HOMEWORK SET 4

CSCI5525 Machine Learning: Analysis and Methods (Fall 2024)

Due 11:59 pm, Dec 11 2024

Instruction Your writeup, either typeset or scanned, should be a single PDF file. For problems requiring coding, organize all codes for all problems into **ONE** Jupyter notebook file (i.e., `.ipynb` file) with cell execution outputs. Your submission to Gradescope should include the single PDF and the one notebook file—please DO NOT zip them! Please assign page(s) to each question to help reduce the TA's navigation time. If your notebook submission does not display properly on Gradescope due to a large file error, please try to remove images and/or figures from your cell outputs and re-upload it. No late submission will be accepted. For each problem, you should acknowledge your collaborators—including AI tools, if any.

About the use of AI tools You are strongly encouraged to use AI tools—they are becoming our workspace friends, such as ChatGPT (https://chat.openai.com/), Claude (https://claude.ai/chats), and Github Copilot (https://github.com/features/copilot), to help you when trying to solve problems. It takes a bit of practice to ask the right and effective questions/prompts to these tools; we highly recommend that you go through this popular free short course **ChatGPT Prompt Engineering for Developers** offered by https://learn.deeplearning.ai/ to get started.

If you use any AI tools for any of the problems, you should include screenshots of your prompting questions and their answers in your writeup. The answers provided by such AI tools often contain factual errors and reasoning gaps. **So, if you only submit an AI answer with such bugs for any problem, you will obtain a zero score for that problem**. You obtain the scores only when you explain the bugs and also correct them in your own writing. You can also choose not to use any of these AI tools, in which case we will grade based on the efforts you have made.

Reminder about notations We will use small letters (e.g., u) for scalars, small boldface letters (e.g., $\boldsymbol{a}$) for vectors, and capital boldface letters (e.g., $\boldsymbol{A}$) for matrices. For a matrix $\boldsymbol{A}$, $\boldsymbol{a}^i$ (supscripting) means its i-th row as a *row vector*, and $\boldsymbol{a}_j$ (subscripting) means the j-the column as a column vector, and a_{ij} means its (i,j)-th element. $\mathbb{R}$ is the set of real numbers. $\mathbb{R}^n$ is the space of n-dimensional real vectors, and similarly $\mathbb{R}^{m\times n}$ is the space of $m \times n$ real matrices. The dotted equal sign $\doteq$ means defining.

Problem 1 (AdaBoost with decision stumps; 5.5/15)

(a) Implement AdaBoost to perform binary classification on the digits dataset (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html). Treat digits 0–4 as the first class and 5–9 as the second. For the weak learner, let us use the ERM rule with decision stumps as the hypothesis class. Finetune and optimize your parameters (e.g., T) and benchmark the performance of your implementation against sklearn's `AdaBoostClassifier`. (1.5/15)

(b) Let $\mathcal{H}^d_{DS}$ be the set of decision stumps in $\mathbb{R}^d$, and consider

$$\mathcal{H}(\mathcal{H}^d_{DS}, T) \doteq \left\{ \boldsymbol{x} \mapsto \text{sign}\left(\sum\nolimits_{t\in[T]} \alpha_t h_t(\boldsymbol{x}) \right) : \boldsymbol{\alpha} \in \mathbb{R}^T, h_t \in \mathcal{H}^d_{DS}\ \forall t \right\}. \tag{1}$$

Let $d_{DS} \doteq \mathsf{VCdim}(\mathcal{H}^d_{DS})$. Prove that

$$d_{T\text{-}DS} \doteq \mathsf{VCdim}\left(\mathcal{H}(\mathcal{H}^d_{DS}, T) \right) \le O\left(T d_{DS} \log\left(T d_{DS} \right) \right), \tag{2}$$

following the steps below. **Throughout this question, we assume** $d_{DS}, T \geq 3$.

(i) Show that $d_{T\text{-}DS} \geq d_{DS}$. $(0.5/15)$

(ii) Show that $d_{T\text{-}DS} \geq T$, hence in combination with (i) to conclude that $d_{T\text{-}DS} \geq \max(T, d_{DS})$. (**Hint: We show in class and in the supplementary notes that** $\mathcal{H}(\mathcal{H}_{DS}^1, T)$ **can represent any piecewise constant** $+1, -1$ **functions with at most** T **pieces.**) $(0.5/15)$

(iii) Show that

$$\Pi_{\mathcal{H}_{T\text{-}DS}}(d_{T\text{-}DS}) \leq d_{T\text{-}DS}^{(d_{DS}+1)T}, \tag{3}$$

where Π denotes the growth function. (**Hint: Sauer's lemma may be useful here.**) $(1/15)$

(iv) Construct a tightest possible lower bound for $\Pi_{\mathcal{H}_{T\text{-}DS}}(d_{T\text{-}DS})$, and combine the lower and upper bounds to establish the claimed result in Eq. (2). $(1/15)$

(c) Assume that our assumption about the weak learner holds: the training error $\varepsilon < 1/2$ for any weights $\boldsymbol{w}$ on the training set. Prove that in AdaBoost, h_{t+1} selected in round $t+1$ must be different from h_t selected in round t. (**Hint: in round** $t+1$**, consider the training error if we were to reuse** h_t **as the base predictor**.) $(1/15)$

Problem 2 (General boosting; $7/15$)

(a) (i) $g(z) = \log(1 + e^{-z})$ is called the *logistic loss* ($\log$ here is the natural log). Is it convex or not? Why or why not? $(0.5/15)$

(ii) Prove that $g(z) \leq e^{-z}$, i.e., logistic loss is a uniform lower bound of the exponential loss. $(0.5/15)$

