**Report Week 1**

**Exercise 1**

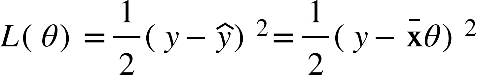
**Introduction to Linear Regression:**

Problem: Input X (single variable or multiple variable), How to predict output Y (single variable)?

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi mathvariant=\"bold\">&#x3B8;</mi><mo>=</mo><mo>[</mo><msub><mi mathvariant=\"bold\">&#x3B8;</mi><mn>0</mn></msub><mo>,</mo><msub><mi mathvariant=\"bold\">&#x3B8;</mi><mn>1</mn></msub><mo>,</mo><msub><mi mathvariant=\"bold\">&#x3B8;</mi><mn>2</mn></msub><mo>,</mo><msub><mi mathvariant=\"bold\">&#x3B8;</mi><mn>3</mn></msub><msup><mo>]</mo><mi>T</mi></msup></mstyle></math>"} : parameters need to be optimized.

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mover><mi mathvariant=\"bold\">x</mi><mo mathvariant=\"bold\">&#xAF;</mo></mover><mo>=</mo><mo>[</mo><mn>1</mn><mo>,</mo><msub><mi>x</mi><mn>1</mn></msub><mo>,</mo><msub><mi>x</mi><mn>2</mn></msub><mo>,</mo><msub><mi>x</mi><mn>3</mn></msub><mo>]</mo></mstyle></math>"}: expanded input

Our target: Optimizing the error between {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>y</mi><mo>&#xA0;</mo><mi>a</mi><mi>n</mi><mi>d</mi><mo>&#xA0;</mo><mover><mi>y</mi><mo>^</mo></mover></mstyle></math>"} with {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mover><mi>y</mi><mo>^</mo></mover><mo>&#xA0;</mo><mo>=</mo><mo>&#xA0;</mo><mover><mi mathvariant=\"bold\">x</mi><mo mathvariant=\"bold\">&#xAF;</mo></mover><mi>&#x3B8;</mi></mstyle></math>"}

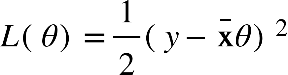
So, we need minimize the loss function here: {"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"/>"}

We have 2 solutions to find the best parameter:

Solution 1: Normal equation

Solution 2: Gradient Descent

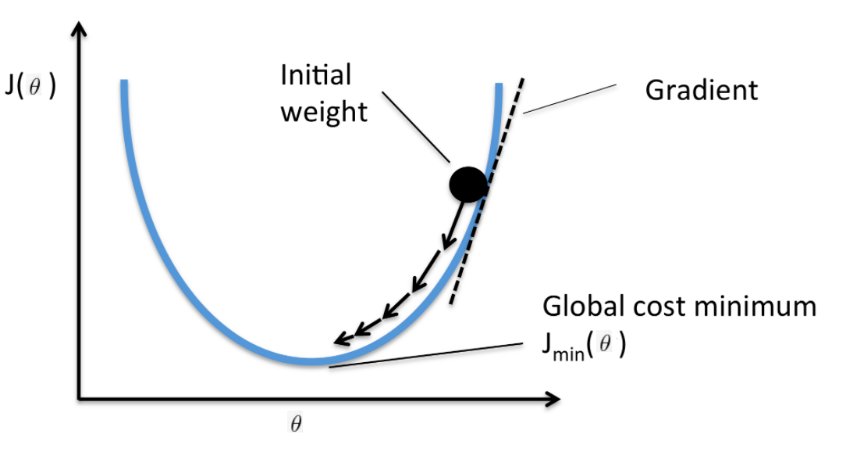
**Solution 1: Normal equation**

We find the best parameter by derivatizing the loss function  and finally get the equation: {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B8;</mi><mo>&#xA0;</mo><mo>=</mo><mo>&#xA0;</mo><mo>(</mo><msup><mi>X</mi><mi>T</mi></msup><mi>X</mi><msup><mo>)</mo><mrow><mo>&#x2212;</mo><mn>1</mn></mrow></msup><msup><mi>X</mi><mi>T</mi></msup><mi>y</mi></mstyle></math>"} .

But in some cases, the equation is more complicated. That is so hard to derivative and solve derivative equations. Gradient Descent is the best solution to pass these issues.

