LE/EECS 5327: INTRODUCTION TO MACHINE LEARNING & PATTERN RECOGNITION

Cheat Sheet (Version α)

S.Toyonaga
York University
Lassonde School of Engineering

Contents

1 Introduction & Preface

These notes were created to supplement the lecture slides and accelerated recitation notes (see the shared Google Drive). On its own, this document is not sufficient for completely preparing you for the midterm and/or exam. Rather, it should be used as a consulting guide, similar to how cue cards are used.

Due to the remaining time left, these notes have only gone through one iteration. Thus, for any errors, please reach out to the author(s).

2 General Definitions

1. **Machine Learning (Arthur Samuel):** A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

   TLDR: $(Experience \wedge Incr(P, T)) \rightarrow Learn(E, T, P)$

2. **Supervised Learning:** Used on labeled data and consists of building models that predict the label for unseen data.

   i.e.,

   (a) Classification (Finite States)

   (b) Regression (Numerical, Continuous States)

3. **Unsupervised Learning:** Used on unlabeled data and consists of algorithms that simplify our data without losing a lot of information. It is often used as a preprocessing stage.

   i.e.,

   (a) Clustering

   (b) Dimensionality Reduction

4. **Reinforcement Learning:** An Agent must navigate an environment and reach a goal. A policy is built over many epochs by assigning certain actions a reward / punishment.

5. **True Risk:** The expected loss of all data. It uses all conceptually applicable data. In practice, while ideal, this is infeasible.

6. **Empirical Risk:** The average loss of data among samples of the model. It includes a subset of the true risk and is practical.

7. **Empirical Risk Minimization (ERM):** Finding a setting of the parameters that minimizes the empirical risk on the training set.

8. **Loss Function $L(y^{(i)}, \hat{y}^{(i)})$:** Measures how inaccurate a model's prediction is at a particular point.

9. **Cost Function (J): The averaged total loss of a particular model.**

10. **Non-Parametric:** The model structure is not specified a priori but is instead determined from data

11. **Lazy Learning:** Generalization of the training data is delayed until a query is made to the system. In other words, the model does not need to be trained beforehand to make predictions. It fits just-in-time...

12. **Binary Classification:** The task of classifying the elements of a set into two groups on the basis of a classification rule.

13. **Multi-Class Classification:** The problem of classifying instances into one of three or more classes.

14. **Multi-Label Classification:** A variant of the classification problem where multiple non-exclusive labels may be assigned to each instance.

15. **Imbalanced Classification:** Classification when there is an unequal distribution of classes in the training dataset.

16. **Step Function:** A function that returns 1 if the value of the input x is positive and 0 otherwise.

17. **Support Vectors:** The data points that are closer to the hyperplane and influence the position and orientation of the hyperplane.

18. **Linearly Separable:** Occurs when two sets of data points can be completely separable by a single straight line.

19. **Complete Separation:** Occurs when the outcome variable separates the predictor variables completely.

20. **Impurity:** A measure of the homogeneity of the labels at the node of a decision tree

21. **Discriminate Model(s):** A class of logistical models used for classification/regression. They distinguish decision boundaries through observed data.

22. **Generative Model(s):** Includes the distributions of the data itself, and tells you how likely a given example is. The distribution characters include the mean and standard deviation.

23. **One-Hot Encoding:** Turning a non-binary categorical feature into several binary features. The output is an array of 0's and 1's.

24. **Problem: Transformation to Binary (Reduction)**

    - **One vs Rest (n classifiers):** Consists of fitting one classifier per class. For each classifier, the class is fitted against all other classes. **The output is probabilities [0, 1].**

- **One vs One ($nC2$ classifiers):** For the N-class instances in the dataset, we have to generate $nC2$ binary classifiers. We split the primary dataset for each class opposite to every other class. **The outputs are labels (final classification is based on the corresponding majority vote)**

25. **Problem: Transformation from Multi-Label**

    - **Binary Relevance:** One-vs-Rest, except that we use one-hot encoding.
    - **(Binary) Classifier Chains:** Linking off-the-shelf binary classifiers together in a chain structure such that class label predictions become features for other classifiers.
    - **(Multi-Class) Label Powerset:** One multi-class classifier trained on all unique label combinations found in the training data.
    - **(Miscellaneous) Ensemble Methods: Decision Tree & Subsets**
    - **(Miscellaneous) Adapted Algorithms:** Multi-Valued Multi-Level Decision Trees (MMDT)

26. **Technique: Extension from Binary (Supporting Multi-Class)**

    - **Decision Tree (Adapted Gini and Entropy Formulas)**
    - **GDA (Adding more clusters)**
    - Logistic Regression (Softmax Function)

      (a) $score(\hat{y} = n) = e^{z_n} = e^{w_1x_1+\cdots+w_nx_n+b}$

      (b) $P(\hat{y} = n) = \frac{e^{z_n}}{e^{z_1}+\cdots+e^{z_n}}$

      (c) $L(y^{(i)}, \hat{y}^{(i)}) = \Sigma_{i=0}^{k} - 1(y^{(i)} = i)log(P(\hat{y}^{(i)} = i)$: If $y = i$, return 1. Else, return 0.

27. **Feedforward:** The process in which the inputs to a neural network are fed through the neural network to make a prediction, given the activation function, weights, and bias.

28. **Backpropagation:** The algorithm used to train a neural network. We initialize the neural network with random weights and biases. For many epochs, we calculate the loss function and its gradient and take small steps in the direction opposite to the gradient to decrease the loss function by a small amount. The final weights and bias obtained correspond to a model that likely fits the data well.

29. **Clustering Algorithms:**

    - **Centroid-Based (K-Means):**
    - **Distribution-Based (Gaussian Mixture Models (GMM)):**
    - **Density-Based (DBSCAN):**
    - **Hierarchical:**
    - **Grid-Based:**

30. **Dimensionality Reduction:** A computationally-expensive task which reduces the number of features in high-dimensional space by combining or removing unnecessary information that does not significantly affect the underlying dataset / variance.

    - **Principal Component Analysis (PCA)**
    - **Autoencoder (Neural Network)** $\rightarrow$ The first $\frac{n}{2}$ layers represents the encoder. The middle layer represents the latent representation (number of features after compression). The last $\frac{n}{2}$ layers are for decoding to obtain the original input layer.

31. **Backpropagation Template:** We use a template to support an inductive/recursive solution. (start, current, next layer(s))

32. **Types of Gradient Descent:**

Metric	Stochastic (SDG)	Mini Batch	Batch
Update Data	Single Observation	Subset	$\forall$
Execution Speed	Fastest	Compromise	Longest
J Curve	Chaotic	Drunk	Smooth
# Epochs	Largest	Compromise	Smallest

33. **Zero Initialization:** Initializing all the weights with zeros. This is bad for neural networks as it leads to the neurons learning the same features during training. Backpropagation will not work here.

34. **Random Initialization:** The weights are randomly initialized in this manner, very close to zero. As a result, for neural networks, symmetry is broken, and each neuron no longer performs the same computation. It aids in the symmetry-breaking process and improves accuracy.

35. **Underfitting:** Occurs when we oversimplify the problem at hand and try to solve it using a simple solution.

