CIS 419/519 Applied Machine Learning
Homework 4

Due: March 2, 2020 11:59pm

Instructions

Read all instructions in this section thoroughly.

Collaboration: Make certain that you understand the collaboration policy described on the course website.

You should feel free to talk to other members of the class in doing the homework. I am more concerned that you learn how to solve the problem than that you demonstrate that you solved it entirely on your own. You may *discuss* the homework to understand the problems and the mathematics behind the various learning algorithms, but you must **write down your solutions yourself**. In particular, **you are not allowed to share problem solutions or your code with any other students.**

We take plagiarism and cheating very seriously, and will be using automatic checking software to detect academic dishonesty, so please don't do it.

You are also prohibited from posting any part of your solution to the internet, even after the course is complete. Similarly, please don't post this PDF file or the homework skeleton code to the internet.

Formatting: This assignment consists of two parts: a problem set and program exercises.

For the problem set, you must write up your solutions electronically and submit it as a single PDF document. We will not accept handwritten or paper copies of the homework. Your problem set solutions must use proper mathematical formatting.

All solutions must be typeset using LaTeX; handwritten solutions of any part of the assignment (including figures or drawings) are not allowed.

Your programming solutions must be implemented in python, following the precise instructions included in Part 2. Portions of the programming exercise will be graded automatically, so it is imperative that your code follows the specified API. A few parts of the programming exercise asks you to create plots or describe results; these should be included in the same PDF that you create for the problem set.

Homework Template and Files to Get You Started:

- https://www.seas.upenn.edu/~cis519/spring2020/homework/hw4_skeleton.ipynb
- https://www.seas.upenn.edu/~cis519/spring2020/homework/ChocolatePipes_trainData.csv
- https://www.seas.upenn.edu/~cis519/spring2020/homework/ChocolatePipes_trainLabels.csv
- https://www.seas.upenn.edu/~cis519/spring2020/homework/ChocolatePipes_leaderboardTestData.csv
- https://www.seas.upenn.edu/~cis519/spring2020/homework/ChocolatePipes_gradingTestData.csv

Citing Your Sources: Any sources of help that you consult while completing this assignment (other students, textbooks, websites, etc.) ***MUST*** be noted in the your PDF file. This includes anyone you briefly discussed the homework with. If you received help from the following sources, you do not need to cite it: course instructor, course teaching assistants, course lecture notes, course textbooks or other readings.

Submitting Your Solution: You will submit this assignment through Gradescope as two parts: the written answers, and your python code for the programming exercise.

CIS 519 ONLY Problems: There are no CIS 519 ONLY problems in this assignment.

Philosophy Behind This Homework We are now moving into the portion of the course where we're really focusing on applications. The problem set will give you some further practice with SVMs. The programming exercise is a challenge, permitting you to test your ability to learn a classifier and apply it to unseen data. Like in the real world, your success (i.e., your grade) will be partially based on how well the ML model does on new data!

PART I: PROBLEM SET

Your solutions to the problems will be submitted as a single PDF document. Be certain that your problems are well-numbered and that it is clear what your answers are.

1 Fitting an SVM by Hand (15 pts)

[Adapted from Murphy & Jaakkola] Consider a dataset with only 2 points in 1D: $(x_1 = 0, y_1 = -1)$ and $(x_2 = \sqrt{2}, y_2 = +1)$. Consider mapping each point to 3D using the feature vector $\phi(x) = [1, \sqrt{2}x, x^2]^\intercal$ (i.e., use a 2nd-order polynomial kernel). The maximum margin classifier has the form

$$\min \|\boldsymbol{w}\|_2^2 \text{ s.t.}$$
$$y_1(\boldsymbol{w}^\intercal \phi(x_1) + w_0) \geq 1$$
$$y_2(\boldsymbol{w}^\intercal \phi(x_2) + w_0) \geq 1$$

a.) Write down a vector that is parallel to the optimal vector $\boldsymbol{w}$. (Hint: recall that $\boldsymbol{w}$ is orthogonal to the decision boundary between the two points in 3D space.)

