COMS 4771 Machine Learning *Fall 2025*

Homework 2

Model selection

The wine dataset collects various features of Italian wines from three different wineries. The details of the dataset (including the features) are given here: `https://archive.ics.uci.edu/dataset/109/wine`. We have split the dataset into training data and test data, available in the file `wine.pkl`. You can load the dataset as follows.

```python
import pickle
wine = pickle.load(open('wine.pkl', 'rb'))
```

The format of the resulting dictionary object `wine` is similar to that of the MNIST dataset. Note that the possible labels are $\mathcal{Y} = \{0, 1, 2\}$ instead of $\{1, 2, 3\}$.

Problem 1. Suppose you would like to find a classifier (to minimize the usual error rate) based on the normal generative model for the wine dataset, but you would like it to only use at most two features. (Why? It may be expensive to obtain the feature values for different wines, so it is desirable to use as few features as possible.) You should do this using some form of cross validation; you may use any reasonable approach you like.

(a) Describe your approach in sufficient detail so that another student would be able to reproduce your results. Give all relevant numerical results from the process (e.g., how you split the training data, validation error rates for the various models you try).

(b) Describe in detail the final classifier you obtain (e.g., which features were used, parameters of the model). Report the training and test error rates of this final classifier.

Optimal classifiers

Suppose (X, Y) is a random example, with range$(X) = [0, 1]$ and range$(Y) = \{0, 1\}$. The distribution of (X, Y) is as follows. The marginal distribution of X is the uniform probability density on the interval $[0, 1]$. For any $x \in [0, 1]$, the conditional probability that $Y = 1$ given $X = x$ is

$$\Pr(Y = 1 \mid X = x) = \begin{cases} 0.25 & \text{if } x \leq 0.2, \\ 0.6 & \text{if } 0.2 < x < 0.8, \\ 0.3 & \text{if } x \geq 0.8. \end{cases}$$

For any classifier $f \colon [0, 1] \to \{0, 1\}$, let $\text{err}[f] = \Pr(f(X) \neq Y)$ denote its error rate. Let $f^\star$ denote an optimal classifier (i.e., a classifier with minimum error rate).

Problem 2.

(a) Compute the exact value of $\text{err}[f^\star]$. Show a brief derivation of your answer.

(b) When the input space $\mathcal{X}$ is (a subset of) $\mathbb{R}$, a decision stump is a decision tree that has the form:

"if $x \leq t$, then return a, else return b."

Above, x is the input (a scalar variable), and the parameters of the decision stump are $t \in \mathbb{R}$, $a \in \{0, 1\}$, and $b \in \{0, 1\}$.

Write a Python function that, given (the parameters of) such a decision stump T, returns its error rate $\text{err}[f_T]$. (Here, f_T refers to the function implemented by the decision stump T.) Use your code to determine the error rates of the following three decision stumps:

$$\begin{aligned} T_1 &= \text{“if } x \leq 0.4\text{, then return 0, else return 1”;} \\ T_2 &= \text{“if } x \leq 0.5\text{, then return 0, else return 1”;} \\ T_3 &= \text{“if } x \leq 0.5\text{, then return 1, else return 0”.} \end{aligned}$$

Your procedure should compute the error rates exactly (up to numerical precision). It should not simply estimate it using a finite number of randomly generated examples.

(c) For each $t \in \mathbb{R}$, let $e(t)$ denote the smallest error rate achievable by a decision stump that uses the predicate "$x \leq t$?" with its non-leaf node.[a] Plot $e(t)$ as a function of t in the range $-0.2 \leq t \leq 1.2$. Make sure the axes of your plot are appropriately labeled.

What is the threshold parameter t of the decision stump T that has the smallest error rate $\text{err}[f_T]$ among all decision stumps? And what is its (exact) error rate?

(d) Write a Python function that, given training dataset $(x_1, y_1), \ldots, (x_n, y_n)$ from $[0, 1] \times \{0, 1\}$, returns (the parameters of) a decision stump that the smallest possible error rate on the training dataset, along with its training error rate. Apply this function to the following dataset (with $n = 17$ examples):

x	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9
y	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1

What is the decision stump you obtain, and what is its training error rate? (Please report three significant digits for the error rate.)

Hint: You don't need to worry about getting the fastest possible runtime with your implementation, but you will need to use it in the next part many times (see below), so it shouldn't be too slow either.

(e) Now let us consider what decision stump might be selected by a learning algorithm that does not know the distribution of (X, Y), but instead only has a finite sample.

