VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Natural Language Processing - Exercise (CO3086)

Lab 6

Math Exercises

Semester 2, Academic Year 2024 - 2025

Teacher:	Bui Khanh Vinh
Students:	Nguyen Quang Phu - 2252621

HO CHI MINH CITY, FEBRUARY 2025

Contents

1 Problem Description and Solution **3**

1.1 Problem 1 3

1.1.1 Description 3

1.1.2 Answer to Question (a) 4

1.1.3 Answer to Question (b) 5

1.1.4 Answer to Question (c) 5

1.2 Problem 2 6

1.2.1 Description 6

1.2.2 Solution to Question (a) 7

1.2.3 Solution to Question (b) 8

1.3 Problem 3 9

1.3.1 Description 9

1.3.2 Solution 9

1.4 Problem 4 10

1.4.1 Questions 10

1.4.2 Solution 11

1.5 Problem 5 13

1.5.1 Description 13

1.5.2 Solution to Question (a) 13

1.5.3 Solution to Question (b) 15

1.6 Problem 6 16

1.6.1 Description 16

1.6.2 Solution 17

1.7 Problem 7 18
1.7.1 Description 18
1.7.2 Solution 19

Chapter 1

Problem Description and Solution

1.1 Problem 1

1.1.1 Description

We are dealing with samples x where x is a single value. We would like to test two alternative regression models:

1. $y = ax + e$
2. $y = ax + bx^2 + e$

We make the same assumptions we had in class about the distribution of e ($e \sim N(0, s^2)$).

(a) Assume we have n samples: $x_1, \ldots, x_n$ with their corresponding y values: $y_1, \ldots, y_n$. Derive the value assigned to b in model 2. You can use a in the equation for b.

(b) Which of the two models is more likely to fit the *training* data better? Explain your answer.

   (a) model 1
   (b) model 2
   (c) both will fit equally well
   (d) impossible to tell

(c) Which of the two models is more likely to fit the *test* data better? Explain your answer.

   (a) model 1
   (b) model 2
   (c) both will fit equally well
   (d) impossible to tell

1.1.2 Answer to Question (a)

Why Use Residual Sum of Squares (RSS)?

To estimate the parameters a and b, we use the **least squares method**, which minimizes the **Residual Sum of Squares (RSS)**:

$$RSS = \sum_{i=1}^{n} (y_i - ax_i - bx_i^2)^2$$

Minimizing RSS ensures that the chosen parameters a and b provide the best fit to the data by reducing the squared error.

Derivation of b

To find b, we differentiate the RSS function with respect to b and set it to zero.

Expanding the RSS function:

$$RSS = \sum_{i} (y_i - ax_i - bx_i^2)^2$$

Taking the partial derivative with respect to b:

$$\frac{\partial RSS}{\partial b} = -2 \sum_{i} x_i^2 (y_i - ax_i - bx_i^2)$$

Setting this equation to zero:

$$\sum_{i} x_i^2 y_i - a \sum_{i} x_i^3 - b \sum_{i} x_i^4 = 0$$

Solving for b:

$$b = \frac{\sum_i y_i x_i^2 - a \sum_i x_i^3}{\sum_i x_i^4}$$

Conclusion

The value of b is obtained using least squares regression by minimizing the Residual Sum of Squares (RSS). This approach provides a closed-form solution that efficiently estimates the parameter.

1.1.3 Answer to Question (b)

Which of the two models is more likely to fit the training data better?

Answer: Model 2.

Explanation:

- Model 2 has an additional parameter b, making it more flexible than model 1.
- More parameters allow model 2 to capture complex patterns in the training data, potentially leading to a lower training error.
- In general, models with more parameters fit training data better because they minimize the residual sum of squares (RSS) more effectively.

1.1.4 Answer to Question (c)

Which of the two models is more likely to fit the test data better?

Answer: Impossible to tell.

Explanation:

- If the true relationship between x and y is linear, then model 1 is preferable. Model 2 would introduce unnecessary complexity, leading to **overfitting**.
- If the true relationship is quadratic, then model 2 is preferable, as model 1 would result in **underfitting**.
- If the training data is limited, model 2 might learn noise rather than the true relationship, making it generalize poorly to new data.
- If the training data is large and supports a quadratic relationship, model 2 will generalize better and outperform model 1.

Thus, the best choice depends on the **underlying model of the data** and the **amount of training data available**.