b.) What is the value of the margin that is achieved by $\boldsymbol{w}$? (Hint: think about the geometry of two points in space, with a line separating one from the other.)

c.) Solve for $\boldsymbol{w}$, using the fact that the margin is equal to $\frac{2}{\|\boldsymbol{w}\|_2}$.

d.) Solve for w_0, using your value of $\boldsymbol{w}$ and the two constraints (2)–(3) for the max margin classifier.

e.) Write down the form of the discriminant $h(x) = w_0 + \boldsymbol{w}^\intercal \phi(x)$ as an explicit function in terms of x.

2 Support Vectors (5 pts)

For an SVM, if we remove one of the support vectors from the training set, does the size of the maximum margin decrease, stay the same, or increase for that dataset? Why? (Explain your answer in 1-2 sentences.) (Hint: Using some scratch paper, you may find it helpful to draw a few simple 2D dataset in which you identify the support vectors, draw the location of the maximum margin hyperplane, remove one of the support vectors, and draw the location of the resulting maximum margin hyperplane. You do not need to submit any drawings with your answer.)

PART II: PROGRAMMING EXERCISES

3 Challenge: Generalizing to Unseen Data

One of the most difficult aspects of machine learning is that your classifier must generalize well to unseen data. In this exercise, we are supplying you with labeled training data and *unlabeled* test data. Specifically, you will *not* have access to the labels for the test data, which we will use to grade your assignment. You will fit the best model that you can to the given data and then use that model to predict labels for the test data. It is these predicted labels that you will submit, and we will grade your submission based on your test accuracy (relative to the best performance you should be able to obtain).

You will submit three sets of predictions: one based on a boosted decision tree classifier (which you will write), one using an SVM, and another set of predictions based on whatever machine learning method you like – you are free to choose any classification method (ones that we've covered in class or otherwise).

There are two sets of test data: one that we will use for a leaderboard and another that will be used for grading. When you submit your predictions, you will see your leaderboard performance, allowing you to see how well you are doing and what level of performance is feasible on the "leaderboard" test data.

Your grade will partially be based upon your test accuracy on the "grading" test data and the true test labels. However, you will not be receiving feedback on your "grading" test performance at the time of submission, so you must do your best without direct feedback on your "grading" test performance.

3.1 The Challenge Data

Although we've altered the data set to make the problem more fun, it is based on real data.

Willy Wonka has been experiencing profit loss due to defective chocolate pipes, and has decided to send his Oompa-Loompas on a mission to find why the pipes are going defective. They have been learning about machine learning and have chosen to employ ML algorithms to determine which pipes are defective. They collected data about pipes and the chocolate that is used to make the sweets and candy. The lucky Oompa-Loompa who provides a reliable model gets to use the Wonkavator and becomes free.

The collected data explores the chocolate quality, quantity, source of the cocoa and such. Beware–some of the features, although numerical, are actually categorical because the Oompa-Loompas didn't want to disclose their actual sources. Therefore, you need to carefully consult the feature descriptions below when figuring out how to process the data. There are also missing values, and a host of other issues with the data.

Categorical Labels

- 0 - Pipe is not functional
- 1 - Pipe is functional but needs repair
- 2 - Pipe is fully functional

