EECS 553 HW 3

Due Thursday, September 19, by 11:59 PM via Gradescope and Canvas

When you upload your solutions to Gradescope, please indicate which pages of your solution are associated with each problem. Also please note that your code should be uploaded to Gradescope (as a .pdf) and Canvas (as .py) as discussed below.

1. OSM Hyperplane Classifier (3 points each)

Let $(\boldsymbol{w}^*, b^*, \boldsymbol{\xi}^*)$ be a solution to the optimal soft-margin hyperplane classifier.

(a) Argue that if $\boldsymbol{x}_i$ is misclassified, then $\xi_i \geq 1$.

(b) Show that if $\xi_i^* > 0$, then ξ_i^* is proportional to the distance from $\boldsymbol{x}_i$ to the margin hyperplane associated with class y_i (that is, the set $\{\boldsymbol{x} : (\boldsymbol{w}^*)^T\boldsymbol{x} + b^* = y_i\}$), and give the constant of proportionality.

2. Optimal soft-margin hyperplane (5 points)

Consider a variation of the optimal soft-margin linear classifier define by

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}[(1-\alpha)\sum_{i:y_i=1}^{n}\xi_i + \alpha\sum_{i:y_i=-1}\xi_i] \\ \text{s.t. } & y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned}$$

where $\alpha \in (0,1)$ is a parameter that captures a desire to penalize either false positives or false negatives more than the other. Show that the resulting linear classifier can also be derived by regularized empirical risk minimization with a certain loss. Determine the loss (which will depend on α), and state the regularization parameter λ in terms of C such that regularized ERM yields the same classifier as the quadratic program above.

3. Naive Bayes for Document Classification: Cars or Motorcycles? (3 points each)

In this problem, you will implement a Naive Bayes classifier on the Newsgroup20 dataset. The documents in the data set are messages sourced from 20 newsgroups (an online message forum). You will build a classifier to determine whether a message originated from a newsgroup about cars or motorcycles, using the bag-of-words representation discussed in class.

You should implement this algorithm yourself. Do not use existing implementations. The following section provides background for the Multinomial Naive Bayes model you will be implementing.

Naive Bayes Background

Each sample in the data set corresponds to one document, and consists of a d-dimensional feature vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^T$ and a binary label $y \in \{0, 1\}$. Each feature x_j is a count of the number of times word j occurs in the document.

For a given feature vector $\boldsymbol{x}$, Naive Bayes makes a classification by estimating quantities in the following formula for the Bayes classifier:

$$\widehat{y} = \arg\max_{k \in \{0,1\}} \pi_k \prod_{j=1}^{d} \Pr(X_j = x_j | Y = k).$$

Here, X_j is the jth feature, viewed as a random variable, Y is the label, and $\Pr(X_j = \ell | Y = k)$ is the probability that word j occurred ℓ times in document i, given the document belongs to class k (bag of words model).

For *multinomial Naive Bayes*, we will model this class-conditional probability as

$$\Pr(X_j = \ell | Y = k) = (p_{kj})^{\ell}$$

using the following estimate for p_{kj}:

$$\widehat{p}_{kj} = \frac{n_{kj} + \alpha}{n_k + \alpha d},$$

where $n_k = \sum_{i:y_i=k} \sum_j x_{ij}$ is the total number of words in label k documents, and $n_{kj} = \sum_{i:y_i=k} x_{ij}$ is the number of times word j appears in documents with label k, with smoothing parameter $\alpha = 1$ to ensure $\widehat{p}_{kj} > 0$.

π_k denotes the class-k priors, which are estimated as $\widehat{\pi}_k = \frac{m_k}{n}$, where $m_k = |\{i : y_i = k\}|$ is the number of label k documents.

Notice that in the classification rule, we take the product over probabilities, and we exponentiate a probability in the class-conditional estimate. In some situations, multiplying many numbers less than one can result in *underflow*, as the product becomes very small. The resulting numerical error can worsen performance of the classifier. Thankfully, underflow can be mitigated by doing operations in log-space, and converting back via exponentiation when needed.

In this problem you will implement the classifier

$$\widehat{y} = \arg\max_{k \in \{0,1\}} \log \left(\widehat{\pi}_k \prod_{j=1}^{d} (\widehat{p}_{kj})^{x_j} \right). \tag{1}$$

The log does not change the maximizer since it is a strictly increasing function. Here and throught out the course, logarithms are natural unless a different base is specified.

(a) Intuitively, p_{kj} is the probability that feature j appears once in a class-k document. What additional assumption about the data, in addition to the naive Bayes assumption, makes this intuition valid?

(b) By applying properties of the natural logarithm, simplify the expression on the right-hand side of (1), substituting the formula for $\widehat{p}_{kj}$. Your final expression should not contain logs of products or divisors.

(c) Download the training and test datasets from Canvas under Files → Homework → HW2 → hw2p2_data.zip. `hw2p2_train_x.npy`, `hw2p2_train_y.npy` are the training features and labels respectively, `hw2p2_test_x.npy` and `hw2p2_test_y.npy` are the test data. The data can be loaded with `numpy.load(filename)`. For example:

`train_x = np.load("hw2p2_train_x.npy")`

`train_y = np.load("hw2p2_train_y.npy")`

Here, `train_x` has dimensions $n_{train} \times d$, and `train_y` is an n_{train}-vector of 0s and 1s, denoting a message from the cars group or motorcycles group, respectively. The training set consists of $n_{train} = 1192$ samples, and the test set has $n_{test} = 794$ samples. There are $d = 1000$ features, where each feature denotes the number of times a particular word appeared in a document.

(i) Use the training data to estimate $\log p_{kj}$ for each class $k = 0, 1$ and for each feature $j = 1, \ldots, d$.

(ii) Also estimate the log-priors $\log \pi_k$ for each class.

Report your answers to (ii).

(d) Use the estimates for $\log p_{kj}$ and $\log \pi_k$ to predict labels for the testing data. That is, apply the decision rule with the samples in `test_x.npy` and `test_y.npy`. Report the test error.

(e) What would the test error be if we always predicted the same class, namely, the majority class from the training data? (Use this result as a sanity check for the error in part (d))

(f) Submit all code used for (c), (d), and (e) as a single .py file. Submit this file to the corresponding assignment on Canvas. In addition, please also submit a .pdf version of your code (just print the .py file to pdf) and upload to gradescope for part (f). This will make it easier for graders who want to take a quick look at your code without running it. Your code should follow best coding practices including the use of comments, indentation, and descriptive variable names.