Megha Vijendra
1001736938
mxv6938

Machine Learning
Project 3

Language used: Python 3
Name of the file: project3_mxv6983.py, elbow_method.py
Libraries used: NumPy, pandas, matplotlib
To run file: python3 project3_mxv6983.py

Overview

A K-means model implemented from scratch to train the model.

1. Exploring the Dataset

The dataset is obtained from the UCI Machine Learning Repository and contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The file is comma separated and the fields/attributes in each record are as follows:

Features

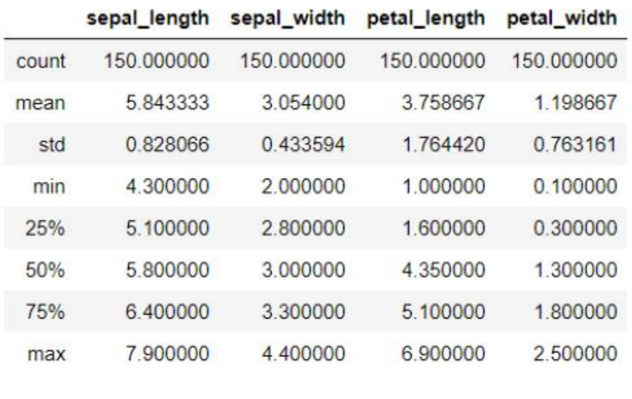
- Sepal Length in cm
- Sepal Width in cm
- Petal Length in cm
- Petal Width in cm

Label

The class of the plant:

