HOMEWORK SET 5

CSCI 5525 Advanced Machine Learning (Spring 2021)

Due 11:59 pm, May 14 2021

Instruction Typesetting your homework in LaTeX is optional but encouraged, and you need to submit it as a single PDF file in Canvas. For programming, include all your codes and running results in a single Jupyter notebook file and submit it alongside the main PDF (since Jupyter notebook also allows text editing, feel free to put your textual answers inside the Jupyter notebook sometimes). No late submission will be accepted.

For each problem, your should acknowledge your collaborators if any. For problems containing multiple subproblems, there are often close logic connections between the subproblems. So always remember to build on previous ones, rather than work from scratch.

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 (Boosting with decision stumps; 3.5/12) The standard Adaboost algorithm for binary classification runs as follows.

Algorithm 1 Adaboost

Input: training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, weak learner (WL), initial weights $\boldsymbol{w}^{(0)} = \frac{1}{N}\mathbf{1} \in \mathbb{R}^N$, total rounds T

1: **for** $t = 1, \ldots, T$ **do**

2: invoke weak learner: $h_t = \text{WL}\left(\boldsymbol{w}^{(t-1)}, S\right)$

3: compute weighted error: $\varepsilon_t = \sum_{i=1}^N w_i^{(t-1)} \mathbb{1}\left\{y_i \neq h_t(\boldsymbol{x}_i)\right\}$

4: compute updating factor: $\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

5: update weights: $w_i^{(t)} = \frac{w_i^{(t-1)}\exp(-y_i\alpha_t h_t(\boldsymbol{x}_i))}{\sum_{i=1}^N w_i^{(t-1)}\exp(-y_i\alpha_t h_t(\boldsymbol{x}_i))}\ \forall\ i$

6: **end for**

Output: the final classifier $h(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(\boldsymbol{x})\right)$

Here, $\mathbf{1}$ is the all-one vector. The weak learner should perform better than random guessing and also be cheap to compute. Decision stumps and decision trees are popular hypothesis classes used in the weak learner. Decision stumps (i.e., decision trees with depth 1) over $\mathbb{R}^d$ are functions of the form

$$\mathcal{H}_{ds} \doteq \{\boldsymbol{x} \mapsto b\,\text{sign}(x_i - \theta) : \theta \in \mathbb{R}, i \in [d], b \in \{\pm 1\}\}, \tag{1}$$

i.e., they are hyperplanes orthogonal to the standard basis vectors $\{\boldsymbol{e}_j\}_{j=1}^d$. Sec. 10.1.1 of [SSS14] discusses how to efficiently perform empirical risk minimization (ERM) with decision stumps, which we also covered in class.

(a) Let's use ERM with decision stumps as the weak learner. Implement Adaboost binary classification on the digits dataset (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits)—treat digits 0–4 as the first class and 5–9 as the second, and compare the classification accuracy with sklearn built-in AdaBoostClassifier (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html). (1/12)

(b) We illustrated in class that linear combination of decision stumps can lead to very powerful new hypothesis classes. Consider

$$\mathcal{H}_{T\text{-}ds} \doteq \left\{ \boldsymbol{x} \mapsto \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t\left(\boldsymbol{x}\right) \right) : \boldsymbol{\alpha} \in \mathbb{R}^T, h_t \in \mathcal{H}_{ds}\ \forall t \right\}. \tag{2}$$

Let d_{ds} be the VC dimension of decision stumps in $\mathbb{R}^d$. Prove that

$$d_{T\text{-}ds} \doteq \dim_{VC}\left(\mathcal{H}_{T\text{-}ds}\right) \leq O\left(T d_{ds} \log\left(T d_{ds}\right)\right) \tag{3}$$

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

(i) Show that $d_{T\text{-}ds} \geq \max\left(T, d_{ds}\right)$. (1/12)

(ii) Show that

$$\Pi_{\mathcal{H}_{T\text{-}ds}}\left(d_{T\text{-}ds}\right) \leq d_{T\text{-}ds}^{(d_{ds}+1)T}, \tag{4}$$

where Π denotes the growth function. (Hint: Sauer's lemma and its upper bounds, i.e., Theorem 3.17 and Corollary 3.18 of [Moh18], are useful here; 1/12)

(iii) Construct a tightest possible lower bound for $\Pi_{\mathcal{H}_{T\text{-}ds}}\left(d_{T\text{-}ds}\right)$, and combine the lower and upper bounds to establish the claimed result in Eq. (3). (0.5/12)

