# Natural Language Processing - CC01 (Math Exercise - Lab 6,7)

Le Thi Phuong Thao - 2252757

March 2025

## 1 Problem 1

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

Our goal is to estimate $b$ using the least squares method by minimizing the squared error:

$$J(a, b) = \sum(y_i - ax_i - bx_i^2)^2.$$

a) To find the optimal value for $b$, we take the partial derivative of $J$ with respect to $b$ and set it to zero.

We have the derivative:

$$\frac{\partial J}{\partial b} = \sum -2x_i^2(y_i - ax_i - bx_i^2) = 0$$

Therefore, solving for $b$:

$$b = \frac{\sum(y_i - ax_i) \cdot x_i^2}{\sum x_i^4}$$

b) Which model fits the training data better?

Model 1 is a simple linear regression model, which assumes a linear relationship between $x$ and $y$:

$$y = ax + e$$

Model 2 is a quadratic regression model, which extends simple linear regression by adding a squared term $bx^2$:

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

This allows Model 2 to capture nonlinear relationships between $x$ and $y$.

Since Model 2 includes an additional term ($bx^2$), it has more flexibility to fit the training data. It can model both linear and quadratic patterns, leading to a lower or equal training error compared to Model 1.

**(b)Model 2 fits the training data better**

c) Which of the two models is more likely to fit the test data better and explain?

If the true relationship in the test data is linear, Model 1 will generalize better, while Model 2 may overfit the training data by capturing noise.

If the true relationship in the test data is quadratic, Model 2 will generalize better because it can capture the curvature of the data.

Since we do not know the true function behind the test data, it is impossible to determine which model will generalize better. Model 2 might overfit the data, while Model 1 might be too simple to capture important patterns.

**(d) Impossible to tell**

## 2 Problem 2

**a)** Comparing Models $A : y = w^2x$ and $B : y = wx$

We compare the following models:

1. Model A:

$$y = w^2x$$

2. Model B:

$$y = wx$$

**Model A:**

- The coefficient $w^2$ is always non-negative due to squaring.
- This means the relationship between $x$ and $y$ is always in the same direction (positive correlation).

**Model B:**

- A standard linear regression model where $y$ changes proportionally with $x$.
- The coefficient $w$ can be positive or negative, allowing for both positive and negative correlations.

- If the true relationship is **linear** ($y = wx$), then Model B will fit better.

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

**b)** Comparing Models $A : y = w_1^2x + w_2x$ and $B : y = wx$ We compare the following models:

1. Model A:

$$y = w_1^2x + w_2x$$

This model has two parameters, $w_1^2$ and $w_2$, both multiplying $x$, allowing more flexibility in fitting different datasets.

2. Model B:

$$y = wx$$

This is a simple linear regression model with only one parameter.

- Model A has more flexibility due to the additional parameter $w_1^2$, allowing it to capture more complex patterns in the data.

- If the true relationship is strictly linear, Model B may generalize better, while Model A could introduce unnecessary complexity.

- If the data has an underlying structure that benefits from the additional flexibility of Model A, then it will perform better.

Since different datasets may favor different models:

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

There exist datasets where Model A performs better and datasets where Model B performs better.

## 3 Problem 3

Derive the optimal value for w_1 when using least squares as the target minimization function (w_2 may appear in your resulting equation).
We are given a regression model of the form:

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

and we want to find the optimal value of $w_1$ using the least squares minimization function. The least squares error function is given by:

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

To find the optimal $w_1$, we differentiate the loss function with respect to $w_1$ and set it to zero.
Taking the partial derivative of $L(w_1, w_2)$ with respect to $w_1$:

$$\begin{aligned}\frac{\partial L}{\partial w_1} &= \sum_{i=1}^{n} 2 \left( y_i - (w_1^2 x_{i,1} + w_2^2 x_{i,2}) \right) (-2 w_1 x_{i,1}) \\ &= -4 w_1 \sum_{i=1}^{n} x_{i,1} \left( y_i - w_1^2 x_{i,1} - w_2^2 x_{i,2} \right)\end{aligned}$$

Setting this derivative to zero:

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

Expanding:

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

Solve for $w_1$:

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

Since $w_1^2$ is squared, there are two possible values for $w_1$:

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

Thus, there are two possible values for $w_1$, corresponding to the positive and negative square roots of the expression.