- Iris Setosa
- Iris Versicolour
- Iris Virginica

Statistics of the dataset



	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

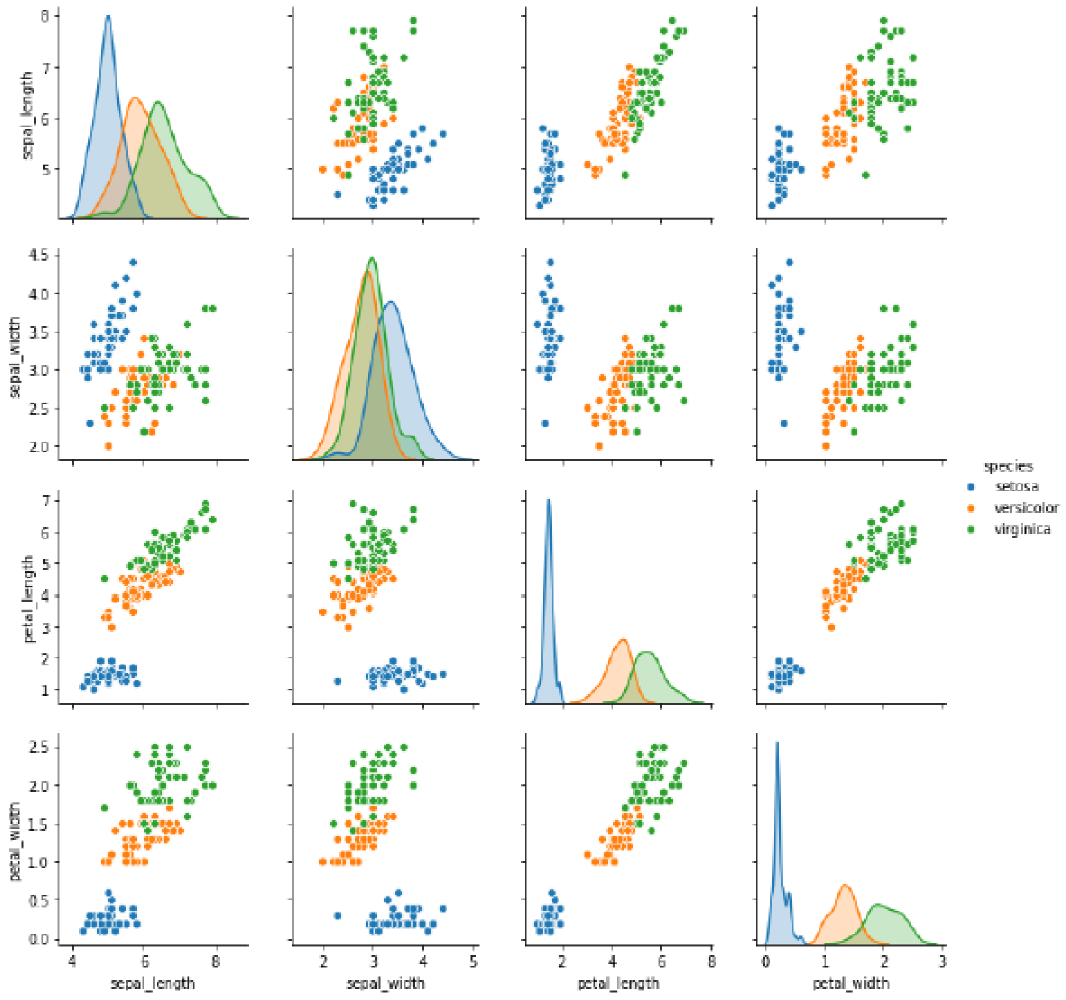
Sample Data

sepal length	sepal width	petal length	petal width	class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa

Visualizing the dataset

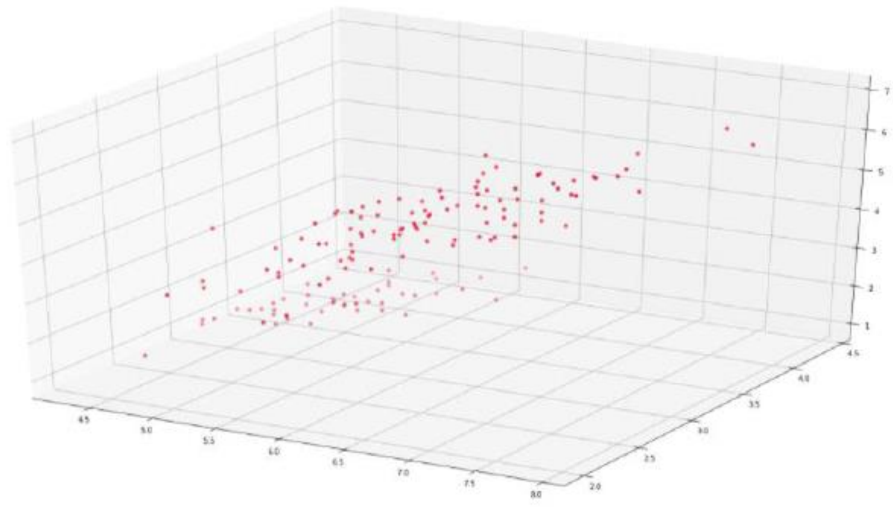
Using seaborn pair plot to see the bivariate relation between each pair of features:

```
Out[30]: <seaborn.axisgrid.PairGrid at 0x12cac12a788>
```



From the above, we can see that **Iris-Setosa** is separated from both other species in all the features.

Visualizing the scores of the dataset using the scatter plot:

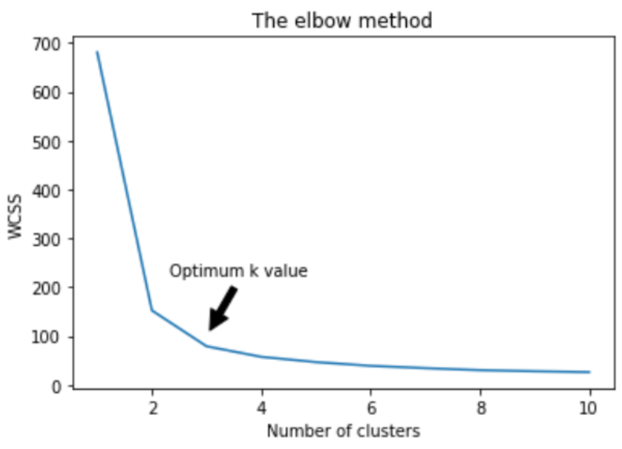


2. K-means Clustering

Clustering is a type of Unsupervised learning algorithm. This is very often used when you do not have labelled data. K-Means Clustering is one of the popular clustering algorithms. The goal of this algorithm is to find groups(clusters) in the given data.
K-Means is a very simple algorithm which clusters the data into K number of clusters.