    - Use a more complicated model (decrease regularization parameter)
    - Use more features

36. **Overfitting:** Occurs when we overcomplicate the solution to a problem and try to solve it with an exceedingly complicated solution. The model tries to fit all of the training data points and fails to see the bigger picture. A good model will capture the trend and generalize well to unseen data points.

    - Collect More Data
    - Use Fewer Features (Regularization)
    - Early Stopping (Fewer Epochs for Training)
    - Increase Regularization Parameter

37. **Model Performance (Appendix):**

Metric	Underfitting	Optimal	Overfitting	Glitch
Training Error	High	Low	Low	High
Validation Error	High	Low	High	Low
Bias	High	Low	Low	
Variance	Low	Low	High	

- High Bias → Certain performance thresholds are bounded by the simplicity of the model. More training examples will not help. Additionally, training for longer (iterations) does not help.
- High Variance → More training points may help to close the gap to a smaller degree. Training too long can lead to overfitting.

38. **Dataset Splitting (Randomized):**

Training Set (60%)	For training the model
Validation Set (Dev) (20%)	For deciding which model to use
Testing Set (20%)	For checking how well our model did (evaluation & reporting purposes). If the model is not good after testing, we throw everything out and start from scratch.

39. **Hyperparameters:** The attributes which affect the learning process of a model.

40. **Regularization:** Regularization refers to techniques that are used to calibrate machine learning models in order to minimize the adjusted loss function and prevent overfitting or underfitting.

L1 Regularization	L2 Regularization
Built-In Feature Selection (Sparsity)	Requires Feature Selection (Non-Sparse)
Computationally Cheap	Computationally Expensive
Cannot be differentiated at 0	Can be differentiated everywhere
Robust to Outliers	Sensitive to Outliers

41. **Error Analysis:** An iterative development process of developing diagnostics that focuses on (1) collecting more data to deal with class imbalance, and (2) developing/defining more sophisticated features. One potential negative is that it isn't 100% explainable at all times.

42. **Data Augmentation:** Adds synthetic data to the data set where distortions are representative of the type of noise that we see in the test set.

43. **k-folds cross validation (evaluation):** A technique for evaluating predictive models. The data set is divided into k subsets or folds. The model is trained and evaluated k times, using a different fold as the validation set each time. Performance metrics from each fold are averaged to estimate the model's generalization performance.

44. **leave-one-out cross validation (small data):** K-fold cross validation taken to its logical extreme, with K equal to N, the number of data points in the set. That means that N separate times, the function approximator is trained on all the data except for one point and a prediction is made for that point.

45. **Forward Search:** Starts with an empty set of attributes. In each subsequent step it proceeds by adding either a random attribute (faster) or an attribute that optimizes some criterion (slower), and accepts the addition if the new feature subset improves the optimization criterion.

46. **Precision:** Shows how often an ML model is correct when predicting the target class. That is, it is the quality of positive predictions.

47. **Recall:** Shows whether an ML model can find all objects of the target class. That is, it is the quality of true positives.

48. **Scaling Matrix:** $\begin{bmatrix} k & 0 \\ 0 & q \end{bmatrix}$ where k scales the shape horizontally and q scales the shape vertically.

49. **Rotation Matrix:** $\begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$ where the original shape is rotated θ degrees counter-clockwise.

50. **Determinant:** The determinant is the difference in area after the transformation. For a 2×2 square matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, $det(A) = ad - bc$. Do note that if $det(A) = 0$, this implies that A has no solutions or infinitely many solutions.

51. **Eigenvector:** An eigenvector or characteristic vector is such a vector where vectors have their directions unchanged by a linear transformation. Thus an eigenvector of a linear transformation is scaled by a constant factor when the linear transformation is applied to it.

52. **Eigenvalue:** Eigenvalues are the special set of scalar values that is associated with the set of linear equations most probably in the matrix equations.

Putting it all together: Given a square $n \times n$ matrix A, then $A\vec{x} = \lambda\vec{x}$ such that:

- $\vec{x}$ is the eigenvector
- λ is the eigenvalue

53. **Principal Component Analysis (PCA):** A dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

    - **Goal:** Maximize the variance (or, minimize the residuals) of the points, post-projection onto the new axis.

- **Principal Components:** The axis that projects the data
- **Limitations**
  - The Principal Component (axis) may not be human-interpretable. That is, there is no clue what the axis represents; PCA is a linear combination of the original data.
  - Information Loss (Dimensionality Reduction)
  - Only suitable for unsupervised learning; using it to construct models or address overfitting is not good. PCA does not take into account the label. Use regularization; the optimizer considers prediction power and the dimensions of the dataset.

54. **Covariance:** The mean value of the product of the deviations of two (or more) variates from their respective means.

55. **Maximum Likelihood Estimation (MLE):** A method for estimating the parameters (p) of an assumed probability distribution (i.e., binomial), given some observed data. The goal is to maximize a likelihood function so that under the assumed statistical model, the observed data is most probable. To find the Maximum Likelihood Estimation (MLE), you must:

    (a) Find the model (i.e., binomial) and parameters (i.e., $p(red)$)

    (b) Find the Likelihood Function based on the observation

    (c) Maximize Probability $\rightarrow$ Set derivative of the likelihood function to 0 and solve for p

    (d) Simplify

56. **Log-Likelihood:** The natural logarithm of the likelihood function. We do this to simplify the equation for cost functions of models (i.e., Linear, Logistic). It has the following beneficial characteristics:

    (a) **Monotonicity:** $f_1 > f_2 \rightarrow log(f_1) > log(f_2)$

    (b) $log(ab) = log(a) + log(b) \rightarrow$ (a) Prevents underflow (b) Makes calculating derivatives more straightforward / easy

    (c) Maximizing $log(f) \longleftrightarrow$ Minimize J

57. **Probability Mass Function (pmf):** A function that gives the probability that a discrete random variable is exactly equal to some value.

58. **Probability Density Function:** A function whose value at any given point can be interpreted as providing a relative likelihood that the value of the random variable would be equal to that sample.

59. **Uniform Distribution ($X \rightarrow U(a, b)$):** Each value in the set of possible values has the same possibility of happening. When displayed as a bar or line graph, this distribution has the same height for each potential outcome. The area under the line must be equal to 1.

60. **Joint Probability Density Function ($X_1, X_2 \rightarrow U(a,b)$):** The joint probability density function (joint pdf) of X and Y is a function $f(x,y)$ giving the probability density at (x, y). That is, the probability that (X, Y) is in a small rectangle of width dx and height dy around (x,y) is $f(x,y)dxdy$. Do note that the volume of the shape must be equal to 1.

61. **Entropy:** In the context of probability, it is a measurement of chaos (surprise) There are two different types of entropy. Shannon Entropy measures randomness in bits and uses log_2 while the ML approach measures natural units and uses ln.

62. **Cross Entropy:** The measured similarity between two populations.

63. **Maximum A Posteriori (MAP):** Involves calculating a conditional probability of observing the data given a model weighted by a prior probability or belief about the model. MAP provides an alternate probability framework to maximum likelihood estimation for machine learning.