**Solution 2: Gradient Descent**

In this section, we use the Gradient Descent formula: {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B8;</mi><mo>=</mo><mi>&#x3B8;</mi><mo>-</mo><mi>&#x3B1;</mi><msub><mo>&#x2207;</mo><mi>&#x3B8;</mi></msub><mi>J</mi><mo>(</mo><mi>&#x3B8;</mi><mo>;</mo><mi mathvariant=\"bold\">x</mi><mo>;</mo><mi mathvariant=\"bold\">y</mi><mo>)</mo></mstyle></math>"}



Solving the Linear Regression Equation by using GD formula: {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B8;</mi><mo>=</mo><mi>&#x3B8;</mi><mo>-</mo><mfrac><mi>&#x3B1;</mi><mi>m</mi></mfrac><msup><mi>X</mi><mi>T</mi></msup><mo>(</mo><mi>X</mi><mi>&#x3B8;</mi><mo>-</mo><mover accent=\"true\"><mi>y</mi><mo>&#x2192;</mo></mover><mo>)</mo></mstyle></math>"}. Also called Batch Gradient Descent

Like the Batch Gradient Descent, we also get the mini-batch GD and Stochastic GD:

Mini-batch GD: {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B8;</mi><mo>=</mo><mi>&#x3B8;</mi><mo>-</mo><mi>&#x3B1;</mi><mo>.</mo><msub><mo>&#x2207;</mo><mi>&#x3B8;</mi></msub><mi>J</mi><mo>(</mo><mi>&#x3B8;</mi><mo>;</mo><msub><mi mathvariant=\"bold\">x</mi><mrow><mi>i</mi><mo>:</mo><mi>i</mi><mo>+</mo><mi>n</mi></mrow></msub><mo>;</mo><msub><mi mathvariant=\"bold\">y</mi><mrow><mi>i</mi><mo>:</mo><mi>i</mi><mo>+</mo><mi>n</mi></mrow></msub><mo>)</mo></mstyle></math>"}

Stochastic GD: {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B8;</mi><mo>=</mo><mi>&#x3B8;</mi><mo>-</mo><mi>&#x3B1;</mi><mo>.</mo><msub><mo>&#x2207;</mo><mi>&#x3B8;</mi></msub><mi>J</mi><mo>(</mo><mi>&#x3B8;</mi><mo>;</mo><msub><mi mathvariant=\"bold\">x</mi><mrow><mi>i</mi><mo>:</mo></mrow></msub><mo>;</mo><msub><mi mathvariant=\"bold\">y</mi><mi>i</mi></msub><mo>)</mo></mstyle></math>"}

Comparison between BGD, Mini-batch GD and SGD:

|  |  |  |
| --- | --- | --- |
| Batch GD | Mini-batch GD | Stochastic GD |
| Computing size in each step is largest.  Fast executive time  The convergence is slower | Computing size in each step is large.  Fast executive time  The convergence is slower | Computing size in each step is small.  Slow executive time  The convergence is faster. |

**Exercise 2**

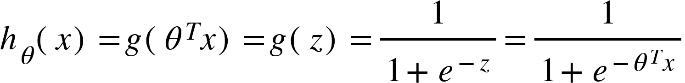
**Introduction to Sigmoid function and Logistic Regression:**

In Logistic Regression, we need to find the best linear line to separate 2 classes:

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>f</mi><mo>(</mo><mi mathvariant=\"bold\">x</mi><mo>)</mo><mo>=</mo><msup><mi mathvariant=\"bold\">&#x3B8;</mi><mi>T</mi></msup><mi mathvariant=\"bold\">x</mi></mstyle></math>"}

We need to find a function that can classify data into 2 classes. So, we realized that Sigmoid function is the best option. Sigmoid function is limited between 0 and 1 that makes the classification easier. Sigmoid function is used to update weight {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B8;</mi></mstyle></math>"}.

Sigmoid function: The sigmoid function, or logistic function, is a function that asymptotes at 0 and 1.



A graph of a function

Description automatically generated

**Loss function:**

We are going to use the sigmoid function to predict how likely it is that a given data point is in category 0. Our hypothesis function: {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msub><mi>h</mi><mi>&#x3B8;</mi></msub><mo>(</mo><mi>x</mi><mo>)</mo><mo>=</mo><mi>P</mi><mo>(</mo><mi>y</mi><mo>=</mo><mn>0</mn><mo>|</mo><mi>x</mi><mo>;</mo><mi>&#x3B8;</mi><mo>)</mo></mstyle></math>"}

