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

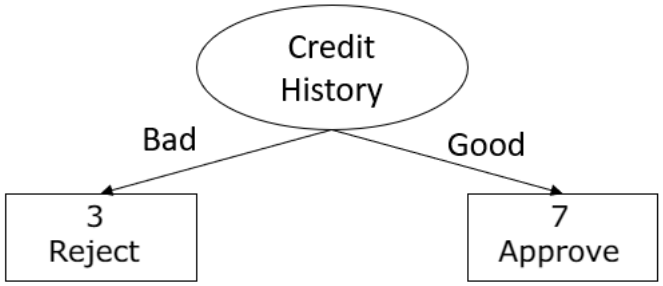
The loans department of DBN (Development Bank of NUS) wanted to use a decision tree to help them make decisions for new applicants. DBN has the following past loan processing records, each containing an applicant's income, credit history, debt, and the final approval decision. Details are shown in Table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

Solution:



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)

Solution:

Calculate initial entropy:

$$I(\frac{2}{10}, \frac{8}{10}) = -\frac{2}{10}\log_2\frac{2}{10} - \frac{8}{10}\log_2\frac{8}{10} = 0.722$$

By choosing to split on Income, we have 6 Approve, 1 Reject for Over 10k and

2 Approve, 1 Reject for 0-10k:

$$\begin{aligned} remainder(Income) &= \frac{3}{10}I(\frac{2}{3},\frac{1}{3}) + \frac{7}{10}I(\frac{6}{7},\frac{1}{7}) \\ &= \frac{3}{10}(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}) + \frac{7}{10}(-\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7}) \\ &= 0.690 \\ IG(Income) &= I(\frac{2}{10},\frac{8}{10}) - remainder(Income) \\ &= 0.722 - 0.690 = \mathbf{0.032} \end{aligned}$$

By choosing to split on Debt, we have 6 Approve, 1 Reject for Low and 2 Approve, 1 Reject for High:

$$\begin{aligned} remainder(Debt) &= \frac{3}{10}I(\frac{2}{3},\frac{1}{3}) + \frac{7}{10}I(\frac{6}{7},\frac{1}{7}) \\ &= remainder(Income) \\ &= 0.690 \\ IG(Debt) &= I(\frac{2}{10},\frac{8}{10}) - remainder(Debt) \\ &= 0.722 - 0.690 = \mathbf{0.032} \end{aligned}$$

By choosing to split on Credit History, we have 7 Approve for Good and 1 Approve, 2 Reject for Bad:

$$\begin{aligned} remainder(CreditHistory) &= \frac{3}{10}I(\frac{1}{3},\frac{2}{3}) + \frac{7}{10}I(\frac{7}{7},\frac{0}{7}) \\ &= \frac{3}{10}(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}) + \frac{7}{10}(-\frac{7}{7}\log_2\frac{7}{7} - \frac{0}{7}\log_2\frac{0}{7}) \\ &= \frac{3}{10}(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}) + \frac{7}{10}(-0-0) \\ &= 0.275 \\ IG(CreditHistory) &= I(\frac{2}{10},\frac{8}{10}) - remainder(CreditHistory) \\ &= 0.722 - 0.275 = \mathbf{0.447} \end{aligned}$$

From the computations above, we may notice the following two facts (which may be proved rigorously):

1. If splitting on attribute a results in the same distribution of samples in the child nodes as splitting on attribute a', then the entropy in their child nodes are equal (Debt and Income in our case).

2. If a node has all its samples with the same classification then its entropy is 0 (See child node for Credit History = Good).

Among the available attributes, Credit History has the highest information gain and thus we choose it as our root.

