Introduction to Machine Learning

Homework 4

Names: Obaida Khateeb, 201278066

Adham Dahli, 318575834

Question 4 – AdaBoost

Let's denote the samples (points) by $x'_1, \dots, x'_6$, where the number given according to their x-value order from the smallest to the largest, and to denote the sample x'_i weight by $d_t(x'_i)$. In addition, $f_t(x'_i)$ represents the t-th weak classifier of the sample x'_i.

1) The first decision stump is already drawn, the arrow points in the positive direction. Calculate the classifier error (ε_1) and weight (α_1).

The classification error ε_1 equals to the sum of the weights of the misclassified points. Initially, all the weights of the points are equal, and since there are 6 points, each of them will have a weight of $\frac{1}{6}$. the points $x'_1, x'_2, x'_4, x'_5, x'_6$ are truly classified where the point x'_3 is misclassified. Therefore:

$$\varepsilon_1 = \sum d_1(x'_i) * [y_i \neq f_1(x'_i)] = \sum_{i \neq 3} d_1(x'_i) * 0 + d_1(x'_3) * 1 = \mathbf{\frac{1}{6}}$$

The weight α_1 is calculated using the following formula:

$$\alpha_1 = \frac{1}{2} * \ln\left|\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right)\right| = \frac{1}{2} * \ln\left(\frac{1-\frac{1}{6}}{\frac{1}{6}}\right) = \frac{1}{2} * \ln(5) \approx \mathbf{0.8}$$

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

The new weight for each of the truly classified points is:

$$d_2(x'_i) = d_1(x'_i) * \exp\left(-\alpha_1 y_i f_t(x'_i)\right) = \frac{1}{6} * e^{-0.8}, i \in \{1,2,4,5,6\}$$

The new weight of the misclassified point is :

$$d_2(x'_3) = d_1(x'_3) * \exp\left(-\alpha_1 y_i f_t(x'_3)\right) = \frac{1}{6} * e^{0.8}$$

To normalize the new weights, we should divide them by their new sum of weights:

$$Z_t = 5 * \frac{1}{6} * e^{-0.8} + \frac{1}{6} * e^{0.8} = \frac{5}{6} * e^{-0.8} + \frac{1}{6} * e^{0.8}$$

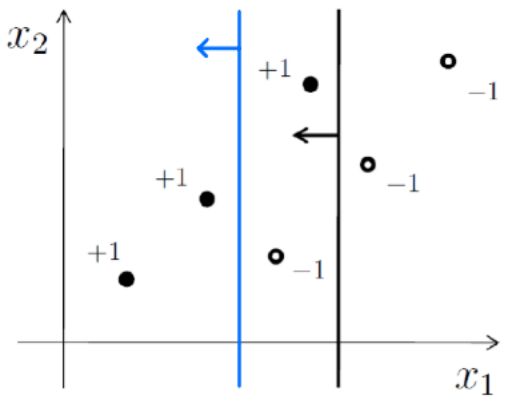
Therefore, the weight for each of the truly classified samples is:

$$d_2(x'_i) = \frac{\frac{1}{6} * e^{-0.8}}{Z_t} = \frac{\frac{1}{6} * e^{-0.8}}{\frac{5}{6} * e^{-0.8} + \frac{1}{6} * e^{0.8}} \approx \frac{0.075}{0.745} \approx \mathbf{0.1}, \boldsymbol{i} \in \{\mathbf{1}, \mathbf{2}, \mathbf{4}, \mathbf{5}, \mathbf{6}\}$$

The weight of the misclassified sample is:

$$d_2(x'_3) = \frac{\frac{1}{6} * e^{0.8}}{\frac{5}{6} * e^{-0.8} + \frac{1}{6} * e^{0.8}} \approx \frac{0.371}{0.745} \approx \mathbf{0.5}$$

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



4) Without calculations, which classifier's weight is larger, α_1 or α_2? Explain why.

Generally, **the second classifier** (α_2) **weight is larger**, because the point that the second stump will misclassify will have a smaller weight since it has been classified correctly by the first stump. Equivalently, It truly classifies a point that have been misclassified by the first stump which make it have a larger weight.