## 4 Problem 4

**a)** Regularization terms:
- **Equation (4.1)** is the standard least squares objective:

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

- **Equation (4.2)** includes $L_2$ regularization:

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

- **Equation (4.3)** includes $L_1$ regularization:

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

**b)** For large values of $\lambda$ in objective 4.2, the bias would increase, decrease, or remain unaffected?

When $\lambda$ increases, the algorithm penalizes large weights, forcing them to decrease in order to optimize the loss function.

This leads to:
- Increase in bias: The model becomes simpler and may not learn enough from the data, leading to underfitting.
- Decrease in variance: The model becomes more stable and less sensitive to noise in the training data.

Thus, a larger $\lambda$ makes the model more constrained, reducing flexibility but improving generalization in some cases.

**Answer: Bias increases.**

c) For large values of $\lambda$ in objective 4.3, the variance would increase, decrease, or remain unaffected?

When $\lambda$ increases in L1 regularization, the model forces some weights to be exactly zero, simplifying the model and reducing complexity.

- Variance decreases: The model becomes less sensitive to noise, improving generalization. - Bias may increase: If $\lambda$ is too large, important features may be removed, leading to underfitting.

**Answer: Variance decreases**

## 5 Problem 5

a) Plot the decision boundary and label the region would predict Y = 1 and Y = 0
b) If the model is trained using logistic regression, the probability of $Y = 1$ is computed using the sigmoid function:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

where:

$$z = w_1 X_1 + w_2 X_2 + w_0 = 3X_1 + 5X_2 - 15$$

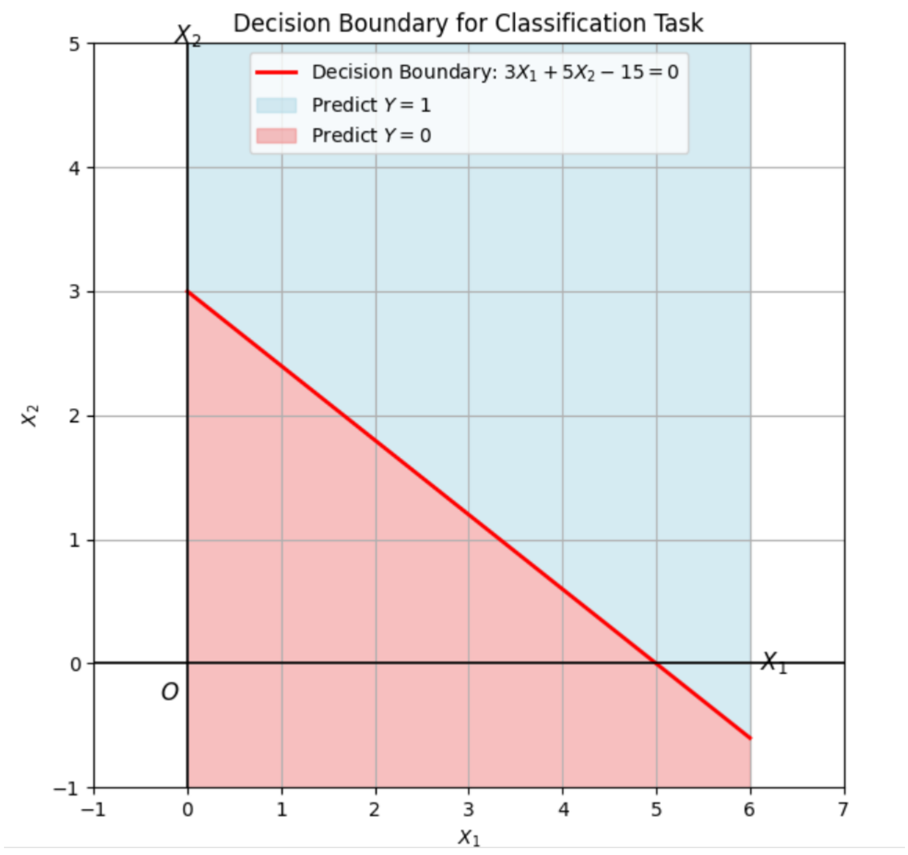

Figure 1: Decision boundary for $3X_1 + 5X_2 - 15 = 0$. The upper region (blue) predicts $Y = 1$, and the lower region (red) predicts $Y = 0$.