Maximum Likelihood Estimation (MLE)	Maximum A Posteriori (MAP)
• Frequentist Approach • Based on Observations • $\hat{\theta}_{MLE} = argmax_\theta p(D\|\theta)$	• Baysian Approach • Based on Prior Knowledge and Observations • $\hat{\theta}_{MAP} = argmax_\theta p(\theta\|D)$

Note: MAD performs better if the outcomes are fair while MLE performs better if the outcomes are biased.

64. **Vanishing Gradient Problem:** Vanishing gradient problem is a phenomenon that occurs during the training of deep neural networks, where the gradients that are used to update the network become extremely small or "vanish" as they are backpropagated from the output layers to the earlier layers. This happens with multiple sigmoid activation functions in particular. Tanh is also suscepible to the vanishing gradient problem. ReLU/LeakyRELU is a good solution because its derivative is 1 when the input is positive and thus, it is widely used in large neural networks.

65. **Exploding Gradient Problem**: Occurs when the weights are too large and this causes unexpected behaviour such as overshooting. Weight Initialization techniques such as Xavier Initialization (tanh/sigmoid) or He Initialization (ReLU) have shown promise it reducing this problem.

66. **Kernel Method for Non-Linearly Separative Data:** A trick for higher order data where x is projected onto a higher-order dimension to separate key points. This is to say that for $wx + b$, we re-project it as $w_1x_1 + w_2x^2 + w_3x^3 + \cdots + b$.

3 Machine Learning Models

3.1 Linear Regression

- **Model:** $\hat{y} = w_1x_1 + \cdots w_nx_n + b \to X^T\vec{\theta}$
- **Goal:** The goal is to find a model (line) that can (1) fit the data well, and (2) predict new data. In essence, it seeks to find parameters w and b to reduce the cost as much as possible.
- **Loss Function:** $L(y^{(i)}, \hat{y}^{(i)}) = (y^{(i)} - \hat{y}^{(i)})^2$
- **Cost Function:** $J(w,b) = \dfrac{1}{2m}\Sigma_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)}) \to \frac{1}{2m}(\vec{y} - X^T\vec{\theta})(\vec{y} - X^T\vec{\theta})$
- **Gradient Descent** $= \begin{cases} w_j = w_j - \frac{\alpha}{m}\vec{x_j}^T(\hat{\vec{y}} - \vec{y}) \\ b = b - \frac{\alpha}{m}\vec{1}^T(\hat{\vec{y}} - \vec{y}) \end{cases}$
- **Types of Linear Regression:**
  1. Univariate Linear Regression
  2. Multiple Linear Regression

     Notation

     (a) $\vec{x}^{(i)}$ are the features of the ith training point

     (b) $x_j^{(i)}$ is the jth feature of the ith training point
  3. Polynomial Linear Regression
  4. Locally Weighted Regression (Non-Parametric, Lazy Learning)

3.2 Perceptron (Discriminate)

- **Model:** $\hat{y} = \begin{cases} 1, \; w_1x_1 + \cdots + w_nx_n + b > 0 \\ 0, \; w_1x_1 + \cdots + w_nx_n + b \leq 0 \\ DB: \; w_1x_1 + \cdots + w_nx_n + b = 0 \end{cases}$
- **Goal:** To minimize the distance of the misclassified examples to the decision boundary.

3.3 Support Vector Machine (SVM) (Discriminate)

- **Model:** $\hat{y} = \begin{cases} 1, \; w_1x_1 + \cdots + w_nx_n + b \geq 1 \\ -1, \; w_1x_1 + \cdots + w_nx_n + b \leq -1 \\ Margin: \; w_1x_1 + \cdots + w_nx_n + b = 0 \end{cases}$
- **Goal:** The goal is to maximize the distance of the margin to the support vectors and classify the data correctly.
- **Soft-Margin SVM:** Allows some misclassification to happen by relaxing the hard constraints of Support Vector Machine

3.4 Logistic Regression (Probabilistic Interpretation) (Discriminate)

- **Model:** $\hat{y} = \sigma(w_1 x_1 + \cdots w_n x_n + b) \rightarrow \sigma(X^T \vec{\theta})$
- **Goal:** To find a model that minimizes the cost function by utilizing Newton's method. Do note that there is no closed-form solution.
- **Interpretation:** $0 < \hat{y} < 1$ and $P(class = 0) + P(class = 1) = 1$
- **Decision Boundary:** $wx + b = 0$ if threshold $= 0.5$, or σ^{-1}
- **Loss Function:** $L(y^{(i)}, \hat{y}^{(i)}) = \begin{cases} -log(\hat{y}^{(i)}),\ y^{(i)} = 0 \\ -log(1 - \hat{y}^{(i)}),\ y^{(i)} = 1 \end{cases}$
  $= -y^{(i)} log(\hat{y}^{(i)}) - (1 - y^{(i)}) log(1 - \hat{y}^{(i)}))$
- **Cost Function:** $J(w, b) = \dfrac{1}{m} \Sigma_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)}) \rightarrow -\dfrac{1}{m}(\vec{y}^T log(\hat{\vec{y}}) + (\vec{1} - \vec{y})^T (\vec{1} - \hat{\vec{y}}))$
- **Gradient Descent:** $\vec{\theta} = \frac{\alpha}{m} X(\hat{\vec{y}} - \vec{y})$

3.5 Decision Trees (Discriminate)

- **Model:**
- **Goal:** The goal is to separate data that is not linearly separable with multiple horizontal lines. We do this by splitting nodes until they are pure.
- Works well on structured data but is very sensitive to small changes. Another advantage is that it is transparent and explainable.
- The tree grows by recursive splitting up until the cross-validation metric no longer grows (Gini Index, Entropy)
- **Cross-Validation Metrics:**
  1. Gini Index: $1 - p(A)^2 - p(B)^2 - \cdots - p(N)^2$
  2. Entropy: $-p(A)log(p(A)) - p(B)log(p(B)) - \cdots - p(N)log(p(N))$
- Weighted Gini Index: $\frac{a}{m} GI_1 + \frac{m-a}{m} GI_2$
- The Decision Tree will stop splitting when:
  - A node is pure
  - A tree reaches a specified maximum depth
  - Impurity score falls below a specified threshold
  - The number of samples in each cell falls below a threshold
- Decision Tree (Application Problems)
  - Multi-Class Features
    * For each feature, create an extra branch when computing the weighted Gini Index

- Continuous Value Features
  - Split the feature by a range (inequality) and then calculate the weighted Gini Index as usual.
- Multi-Class Labels
  - Group the splits with relation to the label by each class of the label and calculate the said Gini Index.
- Continuous Labels
  - We cannot use probabilities because the categories are not discrete. Thus, we leverage calculating the weighted variance.