Mathematically, smaller misclassified samples weights → smaller ε_1 value → larger $(\frac{1-\varepsilon_1}{\varepsilon_1})$ value → larger $(\ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right))$ value → larger α value.

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?

The green is the correct third decision stump. Since the given weights are according to the weights update after finding the third decision stump, the sum of weights of the misclassified points should be equal to 0.5, which is true for the misclassified points by the green stump $(4 * \frac{1}{8} = 0.5)$ but not so for the blue stump $(\frac{5}{28})$.

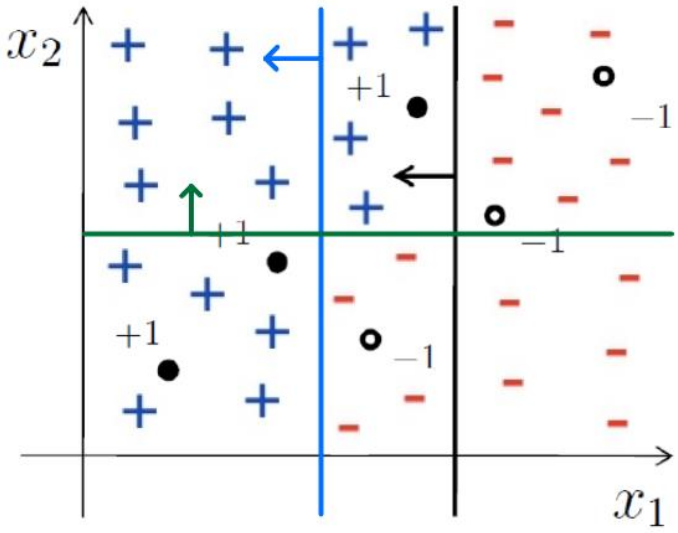
6) Given $\alpha_2 = 1.1$, $\alpha_3 = 0.62$, draw the full classifier, like in slide 13.

What is the train accuracy?

The full classifier will be the combination of the three classifiers, and will be given by the following formula:

$$f_{FINAL}(x') = sign\big(\alpha_1 f_1(x') + \alpha_2 f_2(x') + \alpha_3 f_3(x')\big)$$

$$= sign\big(0.8 f_1(x') + 1.1 f_2(x') + 0.62 f_3(x')\big)$$



Train accuracy:

$$f_{FINAL}(x_1') = sign\big(\alpha_1 f_1(x_1') + \alpha_2 f_2(x_1') + \alpha_3 f_3(x_1')\big)$$
$$= sign(0.8 * 1 + 1.1 * 1 + 0.62 * 1) = (+)$$

$$f_{FINAL}(x_2') = sign\big(\alpha_1 f_1(x_2') + \alpha_2 f_2(x_2') + \alpha_3 f_3(x_2')\big)$$
$$= sign(0.8 * 1 + 1.1 * 1 + 0.62 * 1) = (+)$$

$$f_{FINAL}(x_3') = sign\big(\alpha_1 f_1(x_3') + \alpha_2 f_2(x_3') + \alpha_3 f_3(x_3')\big)$$
$$= sign(0.8 * 1 + 1.1 * (-1) + 0.62 * (-1)) = (-)$$

$$f_{FINAL}(x_4') = sign\big(\alpha_1 f_1(x_4') + \alpha_2 f_2(x_4') + \alpha_3 f_3(x_4')\big)$$
$$= sign(0.8 * 1 + 1.1 * (-1) + 0.62 * 1) = (+)$$

$$f_{FINAL}(x_5') = sign\big(\alpha_1 f_1(x_5') + \alpha_2 f_2(x_5') + \alpha_3 f_3(x_5')\big)$$
$$= sign(0.8 * (-1) + 1.1 * (-1) + 0.62 * (-1)) = (-)$$

$$f_{FINAL}(x_6') = sign\big(\alpha_1 f_1(x_6') + \alpha_2 f_2(x_6') + \alpha_3 f_3(x_6')\big)$$
$$= sign(0.8 * (-1) + 1.1 * (-1) + 0.62 * (-1)) = (-)$$

Thus the training accuracy is:

$$Train\ accuracy = \frac{\#\ of\ correctly\ classified\ samples}{\#\ of\ samples} = \frac{6}{6} = \mathbf{100\%}$$

Question 5 – Kernel PCA

1. Recall that in PCA, we require the samples to be mean-centered, $\frac{1}{n}\sum_{i=1}^{n} x_i = 0$.

   We now want to do the same thing, but after x was mapped into $\varphi(x)$. Denote the following, mean-centered version of $\varphi(x)$:

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

   However, we won't always have access to φ / costs a lot of computation (and wont be able to compute $v_1, \dots, v_n$).

