Indian Institute of Technology Delhi
COL774: Machine Learning
Fall 2024-2025
Practice Problems

Harshit Goyal

November 2024

1 About

The most recent version of this document can be found on the project repository. We encourage you to use this work further. Please see the project repository for the BiBTex citation.

This problem set was written by me, Harshit Goyal during the Teaching Assistantship for the COL774 Fall Machine Learning course 2024-2025, offered by Prof Rahul Garg. We thank the students in this class for feedback on many of these problems.

Contents

2 Introduction

1. You may refer to [Petersen and Pedersen, 2021] for matrix derivatives for the upcoming problems.
2. Below notations are consistent throughout the document.
3. The dataset is denoted as $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ where $\mathbf{x}^{(i)} \in \mathcal{R}^n$ and $y^{(i)} \in \mathcal{R}^n$ for Linear Regression. Further $y^{(i)} \in \{0, 1\}$ for Logistic Regression.
4. $\mathbf{w} \in \mathcal{R}^n$ and $\mathbf{w} = \begin{bmatrix} w_1 & w_2 & w_3 & \dots & w_n \end{bmatrix}^T$.
5. For Linear Regression, $\hat{y}^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)} + b$.
6. For Logistic Regression, $\hat{y}^{(i)} = g(\mathbf{w}^T\mathbf{x}^{(i)} + b)$, where $g : \mathcal{R} \to \mathcal{R}$ is an activation function.

7.

$$
\begin{aligned}
\mathcal{L}_{SSE}(\mathbf{w}, b) &= \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2 \\
\mathcal{L}_{Logistic}(\mathbf{w}, b) &= -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \\
\mathcal{L}_2(\mathbf{w}, b) &= \mathbf{w}^T\mathbf{w} \\
\mathcal{L}_1(\mathbf{w}, b) &= \sum_{i=1}^{n} |w_i|
\end{aligned}
$$

8. When the bias term is taken as a feature, the absorb b in $\mathbf{w}$ and the loss functions in item 7 can be denoted as $\mathcal{L}_{SSE}(\mathbf{w})$, $\mathcal{L}_{Logistic}(\mathbf{w})$, $\mathcal{L}_2(\mathbf{w})$, $\mathcal{L}_1(\mathbf{w})$, respectively.

3 Regression

3.1 GFLOPS Calculation

For matrix multiplication $\mathbf{Y} = \mathbf{W}\mathbf{X}$, where $\mathbf{X} \in \mathcal{R}^{m\times n}$ matrix and $\mathbf{W} \in \mathcal{R}^{n\times 1}$:

1. **Using a Python implementation with nested for loops:**
   - Calculate the GFLOPS (Giga-Floating Point Operations Per Second) for the matrix multiplication operation. Determine the total number of floating-point operations involved, including both multiplication and addition operations, divide this by the time taken to execute the matrix multiplication, and convert the result to GFLOPS by dividing by 10^9.
2. **Using Python's `numpy` library:**
   - Similarly, calculate the GFLOPS for the matrix multiplication operation using `numpy`'s built-in matrix multiplication function.
3. **Comparison and Visualization:**
   - Plot a graph with the x-axis representing m (considering constant n) and the y-axis representing GFLOPS to illustrate the performance difference between the two implementations.

3.2 Binary Classification

For binary classification using the sigmoid function, write the gradient descent update rule for the following methods:

1. **Stochastic Gradient Descent (SGD):**
   Update the model parameters using one data point at a time.
2. **Batch Gradient Descent:**
   Update the model parameters using the entire dataset.
3. **Mini-Batch Gradient Descent:**
   Update the model parameters using a subset of the dataset (a mini-batch).

Use the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$.

3.3 Regularized Linear Regression

For Linear Regression, obtain $\mathbf{w}^*$ that minimizes $\mathcal{L}_{SSE}(\mathbf{w}) + \lambda\mathcal{L}_2(\mathbf{w})$. $\lambda > 0$ is fixed.

3.4 Regression Line passes through mean

For Linear Regression, show that the line $(\mathbf{w}^*, b^*)$ minimizing $\mathcal{L}_{SSE}(\mathbf{w}, b)$ always satisfies

$$(\mathbf{w}^*)^T(\frac{1}{m}\sum_{i=1}^{m}\mathbf{x}^{(i)}) + b^* = \frac{1}{m}\sum_{i=1}^{m} y^{(i)}$$

Hint: See the equation obtained on setting,

$$\frac{\partial}{\partial b}\mathcal{L}_{SSE}(\mathbf{w}, b)\big|_{(\mathbf{w}^*, b^*)} = 0$$

3.5 Convex Functions

A function $f : \mathcal{R}^n \to \mathcal{R}$ is called convex if for all $x_1, x_2 \in \mathcal{R}^n$ and $\theta \in [0, 1]$, the following inequality holds:

$$f(\theta x_1 + (1 - \theta)x_2) \leq \theta f(x_1) + (1 - \theta)f(x_2)$$

This property ensures that the function lies below the straight line connecting any two points on its graph. Convex functions play a crucial role in optimization problems, as they ensure s they ensure that any local minimum is also a global minimum.

3.5.1 Discuss convexity

1. $f(x) = e^{ax}$, $f(x) = \log(x)$. $f(x) = x^3$ in their respective domains.

   (Use the second derivative of each function)

   For the below parts, do it by computing the hessian.

2. $f(\mathbf{x}) = \|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$, where $\mathbf{x} \in \mathcal{R}^n$.

   (This is convex but you can't do it by computing the Hessian as the function is not differentiable at $\mathbf{x} = \mathbf{0}$. You'll need to prove using the definition in Convex Functions and the Cauchy–Schwarz Inequality.)

3. $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$, where $\mathbf{A} \in \mathcal{R}^{n\times n}$, $\mathbf{x} \in \mathcal{R}^n$. $\mathbf{A}$ is Diagonal.

   (Convex if and only if all entries of $\mathbf{A}$ are non-negative)

4. $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$, where $\mathbf{A} \in \mathcal{R}^{n\times n}$, $\mathbf{x} \in \mathcal{R}^n$. $\mathbf{A}$ is Symmetric.

   (Convex if and only if all eigenvalues of $\mathbf{A}$ are non-negative)

5. $f(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{A}\mathbf{y}$, where $\mathbf{A} \in \mathcal{R}^{m\times n}$, $\mathbf{x} \in \mathcal{R}^m$, and $\mathbf{y} \in \mathcal{R}^n$.

   ($f(\mathbf{x}, \mathbf{y})$ is convex $\iff$ $\mathbf{A} = 0$)

3.5.2 Convexity of linear Combination

Using the definition of convexity given above, show that, if $f : \mathcal{R}^n \to \mathcal{R}$ and $g : \mathcal{R}^n \to \mathcal{R}$ are convex, and $a, b > 0$, the function $af + bg : \mathcal{R}^n \to \mathcal{R}$ is convex, i.e., for any $\mathbf{x}, \mathbf{y} \in \mathcal{R}^n$ and $\lambda \in [0, 1]$,

$$(af + bg)(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda(af + bg)(\mathbf{x}) + (1 - \lambda)(af + bg)(\mathbf{y})$$

3.6 Convexity of Regression Objectives

By computing the hessian show that, $\mathcal{L}_{SSE}(\mathbf{w}), \mathcal{L}_{Logistic}(\mathbf{w}), \mathcal{L}_2(\mathbf{w})$ are convex. From the definition of convexity in Convex Functions show that $\mathcal{L}_1(\mathbf{w})$ is convex (as it's not differentiable. Hint: Use the triangle inequality).
Using Convexity of linear Combination do you see that why all four below functions are convex?

$$\{\mathcal{L}_{SSE}(\mathbf{w}), \mathcal{L}_{Logistic}(\mathbf{w})\} + \lambda\{\mathcal{L}_2(\mathbf{w}), \mathcal{L}_1(\mathbf{w})\}$$

3.7 Probabilistic Interpretation of Regularization

In class we looked at the Maximum Likelihood Estimate of $\mathbf{w}$ for the Linear Regression problem. We considered $\mathbf{w}$ as parameter and the optimum $\mathbf{w}_{MLE}$ (for a single example) was defined as

$$\mathbf{w}_{MLE} = \arg\max_{\mathbf{w}} P(y^{(i)}|\mathbf{x}^{(i)}; \mathbf{w})$$

