STA 9890 - Statistical Learning for Data Mining

In-Class Test 1: ML Fundamentals

This is a closed-note, closed-book exam.
You may not use any external resources.

Name: _____

Instructions

This exam will be graded out of **100 points**.

This exam is divided into three sections:

- True/False (24 points; 10 questions at two points each)
- Short Answer (50 points; Best 10 of 12 questions at five points each)
- Mathematics of Machine Learning (26 points; one long question in 5 parts)

You have 90 minutes to complete this exam from the time the instructor says to begin. The instructor will give time warnings at: 30 minutes, 15 minutes, 5 minutes, and 1 minute.

When the instructor announces the end of the exam, you must stop **immediately**. Continuing to work past the time limit may be considered an academic integrity violation.

Write your name on the line above *now* before the exam begins.

Each question has a dedicated answer space. Place all answers in the relevant spot. Answers that are not clearly marked in the correct location **will not** receive full credit. Partial credit may be given at the instructor's discretion.

Mark and write your answers clearly: if I cannot easily identify and read your intended answer, you will not get credit for it.

Additional pages for scratch work are included at the end of the exam packet.

This is a closed-note, closed-book exam.
You may not use any external resources.

True/False: 24 points total at 2 points each

For each question, **CIRCLE** your answer(s).

TF1. True/False: Linear regression is a supervised learning method because it has a matrix of features $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ and a vector of responses $\boldsymbol{y} \in \mathbb{R}^n$ which we attempt to predict using $\boldsymbol{X}$.

TRUE FALSE

TF2. True/False: Models with higher training error always have higher test error.

TRUE FALSE

TF3. True/False: The matrix $\boldsymbol{Q} = \begin{pmatrix} \cos 30^\circ & \sin 30^\circ \\ -\sin 30^\circ & \cos 30^\circ \end{pmatrix} = \begin{pmatrix} \sqrt{3}/2 & 1/2 \\ -1/2 & \sqrt{3}/2 \end{pmatrix}$ is orthogonal.

TRUE FALSE

TF4. True/False: If $f(\cdot)$ is a convex function and $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$ for some $\boldsymbol{x}$, then $\boldsymbol{x}$ is a local minimizer of $f(\cdot)$.

TRUE FALSE

TF5. True/False: OLS always finds the linear model with the lowest training MSE.

TRUE FALSE

TF6. True/False: The loss function $\mathcal{L}(y, \hat{y}) = \log(1 + |y - \hat{y}|)$ is convex.

TRUE FALSE

TF7. True/False: If $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ $(n > p)$ is full-rank and has an SVD given by $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$, the OLS linear regression coefficients of $\boldsymbol{y}$ on $\boldsymbol{X}$ are given by $\hat{\boldsymbol{\beta}} = \boldsymbol{U}\boldsymbol{D}^{-1}\boldsymbol{V}^\top\boldsymbol{y}$

TRUE FALSE

TF8. True/False: Reducing bias always increases variance.

TRUE FALSE

TF9. True/False: K-Nearest Neighbors can be used for regression and classification.

TRUE FALSE

TF10. True/False: For any given vector $\boldsymbol{x} \in \mathbb{R}^p$, we have $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_1$.

TRUE FALSE

TF11. True/False: A model with high complexity is more likely to underfit the training data.

TRUE FALSE

TF12. True/False: Gradient descent iteratively updates model parameters by moving them in the direction of the gradient.

TRUE FALSE

Short Answer: 50 points total at 5 each [Top 10]

SA1. Give an example that demonstrates why the ℓ_0-"norm" is not convex.

SA2. Dynamic pricing (*i.e.*, changing the prices on a website based on user purchasing and search history) is a popular example of *reinforcement learning* (RL). Define RL and explain why dynamic pricing can be analyzed as an RL problem.

SA3. When might *stochastic gradient* methods be preferred over standard (non-stochastic) gradient methods in model fitting and what advantages do they provide?

SA4. Misclassification error seems like a natural loss function for classification, but it is actually quite difficult to work with. Why?

SA5. In your own words, define the strategy of *empirical risk minimization*. Why do we use it? What does it guarantee us? What can't it guarantee us?

SA6. In your own words, explain why use of a holdout set (or similar techniques) is important for choosing hyperparameters in machine learning.

SA7. Bayesian Optimization is one way of applying *active learning* techniques to tune the hyperparameters of a complex ML system. Describe two other applications of active learning (not necessarily in model tuning). For each application, describe both the decision be made at each step and the final quantity being estimated.