   Therefore, we will use the kernel trick. Donate K' as the kernel matrix for $v_1, \dots, v_n$. Show how K' can be calculated using only the original kernel matrix K on $x_1, \dots, x_n$.

   Reminder: $K'_{i,j} = \langle v_i, v_j \rangle$ (since the new kernel is linear in terms of $\varphi(x)$).

We need to calculate K', which is a matrix where each of its elements defined as:

$$K'_{i,j} = < v_i, v_j >$$

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

$$K'_{i,j} = < \varphi(x_i) - \frac{1}{n}\sum_{k=1}^{n} \varphi(x_k), \varphi(x_j) - \frac{1}{n}\sum_{k=1}^{n} \varphi(x_k) >$$

When expanding the inner product we get the following:

$$K'_{i,j} = < \varphi(x_i), \varphi(x_j)$$
$$> - \frac{1}{n}\sum_{k=1}^{n} < \varphi(x_i), \varphi(x_k) > - \frac{1}{n}\sum_{k=1}^{n} < \varphi(x_k), \varphi(x_j) >$$
$$+ \frac{1}{n^2}\sum_{k=1}^{n}\sum_{k'=1}^{n} < \varphi(x_k), \varphi(x_{k'}) >$$

Therefore:

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

Thus, K' can be expressed by K in the following manner:

$$K' = K - \begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \ddots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix} * K - K * \begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \ddots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix} + \begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \ddots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix} * K * \begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \ddots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix}$$

Or equivalently:

$$K' = K - \frac{1}{n} * 1_n * K - \frac{1}{n} * K * 1_n + \frac{1}{n^2}\, 1_n * K * 1_n$$

where 1_n is the matrix of size n, that all of its elements are equal to 1.

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

   We would like to apply PCA to the vectors $\varphi(x_i)$. Denote by $u_1, \ldots, u_k \in R^{d'}$ the first k principal components, the eigenvectors of the scatter matrix S.

   Show that u_j (for j=1,…,k) is linear combination of $\varphi(x_1), \ldots, \varphi(x_n)$.

   Moreover, show who are the $\alpha_1, \ldots, \alpha_k$ such that $u_j = \sum_i \alpha_i \varphi(x_i)$.

The scatter matrix S in the feature space defined as:

$$S = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i) * \phi(x_i)^T$$

The principal components u_j are the eigenvectors of the scatter matrix S, thus to find them we should solve the following:

$$Su_j = \lambda_j u_j$$

where λ_j is the eigenvalue that's corresponded to the eigenvector u_j.

To do so, we substitute S into the above formula, so we get:

$$\left(\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) * \phi(x_i)^T\right)u_j = \lambda_j u_j$$

Since u_j lies in the span of $\phi(x_i)$, lets assume u_j can be written as a linear combination of $\phi(x_i)$, which means there are coefficients α_{ij} such that:

$$u_j = \sum_{i=1}^{n}\alpha_{ij} * \phi(x_i)$$

Substituting u_j into the same formula from above:

$$\left(\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) * \phi(x_i)^T\right)\left(\sum_{i=1}^{n}\alpha_{ij} * \phi(x_i)\right) = \lambda_j\left(\sum_{i=1}^{n}\alpha_{ij} * \phi(x_i)\right)$$

Re-arranging:

$$\left(\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{n}\alpha_{kj}\,\phi(x_i) *< \phi(x_i), \phi(x_k) >\right) = \lambda_j\left(\sum_{i=1}^{n}\alpha_{ij} * \phi(x_i)\right)$$

Where $< \phi(x_i), \phi(x_k) > = k(x_i, x_k)$ is the kernel function.