The algorithm works as follows, assuming we have inputs x_1, x_2, x_3, ..., x_nx and value of K:
Step 1 - Pick K random points as cluster centers called centroids.
Step 2 - Assign each x_i to nearest cluster by calculating its distance to each centroid.
Step 3 - Find new cluster center by taking the average of the assigned points.
Step 4 - Repeat Step 2 and 3 until none of the cluster assignments change.

Here we can choose the value of K by running the elbow_method.py file which gives us the optimum k value by running the kmeans algorithm and give the one with the highest accuracy.
From the file we can see that the optimum k value is 3.
The following diagram shows the optimum kvalue:



Requirements:

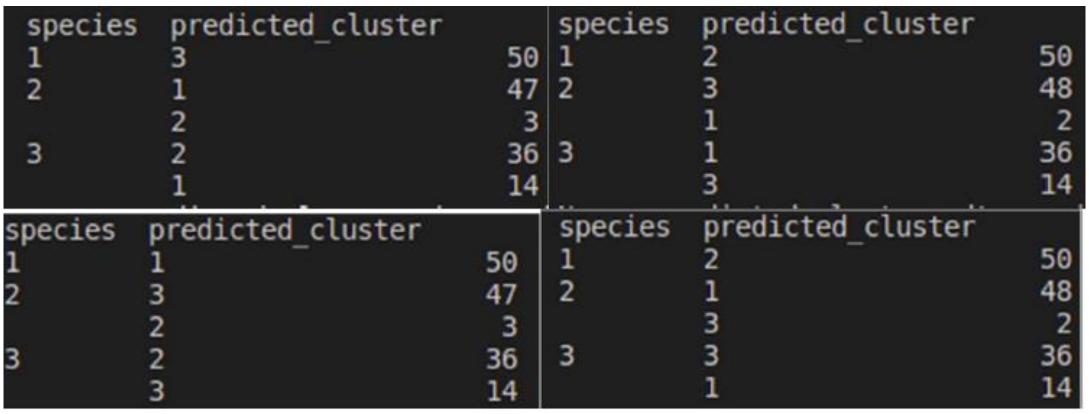
1. Reading from the iris dataset csv and randomly selecting k clusters
The project has included the csv file required to run the program. Formally, the objective is to find

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}\in S_i} \|\mathbf{x}-\boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

Where X is a set of observations (x1, x2, ..., xn), with each observation being a d-dimensional real vector, S is the number of clusters k (<=n) S = {S1, S2, ..., Sk} so as to minimize the within cluster sum of squares.
Here we select k random centroids or 3, as the initial centroids. The decision to select the value for k depends upon the unique number of nominal classes in the species attribute.
2. Setting the initial closest centroids to each of the datapoints
Next, we assign the closes centroids to each datapoint in the dataset. This is accomplished by executing Euclidean distance formula on the dimensional values of the datapoints and figuring out, which centroid each datapoint is closest to.

3. Setting an iteration limit and error limit on the loop condition
K-means by default is an unending loop unless we stop it at an iteration limit or when the error between the old centroids and the new centroids does not change or is less than a particular value.
Here we set the number of iterations at 100 and have a limit that If the difference between the old centroids and the new centroids equal to 0, we exit the loop.
The datapoints are assigned a new centroid every loop if the distance to the created centroids are lesser than the older ones. Even if the centroids are selected at random, they usually converge to the true centroid values.

4.Issues while assigning the clusters to the nominal values.
A common issue faced while assigning the clusters is that the cluster name is not equal to the value of the nominal attribute "species", thus leading to erroneous values if we simply check if the cluster value is equal to the species nominal replaced value. An example of the above can be seen below



