Adan Sulimany - 213298664

Muhammed Shhade - 212022263

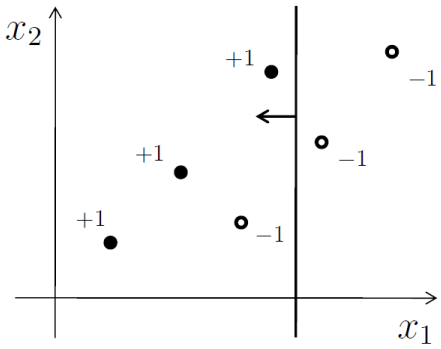
Question 4 – AdaBoost

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:

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

Consider the following dataset in $\mathbb{R}^2$:



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).

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

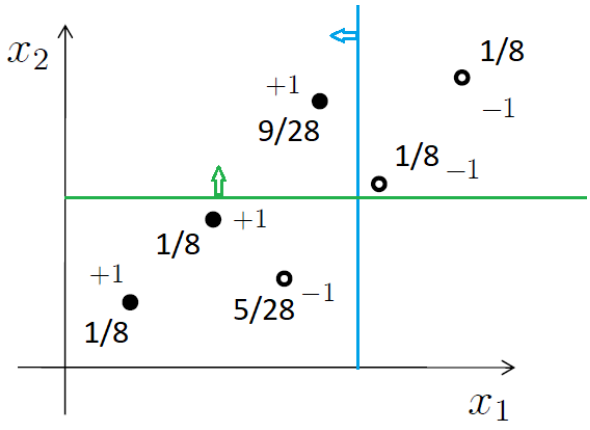
3) Draw the second decision stump. Reminder: the decision stump (our classifiers) are parallel to x/y axis.

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

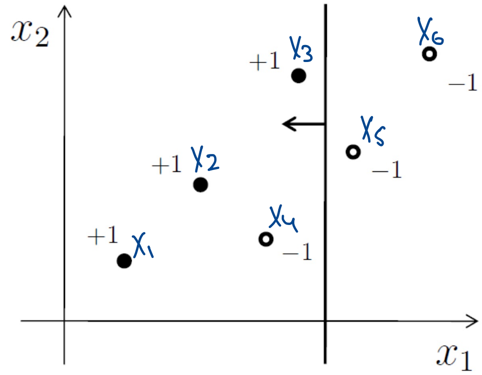
5) In the right image, there is the dataset and the weights for each point, after finding the third decision stump and calculating the new weights. Which of the following (green or blue) is the correct third decision stump?

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

   What is the train accuracy?



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).



השגיאה ε_1 היא הסכום של המשקלים של הדגימות שסווגו בצורה שגויה.
נניח שבפעם הראשונה כל הדגימות מקבלות משקל שווה ל- $\frac{1}{n}$, כאשר n
הוא מספר הדגימות הכולל. יש 6 דגימות נתונים, אז המשקל ההתחלתי של
כל דגימה הוא $\frac{1}{6}$. יש נקודה אחת שסווגה לא נכון, לכן נקבל:

$$\varepsilon_1 = \frac{1}{6}$$

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

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

המשקלים החדשים $w_i^{(2)}$ לכל דגימה מחושבים לפי הנוסחה:

$$w_i^{(2)} = w_i^{(1)} \cdot e^{(-\alpha_1 y_i f_1(x_i))}$$

כאשר y_i הוא התווית האמיתית ו- $f_1(x_i)$ הוא הפלט של הסיווג הראשון.
נחשב את המשקלים החדשים עבור הדגימות שסווגו נכון ולא נכון:

עבור דגימות שסווגו נכון:

$$w_i^{(2)} = w_i^{(1)} e^{(-\alpha_1 \cdot 1 \cdot 1)} = w_i^{(1)} e^{(-0.8)}$$

$$\Rightarrow d_1(x_c) = \frac{1}{6} \cdot e^{(-0.8)} \approx 0.074$$

עבור דגימות שסווגו לא נכון:

$$w_i^{(2)} = w_i^{(1)} \cdot e^{(-\alpha_1 \cdot 1 \cdot (-1))} = w_i^{(1)} e^{(0.8)}$$

$$\Rightarrow d_1(x_f) = \frac{1}{6} \cdot e^{(0.8)} \approx 0.37$$

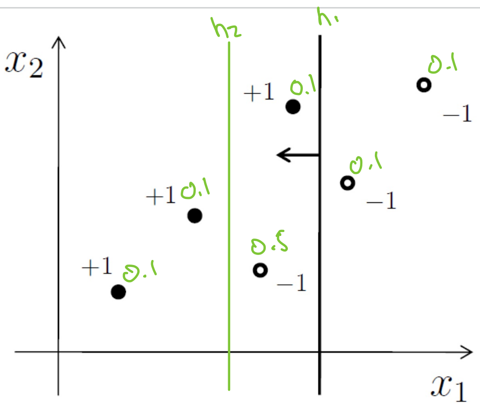
נרמול המשקלים:

$$Z_1 = 0.37 \cdot 1 + 0.074 \cdot 5 \approx 0.74$$

$$\| d_1(x_c) \| = \frac{0.074}{0.74} = \underline{0.1}$$

$$\| d_1(x_f) \| = \frac{0.37}{0.74} = \underline{0.5}$$

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

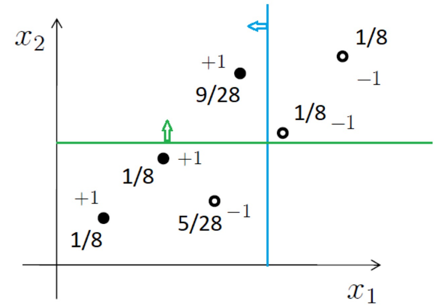


4) Without calculations, which classifier's weight is larger, α_1 or α_2? Explain why.

המשקל של כל סווג מחושב ביחס הפוך לשגיאתו. מכיוון שהסיווג השני $\varepsilon_2 = 0.1$ יש לו

שגיאה נמוכה יותר מהסיווג הראשון $\varepsilon_1 = 0.166$, נקבל ש- $\underline{\alpha_2 > \alpha_1}$

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?



$$\varepsilon_{blue} = \frac{5}{28}$$

$$\varepsilon_{green} = 0.125 \cdot 4 = 0.5$$

התשובה היא ה- decision stump הירוק. גם אם שיעור השגיאה שלו גבוה יותר משיעור השגיאה של הכחול אבל הסיווג הבא לא רק על פי שיעור השגיאה המינימלי אלא גם על פי ההשפעה הכללית על הדגימות והסיווגים האחרים. כלומר,
הדגימה x_2 סווגה לא נכון ע"י המסווג השני.
הדגימה x_4 סווגה לא נכון ע"י המסווג הראשון.
ובגלל מה המשקל שלהם גדל לכן במסווג השלישי צריך להתייחס לשאר הדגימות פחות מהדגימות האלה.
אז נגלה שבמסווג הנכון הוא הירוק.

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

What is the train accuracy?

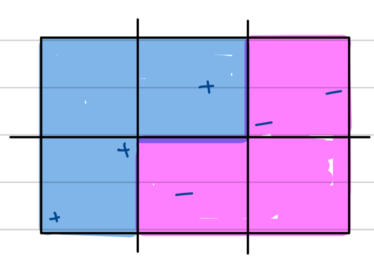
נתון: $\alpha_2 = 1.1$, $\alpha_3 = 0.62$, הסיווג המלא $g(x)$:

$$g(x) = \text{sign}\left(\sum_{t=1}^{3} \alpha_t f_t(x)\right)$$

$$\Rightarrow \text{Sign}(0.8 f_1(x) + 1.1 f_2(x) + 0.62 f_3(x))$$

המיון של הדגימות x_1, x_2, x_5, x_6 אותו מיון עבור כל המסווגים. אבל עבור x_4, x_3 לא אותו מיון עבור הכל. עבור x_3: המסווג הראשון והשלישי חיובי, אבל המסווג השני שלילי, אבל הסכום של משקלי המסווגים הראשון והשלישי גדול יותר מהמשקלים בסיווג השני לכן x_3 מסווג חיובית. וגם עבור x_4 המסווג השני והשלישי חיובית אבל הראשון לא, וגם משקל המסווגים השני והשלישי גדול יותר לכן מסווג חיובית. נקבל:

$\Rightarrow$



Question 5 – Kernel PCA (Bonus 10 points)

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, $\frac{1}{n}\sum_{i=1}^{n} x_i = 0$.

   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 won't be able to compute $v_1, \dots, v_n$).

   Therefore, we will use the kernel trick. Denote 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 $\varphi(x)$).

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 $u_1, \dots, u_k \in R^{d'}$ the first k principal components, the eigenvectors of the scatter matrix S.

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

   Moreover, show who are the $\alpha_{j,1}, \dots, \alpha_{j,k}$ such that $u_j = \sum_i \alpha_i \varphi(x_i)$.

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

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

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 $z^i = [z_1^i \quad \dots \quad z_k^i]$).

   Show how to calculate z_j <u>without</u> 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

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 won't be able to compute $v_1, \dots, v_n$).

   Therefore, we will use the kernel trick. Denote 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 will show how the kernel matrix K' can be calculated using only the orginal kernel matrix K.

firstly, we will compute K using $K_{ij} = \langle \varphi(x_i), \varphi(x_j) \rangle$.
after that, we compute the mean of $\varphi(x)$: $\mu = \frac{1}{n}\hat{\sum_{t=1}}\varphi(x_t)$

then, to find the mean-centered data v_i, we expess it as $\varphi(x_i) - \mu$

from the Reminder : $k'_{ij} = \langle v_i, v_j \rangle$

then we expand the inner product, we get that :

$$k'_{ij} = \underset{*}{\langle \varphi(x_i), \varphi(x_j) \rangle} - \underset{**}{\langle \varphi(x_i), \mu \rangle} - \underset{***}{\langle \mu, \varphi(x_j) \rangle} + \underset{****}{\langle \mu, \mu \rangle}$$

