Wang Xiyu

National University of Singapore
School of Computing
CS2109S: Introduction to AI and Machine Learning
Semester 2, 2024/2025

Tutorial 4
Decision Trees and Linear Models

These questions will be discussed during the tutorial session. Please be prepared to answer them.

Summary of Key Concepts

In this tutorial, we will discuss and explore the following learning points from Lecture:

1. Decision Trees
   (a) Decision Tree Learning Algorithm
2. Linear Models
   (a) Normal Equation
3. Cost functions
4. Gradient Descent
   (a) Effect of learning rates

A Decision Tree

The loans department of DBN (Development Bank of NUS) wanted to use a decision tree to help them make decisions for new applicants. DBN has the following past loan processing records, each containing an applicant's income, credit history, debt, and the final approval decision. Details are shown in Table 1.

Income	Credit History	Debt	Decision
Over 10k	Bad	Low	Reject
Over 10k	Good	High	Approve
0 - 10k	Good	Low	Approve
Over 10k	Good	Low	Approve
Over 10k	Good	Low	Approve
Over 10k	Good	Low	Approve
0 - 10k	Good	Low	Approve
Over 10k	Bad	Low	Reject
Over 10k	Good	High	Approve
0 - 10k	Bad	High	Reject

Table 1: Loan Processing Outcomes

1. Construct the best decision tree to classify the final outcome (Decision) from the three features Income, Credit History, and Debt.

A

Construct Decision Tree

```
if (Credit History == Bad) -> Reject
else -> accept
```

- Income: 0 - 10k -> 0; Over 10k -> 1
- Credit History: Bad -> 0; Good -> 1
- Debt: High -> 0; Low -> 1

$IG(Income) = 1 - [\frac{3}{10}I(\frac{2}{3},\frac{1}{3}) + \frac{7}{10}I(\frac{6}{7},\frac{1}{7})] = 1 - [-\frac{3}{10}(\frac{2}{3}log_2(\frac{2}{3}) - \frac{1}{3}log_2(\frac{1}{3})) + \frac{7}{10}(\frac{6}{7}log_2(\frac{6}{7}) - \frac{1}{7}log_2(\frac{1}{7}))] = 1 - [\frac{3}{10}(\frac{2}{3}(log_2(2) - log_2(3) -))]...$

4

- More data
- add feature
- min sample pruning
- remove outliers

3

2. It turns out in the labeling process, one of the data is mislabeled. The data in table 1 is now altered into table 2.

Income	Credit History	Debt	Decision
Over 10k	Bad	Low	Approve
Over 10k	Good	High	Approve
0 - 10k	Good	Low	Approve
Over 10k	Good	Low	Approve
Over 10k	Good	Low	Approve
Over 10k	Good	Low	Approve
0 - 10k	Good	Low	Approve
Over 10k	Bad	Low	Reject
Over 10k	Good	High	Approve
0 - 10k	Bad	High	Reject

Table 2: Loan Processing Outcomes (Noisy)

Construct the best decision tree that classifies the data in table 2. Justify this decision tree and show why it performs the best by calculating the entropy, information gain values and remainders at each stage.
(Note: $\log_2 \frac{x}{y} = \log_2 x - \log_2 y$, $\log_2 1 = 0$, $\log_2 2 = 1$, $\log_2 3 = 1.585$, $\log_2 4 = 2$, $\log_2 5 = 2.322$, $\log_2 6 = 2.585$, $\log_2 7 = 2.807$, $\log_2 8 = 3$, $\log_2 9 = 3.170$, $\log_2 10 = 3.322$)

3. What is the decision made by the decision tree in question 2 for a person with an **income over 10k**, a **bad credit history**, and **low debt**?

4. The situation in question 3 demonstrates inconsistent data in the decision tree i.e. the attribute does not provide information to differentiate the two classes. In practice, it is usually left undecided. What are some ways to to mitigate inconsistent data?