1.2 Problem 2

1.2.1 Description

(a) Now assume we only observe a single input for each output (that is, a set of $\{x, y\}$ pairs). We would like to compare the following two models on our input dataset (for each one we split into training and testing sets to evaluate the learned model). Assume we have an unlimited amount of data:

$$\begin{aligned} \mathbf{A}: &\quad y = w^2 x \\ \mathbf{B}: &\quad y = wx \end{aligned}$$

Which of the following is correct and Explain:

(a) There are datasets for which A would perform **better** than B.

(b) There are datasets for which B would perform **better** than A.

(c) Both 1 and 2 are correct.

(d) They would perform equally well on all datasets.

(b) For the data above, we are now comparing the following two models:

$$\begin{aligned} \mathbf{A}: &\quad y = w_1^2 x + w_2 x \\ \mathbf{B}: &\quad y = wx \end{aligned}$$

Note that model A now uses two parameters (though both multiply the same input value, x). Again, we assume unlimited data. Which of the following is correct (choose the answer that best describes the outcome) and Explain:

(a) There are datasets for which A would perform **better** than B.

(b) There are datasets for which B would perform **better** than A.

(c) Both 1 and 2 are correct.

(d) They would perform equally well on all datasets.

1.2.2 Solution to Question (a)

We are given the two models:

$$\textbf{Model A: } y = w^2 x$$

$$\textbf{Model B: } y = wx$$

Since we have **unlimited data**, the primary concern is how well these models fit the underlying true relationship between x and y.

Key Observations:

- Both models are **linear in** x (since w^2x is still proportional to x).
- Model A, however, constrains w to be squared, meaning w can only be **non-negative**.
- Model B allows w to be either **positive or negative**.

Implications:

- If the **true relationship** between x and y follows $y = kx$ where k can be negative, **Model B is better** because it can learn both positive and negative slopes.
- If the true relationship is $y = kx$ but k is always **non-negative**, then Model A can also learn it.
- However, since Model A **forces** w **to be non-negative**, it **cannot model negative slopes** properly, making it a less flexible model overall.

Conclusion for (a):

- **Model B is strictly more flexible than Model A** because it can represent both positive and negative relationships.
- **There exist datasets where Model B would perform better than Model A** (when $k < 0$).
- **There are no datasets where Model A would be strictly better than Model B** because any function learnable by Model A can also be learned by Model B.

Thus, the correct choice is **(b): "There are datasets for which B would perform better than A."**

1.2.3 Solution to Question (b)

We are given the new models:

$$\textbf{Model A: } y = w_1^2 x + w_2 x$$
$$\textbf{Model B: } y = wx$$

Key Observations:

- Model A introduces an additional term ($w_1^2 x$ and $w_2 x$), making it a **more flexible model**.
- Model B is just a single linear coefficient applied to x.

Implications:

- **If the true function is exactly linear** ($y = kx$), then Model B is sufficient, and the extra complexity in Model A is unnecessary.
- **If the true function follows a form where an additional term improves the fit** (e.g., something slightly quadratic in nature), then Model A would perform better.
- However, since we have **unlimited data**, Model B will always converge to the true function if it is purely linear.
- Model A introduces unnecessary complexity, which does not provide additional benefits when the true function is linear.

Conclusion for (b):

- Since we have unlimited data, Model B will always fit a linear function optimally.
- Model A adds unnecessary complexity and will not provide any advantages in this scenario.

Thus, the correct choice is **(d): "They would perform equally well on all datasets."**

Final Choices:

- For (a), I choose **(b)**.
- For (b), I choose **(d)**.

1.3 Problem 3

1.3.1 Description

We are given a set of two-dimensional inputs and their corresponding output pair: $\{x_{i,1}, x_{i,2}, y_i\}$. We would like to use the following regression model to predict y:

$$y_i = w_1^2 x_{i,1} + w_2^2 x_{i,2}$$

Derive the optimal value for w_1 when using least squares as the target minimization function (w_2 may appear in your resulting equation). Note that there may be more than one possible value for w_1.