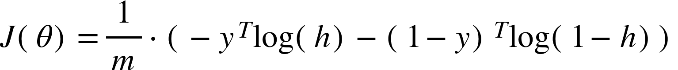
In this case, there are only 2 categories: {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>P</mi><mo>(</mo><mi>y</mi><mo>=</mo><mn>0</mn><mo>|</mo><mi>x</mi><mo>;</mo><mi>&#x3B8;</mi><mo>)</mo><mo>+</mo><mi>P</mi><mo>(</mo><mi>y</mi><mo>=</mo><mn>1</mn><mo>|</mo><mi>x</mi><mo>;</mo><mi>&#x3B8;</mi><mo>)</mo><mo>=</mo><mn>1</mn></mstyle></math>"}

We use Bernoulli Distribution formula to build the loss function:{"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"/>"}{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>P</mi><mo>(</mo><msub><mi>y</mi><mi>i</mi></msub><mo>|</mo><msub><mi mathvariant=\"bold\">x</mi><mi mathvariant=\"bold\">i</mi></msub><mo>;</mo><mi>&#x3B8;</mi><mo>)</mo><mo>&#xA0;</mo><mo>=</mo><mo>&#xA0;</mo><msubsup><mi>h</mi><mi>i</mi><msub><mi>y</mi><mi>i</mi></msub></msubsup><mo>(</mo><mn>1</mn><mo>-</mo><msub><mi>h</mi><mi>i</mi></msub><msup><mo>)</mo><mrow><mn>1</mn><mo>-</mo><msub><mi>y</mi><mi>i</mi></msub></mrow></msup></mstyle></math>"}

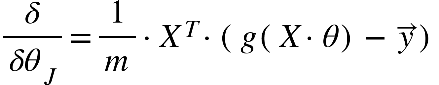
Our target that maximizes the {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>P</mi><mo>(</mo><mi mathvariant=\"bold\">y</mi><mo>|</mo><mi mathvariant=\"bold\">X</mi><mo>;</mo><mi>&#x3B8;</mi><mo>)</mo></mstyle></math>"}.We also minimize this function:

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>J</mi><mo>(</mo><mi>&#x3B8;</mi><mo>)</mo><mo>=</mo><mo>-</mo><mi>log</mi><mi>P</mi><mo>(</mo><msub><mi mathvariant=\"bold\">y</mi><mi mathvariant=\"bold\">i</mi></msub><mo>|</mo><msub><mi mathvariant=\"bold\">x</mi><mi mathvariant=\"bold\">i</mi></msub><mo>;</mo><mi mathvariant=\"bold\">&#x3B8;</mi><mo>)</mo><mo>=</mo><mo>-</mo><mo>(</mo><msub><mi>y</mi><mi>i</mi></msub><mi>log</mi><msub><mi>h</mi><mi>i</mi></msub><mo>+</mo><mo>(</mo><mn>1</mn><mo>-</mo><msub><mi>y</mi><mi>i</mi></msub><mo>)</mo><mi>log</mi><mo>(</mo><mn>1</mn><mo>-</mo><msub><mi>h</mi><mi>i</mi></msub><mo>)</mo><mo>)</mo></mstyle></math>"}

Finally, {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>J</mi><mo>(</mo><mi mathvariant=\"bold\">&#x3B8;</mi><mo>)</mo></mstyle></math>"}is the function that we need optimize. Beside that, we also get a vectorized version of loss function:



To update the parameters {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B8;</mi></mstyle></math>"} , we need do partial derivative the loss function:

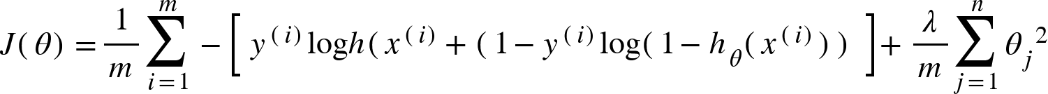


**Regularization:**

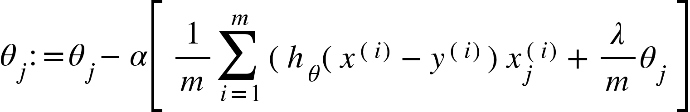
In realistic problems, we sometimes encounter issues related to overfitting or underfitting. In this section, we need to use Regularization method to relieve these issues.

This method proposes the penalty to {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B8;</mi></mstyle></math>"} . Like those noises to make the model more complex, improve the model to predict real problem better.