3.6 Gaussian Discriminate Analysis (GDA) (Generative)

- **Goal:** Classifies the unknown point by finding the distribution of the classes and calculating their characteristics (μ, σ)
- Think of this like bougie clustering so to speak

3.7 Neural Network

- **Model:** See Below
  - **Input Layer:** X
  - **Sum:** $Z_1 = W_1 X + \vec{b_1} \wedge Z_n = W_n A_{n-1} + \vec{b_n}$
  - **Activation Function:** $A_n = f(Z_n)$
  - **Output:** $Y = A_n = g(Z_n)$
  - **Vectorized Backpropagation** $= \begin{cases} w_i = w_i - \frac{\alpha}{m}\frac{\partial J}{\partial Z_i} A_{i-1}^T \\ \vec{b_i} = \vec{b_i} - \frac{\alpha}{m}\frac{\partial J}{\partial Z_i}\vec{1} \end{cases}$
- **Structure:**
  - **Input Layer:** The first layer where the number of nodes is equal to the number of said features.
  - **Hidden Layer:** All the layers between the input and output. They can have varying sizes.
  - **Output Layer:** The last layer, giving the results.
    - **Regression** $\rightarrow$ Strictly positive range should use ReLU while all other cases should use LeakyReLU
    - **Classification** $\rightarrow$ For binary and multi-label, sigmoid/tanh are sufficient. Multiclass can use them as well, but it also has the affordance of using softmax.
  - **Depth:** The depth is equal to the number of layers minus 1 (input layer).
- **Activation Functions**
  - **ReLU:** If the input is negative, return 0. Else, return the input as-is.

  - **TanH:** Similar to sigmoid but less flat and between $[-1, 1]$
  - **LeakyReLU:** ReLU, but a smaller slope for negative values
  - **Sigmoid**

- **Why Non-Linear Activation Functions?**
  - Linear activation functions lose complexity that is enabled by multiple layers.
  - Linear activation functions may not be able to fit more complicated models.
  - Linear functions cannot achieve curved decision boundaries that complicated models will require for training. Hence, this is why ReLU works.

- **Goal:** To achieve the best accuracy (reducing the cost function) by using Gradient Descent. Note that there is no closed-form solution.

- **Limitations:**
  - The hyperparameters are hard to tune; finding the optimal assignments requires substantial computational power.
  - Neural networks are hard to debug. Answers have no human-interpretable justifications and there are ethical considerations to consider such as plagiarism and job security.

3.8 Random Forest

- **Motivation:** Able to overcome data sensitivity by making predictions on multiple trees and choosing the majority-vote. The idea is to sample on different data with replacement.
- **Feature Selection:** At each node, choose from a random subset of $k = \sqrt{n}$ features when n features are available.
- **Bagging (Random Sampling)**
- **Boosting (Error Selections)** $\rightarrow$ Instead of random sampling with replacement, we are more likely to choose samples that previously trained trees misclassified (i.e., XGBoost)

4 Formulas

1. Univ Regr Normal Equation: $w = \dfrac{\Sigma(x^{(i)} - \overline{x})(y^{(i)} - \overline{y})}{\Sigma(x^{(i)} - \overline{x})^2}$, $b = \overline{y} - w\overline{x}$
2. Multiple Regr Normal Equation: $\begin{bmatrix} w_1 \\ w_2 \\ \cdot \\ w_n \\ b \end{bmatrix} = (X^T X)^{-1} \times (X^T Y)$
3. Perceptron Updates: For each example, $w = w - \alpha(\hat{y}^{(i)} - y^{(i)})x^{(i)}$
   $b = b - \alpha(\hat{y}^{(i)} - y^{(i)})$

4. Sigmoid Function: $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1} = 1 - \sigma(-z)$

5. Tanh Function $\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

6. Vector Properties:
   - $\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a} = a_1 b_1 + \cdots + a_n b_n$
   - $\vec{a}^T \cdot \vec{b} = \vec{b}^T \cdot \vec{a} = \vec{a} \cdot b = \vec{b} \cdot \vec{a}$
   - $\vec{a} \circ \frac{1}{\vec{a}} = \begin{bmatrix} 1 \\ \vdots \end{bmatrix}$

7. Matrix Operations:
   - Transpose (Rows $\rightarrow$ Columns)
   - Addition & Subtraction
   - Multiplication (A $[n \times m]$ $\times$ B $[m \times o]$ = C $[n \times o]$)
   - Inverse ($AA^{-1} = A^{-1}A = I$)

8. Matrix Properties
   - $A + B = B + A$
   - $AB \neq BA$
   - $AI = IA = A$
   - $A(B + C) = AB + AC$
   - $A(BC) = (AB)C$
   - $A \circ (B \circ C) = A \circ B + A \circ C$
   - $(A^T)^T = A$
   - $(kA)^T = k(A)^T$
   - $(AB)^T = B^T A^T$
   - $(A + B)^T = A^T + B^T$
   - $(A \circ B)^T + A^T \circ B^T$
   - $(A^{-1})^{-1} = A$
   - $(A^-1)^T = (A^T)^{-1}$
   - $(AB)^{-1} = B^{-1}A^{-1}$
   - $diag(a_i \pm b_i) = diag(a_i) \pm diag(b_i)$
   - $diag(a_i b_i) = diag(a_i) \times diag(b_i)$
   - $\vec{a}^T diag(b_i) = \vec{a}^T \circ \vec{b}^T$
   - **Element-Wise:** $\frac{\partial \vec{y}}{\partial \vec{x}} = diag(f'(x_i))$

9. Dimension of Neural Network (Layers, Weight, Bias)
   - Input Layer: $l_0 \times m$
   - Weights (b/w two layers): $l_n \times l_{n-1}$
   - Bias (b/w two layers): $l_n \times 1$
   - Output Layer: $l_n \times m$

10. General Derivatives
    - Constant Multiply Rule
    - Sum Rule
    - $(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$
    - $(\frac{f(x)}{g(x)})' = \frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}$
    - $f(g(x))' = f'(g(x))g'(x)$

11. Matrix Calculus

- **Scalar-by-Scalar** ($\frac{\partial y}{\partial x}$)**:** Scalar Calculus
- **Scalar-by-Vector** ($\frac{\partial y}{\partial \vec{x}}$) Row vector with size equal to the number of items in $\vec{x}$
- **Vector-by-Scalar** ($\frac{\partial \vec{y}}{\partial x}$)**:** Produces a column vector with size equal to the number of items in $\vec{y}$.
- **Vector-by-Vector** ($\frac{\partial \vec{y}}{\partial \vec{x}}$)**:** Produces a Jacobian Matrix of size $\vec{y} \times \vec{x}$
- **Matrix-by-Scalar** ($\frac{\partial Y}{\partial x}$)**:** Element-wise derivatives on Y objects.
- **Scalar-by-Matrix** ($\frac{\partial y}{\partial X}$)**:** Transposed Element-Wise derivatives on Y objects. That is, if X has dimensions $n \times m$, the resulting matrix will be $m \times n$.
- **Matrix-by-Vector (3D Tensor)** ($\frac{\partial Y}{\partial \vec{x}}$)**:** Nested matrix of size Y
- **Vector-by-Matrix (3D Tensor)** ($\frac{\partial \vec{y}}{\partial X}$)**:** Nested column vector of size $\vec{y}$
- **Matrix-by-Matrix (4D Tensor)** ($\frac{\partial Y}{\partial X}$)**:** Nested matrix of size Y.

