CPSC 340 Final (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 remaining will also be written on the projector). You are welcome to (quietly) leave early if you finish before the final 5 minutes of the exam, but please *do not leave in the last 5 minutes* as it is very distracting to those who want to work up to the last minute.

The final consists of 10 questions, and they will all be equally weighted in the marking scheme. Note that some question have multiple parts, written as **(a)**, **(b)**, and **(c)**. *Clearly mark* where you are answering each part, *show your work/reasoning for each part*, and make sure to check that you answered all parts before handing in your final.

May the force be with you!

1 Training/Validation/Testing

You are asked by a client to build a system that solves a **regression** problem. They give you 3000 training examples and a set of 100 features for each example. You have been assured that the examples have been generated in way that makes them IID, and the examples have been given to you **sorted** based on the values of the first feature. The client not only wants an accurate model, but they **want an unbiased estimate of the accuracy** of the final model.

Assume that the data is stored in a 3000 by 100 matrix X, and the targets are stored in a 3000 by 1 vector y. Assume that you have a 'model' function that depends on **two parameters** k_1 and k_2 with the following interface:

- model = train(X,y,k_1,k_2); % Train model on $\{X, y\}$ with hyper-parameters k_1 and k_2.
- yhat = predict(model,Xhat); % Predicts using the model on $Xhat$.

Assume that the two parameters k_1 and k_2 can take any integer value from 1 to 5.

Give pseudo-code for a training/validation/testing procedure that chooses good values of k_1 and k_2, and then reports an unbiased estimate of the accuracy of the final model.

Answer:

Since the examples were sorted, you first need to**randomize the order of the examples**. One way to do this is to place the rows of X in a random order, re-ordering y using the same order.

Next **split the examples into a training, validation, and testing set**. For example, you could use the first 1000 random examples as the training set (Xtrain, ytrain), the next 1000 for the validation set (Xvalidate, yvalidate), and the final 1000 as the testing set (Xtest, ytest). If you did not place the examples in a random order, you could choose these three sets randomly from the examples, but *ensuring that there is no overlap* between the three sets.

For each value of k_1 from 1 **to** 5 **and k_2 from** 1 **to** 5, perform the following:

- model = train(Xtrain,ytrain,k_1,k_2).
- yhat = predict(model,Xvalidate).
- err(k_1,k_2)= squared difference between yhat and yvalidate (or some other **regression loss**).

Set (k_1, k_2) **to be the minimum of err(k_1,k_2)** across all k_1 and k_2.
(It's ok if they used cross-validation instead of a training/validation set.)

To **report the estimated accuracy of the final model**:

- yhat = predict(model,Xtest).
- accuracy = squared difference between yhat and ytest (or some other **regression loss**).

2 Data Standardization

Consider the following Matlab code for loading a dataset, standardizing the features, training a model, and making predictions on a test set:

```
% Load data {X,y,Xtest}
load data.mat
[n,d] = size(X);
[t,~] = size(Xtest);

% Standardize features
mu = zeros(d,1);
sigma = zeros(d,1);
for j = 1:d
        mu(j) = mean(X(:,j));
        sigma(j) = std(X(:,j));
        X(:,j) = (X(:,j) - mu(j))/sigma(j);
end

% Fit model
model = fit(X,y);

% Make predictions on test set
ytest = model.predict(model,Xtest);
```

Unfortunately, there is a major error in the code above. This error causes the predictions *ytest* to be terrible, even though the training error is low and you don't think that the model has overfit.

(a) What is the major error in this code?

(b) Give the code needed to fix the error, and where it needs to be located in the above script.

Answer:
The code isn't standardizing the test features Xtest. At any location in the code before the final line, you would need to do something like:

```
for j = 1:d
        Xtest(:,j) = (Xtest(:,j) - mu(j))/sigma(j);
end
```

Note that we should use the mu and sigma from the training data, not re-compute them on the test data.

