Assignment 1

Machine Learning, Summer term 2014, Ulrike von Luxburg

To be discussed in exercise groups on April 14-16

Exercise 1 (kNN classifier, 4+4+2 points) In this exercise, you will implement a kNN classifier in Matlab. A good practice for writing codes is to test it on datasets generated by simple rules. You can check every step of your code using these easy-to-visualize datasets. So first we generate a toy dataset:

```
n1 = 20; train_data_class1 = rand(n1,2);
n2 = 20; train_data_class2 = rand(n2,2) + ones(n2,2)*[1 0; 0 0];
train_data = [train_data_class1 ; train_data_class2];
train_label(1:n1) = 1;
train_label(n1+1:n1+n2) = 2;
```

(a) Prepare the dataset and the classifier:

- Describe the data for class 1 and 2 in words.
- Plot your dataset to see it visually

  ```
  figure (1); clf; hold all; axis equal;
  plot(train_data(1:n1,1), train_data(1:n1,2), 'r*');
  plot(train_data(n1+1:n1+n2,1), train_data(n1+1:n1+n2,2), 'bo');
  ```

- Generate 100 test points for each class (`test_data`) and the labels of the test points (`test_label`).
- Write a Matlab function `knnClassify` that gets the training data `train_data,train_label`, the test data `test_data` and `k` as its input, and returns the predicted labels for the test data (helpful Matlab command: `sort`). Save it as `knnClassify.m`:

  ```
  function pred = knnClassify(train_data, train_label, test_data, k);
  ```

(b) Test the classifier:

- Write a Matlab function `loss01` that gets as input a prediction `y_pred` and correct labels `y`. The function should return the average error (empirical risk with respect to the 0-1 loss) of this prediction:

  ```
  function err = loss01(y_pred,y);
  ```

- Test the classifier with different values `k=1,3,5,7,10,15,20` and store their training and test errors.

  ```
  k_values=[1, 3, 5, 7, 10, 15, 20];
  for i = 1:length(k_values)
  predTrain = ...
  errTrain(i) = ...
  ...
  end
  ```

- Plot the training and the test errors. Do results change between different runs? Why?

  ```
  figure(2); hold all;
  plot(k_values,errTrain,'r*:');
  plot(k_values,errTest, 'b.-');
  ```

Web page: `http://www.informatik.uni-hamburg.de/ML/contents/people/luxburg/teaching/2014-ss-vorlesung-ml/`
Login: "machine", Password: "learning"

- Plot your prediction (`predTest`) for the best k (the one with the smallest test error) in order to see which points are missclassified:

```
figure(3); clf; hold all; axis equal;
pred_class1 = find(predTest==1);
pred_class2 = find(predTest==2);
plot(test_data(pred_class1,1),test_data(pred_class1,2),'r*');
plot(test_data(pred_class2,1),test_data(pred_class2,2),'bo');
plot([1 1],[0 1],'k');
```

(c) Evaluate the performance of your classifier in different datasets:

- **More training examples:** Now increase the size of your training data to 100 examples in class 1 and 100 examples in class 2. How does the performance of kNN classifier change?
- **Unbalanced classes:** Test your classifier for unbalanced class sizes: 200 examples in class 1 and 40 examples in class 2. How does the performance of kNN classifier change?

Exercise 2 (kNN for digit classification, 2+3 points) In this exercise you use your kNN classifier for classifying handwritten digits. The training and the test data are available in the web page. We use the USPS handwritten digits dataset to evaluate this algorithm. The USPS dataset 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`).
First you will classify digit 2 versus digit 3.

(a) 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(train_data(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.