CS282BR - Homework 1

Due: September 30, 2019, 11.59pm ET

Guidelines for Submission

Writeup

Please include your name, your email, collaborator names, and the nature of collaborations[1] on the first page of your writeup. Try to be as concise as possible when answering questions. Whenever you use ideas or code from other resources (papers/websites), acknowledge them. Whenever you make any assumptions or design decisions, please mention those explicitly in your writeup. Please remember – **your write-up must be your own**.

Collaboration Policy

It is perfectly fine to discuss your ideas with other students in class. However, it is important to acknowledge these collaborations. If you coded with others, you must have been an active participant. Your write-up must be your own. Any violations will be taken very seriously.

Code Submission

All the code must be implemented using Python, sklearn, numpy, matplotlib, pandas. If you are not familiar with some of these, there are several online tutorials available. Please consult those before reaching out to course staff for guidance. **All the code files (.py or .ipynb) should be uploaded to canvas along with your writeup. We will not accept any submissions without the complete code**. Please include detailed comments in your code.

Notes for Implementation

Unless we ask you to code something explicitly, please feel free to use built-in functions. If and when you are using built-in functions, please stick to our suggestions (if any) as much as possible (since results may vary with different implementations).

Please use one-hot encoding for categorical variables. The datasets you will be working with may have missing values. Python, sklearn, and related libraries have some capability to handle missing values. However, if you are running into errors and need to handle missing values explicitly, do the following: a) In case of continuous variables, impute using the mean of the corresponding attribute, and also create a new variable for the corresponding attribute – set it to 1 when the attribute value is missing for a particular datapoint and a 0 otherwise. b) In case of categorical variables, treat missing values as a category in itself and encode it as part of the one-hot encoding.

[1] discussed ideas vs. team coding

CS282BR - Homework 1

Due: September 30, 2019, 11.59pm ET

Problem 1: Supervised Learning [60 points]

The goal of this problem is to enable you to: a) develop a better understanding of the decision boundaries of various classification models, b) deeply think about which of the classification models are better suited for different real world scenarios requiring interpretability, and c) understand the trade-offs involved when we try to approximate (explain) a certain class of models (e.g., neural networks) using another class of models (e.g., decision trees). All these concepts are extremely useful through out the course and beyond.

1.1 Implementing Classifiers [20 points]

a. [2 points] Why is cross-entropy preferred as a loss function for classification over mean-squared error?

b. [2.5 points] Write down the cross-entropy loss function for logistic regression.

c. [2.5 points] What is the derivative of the aforementioned loss function w.r.t. the weight parameters?

d. [13 points] Using the computed derivatives and loss function, implement gradient descent algorithm for learning a logistic regression classifier. (Implement gradient descent from scratch. Do not use built-in functions.)

Let us now apply the above logistic regression classifier to a Census Income dataset where the goal is to predict whether income exceeds $50K/yr based on census data. Note that the data contains both categorical as well as continuous attributes.

Train the logistic regression model you developed above using the dataset `adult-train.csv`[2].

To evaluate the model[3], compute precision, recall, F-score, and AUC of the ROC curve using the dataset `adult-test.csv`. Also, include the plot of the ROC curve.

1.2 Understanding Decision Boundaries [20 points]

We will now visualize the decision boundaries of the following classifiers – Logistic Regression, Decision Trees, SVMs, and Neural Networks. For this subproblem, please use sklearn implementations of all the classifiers [4]. More specifically, use the following functions and

[2] Dataset description: https://canvas.harvard.edu/files/8482700/download?download_frd=1

[3] Income $>$ 50K is the positive class (1), and Income $\leq$ 50K is the negative class (0)

[4] You may also choose to use your own implementation that you developed in 1.1 for logistic regression

parameter settings:

sklearn.linear_model.LogisticRegression()
sklearn.tree.DecisionTreeClassifier()
sklearn.svm.SVC(kernel='rbf')
sklearn.svm.SVC(kernel='linear')
sklearn.neural_network.MLPClassifier(hidden_layer_sizes=(2, 2, 2, 2))

Note that the multi-layer perceptron (MLP) classifier is initialized to comprise of 4 hidden layers with 2 nodes per layer. We will also be experimenting with 2 different kernels (RBF and Linear) for Support Vector classification.

a. [15 points] Train each of the aforementioned classifiers (the exact functions are given above) using the dataset `boundaries.csv`.

Plot the decision boundaries for each of the aforementioned classifiers i.e., make a 2-d plot of the data, assign a color to each class label and highlight each data point using the color corresponding to its ground truth class label, and finally mark the decision boundaries of the classifiers.

To illustrate, we already made a plot of the decision boundaries for the logistic regression classifier discussed above (Figure 1). Please use the same format and plot the decision boundaries for all the other classifiers.

b. [5 points] What are the shapes of the decision boundaries of different classifiers? Do all of them have the same shape? If not, why are the shapes of the decision boundaries different?

