Assignment 3

Foundations of Machine Learning
IIT-Hyderabad
Aug-Nov 2024

Max Marks: 30
Due: 29th Oct 2024 11:59 pm

This homework is intended to cover theory and programming exercises in the following topics:

- Neural Networks, Ensemble Models, Regularization, Linear Regression

Instructions

- Please upload your submission on Google Classroom by the deadline mentioned above. Your submission should comprise of a single file (PDF/ZIP), named `<Your_Roll_No>_Assign3`, with all your solutions.
- For late submissions, 10% is deducted for each day (including weekend) late after an assignment is due. Note that each student begins the course with 7 grace days for late submission of assignments (of which atmost 4 can be used for a given submission). Late submissions will automatically use your grace days balance, if you have any left. You can see your balance on the FoML Marks and Grace Days document.
- Please use PYTHON for the programming questions.
- Answers for theoretical questions must be typed out in LaTeX.
- Please read the department plagiarism policy. Do not engage in any form of cheating - strict penalties will be imposed for both givers and takers. **We will use tools to check for plagiarism between yourselves or LLM sources, as well as conduct vivas for veracity of your submissions.** Please talk to instructor or TA if you have concerns.

Questions: Theory

1. **Non-Uniform Weights in Linear Regression:** ***(3 marks)*** You are given a dataset in which the data points are denoted by $(\mathbf{x}_n, y_n), n = 1, \cdots, N$. Each data point is associated with a non-negative weighting factor $\sigma_n > 0$. The error function is thus modified to:

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \sigma_n \left(y_n - \mathbf{w}^T \Phi(\mathbf{x}_n) \right)^2$$

where $\Phi(\cdot)$ is any representation of the data. Find an expression for the solution $\mathbf{w}^*$ that minimizes the above error function.

2. **Neural Networks:** ***(4 marks)*** The extension of the cross-entropy error function for a multi-class classification problem is given by:

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \ln y_k(\mathbf{x}_n, \mathbf{w})$$

where K is the number of classes, N is the number of data samples, and $\mathbf{t}_n$ is a one-hot vector which designates the expected output for a data sample $\mathbf{x}_n$ (note that a one-hot vector has a 1 in the correct class' position and zeroes elsewhere, e.g. $\mathbf{t}_n = [0, 0, 1, 0, \cdots, 0]$ for the ground truth of the 3rd class). The network outputs $y_k(\mathbf{x}_n, \mathbf{w}) = p(t_k = 1|\mathbf{x})$ are given by the softmax activation function:

$$y_k(\mathbf{x}, \mathbf{w}) = \frac{\exp(a_k(\mathbf{x}, \mathbf{w}))}{\sum_j \exp(a_k(\mathbf{x}, \mathbf{w}))}$$

which satisfies $0 \leq y_k \leq 1$ and $\sum_k y_k = 1$, where 'a_k's are the pre-softmax activations of the output layer neurons (also called logits). For a given input, show that the derivative of the above error function with respect to the activation a_k for an output unit having a logistic sigmoid activation function is given by:

$$\frac{\partial E}{\partial a_k} = y_k - t_k$$

3. **Ensemble Methods:** ***(2+2=4 marks)*** Consider a convex function $f(x) = x^2$. Show that the average expected sum-of-squares error $E_{AV} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_x[(y_m(x) - f(x))^2]$ of the members of an ensemble model and the expected error $E_{ENS} = \mathbb{E}_x[(\frac{1}{M}\sum_{m=1}^{M} y_m(x) - f(x))^2]$ of the ensemble satisfy:

$$E_{ENS} \leq E_{AV}$$

Show further that the above result holds for any error function $E(y)$, not just sum-of-squares, as long as it is convex in y. *(Hint: The only tool you may need is the Jensen's inequality. Read up about it, and use it!)*

4. **Regularizer:** ***(4 marks)*** Given D-dimensional data $\mathbf{x} = [x_1, x_2, \cdots, x_D]$, consider a linear model of the form:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{k=1}^{D} w_k x_k$$

