Intro To Machine Learning

Bashar Beshoti (207370248) , Ayal Kaabia (322784760)

August 7, 2024

Course Information

- **Course Title:** Intro To Machine Learning
- **Course Code:** 203.4770
- **Assignment :** 4 (Adaboost + Kernel PCA)
- **Due Date :** 07.08.2024

Adaboost

AdaBoost (Adaptive Boosting) is another approach to the ensemble method field. It always uses the entire data and features (unlike before) and aims to create T weighted classifiers (unlike before, where each classifier had same influence). The new classification will be decided by linear combination of all the classifiers, by:

$$g(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x)), \alpha_t \geq 0$$

- Consider the following dataset in R^2 :

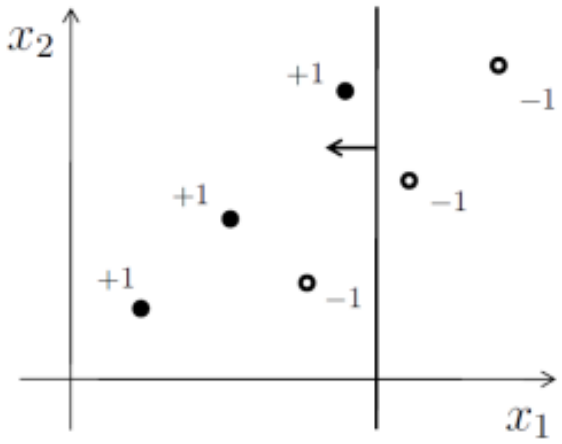

Figure 1: Dataset

1. The first decision stump is already drawn, the arrow points in the positive direction. Calculate the classifier error (ϵ_1) and weight (ϵ_2).

(A) samples weight at the start is equal to $d_0(x_i) = \frac{1}{6}$. The sum of errors $\epsilon_1 = \frac{1}{6}$ where it is the sum of weights of incorrect classified samples.

$\alpha_1 = \frac{1}{2} ln(\frac{1-\epsilon_1}{\epsilon_1}) = \frac{1}{2} ln(5) = 0.8$

2. Calculate the new weights of the samples (and normalize them to get valid distribution).

(A) Updating Weights values :

1. Upon $i \neq 4$, $d_1(x_i) = d_0(x_i) \cdot e^{-\alpha_1} = \frac{1}{6} \cdot e^{(} - 0.8) = 0.074$
2. Upon $i = 4$, $d_1(x_i) = d_0(x_i) \cdot e^{-\alpha_1} = \frac{1}{6} \cdot e^{(}0.8) = 0.37$

The expected example was that the first classifier misclassified received a higher weight and those who correctly classified received a lower weight than they should have

3. Draw the second decision stump. Reminder: the decision stump (our classifiers) are parallel to x/y axis.

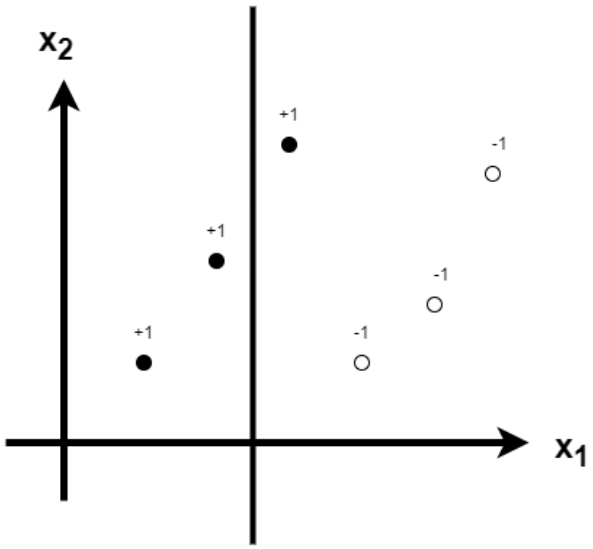

Figure 2: The second classifier will choose to make a mistake in the classification of point whose weight is less than its next point (points 3 and 4 if you consider assigning a number for each point in ascending order

4. Without calculations, which classifier's weight is larger, (ϵ_1) or (ϵ_2)? Explain why.

(A) The second classifier has a high weight created because it correctly classified a point that was difficult for the first classifier to classify correctly (higher weight) and its error is small created by the sum of the weights of the incorrectly classified points.