Furthermore, since all examples for Credit History = Good have the same classification, we only need to build a subtree for Credit History = Bad. For Credit History = Bad, a subtree for the following subset of examples is to be constructed:

Income	Debt	Decision
Over 10k	Low	Approve
Over 10k	Low	Reject
0 - 10k	High	Reject

Calculate entropy of samples in Credit History = Bad node:

$$I(\frac{1}{3},\frac{2}{3}) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918$$

By choosing to split on Income, we have 1 Approve, 1 Reject for Over 10k and 1 Reject for 0-10k:

$$\begin{aligned} remainder(Income) &= \frac{1}{3}I(\frac{0}{1},\frac{1}{1}) + \frac{2}{3}I(\frac{1}{2},\frac{1}{2}) \\ &= \frac{1}{3}(-\frac{0}{1}\log_2\frac{0}{1} - \frac{1}{1}\log_2\frac{1}{1}) + \frac{2}{3}(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}) \\ &= 0.667 \\ IG(Income) &= I(\frac{1}{3},\frac{2}{3}) - remainder(Income) \\ &= 0.918 - 0.667 = \mathbf{0.251} \end{aligned}$$

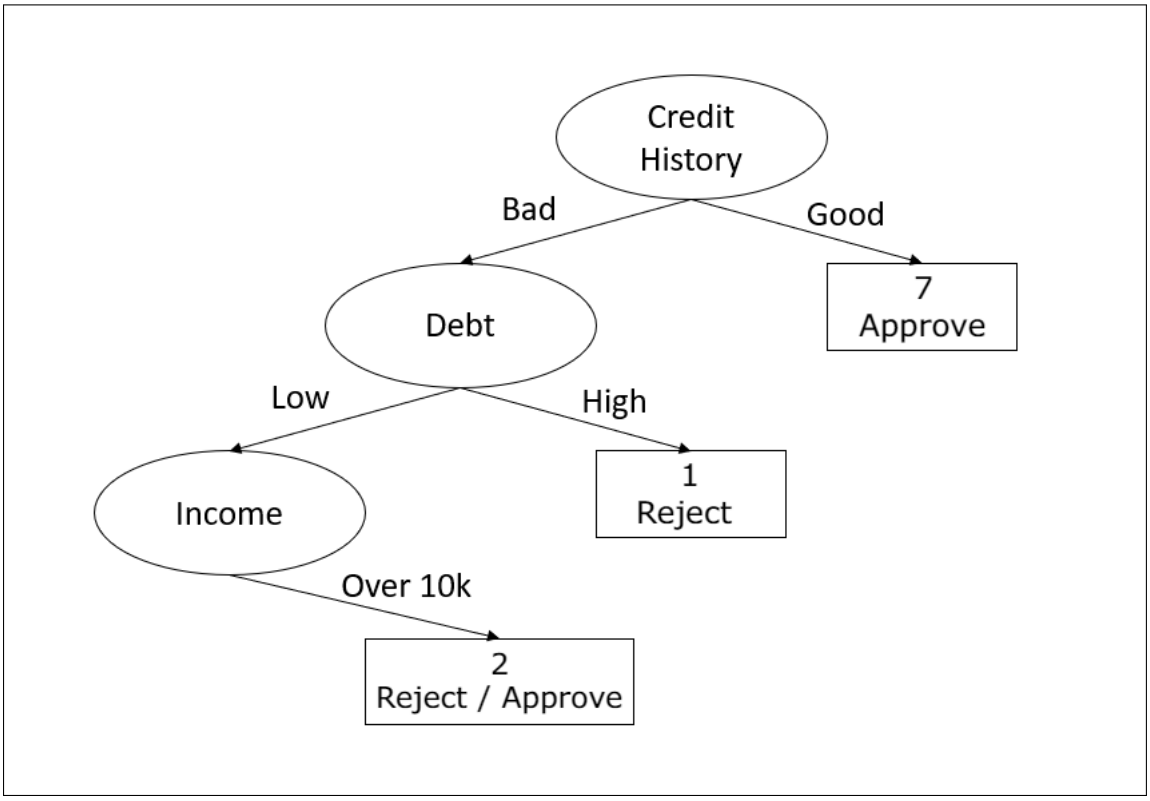
By choosing to split on Debt, we have 1 Approve, 1 Reject for Low and 1 Reject for High:

$$\begin{aligned} remainder(Debt) &= remainder(Income) \\ &= 0.667 \\ IG(Debt) &= I(\frac{1}{3},\frac{2}{3}) - remainder(Debt) \\ &= 0.918 - 0.667 = \mathbf{0.251} \end{aligned}$$