SA8. The square matrix $\boldsymbol{A}$ has a singular value decomposition given by:

$$\boldsymbol{A} = \begin{pmatrix} 0 & -2 & 0 \\ 3 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}^{\top}$$

Is $\boldsymbol{A}$ invertible? If yes, what is its inverse? If no, why not?

SA9. Give three examples of classification problems: one binary, one multi-class, and one ordinal.

SA10. The bias-variance of MSE for prediction has the form Test-MSE $= \text{Bias}^2 + \text{Variance} +$ Irreducible Error. Define each of these terms and explain (in words) where they come from.

SA11. Give two strengths and two weaknesses of K-nearest neighbor methods.

SA12. OLS is known to be *BLUE* in some circumstances. Define what it means for a method to be BLUE, state the conditions under which OLS is BLUE, and explain why this may not be as important as it originally seems.

Mathematics of Machine Learning: 26 points total

In this problem, you will derive a new loss function for regression, use it to pose a new ERM problem, solve it in closed-form using linear algebra, develop an iterative algorithm for fitting this model on exceptionally large data sets, and construct a simulation to estimate its error. Note that not all parts of this problem are weighted equally.

MML1) You are given a set of n training points $\{(x_i, y_i)\}_{i=1}^n$ and want to fit a *quadratic* model (with intercept) this data. To do so, you will need to express your data in terms of linear algebra quantities (vectors or matrices), $\boldsymbol{X}$, $\boldsymbol{y}$, and $\boldsymbol{\beta}$. Write out $\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\beta}$ for this problem, including any necessary feature engineering, and give the dimensions of $\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\beta}$. [4 points]

MML2) For this problem, you want to minimize *squared percent error*, with a (pointwise) loss function:

$$\mathcal{L}(y, \hat{y}) = \left(\frac{y - \hat{y}}{y}\right)^2 = \frac{(y - \hat{y})^2}{y^2} = \left(1 - \frac{\hat{y}}{y}\right)^2$$

Write the *empirical risk minimization* problem corresponding to this loss function using your $\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\beta}$ from the previous part. If you need to define any additional quantities (and you likely do!), make sure to define them clearly and give their dimensions. [4 points]

MML3) Give a closed form (matrix algebra) expression for the estimated coefficient vector $\hat{\boldsymbol{\beta}}$. *Hint:* Differentiate your objective function from the previous part, set the gradient to zero, and solve for $\hat{\boldsymbol{\beta}}$. [5 points]

MML4) When fitting this model to large data sets, the matrix algebra might be too slow to implement exactly. In this context, you might choose to use an iterative (gradient) method instead. Derive a gradient descent method for this problem and write out pseudocode to implement your algorithm. You do not need to specify the step-size and can leave it as a generic step size γ_k. [7 points]

Extra credit (3 points): what is the largest possible (fixed) value of $c = \gamma_k$ for all iterations k that we can use that will still guarantee convergence?

MML5) Unlike OLS, this solution is no longer linear in $\mathbf{y}$ (due to the y_i in the denominator of the loss) which makes it hard to determine the bias or the variance analytically. In pseudo-code, write out a simulation that would determine the MSE of this estimator. You may make any assumptions needed about the true relationship between X and Y, the (marginal) distributions of X or the noise, *etc.* as long as you state them clearly. [6 points]

(Blank page for scratch work - not graded)

(Blank page for scratch work - not graded)

STA9890 - Test 1 - Formula Sheet

Linear Algebra:

- A *n-vector* is an ordered set of n (real) numbers: $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, with addition $\boldsymbol{x} + \boldsymbol{y} = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n)$ and vector (inner / dot) product: $\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x}^\top \boldsymbol{y} = \sum_{i=1}^n x_i y_i$
- Vector norms: $\|\boldsymbol{x}\|_p = \sqrt[p]{\sum_{i=1}^n |x_i|^p}$ with $\|\boldsymbol{x}\|_\infty = \max_i\{|x_i|\}$ and $\|\boldsymbol{x}\|_0 =$ Number of non-zero elements of $\boldsymbol{x}$
- An $m \times n$ matrix is a 2D array of real numbers with m rows and n columns:

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}$$

- A matrix-vector product takes an n-vector as input and gives an m-vector as output:

$$\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{A}_{1\cdot} \cdot \boldsymbol{x}, \boldsymbol{A}_{2\cdot}\boldsymbol{x}, \ldots, \boldsymbol{A}_{m\cdot} \cdot \boldsymbol{x}) \in \mathbb{R}^m$$

