Foundational Mathematical Concepts in Deep Learning

Math and Deep Learning with Intuitions

Sandeep Krishnamurthy - @skm4ml

AWS Deep Learning

Disclaimer

- Mathematical rigor in notations, proofs etc… is not the focus of my talk.
- This is not the complete guide to math behind Deep Learning. However, I try to scratch the surface and provide the foundations sufficient to get started with ideas behind Deep Learning.
- Topics are too simplified. Do not be in a state of know all by end of this talk ;-) There is a lot beyond this talk.

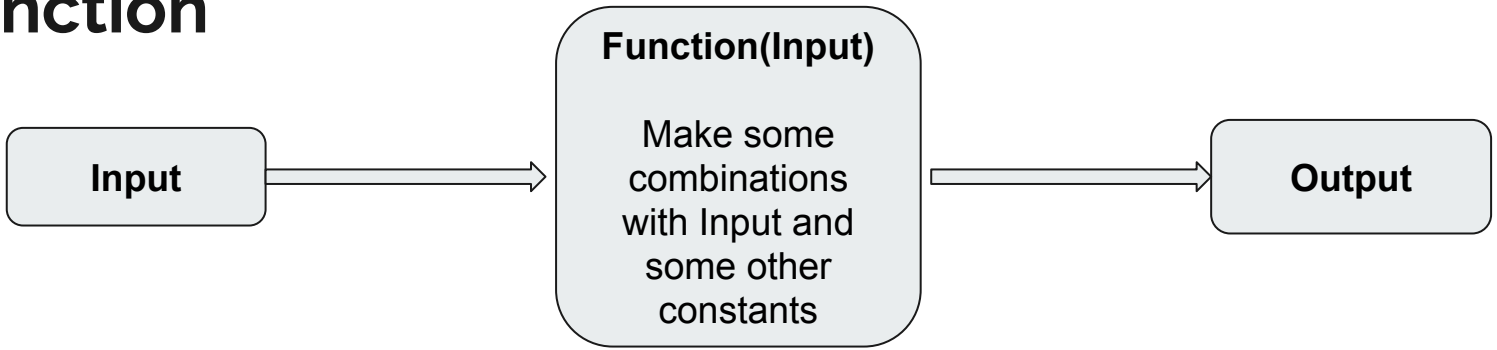
Principle behind the talk

Numerical/Mathematical representation is required for computers.

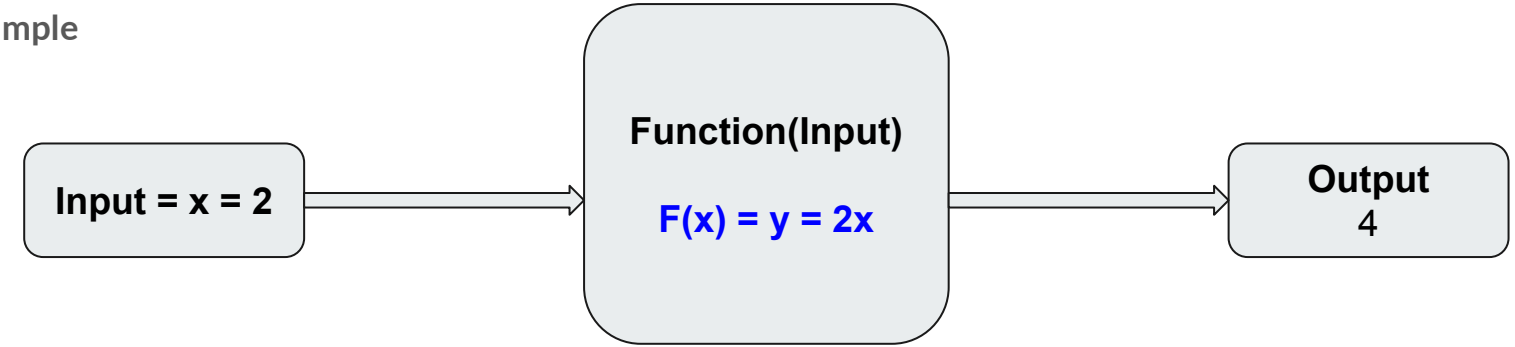
Visual representation is required for humans.

