National University of Singapore
School of Computing
CS2109S: Introduction to AI and Machine Learning
Semester I, 2023/2024

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

Problems

1. **Decisions That Matter**.

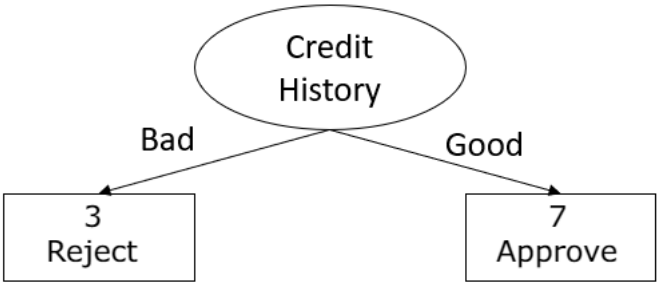
   The loans department of DBN (Development Bank of NUS) wanted to use a decision tree to help them make decisions for new applicants. DBN has the following past loan processing records, each containing an applicant's income, credit history, debt, and the final approval decision. Details are shown in Table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

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

Solution:



(b) 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 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:

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

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

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

Since Credit History has the highest gain, we choose it as the root. 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

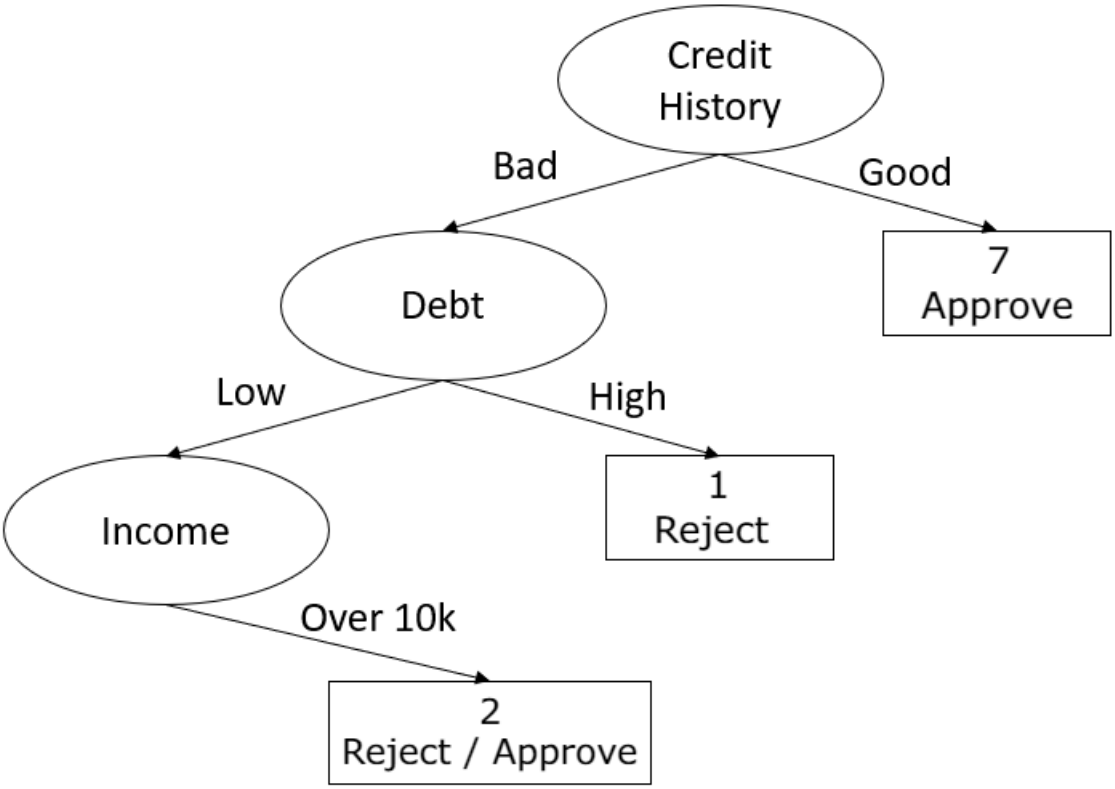
$$
\begin{aligned}
I(\frac{1}{3},\frac{2}{3}) &= -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918 \\
remainder(Income) &= \frac{1}{3}I(\frac{0}{1},\frac{1}{1}) + \frac{2}{3}I(\frac{1}{2},\frac{1}{2}) \\
&= \frac{1}{3}(-\frac{0}{1}\log_2\frac{0}{1} - \frac{1}{1}\log_2\frac{1}{1}) +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

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

Since Debt has the same 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. The final tree is shown below.