5. Let's consider a scenario where you desire a Decision Tree with each leaf node representing a minimum of 3 training data points. Derive the tree by pruning the tree you previously obtained in question 2. Which data(s) do you think are likely outlier(s)?

B Linear Regression Model Fitting

You are given several data points as follows:

x_1	x_2	x_3	y
6	4	11	20
8	5	15	30
12	9	25	50
2	1	3	7

1. Apply the Normal Equation formula to obtain a linear regression model that minimizes MSE of the data points.
   Note: `numpy.linalg.inv` may be used for computing the matrix inverse. While you have seen Gaussian elimination before, we do not expect you to invert matrices in a time-constraint setting.

$$w = (X^T X)^{-1} X^T Y$$

2. The Normal Equation requires the calculation of $(X^T X)^{-1}$. However, $(X^T X)$ is sometimes not invertible. When does this occur? What steps should be taken in such situations?
3. **(Bonus)** Can you derive the Normal Equation from the Mean Squared Error (MSE)?

1

$$IG(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - remainder(A)$$

$$I(P(+), P(-)) = -\frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{n}{p+n} log_2 \frac{n}{p+n}$$

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i})$$

```
import numpy as np
def compute_Info_boolean(pos, neg):
    pos_prob = np.divide(pos, (pos + neg), dtype=np.float128)
    neg_prob = np.divide(neg, (pos + neg), dtype=np.float128)
    return - pos_prob * np.log2(pos_prob) - neg_prob * np.log2(neg_prob)

def entropy(pos, neg):
    total = pos + neg
    if total == 0:
        return 0
    pos_prob = np.divide(pos, total, dtype=np.float128)
    neg_prob = np.divide(neg, total, dtype=np.float128)
    entropy = 0
    if pos_prob > 0:
        entropy -= pos_prob * np.log2(pos_prob)
    if neg_prob > 0:
        entropy -= neg_prob * np.log2(neg_prob)

    return entropy

def compute_IG_boolean(parent_pos, parent_neg, left_pos, left_neg, right_pos, right_neg):
    parent_entropy = entropy(parent_pos, parent_neg)
    total_parent = parent_pos + parent_neg
    total_left = left_pos + left_neg
    total_right = right_pos + right_neg
    left_weight = np.divide(total_left, total_parent, dtype=np.float128) if total_left > 0 else 0
    right_weight = np.divide(total_right, total_parent, dtype=np.float128) if total_right > 0
        else 0
    weighted_child_entropy = left_weight * entropy(left_pos, left_neg) + right_weight *
        entropy(right_pos, right_neg)
    return parent_entropy - weighted_child_entropy



x1 = np.matrix([[6], [8], [12], [2]])
x2 = np.matrix([[4], [5], [9], [1]])
x3 = np.matrix([[11], [9], [25], [3]])
y = np.matrix([[20], [1], [3], [7]])
X = np.hstack([x1, x2, x3])
Xt = np.transpose(X)
w = np.linalg.inv(Xt @ X) @ Xt @ y
print(w)
```

2

When $det(X^T X) = 0$, Linearly dependent column: Highly correlated features.

- remove feature
- regularization
- pseudo invert
- use gradient descent

Normal Equation

Derive:

$$w = (X^T X)^{-1} X^T y. \tag{1}$$

From:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}^{(i)} - y^{(i)})^2 \tag{2}$$

Proof

We start with the Mean Squared Error (MSE) function:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}^{(i)} - y^{(i)})^2, \tag{3}$$

where $\hat{y}^{(i)} = h_w(x^{(i)}) = w^T x^{(i)}$, with $w \in \mathbb{R}^{d\times 1}$ and $y \in \mathbb{R}^{N\times 1}$. Expanding in terms of matrix notation, let $A = Xw - y$, where $X \in \mathbb{R}^{N\times d}$ with rows $x^{(i)}$, then:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (w^T x^{(i)} - y^{(i)})^2 = \frac{1}{N}(A^T A). \tag{4}$$