Function

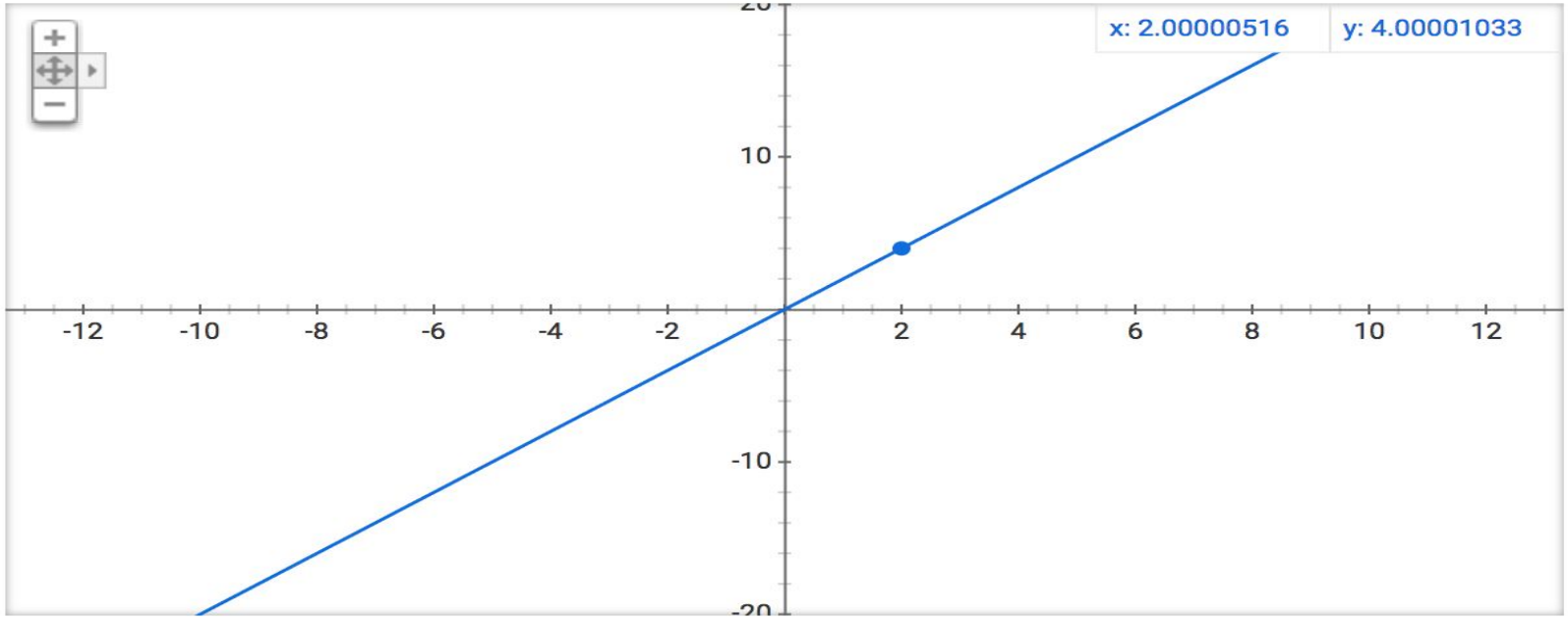
Function



Example



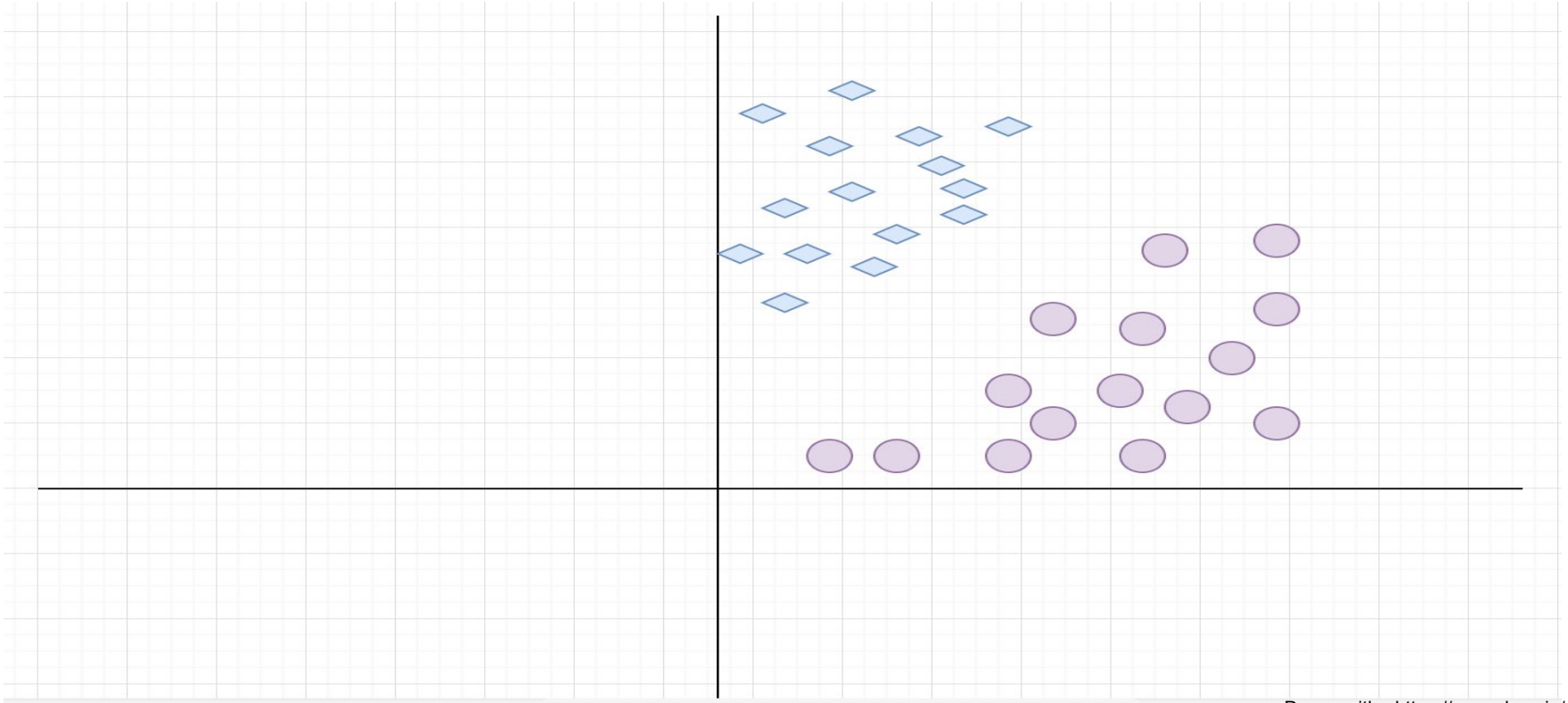