Since Debt has the same information gain as Income, we can arbitrarily choose the root. Suppose we choose Debt as the root. For Debt = Low, a subtree for the following subset of examples is to be constructed:

Income	Decision
Over 10k	Approve
Over 10k	Reject

Since no more splitting can be done, we can construct the final tree as shown below:



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**?

Solution:

For decision tree in question 2, the decision might either reject or accept the person.

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?

Solution:

Multiple ways to handle inconsistent data:

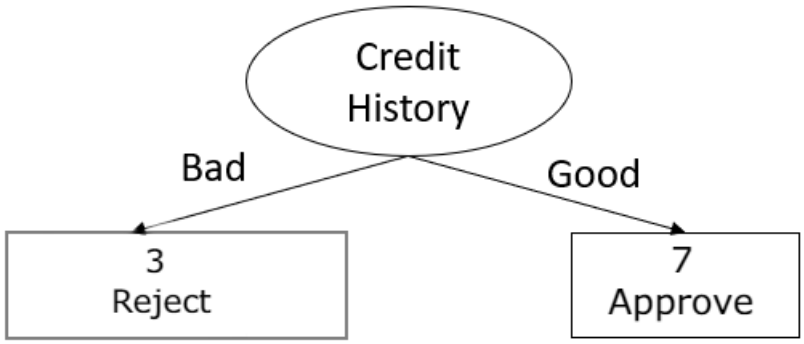
1. Pruning to remove unnecessary branches and nodes, creating simpler decision tree. Examples include Min-sample and Max-depth.
2. Pre-processing of data to remove outliers that create noise.
3. Select only relevant features, so less relevant features that could create inconsistencies are not part of the decision tree.
4. Collect more data on new features to clearly differentiate the inconsistent classes.

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)?

Solution:

The outlier is likely to be the person with an income over 10k, a bad credit history, and low debt.



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.

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

Solution:

We have the following matrices:

$$X = \begin{bmatrix} 1 & 6 & 4 & 11 \\ 1 & 8 & 5 & 15 \\ 1 & 12 & 9 & 25 \\ 1 & 2 & 1 & 3 \end{bmatrix}, X^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 6 & 8 & 12 & 2 \\ 4 & 5 & 9 & 1 \\ 11 & 15 & 25 & 3 \end{bmatrix}, Y = \begin{bmatrix} 20 \\ 30 \\ 50 \\ 7 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 4 & 28 & 19 & 54 \\ 28 & 248 & 174 & 492 \\ 19 & 174 & 123 & 347 \\ 54 & 492 & 347 & 980 \end{bmatrix}, (X^T X)^{-1} = \begin{bmatrix} \frac{11}{2} & -\frac{23}{4} & -4 & 4 \\ -\frac{23}{4} & \frac{15}{2} & \frac{13}{2} & -\frac{23}{4} \\ -4 & \frac{13}{2} & 14 & -8 \\ 4 & -\frac{23}{4} & -8 & \frac{11}{2} \end{bmatrix}$$

Applying the normal equation, we obtain:

$$\begin{aligned} w &= (X^T X)^{-1} X^T Y \\ &= \begin{bmatrix} 4 & -5.5 & -7 & 7 \end{bmatrix}^T, \\ \hat{y} &= 4 - 5.5x_1 - 7x_2 + 7x_3 \end{aligned}$$

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?

Solution:

The matrix $(X^T X)$ becomes non-invertible when it is singular, which occurs when its columns are linearly dependent.

Linear dependency of columns occurs as a result of redundant or highly correlated features.