12. Partial Derivative Formulas for non-3D/4D Tensors

- $\frac{\partial \alpha}{\partial \rho} = 0$
- $\frac{\partial \alpha \rho}{\partial \rho} = \alpha$
- $\frac{\partial \rho}{\partial \rho} = I$
- $\frac{\partial \rho^T \alpha}{\partial \rho} = \alpha^T$
- $\frac{\partial \vec{x}^T \vec{x}}{\partial \vec{x}} = 2\vec{x}^T$
- $\frac{\partial \vec{x}^T A \vec{x}}{\partial \vec{x}} = \vec{x}^T(A + A^T)$
- $\frac{\partial \alpha^T X \beta}{\partial x} = \beta \alpha^T$
- $\frac{\partial \alpha^T X^T \beta}{\partial X} = \alpha \beta^T$
- $\frac{\partial f(p) g(p)}{\partial p} = \frac{\partial f(p)}{\partial p} g(p) + f(p) \frac{\partial g(p)}{\partial p}$
- $\frac{\partial f(g(p))}{\partial p} = \frac{\partial f(g(p))}{\partial g(p)} \frac{\partial g(p)}{\partial p}$

- Partial Derivatives for 3D/4D Tensors include:
  - Constant Multiply Rule
  - Sum Rule
  - Transpose Rule

13. Learning Rate (α) $= \begin{cases} \alpha \times 3, \text{ if the learning rate is too larger (oscillating/overshooting/horizontal line)} \\ \frac{\alpha}{3}, \text{ if the learning rate is too small (not updating/horizontal line)} \end{cases}$

14. Feature Scaling $= \begin{cases} \textbf{Normalization}: x_j = \frac{x_j - min(x)}{max(x) - min(x)} \\ \textbf{Standardization}: x_j = \frac{x_j - \mu}{\sigma} \end{cases}$

15. Precision $= \frac{TP}{TP+FP}$

16. Recall $= \frac{TP}{TP+FN}$

17. F-Score $= \frac{2 \times P \times R}{P+R}$

18. Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$

19. Variance (σ^2) $= \frac{\Sigma(x-\overline{x})^2}{m}$

- **Note:** If the variance is graphed, the higher the variance (and hence, the larger the standard deviation), (1) the more the data is spread (chonky), and (2) the lower the peak.

20. Euclidean Distance (d) $= \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2 + ...}$

21. Within-Cluster Sum of Square (WCSS) $= (x_i - \overline{x})^2 + (y_i - \overline{y})^2 + \cdots$

    Note: A smaller WCSS means that the cluster is more compact (better).

22. Silhouette Score (SC) $= \frac{b^{(i)} - a^{(i)}}{max(a^{(i)}, b^{(i)})}$ where

    - $a^{(i)} \rightarrow$ Average distance to the points in the same cluster (small number is preferred)
    - $b^{(i)} \rightarrow$ Average distance to the points in the nearest cluster (bigger number is preferred)

23. Covariance: $\sigma_{xy}^2 = \frac{1}{m}\Sigma_{i=1}^{m}(x_i - \overline{x})(y_i - \overline{y}) = \sigma_{yx}^2$

24. Shannon Entropy: $H(X) = -\Sigma\, p(X) log_2(p(X))$

25. Natural Units: $H(X) = -\Sigma\, p(X) ln(p(X))$

26. Cross Entropy: $H(p, q) = -\Sigma\, p(X) ln(q(X))$

27. Maximum A Posteriori (MAD): By Bayes rule, $P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$ and $P(D|\theta) \alpha P(D|\theta)P(\theta)$ by the fact that $P(D)$ is a constant. Thus, $\hat{\theta}_{MAP} = argmax_{\theta} P(D|\theta)P(\theta)$

28. MLE for Logistic Regression: $p^{(i)} = \begin{cases} \hat{y}^{(i)}, \hat{y}^{(i)} = 0 \\ 1 - \hat{y}^{(i)}, \hat{y}^{(i)} = 1 \end{cases} \longleftrightarrow (\hat{y}^{(i)})^{y^{(i)}}(1 - \hat{y}^{(i)})^{1-y^{(i)}}$.

29. MLE for Linear Regression: $f(y^{(1)}, y^{(2)}, \cdots, y^{(m)}) = (\frac{1}{\sqrt{2\pi}\sigma})^m\; e^{-\frac{1}{2\sigma^2}\Sigma_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2}$

30. MAP for Logistic Regression: $P(\theta) = (\frac{1}{\sqrt{2\pi}\sigma})^{n+1}\; e^{-\frac{1}{2\sigma^2}(w_0^2 + w_1^2 + \cdots + w_n^2)}$

31. Kernel: $\langle \vec{x}^{(i)}, \vec{x} \rangle = \langle f(\vec{x}^{(i)}), f(\vec{x}) \rangle$

5 Exam Review Lecture (Section A)

5.1 Logistics

- Exam on December 12 from 2:00 p.m. - 5:00 p.m. at TC SOBEYS
- Clarification questions will not be answered
- Formula sheet will be provided
- Calculator is allowed
- **Make sure to watch the tutorial videos!!**
- Cumulative Exam with an Emphasis on Post-Midterm Material(s).
- **Exam Breakdown (ΣPoints = 150):**
  1. Multiple Choice (Singular Answer) [$\times$10]
  2. Multiple Choice (Many Answers) [$\times$20]
  3. Multiple Choice & Short Answer [$\times$41]
  4. Long Answer [$\times$14]
  5. Long Answer [$\times$14]
  6. Long Answer [$\times$25]
  7. Long Answer [$\times$26]

5.2 Choosing & Defining a Project

- **End-to-End System:** For a problem, it does not break it down into smaller pieces, and just take the raw data as input. The goal is strictly generating a good output. An end-to-end system enjoys the advantage of not accumulating errors as in modular systems where errors will accumulate when data flows from one module to the next. However, it also has the limitation that when the context of the problem changes or the technique develops, the entire system may need to be retrained.
- **Modular System:** A modularized system (i.e., non-end-to-end system) can update each module without impacting other modules or only has minor impact.
- The explainability and ethics of a proposal will affect whether machine learning is well-suited to the objectives or not.
- Collecting data and making predictions must be reasonable; that is, the solution must take into account time constraints and access to certain devices.

5.3 Techniques & High-Level Algorithms

- **Recall:** Regression and Classification tasks do not share a strict decision boundary. Often, without a loss of generality, one such task can be reformulated into another.

- Be familiar and comfortable with all techniques and algorithms relating to (1) linear regression, (2) logistic regression, and (3) neural networks. For example, knowing how to derive the vectorized representations (i.e., Gradient Descent) will be a tested question.
- **Midterm:** Make sure that you can answer relevant questions and/or have an intuitive understanding of the following list:
  1. Perceptron (Definition, Goal, Update Function)
  2. SVM (Definition, Goal, Finding the Decision Boundary)
  3. GMM
  4. DBSCAN (Smiley Face)
  5. AutoEncoder (Layers and Bottleneck Compression) [Not Explainable!!]

5.4 Decision Tree & Random Forest

- **Precision:** Plug & Play → Tree Traversal
- **Random Forest**
  - **Motivation:** Overcomes the problem of sensitivity by voting on a majority system. (Bagging, Boosting)

5.5 k-Means

- Have a general understanding of the algorithm (i.e., how to compute the centroids)
- Choosing the optimal cluster (WCSS, Sihouette Score, etc.,)
- **Limitations;** know when to use each type of clustering algorithm.