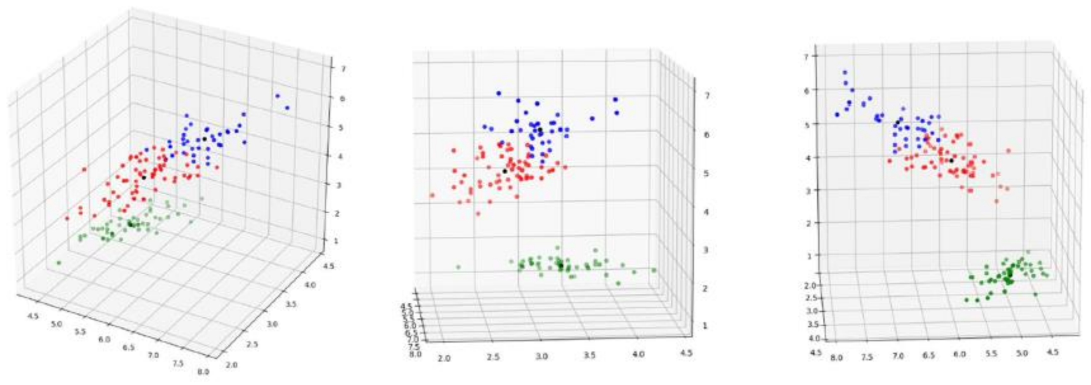
```
species  predicted_cluster
1        3                   50
2        1                   47
         2                    3
3        2                   36
         1                   14
```

```
species  predicted_cluster
1        2                   50
2        3                   48
         1                    2
3        1                   36
         3                   14
```

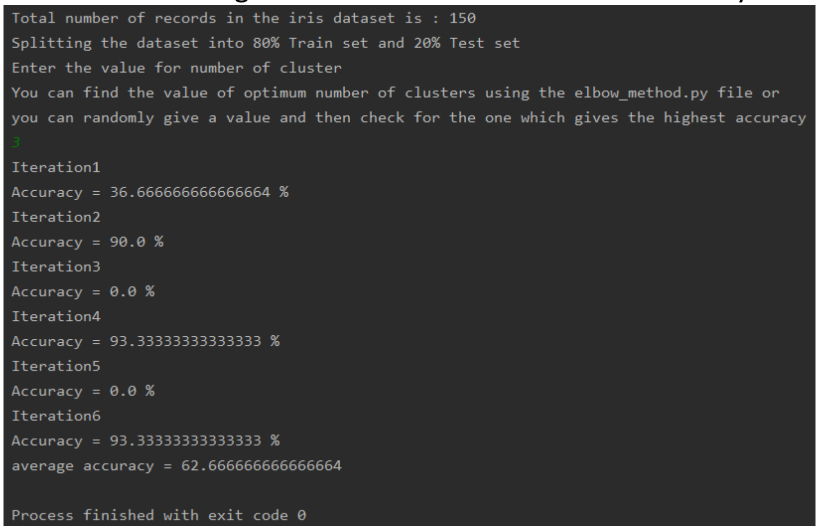
```
species  predicted_cluster
1        1                   50
2        3                   47
         2                    3
3        2                   36
         3                   14
```

```
species  predicted_cluster
1        2                   50
2        1                   48
         3                    2
3        3                   36
         1                   14
```

5.Results:
As the centroids are randomly picked on every run, we would be getting different accuracy values on different runs. The figures below show the results. There is a small variation between results because of the random nature of the selection of the centroids

