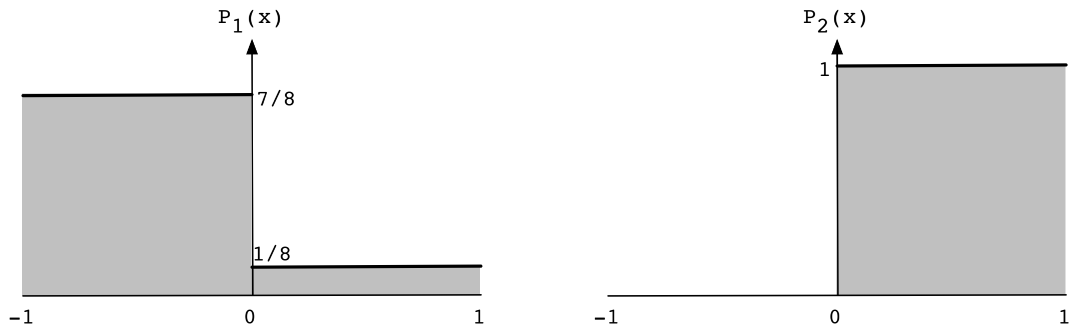
DSE 210: Probability and Statistics using Python

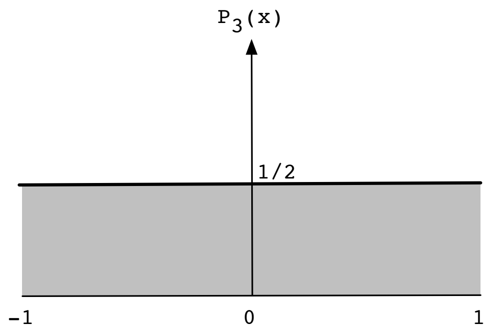
Worksheet 9 — Classification using probability models

1. Suppose $\mathcal{X} = [-1, 1]$ and $\mathcal{Y} = \{1, 2, 3\}$, and that the individual classes have weights

$$\pi_1 = \frac{1}{3}, \quad \pi_2 = \frac{1}{6}, \quad \pi_3 = \frac{1}{2}$$

and densities P_1, P_2, P_3 as shown below.





What is the optimal classifier h^*? Specify it exactly, as a function from $\mathcal{X}$ to $\mathcal{Y}$.

2. Consider the linear classifier $w \cdot x \geq \theta$, where

$$w = \begin{pmatrix} -3 \\ 4 \end{pmatrix} \quad \text{and} \quad \theta = 12.$$

Sketch the decision boundary in $\mathbb{R}^2$. Make sure to label precisely where the boundary intersects the coordinate axes, and also indicate which side of the boundary is the positive side.

3. *Text classification using multinomial Naive Bayes.*

   (a) For this problem, you'll be using the *20 Newsgroups* data set. There are several versions of it on the web. You should download "20news-bydate.tar.gz" from

   `http://qwone.com/~jason/20Newsgroups/`

   Unpack it and look through the directories at some of the files. Overall, there are roughly 19,000 documents, each from one of 20 newsgroups. The label of a document is the identity of its newsgroup. The documents are divided into a training set and a test set.

   (b) The same website has a processed version of the data, "20news-bydate-matlab.tgz", that is particularly convenient to use. Download this and also the file "vocabulary.txt". Look at the first training document in the processed set and the corresponding original text document to understand the relation between the two.

   (c) The words in the documents constitute an overall vocabulary V of size 61188. Build a multinomial Naive Bayes model using the training data. For each of the 20 classes $j = 1, 2, \ldots, 20$, you must have the following:

   - π_j, the fraction of documents that belong to that class; and
   - P_j, a probability distribution over V that models the documents of that class.

   In order to fit P_j, imagine that all the documents of class j are strung together. For each word $w \in V$, let P_{jw} be the fraction of this concatenated document occupied by w. Well, almost: you will need to do smoothing (just add one to the count of how often w occurs).

   (d) Write a routine that uses this naive Bayes model to classify a new document. To avoid underflow, work with logs rather than multiplying together probabilities.

   (e) Evaluate the performance of your model on the test data. What error rate do you achieve?

   (f) If you have the time and inclination: see if you can get a better-performing model.

   - Split the training data into a smaller training set and a validation set. The split could be 80-20, for instance. You'll use this training set to estimate parameters and the validation set to decide between different options.
   - Think of 2-3 ways in which you might improve your earlier model. Examples include: (i) replacing the frequency f of a word in a document by $\log(1 + f)$, (ii) removing stopwords; (iii) reducing the size of the vocabulary; etc. Estimate a revised model for each of these, and use the validation set to choose between them.
   - Evaluate your final model on the test data. What error rate do you achieve?

4. *Handwritten digit recognition using a Gaussian generative model.* In class, we mentioned the MNIST data set of handwritten digits. You can obtain it from:

   `http://yann.lecun.com/exdb/mnist/index.html`

   In this problem, you will build a classifier for this data, by modeling each class as a multivariate (784-dimensional) Gaussian.

   (a) Upon downloading the data, you should have two training files (one with images, one with labels) and two test files. Unzip them.

   In order to load the data into Python you will find the following code helpful:

   `http://cseweb.ucsd.edu/~dasgupta/dse210/loader.py`

For instance, to load in the training data, you can use:

```
x,y = loadmnist('train-images-idx3-ubyte', 'train-labels-idx1-ubyte')
```

This will set x to a 60000×784 array where each row corresponds to an image, and y to a length-60000 array where each entry is a label (0-9). There is also a routine to display images: use `displaychar(x[0])` to show the first data point, for instance.

(b) Split the training set into two pieces – a training set of size 50000, and a separate *validation set* of size 10000. Also load in the test data.

(c) Now fit a Gaussian generative model to the training data of 50000 points:

- Determine the class probabilities: what fraction π_0 of the training points are digit 0, for instance? Call these values $\pi_0, \ldots, \pi_9$.
- Fit a Gaussian to each digit, by finding the mean and the covariance of the corresponding data points. Let the Gaussian for the jth digit be $P_j = N(\mu_j, \Sigma_j)$.

Using these two pieces of information, you can classify new images x using Bayes' rule: simply pick the digit j for which $\pi_j P_j(x)$ is largest.

(d) One last step is needed: it is important to smooth the covariance matrices, and the usual way to do this is to add in cI, where c is some constant and I is the identity matrix. What value of c is right? Use the validation set to help you choose. That is, choose the value of c for which the resulting classifier makes the fewest mistakes on the validation set. What value of c did you get?

(e) Turn in an iPython notebook that includes:

- All your code.
- Error rate on the MNIST test set.
- Out of the misclassified test digits, pick five at random and display them. For each instance, list the posterior probabilities $\Pr(y|x)$ of each of the ten classes.