CPSC 340 2021S Final Exam (Practice)

IMPORTANT: Download the .tex source file here!!!!

`https://www.cs.ubc.ca/~nhgk/courses/cpsc340s21/assignments/practice_final_exam.tex`

Instructions

As with the assignments, please indicate your name and 8-digit UBC student ID number in your solution.

- You have 24 hours to complete this exam. You can submit anytime, however many times during the exam window.
- You may submit a handwritten solution instead of a typeset document. If you do so, please clearly mark your answers with corresponding question numbers.
- The exam is open book, meaning you are allowed to consult course materials, the internet, etc.
- You may NOT communicate with anyone else (other than the instructor) in any way during the exam. This includes posting anything on the internet (e.g. **Discord**, a message board, chat, tutoring service, etc.) during the exam. UBC has a strict policy on academic misconduct, along with disciplinary measures, which I can attest are not fun for anyone involved.
- You may NOT copy/paste solutions from anywhere. If you consulted any online resources and wish to cite this, feel free to drop a link in your exam submission to be safe.
- Announcements or clarifications will be made on this Piazza thread:
  `https://piazza.com/class/koaczotaebx55t?cid=194`.
  Please check it before the exam.
- If you have a question, make a private post on Piazza.
- Do not post or communicate about the exam until at least 1 hour after the scheduled end time.

The submission format for this portion of the midterm exam is identical to the submission format for the homework assignments, except two things: (1) you cannot work with a partner, and (2) you can submit handwritten solutions.

Please follow the official submission instructions. In short, submit to Gradescope and match the questions to pages when prompted by Gradescope. We reserve the rights to deduct 5 points off of your submission if you do not follow the submission instructions.

1 Very-Short Answer Questions

Rubric: {points:22}

Answer the following questions in 1-2 brief sentences.

1. Recall k-means clustering. I claim that k-means clustering is an instance of latent factor models, which maps the $n \times d$ matrix X to an $n \times k$ matrix Z. Explain what each row of Z corresponds to.

   Answer: Each row z_i is a one-hot encoding of the cluster label $\hat{y}_i$.

2. Recall the fundamental trade-off and the L2-regularized PCA for collaborative filtering, where we used only the available values in X to build Z and W. Describe the effect of increasing λ_W and λ_Z, the strengths of regularizers on the Frobenius norms of W and Z respectively, on (1) reconstruction error on available values and (2) "prediction error" for missing values. (Only comment on increasing λ_W and λ_Z at the same time.)

   Answer: As λ_W and λ_Z increase, reconstruction error will increase but prediction error will decrease.

3. Recall the IID assumption. Describe one reason why the IID assumption might be violated.

   Answer: The data might be sampled from a sequence, e.g. a time series or patches of an image.

4. Recall L1-regularization. Describe the effect of increasing λ, the strength of L1-regularization, on the memory use of a linear model.

   Answer: As λ increases, the model will use less memory (because the weights have more 0s).

5. Recall the hinge loss for optimizing support vector machines (SVMs). Describe one reason why minimizing the hinge loss corresponds to implicitly reducing the 0-1 loss.

   Answer: The hinge loss is an upper convex envelope of the 0-1 loss, so minimizing the hinge loss means minimizing the upper bound of the 0-1 loss.

6. Recall multi-class linear classifiers. Describe one key difference between the one-vs-all classifier and a multi-class classifier (e.g. multi-class SVM, multi-class logistic regression).

   Answer: One-vs-all classifier's loss function is completely separable in the rows of W, and there are no constraints taking into account the linear model weights for other classes.

7. Recall feature engineering. Describe one disadvantage of using an overly-complex feature.

   Answer: An overly-complex feature may lead to overfitting.

8. Recall the kernel trick. Describe one scenario where using the kernel trick may not be a good idea.

   Answer: When n is significantly larger than the number of features after the change-of-basis, kernel methods will be slow. Also, when the data is completely linear in the original feature space, kernel methods will introduce unnecessary complexity. Another possibility is when we don't want to use L2-regularization.

9. Recall linear classifiers. Describe one strategy for forming a non-convex decision boundary using a linear classifier.

   Answer: Use change-of-basis or kernel trick and fit a linear model after.

10. Recall stochastic gradient descent. Describe two hyper-parameters associated with a stochastic gradient descent strategy.

    Answer: Batch size and step size.

11. Recall artificial neural networks. Describe why we need a non-linear activation function after a matrix multiplication for each layer of a neural network.

    Answer: Without non-linearity, the network will collapse into a linear model.

2 Decision Trees

Rubric: {points:10}

Suppose I have the following training data with continuous features and categorical labels:

$$X = \begin{bmatrix} 1 & 3 & 7 & 3 \\ 3 & 2 & 1 & 2 \\ 2 & 5 & 2 & 4 \\ 4 & 1 & 1 & 2 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

1. Using the greater-than-or-equal-to ($\geq$) splitting rule on the first feature ($j = 1$), create a decision stump that maximizes classification accuracy. What is the splitting rule? What are the resulting y_{yes} and y_{no}?