Let $\widehat{T}$ be a decision stump with the smallest number of classification mistakes on the training examples $(X_1, Y_1), (X_2, Y_2), \ldots, (X_{100}, Y_{100}) \sim_{\text{i.i.d.}} (X, Y)$. Let $\hat{t}$ denote the threshold in the predicate used by the decision stump $\widehat{T}$. Let $\text{err}[f_{\widehat{T}}]$ denote the error rate of $\widehat{T}$. Use Python to simulate the random generation of the 100 training examples, the construction of $\widehat{T}$, and the computation of $\text{err}[f_{\widehat{T}}]$. Some Python code for carrying this out will be provided, but you will need to use your Python functions from previous parts.

Run the simulation 5000 times, recording the values of $\hat{t}$ and $\text{err}[f_{\widehat{T}}]$. Plot two histograms: one for $\hat{t}$, and another for $\text{err}[f_{\widehat{T}}]$. Make sure the axes of your histograms are appropriately labeled.

(f) (Continuing from Part (e).) Examine the histograms for $\hat{t}$ and $\text{err}[f_{\widehat{T}}]$. In each histogram, if you squint a bit, you should observe a few "peaks" (of varying heights).

- What is the relationship between the peaks across the two histograms? Please note the (approximate) "locations" of the peaks.
- Why do you think these peaks occur? Try your best to give a well-thought-out explanation, but it is not necessary to be completely rigorous.

Hint: You may also want to refer back to the plot of $e(t)$ from Part (c).

[a]There is a question as to whether decision stumps are allowed to have the same label at both leaf nodes. A decision stump where the leaf nodes have the same label is a constant function—the predicate is irrelevant. If such decision stumps are allowed, then $e(t)$ is never larger than the error rate of the "best" constant function. Please state whether or not you are allowing such decision stumps. (Either is fine, but your choice will affect $e(t)$ and what its plot looks like.)

Linear regression

The prostate cancer dataset records prostate-specific antigen (PSA) levels and several other clinical measurements in cancer patients. The dataset is described in *Elements of Statistical Learning* by Hastie, Tibshirani, and Friedman. You can load the dataset as follows.

```
import pickle
prostate = pickle.load(open('prostate.pkl', 'rb'))
```

The resulting dictionary object `prostate` has three keys: `'names'` maps to a list of the variable names, `'data'` maps to the training data, and `'testdata'` maps to the test data. (The test data will not be used in this homework.) The problem associated with this dataset is to predict the `lpsa` (logarithm of the PSA level) variable by a function of the other variables. Note that `lpsa` is the last "column" of `prostate['data']` (and of `prostate['testdata']`).

Problem 3.

(a) For each of the eight features, use the training data to find the best fit affine function of that single variable to the label `lpsa`. (By "best", we mean of minimum sum of squared errors.) Report the slope and intercept in each case.

(b) Use the training data to find the best fit affine function of all eight features (together as a vector in $\mathbb{R}^8$) to the label `lpsa`. Report the coefficients in the weight vector and the intercept term.

Please report three significant digits for each slope/coefficient/intercept.

You should find that some of the variables have a negative coefficient in the weight vector from Problem 3(b), even though its corresponding affine function from Problem 3(a) has a positive slope. This might seem like a paradox: for such a feature, your answers from Problem 3(a) might lead you to think that increasing the feature's value should, on average, increase the (predicted) value of `lpsa`; whereas your answer from Problem 3(b) might lead you to think that increasing the feature's value should, on average, decrease the (predicted) value of `lpsa`.

Of course, there is no paradox. The following problem shows how this can happen.

Problem 4. Suppose the random vector $X = (X_1, X_2)$ follows a bivariate normal distribution, with mean zero and covariance matrix

$$\begin{bmatrix} 1 & 2/3 \\ 2/3 & 1 \end{bmatrix}$$

(which means $\operatorname{var}(X_1) = \operatorname{var}(X_2) = 1$, and $\operatorname{cov}(X_1, X_2) = 2/3$), and define the random variable Y by

$$Y = \frac{3}{2}X_1 - \frac{3}{4}X_2.$$

(a) What is the affine function of X_1 that has smallest mean squared error for predicting Y? Give the slope and intercept, and show a short derivation.

Hint: Use the normal equations but replace "averages" computed over training data by "expectations" with respect to the distribution of (X, Y).

(b) What is the affine function of X_2 that has smallest mean squared error for predicting Y? Again, give the slope and intercept, and show a short derivation.