Thus, the probability prediction is:

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

## 6 Problem 6

### a)

We consider condition 1 as the blue line and condition 2 as the red line. The classification errors are:

- Condition 1 has 1 classification error (the $x$ point).
- Condition 2 has 1 classification error (the second $o$ point).

**b)** Classifier 1:

- $P(y = 0 \mid x = -1; w) \approx 1 - 0.35 \approx 0.65$
- $P(y = 1 \mid x = 0; w) \approx 0.35$
- $P(y = 0 \mid x = 1; w) \approx 1 - 0.35 \approx 0.65$

The joint probability is approximately 0.147.
For Classifier 2:

- $P(y = 0 \mid x = -1; w) = 1$
- $P(y = 1 \mid x = 0; w) = 1$
- $P(y = 0 \mid x = 1; w) = 0$

The joint probability is 0.

From the joint probability, we can conclude that the maximum likelihood solution is classifier 1.

**c)** The maximum likelihood solution is Classifier 1, which means $w_1 = 0$. Therefore, the regularization penalty $\frac{|w_1|^2}{2}$ (which affects only $w_1$) does not influence the ML solution. If we apply regularization, our solution remains the same.

## 7 Problem 7

**a)** SGD Update Rule for $\lambda = 0$
Since $\lambda = 0$, the logistic objective function:

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

Let

$$z = \sum_{i=1}^{d} w_i x_i^{(j)}$$

The derivative of $l$ with respect to $w_i$ is:

$$\frac{\partial l}{\partial w_i} = \underbrace{y^{(j)} x_i^{(j)}}_{\text{Term 1}} - \underbrace{\frac{\exp(z)}{1 + \exp(z)} x_i^{(j)}}_{\text{Term 2}}$$

Simplifying using the logistic function

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

we get:

$$\frac{\partial l}{\partial w_i} = x_i^{(j)} \left( y^{(j)} - \sigma(z) \right)$$

$$\frac{\partial l}{\partial w_i} = x_i^{(j)} \left( y^{(j)} - \frac{1}{1 + \exp\left( -\sum_{i=1}^{d} w_i x_i^{(j)} \right)} \right)$$

For each weight $w_i$, the update rule is:

$$w_i \leftarrow w_i + \eta (y^{(j)} - \sigma(w^T x^{(j)})) x_i^{(j)}$$

where the sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

**b)** Time complexity for $\lambda = 0$:
**Dense data structure:** O(d) time (iterates over all $d$ features) (In a dense structure, each vector $x^{(j)}$ is represented by an array of $d$ elements, including zero values. When performing

SGD updates for each weight $w_i$, we must iterate over all $d$ elements of $x^{(j)}$. )

**Sparse data structure:** O(s) time (iterates only over $s$ non-zero features) (In a sparse structure, only the nonzero elements of $x^{(j)}$ are stored, along with their indices. When updating SGD, we only need to iterate over the $s$ nonzero elements of $x^{(j)}$.)

**c)** For L2-regularized logistic regression, the SGD update rule for weight $w_i$ when processing example $(x^{(j)}, y^{(j)})$ is:

$$w_i \leftarrow w_i(1-\eta\lambda) + \eta(y^{(j)} - \sigma(\mathbf{w}^T x^{(j)}))x_i^{(j)}$$

**d)** Time Complexity for $\lambda = 0$ with dense data

$$O(d) \text{ time, as all weights must decay by } (1-\eta\lambda), \text{ regardless of sparsity.}$$

**e)** From part (c), the SGD update rule when $\lambda > 0$ is:

$$w_i \leftarrow w_i(1-\eta\lambda) + \eta(y^{(j)} - \sigma(\mathbf{w}^T x^{(j)}))x_i^{(j)}$$

If $x_i^{(j)} = 0$, the second term (gradient of the loss):

$$w_i \leftarrow w_i(1-\eta\lambda).$$

Applying $k$ consecutive updates:
Assume that after $t$ steps, the weight is $w_i^{(t)}$.

- At step $t+1$:

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

- At step $t+2$:

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

- After $k$ steps:

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

**f)** Algorithm: Track the last update time for each weight. When encountering a non-zero $x_i^{(j)}$:

- Compute the elapsed steps $k$ since the last update.
- Apply decay $(1-\eta\lambda)^k$.
- Update $w_i$ using the current example.

**Average Time Complexity:**

$O(s)$ per example, as only non-zero features are processed, and decay is applied lazily.