   Answer: Splitting rule: $x_{i1} \geq 3$. $y_{\text{yes}} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$, $y_{\text{no}} = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$

2. Remove the incorrect option in each parenthesis so that the resulting sentence is true:

   Answer: The information gain of this stump is (small / **large**) because the labels in leaf nodes after splitting are more (**homogeneous** / heterogeneous) than before.

3 Designing Objective Functions

Rubric: {points:15}

Using vector and matrix notation covered in the course, write down the objective function f for the following items, minimizing which will achieve each described behaviour. For example,

- An ordinary least squares linear regression

  Answer:

$$f(w) = \frac{1}{2}||Xw - y||^2$$

1. An ordinary least squares linear regression with a unique solution.

   Answer:

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

   ($\frac{1}{2}$ is optional)

2. A linear regression model that is extremely sensitive to outliers whose weights are sparse and small.

   Answer:

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

3. A linear binary classifier that maximizes the Bernoulli likelihood $p(y_i \mid x_i, w) = \frac{1}{1+\exp(-y_i w^T x_i)}$. Assume $y_i \in \{+1, -1\}$.

   Answer:

$$f(w) = \sum_{i=1}^{n} \log\left(1 + \exp(-y_i w^T x_i)\right)$$

4. An L1-regularized matrix factorization model for collaborative filtering, minimizing the reconstruction error of available entries. The resulting Z and W will be sparse. Assume there is a set R that contains all (i, j) tuples corresponding to the indices of the available values of X.

   Answer:

$$f(Z, W) = \frac{1}{2} \sum_{i,j \in R} \left(\langle w^j, z_i \rangle - x_{ij}\right)^2 + \lambda_Z \sum_{c=1}^{k} \sum_{j=1}^{d} |w_{cj}| + \lambda_W \sum_{i=1}^{n} \sum_{c=1}^{k} |z_{ic}|$$

   ($\frac{1}{2}$ is optional)

5. A single-layer neural network model that uses an activation function $h(z)$ and performs robust regression. The encoder uses a $k \times d$ matrix W, and the predictor uses a $k \times 1$ vector v. k is the size of the learned features. Ignore biases.

   Answer:

$$f(v, W) = \sum_{i=1}^{n} |v^T h(W x_i) - y_i|$$

Hint: you can write your solutions with the $\sum$ notation instead of expressing everything as matrix-vector products.

4 MAP Estimation

Rubric: {points:10}

Consider using the Gaussian likelihood with $\sigma = 1$ and example-specific bias μ_i, and using the Laplacian prior with per-variable bias ν_j and scale $b_j > 0$:

$$p(y_i \mid x_i, w) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(w^T x_i - y_i - \mu_i)^2}{2}\right), \qquad p(w_j) = \frac{1}{2b_j} \exp\left(-\frac{|w_j - \nu_j|}{b_j}\right)$$

To answer the following questions, you may use these quantities:

- m, an $n \times 1$ vector whose ith entry corresponds to μ_i, the example-specific bias value.
- ν, a $d \times 1$ vector whose jth entry corresponds to ν_j, the per-variable bias value.
- B, a $d \times d$ diagonal matrix whose jth diagonal entry corresponds to b_j, the per-variable scale value.

1. Under the IID assumption, find the objective function $f(w)$ whose minimizer corresponds to the maximum a posteriori (MAP) estimate of the resulting posterior distribution.

Answer: Taking the negative log likelihood,

$$\begin{aligned}
-\log p(y_i \mid x_i, w) &= -\sum_{i=1}^{n}\left[\log\left(\frac{1}{\sqrt{2\pi}}\right) - \frac{(w^T x_i - y_i - \mu_i)^2}{2}\right] \\
&\approx \frac{1}{2}\sum_{i=1}^{n}\left(w^T x_i - y_i - \mu_i\right)^2 \\
&= \frac{1}{2}||Xw - y - m||^2
\end{aligned}$$

Taking the negative log prior,

$$\begin{aligned}
-\log p(w_j) &= -\sum_{j=1}^{d}\left[\log\left(\frac{1}{2b_j}\right) - \frac{|w_j - \nu_j|}{b_j}\right] \\
&\approx \sum_{j=1}^{d}\frac{|w_j - \nu_j|}{b_j} \\
&= ||B^{-1}(w - \nu)||_1
\end{aligned}$$

Therefore, the objective function is:

$$f(w) = \frac{1}{2}||Xw - y - m||^2 + ||B^{-1}(w - \nu)||_1$$

2. Describe the effect of increasing the per-variable scale b_j on the two parts of the fundamental trade-off.

   Answer: Increasing b_j values together will decrease the amount of regularization. Hence, it will decrease training error but increase approximation error.