Rewriting:

$$MSE = \frac{1}{N}(Xw - y)^T (Xw - y), \tag{5}$$

which expands further as:

$$MSE = \frac{1}{N}\left(w^T X^T X w - w^T X^T y - y^T X w + y^T y\right). \tag{6}$$

Since $w^T X^T X w$ and $y^T X w$ are scalars, we simplify:

$$MSE = \frac{1}{N}\left(w^T X^T X w - 2 w^T X^T y + y^T y\right). \tag{7}$$

To minimize MSE, we take the partial derivative with respect to w:

$$\frac{\partial}{\partial w}\left[\frac{1}{N}\left(w^T X^T X w - 2 w^T X^T y + y^T y\right)\right]. \tag{8}$$

Differentiating term by term:

$$\frac{\partial}{\partial w}(w^T X^T X w) = 2 X^T X w, \tag{9}$$

$$\frac{\partial}{\partial w}(-2 w^T X^T y) = -2 X^T y, \tag{10}$$

$$\frac{\partial}{\partial w}(y^T y) = 0. \tag{11}$$

Thus,

$$\frac{\partial}{\partial w}(MSE) = \frac{1}{N}(2X^T X w - 2X^T y). \tag{12}$$

Setting the derivative to zero for minimization:

$$\frac{2}{N}X^T X w - \frac{2}{N}X^T y = 0. \tag{13}$$

$$X^T X w = X^T y. \tag{14}$$

Solving for w, assuming $X^T X$ is invertible:

$$w = (X^T X)^{-1} X^T y. \tag{15}$$

C Examining Cost Functions

For Linear Regression, there are two popular cost functions,

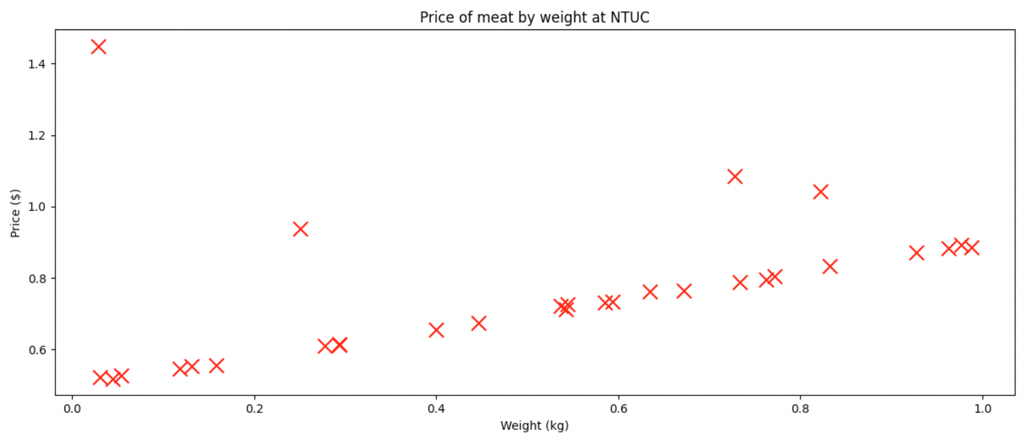
Mean Squared Error:

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

and **Mean Absolute Error**:

$$L(y, \hat{y}) = \frac{1}{2}|y - \hat{y}| \tag{2}$$

1. Given the scatter plot of a dataset containing the weight of meat at NTUC (x), as measured by Bob, and its corresponding price (y), justify your choice of cost function for this problem.
   Hint: As a human, Bob is prone to making the occasional mistake when taking measurements. As we would like our resultant model to not be too affected by these outliers, it might be helpful to consider how the cost functions penalize outliers.