- We can multiply an $m \times n$ matrix with an $n \times p$ matrix - note that the 'inner' dimensions must match:

$$\boldsymbol{A}\boldsymbol{B} = \begin{pmatrix} \boldsymbol{A}_{1\cdot} \cdot \boldsymbol{B}_{\cdot 1} & \boldsymbol{A}_{1\cdot} \cdot \boldsymbol{B}_{\cdot 2} & \ldots & \boldsymbol{A}_{1\cdot} \cdot \boldsymbol{B}_{\cdot n} \\ \boldsymbol{A}_{2\cdot} \cdot \boldsymbol{B}_{\cdot 1} & \boldsymbol{A}_{2\cdot} \cdot \boldsymbol{B}_{\cdot 2} & \ldots & \boldsymbol{A}_{2\cdot} \cdot \boldsymbol{B}_{\cdot n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{m\cdot} \cdot \boldsymbol{B}_{\cdot 1} & \boldsymbol{A}_{m\cdot} \cdot \boldsymbol{B}_{\cdot 2} & \ldots & \boldsymbol{A}_{m\cdot} \cdot \boldsymbol{B}_{\cdot n} \end{pmatrix} \in \mathbb{R}^{m \times p}$$

  Consider n-vectors as *one-column* matrices to make all of these definitions consistent. Requiring the dimensions in multiplication to align is a good way to verify linear algebra claims. (*E.g.*, $\boldsymbol{A}\boldsymbol{A}$ does not work for non-square $\boldsymbol{A}$)
- A matrix inverse satisfies $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$. Only full-rank square matrices have inverses
- An (square) orthogonal matrix $\boldsymbol{Q}$ satisfies $\boldsymbol{Q}^\top = \boldsymbol{Q}^{-1}$. If we take the first $n' \leq n$ columns (rows) of an orthogonal matrix we have $\boldsymbol{Q}_{1:n'\cdot}\boldsymbol{Q}_{1:n'\cdot}^\top = \boldsymbol{I}_{n' \times n'}$ so it's transpose-inverse along the 'short-side'
- Any matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ has a *singular value decomposition*: $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$ where $r = \min\{m, n\}$, $\boldsymbol{D}$ is a non-negative diagonal $r \times r$ matrix, $\boldsymbol{U} \in \mathbb{R}^{m \times r}$ is the first r columns of an orthogonal $m \times m$-matrix, and $\boldsymbol{V} \in \mathbb{R}^{n \times r}$ is the first r columns of an orthogonal $n \times n$ matrix
- Distributive rules: $(\boldsymbol{A}\boldsymbol{B})^\top = \boldsymbol{B}^\top\boldsymbol{A}^\top$ and $(\boldsymbol{A}\boldsymbol{B})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$ (if all defined)

Matrix Calculus:

- Quadratics: $f(\boldsymbol{x}) = \boldsymbol{x}^\top\boldsymbol{A}\boldsymbol{x} \implies \nabla f(\boldsymbol{x}) = 2\boldsymbol{A}\boldsymbol{x}$; $f(\boldsymbol{x}) = \|\boldsymbol{x}\|^2 = 2\boldsymbol{x}$
- Chain rule: $g(\boldsymbol{x}) = f(\boldsymbol{A}\boldsymbol{x}) \implies \nabla g(\boldsymbol{x}) = \boldsymbol{A}^\top(\nabla f)(\boldsymbol{A}\boldsymbol{x})$

Convexity:

- A function $f : \mathbb{R}^p \to \mathbb{R}$ is *convex* if

$$f(\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda)f(\boldsymbol{y}) \text{ for all } \lambda \in [0,1], \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$$

  If f is convex and second-differentiable at a point, its second derivative matrix is *positive semi-definite*
- A set $\mathcal{C} \in \mathbb{R}^p$ is convex if

$$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C} \implies \lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in \mathcal{C} \text{ for all } \lambda \in [0,1], \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$$

- If $\nabla f(\boldsymbol{x}_*) = 0$ for convex $f(\cdot)$, then $\boldsymbol{x}_*$ is a global minimizer of $f(\cdot)$

Gradient Methods:

- Given an optimization problem $\min_{\boldsymbol{x} \in \mathcal{C}} f(\boldsymbol{x})$, gradient descent works by repeating the following update:

$$\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)} - c\nabla f(\boldsymbol{x}^{(k)})$$

  If $c > 0$ is sufficiently small and $\mathcal{C} = \mathbb{R}^p$, $\boldsymbol{x}^{(k)}$ will converge to a minimizer of f