5. In the right image, there is the dataset and the weights for each point, after finding the third decision stump and calculating the new weights. Which of the following (green or blue) is the correct third decision stump?

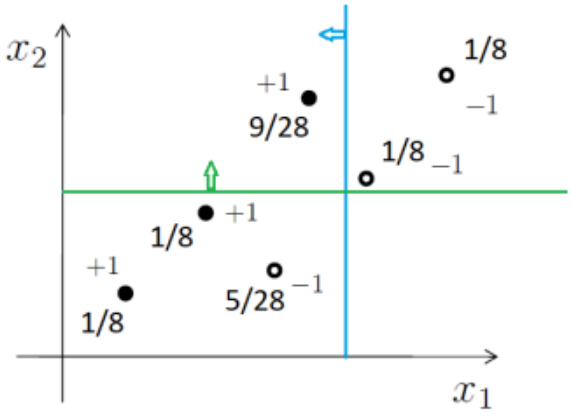

Figure 3: Dataset and the weights for each point

(A) The blue one, after an update, discards for finding the second classifier the third point that was incorrectly classified, you will get it again as a weight cost and the fourth point's weight will be placed in the next step as the third classifier not to make a mistake in the third sample, which is what blue does.

The green is not an option at all, it is wrong in 4 samples.

$$\epsilon_{blue} = \frac{5}{28}$$
$$\epsilon_{green} = \frac{1}{2}$$

6. Given $(\epsilon_2) = 1.1, (\epsilon_3) = 0.62$, draw the full classifier, like in slide 13. What is the train accuracy?

(A)

$$g(x) = sign\Big(\sum_{t=1}^{T} \alpha_t f_t(x)\Big) = \alpha_1 f_1(x) + \alpha_2 t f_2(x) + \alpha_3 f_3(x)$$

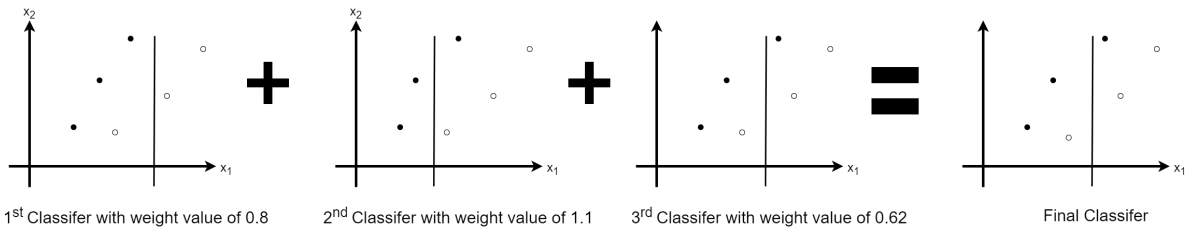

Figure 4: Full Classifier

Classifier f_1 equal to Classifier f_3 such that both of classifiers gives $0.62 + 0.8 = 1.42$ weight which surpasses Classifier f_2 of a weight equal to 1.1. Therefore, The point will be

classified by f_1, f_3.

$$train - accuracy = \frac{6-1}{6} = 83.33\%$$

Kernel PCA

PCA is a useful tool to lower the dimensions of the data and get much less feature that yet represent the data. Even though, sometimes the original data is not good enough, so we want to map the data into higher (or same) dimension!

1. Recall that in PCA, we require the samples to be mean centered, $\frac{1}{n}\sum_{i=0}^{n} x_i = 0$ We now want to do the same thing, but after x was mapped into $\varphi(x)$. Denote the following, mean-centered version of $\varphi(x)$:

$$v_i = \varphi(x_i) - \frac{1}{n}\sum_{i=0}^{n} x_i$$

However, we won't always have access to φ/*costs* a lot of computation (and won't be able to compute $v_1, ..., v_n$).
Therefore, we will use the kernel trick. Denote K' as the kernel matrix for $v_1, ..., v_n$. Show how K' can be calculated using only the original kernel matrix K on $v_1, ..., v_n$.
Reminder: $K'_{i,j} =< v_i, v_j >$ (since the new kernel is linear in terms of $\varphi(x)$).

(A) Given the original kernel matrix K, where $K_{i,j} = \langle\varphi(x_i), \varphi(x_j)\rangle$, we want to find the mean-centered kernel matrix K'.