In Logistic Regression, we can add regularization  to loss function like that:



The next step, we derivative this loss function for gradient descent:



**Linear Separable Issue:**

A diagram of a line with dots and a blue line

Description automatically generatedA diagram of a circle with dots

Description automatically generated

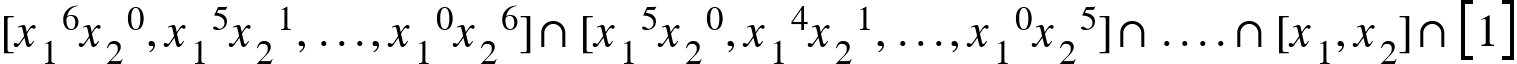
Linear separable Not linear separable

**Feature Mapping:**

The Feature Mapping method is a technique used to convert the original feature space into a new feature space, serving the purpose of solving non-linear problems.

In this section, we use one of Feature Mapping methods: Polynomial features.

Polynomial feature: Transforming the input variables into polynomials or combinations of them. For example, transforming the data from {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mo>[</mo><msub><mi>x</mi><mn>1</mn></msub><mo>,</mo><mo>&#xA0;</mo><msub><mi>x</mi><mn>2</mn></msub><mo>]</mo><mo>&#xA0;</mo></mstyle></math>"}to {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mo>[</mo><mn>1</mn><mo>,</mo><msub><mi>x</mi><mn>1</mn></msub><mo>,</mo><msub><mi>x</mi><mn>2</mn></msub><mo>,</mo><msup><msub><mi>x</mi><mn>1</mn></msub><mn>2</mn></msup><mo>,</mo><msup><msub><mi>x</mi><mn>2</mn></msub><mn>2</mn></msup><mo>,</mo><msub><mi>x</mi><mn>1</mn></msub><msub><mi>x</mi><mn>2</mn></msub><mo>]</mo><mo>.</mo></mstyle></math>"}

In exercise 2, We use transformations {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mo>[</mo><msub><mi>x</mi><mn>1</mn></msub><mo>,</mo><mo>&#xA0;</mo><msub><mi>x</mi><mn>2</mn></msub><mo>]</mo><mo>&#xA0;</mo></mstyle></math>"}into the combinations of {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msub><mi>x</mi><mrow><mn>1</mn><mo>&#xA0;</mo><mo>&#xA0;</mo></mrow></msub><mi>a</mi><mi>n</mi><mi>d</mi><mo>&#xA0;</mo><msub><mi>x</mi><mn>2</mn></msub></mstyle></math>"}:

**Overfitting and Underfitting:**

Overfitting: means the model probably only works on the training set and not well in the real world

Underfitting: means the model is not well trained.

Regularization can ease overfitting, but it can lead to underfitting.

**Exercise 3**

**One-vs-all Classification:**

One-vs-all Classification: Classification problem with multiple classes is decomposed into multiple binary classification sub-problems.

For example, in this exercise we get 10 classes [Number 0->9] to classify. In this case we can simply consider number 3 positive (1) and others are negative (0) and so on. This method works like that. Simple Summary is Using Logistic Regression (Binary Classification) for Multiclass classification task.

A visual representation of One-vs-All classification can be seen below:

A diagram of different colored squares

Description automatically generated

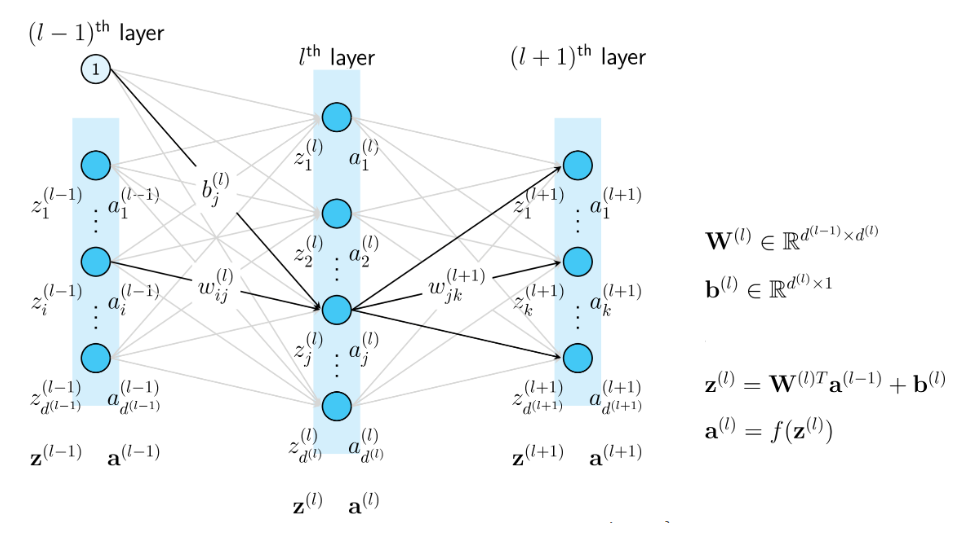
Like previous parameter {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B8;</mi></mstyle></math>"} in logistic regression. For One-vs-all Classification, we store each parameter {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B8;</mi></mstyle></math>"} of binary-classification task in our matrix {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x398;</mi><mo>&#x2208;</mo><msup><mi mathvariant=\"normal\">&#x211D;</mi><mrow><mi>K</mi><mo>&#xD7;</mo><mo>(</mo><mi>N</mi><mo>+</mo><mn>1</mn><mo>)</mo></mrow></msup></mstyle></math>"} with K: the number of categories and N: the number of features of input X.