Now, let's treat $\mathbf{w}$ as a Random Variable, distributed as (this is called a **priori** on $\mathbf{w}$)

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$$

where $\mathbf{\Sigma} = diag\{\lambda_1, \lambda_2..\lambda_n\}$ and $\lambda_1, \lambda_2.....\lambda_n > 0$. We now define $\mathbf{w}_{MAP}$ (MAP stands for **M**aximum **a** **P**osteriori) as

$$\mathbf{w}_{MAP} = \arg\max_{\mathbf{w}} P(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w})P(\mathbf{w}) \tag{1}$$

Now, similar to the MLE case, we assume, over all examples, $y^{(i)} - \hat{y}^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}$ are $i.i.d$ distributed as

$$y^{(i)} - \hat{y}^{(i)}|\mathbf{x}^{(i)}, \mathbf{w} \sim \mathcal{N}(0, \sigma^2)$$

a) Simplify Equation 1 for the entire dataset combined now, and show solving $\mathbf{w}_{MAP}$ is similar to Linear Regression with L_2 Regularization on $\mathbf{w}$.

b) Obtain explicitly, the **priori** on $\mathbf{w}$ that will lead to the L_1 Regularization of $\mathbf{w}$ (keep different penalty coefficient for each of the n components of $\mathbf{w}$). Does this distribution have a name?

Solution to above part. It's called the **Laplace Distribution**.

$$p(w_1, w_2...w_n) = \prod_{i=1}^{n} \frac{\lambda_i}{2} e^{-\lambda_i |w_i|}$$

3.8 Logistic Regression in Perfectly Linearly Separable case

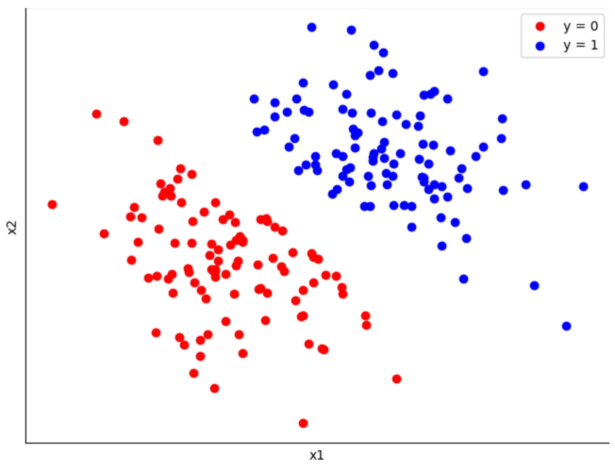

Figure 1: Data for Logistic Regression, features x_1 and x_2

Consider Logistic Regression on **Perfectly Linearly Separable** data, as shown in Figure 1. You may not assume that there are just 2 features as shown in the figure.

a) What is the condition (possibly implicit) for a data to be linearly separable, when the features $\mathbf{x}^{(i)} \in \mathcal{R}^n$.

b) Consider solving for the decision boundary $(\mathbf{w}, b)$ for such data. Show what $\forall \epsilon > 0 \;\; \exists (\mathbf{w}_\epsilon, b_\epsilon)$ such that $\mathcal{L}_{Losgistic}(\mathbf{w}, b) < \epsilon$.

c) Consider running gradient descent, to obtain $(\mathbf{w}, b)$. Assuming the learning rate is small, will the algorithm ever converge in $(\mathbf{w}, b)$? (Answer: $\mathbf{w}$ and b diverge. You may skip it as it's difficult to prove.)

3.9 Logistic Regression for multiple classes

Consider r class classification problem. We build a model that predicted an input belonging to each of the r classes (similar to Logistic Regression).

The model is parametrized by r weight vectors, $\mathbf{w}_1, \mathbf{w}_2 ... \mathbf{w}_r$. For an input $\mathbf{x}^{(i)}$, the model is evaluated as,

$$z_j^{(i)} = \mathbf{w}_j^T \mathbf{x}^{(i)} \quad \text{for j } = 1, 2....r \tag{2}$$

$$\hat{y}_j^{(i)} = \frac{e^{z_j}}{\sum_{k=1}^{r} e^{z_k}} \quad \text{for j } = 1, 2....r \tag{3}$$

Equation 3 is the Softmax function. (Why **Soft**max?) The loss function, for a single example (i), $y^{(i)} \in \{1, 2....r\}$ is

$$\mathcal{L}^{(i)}(\mathbf{w}_1, \mathbf{w}_2 ... \mathbf{w}_r) = -\sum_{j=1}^{r} \mathbb{1}_{\{y^{(i)}=j\}} \log(\hat{y}_j^{(i)}) \tag{4}$$

Where $\mathbb{1}_{(.)}$ is the indicator function.

$$\mathbb{1}_{(x)} = \begin{cases} 1 & \text{if } x \text{ is True} \\ 0 & \text{if } x \text{ is False} \end{cases}$$

Equation 4 is know is the **Cross Entropy Loss function**. Now, define

$$\mathcal{L}(\mathbf{w}_1, \mathbf{w}_2 ... \mathbf{w}_r) = \sum_{i=1}^{m} \mathcal{L}^{(i)}(\mathbf{w}_1, \mathbf{w}_2 ... \mathbf{w}_r)$$

a) Is $\mathcal{L}(\mathbf{w}_1, \mathbf{w}_2 ... \mathbf{w}_r)$ convex? (Answer: Yes! But you may skip it as the proof gets very involved.)

b) Obtain explicitly, $\nabla_{\mathbf{w}_1}\mathcal{L}, \nabla_{\mathbf{w}_2}\mathcal{L} ..... \nabla_{\mathbf{w}_r}\mathcal{L}$. Hint: Start by writing $\mathcal{L}$ in terms of $(\mathbf{w}_1, \mathbf{w}_2 ... \mathbf{w}_r)$.

3.10 LASSO for feature selection

[Goodfellow et al., 2016] have mentioned in the chapter **Regularization for Deep Learning**,

"the well known LASSO (Tibshirani, 1995) (least absolute shrinkage and selection operator) model integrates an L_1 penalty with a linear model and a least-squares cost function. The L_1 penalty causes a subset of the weights to become zero, suggesting that the corresponding features may safely be discarded."

We'll see in this problem how. Let $x \in \mathcal{R}$. Let $a > 0, \lambda > 0, b, c, x \in \mathcal{R}$ and

$$\begin{aligned} w^* &= \arg\min_x ax^2 + bx + c \\ w(\alpha) &= \arg\min_x ax^2 + bx + c + \alpha|x| \end{aligned}$$

a) Compute $w(\alpha)$ in terms of α, a, w^*. What do you conclude?

b) Now let's consider multiple features. Let $\mathbf{A} = diag(a_1, a_2 \ldots, a_n), a_i > 0 \;\; \forall i$ and $\mathbf{b} \in \mathcal{R}^n$.

$$\begin{aligned} \mathbf{w}^* &= \arg\min_{\mathbf{x} \in \mathcal{R}^n} \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \\ \mathbf{w}(\alpha) &= \arg\min_{\mathbf{x} \in \mathcal{R}^n} \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c + \alpha \left\|\mathbf{x}\right\|_1 \end{aligned}$$

Show that

$$\mathbf{w}(\alpha)_i = sign(\mathbf{w}_i)\max(0, |w_i| - \frac{\alpha}{a_i})$$

What happens to components of $\mathbf{w}(\alpha)$ corresponding to smaller diagonal entries of $\mathbf{A}$? How can this be used for feature selection?

