CS194-172 HW3

Due: November 2, 2021 (12:30 pm)

Problem 1: Naive Bayes (10 pt)

Let $X \in \mathbb{R}^d$ be a set of features and Y be a binary variable with values in $\{0, 1\}$ representing the class label. Further assume that the following assumptions hold:

- The label variable Y follows a Bernoulli distribution, with parameter $\pi = P(Y = 1)$.
- For each feature X_j, the conditional probability $P(X_j \mid Y = y_k)$ follows a Gaussian distribution of the form $\mathcal{N}(\mu_{jk}, \sigma_j)$.
- The Naive Bayes assumption applies such that for all $i \neq j, X_i$ and X_j are conditionally independent given Y.

Given these assumptions, compute $P(Y = 1 \mid X)$ and show that it can be written as

$$P(Y = 1 \mid X) = \frac{1}{1 + \exp(-b + w^T X)}$$

Specifically you should be able to write b and w in terms of π, μ_{jk}, and σ_j for $j = 1, \ldots, d$ and $k \in \{0, 1\}$. This shows that there is a close connection between Naive Bayes and logistic regression. *Hint: use Bayes rule!*

Problem 2: Random Forests (20 pt)

(a) Random forests use ensemble learning to reduce the variance of an individual decision tree by averaging predictions across multiple decision trees. We will explore this further in this problem.

i. (4 pt) Consider a set of N uncorrelated variables, $X_1, \ldots, X_N$ with mean μ and variance σ^2. Calculate the variance of their average. (In the context of ensemble learning, X_i is analagous to the prediction made by classifier i.)

ii. (4 pt) Random forests use bagging (short for **b**oostrap **agg**regat**ing**), an ensemble learning method in which each learner is trained on a separate boostrapped dataset.

To construct a boostrapped dataset, we randomly sample M data points (with replacement) from the original training set of size M. In this process, some points may be chosen multiple times, while others may not be chosen at all. The points that are not chosen are denoted as out-of-bag (OOB) samples for that decision tree.

Why do we expect that 37% of samples will be OOB for large values of M? *Hint: There is a well-known limit, which you can look up, that is useful for this problem.*

iii. (2 pt) To determine the appropriate number of decision trees in a random forest model, we could use cross-validation. An alternative is to examine the OOB error. Explain what the OOB error is and how it can be used to choose the appropriate number of decision tress.

iv. (4 pt) In part i., we found that averaging reduces variance for uncorrelated classifiers. Predictions made by different decisions trees, however, will not be completely uncorrelated, but reducing correlation among decision trees will reduce the final variance. To illustrate this, consider a set of correlated random variables $X_1, \ldots, X_N$. Similar to part i., they each have mean μ and variance σ^2. However, this time, $0 \leq \mathrm{Cov}(X_i, X_j) \leq \delta^2$ for all $i \neq j$. Upper bound the variance of the average of the X variables.

v. (2 pt) Lower bound the variance from the previous part as $N \to \infty$. Compare this result to part (a) and comment on the limitations of bagging.

(b) (4 pt) The splitting criterion is a crucial component of a decision tree algorithm. As discussed in class, the most common splitting criteria are the classification error rate, Gini index, and entropy. In this problem, we will examine how these criteria compare to each other.

Suppose there are only two classes. Plot the classification error rate, Gini index, and entropy of a node as a function of p_1 (the probability of the first class).

Definitions: For a K-valued discrete random variable with probability mass function p_i for $i \in \{1, \ldots, K\}$, the classification error rate is defined as $1 - \max_k p_k$, the Gini index is defined as $\sum_{k=1} p_k(1 - p_k)$, and the entropy is defined as $-\sum_{k=1}^{k} p_k \log p_k$.

Problem 3: Support Vector Machines (20 pt)

(a) (10 pt) Recall that the maximum margin hyperplane on a set of n linearly-separable training observations $x_1, \ldots, x_n \in \mathbb{R}^p$ and associated class labels $y_1, \ldots, y_n \in \{-1, 1\}$ is the solution to the following optimization problem:

$$\max_{\beta_0, \beta_1, \ldots, \beta_p, M} M$$

subject to

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}) \geq M \qquad \forall i \in \{1, \ldots, n\}$$

In this problem, we focus on the meaning of the constraints. The second constraint ensures that each training example is on the correct side of the hyperplane, with some cushion (or margin) M. The first constraint adds a geometric meaning to the margin, which you will show in this problem.

