Gabriel Ferreira
Department of Computer Science
Iowa State University

HW4 — Kernel Method

Note: Please look for Python notebook sent with this report for code source.

1 Clustering, SVM, & Kernel Method

1. Consider the Lloyeds algorithm for K-means, given the following clusters:

   - $C_1 = \{(0,0), (10,10), (100,100)\}$
   - $C_2 = \{(1,1), (0,5), (-3,4)\}$
   - $C_3 = \{(-1,1), (0,-10), (30,-4)\}$

   a) Formulate the K-means problem as an optimization problem using the given data (of 9 points).

   Dataset X

   $$X = \begin{bmatrix} 0 & 0 \\ 10 & 10 \\ 100 & 100 \\ 1 & 1 \\ 0 & 5 \\ -3 & 4 \\ -1 & 1 \\ 0 & -10 \\ 30 & -4 \end{bmatrix} \in \mathbb{R}^{9\times 2}$$

   Clusters K where K is the number of clusters, and $C_1, C_2, \ldots, C_K$ represent the individual clusters.

   $C_1 = \{(0,0), (10,10), (100,100)\}$
   $C_2 = \{(1,1), (0,5), (-3,4)\}$
   $C_3 = \{(-1,1), (0,-10), (30,-4)\}$

   Centroids Let $\{centroid_1, centroid_2, \ldots, centroid_K\}$ be the centroids of these clusters, where each $centroid_k$ is the mean position of the points in cluster C_k and $\in \mathbb{R}^2$.

   $Centroid_1 = (36.7, 36.7)$
   $Centroid_2 = (-0.67, 3.33)$
   $Centroid_3 = (9.67, -4.33)$

   b) What are your updated clusters after one step of iteration? Please explain your steps to derive your answer.

   $Distance = ||x_n - c_k||^2$

   $dist(x_1, c_1) = \sqrt{(0-36.7)^2 + (0-36.7)^2} = 51.85$
   $dist(x_1, c_2) = \sqrt{(0-(-0.67))^2 + (0-3.33)^2} = 3.39 \;\; x_1 \in c_2$

$dist(x_1, c_3) = \sqrt{(0 - 9.67)^2 + (0 - (-4.33))^2} = 10.59$

$dist(x_2, c_1) = \sqrt{(10 - 36.7)^2 + (10 - 36.7)^2} = 37.71$
$dist(x_2, c_2) = \sqrt{(10 - (-0.67))^2 + (10 - 3.33)^2} = 12.57 \quad x_2 \in c_2$
$dist(x_2, c_3) = \sqrt{(10 - 9.67)^2 + (10 - (-4.33))^2} = 14.33$

$dist(x_3, c_1) = \sqrt{(100 - 36.7)^2 + (100 - 36.7)^2} = 89.56 \quad x_3 \in c_1$
$dist(x_3, c_2) = \sqrt{(100 - (-0.67))^2 + (100 - 3.33)^2} = 139.56$
$dist(x_3, c_3) = \sqrt{(100 - 9.67)^2 + (100 - (-4.33))^2} = 138$

...

$dist(x_9, c_1) = \sqrt{(30 - 36.7)^2 + (-4 - 36.7)^2} = 41.21$
$dist(x_9, c_2) = \sqrt{(30 - (-0.67))^2 + (-4 - 3.33)^2} = 31.53$
$dist(x_9, c_3) = \sqrt{(30 - 9.67)^2 + (-4 - (-4.33))^2} = 20.34 \quad x_9 \in c_3$

Updated Clusters
$C_1 = \{(100, 100)\}$
$C_2 = \{(0, 0), (10, 10), (1, 1), (0, 5), (-3, 4), (-1, 1)\}$
$C_3 = \{(0, -10), (30, -4)\}$

Updated Centroids
$Centroid_1 = (100, 100)$
$Centroid_2 = (1.17, 3.5)$
$Centroid_3 = (15, -7)$

c) Consider using $l1 - norm$ as the distance (cost) measure.

i. What are your updated clusters after one step of iteration? Please explain your steps to derive your answer.

