Assignment 5

PAC-Bayesian Aggregation (75 points)

Our goal is to repeat the expiriment illustrated in figure 2, section 6 in the paper by Thiemann, Igel, Wintenberger and Seldin on the Ionosphere dataset. We take a sligthly different approach, where we redo the expiriment several times, and aggregate the results, while also providing a standard deviation on the test loss.

Data-handling, cross-validation and choice of paramter grid

We first split the training data into training and test data, with the `train_test_split` function from `sklearn.model_selection` module. We do the same split as the article does - that is 150 testing examples, and the rest as training examples. For the cross-validation, we use the same parameter grid as the article. For the regularisation term C we try $C \in \{10^{-3}, 10^{-2}, \cdots, 10^2, 10^3\}$. For the kernel bandwith parameter γ we use the grid $\gamma \in \{10^{-4}, \gamma_J 10^{-2}, \cdots, \gamma_J 10^2, \gamma_J 10^4\}$, where $\gamma_J = \frac{1}{2\text{median}(G)}$, with $G(X_i) = \min_{(X_j, Y_j) \in S \wedge Y_i \neq Y_j} ||X_i - X_j||$. We compute G by splitting the training data according to the labels, and use the `scipy.spatial` function `distance_matrix` on the two arrays. We get $\gamma_J = 0.1295$.

We do 5-fold cross-validation using the above grid, utilizing the `GridSearchCV` class from `sklearn.model_selection`. For the SVM implementation, we use the `SVC` class from `sklearn.SVM` with the `kernel` parameter set to rbf for a Gaussian kernel. We get the optimal (in terms of cross validation loss) parameter configuration $\gamma = 0.1295$ and $C = 10$. The test loss of this classifier is 0.06. We also time the process of cross-validation and training of the final classifier on all training points

Computation of randomized classifier

We train weak classifiers using the following procedure. We sample $r = d + 1 = 35$ points without replacement in the training data for training the SVM, and use the rest for validation. We use the same value of $C = 10$ as obtained in the cross-validation, but sample γ uniformly from the parameter grid specified above for each hypothesis. This is done in consistency with the article. This procedure is implemented in the `sample_r` function attached in the source code. This returns a list containing the classifier, and it's validation loss.

This process is then repeated m times, giving us an array of classifiers and their corresponding validation losses. We then minimize the bound in theorem 6 in the paper, by alternating application of equation (7) and (8), in the paper. That is, we minimize

$$\frac{\mathbb{E}_\rho(\hat{L}^{\text{val}}(h, S))}{1 - \frac{\lambda}{2}} + \frac{\text{KL}(\rho||\pi) + \log\left(\frac{2\sqrt{n-r}}{\delta}\right)}{\lambda\left(1 - \frac{\lambda}{2}\right)(n-r)}$$

We choose a uniform prior $\pi(h) = 1/m$, and $\delta = 0.05$. Practically, we minimize the bound by initialising $\lambda = 1$, because we know per (8) that the optimal λ is less than 1. We then compute the

bound-minimizing distribution given λ,

$$\rho_\lambda(h) = \frac{\pi(h)e^{-\lambda(n-r)\left(\hat{L}^{\text{val}}(h,S)-\hat{L}^{\text{val}}_{\min}\right)}}{\sum_{h'}\pi(h')e^{-\lambda(n-r)\left(\hat{L}^{\text{val}}(h',S)-\hat{L}^{\text{val}}_{\min}\right)}}$$

And afterwards compute the λ minimizing the bound given ρ by,

$$\lambda_\rho = \frac{2}{\sqrt{\frac{2(n-r)\mathbb{E}_\rho(\hat{L}^{\text{val}}(h,S))}{\text{KL}(\rho||\pi)+\log\left(\frac{2\sqrt{n-r}}{\delta}\right)}+1}+1}$$

We do this iteratively, terminating the process when the distance between successive λ's is small. This is implemented in the function `weight_classifier` in the source code, which takes training examples and m as inputs, and returns the distribution ρ along with the classifiers.