Visual Representation: f(x) = y = 2x



Ref - Google graphs

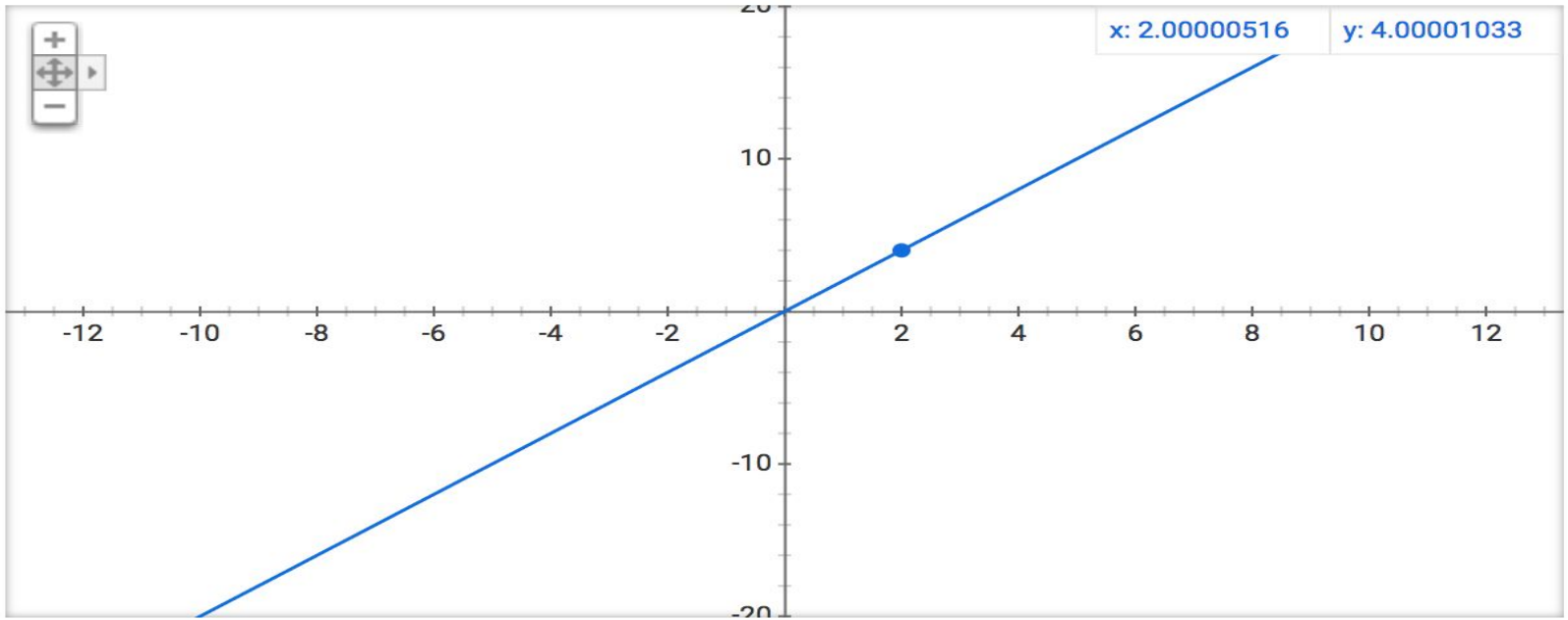
Model

Separate “circles” from “diamonds” ?