Note: *Notice that there is an inconsistency in one of the leaf nodes, possibly due to noisy data. This is covered in part (d).*

(c) What is the decision made by the decision tree in part (b) for a person with an **income over 10k**, a **bad credit history**, and **low debt**?

Solution:

For decision tree in part (b), the decision might either reject or accept the person.

(d) The situation in part (c) 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:

i. *Pruning to remove unnecessary branches and nodes, creating simpler decision tree. Examples include Min-sample and Max-depth.*

ii. *Pre-processing of data to remove outliers that create noise.*

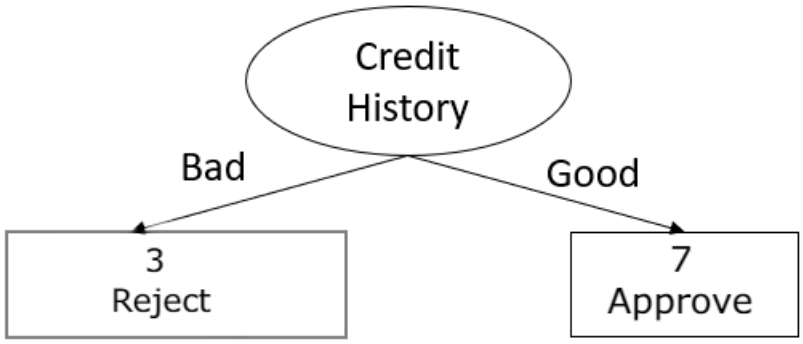
iii. *Select only relevant features, so less relevant features that could create inconsistencies are not part of the decision tree.*

iv. *Collect more data on new features to clearly differentiate the inconsistent classes.*

(e) 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 part (b). Which data(s) do you think are likely outlier(s)?

Solution:

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



2. **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

(a) Apply the Normal Equation formula to obtain a linear regression model that minimizes MSE of the data points.

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

Solution:

We can just make use of the Normal Equation to solve for the weights w*.*

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

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

(b) Normal Equation needs the calculation of $(X^T X)^{-1}$. But sometimes this matrix is not invertible. When will that happen, and what should we do in that situation?

Solution:

The matrix $X^T X$ *is not invertible if it is singular. For this to happen, some rows or columns in the dataset matrix are linearly dependent.*

Having data that is collinear or contains duplicates means that the rows are linearly dependent, while having highly correlated or redundant features means that the columns are linearly dependent.

In such instances, gradient descent can be used to arrive at weights for the linear regression model that minimise the cost function.

Note: *Even if the matrix* $X^T X$ *is invertible, it is possible for the matrix to be ill-conditioned i.e. a slight change in values drastically affects the result. This can occur if the rows and columns are almost linearly dependent, or there are too many features relative to the data points, hence resulting in an almost singular matrix that is hard to invert. Though not directly answering the question, this is a relevant limitation of using Normal Equation to find weights.*

3. **Examining Cost Functions**.

