CPSC 340 Midterm (Fall 2015)

Name:

Student Number:

Please enter your information above, turn off cellphones, space yourselves out throughout the room, and wait until the official start of the exam to begin. You can raise your hand to ask a question, and please look up occasionally in case there are clarifications written on the projector (the time will also be written on the projector). You are welcome to (quietly) leave early if you finish early, and we will ask everyone to stop at 3:55.

The midterm consists of 5 questions, and they will all be equally weighted in the marking scheme. Note that some question have multiple parts, written as **(a)**, **(b)**, and in some cases **(c)**. All parts are equally weighted. Clearly mark where you are answering each part, and make sure to check that you answered all parts before handing in your midterm.

Good luck!

1 Cross-Validation

Consider a supervised classification problem where we have 50 training examples and 10 features, where the features are stored in an 50 by 10 matrix X and the labels are stored in a 50 by 1 vector y (you can assume that the examples are in a random order). As in the assignments, assume that you have a 'model' function that depends on a parameter 'k' with the following interface:

- model = train(X,y,k); % Train model on $\{X, y\}$ with parameter k
- yhat = predict(model,Xhat); % Predicst using the model on $Xhat$.

Assume that k can be either 1, 2, or 3.

Give pseudo-code describing how to choose k using 2-fold cross-validation.

Answer:

Split X and y into two parts so that:

- X_1 and y_1 contain the first 25 training examples.
- X_2 and y_2 contain training examples 26-50.

For each value of k from 1 to 3, perform the following:

- Fold 1:
  - Set Xtrain to $\{X_2\}$ and ytrain to $\{y_2\}$.
  - model = train(Xtrain,ytrain).
  - yhat = predict(model,X_1)
  - err1 = sum of elements where yhat does not equal y_1.
- Fold 2:
  - Set Xtrain to $\{X_1\}$ and ytrain to $\{y_1\}$.
  - model = train(Xtrain,ytrain).
  - yhat = predict(model,X_2)
  - err2 = sum of elements where yhat does not equal y_2.
- err = (err1 + err2)/50.
- Update the minimum if this is the lowest error found.

Return the k that resulted in the lowest error.

2 KNN and Decision Stumps

Consider the dataset below, which has 10 training examples and 2 features:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Suppose that you want to classify the following test example:

$$\hat{x} = \begin{bmatrix} 1 & 1 \end{bmatrix}.$$

(a) What class label would we assign to the test example if we used a k-nearest neighours classifier, with $k = 3$ and the Euclidean distance measure?

(b) Suppose we want to fit a decision stump to this dataset. What is the decision rule (based on a single variable) that minimizes the classification error? (Show your reasoning for the two possible splits.)

(c) Under the decision rule you estimated from part (b), what is the most likely label for the test example?

Answer:
(a) There are three training examples that have a distance of zero to the test example. Two out of three of them have the label '1', so we would classify the example as a '1'.

(b) If we split on the first variable then:

- If $x_1 = 0$ we should predict 1, which gives an error 0/3 times.
- If $x_1 = 1$ we should predict 0, which gives an error 3/7 times.

This gives an error rate of 3/10.

If we split on the second variable then:

- If $x_2 = 0$ then it doesn't matter we predict, we get an error of 3/6..
- If $x_2 = 1$ we should predict 1, which gives an error 1/4 times.

This gives an error rate of 4/10. Since this is higher than the error rate if we split on x_1 (and is actually the same as if didn't split), we should split on the first variable.

(c) If we split on x_1, then we should predict 0 (even though we expect this will be wrong 3/7 times).

(If they answered to split on x_2 in part (b), then they should predict 1, and we expect this to be wrong only 1/4 times.)

3 Less-Naive Bayes

In class, we talked about naive Bayes models. Given a set of features $\{x_1, x_2, x_3, \ldots, x_d\}$, naive Bayes approximates the conditional probability of a label y using

$$\begin{aligned} p(y|x_1, x_2, x_3, \ldots, x_d) &\propto p(y)p(x_1, x_2, x_3, \ldots, x_d|y) \\ &= p(y)p(x_1|y)p(x_2|x_1, y)p(x_3|x_1, x_2, y)\ldots p(x_d|x_1, x_2, x_3, \ldots, x_{d-1}, y) \\ &\approx p(y)p(x_1|y)p(x_2|y)p(x_3|y)\ldots p(x_d|y). \end{aligned}$$

Naive Bayes makes a strong independence assumption ("$\approx$"), and there are various ways to relax it. For example, consider a 'less-naive' Bayes model that depends on a parameter k and assumes that x_j given y is independent of all variables *except* the up to k largest values j where $j < i$. For example, if $k = 3$ then x_6 is conditionally independent of all other variables given y (as in the usual naive Bayes model) *and* given x_3, x_4, and x_5: $p(x_6|x_1, x_2, x_3, \ldots, x_d, y) = p(x_6|x_3, x_4, x_5, y)$.