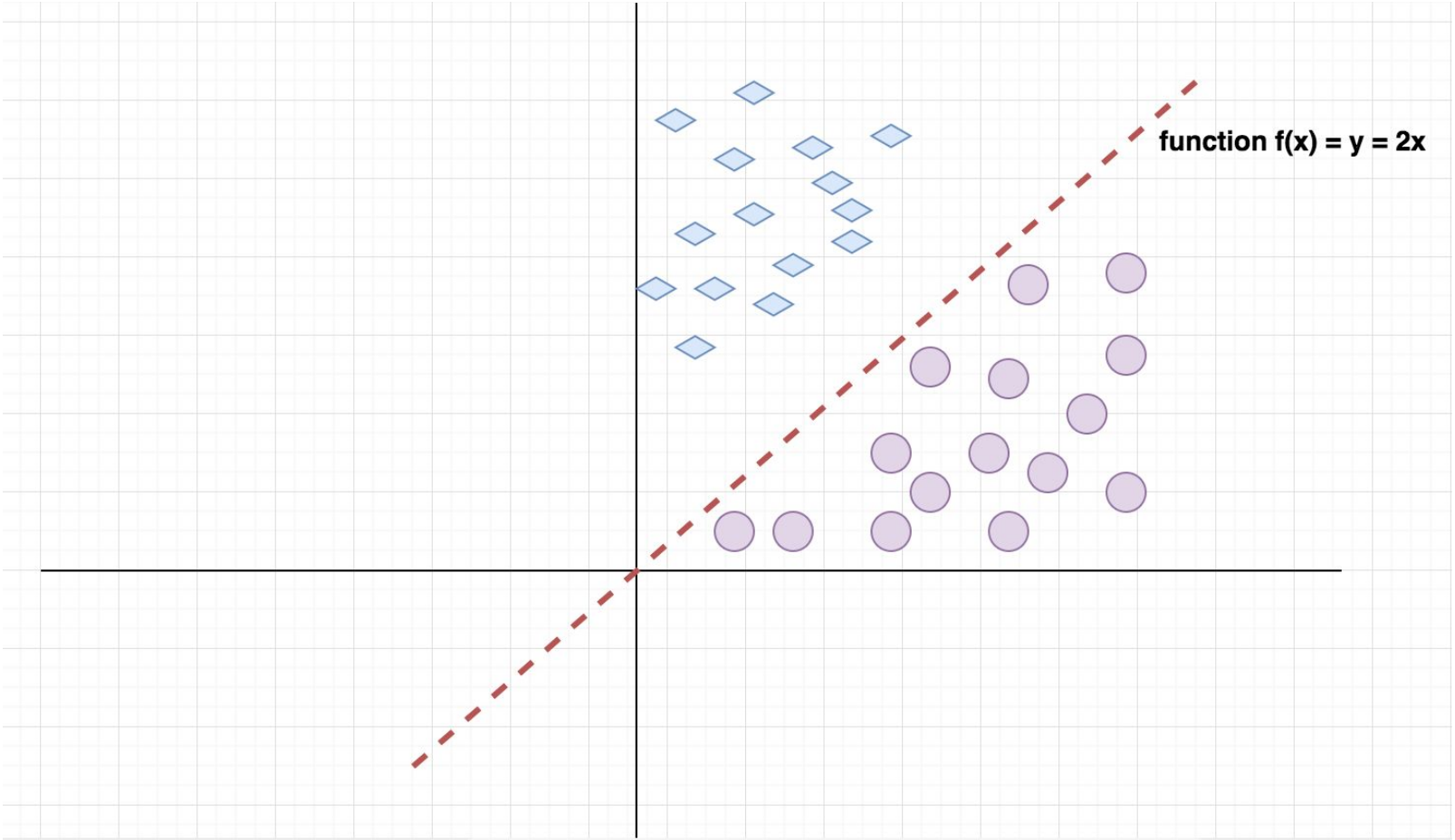


Drawn with - https://www.draw.io/

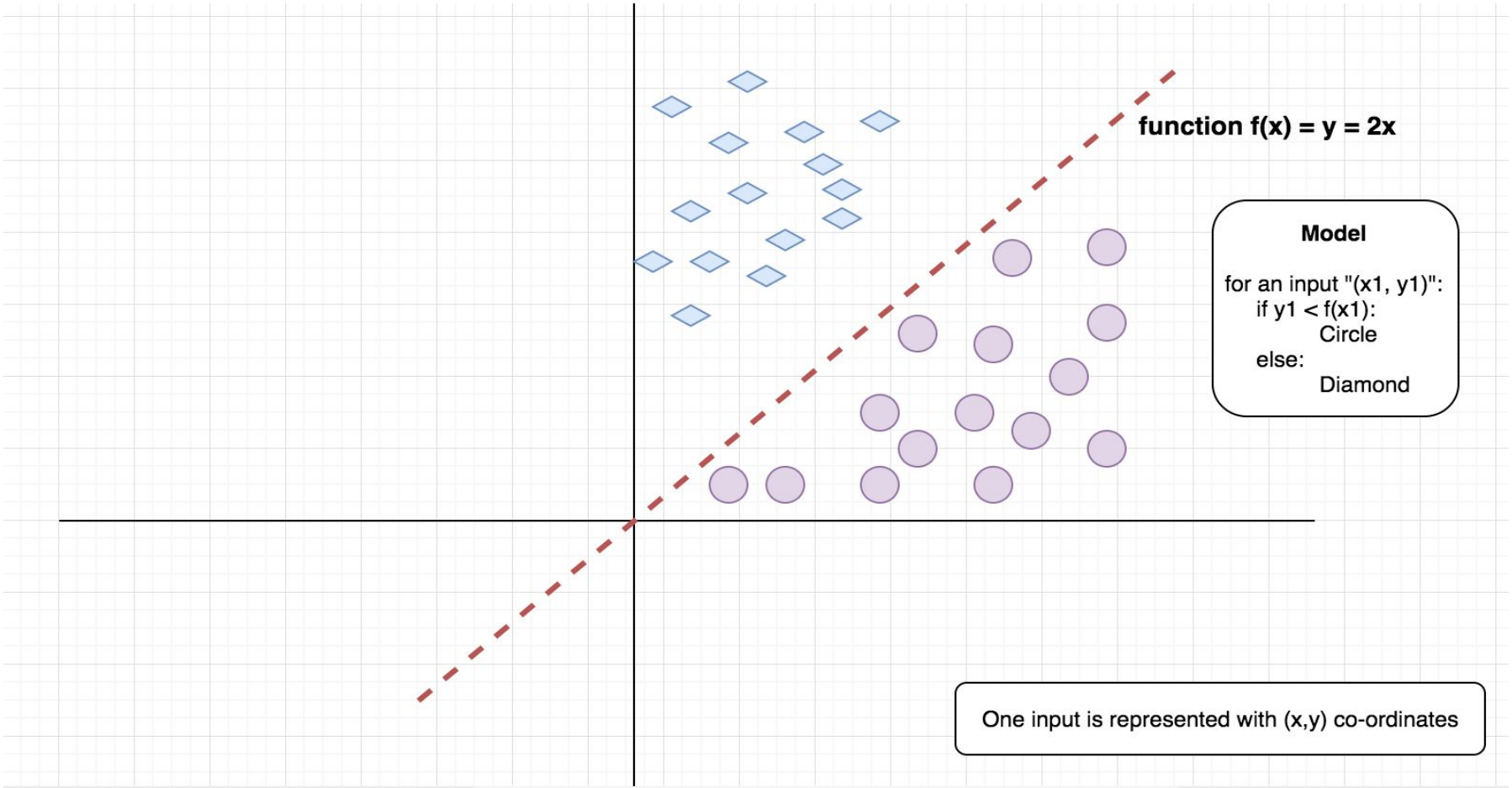
Probably with function f(x) = y = 2x ?