3.11 Limit of Locally Weighted Linear Regression

The problem idea was suggested to me by Prof. Rahul Garg. Recall Locally Weighted Linear Regression done in class. We'll understand what happens in the limit, hyper-parameter $\sigma \to \infty$ and $\sigma \to 0^+$. Consider points $x_1 < x_2 < x_3 .... x_m \in \mathcal{R}$ and $\lambda > 0$ be fixed. We need to regress over points $(x_1, y_1), (x_2, y_2) \ldots (x_m, y_m)$. The loss function, for a point $x \in \mathcal{R}$ is

$$\begin{aligned} \alpha_{\sigma,i}(x) &= \frac{e^{-\frac{(x-x_i)^2}{2\sigma^2}}}{\sum_{j=1}^m e^{-\frac{(x-x_j)^2}{2\sigma^2}}} \\ \hat{y}_i &= wx_i + b \\ \mathcal{L}_\sigma(x, w, b) &= \frac{1}{m}\sum_{i=1}^m \alpha_{\sigma,i}(x)(y_i - \hat{y}_i)^2 + \lambda w^2 \end{aligned}$$

Denote

$$\begin{aligned} w_\sigma(x), b_\sigma(x) &= \arg\min_{w,b} \mathcal{L}_\sigma(x, w, b) \\ f_\sigma(x) &= w_\sigma(x)x + b_\sigma(x) \end{aligned}$$

a) Obtain $f_\infty(x) = \lim_{\sigma \to \infty} f_\sigma(x)$.

b) Obtain $f_0(x) = \lim_{\sigma \to 0^+} f_\sigma(x)$.

3.12 Unique Minima

In class we discussed that the problem of Linear Regression has a unique optimum. We'll try to prove that formally and see what happens to stationary points when convexity is not guaranteed. Consider the function

$$\mathbf{f}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} + c \tag{5}$$

Where $c \in \mathcal{R}$, $\mathbf{x} \in \mathcal{R}^n$, $\mathbf{b} \in \mathcal{R}^n$, $\mathbf{A} \in \mathcal{R}^{n\times n}$ and is Symmetric. All eigenvalues of $\mathbf{A}$ are positive. We define $\mathbf{x}^* \in \mathcal{R}^n$ such that $\nabla_{\mathbf{x}} f|_{\mathbf{x}=\mathbf{x}^*} = 0$.

a) Show that $\mathbf{A}$ is invertible and hence $\mathbf{x}^*$ exists uniquely.

$$\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}^*) > 0 \ \ \forall \mathbf{x} \in \mathcal{R}^n - \{\mathbf{x}^*\} \tag{6}$$

b) We'll now prove Equation 6. Set $\mathbf{x} = \mathbf{x}^* + \epsilon\mathbf{z}$, $\mathbf{z}^T\mathbf{z} = 1$, $\epsilon > 0$ in the equation, and show that

$$\mathbf{f}(\mathbf{x}^* + \epsilon\mathbf{z}) - \mathbf{f}(\mathbf{x}^*) = \frac{\epsilon^2}{2}\mathbf{z}^T\mathbf{A}\mathbf{z} \tag{7}$$

Assume the fact, "Every $n \times n$ Real Symmetric Matrix is Diagonalizable and has a set of n Orthonormal Eigenvectors". Express $\mathbf{z}$ as

$$\mathbf{z} = \sum_{i=1}^{n} a_i \mathbf{v}_i$$

Where $\mathbf{v}_i^T\mathbf{v}_i = \delta_{ij}$ and λ_i be the Eigenvalue of $\mathbf{A}$ corresponding to $\mathbf{v}_i$. Show that

$$\mathbf{f}(\mathbf{x}^* + \epsilon\mathbf{z}) - \mathbf{f}(\mathbf{x}^*) = \frac{\epsilon^2}{2}\sum_{i=1}^{n} a_i^2\lambda_i \tag{8}$$

c) Now, let all eigenvalues of $\mathbf{A}$ be non-zero and not all of the same sign. Obtain **explicitly**, atleast one $\mathbf{z} \in \mathcal{R}^n$ such that $\|z\|_2 = 1$ and $\mathbf{f}(\mathbf{x}^* + \epsilon\mathbf{z}) - \mathbf{f}(\mathbf{x}^*) < 0$ for some $\epsilon > 0$.

d) Show that the Linear Regression objective can be expressed Equation 5. Does the condition, "$\mathbf{A}$ is Symmetric. All eigenvalues of $\mathbf{A}$ are positive" assumed here always hold in linear Regression? Clearly describe the condition on the data when this isn't true. We'll derive these conditions in Multiple Minimas.

3.13 Multiple Minimas

In this problem we'll re what happens when $\mathbf{A}$ in Unique Minima has some eigenvalues 0. Consider the data matrices, $\mathbf{X} \in \mathcal{R}^{m\times n}$ and $\mathbf{Y} \in \mathcal{R}^m$.

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}^{(1)T} \\ \mathbf{x}^{(2)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{pmatrix} \quad \mathbf{Y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

Absorb the bias term in Linear Regression into $\mathbf{x}^{(i)}$'s. We derived the condition for the optimal $\mathbf{w}^*$ as (derive it yourself if you don't remember this),

$$\mathbf{X}^T\mathbf{X}\mathbf{w}^* = \mathbf{X}^T y \tag{9}$$

When $\mathbf{X}^T\mathbf{X}$ is invertible, we get

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y \tag{10}$$

We'll now examine precisely when is $\mathbf{X}^T\mathbf{X}$ not invertible.

a) Show that Equation 9 has atleast one solution.

b) Show that all $\mathbf{w}^*$ satisfying Equation 9 lead to the same $\mathcal{L}_{SSE}(\mathbf{w}^*)$ value.

c) Show that a square matrix $\mathbf{A} \in \mathcal{R}^{n\times n}$ is not invertible if and only if $\mathbf{A}\mathbf{v} = \mathbf{0}$ for some non-zero $\mathbf{v} \in \mathcal{R}^n$.

d) For any $\mathbf{v} \in \mathcal{R}^n$, show that

$$\mathbf{X}^T\mathbf{X}\mathbf{v} = \mathbf{0} \iff \mathbf{X}\mathbf{v} = \mathbf{0} \tag{11}$$

If you need a hint, see this[1]. If you're up for a challenge (this will not be used in later parts), let $\mathbf{C} \in \mathcal{R}^{m\times m}$ be a fixed, Symmetric Positive Definite matrix, show that

$$\mathbf{X}^T\mathbf{C}\mathbf{X}\mathbf{v} = \mathbf{0} \iff \mathbf{X}\mathbf{v} = \mathbf{0}$$

e) Now let $\mathbf{X}^T\mathbf{X}$ be non-invertible. Using the above part, what do you conclude about the columns of $\mathbf{X}$, the data matrix (note that there's also the column of all 1's in $\mathbf{X}$). Let $\mathbf{w}_1^*$ be one $\mathbf{w}$ minimizing $\mathcal{L}_{SSE}(\mathbf{w})$, obtain another.

f) Continuing item e), if you have to choose a subset of the initial n features, for regression, without losing any information, how would you do that?

g) Following, the above part, can you realize why assuming $\mathbf{A}$ in Unique Minima to be positive definite, instead of positive semi-definite, is reasonable and doesn't lead to loss of generality?

h) Redo all above parts for Weighted Linear Regression, with example-wise weights, $u_1, u_2 ... u_m$, all positive. You'll see all these results hold true for this case too!

3.14 Is this Linear Regression?

Consider this regression problem. Here $n > 4$ and the prediction

$$\hat{y}^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)}$$

We have 4 **Linearly Independent Constraints** on $\mathbf{w}$ represented as

$$\mathbf{M}\mathbf{w} = \mathbf{0}$$

i.e. $\mathbf{M} \in \mathcal{R}^{4\times n}$ and $rank(\mathbf{M}) = 4$. Under the above constraints obtain the optimum value of $\mathbf{w}$ that minimizes these loss functions.

a)

$$\mathcal{L}_{SSE}(\mathbf{w})$$

[1]For $\mathbf{z} \in \mathcal{R}^n$, $\mathbf{z}^T\mathbf{z} = 0 \iff \mathbf{z} = \mathbf{0}$.

b)

$$\mathcal{L}_{SSE}(\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{A} \mathbf{w}$$

Where $\lambda \in \mathcal{R}^+$ and $\mathbf{A}$ is a **Symmetric Positive Definite Matrix**, are **constants**.

Note that it is not intended that one uses Lagrangian Multipliers to solve this.

4 Neural Networks

4.1 Backpropagation in Neural Networks

Following the terminology used in class. Write Backpropagation for Neural Networks yourself and hence, the update rule for each weight and bias in the network. Consider the r class classification problem with Softmax in the last layer and the Cross Entropy loss function.