In these cases, several approaches can be used to address the issue:

1. **Gradient Descent:** Instead of relying on the Normal Equation, use gradient descent to iteratively find the weights that minimize the cost function.
2. **Regularization:** Apply regularization techniques (such as Ridge or Lasso regression) to modify the cost function, adding a penalty term that reduces the impact of multicollinearity and makes the matrix $(X^T X + \lambda I)$ (where λ is a regularization parameter) invertible.
3. **Pseudoinverse:** Utilize the Moore-Penrose pseudoinverse, which provides a way to compute a solution even when $(X^T X)$ is not invertible.

Note: Even if $(X^T X)$ is technically invertible, it might still be ill-conditioned, meaning that small changes in data can cause large variations in the result. This occurs when rows or columns are nearly linearly dependent, or when the number of features is too high relative to the number of data points, resulting in a nearly singular matrix. In such cases, the Normal Equation may still be challenging to use, and regularization or gradient-based methods are often preferred.

3. **(Bonus)** Can you derive the Normal Equation from the Mean Squared Error (MSE)?

Solution:

The cost function $J(w)$, which we aim to minimize is the Mean Squared Error (MSE):

$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \frac{1}{2m} \|Y - Xw\|^2$$

Recall the definition of the (Euclidean) norm of a vector $\mathbf{v}$:

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} = \sqrt{\mathbf{v} \cdot \mathbf{v}} = \sqrt{\mathbf{v}^T \mathbf{v}}$$

and note that:

$$\|\mathbf{v}\|^2 = \mathbf{v}^T \mathbf{v}$$

We can simplify this using vector notation:

$$J(w) = \frac{1}{2m} (Y - Xw)^T (Y - Xw)$$

Expand the quadratic term:

$$J(w) = \frac{1}{2m}\left[Y^TY - Y^TXw - (Xw)^TY + (Xw)^T(Xw)\right]$$

Since $(Xw)^TY$ is a scalar, it is equal to its transpose, so:

$$J(w) = \frac{1}{2m}\left[Y^TY - 2Y^TXw + w^TX^TXw\right]$$

To minimize $J(w)$, take the derivative with respect to w and set it to zero:

$$\frac{\partial J(w)}{\partial w} = -\frac{1}{m}X^TY + \frac{1}{m}X^TXw = 0$$

Simplify by multiplying through by m:

$$-X^TY + X^TXw = 0$$

Rearrange to solve for w:

$$X^TXw = X^TY$$

If X^TX is invertible, we can multiply both sides by its inverse to find the normal equation:

$$w = (X^TX)^{-1}X^TY$$

This equation gives the weights w that minimize the cost function in linear regression.

Note: We can also solve for w by minimizing $J(w)$ component-wise. Assuming $w = [w_1, w_2, \ldots, w_n]^T$ is a vector with n components. The cost function $J(w)$ can be defined as:

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(y_i - \sum_{j=1}^{n}x_{ij}w_j\right)^2$$

$$\begin{aligned}J(w) = \frac{1}{2m}\Big[&(y_1 - (x_{11}w_1 + x_{12}w_2 + \cdots + x_{1n}w_n))^2 \\ &+ (y_2 - (x_{21}w_1 + x_{22}w_2 + \cdots + x_{2n}w_n))^2 \\ &+ \cdots \\ &+ (y_m - (x_{m1}w_1 + x_{m2}w_2 + \cdots + x_{mn}w_n))^2\Big].\end{aligned}$$

To find the minimum, take the partial derivative of $J(w)$ with respect to each component w_k and set it to zero:

$$\begin{aligned}\frac{\partial J(w)}{\partial w_k} = -\frac{1}{m}\Big[& x_{1k}\left(y_1 - (x_{11}w_1 + x_{12}w_2 + \cdots + x_{1n}w_n)\right) \\ & + x_{2k}\left(y_2 - (x_{21}w_1 + x_{22}w_2 + \cdots + x_{2n}w_n)\right) \\ & + \dots \\ & + x_{mk}\left(y_m - (x_{m1}w_1 + x_{m2}w_2 + \cdots + x_{mn}w_n)\right)\Big].\end{aligned}$$

$$\frac{\partial J(w)}{\partial w_k} = -\frac{1}{m}\sum_{i=1}^{m} x_{ik}\left(y_i - \sum_{j=1}^{n} x_{ij}w_j\right)$$

This represents a system of equations, one for each component w_k.