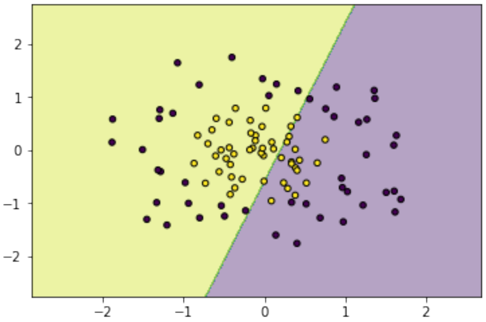

Figure 1: Decision Boundaries for Logistic Regression.

1.3 When to Use What? [5 points]

We have been discussing in class that use cases or applications determine what kind of interpretability is required. Below, we provide two real world scenarios and their corresponding interpretability needs, can you determine which one (or more) of the aforementioned classifiers (Section 1.2) might suit each scenario best?

a. [2.5 points] Let us consider the scenario where we have a classification model which predicts if a patient has rheumatoid arthritis or not. We already know that any of the classifiers discussed in Section 1.2 can make such predictions. But, let us say that this model will be used by a doctor who insists on knowing the relative importance assigned to each of the features by the model. Which classification model(s) would you recommend in this case? Why?

b. [2.5 points] Let us consider another scenario where we have a classification model which predicts if a defendant will commit a crime (or not) if released on bail. We already know that any of the classifiers discussed in Section 1.2 can make such predictions. But, let us say that this model will be used by a judge who insists on seeing the model and tracing each step of its predictive logic for each new defendant (data point) i.e., the judge wants to take the model and (every) new data point and walk through all the steps (of model logic) himself to determine what would be the model's prediction. Which classification model(s) would you recommend in this case? Why?

1.4 What If We Approximate? [15 points]

We discussed in class that it has become common in recent years to approximate (explain) complicated models using simpler ones. Such explanations are called post-hoc explanations. However, such approximations need to be approached with a lot of caution. In this subproblem, let us explore some of the downsides of such post-hoc explanations.

Jack is a machine learning engineer who is building classifiers to predict if a patient has lung cancer (1) or not (0). The dataset he is using is something you already experimented with – `boundaries.csv`. The end goal of his endeavor is to help his client, Dr. Ritter, to make better decisions. Jack has built a multi-layer perceptron (exactly the same model that you built in Section 1.2 with 4 hidden layers and 2 nodes per layer) to make predictions. But, Jack is now concerned that Dr. Ritter might not use his model because it is not readily understandable. To solve this problem, he decided to approximate (explain) the multi-layer perceptron model using decision trees and logistic regression. Jack came to you for advice since he got to know that you are taking CS282BR. You can help him by doing the following:

a. [5 points] Treat the predictions of the multi-layer perceptron classifier you built (Section 1.2) as the ground truth labels and fit a logistic regression model. Note that the input data (X) is the same as in Section 1.2, but the labels have changed.

Plot the decision boundaries of the logistic regression model w.r.t. the new labels in the same format as Figure 1. Include this plot in your writeup. How accurately is the logistic regression model able to approximate the (predictions of) multi-layer perceptron classifier?

b. [5 points] Repeat step a above with decision tree classifier (instead of logistic regression) i.e., treat the predictions of the multi-layer perceptron classifier you built (Section 1.2) as the ground truth labels and fit a decision tree. Plot the decision boundaries of the decision tree w.r.t. the new labels (predictions of multi-layer perceptron) in the same format as Figure 1. Include this plot in your writeup. How accurately is the decision tree able to approximate the (predictions of) multi-layer perceptron classifier?

c. [3 points] Do these post-hoc approximations that you built (logistic regression, decision tree) precisely elucidate how the multi-layer perceptron works? Is one of the approximations (logistic vs. tree) better than the other? If so, which one and why?

d. [2 points] What words of caution would you offer James about such post-hoc approximations (explanations)?

Problem 2: Unsupervised Learning [20 points]

The goal of this problem is to get an intuitive sense of how different clustering algorithms work. This will be particularly useful for thinking about exemplar-based interpretable machine learning algorithms.

a. [5 points] **Apply K-Means to mouse dataset.** In this part of the problem, you will apply the K-Means clustering algorithm to a synthetic mouse dataset downloaded from here: `mouse.csv`. You should use the Scikit-learn implementation of K-Means clustering with default parameters and $K = 3$ clusters. Please plot the data colored by K-Means cluster assignments, using the same format as the example plot in Figure 2.

b. [5 points] **Derive update equations for EM for GMM.** In parts b and c of this problem, you will be deriving and implementing the Expectation Maximization algorithm (EM) for a mixture of Gaussians (GMM). There are 4 key sets of parameters that you will need to update in your implementation: the means of each cluster, the covariances of each cluster, the proportion of data in each cluster, and the cluster assignment probabilities of each datapoint. Please write down the log-likelihood for the GMM model and derive these update equations.