Feature Names and Descriptions

- **Chocolate_quality**: A categorical variable that represents the quality of the chocolate extracted from this pipe (Note: a higher quality number doesnât necessarily mean better)
- **Chocolate_quantity**: A categorical variable that represents the quantity of chocolate extracted from this pipe (Note: higher quantity number doesnât necessarily mean more)
- **Source**: A categorical variable that represents the specific source of the chocolate (eg chocolate river)
- **Source_class**: A categorical variable that represents the class of source that the chocolate comes from (e.g., ground)
- **Pipe_type**: A categorical variable that represents the type of the pipe
- **Size of the chocolate pool**: A numerical variable representing the size of the chocolate pool.
- **Date recorded**: Denotes the date during which the data was surveyed.
- **Funder**: This is the country which gave money to fund the pipes. This is categorical value
- **GPS_height**: It is the height of the pipes, this is a numerical value
- **Installer**: The company which installed the pipes , This is a categorical value.
- **District code**: This is Geographic location represented in coded format.
- **Location**: This is Geographic location. This is a categorical feature.
- **Chocolate consumers in town**: The number of chocolate lovers in an area.
- **Does factory offer tours**: It is a boolean feature that shows if a factory offers tours.
- **Recorded by**: A categorical feature that shows who entered this row of data
- **Oompa loompa management**: A categorical feature that shows who operates the factory
- **Latitude**: GPS coordinate
- **Name of pipe**: Name of the pipe.
- **Cocoa Farm**: The original farm of the cocoa. This is a categorical feature.
- **Country of Factory**: The country where the factory is located. This is a categorical feature.
- **Region Code**: The region code. This is a categorical feature.
- **Official or Unofficial pipe**: Pipe was constructed with or without permit
- **Year constructed**: The year in which pipe was constructed
- **Type of pump**: A categorical feature representing the pumping method used by the pipe
- **Payment scheme**: A categorical feature representing various payment options
- **Management**: How the chocolate pool is managed, as a categorical feature.
- **Management_group**: The group that manages the chocolate pool.

3.2 The Boosted Decision Tree Classifier (30 pts)

In class, we mentioned that boosted decision trees have been shown to be one of the best "out-of-the-box" classifiers. (That is, if you know nothing about the data set and can't do parameter tuning, they will likely work quite well.) Boosting allows the decision trees to represent a much more complex decision surface than a single decision tree.

Write a class that implements a boosted decision tree classifier. Your implementation may rely on the decision tree classifier already provided in scikit_learn (`sklearn.tree.DecisionTreeClassifier`), but you must implement the boosting process yourself. (The scikit_learn module actually provides boosting as a meta-classifier, but you may not use it in your implementation.) Each decision tree in the ensemble should be limited to a maximum depth as specified in the `BoostedDT` constructor. You can configure the maximum depth of the tree via the `max_depth` argument to the `DecisionTreeClassifier` constructor.

Your class must implement the following API:

- `__init__(numBoostingIters = 100, maxTreeDepth = 3)`: the constructor, which takes in the number of boosting iterations (default value: 100) and the maximum depth of the member decision trees (default: 3)
- `fit(X,y,random_seed)`: train the classifier from labeled data (X, y), where both are Pandas data frames, with an optional random seed for the decision tree classifiers
- `predict(X)`: return a $n \times 1$ Pandas dataframe of predictions for each instance of the data frame X

Note that these methods have already been defined correctly for you in `boostedDT`; be very careful not to change the API. You should configure your boosted decision tree classifier to be the best "out-of-the-box" classifier you can; you may not modify the constructor to take in additional parameters (e.g., to configure the individual decision trees).

There is one additional change you need to make to AdaBoost beyond the algorithm described in class. AdaBoost by default only works with binary classes, but in this case, we have a multi-class classification problem. One variant of AdaBoost, called AdaBoost-SAMME, easily adapts AdaBoost to multiple classes. Instead of using the equation $\beta_t = \frac{1}{2}\ln\left(\frac{1-\epsilon}{\epsilon}\right)$ in AdaBoost, you should use the AdaBoost-SAMME equation

$$\beta_t = \frac{1}{2}\left(\ln\left(\frac{1-\epsilon}{\epsilon}\right) + \ln(K-1)\right) \quad ,$$

where K is the total number of classes. This will force $\beta_t \geq 0$ as long as the classifier is worse than random guessing (in this case random guessing would be $1/K$, so the error rate would need to be greater than $1 - 1/K$). Note that when $K = 2$, AdaBoost-SAMME reduces to AdaBoost. For further information on SAMME, see http://web.stanford.edu/~hastie/Papers/samme.pdf.

