CMSI 5350
Problem Set 4

10/30/23
Denis Onwualu

1 Theory: Boosting (6 pts)

1. Give a one-sentence reason for why AdaBoost outperforms a single decision stump.

2. Consider building an ensemble of decision stumps with the AdaBoost algorithm. Figure 1 displays the labeled points in two dimensions as well as the first stump we have chosen. The little arrow in the figure is normal to the stump decision boundary, indicating the positive side where the stump predicts +1. All the points start with uniform weights.

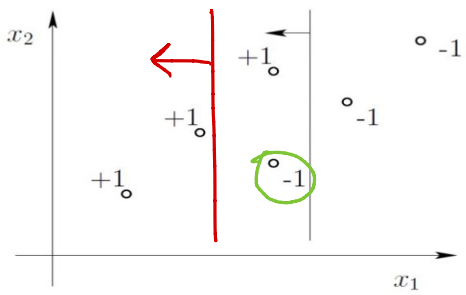

Figure 1: Boosting.

(a) Circle all the point(s) in the figure whose weight will increase as a result of incorporating the first stump.

(b) Draw in the same figure a possible stump that we could select at the next boosting iteration. You need to draw both the decision boundary and its positive orientation (as in the figure for the first stump).

(c) Will the second stump receive a higher coefficient in the ensemble than the first? In other words, will $\alpha_2 > \alpha_1$? Briefly explain your answer. (No calculation should be necessary.)

c. yes, $\alpha_2 > \alpha_1$ because it measures the importance of $h(x^{(i)}; \theta_t)$. Since our points will be increased in weight, α_2 will be greater.

(d) Let us now consider replacing decision stumps with different base classifiers. Suppose the training set is linearly separable, and we use a hard-margin linear support vector machine (no slack) as a base classifier. In the first boosting iteration, what would the resulting $\hat{\alpha}_1$ be?

d) α_1 would be 1, because it will classify the training set perfectly.

ChatGPT (1 pt)

Read through the rest of the assignment. Write down at least one use case for ChatGPT (e.g., explaining a function from a library that you're not familiar with, providing sample code for how to use the function, or debugging your code), and then try it out! Reflect on the output. Was it accurate? Helpful? Did it increase your understanding or make you faster at writing code?

I asked ChatGPT to explain the purpose of the hyperparameter "C". It was helpful because I learned that the implementation of this parameter is inverse to regularization. This increased my knowledge because in class, we learned the opposite. So I initially thought it was directly related to regularization.

2. You may have noticed that the proportion of the two classes (positive and negative) are not equal in the training data. When dividing the data into folds for CV, you should try to keep the class proportions roughly the same across folds. In `main(...)`, use `sklearn.cross_validation.StratifiedKFold(...)` to split the data for 5-fold CV, making sure to stratify using only the training labels. You should shuffle the data and set the random state. Briefly describe why it might be beneficial to maintain class proportions across folds.

It would be beneficial to maintain proportions in order to reduce bias. If class distribution is imbalanced, certain folds might have way fewer instances of a specific class.

3. Now, implement `select_param_linear(...)` to choose a setting for C for a linear SVM based on the training data and the specified metric.

4. Finally, using the training data from the feature extraction part and the functions implemented here, find the best setting for C for each performance measure mentioned above. Report your findings in tabular format (up to the fourth decimal place):

C	accuracy	F1-score	AUROC	precision	sensitivity	specificity
10^{-3}	.6553	.7972	.8686	.6553	1.0	0.0
10^{-2}	.7857	.8535	.8690	.7731	.9537	.4669
10^{-1}	.825	.8720	.9066	.8372	.9100	.6634
10^{0}	.8661	.8992	.9187	.8845	.9154	.7721
10^{1}	.8625	.8968	.9144	.8798	.9154	.7618
10^{2}	.8625	.8968	.9144	.8798	.9154	.7618
best C	1.0	1.0	1.0	1.0	.001	1.0

Your `select_param_linear(...)` function returns the 'best' C given a range of values. How does the 5-fold CV performance vary with C and the performance metric (note that the sklearn documentation states that "the strength of the regularization is inversely proportional to C")? If you observe anything interesting or surprising, comment on what might have caused these results.

For most of the metrics, performance increased directly with C. Then began to degrade once C got too high. I was surprised that the performance for the sensitivity metric was 100% with the lowest C value. This means there was a massive amount of regularization during that run.