4.2 Backpropagation in CNN's

Can you point out what makes backpropagation in CNN's more difficult/ worrisome that vanilla neural networks? **Parameter Sharing!** The weight ($\mathbf{w}$) and bias (b) matrices are shared by every "patch" in a feature map. Consider the input feature map $\mathbf{x}$ of size $h_1 \times w_1 \times d_1$. Denote the loss function on the final output as $\mathcal{L}$. We have the following layers.

- Layer 1: The convolution filter of height and width a each with stride s runs on $\mathbf{x}$. Assume s divides $(h_1 - a)$ and $(w_1 - a)$. The output feature map, called $\hat{\mathbf{z}}$ has depth d_2.
  - Write down the exact dimensions of $\mathbf{w}$, b and $\hat{\mathbf{z}}$.
  - Write the expression for $\hat{\mathbf{z}}_{ijk}$ for each feature in $\hat{\mathbf{z}}$.
  - Assuming for all i, j, k in the size if $\hat{\mathbf{z}}$, $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{z}}_{ijk}}$ is known, obtain $\frac{\partial \mathcal{L}}{\partial w_{pqrk}}$, $\frac{\partial \mathcal{L}}{\partial b_{rk}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{x}_{abr}}$.
- Layer 2: Assume $\hat{\mathbf{z}}$ has size $h_2 \times w_2 \times d_2$. A max pooling filter of shape $a \times a$ with stride a runs over $\hat{\mathbf{z}}$ to produce $\mathbf{z}$. Assume a divides h_1 and w_1.
  - Write down the exact dimensions of $\mathbf{z}$ and the expression for each $\mathbf{z}_{ijk}$ in terms of $\hat{\mathbf{z}}$.
  - Assuming for all i, j, k in the size of $\mathbf{z}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{ijk}}$ is known, obtain $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{z}}_{pqk}}$.
- Layer 3: Assume $\mathbf{z}$ has size $h_3 \times w_3 \times d_3$. The Sigmoid activation is applied to $\mathbf{z}$ pointwise, to obtain $\mathbf{y}$.
  - Write down the exact dimensions of $\mathbf{y}$ and the expression for each $\mathbf{y}_{ijk}$ in terms of $\mathbf{z}$.
  - Assuming for all i, j, k in the size of $\mathbf{y}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{y}_{ijk}}$ is known, obtain $\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{pqk}}$.
- Realize that "sliding" the Convolution Filter over the input feature map is just for intuition. All features in the output feature map can be computed in parallel.

4.3 Building complex Decision Boundaries

We'll denote a neuron with n inputs as f_n i.e. $f_n : \mathcal{R}^n \to \{0, 1\}$ and f_n is designed using weight $\mathbf{w} \in \mathcal{R}^n$ and bias b. For $\mathbf{x} \in \mathcal{R}^n$,

$$f_n(\mathbf{x}) = \mathbb{1}_{\{\mathbf{w}^T\mathbf{x}+b>0\}}$$

For the below problems we'll ignore classifying on exactly the Decision Boundary.

a) Using a single neuron, implement the the OR and AND gates i.e. when $\mathbf{x} \in \{0, 1\}^n$, $f_n(\mathbf{x})$ should output the corresponding boolean logic.

b) Realize that using a single neuron, f_n you can implement every possible linear decision boundary in $\mathcal{R}^n$. Note that for a single geometric hyperplane in $\mathcal{R}^n$, there are two possible decision boundaries (See Figure 2 and Figure 3).

c) If you want to partition $\mathcal{R}^2$ into different regions using straight lines, each classified as one of $\{0, 1\}$, you can draw these lines and use boolean logic to model on which side a point in $\mathcal{R}^2$ lies w.r.t each line (one of $\{0, 1\}$). Taking ideas from the above parts, draw a 2 Layer network with atmost 3 neurons that can draw every decision boundary shaped like Figure 5 and Figure 4. Can the same network be use to draw decision boundaries Figure 2 and Figure 3. Can this network draw Figure 6?

d) Show why you can't implement the XOR gate using a single neuron. Now, use two layers of neurons (atmost 3 neurons in total) to implement XOR, in case of $\mathbf{x} \in \{0, 1\}^2$.

e) Using atmost 3 layers and 5 neurons, implement all decision boundaries like Figure 6. Will this also represent all decision boundaries Figure 2 through Figure 6.

f) Think about the minimum number of layers and neurons needed to implement any boolean function of n variables. Using the ideas above, can you design a Neural Network that can perfectly classify a m points in $\mathcal{R}^2$, without knowing their positions and labels beforehand.