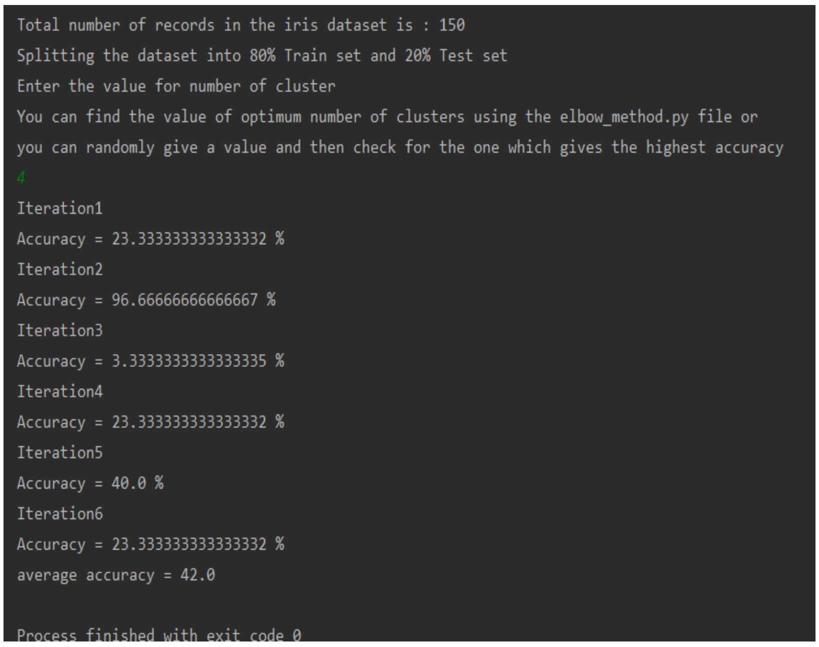


From the below figures we can see the different accuracy values with different k values:



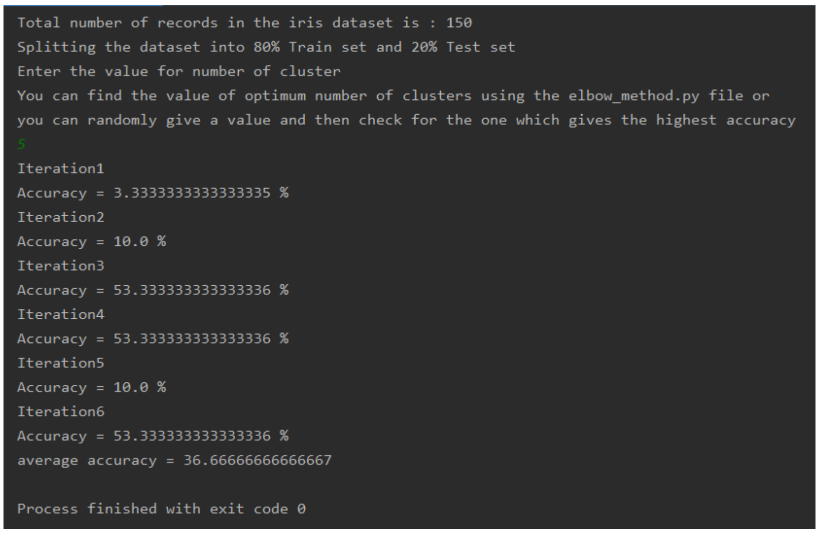
```
Total number of records in the iris dataset is : 150
Splitting the dataset into 80% Train set and 20% Test set
Enter the value for number of cluster
You can find the value of optimum number of clusters using the elbow_method.py file or
you can randomly give a value and then check for the one which gives the highest accuracy
3
Iteration1
Accuracy = 36.666666666666664 %
Iteration2
Accuracy = 90.0 %
Iteration3
Accuracy = 0.0 %
Iteration4
Accuracy = 93.33333333333333 %
Iteration5
Accuracy = 0.0 %
Iteration6
Accuracy = 93.33333333333333 %
average accuracy = 62.666666666666664

Process finished with exit code 0
```