5 Stochastic Gradient

Rubric: {points:10}

Suppose I have the following data:

$$X = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \\ 5 & 6 \\ 6 & 5 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 2 \end{bmatrix}$$

Suppose that I am training an L2-regularized least squares model, minimizing the following objective function

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

Suppose I am training my model with $\lambda = 1$, using stochastic gradient with the step size $\alpha^t = \frac{1}{t^2}$. At the current iteration t, I obtain the current solution:

$$w^t = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Answer the following questions:

1. Suppose that $t = 2$, and $i = 1$ (first row) was randomly selected for computing the gradient. What is the value of w^{t+1} after performing a stochastic gradient step?

   Answer: The gradient for f is:

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

   Using just one example x_1, we get

$$\nabla f_1(w) = x_1 w^T x_1 - x_1 y_1 + \lambda w = x_1 (w^T x_1 - y_i) + \lambda w.$$

   Using w^t, we compute

$$\nabla f_1(w^t) = x_1(\langle w^t, x_1 \rangle - y_i) + \lambda w^t. = \begin{bmatrix} 1 \\ 2 \end{bmatrix} (2 - 1) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

   Finally, the stochastic gradient step is

$$\begin{aligned} w^{t+1} &= w^t - \alpha^t \nabla f_1(w) \\ &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 1 \\ 3 \end{bmatrix} \\ &= \begin{bmatrix} -1/4 \\ 1/4 \end{bmatrix} \end{aligned}$$

2. Suppose that the stochastic gradient iterations continue until $t = 10^9$ (1 billion). Is it likely that $\nabla f(w)$, the gradient computed on all of the n examples, is very close to 0 then?

   Answer: No, the step size $\alpha^t = \frac{1}{t^2}$ is going to converge at a non-stationary point because it decays too fast.

6 Logistic Regression

Rubric: {points:15}

Recall logistic regression. Suppose I have a labeled binary classification dataset whose labels are +1 and -1. Suppose I have obtained a solution

$$w = \begin{bmatrix} 1 \\ 2 \\ 2 \\ -0.5 \end{bmatrix}$$

for logistic regression. Suppose I have a test example

$$\tilde{x}_i = \begin{bmatrix} -2 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

Answer the following questions:

1. What is the predicted label for $\tilde{x}_i$?

   Answer:
   $$w^T \tilde{x}_i = -0.5.$$
   The predicted label $\hat{y}_i = -1$.

2. What is the probability that this test example is a positive example?

   Answer:
   $$p(y_i = +1|\tilde{x}_i, w) = \frac{1}{1+\exp(-w^T\tilde{x}_i)} = \frac{1}{1+\exp(0.5)} \approx 0.378$$

3. What is the probability that this test example is a negative example?

   Answer:
   $$p(y_i = -1|\tilde{x}_i, w) = \frac{1}{1+\exp(w^T\tilde{x}_i)} = \frac{1}{1+\exp(-0.5)} \approx 0.622$$

7 Convex Functions

Rubric: {points:15}

Consider the L1-regularized brittle regression objective:

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

1. Show that this function is convex (assume $\lambda > 0$).

   Answer: Left term: linear mapping $Xw - y$ is convex. L_∞ norm preserves convexity. Right term: L_1-norm preserves convexity. Non-negative scaling preserves convexity. Adding convex terms preserves convexity.

2. Re-write the first term $||Xw - y||_\infty$ with an appropriate approximation so a gradient can be computed.

   Answer:
   $$||Xw - y||_\infty \approx \log\left(\sum_{i=1}^{n} \exp(|w^T x_i - y_i|)\right)$$

3. Suppose I don't need to control the amount of sparsity in w and don't want to use a smooth approximation for the L1-penalty. Describe one way to achieve sparsity for this model.

   Answer: Use projected gradient.

8 Runtime Complexity Analysis

Rubric: {points: 10}

Answer the following questions using the big-O notation.

1. Consider training a polynomial regression model of degree p on a training data where $d = 1$. If we use gradient descent for t iterations, what is the running time required for training this model? Your answer may or may not depend on n, t, d, and/or p.

   Answer: $O(npt)$. With $d = 1$, we have p features after transformation, which takes $O(np)$. Computing the gradient based on the $n \times p$ matrix Z takes $O(np)$. We compute the gradient for t iterations.

2. Consider training a deep, fully-connected neural network of m layers on the training data consisting of X, an $n \times d$ matrix and y, an $n \times 1$ vector. Each layer outputs k features, including the final layer of the encoder. What is the running time for generating the predictions for the training data? Your answer may or may not depend on m, n, d, and/or k.