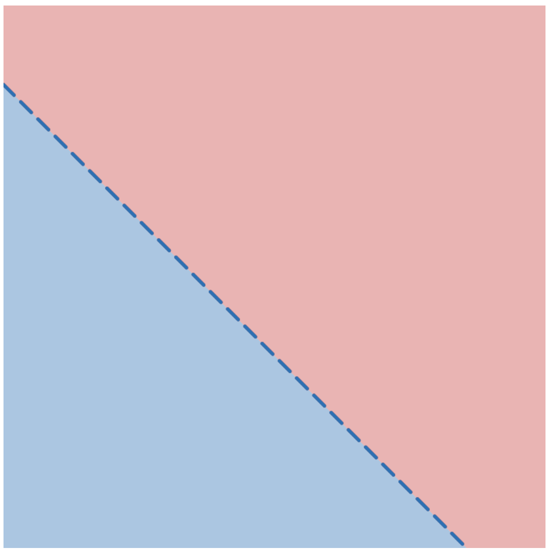
Figure 2: Boundary 1

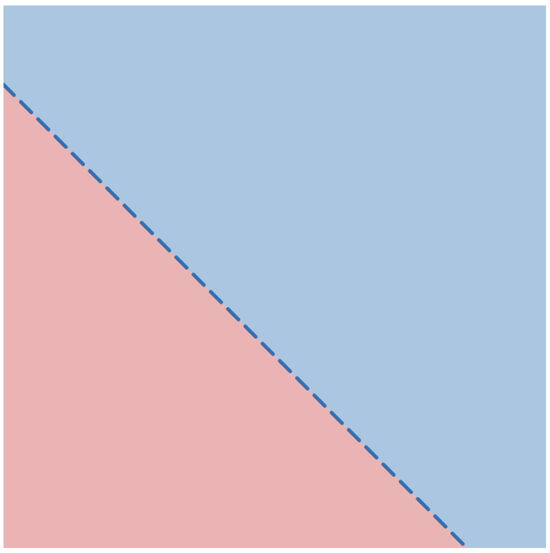
Figure 3: Boundary 2

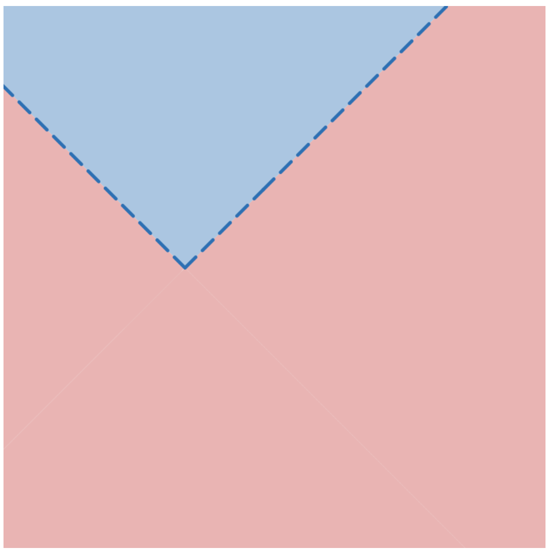
Figure 4: Boundary 3

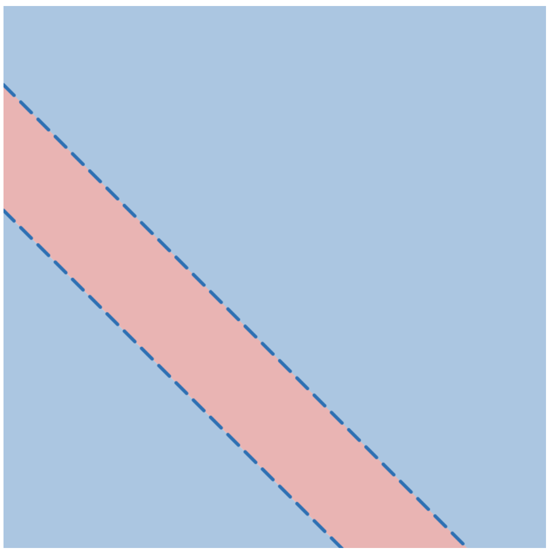
Figure 5: Boundary 4

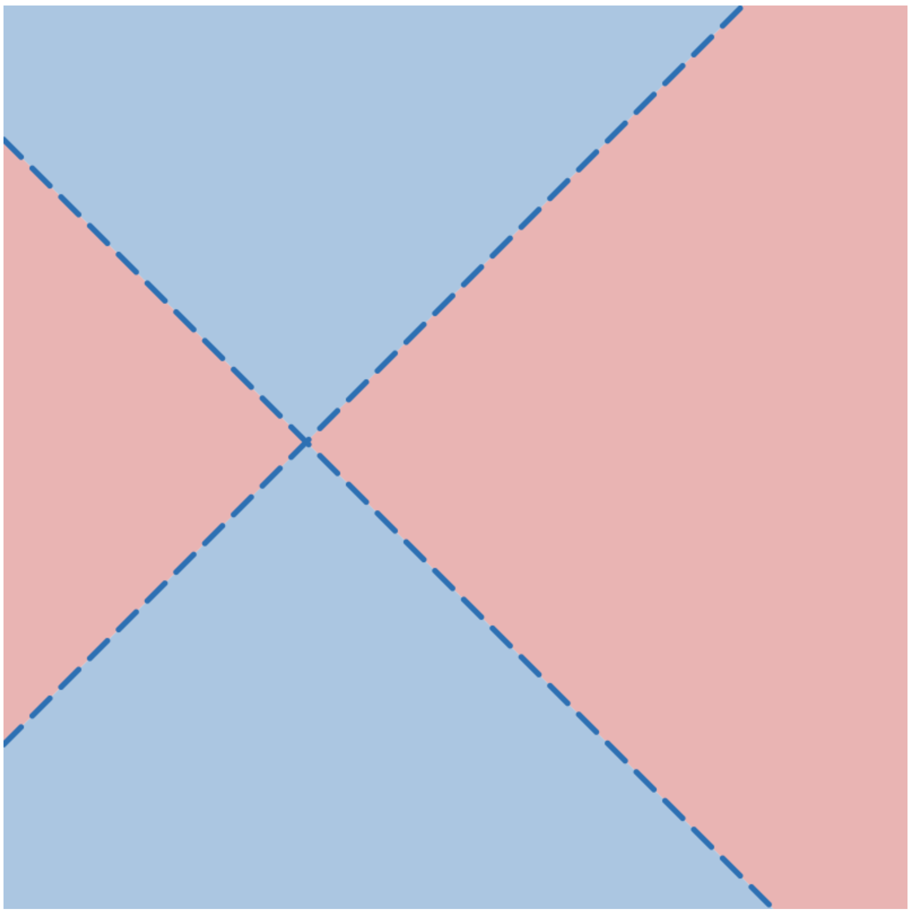

Figure 6: Boundary 5

4.4 Dropouts in Linear Regression

[Goodfellow et al., 2016] have mentioned in the chapter **Regularization for Deep Learning**,

"when applied to linear regression, dropout is equivalent to L_2 weight decay, with a different weight decay coefficient for each input feature"

We'll prove this here. Consider data, $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$, $\mathbf{w}$, b with dimensions as described in Introduction. We'll apply dropouts on the input features $\mathbf{x}_j$ to the model. We'll call our model $\mathbf{M}$. In training mode, for each forward pass each of the n features is independently used with **fixed** probability $p \in (0, 1)$, otherwise **dropped**. During evaluation, all the features are used but the corresponding weight is scaled according to $w_i \to pw_i$, as done in dropouts to prevent inflating the activation (here Identity).

a) While training, notice that the output $\mathbf{M}(\mathbf{x})$ is a random variable. Show that the model output can be expressed precisely as

$$\mathbf{M}(\mathbf{x}) = \mathbf{w}^T(\mathbf{x} \odot \mathbf{d}) + b$$

Where $\odot$ represents the element-wise product and $\mathbf{d} \in \{0, 1\}^n$ is a random vector.

b) Obtain the probability distribution of $\mathbf{d}$. Hence compute $\mathbf{E}[\mathbf{d}]$ and $\mathbf{E}[\mathbf{d}\mathbf{d}^T]$.

c) Prove these trivial properties, $\mathbf{x}$ is a constant.

$$(\mathbf{x} \odot \mathbf{d})(\mathbf{x} \odot \mathbf{d})^T = (\mathbf{x}\mathbf{x}^T) \odot (\mathbf{d}\mathbf{d}^T)$$
$$\mathbf{E}[\mathbf{x} \odot \mathbf{d}] = \mathbf{x} \odot \mathbf{E}[\mathbf{d}]$$
$$\sum_{\mathbf{x}} (\mathbf{d} \odot \mathbf{x}) = \mathbf{d} \odot (\sum_{\mathbf{x}} \mathbf{x})$$

d) Now, while training we'll use the SSE loss i.e.

$$\mathcal{L}^{(i)}(\mathbf{w}, b) = (\mathbf{M}(\mathbf{x}^{(i)}) - y^{(i)})^2$$
$$\mathcal{L}(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)}(\mathbf{w}, b)$$

Notice that $\mathcal{L}(\mathbf{w}, b)$ is a random variable. Show that

$$\mathbf{E}[\mathcal{L}(\mathbf{w}, b)] = \frac{1}{m} \sum_{i=1}^{m} (p\mathbf{w}^T \mathbf{x}^{(i)} + b - y^{(i)})^2 + p(1-p) \sum_{j=1}^{m} \alpha_j \mathbf{w}_j^2$$
$$\text{where } \alpha_j = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{x}_j^{(i)})^2$$

e) Express $\mathcal{L}(\mathbf{w}, b)$ as $\tilde{\mathcal{L}}(\tilde{\mathbf{w}}, b)$ where $\tilde{\mathbf{w}} = p\mathbf{w}$ (because finally we'll use $\tilde{\mathbf{w}}$ for evaluation). Assume all n features have been shifted to zero data mean. Show that

$$\mathbf{E}[\tilde{\mathcal{L}}(\tilde{\mathbf{w}}, b)] = \frac{1}{m} \sum_{i=0}^{m} (\tilde{\mathbf{w}}^T \mathbf{x}^{(i)} + b - y^{(i)})^2 + \frac{1-p}{p} \sum_{j=1}^{m} \sigma_j^2 \tilde{\mathbf{w}}_j^2 \tag{12}$$

Where σ_j is the data standard deviation of the j^{th} feature.

f) Observe that $\tilde{\mathbf{w}} = p\mathbf{w} \implies \mathcal{L}(\mathbf{w}, b) = \tilde{\mathcal{L}}(\tilde{\mathbf{w}}, b)$, hence minimizing $\mathcal{L}(\mathbf{w}, b)$ over $(\mathbf{w}, b)$ is equivalent to minimizing $\tilde{\mathcal{L}}(\tilde{\mathbf{w}}, b)$ over $(\tilde{\mathbf{w}}, b)$. Clearly Equation 12 shows that using dropouts in Linear Regression has the effect of $\mathcal{L}_2$ regularization on $\tilde{\mathbf{w}}$. What can you say about the penalty corresponding to two different features relatively?

g) Think about the strength of regularization in the limits, $p \to 0^+$ and $p \to 1^-$. Was this expected?

5 Gaussian Discriminant Analysis and Naive Bayes

5.1 Distribution of Multivariate Gaussian

Let random vector $\mathbf{X} = \begin{bmatrix} X_1 & X_2 & \dots & X_n \end{bmatrix}^T$. Let $\mathbf{Z} \in \mathcal{R}^l$ be the standard normal random vector (i.e. $Z_i \sim \mathcal{N}(0,1)$ for $i = 1, \dots, \ell$ are i.i.d.). Then $X_1, \dots, X_n$ are *jointly Gaussian* if there exist $\boldsymbol{\mu} \in \mathcal{R}^n$, $\mathbf{A} \in \mathcal{R}^{n \times l}$ such that $\mathbf{X} = \mathbf{Az} + \boldsymbol{\mu}$.