5.6 Principal Component Analysis

- Reduces dimensions by constructing **new** features that are parameterized with principle components.
- **Relevant Math:** Variance, Covariance, Covariance Matrix
- **PCA vs Linear Regression** → PCA cannot make predictions.
- **PCA vs L1 Regularization** → PCA does not completely drop variables, but creates new features (linear combinations). It cannot handle overfitting, nor taking the output into account (via optimizing a cost function.)

5.7 Practical Advice (I-III)

- Understand the strength and weakness of the various types of different Gradient Descent (Batch vs Stochastic vs Min-Batch)
- Updating Learning Rate (By a factor of 3)

- Why and How to Perform Feature Scaling $\rightarrow$ **Note:** We do not change the values for μ, σ across different sets (training, validation, test)
- Weight Initialization (Model Applicability)
- Bias/Underfitting, Variance/Overfitting
- Data Augmentation $\rightarrow$ Synthesis, K-fold Cross Validation, Leave-One-Out
- Skewed Dataset $\rightarrow$ (Recall, Precision, F-score)

6 Group Study Session (Meeting Notes)

1. Session #1 Topics (Thursday December 7, 2023):
   - Midterm Review (Perceptron, SVM, Neural Network) [1 Hour]
   - Post-Midterm (Matrix Calculus $\rightarrow$ MLA & Advanced Topics) [2 Hours]
2. Session #2 Topics (Friday December 8, 2023):
   - Vectorizations & Derivations [2 Hours]
   - Accelerated Cumulative Review [2 Hours]

Note: Session notes are available upon request.

6.1 Pre-Midterm

- SVM (Finding the Equation of the Margin(s) and Decision Boundary)
  - Intuition-based approach involves drawing the margins and then moving towards eyeing the decision boundary.
  - Sometimes its easier to rewrite the equation in terms of x and y rather than x_1 and x_2 to get a better picture of the problem and then convert them back once the solution is found.
- Perceptron (Multi-Feature Updating)
  - The intuitive approach would be to draw the decision boundaries pre and post-update.
  - The algorithmic approach would be to use the formula, hence, plug & play.
- Neural Network (Feedforward & Layer-by-Layer Matrix Representation)
  - **Approach #1:** Feedforward approach to identify the prediction ($\hat{y}$) followed by another pass to quickly fill in the matrix template.
  - **Approach #2:** Neural Network / Matrix Intuition
    1. **Input Layer**
       (a) Given the input X, $Z_1 = W_1X + \vec{b_1}$
    2. **Intermediate Layer**
       (a) $Z_n = W_nA_{n-1} + \vec{b_n}$
       (b) A_n is just Z_n but with the activation function applied in an element-wise fashion.
    3. **Output Layer**
       (a) $A_n = f(Z_n) \longrightarrow Y = A_n = g(Z_n)$

    Note: See the image *nn_template.png* under the img directory for a visual representation.

6.2 Post-Midterm

- Vectorization & Derivations

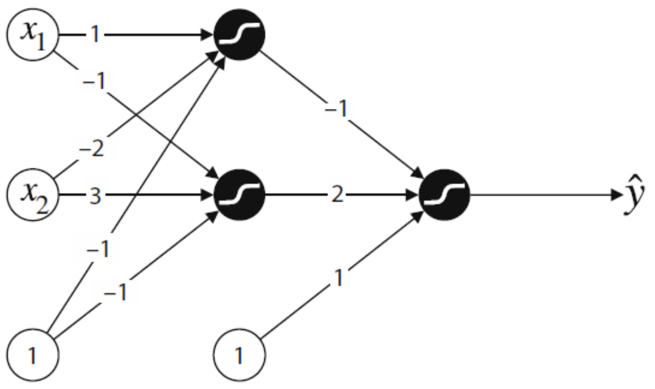
  1. **Cost of Linear Regression:**
     - $J(\vec{\theta}) = \frac{1}{2m}(\vec{y} - X^T\vec{\theta})^T(\vec{y} - X^T\vec{\theta})$
     - The key to understanding how to derive this is to know that $\Sigma_{i=1}^{n} a_i^2 = \vec{a} \cdot \vec{a} = ||\vec{a}||^2 = \vec{a}^T\vec{a}$
  2. **Cost of Logistic Regression**
     - $J(\vec{\theta}) = -\frac{1}{m}(\vec{y}^T log(\hat{\vec{y}}) + (\vec{1} - \vec{y})^T log(\vec{1} - \hat{\vec{y}}))$
     - For intuition, see above point in (1)
  3. **Gradient Descent for Linear Regression**
     - $\vec{\theta} = \vec{\theta} - (\alpha \frac{\partial J(\vec{\theta})}{\partial \vec{\theta}})^T \longrightarrow \vec{\theta} - \frac{\alpha}{m} X(\hat{\vec{y}} - \vec{y})$
     - To get an idea of how to complete step 1, which is to calculate $\frac{\partial J(\vec{\theta})}{\partial \vec{\theta}}$, look at the vectorized cost function of Linear Regression (on the formula sheet!). We can see that from here, it follows that the first substitution in the computation graph to solve $\frac{\partial J(\vec{\theta})}{\partial \vec{\theta}}$ can be done by $\hat{\vec{y}} = X^T\vec{\theta}$. Then, we can see that $\vec{d} = \vec{y} - \hat{\vec{y}}$, and hence, $J = \frac{1}{2m}\vec{d}^T\vec{d}$. The remaining computation is left to the reader.
     - Steps 2 (expanding & simplifying) and 3 (taking the transpose of $\frac{\partial J(\vec{\theta})}{\partial \vec{\theta}}$) is left to the reader. The formula sheet contains the relevant information.
  4. **Gradient Descent for Logistic Regression**
     - $\vec{\theta} = \vec{\theta} - (\alpha \frac{\partial J(\vec{\theta})}{\partial \vec{\theta}})^T \longrightarrow \vec{\theta} - \frac{\alpha}{m} X(\hat{\vec{y}} - \vec{y})$
     - **Step 1:** The goal of pre-processing is to (1) factor out $\frac{1}{m}$ via the constant multiplication rule, and (2) Apply the sum rule to split the partial derivative into more manageable components.
     - **Step 2:** To solve $\frac{\partial(\vec{y}^T log(\hat{\vec{y}}))}{\partial \vec{\theta}}$, you use the computation graph. Again, notice here how the first substitution is that $\vec{z} = X^T\vec{\theta} \longrightarrow \hat{\vec{y}} = \sigma(\vec{z}) \longrightarrow \cdots$ The key thing is to specifically remember the diagonal rule that applies for **element-wise** functions.
     - **Step 3:** To solve $\frac{\partial(\vec{1}-\vec{y})^T log(\vec{1}-\hat{\vec{y}})}{\partial \vec{\theta}}$, see the above points from step 2. It more or less follows a very similar train of thought!
     - **Step 4:** Transposing $(\frac{\partial J(\vec{\theta})}{\partial \vec{\theta}})^T$ is left to the reader as an exercise.
  5. **MLE: Cost Function (Linear Regression)**
     - We start with the observation that $p^{(i)} = (\hat{y}^{(i)})^{y^{(i)}}(1 - \hat{y}^{(i)})^{1-y^{(i)}}$. The idea is to (1) Take the log-likelihood of $p^{(i)}$ and simplify the equation, resulting in f. To get the cost function, $J = -\frac{1}{m}log(f)$
  6. **MLE: Cost Function (Logistic Regression)**
     - Refer to the MLE for Linear Regression. The only difference is that (1) $p^{(i)} = pdf^{(i)} = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y^{(i)}-\hat{y}^{(i)})^2}{2\sigma^2}}$ and (2) the cost function (J) is calculated by using least square error (LSE) $\longrightarrow J = \frac{1}{2m}\Sigma_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$

  7. **MAP** $\longrightarrow$ Refer to Lecture 25 Slides

7 Practice Exam (Iteration IV)

7.1 Calculation Questions

1. The following image shows a neural network in which all the activations are sigmoid functions. What would this neural network predict for the input $(1, 1)$? (Note: Assume that the threshold is equal to 0.5.)