Lets define the kernel matrix K as:

$$K_{ik} = k(x_i, x_k)$$

To substitute it in the formula:

$$\left(\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{n}\alpha_{kj}\,\phi(x_i) * K_{ik}\right) = \lambda_j\left(\sum_{i=1}^{n}\alpha_{ij} * \phi(x_i)\right)$$

In order for the equality to hold, the coefficients of $\phi(x_i)$ must match:

$$\left(\frac{1}{n}\sum_{k=1}^{n}\alpha_{kj} * K_{ik}\right) = \lambda_j \alpha_{ij}$$

This is a matrix equation:

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

This implies that α_j are the eigenvalues of the kernel matrix K:

$$K\alpha_j = n * \lambda_j \alpha_j$$

We have derived that the eigenvectors u_j of the scatter matrix S can be expressed as linear combinations of the mapped data points $\phi(x_i)$. Specifically, the coefficients α_j are the eigenvectors of the kernel matrix K. Hence,

$$u_j = \sum_{i=1}^{n} \alpha_{ij} * \phi(x_i)$$

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

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

In section (2) we showed that:

$$u_j = \sum_{i=1}^{n} \alpha_{ij} * \phi(x_i)$$

Also, we know that the principal components are the eigenvectors of the scatter matrix, that's defined by:

$$S = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i) * \phi(x_i)^T$$

When substituting both the scatter matrix and u_j into the equation $Su_j = \lambda_j u_j$ we get the follwing

$$\left(\frac{1}{n}\sum_{i=1}^{n}\phi(x_i)*\phi(x_i)^T\right)\left(\sum_{i=1}^{n}\alpha_{ij}*\phi(x_i)\right)=\lambda_j\left(\sum_{i=1}^{n}\alpha_{ij}*\phi(x_i)\right)$$

Rearranging get us:

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{i=1}^{n}\alpha_{ij}*\phi(x_i)\,K(x_i,x_k)=\lambda_j\left(\sum_{i=1}^{n}\alpha_{ij}*\phi(x_i)\right)$$

Lets dividing both sides by $\sum_{i=1}^{n}\phi(x_i)$:

$$\frac{1}{n}\sum_{i=1}^{n}\alpha_{ij}*K(x_i,x_k)=\lambda_j\alpha_{ij}$$

This equation can be expressed using the matrix form by:

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

Multiplying both sides by n:

$$\boldsymbol{K\alpha_j=n\lambda_j\alpha_j}$$

Where $n\lambda_j$ are the eigenvalues of K.

So we can state the algorithm as the following:

1. Compute the kernel matrix K, and center the kernel matrix using the formula in section 1.
2. Obtain the eigenvalues and eigenvector by solving the formula:
$$K\alpha_j=n\lambda_j\alpha_j$$
3. Normalize the eigenvectors founded in the (2) phase.

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, $z_j=\langle u_j,\varphi(x)\rangle$ (and $z^i=[z_1^i \quad ... \quad z_k^i]$).

   Show how to calculate z_j <u>without</u> using φ.

From section (2) we know that the principal components u_j can be expressed as a linear combination of the mapped data points by:

$$u_j = \sum_{i=1}^{n} \alpha_{ij} * \phi(x_i)$$

The projection of a new data point x into the principal component u_j is given by:

$$z_j = < u_j, (x) >$$

Substituting $\sum_{i=1}^{n} \alpha_{ij} * \phi(x_i)$ instead of u_j in z_j give us:

$$z_j = < \sum_{i=1}^{n} \alpha_{ij} * \phi(x_i) , (x) >$$

Rephrasing using the linearity of the dot product:

$$z_j = \sum_{i=1}^{n} \alpha_{ij} < \phi(x_i) , (x) >$$

Using the kernel trick we can replace $< \phi(x_i) , (x) >$ by $K(x_i, x)$:

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

This equation allow us to calculate z_j without using ϕ.

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)?

   MAKE SURE TO SAVE IT TO COLAB

The code is provided in the ipynb file.

It did better than regular PCA on the test data.