$l1 = ||x_n - c_k||_1$

$dist(x_1, c_1) = |(0 - 36.7)| + |(0 - 36.7)| = 73.3$
$dist(x_1, c_2) = |(0 - (-0.67))| + |(0 - 3.33)| = 4 \quad x_1 \in c_2$
$dist(x_1, c_3) = |(0 - 9.67)| + |(0 - (-4.33))| = 14$

$dist(x_2, c_1) = |(10 - 36.7)| + |(10 - 36.7)| = 53.3$
$dist(x_2, c_2) = |(10 - (-0.67))| + |(10 - 3.33)| = 17.33$
$dist(x_2, c_3) = |(10 - 9.67)| + |(10 - (-4.33))| = 14.67 \quad x_2 \in c_3$

$dist(x_3, c_1) = |(100 - 36.7)| + |(100 - 36.7)| = 126.6 \quad x_3 \in c_1$
$dist(x_3, c_2) = |(100 - (-0.67))| + |(100 - 3.33)| = 197.34$
$dist(x_3, c_3) = |(100 - 9.67)| + |(100 - (-4.33))| = 194.66$

...

$dist(x_9, c_1) = |(30 - 36.7)| + |(-4 - 36.7)| = 47.33$
$dist(x_9, c_2) = |(30 - (-0.67))| + |(-4 - 3.33)| = 38$
$dist(x_9, c_3) = |(30 - 9.67)| + |(-4 - (-4.33))| = 20.66 \quad x_9 \in c_3$

Updated Clusters
$C_1 = \{(100, 100)\}$
$C_2 = \{(0, 0), (1, 1), (-1, 1), (0, 5), (-3, 4), (0, -10)\}$
$C_3 = \{(10, 10), (30, -4)\}$

Updated Centroids
$Centroid_1 = (100, 100)$
$Centroid_2 = (-0.5, 0.17)$
$Centroid_3 = (20, 3)$

ii. In what situations would you prefer to use this cost instead of the standard K-means clustering?

I'd choose the $L_1 norm$ over the standard $L_2 norm$ for clustering if my data has lots of outliers, or has many zero values except for a few important ones (sparse). The $L_1 norm$ would be better at handling weird or extreme data points, works well when there's a lot of information to sift through, and is good for when only a few details are actually important.

d) Use sklearn.cluster.KMeans to cluster the data set of the above 9 points. Set K = 2 (two clusters). Feel free to play with the arguments. Show me your code, and plot your clustered outcome (use different colors to differentiate the clusters).

```python
from sklearn.cluster import KMeans

# KMeans with K = 2
kmeans = KMeans(n_clusters=2, random_state=0).fit(X)

# Predicted cluster labels for each point
labels = kmeans.labels_

# Calculated centroids
centroids = kmeans.cluster_centers_
```

Figure 1: KMeans Code

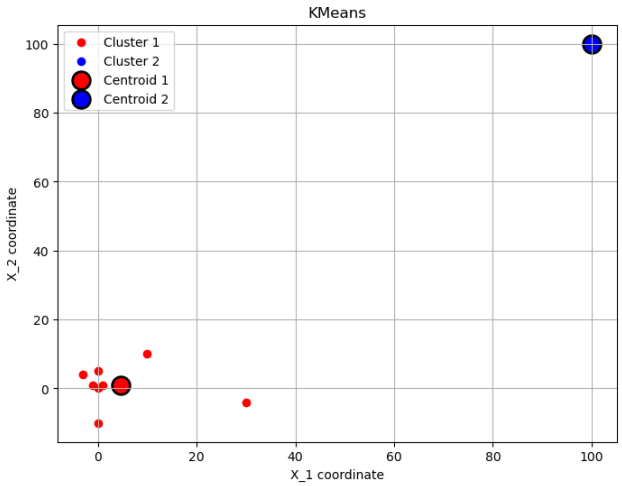

Figure 2: KMeans Plot

Note: Please look for Python notebook sent with this report for code source.

2. Consider the following two clusters:

   - $C_1 = \{(x, y) | y \geq x^2 + 2, x \in \mathbb{R}\}$
   - $C_2 = \{(x, y) | y \leq x^2 - 2, x \in \mathbb{R}\}$

   Are the two clusters linearly separable? What kernel trick can you apply here (specifically what feature transform $\theta : \mathbb{R}^2 \rightarrow F$) to make them linearly separable? Is this kernel trick unique?