Ref - Google graphs



Drawn with - https://www.draw.io/



Drawn with - https://www.draw.io/

Model

- The function **f(x) =y = 2x** is the Model.
- Note: “**2**” in the function - call it as the “weights”.
- Model is one pack of “a function and all parameters in the function(constants)” representing a domain (train data).

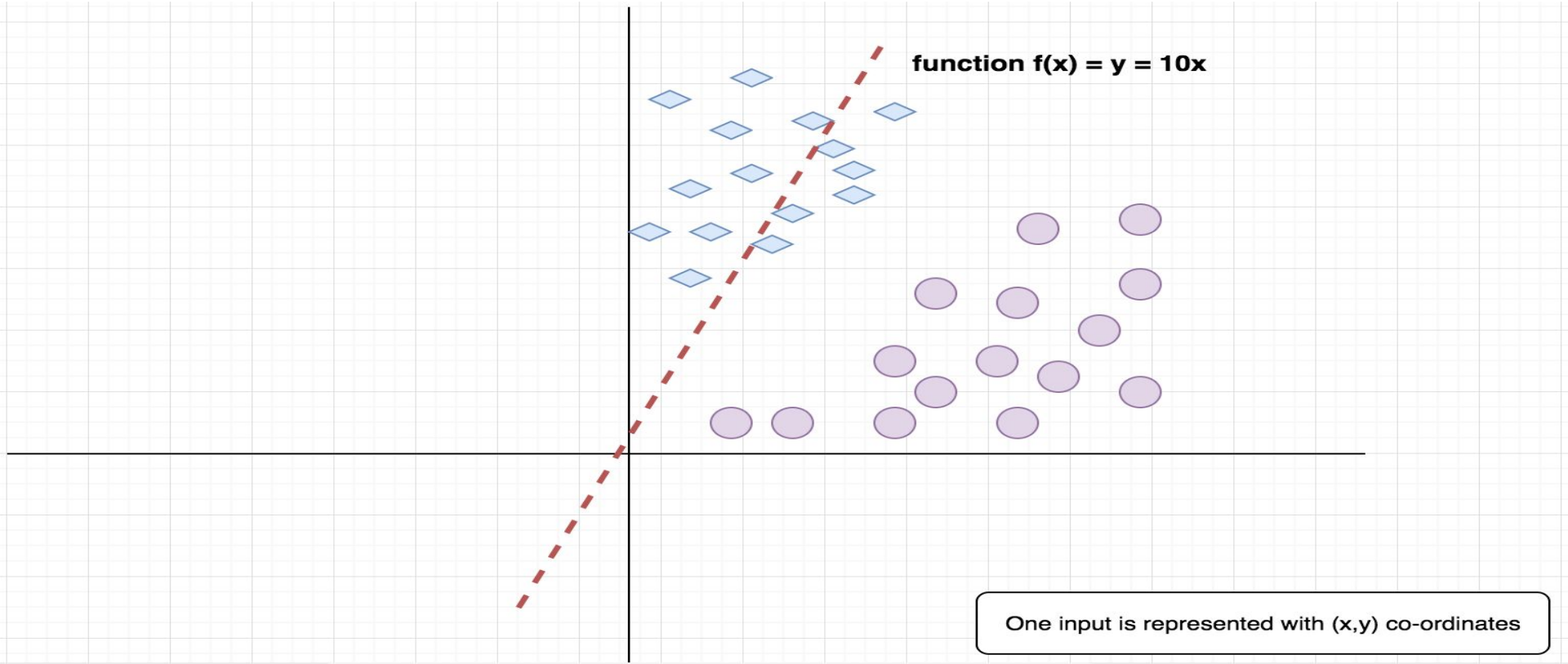
But, wait…

How did we choose “2” in $y = 2x$? Why not 3?

Why the function (model) is $y = mx$?
Why not $y = x^2$?

