Homework 3

36-708, Spring 2021

Due April 16 at 5PM EST

Please attach all code to your homework. In an RMarkdown document for example, this can be done in one line (see Yihui Xie's website for how to do this).

1 Deriving boosting-like variants for square and logistic losses

- Data: Assume n labeled data points $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R} \times \{-1, 1\}$.
- Classifier: Denote by $f = \sum_{t=1}^{T} \alpha_t h_t$ the linear combination of base classifiers h_t with weights α_t.
- Loss function: Let the loss function to be used in the boosting derivation be $L = \frac{1}{n}\sum_{i=1}^{n} \phi(-y_i f(x_i)$ for some function ϕ (to be specified). The 0-1 loss is $\phi(-u) = \mathbb{1}(u \leq 0)$ but we will use surrogates of this.
- Algorithm: *Boosting*, which we will derive from the point of view of coordinate descent on the loss function L. In particular at each step t of boosting, we find the best base classifier h_t and step size α_t and update the function f with the new component $\alpha_t h_t$.

1.1 AdaBoost as coordinate descent

In this problem, we will use $\phi_a(-u) := \exp(-u)$, the exponential loss. We will rederiving boosting as coordinate descent to recover AdaBoost.

(a) Write the objective function we wish to minimize: $F(\alpha) := \frac{1}{n}\sum_{i=1}^{n} \exp\left(-y_i \sum_{j=1}^{m} h_j(x_i)\alpha_j\right)$. Consider the "coordinate" of steepest descent:

$$\operatorname*{argmin}_k \left.\frac{\partial F(\alpha + \eta e_k)}{\partial \eta}\right|_{\eta=0}.$$

Show that

$$\left.\frac{\partial F(\alpha + \eta e_k)}{\partial \eta}\right|_{\eta=0} = \left(\prod_{j=1}^{t-1} Z_j\right) \sum_{i=1}^{n} -y_i h_k(x_i) w_i$$
$$\propto 2\varepsilon_{t,k} - 1$$

where w_i and $\prod_j Z_j$ are the weights and normalizing factor described in the lecture. Finally, conclude that the argmin_k of the above is the weak learner with smallest weighted 0-1 error.

(b) Show that the step-size η is given by $\frac{1}{2}\log\left(\frac{1-\varepsilon_{t,k}}{\varepsilon_{t,k}}\right)$. That is, show that $\frac{1}{2}\log\left(\frac{1-\varepsilon_{t,k}}{\varepsilon_{t,k}}\right)$ is the solution to

$$\frac{\partial F(\alpha + \eta e_k)}{\partial \eta} = 0.$$

(c) Write the pseudo-code for AdaBoost.

1.2 SquareBoost using the squared loss

We will repeat the previous problem but for the squared loss, $\phi_s(-u) = (1-u)^2 \mathbb{1}(u \leq 1)$.

(a) Consider the objective function:

$$F(\alpha) = \frac{1}{n} \sum_{i=1}^{n} \left(1 - y_i \sum_{j=1}^{m} h_j(x_i)\alpha_j \right)^2 \mathbb{1} \left(y_i \sum_{j=1}^{m} h_j(x_i)\alpha_j \leq 1 \right)$$

Using similar steps to before, show that the minimizer of the above loss is the weak learner with best 0-1 loss on the weighted data with weights given by

$$w_i := \frac{(1 - y_i f(x_i))\mathbb{1}\left(y_i \sum_{j=1}^{m} h_j(x_i)\alpha_j \leq 1\right)}{\sum_{i=1}^{n}(1 - y_i f(x_i))\mathbb{1}\left(y_i \sum_{j=1}^{m} h_j(x_i)\alpha_j \leq 1\right)}$$

(b) Unlike AdaBoost, the step size of SquareBoost cannot be computed in closed-form. How would you go about computing it?

(c) Write the pseudo-code for SquareBoost.

1.3 LogisticBoost using the logistic loss

We will repeat the previous problems but for the logistic loss, $\phi_l(-u) = \log(1 + e^{-u})$.

(a) Show that the minimizer of the logistic loss is given by the weak learner with smallest 0-1 loss on the weighted data where weights are given by

$$w_i = \frac{\text{logit}^{-1}(-y_i f(x_i))}{\sum_{i=1}^{n} \text{logit}^{-1}(-y_i f(x_i))}$$

(b) Show how finding the step-size is equivalent to training a logistic regression model with and offset and no intercept. Explain how you could compute this step size.

(c) Write the pseudo-code for LogisticBoost.

1.4 Comparison of AdaBoost, SquareBoost, and LogisticBoost

(a) How do the three previous boosting algorithms differ?

2 Implementing and evaluating boosting

In this problem, we will implement the previous three boosting algorithms and apply them to the spam dataset.

- Dataset: We will use the spam dataset available here.
- Train and test split: Use a random 3:1 split.

(a) Implement AdaBoost, SquareBoost, and LogisticBoost with decision stumps as the base classifiers.

(b) Run the boosting algorithms for various values of the number of boosting rounds t (for some reasonable upper limit T).