The gradient of the function $J(w)$ with respect to the vector $w = [w_1, w_2, \dots, w_n]^T$ can be written as:

$$\frac{\partial J}{\partial w} = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_2} \\ \vdots \\ \frac{\partial J}{\partial w_n} \end{bmatrix}$$

To find the optimal w, we solve:

$$\frac{\partial J}{\partial w} = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_2} \\ \vdots \\ \frac{\partial J}{\partial w_n} \end{bmatrix} = 0.$$

C Examining Cost Functions

For Linear Regression, there are two popular cost functions,

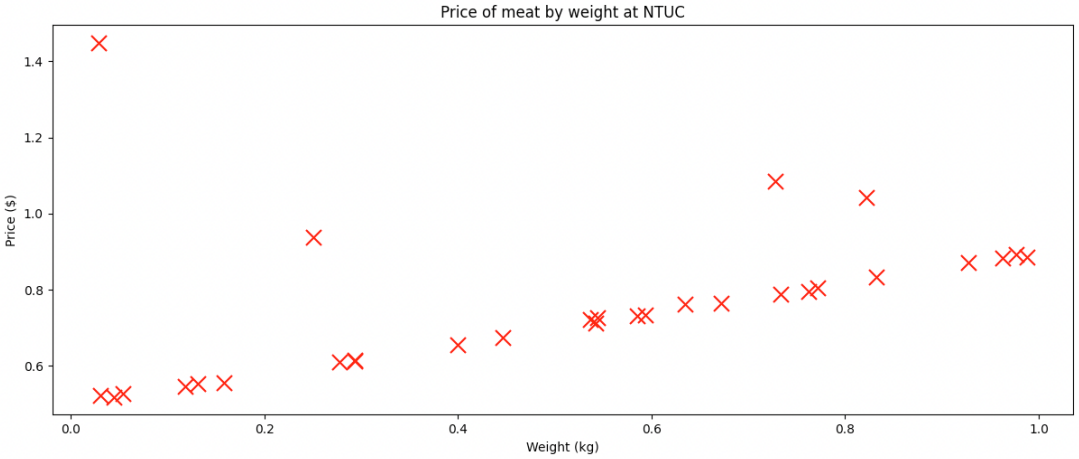
Mean Squared Error:

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

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.