If we get a new input {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>X</mi><mo>&#x2208;</mo><msup><mi mathvariant=\"normal\">&#x211D;</mi><mrow><mn>1</mn><mo>&#xD7;</mo><mi>N</mi><mo>+</mo><mn>1</mn></mrow></msup></mstyle></math>"} , we will the output {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>y</mi><mo>=</mo><mi>X</mi><mo>.</mo><msup><mi>&#x398;</mi><mi>T</mi></msup><mo>&#x2208;</mo><msup><mi mathvariant=\"normal\">&#x211D;</mi><mrow><mn>1</mn><mo>&#xD7;</mo><mi>K</mi></mrow></msup></mstyle></math>"}. We will get each element in this output limited between 0 and 1. It’s like One-hot Encoding. There are 10 categories (10 number

0->9), for example we can get a vector output {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>y</mi><mo>=</mo><mfenced open=\"[\" close=\"]\"><mrow><mn>0</mn><mo>.</mo><mn>1</mn><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mn>0</mn><mo>.</mo><mn>8</mn><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mn>0</mn><mo>.</mo><mn>1</mn><mo>&#xA0;</mo></mrow></mfenced></mstyle></math>"}, so that means input X is probably number 2.

**Multi-class Classification with neural networks:**

Forward propagation of neural networks:



is the index of layer, {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>d</mi></mstyle></math>"} is the number of units in 1 layer.



{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>X</mi><mo>&#xA0;</mo><mo>=</mo><mo>&#xA0;</mo><msup><mi>a</mi><mfenced><mn>0</mn></mfenced></msup></mstyle></math>"} : the first layer values.

: the previous layer; {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>a</mi><mfenced><mi>l</mi></mfenced></msup></mstyle></math>"}: the following layer; {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>b</mi><mfenced><mi>l</mi></mfenced></msup></mstyle></math>"}: the bias of layer {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>l</mi></mstyle></math>"}; {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>W</mi><mfenced><mi>l</mi></mfenced></msup></mstyle></math>"}: the parameter of layer {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>l</mi></mstyle></math>"}.



Note: In this exercise, we use pretrained weights {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B8;</mi></mstyle></math>"} for 2 layers.

**Exercise 4**

**One-hot Encoding:** is a technique used in machine learning and data processing to represent categorical variables as binary vectors.

For example, let's say we have a categorical variable "color" with three categories: red, green, and blue. After one-hot encoding, each category will be represented by a binary vector:

Red: [1, 0, 0]

Green: [0, 1, 0]

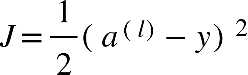
Blue: [0, 0, 1]

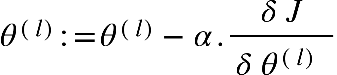
In this exercise, we can apply one-hot encoding to 10 classes from number 0 to number 9 by transforming them into binary vectors: For example, we vectorize number 1 to {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mfenced open=\"[\" close=\"]\"><mrow><mn>1</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo></mrow></mfenced></mstyle></math>"} , 2 to {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mfenced open=\"[\" close=\"]\"><mrow><mn>0</mn><mo>&#xA0;</mo><mn>1</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn><mo>&#xA0;</mo><mn>0</mn></mrow></mfenced></mstyle></math>"} and so on.

**Backpropagation:**

In the previous section, we learn Forward Propagation to calculate output, but it is not enough. We absolutely need do Backpropagation to update the weights on each layer and loss function.