1.3.2 Solution

Given the regression model: $y_i = w_1^2 x_{i,1} + w_2^2 x_{i,2}$. We aim to minimize the least squares loss function:

$$J(w_1, w_2) = \sum_i \left(y_i - \left(w_1^2 x_{i,1} + w_2^2 x_{i,2}\right)\right)^2$$

Taking the derivative with respect to w_1:

$$\frac{\partial J}{\partial w_1} = \sum_i 2\left(y_i - w_1^2 x_{i,1} - w_2^2 x_{i,2}\right)(-2w_1 x_{i,1})$$

$$\frac{\partial J}{\partial w_1} = \sum_i -4w_1 x_{i,1}\left(y_i - w_1^2 x_{i,1} - w_2^2 x_{i,2}\right)$$

Setting the derivative to zero for minimization:

$$\sum_i -4w_1 x_{i,1}\left(y_i - w_1^2 x_{i,1} - w_2^2 x_{i,2}\right) = 0$$

Rearranging and Solving for w_1^2 then we get:

$$w_1^2 = \frac{\sum_i x_{i,1} y_i - w_2^2 \sum_i x_{i,1} x_{i,2}}{\sum_i x_{i,1}^2}$$

Taking the square root and there are two possible values for w_1.

$$w_1 = \pm\sqrt{\frac{\sum_i x_{i,1} y_i - w_2^2 \sum_i x_{i,1} x_{i,2}}{\sum_i x_{i,1}^2}}$$

1.4 Problem 4

You are asked to use regularized linear regression to predict the target $Y \in \mathbb{R}$ from the eight-dimensional feature vector $X \in \mathbb{R}^8$. You define the model $Y = w^T X$ and then you recall from class the following three objective functions:

$$\min_w \sum_{i=1}^{n} \left(y_i - w^T x_i\right)^2 \tag{4.1}$$

$$\min_w \sum_{i=1}^{n} \left(y_i - w^T x_i\right)^2 + \lambda \sum_{j=1}^{8} w_j^2 \tag{4.2}$$

$$\min_w \sum_{i=1}^{n} \left(y_i - w^T x_i\right)^2 + \lambda \sum_{j=1}^{8} |w_j| \tag{4.3}$$

1.4.1 Questions

(a) Show regularization terms in the objective functions above.

(b) For large values of λ in objective (4.2), the bias would: Increase, Decrease, or Remain unaffected

(c) For large values of λ in objective (4.3), the variance would: Increase, Decrease, or Remain unaffected

(d) The following table contains the weights learned for all three objective functions (not in any particular order):

	Column A	**Column B**	**Column C**
w_1	0.60	0.38	0.50
w_2	0.30	0.23	0.20
w_3	-0.10	-0.02	0.00
w_4	0.20	0.15	0.09
w_5	0.30	0.21	0.00
w_6	0.20	0.03	0.00
w_7	0.02	0.04	0.00
w_8	0.26	0.12	0.05

Table 1.1: Weight values for different objective functions

1.4.2 Solution

(a) Show regularization terms in the objective functions.

The regularization terms are:

- For Ridge Regression (Equation 4.2), the regularization term is:

  $$\lambda \sum_{j=1}^{8} w_j^2$$

  This penalizes large weight values, preventing overfitting but retaining all features.

- For Lasso Regression (Equation 4.3), the regularization term is:

  $$\lambda \sum_{j=1}^{8} |w_j|$$

  This encourages sparsity by driving some coefficients to zero, effectively selecting a subset of features.

(b) Effect of Large λ in Objective (4.2) on Bias

- As λ increases in Ridge Regression, the penalty on large weight values increases.
- This results in a model with **smaller** weights, reducing its flexibility.
- A less flexible model underfits the data, leading to an **increase in bias**.

Answer: The bias would increase.

(c) Effect of Large λ in Objective (4.3) on Variance

- In Lasso Regression, a large λ forces many weights to exactly zero, reducing the number of active features.
- This makes model less complex, reducing its sensitivity to variations in the training data.
- Lower model complexity leads to **lower variance**.

Answer: The variance would decrease.

(d) Analysis of Learned Weights for Different Objective Functions

Objective Functions and Regularization

We analyze the following objective functions:

- **Objective 5.1**: Ordinary Least Squares (OLS) regression. This does not apply any regularization, so the learned weights are typically the largest.
- **Objective 5.2**: Ridge Regression (L2 regularization). This penalizes large weights, making them smaller than OLS but still nonzero.
- **Objective 5.3**: Lasso Regression (L1 regularization). This promotes sparsity, leading some weights to be exactly zero.