1. Mean-Centering in Feature Space: The mean-centered version of $\varphi(x_i)$ is:

$$v_i = \varphi(x_i) - \frac{1}{n}\sum_{k=1}^{n} \varphi(x_k)$$

Denote the mean vector in the feature space as:

$$\mu = \frac{1}{n}\sum_{k=1}^{n} \varphi(x_k)$$

2. Kernel Matrix for Mean-Centered Data: We need to calculate $K'_{i,j} = \langle v_i, v_j\rangle$. Substitute v_i and v_j:

$$K'_{i,j} = \langle\varphi(x_i) - \mu, \varphi(x_j) - \mu\rangle$$

Expanding this:

$$K'_{i,j} = \langle\varphi(x_i), \varphi(x_j)\rangle - \langle\varphi(x_i), \mu\rangle - \langle\mu, \varphi(x_j)\rangle + \langle\mu, \mu\rangle$$

3. Expressing in Terms of K: - The term $\langle\varphi(x_i), \varphi(x_j)\rangle$ is $K_{i,j}$. - The term $\langle\varphi(x_i), \mu\rangle$ is:

$$\langle\varphi(x_i), \mu\rangle = \left\langle \varphi(x_i), \frac{1}{n}\sum_{k=1}^{n} \varphi(x_k) \right\rangle = \frac{1}{n}\sum_{k=1}^{n} \langle\varphi(x_i), \varphi(x_k)\rangle = \frac{1}{n}\sum_{k=1}^{n} K_{i,k}$$

- Similarly, $\langle \mu, \varphi(x_j) \rangle$ is:

$$\langle \mu, \varphi(x_j) \rangle = \frac{1}{n} \sum_{k=1}^{n} K_{k,j}$$

- The term $\langle \mu, \mu \rangle$ is:

$$\langle \mu, \mu \rangle = \left\langle \frac{1}{n} \sum_{k=1}^{n} \varphi(x_k), \frac{1}{n} \sum_{l=1}^{n} \varphi(x_l) \right\rangle = \frac{1}{n^2} \sum_{k=1}^{n} \sum_{l=1}^{n} K_{k,l}$$

4. Substituting Back: Combine all these expressions:

$$K'_{i,j} = K_{i,j} - \frac{1}{n} \sum_{k=1}^{n} K_{i,k} - \frac{1}{n} \sum_{k=1}^{n} K_{k,j} + \frac{1}{n^2} \sum_{k=1}^{n} \sum_{l=1}^{n} K_{k,l}$$

Thus, the mean-centered kernel matrix K' is computed from the original kernel matrix K using the above formula.

2. Now, for the rest of the question we assume that $\frac{1}{n} \sum_{i=0}^{n} x_i = 0$.

We would like to apply PCA to the vectors $\varphi(x_i)$. Denote by $u_1, ..., u_k \in R^{d'}$ the first k principal components, the eigenvectors of the scatter matrix S.
Show that u_j (for j=1,...,k) is linear combination of $\varphi(x_1), ..., \varphi(x_n)$.
Moreover, show who are the $a_{j,1}, ..., a_{j,n}$ such that $u_j = \sum_i (\alpha_i \varphi(x_i)$.

(A) To show that u_j is a linear combination of $\varphi(x_i)$ and to find the coefficients $\alpha_{j,i}$, let's proceed with the following steps:

1. Eigenvector Equation in Feature Space: Since u_j is an eigenvector of S, we have:

$$S u_j = \lambda_j u_j$$

Substitute S with its definition:

$$\frac{1}{n} \sum_{i=1}^{n} \varphi(x_i) \varphi(x_i)^T u_j = \lambda_j u_j$$

2. Assume u_j is a Linear Combination: Suppose u_j can be written as:

$$u_j = \sum_{i=1}^{n} \alpha_{j,i} \varphi(x_i)$$

Substituting this into the eigenvector equation:

$$\frac{1}{n} \sum_{i=1}^{n} \varphi(x_i) \varphi(x_i)^T \left(\sum_{k=1}^{n} \alpha_{j,k} \varphi(x_k) \right) = \lambda_j \sum_{k=1}^{n} \alpha_{j,k} \varphi(x_k)$$

3. Simplify the Equation: Distribute and use the linearity of the inner product:

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_{j,k} \varphi(x_i) \langle \varphi(x_i), \varphi(x_k) \rangle = \lambda_j \sum_{k=1}^{n} \alpha_{j,k} \varphi(x_k)$$