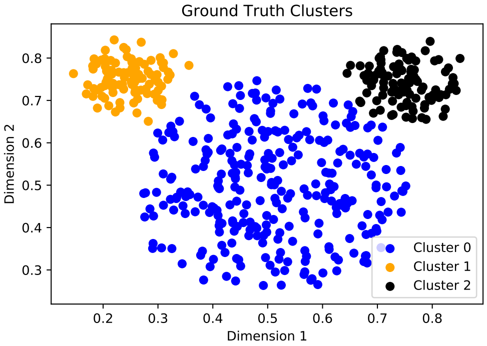

Figure 2: Example plot with ground truth clusters.

c. [5 points] **Implement EM for GMM.** Implement the EM algorithm for GMM that you derived the updates for above. Once you've done so, please apply it to the same dataset used in the first part of this problem with $K = 3$ clusters. When making cluster assignments, assign each datapoint to the cluster with the highest posterior probability. Please plot the data colored by the GMM cluster assignments (just as you did in part a of this problem).

d. [5 points] **Compare and contrast K-Means and EM for GMM.** Clustering with K-Means and EM for Mixtures of Gaussians are closely related, but there are also important differences between them. What differences do you notice between the solutions generated by the 2 algorithms? Which one performs better? Explain the discrepancy.

Problem 3: Overfitting and Cross-Validation [20 points]

The goal of this problem is to help you understand: a) the notions of underfitting and overfitting, and b) how to guard against them when building machine learning models for the real world.

3.1 Detecting Underfitting and Overfitting [5 points]

Consider a synthetic dataset which is generated from the function $sin(2\pi x)$. The true data distribution ($sin(2\pi x)$) is shown by the green curve in Figure 3[5]. The observations generated from this distribution are marked by blue circles in Figure 3.

[5] Note that we do not actually observe the true data distribution in the real world.

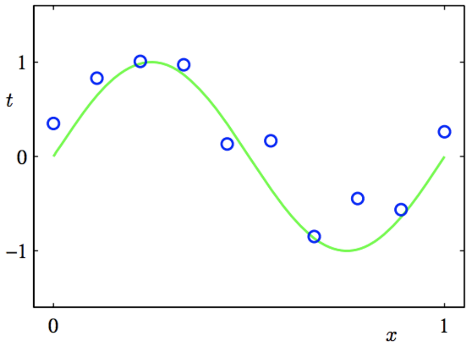

Figure 3: True Underlying Data Distribution (green curve) and Observations (blue circles).

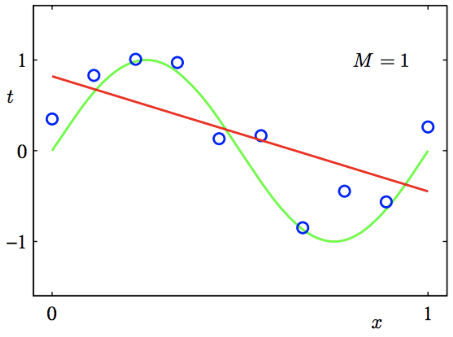

Figure 4: Linear regression model fitted to the observations (red line).

Our ultimate goal would be to find a model (or function) that fits the observations well. But, we should be careful that the chosen model does not underfit or overfit. Let us assume that we decided to fit a polynomial function of the following form:

$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M \tag{1}$$

a. [2.5 points] Figure 4 shows what happens when we fit a linear model (M = 1 in Equation 1) to the observations. Is this model a good fit? If not, is it underfitting or overfitting?

b. [2.5 points] Figure 5 shows what happens when we fit a higher order polynomial (degree = M = 9 in Equation 1) to the observations. Is this model a good fit? If not, is it underfitting or overfitting?

3.2 Cross Validation in Action! [15 points]

Most machine learning and data science classes offer *cross validation* as a solution for overfitting. In this subproblem, we aim to better understand what is cross validation.

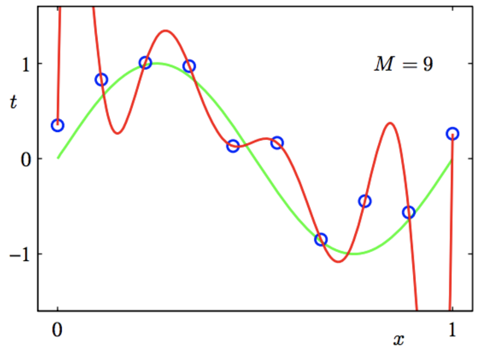

Figure 5: Polynomial of degree 9 fitted to the observations.

a. [5 points] How does cross validation address the problem of overfitting? Does it only identify (or detect) overfitting? Does it also eliminate (or at least reduce) overfitting? Explain your answers.

b. [10 points] The best way to learn something is to do it yourself (aka code it!).
Download the `titanic.csv` dataset[6] and carry out a 10-fold cross validation[7] with each of the following models and parameter settings:

sklearn.linear_model.LogisticRegression()
sklearn.tree.DecisionTreeClassifier()
sklearn.svm.SVC(kernel='rbf')
sklearn.svm.SVC(kernel='linear')
sklearn.neural_network.MLPClassifier(hidden_layer_sizes=(2, 2, 2, 2))

Report the mean and standard deviation of the accuracy for each of the above classifiers. Based on the mean accuracy, which of the above classifiers is the best and which one is the worst?

[6] Dataset description: https://canvas.harvard.edu/files/8482704/download?download_frd=1

[7] Use the built-in function sklearn.cross_validation to carry out 10 fold cross validation