   For Linear Regression, there are two popular cost functions,

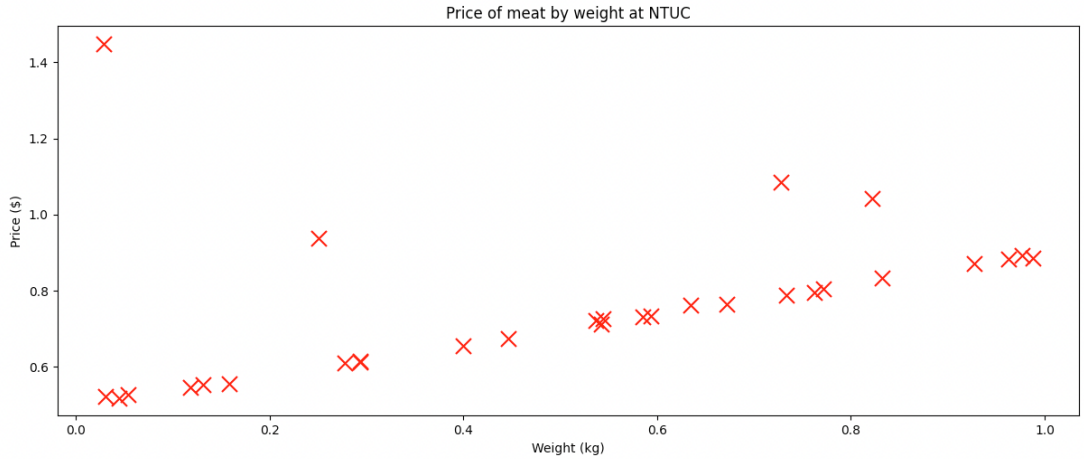
   Mean Squared Error:

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

   and **Mean Absolute Error**:

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

   (a) Given the scatter plot of a dataset containing the actual weight of meat at NTUC (x) and its price (y), justify your choice of cost function for this problem.



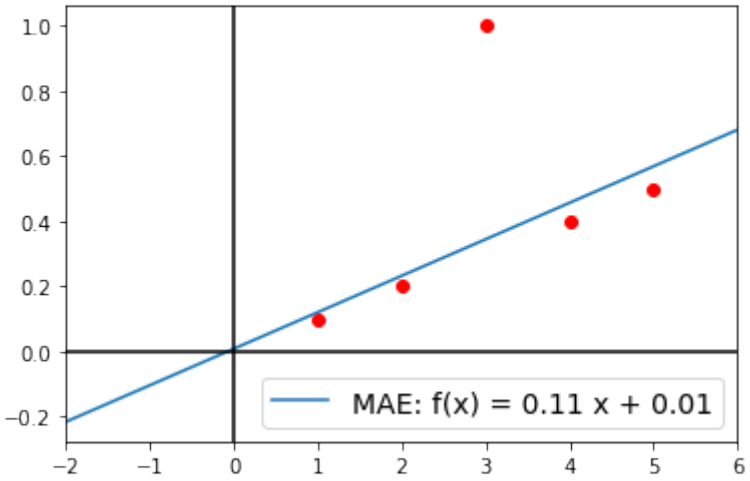
   Solution:

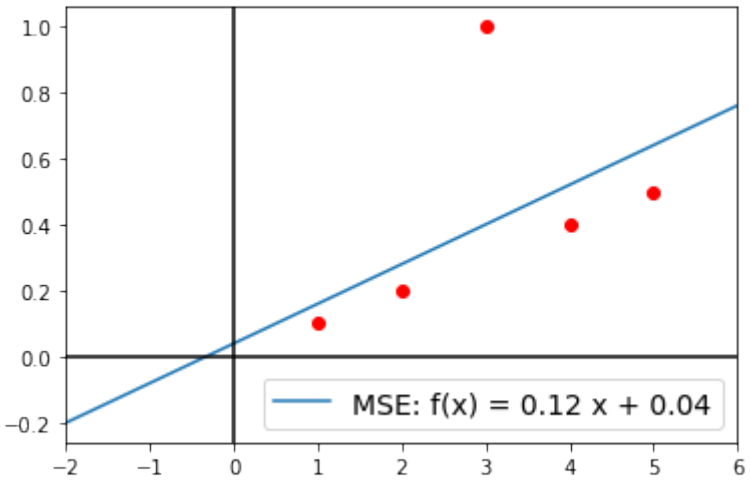
   For this dataset, it contains relatively few outliers and these outliers could be a result of human error. As such, MAE is preferred because it penalises the model less compared to MSE which penalises the outliers more by squaring the error terms, resulting in larger residuals (i.e. vertical differences).

   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.

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.