Determining the Corresponding Objective Functions

By analyzing the weights:

- **Column A** contains the largest absolute weight values, indicating **OLS (Objective 5.1)**.
- **Column B** contains smaller weights but none are exactly zero, suggesting **Ridge Regression (Objective 5.2)**.
- **Column C** has multiple weights exactly equal to zero, which is characteristic of **Lasso Regression (Objective 5.3)**.

Final Answers

- **Objective 5.1: Solution: A**
- **Objective 5.2: Solution: B**
- **Objective 5.3: Solution: C**

1.5 Problem 5

1.5.1 Description

Suppose you are given the following classification task: predict the target $Y \in \{0,1\}$ given two real-valued features $X_1 \in \mathbb{R}$ and $X_2 \in \mathbb{R}$. After some training, you learn the following decision rule:

$$\textbf{Predict } Y = 1 \textbf{ iff } w_1X_1 + w_2X_2 + w_0 \geq 0 \textbf{ and } Y = 0 \textbf{ otherwise}$$

where $w_1 = 3$, $w_2 = 5$, and $w_0 = -15$.

(a) Plot the decision boundary and label the region where we would predict $Y = 1$ and $Y = 0$.

(b) Suppose that we learned the above weights using logistic regression. Using this model, what would be our prediction for $P(Y = 1 \mid X_1, X_2)$? (You may want to use the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$.)

$$P(Y = 1 \mid X_1, X_2) =$$

1.5.2 Solution to Question (a)

Problem Setup

We are given a classification task where we predict the target $Y \in \{0,1\}$ based on two real-valued features X_1 and X_2. The decision rule is:

$$Y = 1 \quad \text{iff} \quad w_1X_1 + w_2X_2 + w_0 \geq 0$$

where:

$$w_1 = 3, \quad w_2 = 5, \quad w_0 = -15.$$

Finding the Decision Boundary

To find the decision boundary, we set:

$$3X_1 + 5X_2 - 15 = 0$$

Solving for X_2:

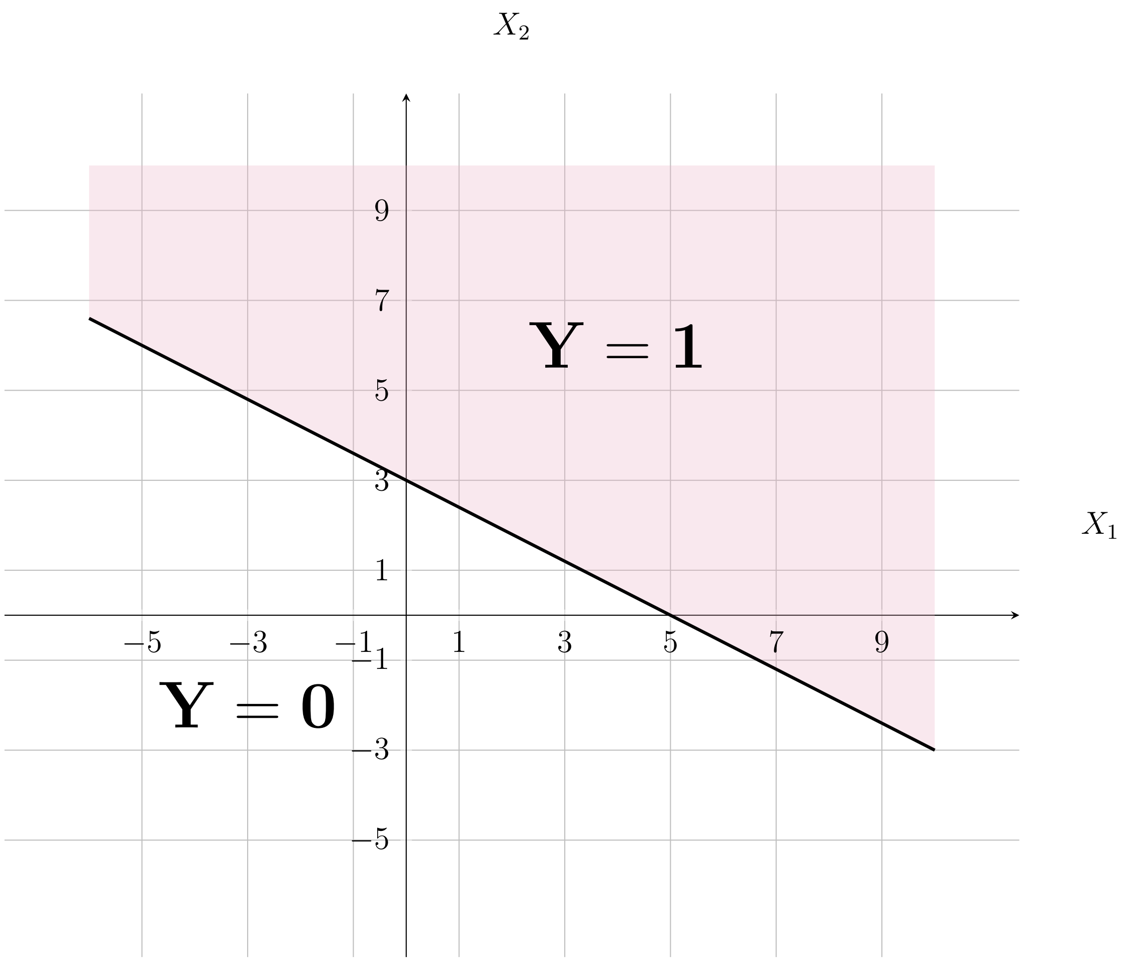
$$X_2 = -\frac{3}{5}X_1 + 3$$

This equation represents a straight line in the X_1, X_2 plane.

Identifying the Regions

- **Above the line** $X_2 = -\frac{3}{5}X_1 + 3$, the model predicts $Y = 1$.
- **Below the line**, the model predicts $Y = 0$.

Graphical Representation



The figure above shows:

- The black **decision boundary** line given by $X_2 = -\frac{3}{5}X_1 + 3$.
- The **shaded upper region** where $Y = 1$.
- The **unshaded lower region** where $Y = 0$.

1.5.3 Solution to Question (b)

The probability can be written as:

$$P(Y = 1 \mid X_1, X_2) = \sigma(3X_1 + 5X_2 - 15)$$

Expanding the sigmoid function:

$$P(Y = 1 \mid X_1, X_2) = \frac{1}{1 + \exp(-(3X_1 + 5X_2 - 15))}$$

Interpretation

- The term $3X_1 + 5X_2 - 15$ is the **logit function**.
- The sigmoid function ensures that the output is always between 0 and 1.
- If $3X_1 + 5X_2 - 15$ is large and positive, $P(Y = 1) \approx 1$.
- If $3X_1 + 5X_2 - 15$ is large and negative, $P(Y = 1) \approx 0$.
- When $3X_1 + 5X_2 - 15 = 0$, the probability is 0.5, which is the decision boundary.

1.6 Problem 6

1.6.1 Description

Consider a simple one-dimensional logistic regression model:

$$P(y = 1 \mid x, \mathbf{w}) = g(w_0 + w_1 x)$$

where $g(z) = \frac{1}{1+\exp(-z)}$ is the logistic function. The following figure shows two possible conditional distributions $P(y = 1 \mid x; \mathbf{w})$, viewed as a function of x, that we can get by changing the parameters $\mathbf{w}$.

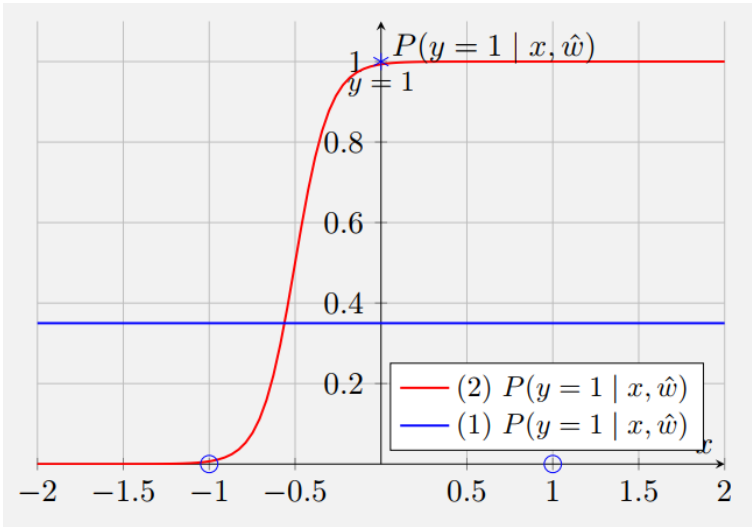


Figure 1.1: Two possible logistic regression classifiers

(a) Please indicate the number of classification errors for each conditional given the labeled examples in the same figure.