Calculation of test loss and bound & results

To do a prediction using the randomized classifier obtained from the above procedure, we sample a weak classifier according to ρ, and use that classifier's prediction as the output of the randomized classfier. To calculate the test loss we do this for every point in the test dataset, and compute the 0-1 loss for each of these points. We also compute the bound on the randomized classifier corresponding to theorem 2 in the paper, based on the test loss. We invert kl, numerically as we did in assignment 2. That is, we compute the bound,

$$\mathbb{E}_\rho(L(h)) \leq \text{kl}^{-1^+}\left(\mathbb{E}_\rho(\hat{L}_{\text{test}}(h,S)), \frac{\text{KL}(\rho||\pi)+\log\left(\frac{2\sqrt{n_{\text{test}}}}{\delta}\right)}{n_{\text{test}}}\right)$$

Reusing the notation used in that assignment. We also compute the time it takes to train the weak classifiers and compute the probability distribution over these. All of this is implemented in the `predict` function, which takes a randomized classifier generated by the `weight_classifer` function as well as a test dataset as input, and outputs the training time, the test loss, and the theorem 2 bound on the expected loss. We train randomized classifiers for values of $m \in \{1, 4, 7, \cdots, 198\}$, 60 times. We then compute the mean of the test loss, the mean of the bound and the mean of the training time for each m. We also compute the standard deviation of the test loss for each m. Below we plot the results of the experiment, where the shaded area around the test loss, is the test loss plus-minus one standard deviation.

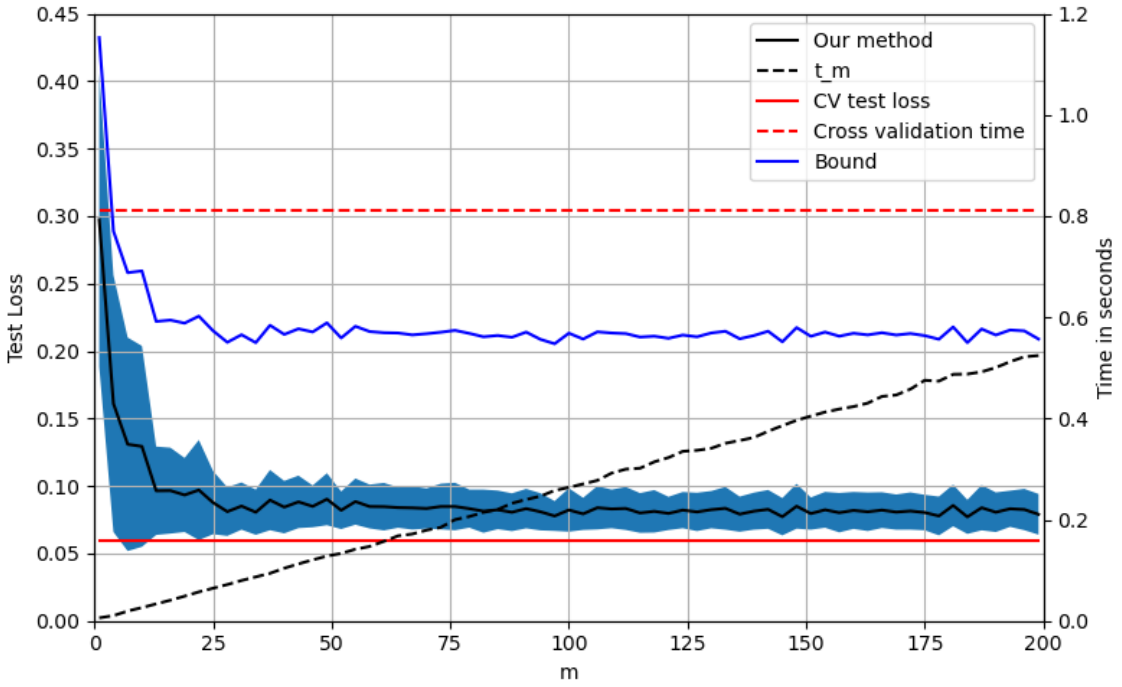

Figure 1: Results of experiment