2. Can you provide examples of cost functions that are better suited to handle outliers more effectively?

B

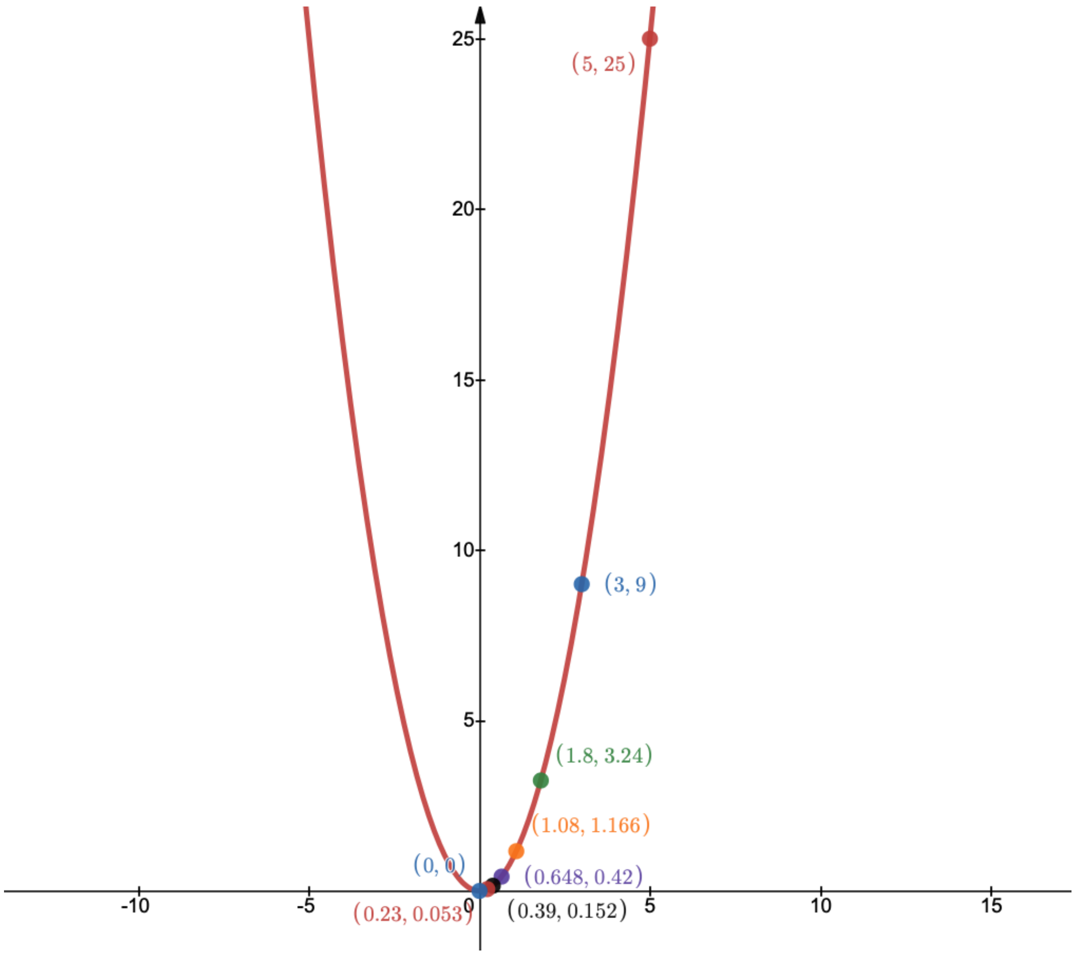
a

To minimize the impact of outliers, use MAE it penalize the impact of outliers less

D Choosing Learning Rates

1. Given a simple function $y = x^2$, we know the gradient is $\frac{dy}{dx} = 2x$. As such, the minimum of this function is 0.

   Start from the point $a = (5, 25)$, and using the different α values from $\{10, 1, 0.1, 0.01\}$, perform Gradient Descent on values of x. Record the value of $a = (x, y)$ (where $y = x^2$) over **5 iterations** for each learning rate in tabular format. Observe the oscillations of the value and the convergence to $(0, 0)$. An example is shown below for $\alpha = 0.2$ over 7 steps:



2. During the course of training for a large number of epochs/iterations, what can be done to the value of the learning rate α to enable better convergence?

   Hint: Consider the equation for gradient descent $w_j = w_j - \alpha\frac{\partial J(w_1,w_2,...)}{\partial w_j}$

1