1. Based on the results you obtained in the table above, choose a hyperparameter setting for the linear-kernel SVM. Explain your choice.

C = 1.0 because majority of the metrics had the best score for this value of hyperparameter.

(Optional) Hyperparameter Selection for RBF-kernel SVM (+3 pt)

Similar to the hyperparameter selection for a linear-kernel SVM, you will perform hyperparameter selection for an RBF-kernel SVM (`kernel='rbf'`). This time, you will search across `C` *and* `gamma`.

The `scikit-learn` documentation tells us that the RBF kernel is specified by $\exp(-\gamma||x-z||^2)$, where $\gamma > 0$. The γ hyperparameter is related to the "spread" of the RBF kernel (inversely proportional to the variance, i.e. $\gamma = 1/\sigma^2$). As γ increases, the variance decreases and the RBF kernel results in 'tighter' decision boundaries. In other words, as γ increases, the algorithm tries harder to avoid misclassifying training data. Thus, as γ continues to increase the classifier will start to overfit.

1. Implement `select_param_rbf(...)` to choose a setting for C and γ via a grid search (just as we did for you in the linear-kernel SVM for selecting C). Your function should call `cv_performance(...)`, passing in instances of `SVC(kernel='rbf', C=c, gamma=gamma)` with different values for `C` and `gamma`.

2. Finally, using the training data from the feature extraction part and the function implemented here, find the best setting for C and γ for each performance measure mentioned above. Report your findings in tabular format. This time, because we have a two-dimensional grid search, report only the best score for each metric, along with the accompanying C and γ setting:

metric	score	C	γ
accuracy	.868	100	.01
F1-score	.900	100	.01
AUROC	.921	100	.01
precision	.886	100	.01
sensitivity	1.0	.001	10^{-3}
specificity	.778	100	.01

How does the CV performance vary with the hyperparameters of the RBF-kernel SVM?

It seems like a higher C value & a lower γ value is preferred for optimization.

3. For each performance metric, use `performance_CI(...)` and the trained linear-kernel SVM classifier to measure performance on the test data. Report the results. Be sure to include the name of the performance metric employed, the performance on the test data, and the corresponding confidence interval.

	score	95% interval
accuracy	.71	.65 – .84
f1	.74	.74 – .90
auroc	.75	.68 – .92
precision	.82	.77 – .96
sensitivity	.76	.66 – .89
specificity	.61	.45 – .89

Feature Importances (2 pts)

1. Now let us consider your trained linear-kernel SVM. How would you use the coefficients of the learned classifier to find the most important features (i.e., words) for sentiment analysis?

I would look at the features with the larger magnitude coefficients. These are considered more important.

2. In order of rank, e.g. from most important to least important, list the 10 most important features for tweets with positive sentiment and the 10 most important features for tweets with negative sentiment. Comment on the resulting features.

   Note that one limitation of this approach for determining feature importance is we have used a bag-of-words model that takes on only binary values (to indicate presence or absence). Therefore, we may

Positive

1.) liked
2.) excellent
3.) rocked
4.) end
5.) good
6.) awesome
7.) d
8.) funny
9.) must
10.) great

Negative

1.) shit
2.) not
3.) hated
4.) disappointing
5.)]
6.) hilariooooowss
7.) all
8.) boring
9.) 2012
10.) complete

Some of these features can definitely apply to both positive & negative reviews. It's interesting that a "bracket" or "]" can be considered negative. It could be a part of a sad face and/or emoji.

Explanation (3 pts)

Suppose you had to explain the results of your learning to someone not familiar with computer science or machine learning (e.g., a businessman or film director). Using a single paragraph plus one optional graph or table, explain how this machine learning model works and what you have found (1 pt for problem statement; 1 pt for experimental setup; 1 pt for explaining how to find the important words; -1 pt for using jargon). Do NOT use technical jargon or describe performance; rather, describe what your model's *learned* about tweets.

There's a murder mystery and all we have to solve the crime are these tweets. This model looks at all the clues (words) and assigns a number to it that says how important each word is is to solving the case. The model adds up the numbers for all the pieces and tries to make a decision about the mystery. Our model only knows how likely each tweet corresponds to each culprit in the case, and over time it begins to make decisions about the case based on how many words in the tweets.