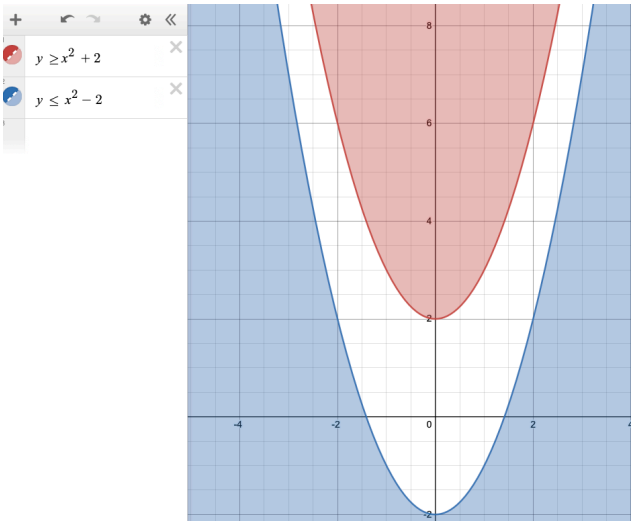

Figure 3: Clusters Plot

The given sets are not linear separable in $\mathbb{R}^2$ as we can see from Figure 3. We can apply the Polynomial kernel to make these clusters separable. The Polynomial kernel trick is not unique. There are other kernel tricks that could separate these clusters as well.

3. Recall the proof of convergence of Gradient Descent with L-smooth function. Try to use a similar idea to prove the convergence of Lloyeds algorithm for K-means.

   Object function $J = \sum_{i=1}^{n} \min_{k=1,...,k} ||x_i - c_k||^2$

   Being $c_k = \frac{1}{S_k} \sum_{x_i \in S_k} x_i$ where S_k is the set of points assigned to cluster k.

   Let $x_1, ..., x_n \in \mathbb{R}^d$

   Say at iteration t of Lloyd's algorithm, we have the current assignment:

   $c_1^t, c_2^t, ..., c_k^t \in \mathbb{R}^d$

   After calculating the current assignment, say we update it to the following:

   $c_1^{t+1}, c_2^{t+1}, ..., c_k^{t+1} \in \mathbb{R}^d$

   In the update process at iteration t, centroids are computed as the mean of the data points assigned to each cluster based on the current assignment. These centroids are denoted by $c_1^t, c_2^t, ..., c_k^t$ as we mentioned above. In the next step, each data point in the dataset will have its distance to these centroids recalculated, leading to a potential reassignment of clusters, aiming to minimize the following objective function:

   $J(A, c_1^t, c_2^t, ..., c_k^t) = \sum_{j=1}^{k} \sum_{i:A(i)=j} ||x_i - c_j^t||^2$

   Here, A is the assignment of data points to clusters that reflects the current clustering configuration. This objective function quantifies the contribution of each data point to the sum of squared distances from points to their assigned centroids. The goal in iteration $t+1$ will be to reassign points and possibly update centroids to further reduce this sum.

   Thus, we can claim the following:
   $\sum_{i=1}^{n} ||x_i - c_{k_i}^t||^2 \geq \sum_{i=1}^{n} ||x_i - c_{k_i}^{t+1}||^2$

   The above inequality demonstrates the principle of convergence clearly: the total sum of squared distances from points to their assigned centroids cannot increase after the update and potential reassignment in the next iteration. We can see that it strictly reduces at each partition assignment. And given the fact we have a finite number of partition assignments, **the algorithm must converge**.

   Note: This does not mean it will converge to the global minimal as it is guaranteed when using Gradient Descent with L-smooth function since they work with a convex assumption. In this case, we are working with a non-convex object function, thus, we could converge to a local minimal instead.

4. Consider the following data set with features $xi \in \mathbb{R}^2, \forall i$ and binary class labels $yi \in \{1, -1\}, \forall i, X = \{(0, 2), (0.4, 1), (0.6, 1), (1, 0)\}, y = \{-1, -1, 1, 1\}$:

   a) Using a scatter plot of the data, devise a linear classifier of the form

   $$\hat{y} = \begin{cases} 1 & \text{if } b + w^T x \geq 0 \\ -1 & \text{if } b + w^T x < 0 \end{cases}$$

that separates the two classes. What is your selected b and w? Is the selection unique?