(b) One of the two classifiers corresponds to the maximum likelihood setting of the parameters $\mathbf{w}$ based on the labeled data in the figure, i.e., its parameters maximize the joint probability:

$$P(y = 0 \mid x = -1; \mathbf{w}) \quad P(y = 1 \mid x = 0; \mathbf{w}) \quad P(y = 0 \mid x = 1; \mathbf{w})$$

Circle which one is the ML solution and briefly explain: **Classifier 1 or Classifier 2**.

(c) Would adding a regularization penalty $|w_1|^2/2$ to the log-likelihood estimation criterion affect your choice of solution (Y/N)? (Note that the penalty above only regularizes w_1, not w_0.) Briefly explain why.

1.6.2 Solution

(a) Classification Errors

- **Conditional (1) makes:** 0 classification errors. Because you can see that from the image, the classifier 1 (in blue) has separate the space into 2 halves: top and bottom. For the top part and contains only the x point and for the bottom part it contains o points. So it is correctly classified the samples.
- **Conditional (2) makes:** 1 classification error. The classifier one in red has missed classified the o point at $x = 1$ because based on the image, when $x = 1$, the red classifier must return 1 instead of 0 like the image.

(b) Maximum Likelihood Estimation (MLE)

The maximum likelihood estimate (MLE) maximizes the joint probability:

$$P(y = 0|x = -1; w), \quad P(y = 1|x = 0; w), \quad P(y = 0|x = 1; w)$$

Answer: Classifier 2 (Red Curve).

Explanation:

- Classifier 1 (Blue Line) assigns a constant probability around 0.5, which does not adapt to the labeled data.
- Classifier 2 (Red Curve) adjusts the probability based on x, aligning more closely with the observed labels.
- Thus, the classifier that follows the MLE principle is **Classifier 2 (Red Curve)**.

(c) Effect of Regularization

The addition of an $L2$ regularization penalty: $\frac{|w_1|^2}{2}$ penalizes large values of w_1, making the decision boundary more gradual.

Would this affect the choice of solution? Yes (Y).

Explanation:

- Regularization reduces the magnitude of w_1, making the logistic function more gradual.
- If the penalty is too strong, the decision boundary flattens out, making Classifier 2 resemble Classifier 1.
- This means **Classifier 2 would be affected**, shifting towards a suboptimal solution.

1.7 Problem 7

1.7.1 Description

In many real-world scenarios, our **data has millions of dimensions, but a given example has only hundreds of non-zero features**. For example, in document analysis with word counts for features, our dictionary may have millions of words, but a given document has only hundreds of unique words. In this question, we will make l_2 **regularized SGD efficient** when our input data is sparse. Recall that in l_2 regularized logistic regression, we want to maximize the following objective **(in this problem we have excluded w_0 for simplicity)**:

$$F(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} l(x^{(j)}, y^{(j)}, \mathbf{w}) - \frac{\lambda}{2}\sum_{i=1}^{d} w_i^2$$

where $l(x^{(j)}, y^{(j)}, \mathbf{w})$ is the logistic objective function:

$$l(x^{(j)}, y^{(j)}, \mathbf{w}) = y^{(j)}\left(\sum_{i=1}^{d} w_i x_i^{(j)}\right) - \ln\left(1 + \exp\left(\sum_{i=1}^{d} w_i x_i^{(j)}\right)\right)$$

and the remaining sum is our regularization penalty. When we do stochastic gradient descent (SGD) on point $(x^{(j)}, y^{(j)})$, we approximate the objective function as:

$$F(\mathbf{w}) \approx l(x^{(j)}, y^{(j)}, \mathbf{w}) - \frac{\lambda}{2}\sum_{i=1}^{d} w_i^2$$

Definition of Sparsity: Assume that our input data has d features, i.e., $x^{(j)} \in \mathbb{R}^d$. In this problem, we consider the scenario where $x^{(j)}$ is sparse. Formally, let s be the average number of nonzero elements in each example. We say that the data is sparse when $s \ll d$.

In the following questions, **your answer should take the sparsity of $x^{(j)}$ into consideration when possible.**

Note: When we use a sparse data structure, we can iterate over the non-zero elements in $O(s)$ time, whereas a dense data structure requires $O(d)$ time.

(a) Let us first consider the case when $\lambda = 0$. Write down the SGD update rule for $\mathbf{w}$, where $\lambda = 0$, using step size η, when the example $(x^{(j)}, y^{(j)})$ is given.