$X =$	$W_2 =$	$\vdots$ $(ExtraSpace)$
$W_1 =$	$b_2 =$	$\vdots$
$b_1 =$	$Z_2 =$	$\vdots$
$Z_1 =$	$\hat{y} =$	$\vdots$
$A_1 =$	$\vdots$	$\vdots$

2. Given $z = f(x, y) = e^{xy} + x^2 + y^3 + x^y$, calculate $\frac{\partial z}{\partial x}$ and $\frac{\partial z}{\partial y}$.

3. Given that $\vec{y} = \begin{bmatrix} x_1 x_2 + x_3^2 \\ x_1 + x_2 \end{bmatrix}$ and $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, what will be the dimensions of $\frac{\partial \vec{y}}{\partial \vec{x}}$? Produce the resultant answer for $\frac{\partial \vec{y}}{\partial \vec{x}}$ below.

4. Determine $\frac{\partial (A\vec{\theta})^T (A\vec{\theta})}{\partial \vec{\theta}}$

5. Derive the Vectorized Gradient Descent for Linear Regression in Numerator Layout. Also, determine the normal equation.

Note: Your vectorized gradient descent should be as follows: $\vec{\theta} = \vec{\theta} - \alpha(\frac{\partial J(\vec{\theta})}{\partial \vec{\theta}})^T$ which simplifies to $\vec{\theta} = \frac{\alpha}{m} X(\hat{\vec{y}} - \vec{y})$.

Note: The vectorized normal equation should be: $\vec{\theta} = (X^T X)^{-1} x\vec{y}$.
(Set one of the equations you solved for above equal to 0 and isolate for $\vec{\theta}$!)

6. Given the data $\{1, 3, 3, 3, 4, 4, 6, 8\}$, what is the value of 8 after applying feature scaling via (1) normalization and (2) standardization?

7. **Feature Scaling on a Dataset:** Assume that the following dataset has undergone train-test-split and has been organized into the following form. Apply normalization onto the x and y variables.

ID	(x, y)	Normalized x	Normalized y
Training Set	$(3, 8.58)$		
Training Set	$(8, 24.87)$		
Training Set	$(1, 3.34)$		
Training Set	$(5, 14.46)$		
Training Set	$(10, 30.95)$		
Training Set	$(6, 17.35)$		
Validation Set	$(4, 12.56)$		
Validation Set	$(7, 21.01)$		
Test Set	$(2, 5.85)$		
Test Set	$(9, 26.12)$		

Hint: *The minimum and maxmimum values with relation to the featurization formulas pertain to the training set.*

8. In this example where $m = 1000$, Sam trained a model with the following confusion matrix:

$\vdots$	Predict 1	Predict 0
Actual 1	$TP = 480$	$FN = 120$
Actual 0	$FP = 240$	$TN = 160$

Calculate (a) Precision, (b) Recall, (c) F-score, (d) Accuracy

9. **Multi-Class Features:** Given the following table, calculate the weighted Gini Sum if you perform an initial split by, 'Love Content?'

Sample Index	Love Content?	Love Course?
1	*Like*	*Yes*
2	*Dislike*	*No*
3	*Like*	*No*
4	*Neutral*	*No*
5	*Like*	*Yes*
6	*Like*	*Yes*
7	*Neutral*	*No*
8	*Dislike*	*No*
9	*Like*	*No*

10. **Continuous Features:** Given the following table, calculate the weighted Gini Sum if an initial split occurs on, 'Love Content' for $x < 1.5$, $x > 1.5$. Try it for other numbers and see if you can find an optimal split value.

Sample Index	Love Content?	Love Course?
1	5	*Yes*
2	2	*No*
3	4	*No*
4	3	*No*
5	4	*Yes*
6	5	*Yes*
7	3	*No*
8	1	*No*
9	5	*No*

11. **Continuous Output**:** Given the following table, perform an initial split on 'Love Content.' Compare how this performs against an initial split on, 'Love Instructor Style.' Which one is better, and why?
(Hint: Calculate the weighted variance)

Sample Index	Love Content?	Love Instructor Style?	Love Course Rating
1	*Yes*	*Yes*	5
2	*No*	*Yes*	1
3	*Yes*	*No*	2
4	*No*	*No*	1
5	*Yes*	*Yes*	4
6	*Yes*	*Yes*	5
7	*Yes*	*No*	1
8	*No*	*Yes*	2
9	*Yes*	*No*	1

12. Given the matrix $A = \begin{bmatrix} 3 & 0 \\ 8 & 1 \end{bmatrix}$, and the vectors $\vec{a} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $\vec{b} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$, $\vec{c} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$, which of the vectors are eigenvectors of A? Of the corresponding eigenvectors, what are their eigenvalues?

13. What is the determinant of the matrix $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 3 & 3 \end{bmatrix}$. What does the determinant in this case imply?

14. Given the current centroids and their associated points, calculate the Within-Cluster Sum of Squares (WCSS).

Red Centroid $(1.725, 5.225)$	Blue Centroid $(1.725, 1.975)$
• $(1.3, 5.4)$	• $(1.3, 2.5)$
• $(1.5, 4.5)$	• $(1.5, 1.0)$
• $(1.9, 5.0)$	• $(2.0, 1.7)$
• $(2.2, 6.0)$	• $(2.1, 2.7)$

15. Calculate the Covariance of the following data.

x_1	x_2	$x_1^{(i)} - \overline{x_1}$	$x_2^{(i)} - \overline{x_2}$	$(x_1^{(i)} - \overline{x_1})(x_2^{(i)} - \overline{x_2})$
1	2			
2	4			
3	6			
4	8			
5	10			

(a) $\mu(x_1) =$

(b) $\mu(x_2) =$

(c) $\sigma^2_{x_1,x_2} = \sigma^2_{x_2,x_1} =$

16. When Sam draws a letter randomly from the bag containing "ABBCDDDD", what is the expected number of yes/no questions required to know which letter that Sam got? Use Shannon's Entropy.