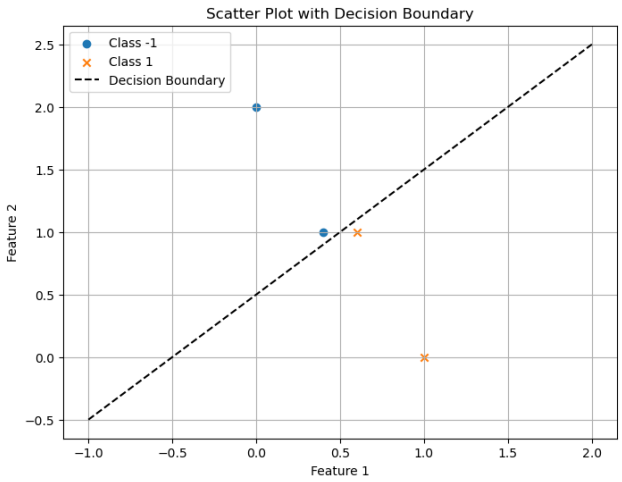

Figure 4: Scatter Plot with Decision Boundary

My selected b was 0.5 and my w was $[1, -1]$. This set of parameters is not unique.

b) Compute the distance of the closest sample to the classifier boundary. Show me your equation and the sample(s) that are the closest.

$SpreadDistance = \frac{|w^T xi+b|}{||w||}$

$SpreadDistance\ x_1 = \frac{|[1,-1]^T[0,2]+0.5|}{\sqrt{1^2+(-1)^2}} = \frac{1.5}{\sqrt{2}} \approx 1.06$

$SpreadDistance\ x_2 = \frac{|[1,-1]^T[0.4,1]+0.5|}{\sqrt{1^2+(-1)^2}} = \frac{.1}{\sqrt{2}} \approx .07$

$SpreadDistance\ x_3 = \frac{|[1,-1]^T[0.6,1]+0.5|}{\sqrt{1^2+(-1)^2}} = \frac{.1}{\sqrt{2}} \approx .07$

$SpreadDistance\ x_4 = \frac{|[1,-1]^T[1,0]+0.5|}{\sqrt{1^2+(-1)^2}} = \frac{1.5}{\sqrt{2}} \approx 1.06$

Samples $x_2(0.4, 1)$ and $x_3(0.6, 1)$ are the closest.

c) Scale your classifier so that $y_i(b + w^T x_i) = 1$ for the closest sample(s) i and report the new b and w. Is $\frac{1}{||w||}$ the same with the minimum distance from question 4(b)?

Closest point: x_2

$-1(0.5 + [1, -1]^T[0.4, 1] = 0.1$

$scaler = \frac{1}{v}$ where v is the above.

$scaler = \frac{1}{.1} = 10$

$w = s \times w \rightarrow w = 10 \times [1, -1] = [10, -10]$
$b = s \times b \rightarrow b = 10 \times 0.5 = 5$

$-1(5 + [10, -10]^T[0.4, 1] = 1$

Closest point: x_3

$1(0.5 + [1, -1]^T[0.6, 1] = 0.1$

$scaler = \frac{1}{.1} = 10$

$w = s \times w \rightarrow w = 10 \times [1, -1] = [10, -10]$
$b = s \times b \rightarrow b = 10 \times 0.5 = 5$

$1(5 + [10, -10]^T[0.6, 1] = 1$

After we have scaled the classifier, the new b value is 5, and the new w values is $[10, -10]$. And now, the $||w|| = \sqrt{10^2 + (-10)^2} = \sqrt{200}$, so $\frac{1}{||w||} = \frac{1}{\sqrt{200}} = 0.07$. Thus, $\frac{1}{||w||}$ is the same with the minimum distance from question 4(b).

5. Create a data set yourself with $X\mathbb{R}^n$ and the classified labels $y \in -1, 1$.

$X = \{(0, 0), (0.936, 2.376), (1.829, 1.496), (0.390, 0.389), (0.145, 2.165), (1.502, 1.770), (2.551, 4.924), (4.588, 3.030), (2.954, 2.958), (3.260, 3.811), (3.579, 3.228), (10.577, 8.228)\}$

$y = \{-1, -1, -1, -1, -1, -1, \ 1, \ 1, \ 1, \ 1, \ 1, \ 1\}$

a) Give the scatter plot of the data you created, use colors to differentiate the different labels.