We suppose that loss function is:

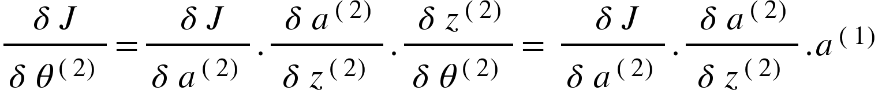


We need update {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>&#x3B8;</mi><mfenced><mi>l</mi></mfenced></msup></mstyle></math>"} by do partial derivative {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>J</mi></mstyle></math>"} to {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>&#x3B8;</mi><mfenced><mi>l</mi></mfenced></msup></mstyle></math>"}: 

We formally built the Backpropagation for this exercise with 2 layers here:

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>a</mi><mfenced><mrow><mi>l</mi><mo>+</mo><mn>1</mn></mrow></mfenced></msup><mo>=</mo><mi>f</mi><mfenced><msup><mi>z</mi><mfenced><mrow><mi>l</mi><mo>+</mo><mn>1</mn></mrow></mfenced></msup></mfenced></mstyle></math>"}

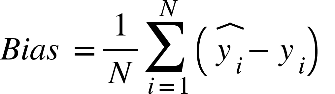
{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>z</mi><mfenced><mrow><mi>l</mi><mo>+</mo><mn>1</mn></mrow></mfenced></msup><mo>=</mo><msup><mi>&#x3B8;</mi><mi>T</mi></msup><msup><mi>a</mi><mfenced><mi>l</mi></mfenced></msup></mstyle></math>"}

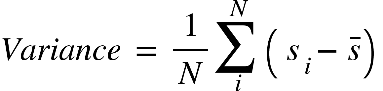


{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mfrac><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><mi>J</mi></mrow><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>&#x3B8;</mi><mfenced><mn>1</mn></mfenced></msup></mrow></mfrac><mo>=</mo><mfrac><mrow><mo>&#xA0;</mo><mi>&#x3B4;</mi><mo>&#xA0;</mo><mi>J</mi></mrow><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>a</mi><mfenced><mn>1</mn></mfenced></msup></mrow></mfrac><mo>.</mo><mfrac><mrow><mo>&#xA0;</mo><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>a</mi><mfenced><mn>1</mn></mfenced></msup></mrow><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>z</mi><mfenced><mn>1</mn></mfenced></msup></mrow></mfrac><mo>.</mo><mfrac><mrow><mo>&#xA0;</mo><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>z</mi><mfenced><mn>1</mn></mfenced></msup></mrow><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>&#x3B8;</mi><mfenced><mn>1</mn></mfenced></msup></mrow></mfrac><mo>=</mo><mo>&#xA0;</mo><mfrac><mrow><mo>&#xA0;</mo><mi>&#x3B4;</mi><mo>&#xA0;</mo><mi>J</mi></mrow><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>a</mi><mfenced><mn>2</mn></mfenced></msup></mrow></mfrac><mo>.</mo><mfrac><mrow><mo>&#xA0;</mo><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>a</mi><mfenced><mn>2</mn></mfenced></msup></mrow><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>z</mi><mfenced><mn>2</mn></mfenced></msup></mrow></mfrac><mo>.</mo><mfrac><mrow><mo>&#xA0;</mo><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>z</mi><mfenced><mn>2</mn></mfenced></msup></mrow><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>a</mi><mfenced><mn>1</mn></mfenced></msup></mrow></mfrac><mo>.</mo><mfrac><mrow><mo>&#xA0;</mo><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>a</mi><mfenced><mn>1</mn></mfenced></msup></mrow><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>z</mi><mfenced><mn>1</mn></mfenced></msup></mrow></mfrac><mo>.</mo><msup><mi>a</mi><mfenced><mn>0</mn></mfenced></msup><mo>=</mo><mfrac><mrow><mo>&#xA0;</mo><mi>&#x3B4;</mi><mo>&#xA0;</mo><mi>J</mi></mrow><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>a</mi><mfenced><mn>2</mn></mfenced></msup></mrow></mfrac><mo>.</mo><mfrac><mrow><mo>&#xA0;</mo><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>a</mi><mfenced><mn>2</mn></mfenced></msup></mrow><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>z</mi><mfenced><mn>2</mn></mfenced></msup></mrow></mfrac><mo>.</mo><mo>&#xA0;</mo><msup><mi>&#x3B8;</mi><mfenced><mn>2</mn></mfenced></msup><mo>.</mo><mfrac><mrow><mo>&#xA0;</mo><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>a</mi><mfenced><mn>1</mn></mfenced></msup></mrow><mrow><mi>&#x3B4;</mi><mo>&#xA0;</mo><msup><mi>z</mi><mfenced><mn>1</mn></mfenced></msup></mrow></mfrac><mo>.</mo><msup><mi>a</mi><mfenced><mn>0</mn></mfenced></msup></mstyle></math>"}