(b) If we use a dense data structure, what is the average time complexity to update w_i when $\lambda = 0$? What if we use a sparse data structure? Justify your answer in one or two sentences.

(c) Now let us consider the general case when $\lambda > 0$. Write down the SGD update rule for w_i when $\lambda > 0$, using step size η, given the example $(x^{(j)}, y^{(j)})$.

(d) If we use a dense data structure, what is the average time complexity to update w_i when $\lambda > 0$?

(e) Let $w_i^{(t)}$ be the weight vector after the t-th update. Now imagine that we perform k SGD updates on $\mathbf{w}$ using examples $(x^{(t+1)}, y^{(t+1)}), \ldots, (x^{(t+k)}, y^{(t+k)})$, where $x_i^{(j)} = 0$ for every example in the sequence. (i.e., the i-th feature is zero for all examples in the sequence). Express the new weight, $w_i^{(t+k)}$, in terms of $w_i^{(t)}$, k, η, and λ.

(f) Using your answer in the previous part, come up with an efficient algorithm for regularized SGD when we use a sparse data structure. What is the average time complexity per example? *(Hint: when do you need to update w_i?)*

1.7.2 Solution

(a) SGD Update Rule for $\boldsymbol{\lambda = 0}$

Derivation of the SGD Update Rule

Given the L_2-regularized logistic regression objective function:

$$F(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w}) - \frac{\lambda}{2}\sum_{i=1}^{d} w_i^2$$

where the logistic loss function is defined as:

$$l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w}) = y^{(j)}\sum_{i=1}^{d} w_i x_i^{(j)} - \ln\left(1 + \exp\left(\sum_{i=1}^{d} w_i x_i^{(j)}\right)\right).$$

From the problem statement, we see that we are dealing with **logistic regression with L_2 regularization**, and our objective function $F(\mathbf{w})$ is:

$$F(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w}) - \frac{\lambda}{2}\sum_{i=1}^{d} w_i^2$$

where $l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w})$ is the logistic loss function.

Maximization vs. Minimization

- In many machine learning problems, we either **maximize** an objective function (such as log-likelihood in probabilistic models) or **minimize** a loss function (such as logistic loss or squared loss).
- Here, the problem explicitly states that we want to **maximize** $F(\mathbf{w})$, not minimize a loss.

- Since the gradient points in the direction of **steepest ascent**, the update rule follows the **gradient ascent** rule:

$$w_i^{(t+1)} = w_i^{(t)} + \eta \frac{\partial F(\mathbf{w})}{\partial w_i}$$

where:

- η is the learning rate,
- $\frac{\partial F(\mathbf{w})}{\partial w_i}$ is the gradient.

Gradient Computation

The gradient of the objective function with respect to w_i is:

$$\frac{\partial F(\mathbf{w})}{\partial w_i} = \frac{\partial l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w})}{\partial w_i} - \lambda w_i.$$

Computing the derivative of the logistic loss:

$$\frac{\partial l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w})}{\partial w_i} = x_i^{(j)} \left(y^{(j)} - \frac{1}{1 + \exp(-\sum_k w_k x_k^{(j)})} \right).$$

SGD Update Rule for $\lambda = 0$

Since our goal is to **maximize** $F(\mathbf{w})$, we apply **gradient ascent**, when $\lambda = 0$, the SGD update rule simplifies to:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta x_i^{(j)} \left(y^{(j)} - \frac{1}{1 + \exp(-\sum_k w_k x_k^{(j)})} \right).$$

(b) Time Complexity Analysis

The SGD update rule for logistic regression is:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta x_i^{(j)} \left(y^{(j)} - \frac{1}{1 + \exp(-\sum_k w_k x_k^{(j)})} \right).$$

The most computationally expensive step is calculating the term:

$$\sum_k w_k x_k^{(j)}$$

which is needed to compute the probability in logistic regression.

Time Complexity for Dense Data

- A **dense** data structure means most $x_k^{(j)}$ are nonzero.
- We sum over all d features, requiring $O(d)$ time.

Time Complexity for Sparse Data

- A **sparse** data structure means only s features are nonzero, where $s \ll d$.
- Instead of summing over all d, we only sum over the s nonzero values, reducing computation to $O(s)$.

