CMSC 726 : Machine Learning *First Last, UID, email*

HW2

The deadline for submission is 11:59:59 PM, Thursday, September 30.

0.1 Instructions

- Submit your assignment via ELMS, at `http://elms.umd.edu`.
- Your submission should include your write-up, as well as any code you implemented.
- Do *not* include code in the write-up, unless asked to.
- All MATLAB code should be documented with inline comments in the code. This includes things like a brief script/function description, input/output arguments and any implementation details that seem important to document. If you have any instructions on the high-level organization of your code, or how to run it, please put them in the write-up.
- You may use LaTex or Word to create your write-up, but **you must submit in PDF format**.
- For your convenience, you can simply edit the included .tex file. To do so,
  1. Edit the header information with your own personal information (otherwise, I won't know whose submission it is).
  2. Uncomment the `\begin{solution} ... \end{solution}` blocks and fill in your solution. (Note: you don't *need* to use the solution blocks if you find the formatting too restrictive. If you find that your solution spans multiple pages, you can break the solution block into multiple parts. Also, feel free to use `\begin{proof} ... \end{proof}` for proofs.)
- To add an image to this tex file, use the command `\includegraphics{filename}` (search the LaTeX documentation for additional arguments).

1 Decision Trees (continued)

1.1 KL-divergence, Information Gain, and Entropy

When we discussed learning decision trees in class, we chose the next attribute to split on by choosing the one with maximum information gain, which was defined in terms of entropy. To further our understanding of information gain, we will explore its connection to *KL-divergence*, an important concept in information theory and machine learning. For more on these concepts, refer to Section 1.6 in Bishop.

The KL-divergence from a distribution $p(x)$ to a distribution $q(x)$ can be thought of as a distance measure from P to Q:

$$KL(p||q) = -\sum p(x) \log_2 \frac{q(x)}{p(x)}$$

From an information theory perspective, the KL-divergence specifies the number of additional bits required on average to transmit values of x if the values are distributed with respect to $p(x)$ but we encode them assuming the distribution $q(x)$. If $p(x) = q(x)$, then $KL(p||q) = 0$. Otherwise, $KL(p||q) > 0$. The smaller the KL-divergence, the more similar the two distributions.

We can define information gain as the KL-divergence from the observed joint distribution of X and Y to the product of their observed marginals.

$$IG(x,y) \equiv KL\left(p(x,y)||p(x)p(y)\right) = -\sum_x \sum_y p(x,y) \log_2 \left(\frac{p(x)p(y)}{p(x,y)}\right)$$

When the information gain is high, it indicates that adding a split to the decision tree will give a more accurate model.

Exercise 1a Show that this definition of information gain is equivalent to the one given in class. That is, show that $IG(x,y) = H[x] - H[x|y] = H[y] - H[y|x]$, starting from the definition in terms of KL-divergence.

1.2 Shortest Tree

We know that a tree with lower complexity will tend to have better generalization properties. So one (rather simplistic) option to help avoid overfitting is to find the simplest tree that fits the data. This is a principle known as *Occam's Razor.* One simple way to define "simplest" is based on the *depth* of the tree. Specifically, the depth is the number of nodes along the longest root-to-leaf path. For example, the tree from part 1 would have depth 2. In this problem, we will be interested in learning the tree of least depth that fits the data.

Suppose the training examples are n-dimensional boolean vectors, where $n > 2$ is some constant integer. (For example (T, F, F, T, T) is a 5 dimensional boolean vector). We know that the ID3 decision tree learning algorithm is guaranteed to find a decision tree consistent[1] with any set of (not self-contradicting) training examples, but that doesn't necessarily mean it will find a short tree.

Exercise 1b For $n = 3$, does ID3 always find a consistent decision tree of depth ≤ 2 if one exists? If so, prove it. If not, provide a counterexample (a set of examples, similar to the table from the previous assignment, but with 3 variables), with an explanation.

Exercise 1c Propose your own learning algorithm that finds a shortest decision tree consistent with any set of training examples (your algorithm can have running time exponential in the depth of the **shortest** tree, but should not necessarily enumerate all possible trees of that depth). Give the pseudocode and a brief explanation. Use the following notation: Data[$X_i = T$] are the examples in Data with $X_i = T$, and similarly

[1] A "consistent" tree is one with zero training error.

for Data[$X_i = F$]. To start you off on the right foot, consider an algorithm (below) that uses a subroutine LearnShortest(Data,d) that returns a consistent tree of depth d if one exists. Then provide the pseudocode for LearnShortest(Data,d).

```
Algorithm:
Let d=0
Do
   Run LearnShortest(Data,d)
   If it returns a tree, output that tree
   Otherwise, let $d \leftarrow d + 1$
While($d \leq n$)
If we didn't return above, FAIL.
```

2 Linear Regression

The following questions deal with linear regression.

2.1 Ordinary Least Squares

As we have seen in class, **ordinary least squares** (OLS) is a method of estimating the parameters of a linear regression. The closed-form solution is given in matrix notation as,

$$w = (X^\top X)^{-1} X^\top y.$$

Exercise 2a What is the computational complexity of OLS? In other words, which step causes the greatest computational burden? Please justify your answer. If you make any assumptions about the data or the model, please explain why. Finally, give an asymptotic bound in big-O notation.