(c) And finally, what is the affine function of (X_1, X_2) that has smallest mean squared error for predicting Y? Give the weight vector and intercept, and show a short derivation.

You should find that even though each of X_1 and X_2 is positively correlated with Y (analogous to the situation in Problem 3(a)), the best affine predictor of Y that considers both X_1 and X_2 has a positive coefficient for one variable and a negative coefficient for the other variable (analogous to the situation in Problem 3(b)). This example shows that one must be careful when interpreting the sign of the regression coefficient.

Freedman's paradox

Load the training dataset `freedman.pkl` as follows:

```
import pickle
freedman = pickle.load(open('freedman.pkl', 'rb'))
```

In `freedman['data']`, there is an $n \times d$ matrix whose rows correspond to n feature vectors $x^{(1)}, \dots, x^{(n)} \in \mathbb{R}^d$. In `freedman['labels']`, we have the corresponding labels $y^{(1)}, \dots, y^{(n)} \in \mathbb{R}$. Features and labels are *approximately* standardized (i.e., mean zero and variance one).

Suppose we are interested in predicting the label from the feature vectors using a (homogeneous) linear function. Note that $n = 100$ and $d = 1000$, so we don't expect ordinary least squares to work well on this dataset. But we might hope that ordinary least squares will work well if we only consider a small number of features. So consider the following three-step procedure:

- (Step 1.) Estimate the correlations between the features and the label:

  $$\hat{\rho}_j := \frac{1}{n} \sum_{i=1}^{n} x_j^{(i)} y^{(i)} \quad \text{for } j = 1, \dots, d.$$

  (This is a reasonable estimate since the features and labels are approximately standardized.)

- (Step 2.) Let $\widehat{J}$ be the set of features (well, technically the indices of these features) for which the estimated correlation is larger than $2/\sqrt{n}$:

  $$\widehat{J} := \{j \in \{1, \dots, d\} : |\hat{\rho}_j| > 2/\sqrt{n}\}.$$

- (Step 3.) Now construct the ordinary least squares estimate $\hat{w} \in \mathbb{R}^d$ using only features in $\widehat{J}$. In other words,

  $$\hat{w} \in \mathop{\arg\min}_{\substack{w \in \mathbb{R}^d \text{ such that} \\ w_j = 0 \text{ for all } j \notin \widehat{J}}} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2.$$

  This is equivalent to removing the columns of `freedman['data']` that are *not* in $\widehat{J}$, and then applying ordinary least squares with the resulting dataset.

The first two steps comprise a "screening procedure" whose purpose is to remove irrelevant features. If the number of features $|\widehat{J}|$ remaining after the screening is much smaller than n, then one might think that ordinary least squares (using only features in $\widehat{J}$) will work well; this is the motivation for Step 3.

Problem 5.

(a) Carry-out the procedure described above on the given training data. How many features are in $\widehat{J}$ after Step 2? It should be much smaller than n.

(b) What is the empirical risk of $\hat{w}$ after Step 3, i.e.,

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{w}^{\mathsf{T}}x^{(i)} - y^{(i)})^2?$$

It should be much smaller than the variance of the labels on this dataset (which is close to 1).

(c) A separate test set is available in `freedman['testdata']`, `freedman['testlabels']`. What is the risk of $\hat{w}$ on this test set? (This can be regarded as "test risk" of $\hat{w}$.)

(d) It turns out that all features and labels are drawn independently from $\mathrm{N}(0, 1)$—i.e., there is no real correlation between the features and labels. Yet, based on the relatively small empirical risk of $\hat{w}$, together with the small number of features used in $\hat{w}$ relative to the sample size n, one might have thought that there *is* a relationship between the features and labels! This apparent paradox is called "Freedman's paradox". Briefly explain why this happens.

Optional: What is the "true" risk of $\hat{w}$? The answer can be expressed as a certain function of $\hat{w}$.

(e) Suppose Step 3 was modified to use an independent and identically distributed dataset (instead of the same dataset used in Steps 1 and 2). In fact, `freedman['data2']` and `freedman['labels2']` can be used for this purpose. So $\widehat{J}$ is determined using the first dataset, but $\hat{w}$ is now obtained using the second dataset. Do you think the empirical risk (on the second dataset) of the resulting weight vector $\hat{w}$ will be higher or lower than the answer from Part (b)? Please explain your answer. (Try to come up with an answer and explanation before trying it out using the data.)

Solutions

`https://drive.google.com/file/d/1eureIOCOlwZnj_1BdQqv4qmBZTZk9fNF/view?usp=drive_link`