We will see how this leads to the famous multivariate Gaussian distribution when the covariance matrix $\mathbf{\Sigma}$ is invertible. The joint moment generation function of random variables $X_1, X_2 ... X_n$ is defined as

$$M(t_1, t_2, \dots, t_n) = E[e^{t_1 X_1, \dots, t_n X_n}]$$

[Ross, 2014] in "Chapter 7 Properties of Expectation" has mentioned

It can be proven (although the proof is too advanced for this text) that the joint moment generating function $M(t_1, \dots, t_n)$ uniquely determines the joint distribution of $X_1, \dots, X_n$.

Denote $\mathbf{t} = \begin{bmatrix} t_1 & t_2 & \dots & t_n \end{bmatrix}^T$. Given the facts

$$Y \sim \mathcal{N}(\mu, \sigma^2) \implies \mathbb{E}[\exp(Y)] = \exp(\mu + \frac{\sigma^2}{2})$$

$$\int_{\mathbf{x} \in \mathcal{R}^n} f(\mathbf{x}) d^n \mathbf{x} = \int_{\boldsymbol{\alpha} \in \mathcal{R}^n} f(\mathbf{Q}\boldsymbol{\alpha} + \mathbf{b}) |\mathbf{Q}| d^n \boldsymbol{\alpha}$$

where $\mathbf{b} \in \mathcal{R}^n$ and $\mathbf{Q} \in \mathcal{R}^{n \times n}$ is invertible.

a) Prove that

$$\mathbb{E}[\exp(\mathbf{t}^T \mathbf{X})] = \exp(\mathbf{t}^T \boldsymbol{\mu} + \frac{\mathbf{t}^T \mathbf{A} \mathbf{A}^T \mathbf{t}}{2})$$

b) Prove that if PDF of $\mathbf{X}$ is

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{|\mathbf{\Sigma}|^{\frac{1}{2}} (2\pi)^{\frac{n}{2}}} \exp(\frac{-(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}{2})$$

then

$$\mathbb{E}[\exp(\mathbf{t}^T \mathbf{X})] = \exp(\mathbf{t}^T \boldsymbol{\mu} + \frac{\mathbf{t}^T \mathbf{\Sigma} \mathbf{t}}{2})$$

If you need a hint see this.[2]

5.2 GDA: Introduction

The GDA model learns the parameters $\phi \in (0,1)$, $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1 \in \mathcal{R}^n$, and $\mathbf{\Sigma}_0, \mathbf{\Sigma}_1 \in \mathcal{R}^{n \times n}$. $\mathbf{\Sigma}_0, \mathbf{\Sigma}_1$ should be symmetric positive definite. Here $\mathbf{x} \in \mathcal{R}^n$.

$$y \sim \text{Bernoulli}(\phi)$$
$$\mathbf{x} | y = 0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{\Sigma}_0)$$
$$\mathbf{x} | y = 1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \mathbf{\Sigma}_1)$$

[2] $\mathbf{\Sigma}$ is symmetric positive definite, so it has a unique symmetric positive definite square root, Write the integral and substitute $\mathbf{x} = \mathbf{\Sigma}^{\frac{1}{2}} \boldsymbol{\alpha} + (\boldsymbol{\mu} + \mathbf{\Sigma t})$.

5.3 Decision Boundary in GDA

Show that the decision boundary of GDA has equation

$$\frac{1}{2}\mathbf{x}^T(\mathbf{\Sigma}_1^{-1} - \mathbf{\Sigma}_0^{-1})\mathbf{x} - (\boldsymbol{\mu}_1^T\mathbf{\Sigma}_1^{-1} - \boldsymbol{\mu}_0^T\mathbf{\Sigma}_0^{-1})\mathbf{x} + \frac{1}{2}(\boldsymbol{\mu}_1^T\mathbf{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^T\mathbf{\Sigma}_0^{-1}\boldsymbol{\mu}_0) + ln(\frac{1-\phi}{\phi}\sqrt{\frac{|\mathbf{\Sigma}_1|}{|\mathbf{\Sigma}_0|}}) = 0$$

Give an example of values of $\phi, \boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \mathbf{\Sigma}_0, \mathbf{\Sigma}_1$ so that the decision boundary has equation

$$\mathbf{w}^T\mathbf{x} + b = 0$$

5.4 Logistic Regression is robust!

[Ng and Ma, 2023] mentioned in **Generative learning algorithms**

"There are many different sets of assumptions that would lead to $p(y|x)$ taking the form of a logistic function."

a) Consider the modeling assumptions of GDA: Introduction. Find the expression for $p(y|\mathbf{X} = \mathbf{x})$. Constrain $\mathbf{\Sigma}_0 = \mathbf{\Sigma}_1 = \mathbf{\Sigma}$. Show that it takes the logistic form

$$p(y = 1|\mathbf{x}; \phi, \boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \mathbf{\Sigma}) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}}$$

b) Let $\lambda_0, \lambda_1 > 0$, $0 < \phi < 1$ and

$$\begin{aligned} y &\sim \text{Bernoulli}(\phi) \\ X|y = 0 &\sim \text{Poisson}(\lambda_0) \\ X|y = 1 &\sim \text{Poisson}(\lambda_1) \end{aligned}$$

Show that $p(y|X = x)$ takes the logistic form

$$p(y = 1|X = x; \phi, \lambda_0, \lambda_1) = \frac{1}{1 + e^{-(wx+b)}}$$

c) Despite the result in (a), training Logistic Regression and GDA on the same data, don't learn the same boundary. Why? **Hint:** Write down the respective loss functions.

5.5 Probabilistic Interpretation of Laplace Smoothing

See **4.2.2 Event models for text classification** in [Ng and Ma, 2023]. We have a training set $\{(\mathbf{x}^{(i)}, y^{(i)}); i = 1, \ldots, m\}$ where $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_{d_n}^{(i)})$. Here, d_i is the number of words in the training example i^{th}. The parameters $\phi, \phi_{k0}, \phi_{k1}, k \in \{1, 2, \ldots, |V|\}$ have meaning as discussed in class. Denote $\Phi_0 = (\phi_{10}, \ldots, \phi_{|V|0})$ and $\Phi_1 = (\phi_{11}, \ldots, \phi_{|V|1})$. We will treat them as random variables with the prior distributions below and obtain the MAP estimates.

$$\begin{aligned} \phi &\sim \mathcal{U}(0, 1) \\ p(\Phi_0) &\propto \left(\prod_{k=1}^{|V|}(\phi_{k0})^{a_0}\right) \text{ for } 0 < \phi_{k0} < 1, \quad \Sigma\phi_{k0} = 1 \\ p(\Phi_1) &\propto \left(\prod_{k=1}^{|V|}(\phi_{k1})^{a_1}\right) \text{ for } 0 < \phi_{k1} < 1, \quad \Sigma\phi_{k1} = 1 \end{aligned}$$

Where $a_0, a_1 > 0$ are fixed. Φ_0 and Φ_1 follow the distributions, **Dirichlet**$(a_0+1, \dots, a_0+1)$ and **Dirichlet**$(a_1+1, \dots, a_1+1)$, more popularly the **Beta** distribution, when $|V| = 2$. Assume that

- All training examples are **independent**.
- There is atleast one example of each class.
- Each word in vocabulary V is present in atleast one example of each class.
- ϕ, Φ_0 and Φ_1 are independent.

a) Show that

$$p(\phi, \Phi_0, \Phi_1 | \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m) = \frac{p(\phi)p(\Phi_0)p(\Phi_1)\prod_{i=1}^m p(\mathbf{x}^{(i)}, y^{(i)}|\phi, \Phi_0, \Phi_1)}{p(\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m)}$$

b) The naive Bayes assumption is that the words (random variables) in different positions $X_1, X_2, \dots, X_{d_i}$ give y are independent. Show that

$$p(\mathbf{x}^{(i)}, y^{(i)}|\phi, \Phi_0, \Phi_1) = \phi^{y^{(i)}}(1-\phi)^{(1-y^{(i)})}\prod_{j=1}^{d_i}\left(\prod_{k=1}^{|V|}\phi_{k0}^{1\{x_j^{(i)}=k \wedge y^{(i)}=0\}}\phi_{k1}^{1\{x_j^{(i)}=k \wedge y^{(i)}=1\}}\right)$$

c) Show that $\arg\max_{\phi, \Phi_0, \Phi_1} p(\phi, \Phi_0, \Phi_1|\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m)$ gives

$$\begin{aligned}
\phi_{k0} &= \frac{a_0 + \sum_{i=1}^m \sum_{j=1}^{d_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 0\}}{a_0|V| + \sum_{i=1}^m 1\{y^{(i)} = 0\}d_i} \\
\phi_{k1} &= \frac{a_1 + \sum_{i=1}^m \sum_{j=1}^{d_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = 1\}}{a_1|V| + \sum_{i=1}^m 1\{y^{(i)} = 1\}d_i} \\
\phi &= \frac{\sum_{i=1}^m 1\{y^{(i)} = 1\}}{m}
\end{aligned}$$

