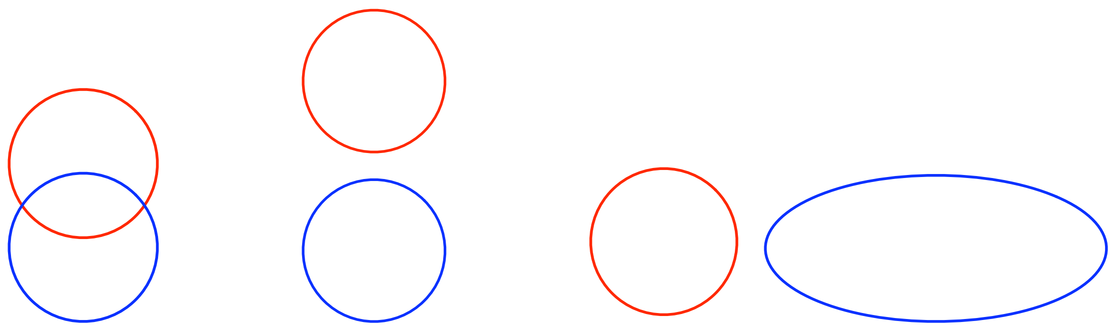
Course: DD2427 - Exercise Set 6

Exercise 1: *Pen and Paper*

The figure below shows 3 different cases **a**, **b** and **c** of distributions of feature vectors from two classes. (All feature vectors from one class lie within one of the coloured circles and/or the coloured ellipse.) Sketch the discriminant line found (if it is found) using

1. a linear perpceptron and
2. a linear minimum squared error classifier

for the 3 cases shown. (You can sketch your answers on a print of this page.)



(**a**) (**b**) (**c**)

Exercise 2: *Linear perceptron learning - Load training data*

In this exercise we investigate finding a hyper-plane that discriminates between images of one digit from the other digits, for instance the digit “0” and all the other digits $1, \ldots, 9$. You will use perceptron learning.

First download a small subset of the MNIST digit database contained in `DigitImages.tar` from the course website. This `tar` file contains 200 images for each digit in the directory `DData`. Half of them will be used for training and the other half for testing. Note the digits have been automatically deskewed.

So your first task is to load these images and keep a record of the label of each image. The prototype of the function will be

```
function [X, labs, w, h] = LoadData(DirName)
```

In this function for each image you read in from `DirName` you should:

1. Next normalize this pixel data to introduce some invariance to illumination effects. To do this compute the mean pixel intensity value, μ, and its standard deviation, σ, and then normalize:

   $$I(x, y) = \frac{I(x, y) - \mu}{\sigma}$$

   After this the mean of the pixel intensities will have mean 0 and standard deviation 1. Note that the *Matlab* functions `mean` and `std` can be used to compute the mean and standard deviation of a vector of numbers.

2. Then you should store this normalized pixel data as a column of `X`.

Thus this function will return an array, `X`, of size `d`$\times$`n` where `d` is the number of pixels in each image and `n` is the number of images loaded. While `labs` is an array of size `1`$\times$`n` which contains the digit label, `0, 1, ..., 9`, of each image you loaded. You should probably keep a record of the original dimensions of the images you used to create X. These dimensions will be useful when displaying images later.

Exercise 3: *Linear perceptron learning - training & testing*

You have loaded all the training images and stored them as columns in a data matrix `X` of size `784` $\times$ `1000` and as well as their associated labels in `lab`. Note that the feature vector you are using is just the raw pixel data. In this exercise you will try and find a separating hyper-plane between digits of one type and all the others and test how good this separating hyperplane is. You will use perceptron learning to do this. In the lecture notes:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta \sum_{\boldsymbol{\phi}_i \in \mathcal{M}} \boldsymbol{\phi}_i \, y_i$$

However, in this exercise we don't transform the feature vector thus:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta \sum_{\mathbf{x}_i \in \mathcal{M}} \mathbf{x}_i \, y_i$$

Remember $\mathcal{M}$ is the set of feature vectors that are incorrectly classified by $\mathbf{w}^{(\tau)}$ and that each vector $\mathbf{x}_i$ is augmented by the value one to take care of the bias ($\mathbf{x}_i = (1, \mathbf{x}_i^T)^T$).

To summarize you need to write functions to perform the following:

- Write the function to compute the weight vector `w` of size `785` $\times$ `1`

  ```
  function w = PerceptronLearning(X, labs, d1)
  ```

  using the perceptron learning algorithm. The inputs to the function are the training data `X`, the labels of the images and an integer `d1` representing the digit you want to recognise. Basically, you will find a separating hyperplane between digits of type `d1` and all the other digits. These are the two classes and have labels `y= -1` and `1`.

  You'll need an initial guess $\mathbf{w}^{(0)}$ for the hyper-plane. I found for this example a vector of `(1, 1, 0, 0, ..., 0)` was adequate and a learning rate of $\eta = .0001$.

- Once you have learnt `w`, load the test images into another data matrix `X1` and the labels `labs1`, of course, remember to normalize. Then see how many of the digits of type `d1` you can classify correctly, the number of true positives `tp`, in conjunction with the number of true negatives `tn`. Do all this in a function called

  ```
  function [tp, tn] = TestHyperPlane(X1, labs1, d1, w)
  ```

  The numbers I got when I learnt a separating hyperplane for each of the digits against all the others are shown in a table below. Note that your numbers may differ depending on your initial `w` etc.

Classified Digit	`tp`	`tn`
0	91	879
1	97	899
2	71	831
3	72	863
4	54	867
5	78	829
6	69	859
7	72	873
8	61	810
9	65	837

For the lecture: 16th April

Bring the following:

- *a print out of your code*
- *a print out of the* `tp` *and* `tn` *for the digits 0, 4 and 9.*