COMS 4771 FA18 HW3

Due: Mon Nov 19, 2018 at 11:59pm

You are allowed to write up solutions in groups of (at max) three students. These group members don't necessarily have to be the same from previous homeworks. Only one submission per group is required by the due date on Gradescope. Name and UNI of all group members must be clearly specified on the homework. No late homeworks are allowed. To receive credit, a typesetted copy of the homework pdf must be uploaded to Gradescope by the due date. You must show your work to receive full credit. Discussing possible solutions for homework questions is encouraged on piazza and with peers outside your group, but every group must write their own individual solutions. You must cite all external references you used (including the names of individuals you discussed the solutions with) to complete the homework.

1(a) **[Mixing Ridge and Lasso regression]** You want to design a regressor with good prediction accuracy. Knowing that various kinds of regularizations help, you decide to incorporate both the lasso and ridge penalties in the design of your optimization criterion. Given training data $(x_1, y_1), \ldots, (x_n, y_n)$ (where each $x_i \in \mathbb{R}^d$ and each $y_i \in \mathbb{R}$) you come up with the following optimization

$$\arg\min_{w} \ \big\|wX - y\big\|^2 + \lambda\Big[\alpha\|w\|_2^2 + (1-\alpha)\|w\|_1\Big],$$

where: (i) X is a $d \times n$ data matrix where the i^{th} column corresponds to training data x_i, (ii) y is a $1 \times n$ output vector where the i^{th} component corresponds to the training output y_i, and (iii) w is a $1 \times d$ parameter weight vector. $\lambda \geq 0$ and $\alpha \in [0, 1]$ are known trade-off parameters.

Show that this optimization can be turned into a lasso regression by appropriately augmenting the training data.

1(b) **[Bayesian interpretation of ridge regression]** Consider the following data generating process for linear regression problem in $\mathbb{R}^d$. Nature first selects d weight coefficients $w_1, \ldots, w_d$ as $w_i \sim N(0, \tau^2)$ i.i.d. Given n examples $x_1, \ldots, x_n \in \mathbb{R}^d$, nature generates the output variable y_i as

$$y_i = \sum_{j=1}^{d} w_j x_{i,j} + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$ i.i.d.

Show that finding the coefficients $w_1, \ldots, w_d$ that maximizes $P[w_1, \ldots, w_d | (x_1, y_1) \ldots, (x_n, y_n)]$ is equivalent to minimizing the ridge optimization criterion.

2 **[Combining multiple classifiers]** The concept of "wisdom-of-the-crowd" posits that collective knowledge of a group as expressed through their aggregated actions or opinions is superior to the decision of any one individual in the group. Here we will study a version of the "wisdom-of-the-crowd" for binary classifiers: how can one *combine* prediction outputs from multiple possibly low-quality binary classifiers to achieve an aggregate high-quality final output? Consider the following iterative procedure to combine classifier results.

Input:
- S – a set of training samples: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, where each $y_i \in \{-1, +1\}$
- T – number of iterations (also, number of classifiers to combine)
- $\mathcal{F}$ – a set of (possibly low-quality) classifiers. Each $f \in \mathcal{F}$, is of the form $f : X \to \{-1, +1\}$

Output:
- F – a set of selected classifiers $\{f_1, \ldots, f_T\}$, where each $f_i \in \mathcal{F}$.
- A – a set of combination weights $\{\alpha_1, \ldots, \alpha_T\}$

Iterative Combination Procedure:
- Initialize distribution weights $D_1(i) = \frac{1}{m}$ [for $i = 1, \ldots, m$]
- **for** $t = 1, \ldots, T$ do
- **`// `** ϵ_j **`is weighted error of j-th classifier w.r.t.`** D_t
- Define $\epsilon_j := \sum_{i=1}^{m} D_t(i) \cdot \mathbf{1}[y_i \neq f_j(x_i)]$ [for each $f_j \in \mathcal{F}$]
- **`// select the classifier with the smallest (weighted) error`**
- $f_t = \arg\min_{f_j \in \mathcal{F}} \epsilon_j$
- $\epsilon_t = \min_{f_j \in \mathcal{F}} \epsilon_j$
- **`// recompute weights w.r.t. performance of`** f_t
- Compute classifier weight $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- Compute distribution weight $D_{t+1}(i) = D_t(i) \exp(-\alpha_t y_i f_t(x_i))$
- Normalize distribution weights $D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_i D_{t+1}(i)}$
- **endfor**
- return weights α_t, and classifiers f_t for $t = 1, \ldots, T$.