Now, for N such data samples with their corresponding labels $(\mathbf{x}_i, t_i), i = 1, 2, \cdots, N$, the sum-of-squares error (or mean-squared-error) function is given by:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\Big(y(\mathbf{x}_i, \mathbf{w}) - t_i\Big)^2$$

Now, suppose that Gaussian noise $\epsilon_k \sim \mathcal{N}(0, \sigma^2)$ (i.e. zero mean and variance σ^2) is added independently to each of the input variables x_k. Find a relation between: minimizing the above sum-of-squares error averaged over the noisy data, and minimizing the standard sum-of-squares error (averaged over noise-free input data) with a $\mathcal{L}_2$ weight-decay regularization term, in which the bias parameter w_0 is omitted from the regularizer.

Questions: Programming

5. **Random Forests:** ***(5 + 2 + 2 = 9 marks)***

   (a) Write your own random forest classifier (this should be relatively easy, given you have written your own decision tree code) to apply to the *Spam* dataset [`data`, `information`]. Use 30% of the provided data as test data and the remaining for training. Compare your results in terms of accuracy and time taken with Scikitlearn's built-in random forest classifier. (Note that you can't use in-built decision tree functions to implement your code. You can modify your decision tree code of the Assignment 1, or code a new one, to implement a random forest. You can however use the inbuilt train test split of `sklearn` to divide the data into train and test.)

   (b) Explore the sensitivity of Random Forests to the parameter m (the number of features used for best split).

   (c) Plot the OOB (out-of-bag) error (you have to find what this is, and read about it!) and the test error against a suitably chosen range of values for m. (Use your implementation of random forest to perform this analysis.)

6. **Gradient Boosting:** ***(2 + 4 = 6 marks)*** In this question, we will explore the use of pre-processing methods and Gradient Boosting on the popular Lending Club dataset. You are provided with two files: `loan_train.csv` and `loan_test.csv`. The dataset is almost as provided by the the original source, and you may have to make the necessary changes to make it suitable for applying ML algorithms. (If required, you can further divide `loan_train.csv` into a validation set for model selection.) Your efforts will be to pre-process the data appropriately, and then apply gradient boosting to classify whether a customer should be given a loan or not. The target attribute is in the column `loan_status`, which has values "Fully Paid" for which you can assign +1 to, and "Charged off" for which you can assign -1 to. The other records with `loan_status` values "Current" (in both train and test) are not relevant to this problem. You can see this link and this link to know more about the different attributes on the dataset (but please use the provided data, there are several versions of the dataset online.)
   Your tasks are to do the following:

   (a) Pre-process the data as needed to apply the classifier to the training data (you are free to use `pandas` or other relevant libraries. Note that test data should not be used for pre-processing in any way, but the same pre-processing steps can be used on test data. Some steps to consider:

   - Check for missing values, and how you want to handle them (you can delete the records, or replace the missing value with mean/median of the attribute - this is a decision you must make. Please document your decisions/choices in the final submitted report.)
   - Check whether you really need all the provided attributes, and choose the necessary attributes. (You can employ feature selection methods, if you are familiar; if not, you can eyeball.)
   - Transform categorical data into binary features, and any other relevant columns to suitable datatypes
   - Any other steps that help you perform better

(b) Apply gradient boosting using the function `sklearn.ensemble.GradientBoostingClassifier` for training the model. You will need to import `sklearn`, `sklearn.ensemble`, and `numpy`. Your effort will be focused on predicting whether or not a loan is likely to default.

- Get the best test accuracy you can, and show what hyperparameters led to this accuracy. Report the precision and recall for each of the models that you built.
- In particular, study the effect of increasing the number of trees in the classifier.
- Compare your final best performance (accuracy, precision, recall) against a simple decision tree built using information gain. (You can use `sklearn`'s inbuilt decision tree function for this.)

Submission Guidelines:

- Submit your code in a Jupyter notebook (or similar format), with explanations and results clearly documented.
- Include a PDF report (Typed in LATeX) for the theoretical questions and comparative analysis.
- Ensure your code is well-commented and that you have clearly indicated any external libraries used.