- $P(A) =$
- $P(B) =$
- $P(C) =$
- $P(D) =$
- *Answer* :

17. Sam tosses 1 coin 5 times and got $\{H, T, T, H, T\}$. Sam has 2 guesses. One is that the coin is a fair coin, and the other is that the coin is biased with a probability of 0.75 to get a tail. However, Sam is 80% confident that the coin is biased. Under this assumption and the data observed, which model is more likely? Show your work.

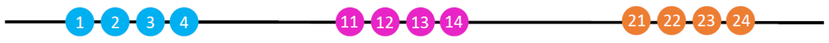
18. Sam tosses 1 coin 5 times and got $\{H, T, T, H, T\}$. Sam has 2 guesses. One is that the coin is a fair coin, and the other is that the coin is biased with a probability of 0.75 to get a tail. Sam has no preference between these two models. Which model is more likely? Show your work.

19. Assume that we are working with Maximum Likelihood Estimation. Then, given that $L(p) = p^4(1-p)^6$, find the value for p that maximizes the likelihood function.

20. Using the Maximum Likelihood Estimation (MLE) interpretation of Logistic Regression, derive its cost equation, J.

21. Using the Maximum Likelihood Estimation (MLE) interpretation of Linear Regression, derive its cost equation, J.

22. What is the Silhouette Coefficient of the sample whose value is 4, given the following clustering?



23. Given the information about two clusters:

Green Cluster $(-6, -2.67)$	Red Cluster $(5, 4.75)$
• $(-7, -3)$	• $(4, 4)$
• $(-6, -2)$	• $(4, 7)$
• $(-5, -3)$	• $(6, 3)$
	• $(6, 5)$

(a) What is the Silhouette Score of the sample whose value is (-6, -2), given the following clustering?

(b) Given that we have two centroids (and we have the points assigned to the centroid in the table), what is the within-cluster sum of squares?

7.2 True / False

1. In general, a cluster that has a small sum of squares (WCSS) is more compact than a cluster that has a large sum of squares (T/F)
2. A consequence of the Elbow Method when choosing the optimal number of centroids is that sometimes there is no clear turning point (T/F)
3. When the feature space is larger, overfitting is less likely (T/F).
4. Non-parametric models do not have parameters.(T/F)
5. For a fixed size of the training and test set, increasing the complexity of a model always leads to a reduction of the test error. (T/F)
6. It is not a good machine learning practice to use the test set to help adjust the hyperparameters of your model.(T/F)
7. Given a 200×200 pixel image, it would be advisable to have an input layer of 40,000 such features (T/F).
8. Your peer Sam the robot wants to implement a classifier to predict the type of fruit in grocery stores. One of the features is the taste of the fruit that requires the fruit to be cut. This a reasonable proposal with sound methodologies. (T/F)
9. An application does not need to address every single problem. It can, and should focus on one small problem in a domain (T/F).
10. The normal equation for logistic regression is more optimal than performing Gradient Descent or Newton's Method (T/F)
11. Hidden layers of a neural network do not have varying size (T/F)
12. The most common architecture of a neural network occurs when all of the layers use ReLU except the last one which uses sigmoid (T/F)
13. A very large learning rate will lead to overshooting an the result can be bizarre. However, it is possible to find a local minima in the end (T/F)
14. Feature scaling must be done prior to training. The benefits of feature scaling are that they prevent certain features from dominating and that it allows for the model to be trained faster (low number of iterations to converge...) (T/F)
15. Sam splits his data into Train/Dev/Test sets. Training his Random Forest model, he found that the model performed well on the training set and validation set, but poorly on the testing set. Sam does not need to start over again. (T/F).
16. A cluster with a larger WCSS is a better solution (more compact) (T/F)
17. A point with a larger Silhouette coefficient is more preferred. This means that it is very compact within the cluster to which it belongs and far away from the other clusters. (T/F)
18. If a feature has values ranging within $-100 \leq x \leq 100$, it requires rescaling. (T/F)

7.3 Fill-in-the-Blanks

1. When you design your project, you will need to think about each step, including ______, ______, ______, and seeing whether they are practical.
2. The ______ is a limitation that exists in every paper.
3. The goal for MLE is to find the ______ and ______. Then, it is critical to find the ______ function based on the observations seen and ______ the probability. Lastly, you want to find the result ______.
4. The decision boundary for Logistic Regression classifies points as positive if the estimated value is ______ the threshold and negative if the estimated value is ______ the threshold.
5. Discriminant models use a ______ while Generative models use ______ and ______ as their characteristics to classify points.
6. The smaller array is ______ to a larger array so that they have ______ shapes.
7. When performing non-vectorized Gradient Descent, we must first find the updated value for all parameters and save them to ______. Then, you must update the values ______, otherwise, the updated value of the next parameter will be different if we assign new values to the previous parameters.
8. In the non-convex approach for weight initialization (i.e., Neural Network), we want to use ______ as it avoids going to the same ______ and improves ______.
9. Random Forest overcomes data sensitivity by making predictions on multiple trees through ______.
10. If the determinant is 0, this means that there are ______ solutions.
11. The goal of PCA is to ______ variance of the points, post-projection and ______ residuals.
12. If we are to visualize a distribution, the higher the variance, the ______ the spread and the ______ the peak is.
13. The Eigenvectors of a Covariance Matrix are the ______.
14. The goal of ______ is to get the model's features in a ______ range (i.e., $-1 \leq x \leq 1$). There two common methods are ______ and ______.
15. To apply feature-scaling (normalization) to a dataset (train-test-split), the ______ and ______ values for the equations must be with relation to the ______ set.
16. If a Neural Network is overfitting, this means that the structure is too ______. It must be ______ by reducing ______. The opposite holds if the Neural network is underfitting.
17. SGD/BGD defines how to get to the minima. Regularization defines ______________________________.
18. ______ is used to determine the optimal choice of centroid while ______, ______, and ______ can be used to determine the optimal number of centroids.

8 Additional Resources

1. **Calculus Review**
    - Comprehensive Derivatives Cheat Sheet ($\approx SC/MATH$ 1300)
    - Tutorial on Partial Derivatives ($\approx SC/MATH$ 2015)
2. **Linear Algebra Review**
    - Introduction to Linear Algebra Notes by Professor Dawkins ($\approx SC/MATH$ 1021)
    - Properties of Matrix Operations (Cheat Sheet)
    - Comprehensive Lectures by Professor Strang
    - Exercises and Tutorials by PatrickJMT
    - 3Blue1Brown Tutorials
3. **Probability** ($\approx SC/MATH$ 2030)
    - Probability Cheat Sheet
4. **General Machine Learning**
    - Kaggle Tutorials on Machine Learning Techniques & Applications
    - Grokking Machine Learning (Textbook) $\longrightarrow$ Really helpful explanations and exercise problems that covers content up to the course midterm.
    - StatsQuest YouTube Videos $\longrightarrow$ Very helpful explanations and guided walk-through of post-midterm material.
    - Mining of Massive Datasets (Textbook) $\longrightarrow$ A pretty helpful textbook that has practical examples related to some of the content taught in this class. In particular, it was helpful to read (or skim) Chapters 7, 11, 12, & 13.
5. **Miscellaneous**
    - CSE446 Resources
    - CS 229 Resources
    - CPSC 340 Resources
    - COMP451 Resources
    - CS411 Resources