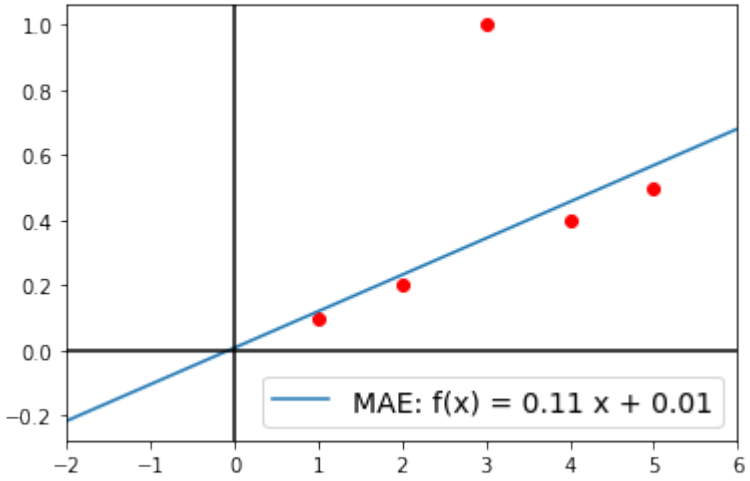


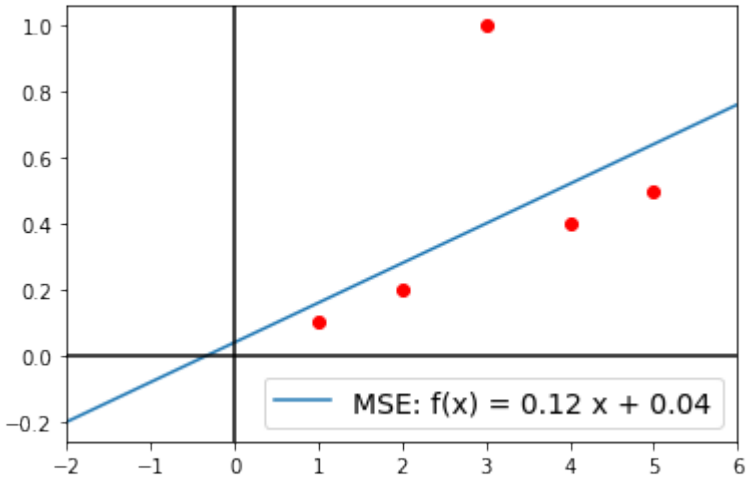
Solution:

The choice between MSE and MAE will depend on the goals of the model and the nature of the data. If we consider outliers as important and should be penalized heavily, MSE may be the preferred metric. If outliers are considered less important and should have a smaller impact, MAE may be the preferred metric.

For this dataset, there are relatively few outliers and these outliers could be a result of human error. As such, MAE is preferred because it penalises the model less compared to MSE which penalises the outliers more by squaring the error terms, resulting in larger residuals (i.e. vertical differences).

Two examples are shown below when the cost functions are MAE and MSE respectively.





Note: As shown in these examples, outliers can have a greater impact on MSE than MAE, even if the y-values are between 0 and 1.

This is because MAE penalizes small deviations (< 1) more than MSE and thus minimizing MAE avoids these smaller deviations. On the other hand, MSE penalizes large deviations (> 1) more than MAE and thus minimizing MSE reduces these larger deviations. The combined effect results in best fit lines using MSE that tend to have greater vertical distances to data points within a distance of < 1 but smaller vertical distances to data points with a distance > 1.

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

Solution:

Huber loss is a combination of MSE and MAE and is defined as:

$$L(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| \leq \delta \\ \delta \cdot |y - \hat{y}| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{3}$$

where δ is a threshold that determines the transition between the MSE and MAE behaviors.

Log-cosh loss is defined as:

