COMS 4721 S18 Homework 4 (due Monday April 2)

Instructions

Submit your write-up on Gradescope as a neatly typeset (not scanned or handwritten) PDF document by 11:59 PM of the due date.

If you are working in a group, the group should produce a *single write-up* that is submitted by one of the group members. On Gradescope, the submitter must specify all of the group members upon submission of the write-up. More details can be found on the Gradescope Student Workflow help page (https://gradescope.com/help#help-center-section-student-workflow).

(Each group member should verify that the submission was successful.)

Make sure the names and UNIs of every member of your group appear prominently on the first page of your write-up.

On Gradescope, be sure to select the pages containing your answer for each problem. More details can be found on the Gradescope Student Workflow help page (https://gradescope.com/help#help-center-section-student-workflow). If you fail to select the pages containing your answer to a given problem, you will not receive any credit for that problem.

You are welcome to use the Markdown or LaTeX source for the assignment as a template for your write-up. I use Pandoc (http://pandoc.org) to translate the Markdown to LaTeX and ultimately to PDF.

Source code

Please combine all requested source code files into a *single* ZIP file[1], along with a plain text file called `README` that contains your name(s) and briefly describes all of the other files in the ZIP file. **Do not include the data files.** Submit this ZIP file on Courseworks. (Only one group member per group should do this.)

Clarity and precision

One of the goals in this class is for you to learn (i) to reason about machine learning problems and algorithms, and (ii) to make *clear* and *precise* claims and arguments about them.

A clear and precise argument is not the same as a long, excessively detailed argument. Unnecessary details and irrelevant side-remarks often make an argument less clear. And non-factual statements also detract from the clarity of an argument.

Points may be deducted for answers and arguments that lack sufficient clarity or precision. Moreover, a best-effort (but time-economical) attempt will be made to understand such answers/arguments, and the grade you will receive will be based on the correctness of this understanding.

[1] See https://en.wikipedia.org/wiki/Zip_(file_format).

Problem 1

In this problem you will practice constructing a conditional probability estimator.

Download the data set `hw4data.mat` from Courseworks, which has features vectors $x_1, \ldots, x_n \in \mathbb{R}^d$ and labels $y_1, \ldots, y_n \in \{-1, +1\}$ stored as `data` and `labels`, respectively; these examples were drawn iid from a distribution P. Using this data, produce a conditional probability estimator $\hat{\eta}\colon \mathbb{R}^d \to [0, 1]$ using the following approach.

1. Split the data into "training data" (first 75% of the examples) and "test data" (last 25% of the examples). I have already shuffled the order of the examples for you.
2. Using the training data, find a function $\hat{g}\colon \mathbb{R}^d \to \mathbb{R}$ with the aim of minimizing square loss risk. You should consider several different models (e.g., linear models with different feature expansions, neural networks, random forests) with various hyperparameters and use model selection. Your goal is to achieve square loss risk below 0.45. (The best possible risk is around 0.40, and the best constant prediction has risk around 0.51.) Of course, you cannot actually measure the square loss risk since you do not have access to the distribution P. You are free to use any existing packages to do the training and model selection, but you must appropriately cite any works/software packages that you did not write yourself.
3. Construct a conditional probability estimator $\hat{\eta}$ from $\hat{g}$.
4. Estimate the square loss risk of $\hat{g}$ using the test data, and report this estimate.
5. Here is a very simple example of the kind of inference that can be done with $\hat{\eta}$. Also included in the `hw4data.mat` is a matrix of feature vectors $x'_1, \ldots, x'_m \in \mathbb{R}^d$ stored as `quiz`. These examples were drawn iid from a distribution Q over $\mathbb{R}^d \times \{-1, +1\}$, but the labels have been omitted. Here, Q has the same conditional probability function as P (i.e., for $(X, Y) \sim P$ and $(X', Y') \sim Q$, the conditional distribution Y given $X = x$ is the same as that of Y' given $X' = x$ for all $x \in \mathbb{R}^d$), but the marginal of Q over $\mathbb{R}^d$ is different from that of P. Using $\hat{\eta}$, report the following estimate of the proportion of examples drawn from Q with a $+1$ label (i.e., $p_Q := \Pr_{(X', Y') \sim Q}(Y' = +1)$):

$$\widehat{p_Q} := \frac{1}{m} \sum_{i=1}^{m} \hat{\eta}(x'_i).$$

What to submit in your write-up:

(a) A description of your entire process in steps 2 and 3. Which models did you consider? How did you do model selection? How did you construct $\hat{g}$ and $\hat{\eta}$? Provide enough details so that a fellow student can reproduce your $\hat{g}$ and $\hat{\eta}$.
(b) Your estimate of square loss risk of $\hat{g}$ based on the test data.
(c) Your estimate of p_Q.

Please submit your source code on Courseworks.