2.2 Dealing with "Bad" (i.e. Real-world) Data

One of the assumptions of linear regression is that the **covariate matrix** X (a.k.a. the **design matrix** or **independent variables**) must have **full column rank**. Remember from linear algebra that the column rank of a matrix is equal to the number of linearly independent columns. If a matrix does not have full rank, it is said to be **multicollinear**, meaning that at least one column is a linear combination of other columns.

Exercise 2b Why is this assumption crucial to linear regression? What will happen if X is multicollinear? Please justify your answer. (You may want to consider a simple example consisting of just 2 variables which are perfectly correlated.)

Exercise 2c Multicollinear (so-called "bad") data is actually quite common in real-world scenarios. A good linear regression algorithm must be robust to such data sources and still produce a model that fits the dependent variable y. Design an algorithm that accommodates multicollinear data and is still able to estimate a model. Express your algorithm in either pseudocode or just a high-level description. Explain why it works and discuss the computational complexity.

Exercise 2d Bravo! Your algorithm can produce a model that fits the dependent variable. But is it the only valid model? Is there a unique solution to the normal equations when the covariate matrix is multicollinear? Please justify your answer.

Exercise 2e Now it's time to put your money where your mouth is: you will implement your improved OLS algorithm in MATLAB and test it on some multicollinear data. Your algorithm should be implemented as a function with the following API:

```
function [w, mse] = ols(X, y),
```

where X is the covariate matrix, y is the dependent vector, w is the coefficient vector and mse is the mean squared error of the residuals,

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(w^\top x_i - y_i)^2.$$

Considerations:

- Though the X matrix will not include a column for the intercept, you are expected to estimate the intercept.
- You **must** follow the aforementioned API **exactly** or the grading script will not work and the TA will become very cross with you.

Test your function on the included file "concrete_data.mat". To load the variables X and y, use the command `load 'concrete_data.mat'`. Your submission should include the .m file that implements your function, as well as any dependencies.

3 Discriminative vs. Generative Classifiers

A common debate in machine learning has been over generative versus discriminative models for classification. In this question we will explore this issue, both theoretically and practically. We will consider naive Bayes (NB) and logistic regression classification algorithms.

Exercise 3a [Objective Functions] Briefly describe the functions that NB and logistic regression maximize.

3.1 Double counting the evidence

1. Consider the two class problem where class label $y \in \{T, F\}$ and each training example X has 2 binary attributes $X_1, X_2 \in \{T, F\}$.

   Exercise 3b How many parameters will you *need* to know/evaluate if you are to classify an example using the Naive Bayes classifier?

2. Let the class prior be $\Pr(Y = T) = 0.5$ and also let $\Pr(X_1 = T \mid Y = T) = 0.8$ and $\Pr(X_1 = F \mid Y = F) = 0.7$. , $\Pr(X_2 = T \mid Y = T) = 0.5$ and $\Pr(X_2 = F \mid Y = F) = 0.9$.

   (a) **Exercise 3c** Assume X_1 and X_2 are truly independent given Y. Write down the Naive Bayes decision rule given $X_1 = x_1$ and $X_2 = x_2$.

   (b) **Exercise 3d** Show that if Naive Bayes uses both attributes, X_1 and X_2, the error rate is 0.235. Is it better than using only a single attribute (X_1 or X_2)? Why?

   (c) **Exercise 3e** Now, suppose that we create a new attribute X_3, which is an exact copy of X_2. Are X_2 and X_3 conditionally independent given Y? What is the error rate of Naive Bayes now?

   (d) **Exercise 3f** Explain what is happening with Naive Bayes? Does Logistic Regression suffer from the same problem? Explain why.

   should we still assume conditionally independent
   or find a way to solve for dependent

4 Learning Curves of Naive Bayes and Logistic Regression

For this problem, you will implement both naive Bayes and logistic regression (using gradient descent). You will then train and test your classifiers using the UCI breast cancer dataset, which has been included with this assignment. We encourage you to use the included MATLAB function, `loaddata.m`, which will return the X matrix and y vector.

For the NB classifier, assume that $\Pr(x_i \mid y)$ – where x_i is a feature in the breast cancer data (that is, i is the number of column in the data file) and y is the label – is of the following multinomial distribution form:

For $x_i \in \{v_1, v_2, ..., v_n\}$,

$$\Pr(x_i = v_k \mid y = j) = \theta^i_{jk} \;\; s.t. \; \forall i, j : \sum_{k=1}^{n} \theta^i_{jk} = 1$$

where $0 \leq \theta_{jk} \leq 1$. It may be easier to think of this as a normalized histogram or as a multi-value extension of the Bernoulli.

For each algorithm:

1. **Exercise 4a** *Briefly* describe how you implement it by giving the pseudocode.
2. **Exercise 4b** Keeping a constant holdout of some percentage of the data (up to you to decide), train with varying sizes of training data drawn from the remainder. Then plot a learning curve: testing accuracy vs. the size of the training data.
3. **Exercise 4c** What conclusions can you draw from your experiments? Specifically, what can you say about the speed of convergence of the classifiers? Also, you may or may not find that the categorical Naive Bayes sometimes performs worse than Logistic Regression for small training sets and better than Logistic Regression for large training sets. Why might this happen?

Submit your MATLAB code with your assignment, but do *not* include your code in your write-up.