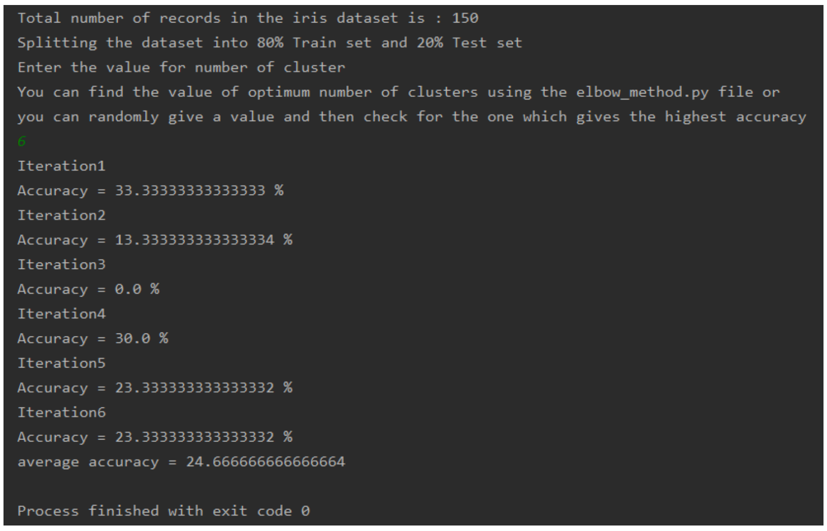
```
Total number of records in the iris dataset is : 150
Splitting the dataset into 80% Train set and 20% Test set
Enter the value for number of cluster
You can find the value of optimum number of clusters using the elbow_method.py file or
you can randomly give a value and then check for the one which gives the highest accuracy
4
Iteration1
Accuracy = 23.333333333333332 %
Iteration2
Accuracy = 96.66666666666667 %
Iteration3
Accuracy = 3.3333333333333335 %
Iteration4
Accuracy = 23.333333333333332 %
Iteration5
Accuracy = 40.0 %
Iteration6
Accuracy = 23.333333333333332 %
average accuracy = 42.0

Process finished with exit code 0
```



```
Total number of records in the iris dataset is : 150
Splitting the dataset into 80% Train set and 20% Test set
Enter the value for number of cluster
You can find the value of optimum number of clusters using the elbow_method.py file or
you can randomly give a value and then check for the one which gives the highest accuracy
5
Iteration1
Accuracy = 3.3333333333333335 %
Iteration2
Accuracy = 10.0 %
Iteration3
Accuracy = 53.333333333333336 %
Iteration4
Accuracy = 53.333333333333336 %
Iteration5
Accuracy = 10.0 %
Iteration6
Accuracy = 53.333333333333336 %
average accuracy = 36.66666666666667

Process finished with exit code 0
```



```
Total number of records in the iris dataset is : 150
Splitting the dataset into 80% Train set and 20% Test set
Enter the value for number of cluster
You can find the value of optimum number of clusters using the elbow_method.py file or
you can randomly give a value and then check for the one which gives the highest accuracy
6
Iteration1
Accuracy = 33.33333333333333 %
Iteration2
Accuracy = 13.333333333333334 %
Iteration3
Accuracy = 0.0 %
Iteration4
Accuracy = 30.0 %
Iteration5
Accuracy = 23.333333333333332 %
Iteration6
Accuracy = 23.333333333333332 %
average accuracy = 24.666666666666664

Process finished with exit code 0
```

3. Methods used

There are different methods in the project3_mxv6938.py file. They are:

1. **def data_preprocessing():**
   This method is used to read the data from the iris.csv datafile and Check if there are any null values in the dataset and to encode the target column to 0,1,2 ordinal values so that it would be easy while testing the dataset.

2. **def data_shuffle_split(df_iris):**
   This method is used to shuffle all the rows in the dataset so as to get better accuracy while training the model and also this method is used to split the data into train and test sets i.e in the ratio 80:20.

3. **def dist(a, b, ax=1):**
   This method is used to find the Euclidean distance between two given points using linalg.norm of the np library.

4. **def create_k_means_model(K, data):**

This method is used to randomly assign initial k centroids and then find the distance between the datapoint to the centroids and assign the datapoints to the closet centroid. Next finds the difference in the values and updates the new centroid, as we update the centroid the error between the old centroid and new centroid if found. The goal is it minimise the error and converge the k-means algorithm. The iteration limit is also set in case the error is not minimised between the centers so that the algorithm can converge.

5. **def predict(model, test):**
   This method is used to assign the datapoints in the test set to the nearest centroid which is the class to which the point belongs to.

6. **def getAcc(model, test):**
   Calculating the accuracy of the model by checking the predicted class by the model and the actual class given in the dataset.

7. **def main():**
   This method is used to input the number of centroids from the user and then show the results of the model.

REFERENCES:

[1] https://blogs.oracle.com/datascience/introduction-to-k-means-clustering
[2] http://worldcomp-proceedings.com/proc/p2015/CSC2663.pdf
[3] https://datascience.stackexchange.com/questions/22/k-means-clustering-for-mixed-numeric-and-categorical-data
[4] http://cs229.stanford.edu/notes/cs229-notes7a.pdf
[5] http://people.csail.mit.edu/dsontag/courses/ml13/slides/lecture14.pdf