Test your implementation by running `test_boostedDT()`, which compares your `BoostedDT` model to a regular decision tree on the iris data with a 50:50 training/testing split. You should see that your `BoostedDT` model is able to obtain ~97.3% accuracy vs the 96% accuracy of regular decision trees. Make certain that your implementation works correctly before moving on to the next paragraph.

Once your boosted decision tree is working, train your `BoostedDT` on the labeled data available in the files `ChocolatePipes_trainData.csv` and `ChocolatePipes_trainLabels.csv`. You may tune the number of boosting iterations and maximum tree depth however you like. Measure your training accuracy, estimate your generalization accuracy, and tune your classifier to be as best as possible. Save your training accuracy and estimate of the generalization accuracy, since you'll need to report them later.

Once you're finished tuning, use the trained `BoostedDT` classifier to predict a label $y \in \{0, 1, 2\}$ for each unlabeled instance in `ChocolatePipes_leaderboardTestData.csv`. Save your predictions on the test data into a csv file named `predictions-leaderboard-BoostedDT.csv` with two columns: *id* and *label*. Repeat this same process to predict labels for the instances in `ChocolatePipes_gradingTestData.csv`, saving the predictions to a csv file named `predictions-grading-BoostedDT.csv`.

The format of your predictions files should match `ChocolatePipes_trainLabels.csv` and the number of predictions should match the number of unlabeled test instances.

Notice that you're providing predictions for two different sets of test data. The "leaderboard test data" will be used when you submit to display a leaderboard of everyone's performance, giving you an idea of how well you're doing and what level of performanc is achievable. Your accuracy on the "grading test data" is what we will use to assign a grade for this component.

3.3 Comparing Boosted Decision Trees with SVMs (10 pts)

Use sklearn's implementation of the support vector machine (sklearn.svm.SVC) to train a model for the challenge data, tuning the model parameters and your choice of kernel. You may find it useful to consult https://scikit-learn.org/stable/modules/svm.html.

Again, measure the SVC's training accuracy and estimate the SVC's generalization accuracy, and generate a set of predictions for the leaderboard and grading test data. Save your predictions into csv files named, respectively, `predictions-leaderboard-SVC.csv` and `predictions-grading-SVC.csv`, each with two columns: *id* and *label*. Double-check the format of your predictions files and make certain you have include a prediction for each test instance.

3.4 Training the Best Classifier (10 pts)

Now, train the very best classifier you can for the challenge data, and use that classifier to output predictions for all test instances. You may use any machine learning algorithm you like, and may tune it any way you wish. You may use the methods and helper functions built into scikit_learn; you do not need to implement the method yourself, but may if you wish. If you can think of a way that the unlabeled test data would be useful during the training process, you are welcome to let your classifier have access to it during training.

Note that you will not be submitting an implementation of your optimal model, just its predictions.

Once again, measure the training accuracy and estimate the generalization accuracy of your best classifier, and generate a set of predictions for the leaderboard and grading test data. Save your predictions into csv files named, respectively, `predictions-leaderboard-best.csv` and `predictions-grading-best.csv`, each with two columns: *id* and *label*. Double-check the format of your predictions files and make certain you have include a prediction for each test instance.

If you believe that your boostedDT classifier is actually the best set of predictions for this challenge data, then you would submit the boostedDT predictions twice (and have two identical files of predictions). The same goes for your SVC predictions if you think the SVC is the best classifier you can obtain.

3.5 Comparing Your Algorithms (10 pts)

Write three brief paragraphs, 3–4 sentences each:

Preprocessing Describe how you preprocessed the challenge data in detail. It should be possible to reconstruct your preprocessing procedure entirely from this paragraph.

The Best Classifier Describe the best machine learning classifier you found, its optimal parameter settings (if any), and how you trained the model. If you are using a method that we haven't covered in class, you must briefly describe how that method works.

Results and Discussion Create a table comparing the training and expected generalization performances of your three methods. (Even if your best classifier is either boostedDT or SVC, you must include at least one other method for comparison in the table). In a brief paragraph, discuss your results.

Include the three paragraphs and the table in your PDF writeup.