3 Terminlogy and Key Properties

Consider the following list of tools we discussed in the course:

1. Median.
2. Scatterplot.
3. Cross-validation.
4. KNN.
5. Density-based clustering.
6. Polynomial basis.
7. L1-regularization.
8. Softmax probability.
9. Multi-dimensional scaling (MDS).
10. PageRank.

Match each one of the above items to one of the key properties below:

1. Assigning a score to a node in a graph.
2. Choosing regularization parameter.
3. Feature selection.
4. Finding non-convex clusters.
5. Multi-class loss function.
6. Non-linear regression.
7. Non-parametric classifier.
8. Non-parametric visualization.
9. Robust alternative to mean.
10. Visualizing dependency between 2 variables.

Answer:

1. Assigning a score to a node in a graph: PageRank.
2. Choosing regularization parameter: Cross-Validation.
3. Feature selection: L1-regularization.
4. Finding non-convex clusters: DBSCAN.
5. Multi-class loss function: Softmax probability.
6. Non-linear regression: Polynomial basis.
7. Non-parametric classifier: KNN.
8. Non-parametric visualization: MDS.
9. Robust alternative to mean: median.
10. Visualizing dependency between 2 variables: scatterplot.

4 Softmax and Decision Stumps

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

$$X = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \\ 3 \\ 3 \\ 3 \\ 3 \end{bmatrix},$$

and consider a single test example:

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

(a) Suppose we fit a multi-class linear classifier using the softmax loss, and we obtain the following weight matrix:

$$W = \begin{bmatrix} -1 & +2 & 0 \\ +2 & 0 & +3 \end{bmatrix}$$

Under this model, what class label would we assign to the test example? (Show your work.)

(b) What is the cost of prediction on t test examples with a softmax model? State your result in $O()$ notation in terms of the number of training examples n, the number of test examples t, the number of features d, and the number of classes k?

(c) What is the decision stump that minimizes the classification error?

Answer:
(a) This model bases its decision of maximizing the inner-product, $w_c^T\hat{x}$.

For class 1 we have
$$w_1^T\hat{x} = (-2)1 + (2)1 = 0.$$
For class 2 we have
$$w_2^T\hat{x} = (+2)1 + (0)1 = 2.$$
For class 3 we have
$$w_3^T\hat{x} = (0)1 + (3)1 = 3.$$
So this model would predict '3'.

(b) For an individual test example $\hat{x}_i$, we need compute the inner product $w_c^T\hat{x}_i$ for each value of c. Each of these inner products costs $O(d)$, so to compute them for c classes on t examples gives $O(tdc)$. Taking the maximum among these inner products only costs $O(c)$ per example, so the total time is $O(tdc)$.

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

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

This gives an error rate of 4/10.

If we split on the second variable then:

- If $x_2 = 0$ then we should predict 3, which gives an error 3/6 times.
- If $x_2 = 1$ we should predict 2, which gives an error 2/4 times.

This gives an error rate of 5/10. Since this is higher than the error rate if we split on x_1 (and we would get 6/10 wrong with the baseline rule), we should split on x_1:

If $x_1 = 0$ then predict 1 and otherwise predict 3.

5 Parametric vs. Non-Parametric

Next to each of the methods below, state whether it is *parametric* or *non-parametric*:

1. Histogram with 10 bins.
2. Decision tree learner of depth $\sqrt{n}$.
3. 10-nearest neighbours.
4. K-means with $\sqrt{d}$ means.
5. Density-based clustering.
6. Linear regression with polynomial basis of degree n.
7. Softmax multi-class classification (linear basis).
8. Robust PCA.
9. ISOMAP (with MDS to compute z_i).
10. Deep neural network.

Anwer:

1. Histogram with 10 bins: parametric.
2. Decision tree learner of depth $\sqrt{n}$: non-parametric.
3. 10-nearest neighbours: non-parametric.
4. K-means with $\sqrt{d}$ means: parametric.
5. Density-based clustering: non-parametric.
6. Linear regression with polynomial basis of degree n: non-parametric.
7. Softmax multi-class classification (linear basis): parametric.
8. Robust PCA: parametric.
9. ISOMAP (with MDS to compute z_i): non-parametric.
10. Neural networks: parametric.