Problem 2 (Generalizations of Adaboost; 6/12) Given a training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ and a base hypothesis class $\mathcal{H}$, boosting can be understood as trying to construct a combined predictor $f = \sum_{t=1}^T \alpha_t h_t$ where $h_t \in \mathcal{H}\ \forall\ t$, by minimizing the empirical training error

$$\sum_{i=1}^{N} \ell\left(f\left(\boldsymbol{x}_i\right), y_i\right) \tag{5}$$

using a greedy method. In each iteration, given a partial predictor $f^{(t-1)}$ that is already constructed, boosting tries to solve

$$(\alpha_t, h_t) = \underset{\alpha,\, h \in \mathcal{H}}{\arg\min} \sum_{i=1}^{N} \ell\left(f^{(t-1)}\left(\boldsymbol{x}_i\right) + \alpha h\left(\boldsymbol{x}_i\right), y_i\right), \tag{6}$$

and then updates the predictor as $f^{(t)} = f^{(t-1)} + \alpha_t h_t$. The canonical Adaboost uses the exponential loss $\ell\left(f\left(\boldsymbol{x}\right), y\right) = \exp\left(-y f\left(\boldsymbol{x}\right)\right)$, and the greedy subproblem is:

$$(\alpha_t, h_t) = \underset{\alpha,\, h \in \mathcal{H}}{\arg\min} \sum_{i=1}^{N} \exp\left(-y_i f^{(t-1)}\left(\boldsymbol{x}_i\right)\right) \exp\left(-\alpha y_i h(\boldsymbol{x}_i)\right) \propto \sum_{i=1}^{N} w_i^{(t)} \exp\left(-\alpha y_i h(\boldsymbol{x}_i)\right). \tag{7}$$

Now

$$\min_{\alpha,\, h\in\mathcal{H}} \sum_{i=1}^{N} w_i^{(t)} \exp\left(-\alpha y_i h(\boldsymbol{x}_i)\right) \equiv \min_{\alpha} \left(\min_{h\in\mathcal{H}} \sum_{i=1}^{N} w_i^{(t)} \exp\left(-\alpha y_i h(\boldsymbol{x}_i)\right) \right). \tag{8}$$

For binary classification problems, $y_i h\left(\boldsymbol{x}_i\right)$ is either 1 or -1 for all i, for any fixed α,

$$\min_{h\in\mathcal{H}} \sum_{i=1}^{N} w_i^{(t)} \exp\left(-\alpha y_i h(\boldsymbol{x}_i)\right) = e^{-\alpha} \sum_{i=1}^{N} w_i^{(t)} \mathbb{1}\left\{y_i = h\left(\boldsymbol{x}_i\right)\right\} + e^{\alpha} \sum_{i=1}^{N} w_i^{(t)} \mathbb{1}\left\{y_i \neq h\left(\boldsymbol{x}_i\right)\right\} \tag{9}$$

$$= \left(e^{\alpha} - e^{-\alpha}\right) \sum_{i=1}^{N} w_i^{(t)} \mathbb{1}\left\{y_i \neq h\left(\boldsymbol{x}_i\right)\right\} + e^{-\alpha}. \tag{10}$$

If $\alpha \geq 0$, the minimization is equivalent to

$$\min_{h\in\mathcal{H}} \sum_{i=1}^{N} w_i^{(t)} \mathbb{1}\left\{y_i \neq h\left(\boldsymbol{x}_i\right)\right\}. \tag{11}$$

For decision stumps and trees, this weighted training error can be easily solved. Then, fix the h_t found, one continues to solve the outer optimization

$$\min_{\alpha} \sum_{i=1}^{N} w_i^{(t)} \exp\left(-\alpha y_i h_t(\boldsymbol{x}_i)\right) = e^{\alpha}\varepsilon_t + e^{-\alpha}\left(1-\varepsilon_t\right).$$

Applying the unconstrained first-order optimality condition to $g\left(\alpha\right) \doteq e^{\alpha}\varepsilon_t + e^{-\alpha}\left(1-\varepsilon_t\right)$ yields $\alpha_* = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$. Since $\varepsilon_t < \frac{1}{2}$ by our assumption on our weak learner, $\alpha_* > 0$. So $\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.

(a) Choose $\ell\left(f\left(\boldsymbol{x}\right), y\right) = \left(f\left(\boldsymbol{x}\right) - y\right)^2$ and derive the resulting boosting algorithm for binary classification problems. You should try to present the update steps in the most explicit forms whenever possible. (Hint: solve for the optimal α first for any given h, and then find the best h; 1/12)

(b) The gradient boosting generalization takes the 1st-order (functional) Taylor expansion of the objective in Eq. (6) around $f^{(t-1)}$ (i.e., think of $\alpha h\left(\boldsymbol{x}_i\right)$ as a small perturbation and assume $\alpha \geq 0$) and tries to optimize the expansion with respect to h:

$$h_t = \arg\min_{h\in\mathcal{H}} \sum_{i=1}^{N} \ell\left(f^{(t-1)}\left(\boldsymbol{x}_i\right), y_i\right) + \alpha \frac{\partial \ell}{\partial f\left(\boldsymbol{x}_i\right)}\left(f^{(t-1)}\left(\boldsymbol{x}_i\right), y_i\right) h\left(\boldsymbol{x}_i\right) \tag{12}$$

$$= \arg\min_{h\in\mathcal{H}} \sum_{i=1}^{N} \frac{\partial \ell}{\partial f\left(\boldsymbol{x}_i\right)}\left(f^{(t-1)}\left(\boldsymbol{x}_i\right), y_i\right) h\left(\boldsymbol{x}_i\right). \tag{13}$$