**Exercise 5**

**Bias and Variance:**



We suppose {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msub><mi>s</mi><mi>i</mi></msub><mo>&#xA0;</mo><mo>=</mo><mo>&#xA0;</mo><mover><msub><mi>y</mi><mi>i</mi></msub><mo>^</mo></mover><mo>-</mo><msub><mi>y</mi><mi>i</mi></msub></mstyle></math>"} , so we get {"mathml":"<math style=\"font-family:stix;font-size:16px;\"/>"}

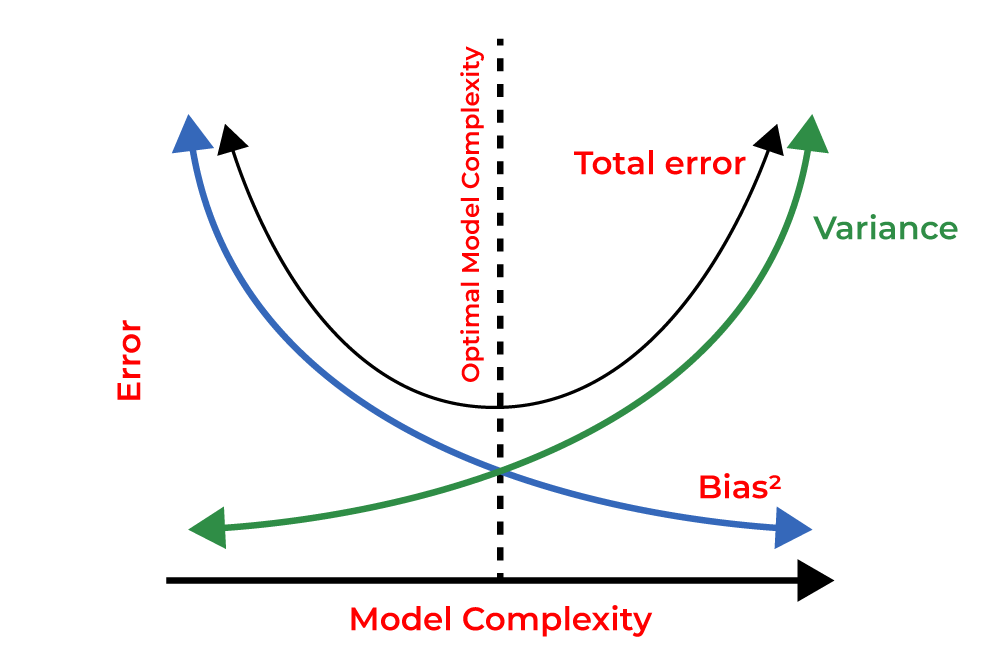
Visual Representation of Bias and Variance: The red region is supposed to be real value and the symbol of X is supposed to be predicted value. Your target is finding the best predicted value with low bias and variance.

A diagram of different types of red circles

Description automatically generated

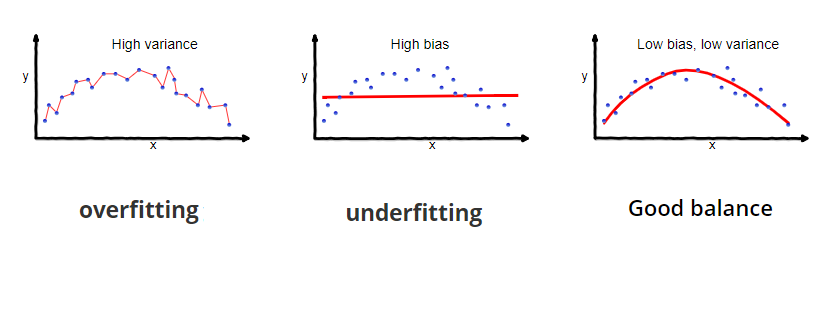
There is not any formal role that the inverse relationship between Bias and Variance. In fact, when the model has high bias that means it is so simple to learn the complexity of dataset. In the contrast, when the model has high variance and it is so much complex, it can be sensitive to new dataset.

The visual relationship between Bias and Variance:



Our target is finding the balance between Bias and Variance.

The example of the model with high bias, high variance and the balanced of bias and variance:



The model with high bias means leading to underfitting and the model with high variance means leading to overfitting.