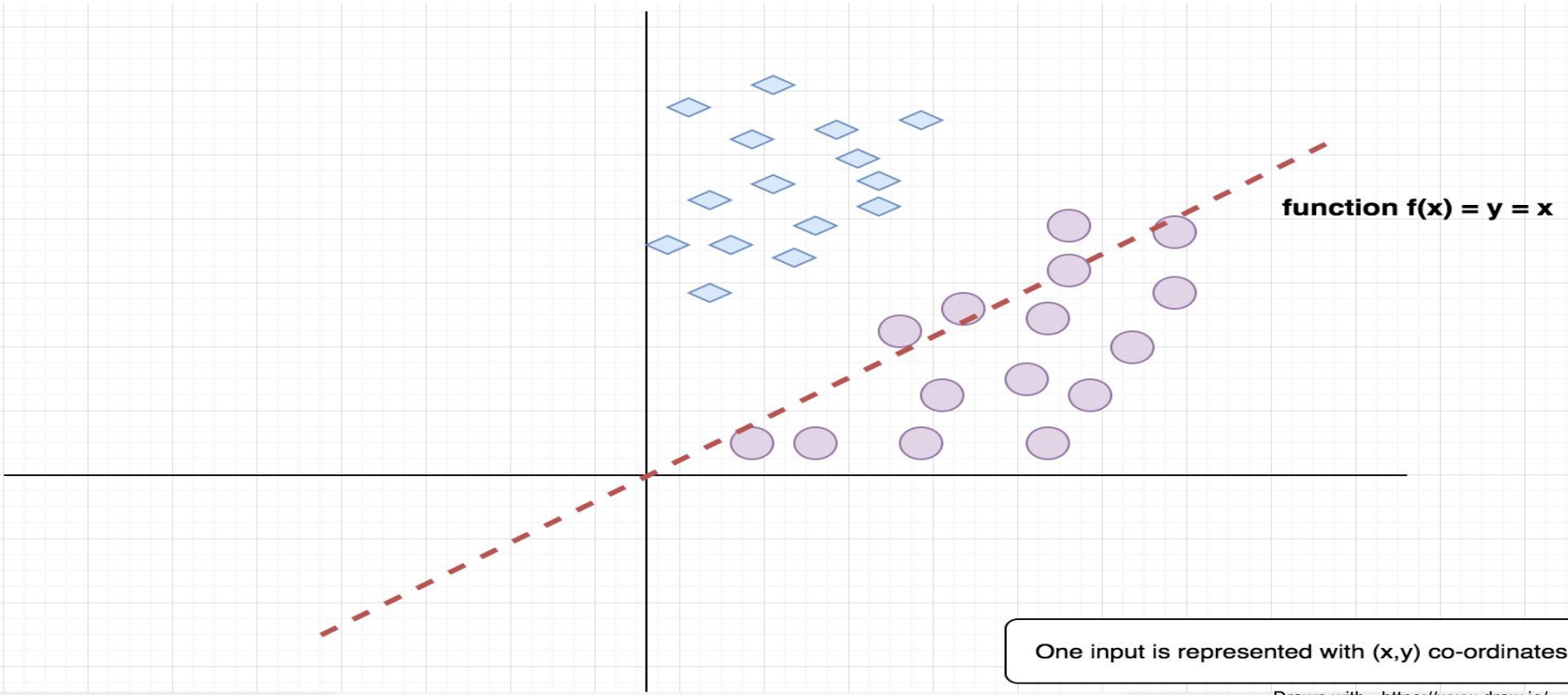
Model Training (Learning)

y = 10x ?? Not so perfect..



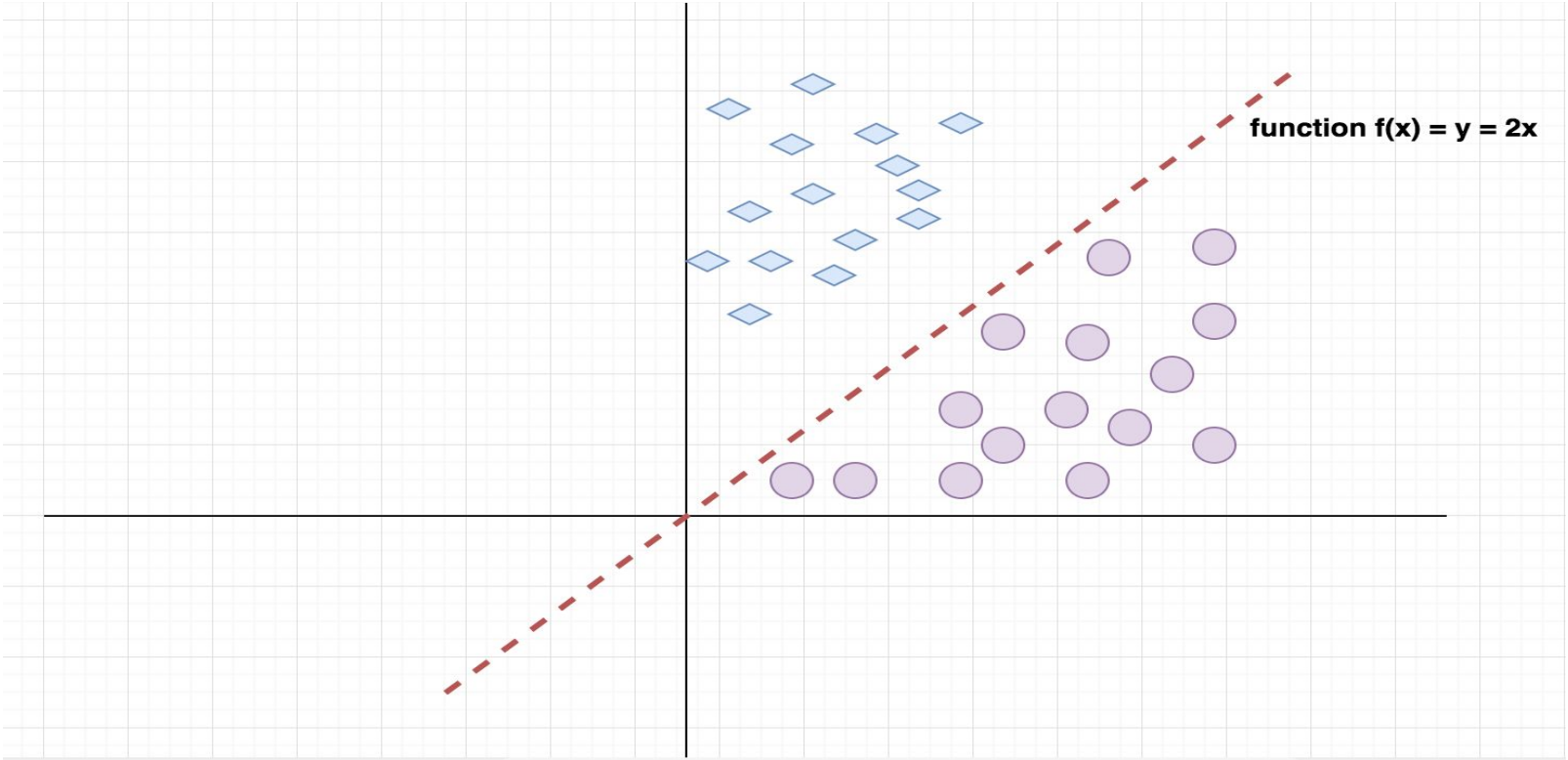
Drawn with - https://www.draw.io/

Hmm... may be “y = x”?? Again, not so perfect..



