CPSC 340 Midterm (Fall 2016)

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 2:55.

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)** and **(b)**. All parts are equally weighted. *Clearly show 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 **regression** problem where we have **60** training examples and 10 features, where the features are stored in a **60** by 10 matrix X and the labels are stored in a **60** by 1 vector y. The examples are given to you **sorted** by y_i values. 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 *3-fold* cross-validation.

Answer:

Randomly re-order the rows of X (and the corresponding elements of y) to remove the effect of the sorting. Split X and y into two parts so that:

- X_1 and y_1 contain the first 20 training examples.
- X_2 and y_2 contain training examples 21-40.
- X_3 and y_3 contain training examples 41-60.

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

- Fold 1:
  - Set Xtrain to $\{X_2, X_3\}$ and ytrain to $\{y_2, y_3\}$.
  - model = train(Xtrain,ytrain,k).
  - yhat = predict(model,X_1)
  - err1 = squared error between yhat and y_1 (or some other regression error).
- Fold 2:
  - Set Xtrain to $\{X_1, X_3\}$ and ytrain to $\{y_1, y_3\}$.
  - model = train(Xtrain,ytrain,k).
  - yhat = predict(model,X_2)
  - err2 = squared error between yhat and y_2 (or some other regression error).
- Fold 3:
  - Set Xtrain to $\{X_1, X_2\}$ and ytrain to $\{y_1, y_2\}$.
  - model = train(Xtrain,ytrain,k).
  - yhat = predict(model,X_3)
  - err3 = squared error between yhat and y_3 (or some other regression error).
- err = (err1 + err2 + err3)/60.
- Update the minimum if this is the lowest error found.

Return the k that resulted in the lowest error.

2 Predictions for Linear Regression and K-Means

Consider the dataset below, which has 5 training examples and 2 features:

$$X = \begin{bmatrix} 2 & 1 \\ 10 & 4 \\ 10 & 9 \\ 5 & 8 \\ 8 & 10 \end{bmatrix}, \quad y = \begin{bmatrix} 8.0 \\ 6.5 \\ 4.0 \\ 7.5 \\ 2.0 \end{bmatrix}.$$

Suppose that you have the following test example:

$$\hat{x} = \begin{bmatrix} 9 & 4 \end{bmatrix}.$$

(a) Suppose we use a linear regression model with coefficients given by

$$w = \begin{bmatrix} 2/3 \\ -1/4 \end{bmatrix}.$$

What prediction $\hat{y}$ would we make for the test example?

(b) Suppose we fit an unsupervised k-means model to this dataset and obtained the following means:

$$W = \begin{bmatrix} 9.0 & 9.5 \\ 3.0 & 5.0 \\ 9.5 & 4.0 \end{bmatrix}$$

Which cluster would the test example $\hat{x}$ get assigned to?

Answer:
(a)

$$w^T x_i = (2/3)(9) + (-1/4)(4) = 6 - 1 = 5.$$

(b) The closest row of W is clearly [9.5 4.0], so we would assign it cluster 3.

3 Fundamental Trade-Off for CNN and Laplace Smoothing

On the assignment, you implemented a **condensed nearest neighbours** (CNN) classifier as a faster alternative to k-nearest neighbours. In your CNN method, you went through the training set *once* and stored examples that were misclassified. Consider a variation of CNN where you go through the training examples k times, each time adding even more examples that are misclassified.

So $k = 0$ is the usual CNN method, $k = 1$ will go through the dataset again and have more training examples in the subset than $k = 0$, and $k = 2$ will have more training examples than $k = 1$.

(a) Briefly explain how k would affect the two parts of the fundamental trade-off, and why it would have this effect on each part.