Your solution:

(a)

(b)

(c)

Problem 2

Download the (modified) census data set from Courseworks. The file `adult-new.data` contains the training data, and the file `adult-new.test` contains the test data; both are plain text files. A description of the data can be found at https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.names. The prediction task is to determine whether a person makes more than $50,000 a year or not (i.e., map `>50K` to 1 and `<=50K` to 0.) Many of the features are *categorical*. For example, the feature `education` can take 16 different possible values. You should convert this feature into 16 *indicator features* (also called *dummy variables*), one per possible value.[2] Do this for all of the categorical features. Note that for three of the features, some rows show a value of "?" (which means "missing" or "unknown"); regard "?" as another possible value for those three features. In all, after this expansion, there should be 108 features.

Train three different classifiers using the training data. The first must be an affine classifier (in $\mathbb{R}^{108}$), the second must be based on decision or regression trees (perhaps with bagging or boosting), and the third can be any type of classifier of your choosing. Use model selection to determine hyperparameters. Your goal is to achieve (zero-one loss) risk below 0.16 (though it should be possible to do at least a couple of percentage points better). You are free to use any existing packages to do the training and model selection, but you must appropriately cite any works/software packages that you did not write yourself.

Compute the training and test error rates of each classifier. Also for each classifier, compute the false positive and false negative rates on the subset of test examples for which the `sex = Female`; do the same on the subset of test examples for which `sex = Male`. Report these values in a table to assess the "equalized odds" criterion for fairness for each of these three classifiers.

What to submit in your write-up:

(a) A description of your entire process for training the three classifiers. Provide enough details so that a fellow student can reproduce your classifiers.
(b) Training error rates and test error rates of the three classifiers.
(c) The false positive and false negative rates in a table as described above.

Please submit your source code on Courseworks.

Your solution:

(a)

(b)

(c)

[2] One can in fact get away with 15 features here.

Problem 3

(a) Let n be a positive integer larger than one. Suppose you have labeled examples $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{0, 1\}$ such that no two feature vectors are identical. Now you use bagging with a greedy decision tree learning algorithm to produce a classifier $\hat{f}$ that is the majority vote of m decision trees. The greedy decision tree learning algorithm is one that greedily grows a tree until every leaf is "pure" (which is possible because no two feature vectors are identical). Prove that if

$$m \geq 28 \ln(100n),$$

then with probability at least 0.99 (over the randomness used to perform bootstrap sampling), the training error rate of $\hat{f}$ is zero.

Hint: Consider using the union bound and a tail bound for sums of iid Bernoulli random variables (or "Chernoff bounds", below) to find upper bounds on the following probabilities:

1. The probability that the first decision tree misclassifies (x_1, y_1).
2. The probability that $\hat{f}$ misclassifies (x_1, y_1).
3. The probability that there exists $i \in \{1, \ldots, n\}$ such that $\hat{f}$ misclassifies (x_i, y_i).

(b) Let $\mathcal{C}$ be a concept class on domain $\mathcal{X}$. Suppose you have an algorithm A that, for any $\epsilon \in (0, 1)$, any distribution P over $\mathcal{X}$, and any $c \in \mathcal{C}$, if A is given $(x_1, c(x_1)), \ldots, (x_n, c(x_n))$ as input, where $x_1, \ldots, x_n \sim_{\text{iid}} P$ and $n \geq s_A(1/\epsilon)$, then with probability at least $1/4$, A runs in time $t_A(1/\epsilon)$ and returns a hypothesis h such that $\Pr_{x \sim P}(h(x) \neq c(x)) \leq \epsilon$. Here, s_A and t_A are fixed univariate polynomials.

Design an algorithm B (which can use A as a subroutine) that efficiently PAC-learns every concept in $\mathcal{C}$. Specifically, for any $\epsilon, \delta \in (0, 1)$, any distribution P over $\mathcal{X}$, and any $c \in \mathcal{C}$, if B is given $(x_1, c(x_1)), \ldots, (x_n, c(x_n))$ as input, where $x_1, \ldots, x_n \sim_{\text{iid}} P$ and $n \geq s_B(1/\epsilon, 1/\delta)$, then with probability at least $1 - \delta$, B must run in time $t_B(1/\epsilon, 1/\delta)$ and return a hypothesis h such that $\Pr_{x \sim P}(h(x) \neq c(x)) \leq \epsilon$. Here, s_B and t_B must be fixed bivariate polynomials. Prove that B works as required.

Hint: Suppose you split the examples into m groups (perhaps with $n/m \geq s_A(1/\epsilon)$), and run A on each group. If A "succeeds" in one of these m runs, then you could just return the hypothesis from that run. Note that it is possible A does not return a hypothesis in some cases (e.g., perhaps it outputs "I give up"), and it is also possible that it returns a hypothesis with error rate more than ϵ. How can you recognize when a run of A is successful?

You may use the following inequalities, called *Chernoff bounds*. Let $X_1, \ldots, X_n \sim_{\text{iid}} \text{Bern}(p)$, and let $S := X_1 + \cdots + X_n$. For any $\gamma \in [0, 1]$,

$$\Pr(S > (1 + \gamma)pn) \leq e^{-np\gamma^2/3},$$
$$\Pr(S < (1 - \gamma)pn) \leq e^{-np\gamma^2/2}.$$

Your solution:

(a)

(b)