6 Linear Regression Variations

The *tilted* least squares objective function has the form

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \lambda \sum_{j=1}^{d} w_j v_j,$$

for a vector v with real-valued elements v_j and where λ can be any real number.

(a) Show that this objective function is convex.

(b) Write the minimizer of the above objective function as the solution of a linear system.

(c) Consider the L∞-regularized weighted absolute loss

$$f(w) = \sum_{i=1}^{n} z_i |w^T x_i - y_i| + \lambda \max_j |w_j|,$$

where the max is taken over j in $\{1, 2, \ldots, d\}$. Write this objective in matrix notation.

Let's first write this in matrix notation:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda w^T v.$$

(a) We know that squared norms are convex, so the first term is convex because it is a convex function composed with an affine function $Xw - y$. The second term is linear so it's convex, and the sum of convex functions is convex.

(b) We have

$$\nabla f(w) = X^T(Xw - y) + \lambda v.$$

Equating this with zero we get that w solves

$$(X^T X)w = X^T y + \lambda v.$$

(c)

$$f(w) = \|Z(Xw - y)\|_1 + \lambda \|w\|_\infty.$$

7 MLE for Survival Analysis with Censoring

Suppose we have a set of training examples (x_i, y_i, v_i) where

- y_i is a positive number giving the time that user i quit using Microsoft Windows forever.
- $v_i = 1$ if user i has quit using Microsoft Windows and $v_i = 0$ if they are still using it.

If we assume that the "instantaneous rate" that users quit using Windows depends on the features x_i but not the total time that they've been using Windows, then a standard *censored survival analysis* likelihood is given by

$$p(y_i, v_i | x_i, w) = \exp(v_i w^T x_i)\exp(-y_i \exp(w^T x_i)).$$

(a) If a dataset consists of n IID examples (x_i, y_i, v_i), show how finding the maximum likelihood estimate (MLE) for w corresponds to minimizing an additive loss function,

$$f(w) = \sum_{i=1}^{n} g(y_i, v_i, w, x_i),$$

and derive the form of the loss function f (simplifying as much as possible).

(b) If the largest value of y_i in the training set is k, what is the cost of evaluating this objective function in terms of n, d, and k?

(c) Show that the function derived in part (a) is convex.

Answer:

(a) By writing the negative log-likelihood we have

$$f(w) = \sum_{i=1}^{n} [-v_i w^T x_i + y_i \exp(w^T x_i)].$$

(b) The dominant cost is the n inner products of size d, so the cost is $O(nd)$, there is no dependence on k.

(c) The first set of terms is linear so they are convex. The second set of terms is a convex function exp() composed with a linear function so they are also convex. So f is convex because it's a sum of convex functions.

8 PCA and Regularization

Someone gives you a code for PCA, and when you run the code it returns a matrix W whose rows have a norm of 1 and are orthogonal. However, when you run the code again it returns $-W$.

(a) Why is this not necessarily a bug?

(b) Consider a regularized objective for filling in missing entries x_{ij} of a matrix given a set of examples x_{ij} values (denoted by R) from the matrix.

$$f(Z, W) = \frac{1}{2} \sum_{(i,j) \in R} (w_j^T z_i - x_{ij})^2 + \frac{\lambda_W}{2} \|W\|_F^2,$$

with $\lambda_W > 0$.

What is the effect of λ_W on the fundamental trade-off?

(c) Consider the following variation on the objective function,

$$f(Z, W) = \frac{1}{2} \sum_{(i,j) \in R} (w_j^T z_i - x_{ij})^2 + \frac{\lambda_W}{2} \|W\|_F^2 + \frac{\lambda_Z}{2} \|Z\|_F^2,$$

for $\lambda_W > 0$ and $\lambda_Z > 0$. Would the answer to (b) change?