Afterward, a step α_t is taken:

$$f^{(t)} = f^{(t-1)} + \alpha_t h_t. \tag{14}$$

The step size can be optimized by one-dimensional optimization as above. But for general classification and regression problems, it is often set as a small constant, also called the *shrinkage factor*, to avoid overfitting. Take $\ell\left(f\left(\boldsymbol{x}\right), y\right) = \exp\left(-y f\left(\boldsymbol{x}\right)\right)$ again. We have

$$\frac{\partial \ell}{\partial f\left(\boldsymbol{x}\right)} = -y \exp\left(-y f\left(\boldsymbol{x}\right)\right). \tag{15}$$

So to find the best h we solve

$$\min_{h\in\mathcal{H}} \sum_{i=1}^{N} -y_i \exp\left(-y_i f^{(t-1)}\left(\boldsymbol{x}_i\right)\right) h\left(\boldsymbol{x}_i\right) \propto \sum_{i=1}^{N} w_i^{(t)}(-y_i)h\left(\boldsymbol{x}_i\right) \tag{16}$$

$$\equiv \min_{h\in\mathcal{H}} \sum_{i=1}^{N} w_i^{(t)} \mathbb{1}\left\{y_i \neq h\left(\boldsymbol{x}_i\right)\right\} \tag{17}$$

for binary classification problems. This reproduces the h step in Adaboost. The power of gradient boosting is that it can handle arbitrary differentiable loss ℓ, for both classification and regression problems.

(i) $g\left(z\right) = \log\left(1+e^{-z}\right)$ is called the *logistic loss* ($\log$ here is the natural log). Is it convex or not? Why or why not? (0.5/12)

(ii) Prove that $g\left(z\right) \leq e^{-z}$, i.e., logistic loss is a uniform lower bound of the exponential loss. (0.5/12)

(iii) Consider the loss $\ell\left(f\left(\boldsymbol{x}\right), y\right) = \log\left(1+e^{-yf(\boldsymbol{x})}\right)$ for binary classification, and derive the rule for finding h based on gradient boosting—phrase it as weighted error minimization as in Adaboost. (1/12)

Compare the weights used here with that of the canonical Adaboost, what do you observe? Any potential advantages? (0.5/12)

Implement this version of boosting with the decision stumps, probably try different step sizes to find the best performance, and compare it with Prob 1(a) on digit classification. (1/12)

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, $\log\left(1+e^{-z}\right) = e^{-z} + o(e^{-z})$, it is very close to the exponential loss e^{-z}. However, when $-z$ is large, e^{-z} grows exponentially whereas $\log\left(1+e^{-z}\right) \approx -z$ only grows linearly. So the exponential loss penalizes large errors harshly (i.e., when $-yh\left(\boldsymbol{x}\right)$ 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.

(iv) Now let's switch to boosted regression. Classification and regression trees (CARTs) are popular base hypothesis classes for both 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 (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

$$\arg\min_{h\in\mathcal{H}} \sum_{i=1}^{N} \frac{\partial \ell}{\partial f\left(\boldsymbol{x}_i\right)}\left(f^{(t-1)}\left(\boldsymbol{x}_i\right), y_i\right) h\left(\boldsymbol{x}_i\right). \tag{18}$$

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:

$$\arg\min_{h\in\mathcal{H}} \sum_{i=1}^{N} \left(-\frac{\partial \ell}{\partial f\left(\boldsymbol{x}_i\right)} \left(f^{(t-1)}\left(\boldsymbol{x}_i\right), y_i \right) - h\left(\boldsymbol{x}_i\right) \right)^2. \tag{19}$$

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). (0.5/12)

Implement your boosted CART regression algorithm, and test it on the Boston house price dataset (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html). Compare the performance with that of sklearn built-in (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html). (1/12)

Problem 3 (Random forests; 2.5/12)

(a) Read Sec. 8.7 of [HTF09], implement bagging with CART as the base regressor, and test it on the Boston house price dataset. (1/12)

(b) Read Secs 15.2 & 15.3 of [HTF09] and implement random forests with CART as the regression tress. Note that in Algorithm 15.1, m features out of p are randomly picked 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 Boston house price dataset also. (1/12)

(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 [HTF09], but you only need to try 10 different numbers for the horizontal axis. (0.5/12)

References

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The elements of statistical learning: Data mining, inference, and prediction, second edition*, SPRINGER NATURE, 2009.

[Moh18] Mehryar Mohri, *Foundations of machine learning*, 2 ed., The MIT Press, Cambridge, Massachusetts, 2018.

[SSS14] Shai Ben-David Shai Shalev-Shwartz, *Understanding machine learning*, Cambridge University Press, 2014.