Recognize that $\langle \varphi(x_i), \varphi(x_k) \rangle = K_{i,k}$, the kernel matrix elements:

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_{j,k} \varphi(x_i) K_{i,k} = \lambda_j \sum_{k=1}^{n} \alpha_{j,k} \varphi(x_k)$$

4. Extract Coefficients: Since the $\varphi(x_i)$ are linearly independent in the feature space, the coefficients of $\varphi(x_i)$ on both sides of the equation must be equal. Thus:

$$\frac{1}{n} \sum_{k=1}^{n} K_{i,k} \alpha_{j,k} = \lambda_j \alpha_{j,i}$$

This can be written in matrix form as:

$$\frac{1}{n} K \alpha_j = \lambda_j \alpha_j$$

Where $\alpha_j = (\alpha_{j,1}, \alpha_{j,2}, \ldots, \alpha_{j,n})^T$.

5. Eigenvectors of the Kernel Matrix: This implies that α_j are the eigenvectors of the kernel matrix K, scaled by $\frac{1}{n}$.

Thus, the eigenvectors u_j in the feature space are linear combinations of the mapped data points $\varphi(x_i)$, with the coefficients α_j being the eigenvectors of the kernel matrix K.

3. Use the expression for $a_{j,1}, ..., a_{j,n}$ and find an algorithm to obtain the entire $a_j = [a_{j,1}, ..., a_{j,n}]$ using K.

Hint: substitute u_j you found in the expression for $a_{i,j}$ you found.

To find the vector $\alpha_j = [\alpha_{j,1}, \alpha_{j,2}, \ldots, \alpha_{j,n}]^T$ using the kernel matrix K, follow these steps:

1. Compute the Kernel Matrix K: Calculate the kernel matrix K using the given kernel function $K(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$.

2. Center the Kernel Matrix: Center the kernel matrix K to get K' using the formula:

$$K' = K - \mathbf{1}_n K - K \mathbf{1}_n + \mathbf{1}_n K \mathbf{1}_n$$

Here, $\mathbf{1}_n$ is an $n \times n$ matrix with all elements equal to $\frac{1}{n}$.

3. Solve the Eigenvalue Problem: Solve the eigenvalue problem for the centered kernel matrix K':

$$K' \alpha_j = \lambda_j \alpha_j$$

Obtain the eigenvalues λ_j and the corresponding eigenvectors α_j.

4. Normalize the Eigenvectors: Normalize the eigenvectors α_j to ensure they have unit length.

4. Since we don't really know φ, we can't use u_j. But we also don't need them! Their only purpose was to project $\varphi(x)$ onto the new dimension, using the dot product, $z_j =< u_j, \varphi(x) >$ and $z^i = [z_1^i, ..., z_k^i]$.

Show how to calculate z_j without using φ.

To calculate z_j without using φ, we use the following steps:

1. Principal Component Representation:

Each principal component u_j can be represented as a linear combination of $\varphi(x_i)$:

$$u_j = \sum_{i=1}^{n} \alpha_{j,i} \varphi(x_i)$$

where $\alpha_{j,i}$ are the coefficients obtained from the eigenvectors of the centered kernel matrix.

2. Dot Product Expression:

The dot product $z_j = \langle u_j, \varphi(x) \rangle$ can be written as:

$$z_j = \left\langle \sum_{i=1}^{n} \alpha_{j,i} \varphi(x_i), \varphi(x) \right\rangle = \sum_{i=1}^{n} \alpha_{j,i} \langle \varphi(x_i), \varphi(x) \rangle$$

3. Kernel Function:

Since $\langle \varphi(x_i), \varphi(x) \rangle = K(x_i, x)$, where K is the kernel function, we can rewrite the dot product as:

$$z_j = \sum_{i=1}^{n} \alpha_{j,i} K(x_i, x)$$

Thus, z_j can be calculated without explicitly using φ, but instead using the kernel function K and the coefficients $\alpha_{j,i}$.

5. Now, use sklearn and apply Kernel PCA with 'cosine' kernel (doesn't have hyperparameters) and classify with KNN using 125 neighbors, on the same data from question 3.

Did it do better than the regular PCA (on the test set)? you can check the code in the assignment.