

Homework Three, due Tue 2/14

CSE 250B

- Hand in your homework in hard copy.
- Present your results clearly and succinctly using tables and graphs as appropriate. Discuss your results in precise and lucid prose. Content is king, but looks matter too!
- Please turn in code *only* for the functions specifically mentioned in the exercise (plus any helper functions which they call). The code for these functions should be emailed to <seano@cs.ucsd.edu>. Please include CSE 250B HW 3 in the subject line of your email.

1. VC dimension.

- (a) Let the input space be the real line, and let H be the hypothesis class of *unions of k intervals*. That is, each hypothesis h is associated with k closed intervals $[a_1, b_1], \dots, [a_k, b_k]$; and $h(x)$ is 1 if and only if x lies in the union of these intervals. What is the VC dimension of H ?
- (b) Let the input space be \mathbf{R}^2 , and let H consist of all *homogeneous* linear separators (i.e. linear separators which pass through the origin). Show that H has a VC dimension of 2.
- (c) In general, the VC dimension of homogeneous linear separators in \mathbf{R}^d is d . This means that any m data points in \mathbf{R}^d can be split in at most $(m+1)^d$ different ways by hyperplanes through the origin. However, it turns out that the number is much smaller if we require also that each split has a significant *margin*.

To this end, consider any m data points $x_1, \dots, x_m \in \mathbf{R}^d$, all of length at most 1. We will say that linear separator w splits the data with margin $\gamma > 0$ if

- $\|w\| = 1$, and
- $|w \cdot x_i| \geq \gamma$ for all $i = 1, 2, \dots, m$.

Show that there are at most m^{1/γ^2} ways to split the data points with margin γ . *Hint*: use the perceptron algorithm!

This means that if we only consider linear separators with margin at least γ , then the VC dimension is effectively just $1/\gamma^2$, no matter what d is! It is this curious fact that will allow us to learn linear separators in infinite-dimensional spaces later in the course.

2. Text classification with a “Naive Bayes” generative model.

From the course website, obtain the files `train.data` and `test.data`. These were created, as will be described below, from 19,996 text files found on 20 different newsgroups (about 1000 documents per newsgroup). Each document should be thought of as a data point; its label is the newsgroup it comes from. The goal is to build a generative model of each newsgroup, and to use these models to classify future documents.

- (a) It is problematic that the documents are of different lengths. One way to overcome this is by representing each document by a fixed-length vector called a “bag of words”. The idea is to choose a set of important words (perhaps all words which appear in the training data) w_1, \dots, w_d , and then to represent a document by a d -dimensional binary vector whose i^{th} position is 1 if w_i appears in the document, 0 otherwise. There are other variants in which counts of words are maintained, but we won’t deal with them here.

I have already chosen 1000 key words and converted each document into a 1000-dimensional vector. The files above contain one data point per line, consisting of 1000 binary values followed by a class label in the range 1–20. There are 17,000 training points and 2,996 test points.

If you are curious about the original data set, check out:

<http://people.csail.mit.edu/jrennie/20Newsgroups/>

- (b) For each of the 20 newsgroups, use the training data to build a simple probabilistic model, assuming that the different features are independent. The model for a newsgroup should have 1000 parameters $p_i \in [0, 1]$; the probability of a particular document $x \in \{0, 1\}^{1000}$ is then

$$\prod_{i=1}^{1000} p_i^{x_i} (1 - p_i)^{1-x_i}.$$

A natural choice is to set p_i to the proportion of training documents (from that particular newsgroup) for which $x_i = 1$. In practice, this can be dangerous – when there are lots of features, and any given feature is 1 only a tiny fraction of the time, there often isn’t enough data to reliably estimate all the p_i in this way. Therefore, it is common to “smooth” the estimates somewhat, by setting:

$$p_i = \frac{(\text{number of points with } x_i = 1) + n\tilde{p}}{(\text{number of points}) + n},$$

where n is a small integer and \tilde{p} is a prior estimate of the value of p_i . To keep this simple, use $n = 2$ and $\tilde{p} = 0.5$.

- (c) Now classify the test documents by applying Bayes’ rule. How many does it get right? Notice that the overall model of the data (for all classes considered together) is a *mixture of product distributions*. Classification based upon such a model is often called *Naive Bayes*.
- (d) The probabilistic model can be used not only to predict the most likely newsgroup for a given document, but also to rank all the newsgroups. By looking at the true labels, determine how often the correct newsgroup was ranked first, ranked second, and ranked third.
- (e) Show that this Naive Bayes model can alternatively be described by 20 *linear* functions

$$f_1, \dots, f_{20} : \{0, 1\}^{1000} \rightarrow \mathbf{R},$$

such that any document x is classified as

$$\arg \max_{1 \leq j \leq 20} f_j(x).$$

- (f) *Challenge:* Develop an alternative generative model with better classification performance. Here are some ideas:
- Certain words like “the” and “where” are useless in distinguishing between classes, and the corresponding features can thus be thought of as pure noise. It might help to retain only a subset of the features (words), those with good discriminatory power.
 - Even though the words “San” and “Diego” often go together, the naive Bayes model will treat these as separate pieces of evidence: within each class, it models the features as being independent. It might help to identify a subset of features that truly are quite independent, or alternatively to use a probabilistic model that is able to accommodate dependencies.

You might find it useful to keep aside a portion of the training data as a validation set.

- i. Give a high-level description of your method, followed by Matlab code. The learning phase should look only at the training (and validation) sets.
- ii. What is the misclassification rate of your model on the test data?