(d) Consider another variation on the objective function,

$$f(Z, W) = \frac{1}{2} \sum_{(i,j) \in R} (w_j^T z_i - x_{ij})^2 + \frac{\lambda_W}{2} \|W\|_0,$$

for $\lambda_W > 0$. Would the answer to (b) change?

Answer:
(a) If we let ZW denote the predictions under the optimal PCA model, then we obtain equivalent predictions with $-W$ if we also use $-Z$. So multiplying W by -1 gives an equivalent model.

(b) The regularization parameter λ_W doesn't affect the fundamental trade-off, since we can increase the norm of Z to compensate for any decrease in the norm of W.

(c) With the norm of Z restricted, increasing λ_W will decrease our training our training error but make the training error a better approximation of test error.

(d) As λ_W becomes large the regularizer outweights the loss and the elements of W become sparse. With elements of W that are exactly zero we can no longer increase Z to compensate for the lost information. This means as the λ_W increases we will decrease our training error but make the training error a better approximation of test error.

9 Grouped PageRank

In the last assignment we implemented the PageRank algorithm using a random walk strategy. Consider a variation of the webpage ranking problem where we have *groups* of webpages, and instead of ranking individual webpages to want to give *rankings to the groups*.

As an example with 6 webpages, group 1 could consist of webpages $\{1, 2, 5\}$ and gets a rank of 0.20, group 2 could consist of $\{3, 4\}$ and get a rank of 0.70, and group 3 could consist of $\{6, \}$ and get a rank of 0.10. The groups might represent webpages that come from the same server, that have the same theme, or that have some other logical organization.

Write pseudo-code for a random walk algorithm that computes a PageRank-like score for groups of webpages.

Remember that links are between individual webpages, and if you need it please use the following notation:

- Use t to denote the "timestep" of the random walk.
- Use i to refer to an individual webpage.
- Use N_i to denote the outlinks of webpage i, and $|N_i|$ to denote the number of outlinks.
- Use g_i to refer to the group of webpage i (assume each webpage is assigned to exacty one group).
- Use c_i to denote a count of the number of times we've visited page i.
- Use k_g to denote a count of the number of times we've visited group g.

Define any extra notation that you use.

Answer:
There are many possible answers here. One logical one is to just compute the normal PageRank and to give a score to the group just sum up the score for the group. Below is pseudo-code for this strategy:

- Initialize a vector c to 0.
- At $t = 0$, choose a random webpage.
- At $t > 0$, augment c_g by one and follow a random link from the current webpage i.
- After some very long time, return c_g/t as the ranking for group g.

A reasonable variation would include a parameter $\alpha > 0$ giving the probability of jumping to a random webpage. Alternately, you could just run the normal PageRank algorithm can sum up the c_i values across each group.

Another reasonable strategy would be to aggregate all the webpages in the same group together, and form a graph on the groups. Then you could just run the standard PageRank algorithm on this graph.

10 Neural Networks

Consider a two-layer neural network model with L2-regularization,

$$f(W^{(1)}, W^{(2)}, w) = \frac{1}{2}\sum_{i=1}^{n}(w^T h(W^{(2)} h(W^{(1)} x_i)) - y_i)^2 + \frac{\lambda}{2}\|w\|^2 + \frac{\lambda_1}{2}\|W^{(1)}\|_F^2 + \frac{\lambda_2}{2}\|W^{(2)}\|_F^2,$$

where W^1 is k_1 by d, W^2 is k_2 by k_1, and w is k_2 by 1.

(a) Describe how the following factors affect the two parts of the fundamental trade-off:

1. **The parameter k.**
2. **The parmaeter λ.**
3. **Increasing the depth (number of hidden layers) from 2 to a larger number m**

(b) Given only a training set, how can we choose values of the above parameters?

Answer:
(a) Increasing k or m will decrease the training error, while making the training error a worse approximation of the test error. The parameter λ has the opposite effect.

(b) Cross-validation or validation set.