Assignment 2

Machine Learning, Summer term 2015
Tobias Lang and Ulrike von Luxburg

To be discussed in exercise groups on April 20/22

Total number of points: 23

You can download the data-sets and referenced Matlab functions from the course web page.

Exercise 1 (kNN for digit classification, 5 points (2+3))
In this exercise, we use a kNN classifier to classify handwritten digits from the USPS data-set. You can reuse your kNN classifier from Assignment 1 or use libraries from Matlab (`knnclassify`, `fitcknn`). The USPS data-set contains grayscale handwritten digit images scanned from envelopes by the U.S. Postal Service. The images are of size 16 x 16 (256 pixel) with pixel values in the range 0 to 255. We have 10 classes $\{1, 2, ..., 9, 0\}$. The training data has 500 images, stored in a 500 x 256 Matlab matrix (`usps_train.mat`). The training label is a 500 x 1 vector revealing the labels for the training data. There are 200 test images for evaluating your algorithm in the test data (`usps_test.mat`).

(a) First, we want to classify digit 2 versus digit 3. Prepare the training data:

- Load the train data and prepare the training set for classes 2 and 3. We need to convert the data type from `uint8` (8-bit unsigned integer) to `double`.

```
clear all; clc;
load('usps_train.mat');
trList = find(train_label==2 | train_label==3);
x_train = double(train_data(trList,:));
y_train = double(train_label(trList));
```

- Do the same for the test data.

You can visualize the training example number 155 by using the command `imagesc :`

```
dig = reshape(x_train(155,:),16,16);
imagesc(dig);
colormap('gray');
```

(b) Evaluate the performance of your classifier:

- Test your classifier with different values `k=1,3,5,7,10,15` and plot the training and the test errors.
- Now you can classify other digits. Run your algorithm to classify digit 3 from 8 and compare its performance with results from digit 2 versus 3.

Exercise 2 (kNN for text classification, 5 points (3+2))
In this exercise, we apply the kNN classifier for automatically classifying user posts in a newsgroup. The 20 Newsgroups dataset is a collection of approximately 19,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups (the groups are listed in `20NgClasses.txt`). To simplify the work, we use a bag of words representation of documents. In this representation, the order of words are ignored and we only count the appearance of a word in a document. Your training data is a 11269 x 53975 matrix, where each row corresponds to a document and element (i, j)

shows how often the word j occurs in the document i. You do not need to know the corresponding word for each feature, but they are available in `vocabulary.txt`.
We want to classify documents in class 6 (`comp.windows.x`) and 8 (`rec.autos`).

(a) Classify classes 6 and 8

- Load the dataset and prepare the training set for classes 6 and 8. For simplicity, select $n = 40$ random training example from each class :

```
load('20Newsgroup.mat');
n = 40;
trIdx6 = find(y_train==6);
rand_train_6 = randperm(length(trIdx6));
% Do the same for class 8 to get trIdx8 and  rand_train_8
trList = [trIdx6(rand_train_6(1:n)) ; trIdx8(rand_train_8(1:n))];
x_train_6_8 = x_train(trList,:);
y_train_6_8 = y_train(trList);
```

- Do the same for the test data (on 40 random samples).
- Note that the train and the test data have different feature sizes. The reason is the existence of words in the test set which never appeared in the training set. You can ignore those features by simply cropping them: `x_test_cropped = x_test(:,1:size(x_train,2));`.
- Plot the training and the test error for `k=1,3,5,7,9`. Does it change in different runs?

(b) In this sub-exercise, you are required to use your own kNN-implementation (that is, not a Matlab library). Can you train your classifier using all training points with your kNN-implementation? If you use `for` loops in your classifier, the training on the whole dataset would take several hours. Try to rewrite your code using only matrix operations.

Exercise 3 (Bayes classifier, 8 points (4+4)) In this exercise you will do the maximum likelihood and the Bayes classification by hand. First generate a simple synthetic dataset

```
[X,Y] = mixGaussian1d(1000,0.5,0.5,0,6,1,2);
```

(a) Marginals, priors and class conditional distribution

- Read the Matlab code in `mixGaussian1d.m` and describe its functionality in words.
- Plot the points `[X,Y]` in 2d (this reflects the joint distribution $P(X, Y)$).
- As your data is 1-dimensional, it is hard to interpret this plot. A better way to display the data is to plot the marginal $P(X)$. Estimate the density by its histogram:

```
figure (1); clf; hold all;
[countC,binsX] = hist(X,30);
PX = countC/size(X,1);
plot(binsX,PX,'.-');
```

- Plot the class conditional distributions $P(X|Y = 1)$ and $P(X|Y = 2)$ with different colors in the same figure. To make histogram bins compatible with each other, pass `binsX` as an argument to `hist`.
- Estimate priors $P(Y = 1)$ and $P(Y = 2)$.

(b) Maximum likelihood and Bayesian maximum a-posteriori

- Based only on looking your plots, predict the labels for $X' = \{0, 1, 2, 2.5, 3, 4, 5\}$ using the maximum likelihood criterion.
- Predict the labels with the Bayesian maximum a-posteriori criterion. To do so, draw a new plot which combines the conditional probabilities (used in the maximum likelihood estimation) with the priors.
- Repeat the exercise with `[X,Y] = mixGaussian1d(1000,0.1,0.6,0,6,1,2);`.

Exercise 4 (Letter classification, 2 points (1+1))
Assume you are going to write a two-class classifier which distinguishes the letter j from the letter k. The frequency of the letter j in a typical English text is 0.015, and for the letter k it is 0.045.

(a) Assume your input data are all English letters and you want to separate the letter j from others. Will you buy a classifier with probability of error 0.02?

(b) Assume your input data consists only of letters j and k. Will you buy a classifier with probability of error 0.3? What would be the lowest probability of error if you do the classification without even looking at your input data?

Exercise 5 (Types of errors, 3 points (1+1+1))
You are asked to write a spam filtering algorithm. Two types of errors can occur during your classification: A false-positive occurs when spam filtering wrongly classifies a legitimate email message as spam. While most anti-spam tactics can block or filter a high percentage of unwanted emails, doing so without creating significant false-positive results is a much more demanding task. A false-negative occurs when a spam email is not detected as spam, but is classified as non-spam. A low number of false negatives is an indicator of the efficiency of spam filtering.

(a) Your algorithm finds 85% of spams and miss-classifies 5% of legitimate emails. Additionally you know that about 60% of your incoming emails are spam. What is the probability of a false-positive error and a false-negative error for your classifier? What is the total probability of an error?

(b) Find a classification algorithm with false positive rate 0. Find a classification algorithm with false negative rate 0.

(c) Assume we know $P(spam \mid x)$ for an e-mail x. Sketch by hand the Bayesian decision rule to decide for spam / non-spam for the following two loss functions in a plot ($P(spam \mid x)$ on the x-axis; lines for the point-wise losses for predicting spam / non-spam):

- 0-1 loss
- Asymmetric loss: $\ell(pred = spam, class = non - spam) = 10$,
  $\ell(pred = non - spam, class = spam) = 1$.