**Polynomial Regression:**

The problem with our linear model was that it was too simple for the data and resulted in underfitting (high bias). In this part of the exercise, you will address this problem by adding more features.

A graph with red and blue lines

Description automatically generated

In this exercise, we realized that the linear line cannot be fit to the data. So that we propose an idea to transform feature into higher degree {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>p</mi></mstyle></math>"}. That means we transform input {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>X</mi><mo>&#x2208;</mo><msup><mi mathvariant=\"normal\">&#x211D;</mi><mrow><mi>m</mi><mo>&#xD7;</mo><mn>1</mn></mrow></msup><mo>&#xA0;</mo></mstyle></math>"}into {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>X</mi><mo>&#x2208;</mo><msup><mi mathvariant=\"normal\">&#x211D;</mi><mrow><mi>m</mi><mo>&#xD7;</mo><mi>p</mi></mrow></msup><mo>&#xA0;</mo></mstyle></math>"}

In this exercise, we get the transformation here:

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mfenced open=\"[\" close=\"]\"><mrow><mn>1</mn><mo>,</mo><mi>X</mi></mrow></mfenced><mo>&#xA0;</mo><mo>&#x2192;</mo><mfenced open=\"[\" close=\"]\"><mrow><mn>1</mn><mo>,</mo><mi>X</mi><mo>,</mo><msup><mi>X</mi><mn>2</mn></msup><mo>,</mo><msup><mi>X</mi><mn>3</mn></msup><mo>,</mo><msup><mi>X</mi><mn>4</mn></msup><mo>,</mo><msup><mi>X</mi><mn>5</mn></msup><mo>,</mo><msup><mi>X</mi><mn>6</mn></msup><mo>,</mo><msup><mi>X</mi><mn>7</mn></msup><mo>,</mo><msup><mi>X</mi><mn>8</mn></msup></mrow></mfenced><mo>&#xA0;</mo><mspace linebreak=\"newline\"/></mstyle></math>"} with new weights {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mfenced open=\"[\" close=\"]\"><mrow><msub><mi>&#x3B8;</mi><mn>0</mn></msub><mo>,</mo><msub><mi>&#x3B8;</mi><mn>1</mn></msub><mo>,</mo><msub><mi>&#x3B8;</mi><mn>2</mn></msub><mo>,</mo><msub><mi>&#x3B8;</mi><mn>3</mn></msub><mo>,</mo><msub><mi>&#x3B8;</mi><mn>4</mn></msub><mo>,</mo><msub><mi>&#x3B8;</mi><mn>5</mn></msub><mo>,</mo><msub><mi>&#x3B8;</mi><mn>6</mn></msub><mo>,</mo><msub><mi>&#x3B8;</mi><mn>7</mn></msub><mo>,</mo><msub><mi>&#x3B8;</mi><mn>8</mn></msub></mrow></mfenced></mstyle></math>"}

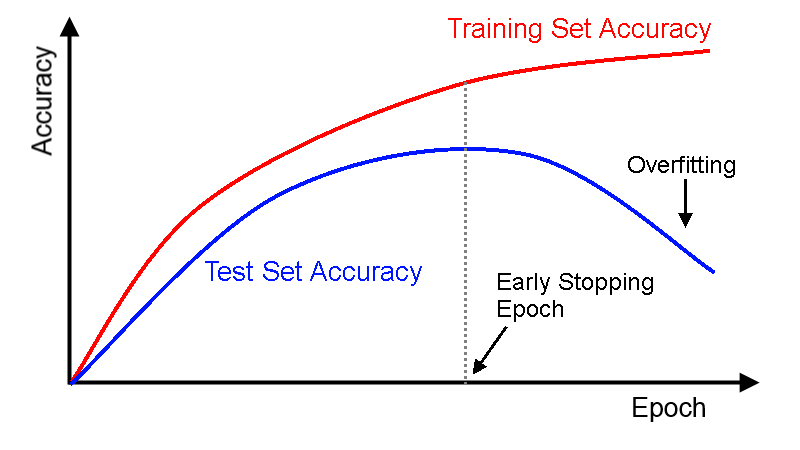
Finally, we get the better prediction:

A graph of a graph

Description automatically generated

**Learning curve:**

Learning curve refers to a graphical representation of model's performance



A learning curve plot on training and test set can show the visual representation of Overfitting.

One of the methods to prevent the overfitting is Early stopping method. It involves monitoring the performance of a model during training and stopping the training process before it reaches full convergence.