Homework no. 3 - Perceptron & Boosting

Submission requirements

Your submission must include the relevant code files together with a PDF file containing the different plots and other required answers. Also, please include your name and ID in the PDF file submitted. Lastly, you can use either Python or Matlab in this exercise.

Ex. 1 - Perceptron with RBF and Polynomial kernels

In this exercise we implement the kernalized version of the perceptron algorithm shown in class.

Algorithm 1 Kernelized Perceptron

Input: A sample $S = \{(\mathbf{x}^{(i)}, y_i)\}_{i=1}^m$.
Initialize: $\bar{\bar{\alpha}} = 0 \in \mathbb{R}^m$, $K_{i,j} = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$.

For t=1 to T
For i=1 to m
Set $\hat{y}_i = \text{sgn}(\bar{\alpha} K_i)$ $\%K_i$ is the i-th column of K
Update

$$\bar{\alpha}_i \mathrel{+}= \frac{1}{2}(y_i - \hat{y}_i) \qquad \text{i.e.}\%\mathbf{w}_i = \begin{cases} \mathbf{w}_{i-1} & y_i = \hat{y}_i \\ \mathbf{w}_{i-1} - \hat{y}_i \Phi(\mathbf{x}_i) & y_i \neq \hat{y}_i \end{cases}$$

End For
End For

The train and test data are given in train.csv and test.csv. If you are using python with numpy you can use the command `train = np.loadtxt('train.csv', delimiter=',')` which will store an $m \times 3$ matrix where the first two columns represent the input x with dimension $d = 2$ and the last column is the labeling of the respective input data. We assume that the inputs (x, y) are defined in $x \in X = [-1, 1] \times [-1, 1]$ and $y \in Y = \{-1, 1\}$

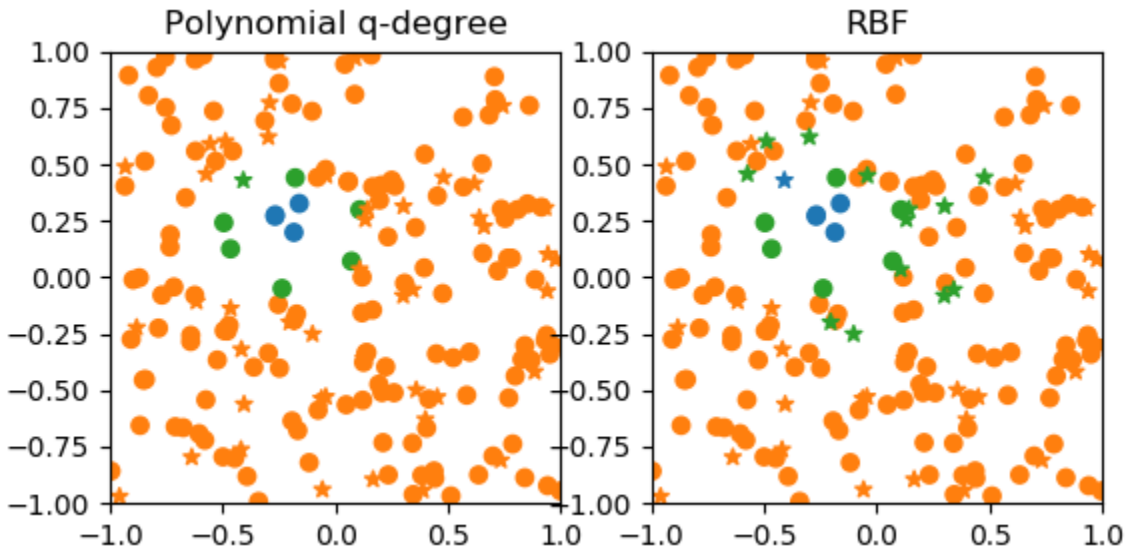
1. Load the train.csv file and add the 2D scatter plot of it, while setting different colors for each of the labels ± 1.
2. Build the RBF(radial basis kernel) for some σ and the polynomial basis kernel for some degree q.
3. Implement the Kernalized Perceptron algorithm, and apply it on the train data set once with the RBF kernel and once with the Polynomial kernel.