Naive Bayes corresponds to $k = 0$, and we make weaker assumptions as k grows. As another example, if $k = 2$ then we use

$$\begin{aligned} p(y|x_1, x_2, x_3, \ldots, x_d) &\propto p(y)p(x_1, x_2, x_3, \ldots, x_d|y) \\ &= p(y)p(x_1|y)p(x_2|x_1, y)p(x_3|x_1, x_2, y)\ldots p(x_d|x_1, x_2, x_3, \ldots, x_{d-1}, y) \\ &\approx p(y)p(x_1|y)p(x_2|x_1, y)p(x_3|x_2, x_1, y)\ldots p(x_d|x_{d-1}, x_{d-2}, y). \end{aligned}$$

Instead of simply estimating conditionals like $p(x_5 = 1|y = 1)$, this will now involve estimating conditionals like $p(x_5 = 1|x_4 = 1, x_3 = 0, y = 1)$.

(a) What are the two parts of the fundamental trade-off in machine learning?

(b) For the less-naive Bayes model, how would the choice of k affect the two parts of the fundamental trade-off?

Answer:
(a)

1. How small can the model make the training error?
2. How well does the training error of of the model approximate the test error?

(b)

1. As k *increases*, you can model more complicated dependences so the *training error will go down*.
2. As k *increases*, the model you learn becomes more sensitive to your particular training set, so the *training error becomes a worse approximation of the test error*.

4 Runtime of K-Means

One of the outputs of the k-means algorithm is a set of cluster means μ_c. To find the cluster of a new data point $\hat{x}$, we can perform the following:

- For each cluster c, compute the Euclidean distance between $\hat{x}$ and the cluster mean μ_c.
- Assign $\hat{x}$ to the cluster c with the minimum distance.

Following our usual convention, we'll use:

1. d as the length of an $\hat{x}$ and μ_c.
2. k as the number of clusters.
3. t as the number of test examples.

If we use the above two steps to find the cluster of t test examples, what is the total cost in terms of d, k, and t?

Answer:
Computing a Euclidean distance between vectors of length d costs $O(d)$. We need to compute the distance between all t examples and all k clusters, so there are $O(tk)$ distances to compute. This gives a total cost of $O(tdk)$.

Given the k distances for a single example, we can search through them to find the minimum distance in $O(k)$. We need to do this for all t examples, giving a cost of $O(tk)$. But this is smaller than $O(tdk)$, so the total cost is still $O(tdk)$.

5 Regularized Linear Regression in 1D

Consider the problem of performing linear regression in 1-dimension where:

- We use the squared error as our loss function.
- We use an ℓ_2-regularizer with weight λ.

This gives us the following objective:

$$\operatorname*{argmin}_w \frac{1}{2}\sum_{i=1}^{n}\left[(y_i - wx_i)^2\right] + \frac{\lambda}{2}w^2.$$

(a) Compute the derivative of this objective function with respect to w.

(b) By equating the derivative of this objective (which is convex and quadratic) with 0, compute the solution of this problem in terms of the x_i, y_i, and λ (show your work).

Answer:
(a) Let

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}\left[(y_i - wx_i)^2\right] + \frac{\lambda}{2}w^2.$$

Then we have

$$f'(w) = \sum_{i=1}^{n}\left[-x_i(y - wx_i)\right] + \lambda w.$$

(b) Setting $f'(w) = 0$ gives

$$0 = \sum_{i=1}^{n}\left[-x_i(y - wx_i)\right] + \lambda w.$$

Moving all the terms depending on w to one side we get

$$\sum_{i=1}^{n}\left[x_i^2 w\right] + \lambda w = \sum_{i=1}^{n} x_i y_i,$$

and factoring out a w of the left side gives

$$w(\sum_{i=1}^{n}\left[x_i^2\right] + \lambda) = \sum_{i=1}^{n} x_i y_i.$$

Dividing both sides by the big factor on the left we get

$$w = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n}\left[x_i^2\right] + \lambda}.$$