   Answer: Consider generating the prediction for just one example. The first layer's matrix multiplication is between a $k \times d$ matrix and a $d \times 1$ vector, hence $O(dk)$. Second layer is between a $k \times k$ matrix and a $k \times 1$ vector, accruing $O(k^2)$ time. The element-wise non-linear activation will take $O(k)$, which is smaller than $O(k^2)$ so we will suppress this term. Starting there, we have a sequence of $m - 1$ matrix-vector multiplications with a $k \times k$ matrix, hence $O(mk^2)$ time. Finally, the predictor computes the dot product between $k \times 1$ vectors, taking $O(k)$ time. The total time for just one example is $O(dk + mk^2 + k) = O(dk + mk^2)$. Then for n examples, total running time is $O(ndk + nmk^2)$.

9 Memory Complexity Analysis

Rubric: {points: 10}

Answer the following questions using the big-O notation.

1. Consider fitting a multi-class logistic regression model to a labeled dataset with an $n \times d$ feature matrix X and an $n \times 1$ label vector y, consisting of k classes. If we use Gaussian RBF basis, what is the memory complexity required to store the parameters of this multi-class logistic model? Your answer may or may not depend on n, d, and/or k.

   Answer: $O(nd + nk)$. Gaussian RBF basis creates an $n \times n$ matrix Z after feature transforms, and the weights for the multi-class linear classifier W is an $k \times n$ matrix. For predicting the labels of test examples, we need to store both X and W, because $\tilde{Z}$ will depend on X and $\hat{y}$ will be computed based on $\tilde{Z}W^T$.

2. Consider fitting a principal component analysis (PCA) model to an $n \times d$ matrix X to learn the matrix factorization $X = ZW$ with $k = d$. What is the total memory complexity required for storing the result, including both Z and W? Your answer may or may not depend on n, d, or k.

   Answer: $O(nd + d^2)$. With $k = d$, Z is an $n \times d$ matrix and W is a $d \times d$ matrix.

10 Objective Function: "Sum"-Rule Multi-Class Hinge Loss

Rubric: {points: 20}

Recall the constraints we introduce for multi-class linear classifiers. Plainly, for an example whose label is y_i, we want the linear score for y_i to be larger than the linear scores for other classes.

More formally, we want:

$$w_{y_i}^T x_i > w_{c'}^T x_i, \quad y_i \neq c' \in \{1, \cdots, k\}, \tag{1}$$

where k is the number of classes in labels.

Recall that counting the number of times this constraint is violated is how we derive the multi-class hinge loss, which yields a multi-class support vector machine. More formally, for each example whose label is y_i, we incur this penalty:

$$\sum_{c' \neq y_i} I(w_{y_i}^T x_i \leq w_{c'}^T x_i). \tag{2}$$

I is the indicator function which evaluates to 1 if the clause inside evaluates to true, and 0 otherwise.

Using this (2) to count constraint violations, we will derive the "sum"-rule multi-class hinge loss.

Answer the following questions:

1. Write down the objective function f that sums the violations over all examples in the training data.

   Answer: Our objective function is:

$$f(W) = \sum_{i=1}^{n} \sum_{c' \neq y_i} I(w_{y_i}^T x_i \leq w_{c'}^T x_i). \tag{3}$$

2. Rearrange the expression $I(w_{y_i}^T x_i \leq w_{c'}^T x_i)$ and incorporate a convex approximation to the indicator function using the max operation. Write down the new objective function f with this change.

   Answer: First, we restructure what's inside the indicator.

$$I(w_{y_i}^T x_i \leq w_{c'}^T x_i) = I(0 \leq -w_{y_i}^T x_i + w_{c'}^T x_i). \tag{4}$$

   Using max, we approximate the indicator with:

$$I(0 \leq -w_{y_i}^T x_i + w_{c'}^T x_i) \approx \max\{0, -w_{y_i}^T x_i + w_{c'}^T x_i\} \tag{5}$$

   Then the objective function is

$$f(W) \approx \sum_{i=1}^{n} \sum_{c' \neq y_i} \max\{0, -w_{y_i}^T x_i + w_{c'}^T x_i\}. \tag{6}$$

3. Use the "margin-of-1" trick to address the degeneracy when $W = 0$.

   Answer:

$$f(W) \approx \sum_{i=1}^{n} \sum_{c' \neq y_i} \max\{0, 1 - w_{y_i}^T x_i + w_{c'}^T x_i\}. \tag{7}$$

4. Show that $f(W) = nk - n$ when $W = 0$ with the "sum"-rule multi-class hinge loss f.

   Answer:

$$\begin{aligned} f(0) &= \sum_{i=1}^{n} \sum_{c' \neq y_i} \max\{0, 1\} && (8) \\ &= \sum_{i=1}^{n} k - 1 && (9) \\ &= n(k-1) = nk - n. && (10) \end{aligned}$$