Drawn with - https://www.draw.io/

Ok, may be “y = 2x”?? Ahh, looks good..



Drawn with - https://www.draw.io/

This process of trial and error ending with a function that best represents our problem is called - “Learning” or “Model Training”

Training Data
Test Data

Training Data

Input	Label
(2,1)	Circle
(2,4)	Diamond
(4,10)	Diamond

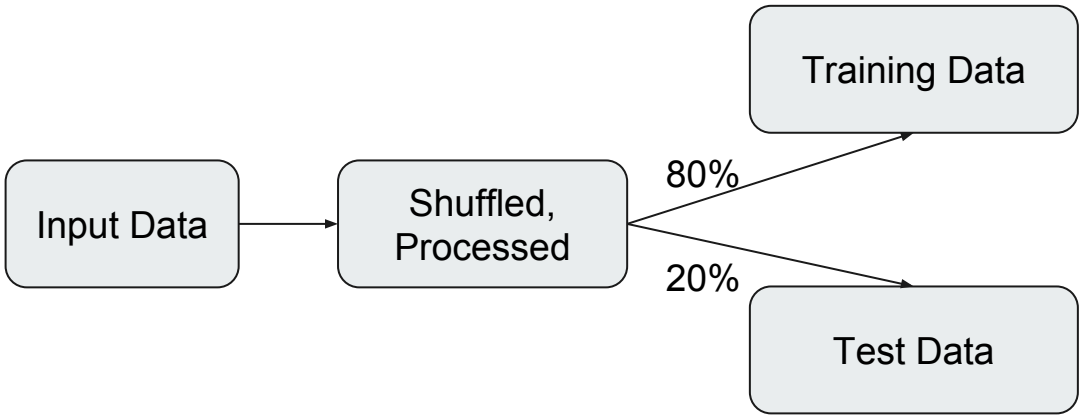
And more such example input and expected shape name...

Test Data

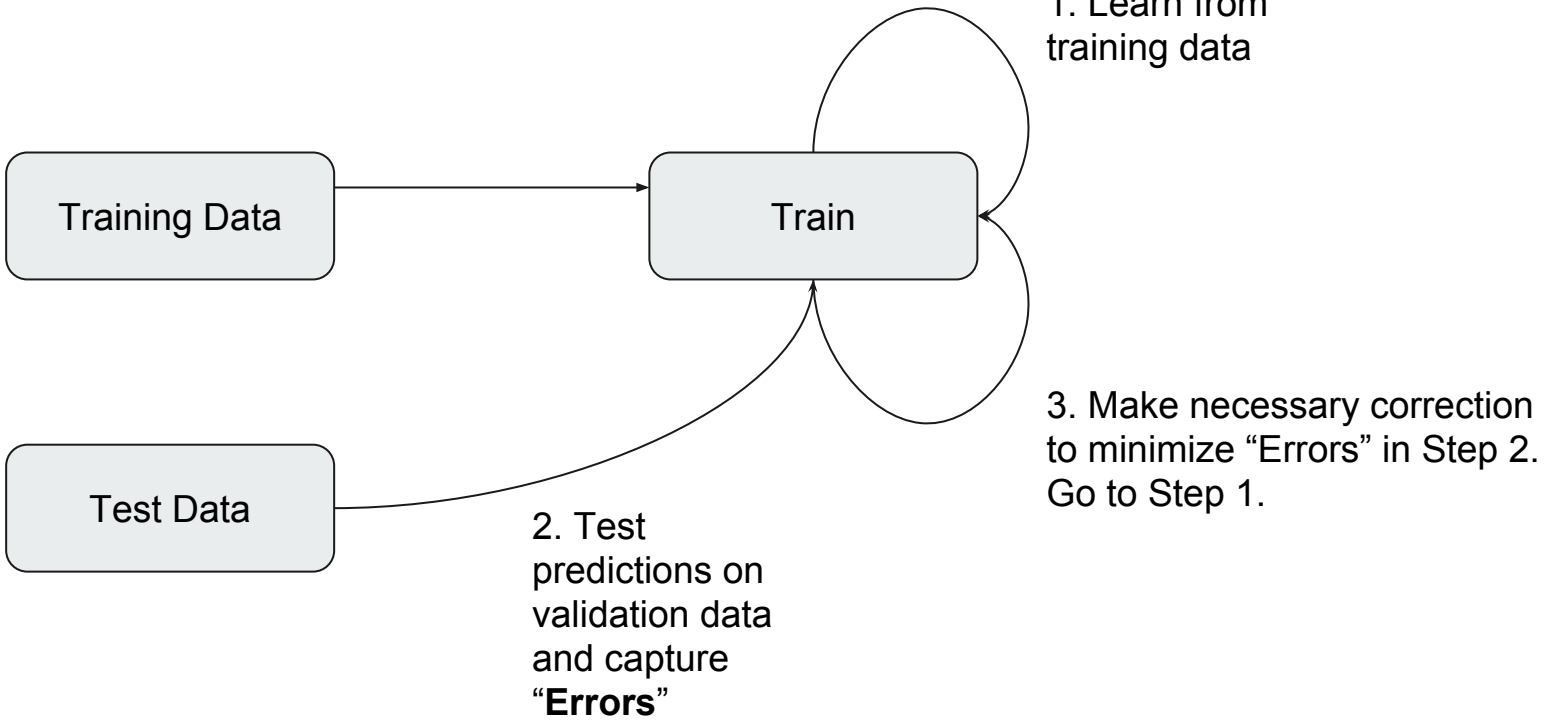
Input	Label
(2,1)	Circle
(2,4)	Diamond
(4,10)	Diamond

This is similar to training data, but, not used in training.