In the most basic naive Bayes model, we estimate our conditional probabilities from the data using

$$p(x_{ij} = c|y_i = 1) = \frac{(\text{number of times } x_{ij} = c \text{ and } y_i = 1)}{(\text{number of times } y_i = 1)}.$$

This can cause a problem if we've never seen a training example where $x_{ij} = c$ and $y_i = 1$, since the probability will be 0. The standard solution to this problem is to estimate the conditional probabilites with **Laplace smoothing**,

$$p(x_{ij} = c|y_i = 1) = \frac{(\text{number of times } x_{ij} = c \text{ and } y_i = 1) + \beta}{(\text{number of times } y_i = 1) + \beta k},$$

where k is the number of values that c can take and β is a positive constant.

(b) Briefly explain how β would affect the two parts of the fundamental trade-off, and why it would have this effect on each part.

Answer:
(a)

1. As k increases, we are explicitly making sure that we have classified more examples correctly, so the training error will go down.
2. As k increases, our model is getting more complicated and more dependent on our specific training set, so the training error will become a worse approximation of the test error.

(b)

1. As β increases, the conditional probabilities move towards $1/k$ and we start ignoring the features, so we would expect the training error to go up.
2. As β increases, the conditional probabilities become less sensitive to our particular dataset, so we would expect the training error to be a better approximation of the test error.

4 Runtime of Random Trees

In the assignment, you implemented a variant on decision trees called a *random tree*. In a *random tree*, each time we fit a decision stump we only consider $\sqrt{d}$ random features as candidates for the split variable. Following our usual convention, we'll use:

- n as the number of training examples.
- d as the number of features..
- m as the depth of the decision tree.
- t as the number of test examples.

(a) What is the cost of fitting a random tree of depth m to our training data in terms of the above four quantities? (Briefly justify your answer.)

(b) What is the cost of prediction on t test examples using a random tree in terms of the above four quantities? (Briefly justify your answer.)

Answer:
(a) Our normal cost for fitting a decision tree is $O(mnd \log n)$, and the factor of d comes from searching through all features for each split. If we only search $\sqrt{d}$ features this will be reduced to $O(mn\sqrt{d} \log n)$.

(b) For each training example t, we only do an $O(1)$ operation at each level of the decision tree to decide whether we satisfy the rule. This gives a total cost of $O(tm)$.

5 Weighted Regression with Weighted Regularization

Consider a variation on L2-regularized least squares where we have a **positive** weight z_i for each training example i and a positive variable-specific regularization weight λ_j for each feature j. In this case our objective function can be written as

$$f(w) = \frac{1}{2}(Xw - y)^T Z(Xw - y) + \frac{1}{2}w^T \Lambda w,$$

where Z is an n by n diagonal matrix with the z_i values along its diagonal and Λ is a d by d matrix with the λ_j values along its diagonal.

(a) Write finding a minimizer w of this (convex) function as a system of linear equations.

Consider a variant where to gain robustness we use the absolute error and where we also regularize by a weighted sum of absolute values (instead of squaring the w_j),

$$f(w) = \sum_{i=1}^{n} z_i |w^T x_i - y_i| + \sum_{j=1}^{d} \lambda_j |w_j|.$$

(b) Express this function in terms of vectors, matrices, and norms (there should be no summations in the final answer, and you can use Z and Λ as defined in the previous part).

(a) Expanding we have

$$f(w) = \frac{1}{2}w^T X^T Z X w + w^T X^T Z y + \frac{1}{2}y^T Z y + \frac{1}{2}w^T \Lambda w.$$

Taking the gradient we get

$$\nabla f(w) = X^T Z X w + X^T Z y + \Lambda w.$$

Equating with zero we obtain the linear system

$$(X^T Z X + \Lambda)w = X^T Z y.$$

(b) Following our standard approach we'll get

$$\|Z(Xw - y)\|_1 + \|\Lambda w\|_1,$$

It would also be ok to define vectors z and v and use element-wise products $\circ$ (or whatever notation as long as it's defined)

$$\|z \circ (Xw - y)\|_1 + \|v \circ w\|_1.$$