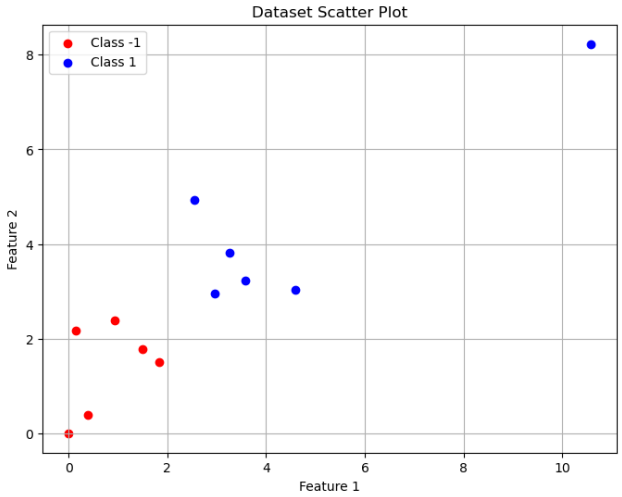

Figure 5: Dataset Scatter Plot

b) Use the SVC tools from sklearn to create a linear classifier for your data set and show it on the plot (note sklearn has at least two ways to implement a linear SVC: svm.LinearSVC and svm.SVC with the argument kernel set to linear", svm.LinearSVC is generally faster).

SVC with kernel = "linear"

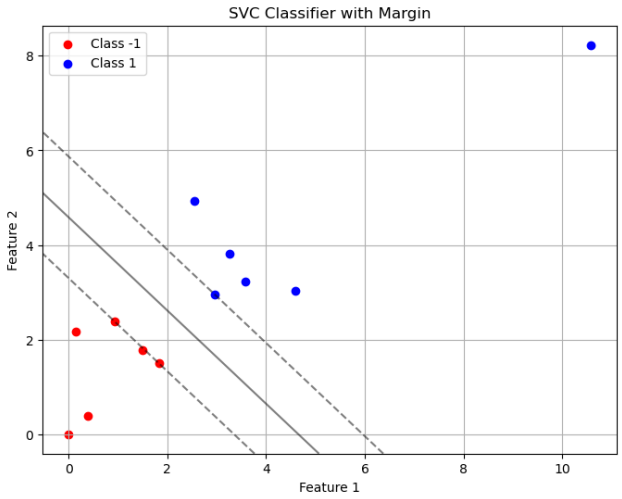

Figure 6: SVC Classifier with Margin

c) Recall the logistic regression introduced in the previous lecture, use sklearn.linearmodel.LogisticRegression to create another linear classifier for your data set. Show it on the plot, and compare it with the SVC-based solution.

Logistic Regression

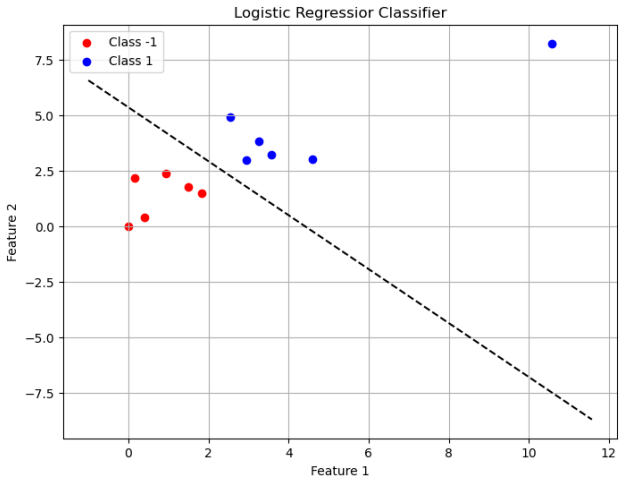

Figure 7: Logistic Regression Classifier

SVC Vs. Logistic Regression

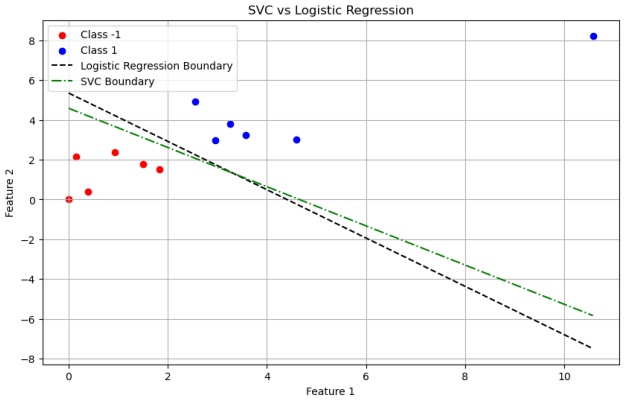

Figure 8: SVC Vs. Logistic Regression

From the above image, we can see the Logistic Regression is likely to be influenced by all data points, including outliers, which could skew its decision boundary if outliers are present. In contrast, SVC focuses on the support vectors closest to the decision boundary and is generally more robust to outliers.

Submitted by Gabriel Ferreira on .