COMS 4771 HW3 (Spring 2022)

Due: Sun April 10, 2022 at 11:59pm

This homework is to be done **alone**. 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 approaches for solutions for homework questions is encouraged on the course discussion board and with your peers, but you must write your own individual solutions and **not** share your written work/code. You must cite all resources (including online material, books, articles, help taken from/given to specific individuals, etc.) you used to complete your work.

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

2 Estimating Model Parameters for Regression

Let P_β be the probability distribution on $\mathbb{R}^d \times \mathbb{R}$ for the random pair (X, Y) (where $X = (X_1, \ldots, X_d)$) such that

$$X_1, \ldots, X_d \sim_{iid} N(0,1), \qquad \text{and} \qquad Y|X = x \;\sim\; N(x^\mathsf{T}\beta, \|x\|^2), \quad x \in \mathbb{R}^d$$

Here, $\beta = (\beta_1, \ldots, \beta_d) \in \mathbb{R}^d$ are the parameters of P_β, and $N(\mu, \sigma^2)$ denotes the Gaussian distribution with mean μ and variance σ^2.

(i) Let $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ be a given sample, and assume $x_i \neq 0$ for all $i = 1, \ldots, n$. Let f_β be the probability density for P_β as defined above. Define $Q : \mathbb{R}^d \to \mathbb{R}$ by

$$Q(\beta) := \frac{1}{n} \sum_{i=1}^{n} \ln f_\beta(x_i, y_i), \quad \beta \in \mathbb{R}^d.$$

Write a convex optimization problem over the variables $\beta = (\beta_1, \ldots, \beta_d) \in \mathbb{R}^d$ such that its optimal solutions are maximizers of Q over all vector of Euclidean length at most one.

(ii) Let $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ be a given sample, and assume $x_i \neq 0$ for all $i = 1, \ldots, n$. Let f_β be the probability density for P_β as defined above. Define $Q : \mathbb{R}^d \to \mathbb{R}$ by

$$Q(\beta) := \frac{1}{n} \sum_{i=1}^{n} \ln f_\beta(x_i, y_i), \quad \beta \in \mathbb{R}^d.$$

Find a system of linear equations $A\beta = b$ over variables $\beta = (\beta_1, \ldots, \beta_d) \in \mathbb{R}^d$ such that the solutions are maximizers of Q over all vectors in $\mathbb{R}^d$.

3 Different learning rates for different strategies (Extra Credit)

Here will explore how the rate by which an algorithm learns a concept can change based on how labeled examples are obtained. For that we look at three settings: (i) an active learning setting where the algorithm has the luxury of specifying a data point and querying its label, (ii) a passive learning setting where labeled examples are drawn at random and (iii) an adversarial setting where training examples are given by an adversary that tries to make your life hard.

Consider a binary classification problem where each data point consists of d binary features. Let $\mathcal{F}$ be the hypothesis class of conjunctions of subsets of the d features and their negations. So for example one hypothesis could be $f_1(x) = x_1 \wedge x_2 \wedge \neg x_d$ (where $\wedge$ denotes the logical "and" and $\neg$ denotes the logical "not"). Another hypothesis could be $f_2(x) = \neg x_3 \wedge x_5$. A conjunction in $\mathcal{F}$ cannot contain both x_i and $\neg x_i$. We assume a consistent learning scenario where there exists a hypothesis $f^* \in \mathcal{F}$ that is consistent with the true label for all data points.

(i) In the active learning setting, the learning algorithm can query the label of an unlabeled example. Assume that you can query *any possible example*. Show that, starting with a single positive example, you can exactly learn the true hypothesis f^* using d queries.

(ii) In the passive learning setting, where the examples are drawn i.i.d. from an underlying fixed distribution $\mathcal{D}$. How many examples are sufficient to guarantee a generalization error less than ϵ with probability $\geq 1 - \delta$?

(iii) Show that if the training data is not representative of the underlying distribution, a consistent hypothesis f^* can perform poorly. Specifically, assume that the true hypothesis f^* is a conjunction of k out of the d features for some $k > 0$ and that all possible data points are equally likely. Show that there exists a training set of $2^{(d-k)}$ unique examples and a hypothesis f that is consistent with this training set but achieves a classification error $\geq 50\%$ when tested on all possible data points.

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:
`https://www.kaggle.com/c/coms4771-spring-2022-regression-competition/`
You shall develop a regressor for a 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 UNI as your team name in order to get points. For example:
  - If your uni is abc123 then your teamname should be: `abc123`
- You must get a Mean Absolute Error (MAE) score of at most 500 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 the top ranked participants, using the following scoring mechanism:

$$\text{Extra credit amount} = 5\left(1 - \frac{\text{Your Rank} - 1}{\text{Total number of participants with MAE} < 300}\right) \cdot \mathbf{1}\Big[\text{Your score} < 300\Big]$$