(b) 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:*

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

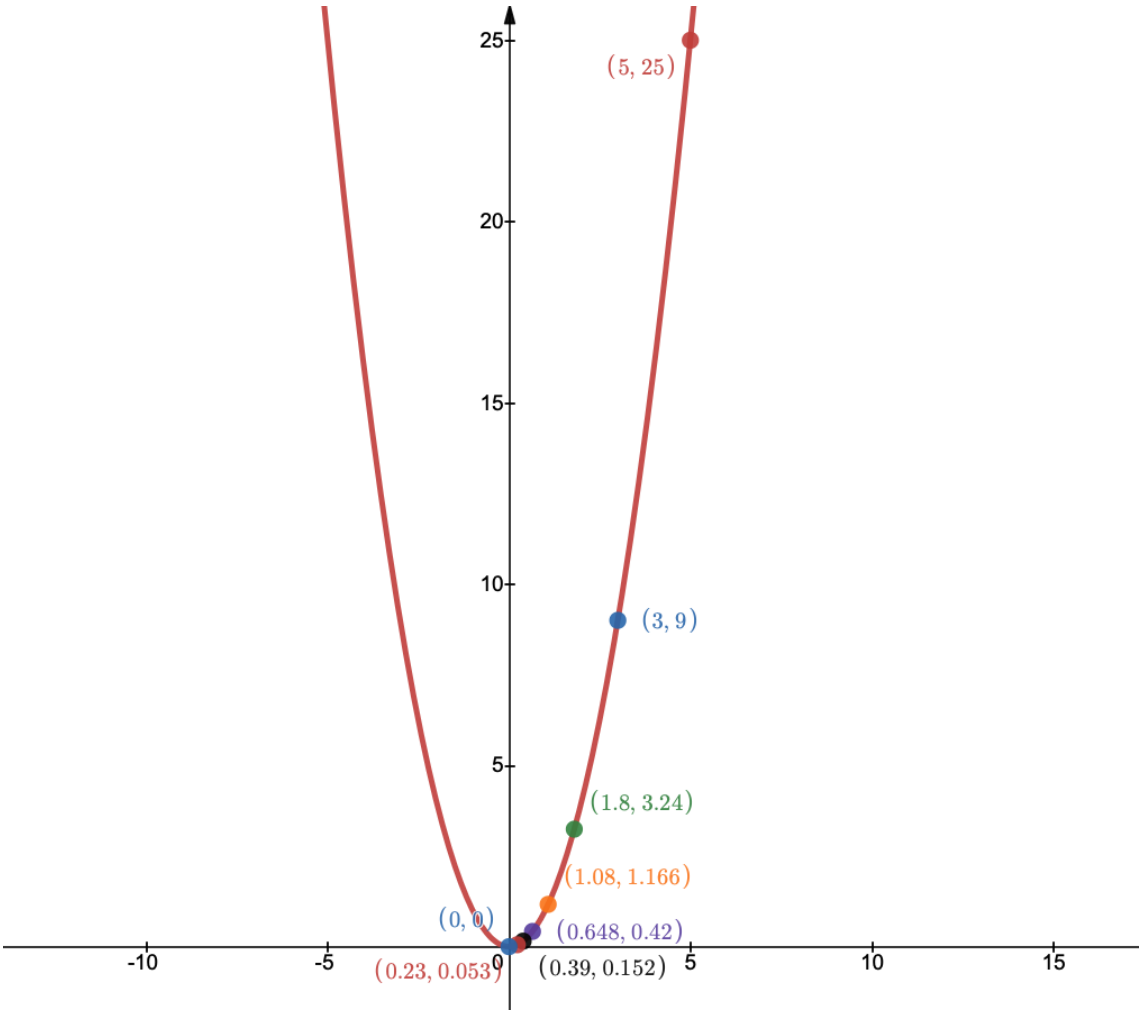
For small values of x, $\log(\cosh(x)) \approx \frac{1}{2}x^2$, which is similar to MSE. For larger values of x, $\log(\cosh(x)) \approx |x| - \log(2)$, which is similar to MAE. Log cosh approximates MSE and MAE and is similar to the Huber loss function.

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.

4. **Choosing Learning Rates**.

(a) 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:



Solution:

The one that gives the best convergence is $\alpha = 0.1$.

$\alpha = 10$	$\alpha = 1$	$\alpha = 0.1$	$\alpha = 0.01$
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

(b) During the course of training for a large number of epochs/iterations, what can be done to the value of the learning rate $alpha$ to enable better convergence?

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

Solution:

Learning rate α *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.*