$$L(y, \hat{y}) = \log(\cosh(y_i - \hat{y}_i)) \tag{4}$$

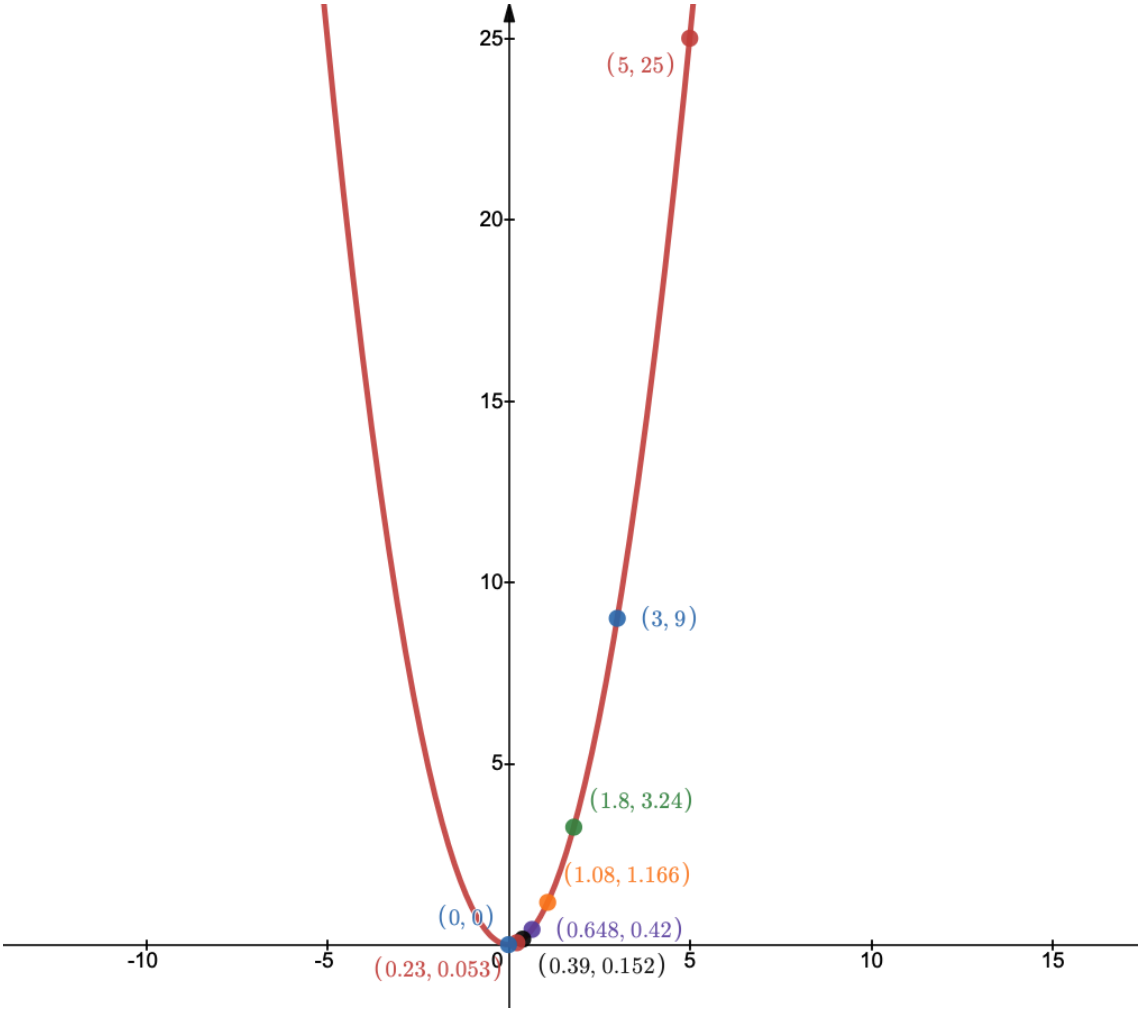
For small values of x, $\log(\cosh(x)) \approx \frac{1}{2}x^2$, which is similar to MSE. For larger values of x, $\log(\cosh(x)) \approx |x| - \log(2)$, which is similar to MAE. Log cosh approximates MSE and MAE and is similar to the Huber loss function.

Huber loss and Log cosh loss are more robust to outliers compared to MSE and MAE because they don't give as much weight to extreme values. This makes them useful in cases where the presence of outliers might negatively impact model performance if using MSE or MAE.

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 γ 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:



Solution:

The one that gives the best convergence is $\gamma = 0.1$. The corresponding x values for each γ are:

$\gamma = 10$	$\gamma = 1$	$\gamma = 0.1$	$\gamma = 0.01$
5	5	5	5
-95	-5	4.0	4.9
1805	5	3.2	4.802
-34295	-5	2.56	4.706
651605	5	2.048	4.612
-12380495	-5	1.638	4.520

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

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

   Solution:

   Learning rate γ can be decreased through the course of training. At first, we'll be taking large steps to reach the optimal values faster. As we approach the optimal values, we slow down our step size to allow for more gradual convergence. In fact, this is the logic behind a **learning rate scheduler**; it is widely used in industry and research.