SGN-13006 Introduction to Pattern Recognition and Machine Learning
Department of Signal Processing, Tampere University of Technology
Joni Kamarainen and Katariina Mahkonen
Exercise 6

Complete / start preparing your solutions before the exercise session. During the exercise session you can consult the teacher, and once finished your work, show your solutions to the teacher to get the exercise points.

1. **Linear SVM** (2 points)

   When searching for the decision boundary for a binary classifier with SVM-method, the problem of finding the weights $\mathbf{w}$ and the bias b of the classifier is defined as linear inequalities $y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1$ for all the training data points $i = 1 \cdots N$. For linearly separable data (i.e. there is an empty zone between classes), there are multiple solutions that satisfy these inequalities. To find the one that is optimized for the "support vectors" (the outermost points) from both classes, a minimization of weights (the length of the weight vector $\mathbf{w}^T\mathbf{w}$) is applied. So, the minimization problem is

   $$\min_{\mathbf{w}} \mathbf{w}^T\mathbf{w} \qquad \text{such that} \qquad y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1 \tag{1}$$

   In Matlab and Octave there are optimisation functions to solve this kind of problems. In Matlab,see help for quadprog(), and in Octave check help for `qp()` function.

   To do some SVM classification, create features for items from two artificial classes:

   ```
   X_C1 = rand(100,2);
   X_C2 = bsxfun(@plus,rand(100,2), [0.2 1.2]);
   ```

   Make also SVM-compatible ;-) class labels $y_i \in [-1, 1]$ for the items. Then formulate the necessary matrices of the optimization function (`H,f,A` and `b` for `quadprog()` in Matlab) and run the optimization function to find the SVM solution for the decision boundary parameters $\mathbf{w}$ and b.

   Find the support vectors by testing the "distance" $|\mathbf{x}_i^T\mathbf{w}+b|$ of each point to the found linear discriminant. The support vectors are the points with the minimum distance, i.e. $|\mathbf{x}_i^T\mathbf{w} + b| = 1$ from the decision boudary. Plot points, plot the SVM decision boundary and circle the support vectors.

2. **Mesuring similarity** (1 point) Find the equations giving the following similarity/difference measures and sketch their equal distance "circles" from point [2,1] in two dimensional space, i.e. all the points around the point [2,1] where "distance"=1.

   (a) Euclidean distance
   (b) Manhattan distance
   (c) Chebychev distance
   (d) Mahalanobis distance
   (e) dot product

   Not all of these are mathematically defined as distance metrics. However, in machine learning they are often used to measure similarity or difference.

3. **k-means clustering with Matlab** (3 points)

   Implement k-means algorithm with Matlab by your self. The syntax can be for example:
   `[means, inds] = MyKmeans(data,K,D)`
   The inputs should include the data matrix `'data'`, number of clusters `'K'` and the distance measure identifier `'D'`. Include some Minkowski distance measures (Wikipedia: Minkowski distance) with $p \geq 0$ to your implementation. The outputs of the `MyKmeans` function are the resulting cluster centers `means` and indexes `inds` to which cluster each training data-vector belongs to.

   This time we will not use ImageNet data for testing. Instead, get a data `MFCCfeatures.mat` form POP. It contains MFCC features form spoken finnish vowels 'a','e','i','o','u','y','ä' and 'ö'. Thus there are eight classes. To make the data easier to visualize, we use only the features number 4,5 and 8 in this exercise. Plot the data points in 3 dimensions with `plot3` function. Use your k-means function to find eight clusters (ergo, without using the class information). Finally, plot the cluster centers into the same image with the data.

   There is a function `kmeans` in Matlab. You can also compare your cluster centers to centers given by that.