(c) SGD Update Rule for $\boldsymbol{\lambda > 0}$

Understanding the SGD Update Rule for $\lambda > 0$

The L_2-regularized logistic regression objective function is:

$$F(\mathbf{w}) = \frac{1}{N} \sum_{j=1}^{N} l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w}) - \frac{\lambda}{2} \sum_{i=1}^{d} w_i^2$$

where the logistic loss function is:

$$l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w}) = y^{(j)} \sum_{i=1}^{d} w_i x_i^{(j)} - \ln\left(1 + \exp\left(\sum_{i=1}^{d} w_i x_i^{(j)}\right)\right).$$

Gradient Computation and Update Rule

The gradient of the objective function with respect to w_i is:

$$\frac{\partial F(\mathbf{w})}{\partial w_i} = x_i^{(j)} \left(y^{(j)} - \frac{1}{1 + \exp(-\sum_k w_k x_k^{(j)})} \right) - \lambda w_i.$$

Thus, the SGD update rule is:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta \lambda w_i^{(t)} + \eta x_i^{(j)} \left(y^{(j)} - \frac{1}{1 + \exp(-\sum_k w_k x_k^{(j)})} \right).$$

This update consists of:

- A regularization decay term $-\eta\lambda w_i^{(t)}$, which shrinks weights.
- The standard SGD update term for logistic regression.

(d) Time Complexity for Regularized SGD

The main computational cost is the summation:

$$\sum_k w_k x_k^{(j)}$$

which takes $O(d)$ time in a dense data structure. Since we update all w_i, the total time complexity is: $O(d)$

(e) Weight Update Expression for k Updates

We consider a sequence of k stochastic gradient descent (SGD) updates on a weight w_i in L2-regularized logistic regression. The general SGD update rule is:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta\lambda w_i^{(t)} + \eta x_i^{(j)} \left(y^{(j)} - \frac{1}{1+\exp(-\sum_k w_k x_k^{(j)})} \right).$$

Case When $x_i^{(j)} = 0$

When the feature $x_i^{(j)} = 0$ for all examples in the sequence, the logistic regression term vanishes, and the update simplifies to:

$$w_i^{(t+1)} = w_i^{(t)} - \eta\lambda w_i^{(t)}.$$

This can be rewritten as:

$$w_i^{(t+1)} = w_i^{(t)}(1-\eta\lambda).$$

Generalizing Over k Updates

Applying this update iteratively over k updates:

$$w_i^{(t+2)} = w_i^{(t+1)}(1-\eta\lambda) = w_i^{(t)}(1-\eta\lambda)^2.$$

By induction, after k updates:

$$w_i^{(t+k)} = w_i^{(t)}(1-\eta\lambda)^k.$$

(f) Efficient Regularized SGD Algorithm

Motivation

- In a dense dataset, each feature vector $x^{(j)}$ has d nonzero values, so updating all w_i takes $O(d)$ time.
- In a sparse dataset, only a small subset s of the features are nonzero ($s \ll d$).
- The goal is to efficiently update weights without unnecessary computations.

Algorithm Explanation

The algorithm efficiently applies **stochastic gradient descent (SGD) with L_2 regularization** when working with sparse data.

Algorithm 1 Sparse SGD Algorithm for Logistic Regression with Regularization

```
1: Initialize $c_i \leftarrow 0$ for $i \in \{1, 2, \dots, d\}$
2: for $j \in \{1, 2, \dots, n\}$ do
3:     $\hat{p} \leftarrow \frac{1}{1+\exp(-\sum_k w_k x_k^{(j)})}$
4:     for $i$ such that $x_i^{(j)} \neq 0$ do
5:         $k \leftarrow j - c_i$        ▷ auxiliary variable $c_i$ holds the index of last time we see $x_i^{(j)} \neq 0$
6:         $\mathbf{w}_i \leftarrow \mathbf{w}_i(1 - \eta\lambda)^k$        ▷ apply all the regularization updates
7:         $\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta x_i^{(j)}(y^{(j)} - \hat{p})$        ▷ regularization is done in previous step
8:         $c_i \leftarrow j$        ▷ remember last time we see $x_i^{(j)} \neq 0$
9:     end for
10: end for
```

The idea is to only update $\mathbf{w}_i$ when $x_i^{(j)} \neq 0$. Before we do the update, we apply all the regularization updates we skipped before, using the answer from the previous question. You can checkout Algorithm 1 for details. Using this trick, each update takes $\mathcal{O}(s)$ time. (Note: we can use the same trick applies for SGD with l_1 regularization.)