4. Try to optimize the number of iterations T and the different parameters of degree q or σ such that for both kernals the empirical error is 0 (In other words, it is an ERM algorithm). Cross-validation is not required, just play with the parameters until you are satisfied. Write down the final parameters you chose in the PDF report.

5. Load the test.csv and estimate the labels, using both classifiers (RBF and Polynomial) that you optimized.

6. Scatter plot the test set and assign a color based on the estimated labels over the test set. On the same plot, add also the train set and assign a color based on the estimated labels over the train set. To differentiate between the train and test plots, set a different marker for each (for example using Python and matplotlib `scatter(X_test[:, 0], X_test[:, 1], marker='*', c=c)`). Lastly, for each point that we misclassified for either the test and train set, assign it a third different color.

Figure 0.1: Example of such a plot: where the circle marker represents a train set point and the star marker represents a test set point. The color orange is a correct classification of -1 and blue is a correct classification of $+1$, green is a misclassified point.



7. Print and add to the report, the number of errors that your classier had on the test set and on the train set for each of the two classifiers you trained.

Ex. 2 - Boosting

In this exercise, we will study the performance of the AdaBoost on the MNIST dataset. We will investigate how well we can classify a digit to 0 or 8. We divide the data into training set and test set where the label $+1$ represents the digit '8' and label -1 represents the digit '0'. To load the data with numpy use `X_train = np.loadtxt(fname='MNIST_train_images.csv', delimiter=',')` and do the same for the labels. Note that X_train is a 2D array where each row is a sample of dimension $d = 28 \times 28 = 784$ since each sample is a reshaped image of size 28×28.

1. Choose an arbitrary sample from the train set, reshape it to an image of size 28 and plot it. To do so with matplotlib use `plt.imshow(np.asarray(np.reshape(X_train[index, :], (28, 28))), cmap='gray', vmin=0, vmax=255)`. Add this plot to the PDF report and note the chosen index.

2. Implement the AdaBoost algorithm. The class of weak learners we will use is the class of hypothesis of the form

$$h_{\theta,j}(x) = \begin{cases} 1 & x(j) \leq \theta \\ -1 & x(j) > \theta \end{cases} \quad \text{and} \quad h_{\theta,j}(x) = \begin{cases} -1 & x(j) \leq \theta \\ 1 & x(j) > \theta \end{cases}$$

   for any $j \in [784]$. That is, comparing a single pixel to a threshold. At each iteration, AdaBoost will select the best weak learner.

Algorithm 2 AdaBoost

Input: A sample $S = \{(x_i, y_i)\}_{i=1}^m$ where $x_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$ and a class of weak learners $\mathcal{H}$.

Initialization

$p_1(i) = \frac{1}{m} \quad \forall i \in \{1, \ldots, m\}$ where $p_t(i)$ is the weight probability of example (x_i, y_i) at iteration t.

for $t = 1, 2 \ldots T$ **do**

Find $h_t = \arg\min_{h \in \mathcal{H}} \mathbb{P}_{x \sim p_t}[h(x) \neq y]$.

Set $\epsilon_t = \mathbb{P}_{x \sim p_t}[h_t(x) \neq y]$.

Set $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.

Update $p_{t+1}(i) = \frac{p_t(i) \exp\big(-\alpha_t y_i h_t(x_i)\big)}{\sum_{j=1}^m p_t(j) \exp\big(-\alpha_t y_j h_t(x_j)\big)}$ for all $i \in \{1, \ldots, m\}$.

end for

return $h(x) = \text{sign}\big(\sum_{t=1}^T \alpha_t h_t(x)\big)$

Note that $\mathbb{P}_{x \sim p_t}[h(x) \neq y] \triangleq \sum_{i=1}^m p_t(i) \cdot \mathbb{1}\{h(x_i) \neq y_i\}$, thus in iteration t one can find the best weak learner h_t since the number of pixels d is finite and also the possible thresholds θ candidates per pixel are $m + 1$ (finite).

- Run AdaBoost with $T = 30$. Show plots for the training error and the test error of the classifier implied at each iteration t (Namely, $\text{sign}\big(\sum_{s=1}^t \alpha_s h_s(x)\big)$).

Remark: try to vectorize most operations for a more time efficient implementation.