Final Combined Prediction:
- For any test input x, define the aggregation function as: $g(x) := \sum_t \alpha_t f_t(x)$, and return the prediction as $\text{sign}(g(x))$.

We'll prove the following statement: If for each iteration t there is some $\gamma_t > 0$ such that $\epsilon_t = \frac{1}{2} - \gamma_t$ (that is, assuming that at each iteration the error of the classifier f_t is just γ_t better than random guessing), then error of the aggregate classifier

$$\text{err}(g) := \frac{1}{m} \sum_i \mathbf{1}[y_i \neq \text{sign}(g(x_i))] \leq \exp(-2 \sum_{t=1}^{T} \gamma_t^2).$$

That is, the error of the aggregate classifier g decreases exponentially fast with the number of combinations T!

(i) Let $Z_t := \sum_i D_{t+1}(i)$ (i.e., Z_t denotes the normalization constant for the weighted distribution D_{t+1}). Show that

$$D_{T+1}(i) = \frac{1}{m} \frac{1}{\prod_t Z_t} \exp(-y_i g(x_i)).$$

(ii) Show that error of the aggregate classifier g is upper bounded by the product of Z_t: $\text{err}(g) \leq \prod_t Z_t$.
(hint: use the fact that 0-1 loss is upper bounded by exponential loss)

(iii) Show that $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$.
(hint: noting $Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i f_t(x_i))$, separate the expression for correctly and incorrectly classified cases and express it in terms of ϵ_t)

(iv) By combining results from (ii) and (iii), we have that $\text{err}(g) \leq \prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)}$, now show that:

$$\prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_t \sqrt{1-4\gamma_t^2} \leq \exp(-2\sum_t \gamma_t^2).$$

Thus establishing that $\text{err}(g) \leq \exp(-2\sum_t \gamma_t^2)$.

3 **[Low-dimensional information-preserving transformations]** (*hashing the cube*) You have a collection of nonzero distinct binary vectors $x_1, \ldots, x_m \in \{0,1\}^n$. To facilitate later lookup, you decide to hash them to vectors of length $p < n$ by means of a linear mapping $x_i \mapsto Ax_i$, where A is a $p \times n$ matrix with 0-1 entries, and all computations are performed modulo 2. Suppose the entries of the matrix are picked uniformly at random (ie, each an independent coin toss)

(i) Pick any $1 \leq i \leq m$, and any $b \in \{0,1\}^p$. Show that the probability (over the choice of A) that x_i hashes to b is exactly $1/2^p$. (Hint: focus on a coordinate $1 \leq j \leq n$ for which $x_{ij} = 1$.)

(ii) Pick any $1 \leq i < j \leq m$. What is the probability that x_i and x_j hash to the same vector? This is called a *collision*.

(iii) Show that if $p \geq 2\log_2 m$, then with probability at least $1/2$, there are no collisions among the x_i. Thus: to avoid collisions, it is enough to linearly hash into $O(\log m)$ dimensions!

(question credit: Prof. Sanjoy Dasgupta)

4 **[Regression Competition!]** You'll compete with your classmates on designing a good quality regressor for a large scale dataset.

(i) Signup on http://www.kaggle.com with your columbia email address.

(ii) Visit the COMS 4771 competition at: https://www.kaggle.com/t/025789e9024b4701b2ea7dada52834e9 and develop a regressor for large-scale dataset available there. You can use any resource publicly available to develop your regressor. (You don't need to submit your code for this question.)

(iii) Your pdf writeup should describe the design for your regressor: What preprocessing techniques and regressor you used? Why you made these choices? What resources you used and were helpful? What worked and what didn't work?

Make sure cite all the resources that you used.

Evaluation criterion:

- You must use your (and your group members') UNI as your team name in order to get points. For example:
  - If you have two group members with uni: `ab1234` and `xy0987`,
  - the teamname should be: `ab1234_xy0987`
- Your team must get a Mean Absolute Error (MAE) score of at most 6.800 on the private holdout test-dataset to get full credit. (This involves employing sound ML principles in the design and selection of your best performing model.)
- Extra points will be awarded to best performing teams, using the following scoring mechanism:

$$\text{Extra credit amount} = 5\left(1 - \frac{\text{Your Rank} - 1}{\text{Total number of valid group submissions}}\right) \cdot \mathbf{1}\Big[\text{Your score} < 6.5\Big]$$