Now, will replace the values from K :

* : this is the orginal kernel matrix element , k_{ij}
** : $\langle \varphi(x_i), \frac{1}{n}\hat{\sum_{t=1}}\varphi(x_t) \rangle = \frac{1}{n}\hat{\sum_{t=1}}\langle \varphi(x_i), \varphi(x_t) \rangle = \frac{1}{n}\hat{\sum_{t=1}}k_{i,t}$
*** : $\frac{1}{n}\hat{\sum_{t=1}}k_{t,j}$
**** : this will be the mean with itself , $\frac{1}{n^2}\hat{\sum_{t=1}}\hat{\sum_{s=1}}k_{s,t}$

So, we git ,

$$k'_{ij} = k_{ij} - \frac{1}{n}\hat{\sum_{t=1}}k_{i,t} - \frac{1}{n}\hat{\sum_{t=1}}k_{t,j} + \frac{1}{n^2}\hat{\sum_{t=1}}\hat{\sum_{s=1}}k_{s,t}$$

finally, we will express this in matrix form :

$$K' = K - \frac{1}{n}1K - \frac{1}{n}K1 + \frac{1}{n^2}1K1$$

Thus, we derived the mean-centered kernel matrix K' using only the original matrix K.

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 $\alpha_{j,1}, \ldots, \alpha_{j,k}$ such that $u_j = \sum_i \alpha_i \varphi(x_i)$.

here we assume that $\varphi(x_i)$ is already mean-centered.

we will show that each u_j is a linear combination of $\varphi(x_1), \ldots \varphi(x_n)$, and to identify the coefficients $\alpha_{j,1}, \ldots \alpha_{j,k}$ such that $u_j = \sum_{i=1}^{n} \alpha_{j,i}\varphi(x_i)$.

firstly, we represent the scatter matrix S in the feature space as: $\sum_{i=1}^{n}\varphi(x_i)\varphi(x_i)^T$.

then we consider the eigenvalue equation for the scatter matrix : $Su_j = \lambda_j u_j$

we will replace this expression into the eigenvalue equation, we get:

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

next, we expand the scatter matrix S, we get:

$$\sum_{i=1}^{n}\sum_{l=1}^{n}\alpha_{j,l}\varphi(x_i)\underbrace{\langle\varphi(x_i),\varphi(x_l)\rangle}_{k_{i,l}} = \lambda_j \sum_{i=1}^{n}\alpha_{j,i}\varphi(x_i)$$

So, we get : $\sum_{l=1}^{n}k_{i,l}\alpha_{j,l} = \lambda_j \alpha_{j,i}$, and at the matrix form : $K\alpha_j = \lambda_j \alpha_j$

Thus, we show that each principal component u_j is a linear combination of $\varphi(x_1), \ldots, \varphi(x_n)$ with coefficients α_j being the eigenvectors of K.

we can express $u_j = \sum_{i=1}^{n} \alpha_{ji}\, \phi(x_i)$, where the α_j are the eigenvectors corresponding to the eigenvalue λ_j.

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.

to find an algorithm for obtaining the entire $\alpha_j = [\alpha_{j,1} \cdots \alpha_{n,n}]$ using k, we will use the previous relationship.
from part 2, each u_j can be expressed as: $\sum_{i=1}^{n} \alpha_{j,i}\, \phi(x_i)$ *
then as part 2, we can get that $K\alpha_j = \lambda_j \alpha_j$

to find α_j, we should solve the eigenvalue problem for k.

Algorithem:

1- calculate $k_{i,j} = \langle \phi(x_i), \phi(x_j) \rangle$ for all data points.
2- decompose the kernel matrix to obtain its eigenvalues λ_i and corresponding eigenvectors α_j.
3- α_j that we get represent the coefficients $\alpha_{j,1}, \ldots, \alpha_{j,n}$
4- then, use the coefficients to express each principal component u_j as *.

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 $z^i = [z_1^i \quad \dots \quad z_k^i]$).

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

Our goal is to compute $z_j = \langle u_j, \varphi(x) \rangle$ without using $\varphi(x)$.
we will use the kernel trick and q_j

firstly, we know that: $u_j = \sum_{i=1}^{n} q_{j,i} \varphi(x_i)$

next, we need to calculate the inner product $z_j = \langle u_j, \varphi(x) \rangle$, after replacing u_j we get:

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

↑ linearity of inner product

we know that: $\langle \varphi(x_i), \varphi(x) \rangle$ is equivalent to $k(x_i, x)$, we get:

$$z_j = \sum_{i=1}^{n} q_{j,i} k(x_i, x) = q_j^T k(x)$$

↑ in matrix notation

Thus, we have shown how 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

After applying kernel PCA with 'cosine' kernel and classifying with KNN on the dataset, it performed better than regular PCA on the test set. This indicates that kernel PCA with the 'cosine' kernel captured the data's structure more effectively, improving classification accuracy.