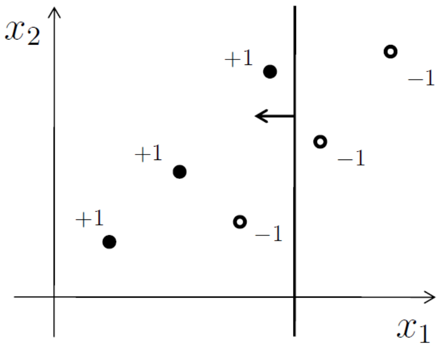
Question 4 – AdaBoost

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).

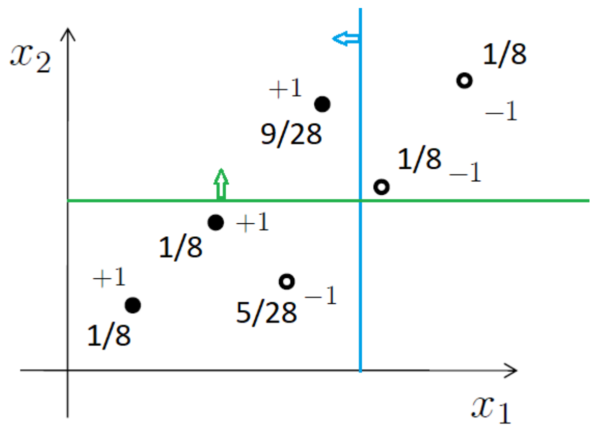
3) Draw the second decision stump. Reminder: the decision stump (our classifiers) are parallel to x/y axis.

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 <u>third</u> 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?



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