Specifically, prove the following statement: given a hyperplane defined by $\beta_0, \beta_1, \ldots, \beta_p$, the perpendicular distance from x_i to the hyperplane is given by

$$y_i(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}).$$

assuming that

$$\sum_{j=1}^{p} \beta_j^2 = 1.$$

(b) (6 pt) The support vector classifier is the solution to the following optimization problem:

$$\max_{\beta_0, \beta_1, \ldots, \beta_p, \epsilon_1, \ldots, \epsilon_n} M$$

subject to

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}) \geq M(1 - \epsilon_i) \qquad \forall i \in \{1, \ldots, n\}$$

$$\epsilon_i \geq 0 \qquad \forall i \in \{1, \ldots, n\}$$

$$\sum_{i=1}^{n} \epsilon_i \leq C$$

As with maximal margin classifiers, let M denote the margin. In this problem, we focus on the interpretation of the slack variables ϵ_i.

- (2 pt) If $\epsilon_i = 0$, is x_i on the correct side of the hyperplane? If so, is it also on the correct side of the margin?
- (2 pt) If $0 < \epsilon_i < 1$, is x_i on the correct side of the hyperplane? If so, is it also on the correct side of the margin?
- (2 pt) If $\epsilon_i > 1$, is x_i on the correct side of the hyperplane? If so, is it also on the correct side of the margin?

(c) (4 pt) In class, we discussed that a support vector classifier takes the form

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \ldots + \beta_p x_p^*$$

for a new observation x^* in $\mathbb{R}^p$. We also mentioned that the above representation is equivalent to

$$f(x^*) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x^*, x_i \rangle$$

for some appropriate chosen values of α. (x_i denotes the i^{th} training observation.

Prove that the first representation is equivalent to the second. Specifically, find values for β_j for $j \in \{1, \ldots, p\}$ in terms of α_i and x_i for $i \in \{1, \ldots, n\}$.

Problem 4: Cross-validation (15 pt)

Pulsars are a rare type of Neutron star that produce radio emissions detectable on Earth. In this problem, you will train SVM models to detect pulsars using features derived from radio emissions.

Starter code to generate the training and test datasets from `pulsar_data.csv` file has been provided in `p4.py`.

(a) (5 pt) Train SVM classifiers on the `(X_train, Y_train)` dataset with four different kernels: linear, polynomial with degree 2, polynomial with degree 5, and radial-basis function (RBF). All classifiers should be trained using the `sklearn.svm.SVC` model class.

Each of these classifiers is highly dependent on the value of the hyperparameter C, which controls the width of the margin. To determine the appropriate value of C, implement 5-fold cross-validation *without using any existing libraries*. The appropriate value of C should be chosen among powers of 10 from 10^{-1} to 10^{3}. After choosing the value of C that maximizes the mean accuracy across the 5 folds (ties between equivalently good hyperparameters can be broken arbitrarily), retrain the model on the full `(X_train, Y_train)` dataset using that value of C.

(b) (5 pt) For each of the four classifiers, report the best value of C, the mean cross-validation accuracy for the best value of C, the test accuracy, and the running time of cross-validation in a table.

(c) (5 pt) For each of the four classifiers, plot the ROC curve curve on the test set and display the associated AUROC values.

(d) (2 pt) *Extra credit*: In part (b), you should have found that the linear SVM takes the longest to train despite being the simplest kernel. Explain why you observe this.

Problem 5: Training a variant effect predictor (35 pt)

In this problem, you will train a machine learning algorithm to predict the pathogenicity of missense variants. Your model will be trained on variants from ClinVar, a publicly accessible database of human variants and their associated phenotypes.

(a) (3 pt) First, we will construct our dataset using the data from ClinVar. We have already preprocessed the raw ClinVar data and extracted only missense variants to simplify this process. The processed data is located in `clinvar_missense.vcf`. However, we should not construct a dataset using all variants in this file as not all of them have benign or pathogenic clinical significances.[1]

To isolate only benign and pathogenic missense variants, exclude all variants that do not have a *Benign* or *Pathogenic* `CLNSIG` value. After this filtering, how many benign and pathogenic variants remain? Assign pathogenic variants a class label of 1 and benign variants a class label of 0.