Keep in mind that the parameters are constrained, $\sum_{k=1}^{|V|}\phi_{k0} = \sum_{k=1}^{|V|}\phi_{k1} = 1$

6 Principal Component Analysis

6.1 Variance and covariance of projected features

Consider the data $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^m$. Let $\mathbf{u}, \mathbf{v} \in \mathcal{R}^n$ be such that $\|u\|_2 = \|v\|_2 = 1$. $\mathbf{\Sigma} \in \mathcal{R}^{n\times n}$ is the covariance if $\mathcal{D}$. We now look at features formed by projecting $\mathcal{D}$ along the directions $\mathbf{u}$ and $\mathbf{v}$.

a) Show that

$$\begin{aligned}\mathrm{Var}(\mathbf{u}^T\mathbf{x}^{(i)}) &= \mathbf{u}^T\mathbf{\Sigma}\mathbf{u}\\ \mathrm{Cov}(\mathbf{u}^T\mathbf{x}^{(i)}, \mathbf{v}^T\mathbf{x}^{(i)}) &= \mathbf{u}^T\mathbf{\Sigma}\mathbf{v}\end{aligned}$$

b) Let $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ be eigenvectors of $\mathbf{\Sigma}$, chosen orthonormal. Show that

$$\begin{aligned}\mathrm{Cov}(\mathbf{v}_i^T\mathbf{x}^{(i)}, \mathbf{v}_j^T\mathbf{x}^{(i)}) &= 0, \quad i \neq j\\ \sum_{j=1}^n \mathrm{Var}(\mathbf{v}_j^T\mathbf{x}^{(i)}) &= \sum_{j=1}^n \mathrm{Var}(\mathbf{e}_j^T\mathbf{x}^{(i)})\end{aligned}$$

By finding $\mathrm{Var}(\mathbf{v}_i^T\mathbf{x}^{(i)})$ explicitly. Here $\mathbf{e}_1, \mathbf{e}_2, \dots$ are respectively $\begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}^T$, $\begin{bmatrix}0 & 1 & \dots & 0\end{bmatrix}^T$ and so on.

6.2 Projection Loss and maximizing variance

Consider the data $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$. $\mathbf{x}^{(i)} \in \mathcal{R}^n, y^{(i)} \in \mathcal{R}$ Assume $\sum_{i=1}^m \mathbf{x}^{(i)} = \boldsymbol{\mu}$.
We have $k \leq n$ and the set of constraints,

$$\mathbf{u}_i \in \mathcal{R}^n \tag{13}$$
$$\mathbf{u}_i^T\mathbf{u}_i = 1 \quad i \in \{1, 2, \dots, k\} \tag{14}$$
$$\mathbf{u}_i^T\mathbf{u}_j = 0 \quad i, j \in \{1, 2, \dots, k\} \quad i \neq j \tag{15}$$

The projection $\hat{\mathbf{x}}^{(i)}$ of $\mathbf{x}^{(i)}$ is given as

$$\hat{\mathbf{x}}^{(i)} = \boldsymbol{\mu} + \sum_{j=1}^k \mathbf{u}_j\mathbf{u}_j^T(\mathbf{x}^{(i)} - \boldsymbol{\mu})$$

Denote $\overline{\hat{\mathbf{x}}} = \frac{1}{m}\sum_{i=1}^m \hat{\mathbf{x}}^{(i)}$. We have the two optimization problems (under the constraints given before)

1. Maximizing projected variance

$$\max_{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k} \frac{1}{m \cdot n} \sum_{i=1}^m \left\|\hat{\mathbf{x}}^{(i)} - \overline{\hat{\mathbf{x}}}\right\|_2^2 \tag{16}$$

2. Minimizing projection loss

$$\min_{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k} \sum_{i=1}^m \left\|\mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)}\right\|_2^2 \tag{17}$$

a) Assume all $\mathbf{x}^{(i)}$'s are exactly the same (i.e. the total variance in 0) but all $y^{(i)}$'s are distinct. Can you learn anything from this data? Do you realize why variance in the data is related to the information it gives?

b) Show that both the problems above are equivalent. That is any set $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k\}$ maximizing Equation 16 minimizes Equation 17 and vice-versa.

6.3 Is the solution unique?

Look at the optimization problem described in Projection Loss and maximizing variance for minimizing projection loss, Equation 17. $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$ is a set of orthonormal vectors. Denote $\mathbf{U} \in \mathcal{R}^{n\times k}$ as the matrix with columns $\mathbf{u}_i$'s in that order. Let $\mathbf{A} \in \mathcal{R}^{k\times k}$ be orthonormal, and $\mathbf{U}' = \mathbf{UA}$. Let $\{\mathbf{u}'_1, \mathbf{u}'_2, \ldots, \mathbf{u}'_k\}$ be the columns of $\mathbf{U}'$.

a) Show that $\{\mathbf{u}'_1, \mathbf{u}'_2, \ldots, \mathbf{u}'_k\}$ are orthonormal.

b) Show that both $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$ and $\{\mathbf{u}'_1, \mathbf{u}'_2, \ldots, \mathbf{u}'_k\}$ give the same objective value in Equation 17.

Since you already know, taking the eigenvectors of $\mathbf{\Sigma}$, the data covariance as $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$ minimizes the projection loss. Now can you comment if this is the only solution to the problem of minimizing projection loss? We asked you the case, $k = 1$ of this problem in Quiz 10!

With some more work, you can show that $span(\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}) = span(\{\mathbf{u}'_1, \mathbf{u}'_2, \ldots, \mathbf{u}'_k\})$. Using this, grasp the fact that while minimizing projection loss, what we're actually looking for is a subspace to project the data onto. The Principal Components are actually a more informative representation of this subspace, because for every $k \leq n$, the span of the top k eigenvectors is the optimal subspace.

7 Decision Trees

7.1 Fast Greedy Pruning

Consider data of the form $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ where $\mathbf{x}^{(i)}$ is an n-dimensional feature vector and $y^{(i)} \in \{0, 1\}$. Denote the validation set by V. We wish to implement the Greedy Pruning algorithm to prune a trained decision tree T using the validation set V.

In each iteration, we consider all non-leaf nodes in T, and remove the entire subtree of a node (hence making it the leaf) that leads to maximum increase in accuracy on V. The process ends when any remaining nodes in the tree lead to a decrease in accuracy on V.

Naively implementing this algorithm takes $\mathcal{O}(n_{iters} \cdot |T| \cdot |V| \cdot depth(T))$ time. Here, in each iteration we temporarily prune a node and run V through the tree, to get accuracy on $|V|$. Finally, we actually prune the node based on the rule described above.

Think of an algorithm that can do this in $\mathcal{O}(|V| \cdot depth(T) + n_{iters}(|T| + depth(T)))$ time. Realize why this is a very great speed up.

You can find a well-tested Python implementation of this algorithm here [Harshit0143, 2023]. Hints for the Algorithm

1. During the pruning process, the split rule at every node remains the same. The set of examples from V that reach any node of T during pruning remains exactly the same.
2. Store n_0, n_1, n_{true}, the number of examples from class 0, class 1 from $|V|$ that reach this node and number of examples from these that are correctly classified, respectively. This can be done in $\mathcal{O}(|V| \cdot depth(T))$ time.
3. For a non-leaf fixed node, the change in accuracy on V, when pruning it can be obtained in $\mathcal{O}(1)$ time.

4. If you have to prune a non-leaf node ver, then n_0, n_1 remains unchanged throughout the tree. For which nodes does n_{true} change? You can update this value for all such nodes in total $\mathcal{O}(depth(T))$ time.