(iii) Consider the loss $\ell(y, f(\boldsymbol{x})) = \log\left(1 + e^{-yf(\boldsymbol{x})}\right)$ for binary classification, and derive the rule for finding h_t's based on gradient boosting—phrase it as weighted error minimization as in AdaBoost. $(1/15)$

Compare the weights (on training points) used here with those of the original AdaBoost, what do you observe? Any potential advantages? $(0.5/15)$

Implement this version of boosting with decision stumps as the hypothesis class, probably try different step sizes to find the best performance, and compare it with Prob 1(a) on digit classification. $(1/15)$

Why is this interesting? This particular loss is called the *deviance*, or logistic loss. This is the default loss choice for the sklearn implementation of gradient boosting (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html). To see why, note that when e^{-z} is small, $\log(1 + e^{-z}) = e^{-z} + o(e^{-z})$, it is very close to the exponential loss e^{-z}. However, when $-z$ is large, e^{-z} grows exponentially, whereas $\log(1 + e^{-z}) \approx -z$ only grows linearly. So the exponential loss penalizes large errors harshly (i.e., when $-yh(\boldsymbol{x})$ gets large), but logistic loss does so much more mildly. This can be beneficial in terms of robustness when there are stubborn data points, e.g., outliers.

(b) Now let's switch to boosted regression. Classification and regression trees (CARTs) are popular base hypothesis classes for both boosted classification and regression in practice. We described in class that CART partitions the space hierarchically using decision stumps, and finally assigns a constant value (label) for each cell based on averaging (or majority voting). Sklearn implementation of CART for regression can be found here (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html)

To implement boosted CART based on gradient boosting, we need to solve

$$\arg\min_{h\in\mathcal{H}} \sum_{i=1}^{N} \left.\frac{\partial \ell(y_i, z)}{\partial z}\right|_{z=f^{(t-1)}(\boldsymbol{x}_i)} h\left(\boldsymbol{x}_i\right). \tag{4}$$

This turns out to be relatively simple for classification but nontrivial for regression. For regression, it is common to solve a proxy least squares problem:

$$\arg\min_{h\in\mathcal{H}} \sum_{i=1}^{N} \left(-\left.\frac{\partial \ell(y_i, z)}{\partial z}\right|_{z=f^{(t-1)}(\boldsymbol{x}_i)} - h\left(\boldsymbol{x}_i\right)\right)^2. \tag{5}$$

One may be tempted to add an optimizable scaling factor c in front of $h\left(\boldsymbol{x}_i\right)$. But note that $ch \in \mathcal{H}$ if $h \in \mathcal{H}$ when $\mathcal{H}$ is CART, and so the scaling factor is unnecessary.

Derive gradient boosting rule for $\ell\left(f\left(\boldsymbol{x}\right), y\right) = \left(f\left(\boldsymbol{x}\right) - y\right)^2$ with CART (CART can be used as an off-the-shelf module here). (0.5/15)

Implement your boosted CART regression algorithm, and test it on the California house price dataset (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html). Compare the performance with that of sklearn built-in (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html). (1/15)

(c) Here, we study how to extend AdaBoost for binary classification to multiclass classification. For this purpose, we assume $\mathcal{Y} = \{1, \ldots, K\}$, i.e., there are K classes, and represent each label as a vector $\boldsymbol{y} \in \mathbb{R}^K$, so that

$$\boldsymbol{x} \text{ in class } k \iff y_j = \begin{cases} 1 & j = k \\ -\frac{1}{K-1} & j \neq k \end{cases}. \tag{6}$$

In other words, for class k, the corresponding $\boldsymbol{y}$ is all $-\frac{1}{K-1}$ except for the k-th coordinate, which takes 1. So, we write the training set as $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i\in[N]}$—note that all the labels $\boldsymbol{y}_i$'s are vectors in $\mathbb{R}^K$. Moreover, we consider a base hypothesis class $\mathcal{H}$ that consists of functions of the form $\mathcal{X} \to \mathbb{R}^K$, where the output is always a valid label vector.

(i) For any predictor $f : \mathcal{X} \to \mathbb{R}^K$, consider the multiclass exponential loss $\ell(\boldsymbol{y}, f(\boldsymbol{x})) \doteq \exp(-\frac{1}{K}\boldsymbol{y}^\intercal f(\boldsymbol{x}))$. Implement the greedy method to solve the problem

$$\min_{\forall t\in[T]:\alpha_t, h_t\in\mathcal{H}} \frac{1}{N}\sum_{i\in[N]} \ell(\boldsymbol{y}_i, \sum_{t\in[T]} \alpha_t h_t(\boldsymbol{x}_i)). \tag{7}$$

Try to make your analytical expressions as explicit as possible. (1/15)

(ii) What will be a reasonable final prediction rule that outputs a single label? Justify your answer. Also, emulate AdaBoost in our supplementary notes to write down detailed steps of the whole algorithm. (1/15)

Problem 3 (Random forests; $2.5/15$)

(a) Implement bagging with CART as the base regressor, and test it on the California house price dataset (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html). $(1/15)$

(b) Implement random forests with CART as the regression tress. Note that each time, d' from d features are randomly selected each time. Carefully read the documentation of CART implemetation (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html) and figure out how to do this. Test your implementation on the California house price dataset also. $(1/15)$

(c) Compare the performance of your implementation of boosted CART, bagged CART, and random forest based on CART, and plot the test error vs the number of CARTs used—produce a figure similar of Fig 15.1 of [Has09], but you only need to try 10 different numbers for the horizontal axis. $(0.5/15)$

References

[Has09] Trevor Hastie, *The elements of statistical learning: data mining, inference, and prediction*, 2009.