Note: Do not include variants with *Likely_benign* or *Pathogenic/Likely_pathogenic* clinical significances. The significance string should exactly match either *Benign* or *Pathogenic*.

(b) (2 pt) Randomly split the dataset from part (a) into a training set and a validation set such that the training set contains 80% of the variants. How many benign and pathogenic variants are in each split?

(c) (5 pt) Now, we are ready to construct features for each variant. We will first include two features that quantify the harmfulness of mutating a gene: RVIS and o/e. RVIS (Residual Variation Intolerance Score) ranks genes according to whether they have more or less common functional (i.e. missense or nonsense) variation relative to the amount of neutral variation.

Intuitively, the RVIS score captures the following phenomenon. If a gene contains 1000 variants from a sequencing study of healthy individuals and 900 of them are common (i.e. present with an AF $> 1\%$) missense variants, then mutations in this gene are tolerated. On the other hand, if the gene contained only 10 common missense variants out of the 1000 observed, then mutations in this gene are not tolerated.

Download the RVIS data from `http://genic-intolerance.org/data/RVIS_Unpublished_ExACv2_March2017.txt`. The important columns in this file are the `CCDSr20` column, which contains the gene symbol, and the `%RVIS[pop_maf_0.05%(any)]` column, which contains the RVIS score—normalized as a percentile—for that gene.

Find the RVIS score for each variant in our dataset. To determine which gene each variant overlaps, examine the `GENES` info field in the VCF. For genes without an RVIS score, assign a default percentile of 50, and for variants that overlap multiple genes, take an average of their RVIS scores.

Plot the histogram of RVIS values for pathogenic variants and benign variants. Based on the plot, do you think RVIS is a good feature to add to our model?

(d) (5 pt) The o/e (observed/expected) ratio is a similar metric computed using newer data than RVIS, gnomAD instead of ExAC, and more sophisticated statistical models. Like RVIS, the o/e ratio provides a score for each gene.

[1] `https://www.ncbi.nlm.nih.gov/clinvar/docs/clinsig/`

Download the o/e data from `https://storage.googleapis.com/gcp-public-data--gnomad/release/2.1.1/constraint/gnomad.v2.1.1.lof_metrics.by_gene.txt.bgz`. The important columns in this file are the `gene` column and the `oe_lof_upper_rank` column, which provides the o/e rank for the gene.

Similar to the previous part, find the o/e score for each variant in our dataset. Plot the histogram of o/e values for pathogenic variants and benign variants. Based on the plot, do you think the o/e metric is a good feature?

Hint: Similar to the previous part, you will need to handle the following two edge cases: (a) genes without o/e scores and (b) variants overlapping multiple genes.

(e) (5 pt) As described earlier this semester, phastCons is a two-state Hidden Markov Model (HMM) that divides the genome into conserved and nonconserved segments using multiple sequence alignments and phylogenetic data. The phastCons score for each nucleotide describes the probability that the nucleotide is part of a conserved region and can be incorporated as a feature to determine whether a missense mutation is likely to be damaging.

The nucleotide-level phastCons scores are stored in a bigWig file, which can be downloaded from `https://hgdownload.cse.ucsc.edu/goldenPath/hg38/phastCons100way/hg38.phastCons100way.bw`. (Warning: This file is more than 5 GB!) To read bigWig files, use the `pyBigWig` library. Documentation can be found on its Github repo: `https://github.com/deeptools/pyBigWig`.

Plot the histogram of phastCons values for pathogenic variants and benign variants. Submit that graph. Do you think this is a good feature?

(f) (5 pt) Train a model using these three features. You are allowed to choose among any of the model classes (Naive Bayes, Random Forests, SVMs) we have previously discussed in class. Be sure to do careful cross-validation in order to choose the appropriate model class and other hyperparameters.

List the model you used and any other hyperparameters chosen using cross-validation. Plot the ROC curve on the validation set and display the AUROC.

(g) (10 pt) On Monday, October 25, we will release an independent test set on which we will assess the performance of your model. More details will be posted on Piazza before then.