8 Support Vector Machines

8.1 SVM Kernels: But why bother duals?

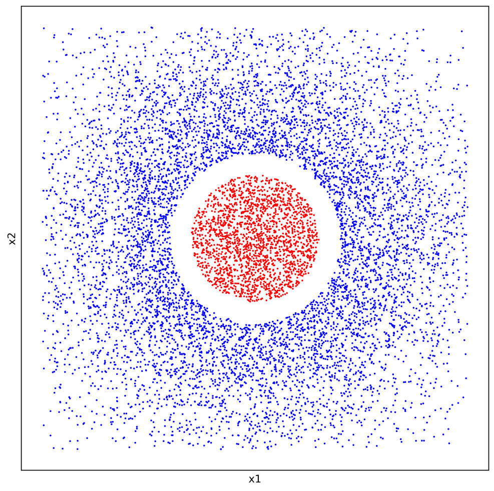

Figure 7: Classification data for SVM

$$\begin{aligned}&\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2\\ &\text{s.t. } y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b) \geq 1, \quad i \in \{1,2,\ldots,m\}\end{aligned}$$

Figure 8: SVM: Primal Problem

$$\begin{aligned}\max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle \\ \text{s.t.} \ & \alpha_i \geq 0, \quad i \in \{1, 2, \ldots, m\} \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0.\end{aligned}$$

Figure 9: SVM: Dual Problem

Look at the data in Figure 7. The feature vector is expressed as $\mathbf{x} = [x_1 \ \ x_2]^T$. The data is generated by sampling $\mathbf{x}$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and filtering out points, to $|x_1| \leq 5$ and $|x_2| \leq 5$. Points within the circle with center $(0, 0)$ and radius 3 are labeled red and those outside that of radius 4 are labeled blue. Other points are ignored.

a) Can simple SVM's (without using kernels) be used to classify the data perfectly in Figure 7.

b) Suggest a transformation $\phi : \mathcal{R}^2 \rightarrow \mathcal{R}^k$ such that when applied to the feature vectors in the data first, the data in Figure 7 can be perfectly classified using SVM.

c) When the number of features is 3, to be able to learn any quadratic decision boundary (using SVM in Figure 8), you can use the transform

$$\phi([x_1 \ \ x_2 \ \ x_3]^T) = [x_1^2 \ \ x_2^2 \ \ x_3^2 \ \ x_1x_2 \ \ x_2x_3 \ \ x_3x_1 \ \ x_1 \ \ x_2 \ \ x_3]^T$$

Think of a ϕ that can be used when $\mathbf{x} \in \mathcal{R}^n$ and you've to be able to learn all boundaries of degree $\leq d$. What is the size of the output vector?[3]

d) Continuing the previous part, you have a Black Box that can solve Figure 8. What is the size of this problem, i.e. the number of coefficients in the objective and constraints. How much time does it take to compute all of them?

e) In the dual problem, notice that $\mathbf{x}$ appears only in the inner product. You have been given a ϕ that maps $\mathbf{x} \in \mathcal{R}^n$ to space with features, all products of $x_1, x_2, \ldots, x_n$ of degree $\leq d$.

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle = (\mathbf{x}^T \mathbf{z} + c)^d \tag{18}$$

c is a hyperparameter that decides weight for terms of lower degrees. You have a Black Box that can solve Figure 9. Do you realize that, given Equation 18, you do not need to worry about the exact form of ϕ? What is the size of the problem now? How much time does it take to compute all the coefficients, without (a rough lower bound) and with the kernel trick in Equation 18?

8.2 Soft Margin SVM

We have data $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ where $\mathbf{x}^{(i)} \in \mathcal{R}^n$ and $y^{(i)} \in \{-1, 1\}$. Consider the soft SVM optimization problem

$$\min_{\mathbf{w}, b, \boldsymbol{\epsilon}} \quad \mathcal{L}_C(\mathbf{w}, b, \boldsymbol{\epsilon}) = \frac{1}{1+C} \frac{\mathbf{w}^T \mathbf{w}}{2} + \frac{C}{1+C} \sum_{i=1}^{m} \epsilon_i$$

[3]Hint: The number of terms in the expansion of $(x_1 + x_2 + \cdots + x_n + c)^d$, c is a constant.

Under constraints

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \epsilon_i \quad i \in \{1, 2, \dots, m\}$$
$$\epsilon_i \geq 0 \quad i \in \{1, 2, \dots, m\}$$

Denote $\boldsymbol{\epsilon} = [\epsilon_1 \quad \epsilon_2 \quad \dots \quad \epsilon_m]^T$. $C \geq 0$ is a hyperparameter. In each of the below cases, give the optimum value of the objectives and construct the exact values of $\mathbf{w}, b, \boldsymbol{\epsilon}$ that lead to that objective.

a) $\mathcal{L}_0(\mathbf{w}, b, \boldsymbol{\epsilon})$

b) $\mathcal{L}_\infty(\mathbf{w}, b, \boldsymbol{\epsilon})$ when $\mathcal{D}$ is linearly separable.

c) Comment on the value of objective $\mathcal{L}_\infty(\mathbf{w}, b, \boldsymbol{\epsilon})$ when $\mathcal{D}$ is **not** linearly separable.

9 Clustering Algorithms

9.1 GDA classification and GMM clustering

We have $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ where $\mathbf{x}^{(i)} \in \mathcal{R}^n$ and $y^{(i)} \in \{1, 2, \dots, k\}$. $\mathcal{D}$ was generated by first selecting $y^{(i)}$ in $\{1, 2, \dots, k\}$ using Multinoulli($\boldsymbol{\phi}$), $\phi \in \mathcal{R}^k$, and then choosing $\mathbf{x}^{(i)}$ from $\mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j), \boldsymbol{\mu}_j \in \mathcal{R}^n, \boldsymbol{\Sigma} \in \mathcal{R}^{n\times n}, j \in \{1, 2, \dots, k\}$, where j is the label picked earlier. This process is independent for each example.

When the labels $y^{(i)}$ are hidden (not available to us), they are denoted as $z^{(i)}$ and we can run the GMM algorithm on the data to figure out which examples belong to which "cluster" (and not class as that is unknown).

a) Set up the MLE problem, and give MLE estimates for parameters $\boldsymbol{\phi}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$. Do this by maximizing $p(\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m)$. Recall our discussion from Gaussian Discriminant Analysis and Naive Bayes.

b) Now, let us say in the data $\mathcal{D}$, for very few examples, the labels $y^{(i)}$ are missing. How will you label these examples for another downstream task that needs to be labeled $\mathcal{D}$.

9.2 Modified GMM Clustering

We have data $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^m, \mathbf{x}^{(i)} \in \mathcal{R}^n$. Consider clustering using GMM. We modify it slightly here. We now model,

$$z^{(i)} \sim \text{Multinoulli}(\boldsymbol{\phi})$$
$$\mathbf{x}^{(i)} \mid (z^{(i)} = j) \sim \mathcal{N}(\boldsymbol{\mu}_j, \sigma^2\mathbf{I}_n) \quad j \in \{1, 2, \dots, k\}$$

Here $\boldsymbol{\phi} \in \mathcal{R}^k, \boldsymbol{\mu}_j \in \mathcal{R}^n$ are learnable but $\sigma^2 > 0$ is a **hyperparameter**.

a) Write down **E. Step** (update of w_{ij}) and **M. Step** (update of learnable parameters) for this problem.

b) Using w_{ij} computed in the above problem, show that in the limit $\sigma \to 0^+$, the algorithm becomes the K-means clustering algorithm.

References

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning.* MIT Press. http://www.deeplearningbook.org.

[Harshit0143, 2023] Harshit0143 (2023). COL774 Fall 2023-2024. https://github.com/Harshit0143/Machine-Learning-Assignments.

[Ng and Ma, 2023] Ng, A. and Ma, T. (June 11, 2023). CS229 Lecture Notes. https://cs229.stanford.edu/main_notes.pdf.

[Petersen and Pedersen, 2021] Petersen, K. B. and Pedersen, M. S. (November 2021). The matrix cookbook. https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf.

[Ross, 2014] Ross, S. M. (2014). *A First Course in Probability.* Pearson, Boston, 9th edition.