(c) Plot the train and test errors of all three algorithms as a function of t.

(d) Summarize your findings. In particular, comment on the following.

- How fast does the training error decrease?
- Does the training error reach zero? What happens to the testing error if you train longer?

3 Convex combinations and margins

3.1 Rademacher complexity of convex combinations

Recall the definition of empirical Rademacher complexity for a function class $\mathcal{F}$:

$$\mathfrak{R}(\mathcal{F}) = \frac{1}{m}\mathbb{E}_\sigma\left[\sup_{f\in\mathcal{F}}\sum_{i=1}^{m}\sigma_i f(x_i)\right],$$

where $x_1, \ldots, x_m$ is a sample which we treat as fixed, and $\sigma_1, \ldots, \sigma_m$ are iid Rademacher random variables (i.e. taking on values 1 and -1 with equal probability).

Let $\mathcal{H}$ denote a class of functions mapping the feature space to $\{-1, 1\}$. Define the convex hull $\operatorname{conv}(\mathcal{H})$ of $\mathcal{H}$ by

$$\operatorname{conv}(\mathcal{H}) = \left\{\sum_{i=1}^{T}\alpha_t h_t : h_t \in \mathcal{H},\ \alpha_t \geq 0,\ \sum_{t=1}^{T}\alpha_t \leq 1\right\}.$$

(a) Show that $\mathfrak{R}(\operatorname{conv}(\mathcal{H})) = \mathfrak{R}(\mathcal{H})$. That is, the class of convex combinations of learners is no more Rademacher-complex than the class of base learners.

3.2 VC dimension of convex combinations (Bonus)

(a) Given the same setup as the previous problem, suppose that $\mathcal{H}$ has VC-dimension $\mathrm{VC}(\mathcal{H}) \in \{3, \ldots, T\}$. Show that the VC-dimension of $\operatorname{conv}(\mathcal{H})$ is upper bounded as

$$\mathrm{VC}(\operatorname{conv}(\mathcal{H})) \leq O\left(\mathrm{VC}(\mathcal{H}) \cdot T\log(T)\right).$$

Comment on the difference between this and the Rademacher complexity of $\operatorname{conv}(\mathcal{H})$.

Note: Please ensure that all steps of the proof are clear. Partial solutions will not be eligible for bonus credit.

3.3 Perfect classifiers and margins

(a) Recall that a perfect linear classifier is a vector $w \in \mathbb{R}^d$ such that $\operatorname{sign}(w^T x_i) = \operatorname{sign}(y_i)$ for each $i = 1, \ldots, m$. Define $a_i := x_i y_i$ and arrange these d-dimensional vectors into a $d \times m$ matrix A,

$$A = \begin{pmatrix} a_1^T & a_2^T & \cdots & a_m^T \end{pmatrix}.$$

Recall that the margin ρ is given by

$$\rho := \sup_{\|w\|_2=1}\inf_{p\in\Delta} w^T A p.$$

Show that a perfect linear classifier w exists if and only if $\rho > 0$.

4 Shapley values

Consider a simple cost allocation problem. Suppose there are n people that wish to share transportation to get from point A to their respective destinations, which are all in succession on the same street. Suppose that the cost of going from point A to the i^{th} person's destination costs c_i, and without loss of generality suppose $c_1 < c_2 \cdots < c_n$. The 'cost', $\nu : 2^{[n]} \to \mathbb{R}$ of a trip is defined in the following natural way,

$$\begin{aligned}&\nu(\emptyset) = 0\\&\nu(\{i\}) = c_i\\&\nu(S) = c_j \text{ where } j = \max(S)\end{aligned}$$

(a) Define the 'sub-cost' functions:

$$\nu_j(S) = \begin{cases} c_j - c_{j-1} & \text{if } j \leq \max(S) \\ 0 & \text{otherwise,} \end{cases}$$

where $c_0 \equiv 0$ by convention. That is, ν_j is the additional cost for going to the j^{th} stop. It is 0 if nobody needs to go beyond stop j, and otherwise is $c_j - c_{j-1}$. Show that

$$\phi_i(\nu) = \sum_{j=1}^{n} \phi_i(\nu_j)$$

(b) Show that for any $i < j$, person i is a 'null player' with respect to cost ν_j. Further, show that persons $i_1, i_2 \geq j$ are equivalent with respect to cost ν_j.

(c) Use (b) to show that

$$\phi_i(\nu_j) = \begin{cases} \frac{c_j - c_{j-1}}{n-j+1} & \text{if } i \geq j \\ 0 & \text{otherwise.} \end{cases}$$

Finally, conclude that

$$\phi_i(\nu) = \sum_{j=1}^{i} \frac{c_j - c_{j-1}}{n - j + 1}.$$

(d) Interpret the Shapley values. Does such a cost division make sense? If you didn't know about the Shapley values, what would your fair division look like? What are some advantages and disadvantages of your division strategy compared to Shapley's? (If you *would* do the same thing as Shapley, comment on its advantages and disadvantages.)