}(x_i, y_i \mid \theta) = \prod_{c=1}^{C} \pi_c^{\mathbb{1}(y_i = c)} \prod_{j=1}^{D} \prod_{c=1}^{C} \mathbb{P}(x_{ij} \mid \theta_{jc})^{\mathbb{1}(y_i = c)}.$$

Hence the log-likelihood is:

$$\log \mathbb{P}(\mathcal{D} \mid \theta) = \sum_{c=1}^{C} N_c \log(\pi_c) + \sum_{j=1}^{D} \sum_{c=1}^{C} \sum_{i : y_i = c} \log \mathbb{P}(x_{ij} \mid \theta_{jc})$$

N_c: number of type c in the training data

We have a sum of terms concerning π plus a sum of DC terms concerning the θ_{jc}'s. Hence, we can optimize those parameters separately.

The MLE for the class prior is: $\hat{\pi}_c = \frac{N_c}{N}$

The MLE for the likelihood depends on the type of distribution we use for each feature. Suppose $x_j \mid y = c \sim \text{Ber}(\theta_{jc})$, then the MLE becomes $\hat{\theta}_{jc} = \frac{N_{jc}}{N_c}$

2) Example: classifying text

The 20 newsgroup dataset consists of ~ 20000 documents, each of them is a post to one of the 20 newsgroups. We want to build a classification system that assigns a new document to one of these 20 classes.

We need to encode a document as a vector of numerical values. The most common way is to use the bag-of-words model. If the total number of unique words in all documents is M (i.e. the vocabulary used consists of M words), each document is represented as an M-dim. vector. The vector for the n-th document x_n is made up of the counts of the number of times each word appears. x_{nm} is therefore the number of times the word m appears in document n.

We will use multinomials for the class-conditional distributions:

$$p(x_n \mid q) = \left(\frac{s_n!}{\prod_{m=1}^{M} x_{nm}!} \right) \prod_{m=1}^{M} q_m^{x_{nm}}$$

where $s_n = \sum_{m=1}^{M} x_{nm}$ and $q = [q_1, \dots, q_M]^T$ are a set of parameters, each of which is a probability ($\sum_{m=1}^{M} q_m = 1$).

In this case, the MLE is (for class c)

$$q_{cm} = \frac{\sum_{n=1}^{N_c} x_{nm}}{\sum_{m'=1}^{M} \sum_{n=1}^{N_c} x_{nm'}}$$

← summation over objects of class c

Remark: The "naive" assumption simplifies the problem. The number of parameters we require to define each class-conditional is roughly equal to the number of words. If we looked at pairwise dependencies, we would need the order of M^2 parameters. Given that a typical vocabulary might include 50000 words, this is already a significant challenge.

3) Bayesian NBC

In the above example, suppose that a particular word never appears in documents from one class (say c). This will result in $q_{cm} = 0$. If we are trying to compute the classification probability for a document x_{new} that happens to include word m, the likelihood $P(x_{new} \mid y_{new} = c, q_c)$ will equal zero and hence: $P(y_{new} = c \mid x_{new}, X, y) = 0$. A document including a word that doesn't appear in any of the training documents will have probability 0 of belonging to all classes. This is another example of over-fitting the training data and we can overcome this by using a Bayesian procedure: place a prior density on q that encodes the belief that all the probabilities are greater than 0.

Once we have defined this prior, we can set with the MAP estimate (maximum a posteriori)

For the prior, we use the Dirichlet density, defined as:

$$\mathbb{P}(q_c|\alpha) = \frac{\Gamma\left(\sum_{m=1}^{n} \alpha_m\right)}{\prod_{m=1}^{n} \Gamma(\alpha_m)} \prod_{m=1}^{n} q_{cm}^{\alpha_m - 1}$$

gamma function

$$\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t}\, dt$$

$$(\Rightarrow \Gamma(n) = n!)$$

We will simplify further by assuming that $\alpha_m = \alpha$ $(\forall m)$. After some computations, we get the following MAP estimate:

$$q_{cm} = \frac{\alpha - 1 + \sum_{n=1}^{N_c} x_{nm}}{M(\alpha - 1) + \sum_{m'=1}^{M} \sum_{n=1}^{N_c} x_{nm'}}$$

this technique is often refered to as <u>smoothing</u>.

<u>Exercises</u>: Homework 01

+ p. 68-69-70 of the python book