General Idea - Data

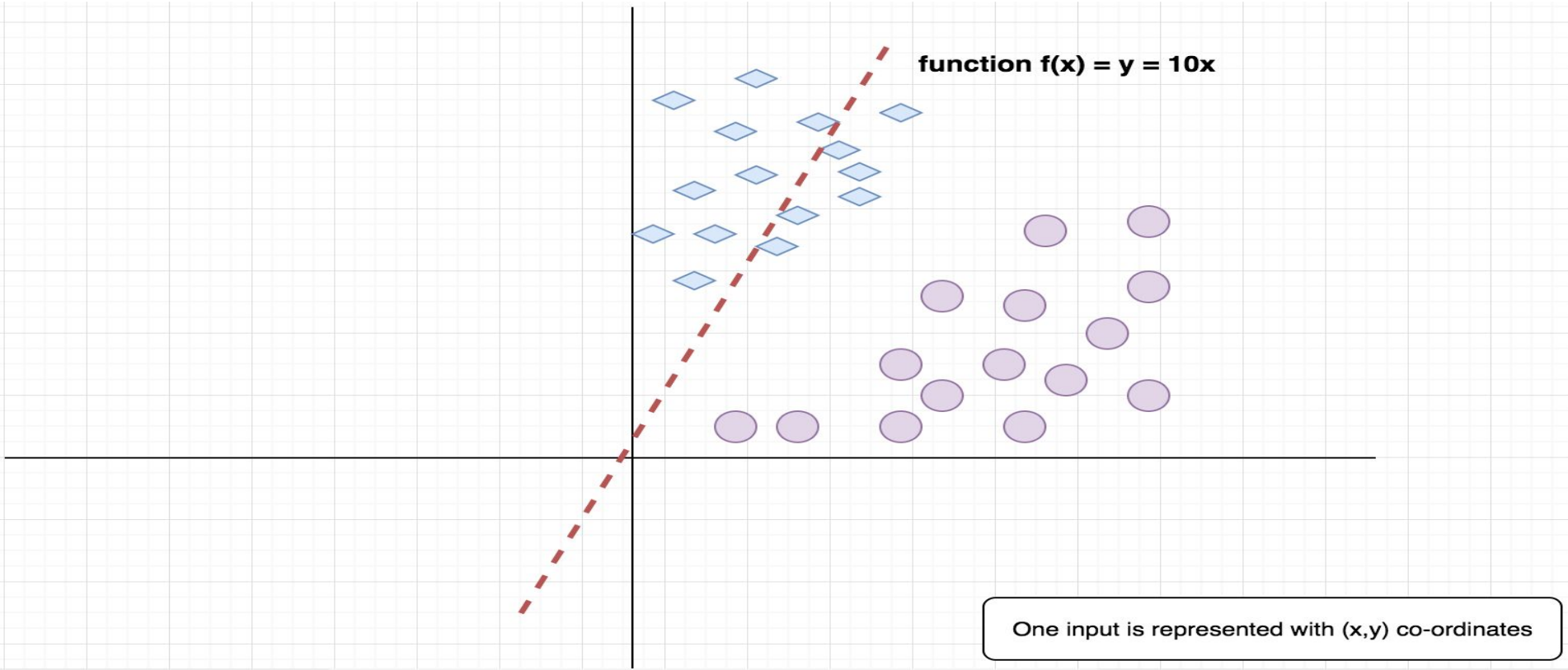


Model Training



Loss / Cost
Loss Function
Training Accuracy
Test Accuracy

y = 10x ?? Not so perfect.. (From previous example..)



Drawn with - https://www.draw.io/

Loss

- What is the **“error”** in **current input prediction**?
- Let us say our method to calculate error (loss) is :
  - 0 for correct shape classification.
  - 1 for wrong shape classification.
- Loss represent error per input.

Cost

- What is the **total error** for **all input** predictions?
- Cost represent total error after seeing all inputs.
- In our current example,
  - Cost = Sum of loss of all inputs = Number of mis-classifications.
- Generally, we take average of the sum.
  - Cost = 1/n (sum of loss of all inputs)

EPOCH - When you complete one loop for all input in training data, we call it 1 epoch.

Loss Functions

- A function that gives an error value for given actual output and predicted output.
  - **Loss Function = F(actual_ouput, predicted_output) => Error**
- There are many types of Loss Functions.
- Past experiments and experience gives idea on what Loss Functions is suitable for given problem.

Examples:

Mean Squared Loss = $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$

Cross Entropy Loss = $-y\log\hat{y} - (1-y)\log(1-\hat{y})$

Training Accuracy, Test Accuracy

- Accuracy of predictions on Training Data -> Training Accuracy
- Accuracy of predictions on Test Data -> Test Accuracy
- Training Accuracy is not a good measure of performance. Why? You test on something you already have seen.
- Test Accuracy is a good measure for a model performance.
- Training Accuracy is used in other factors - Overfitting, Underfitting. (Will be covered in detail in further sessions)

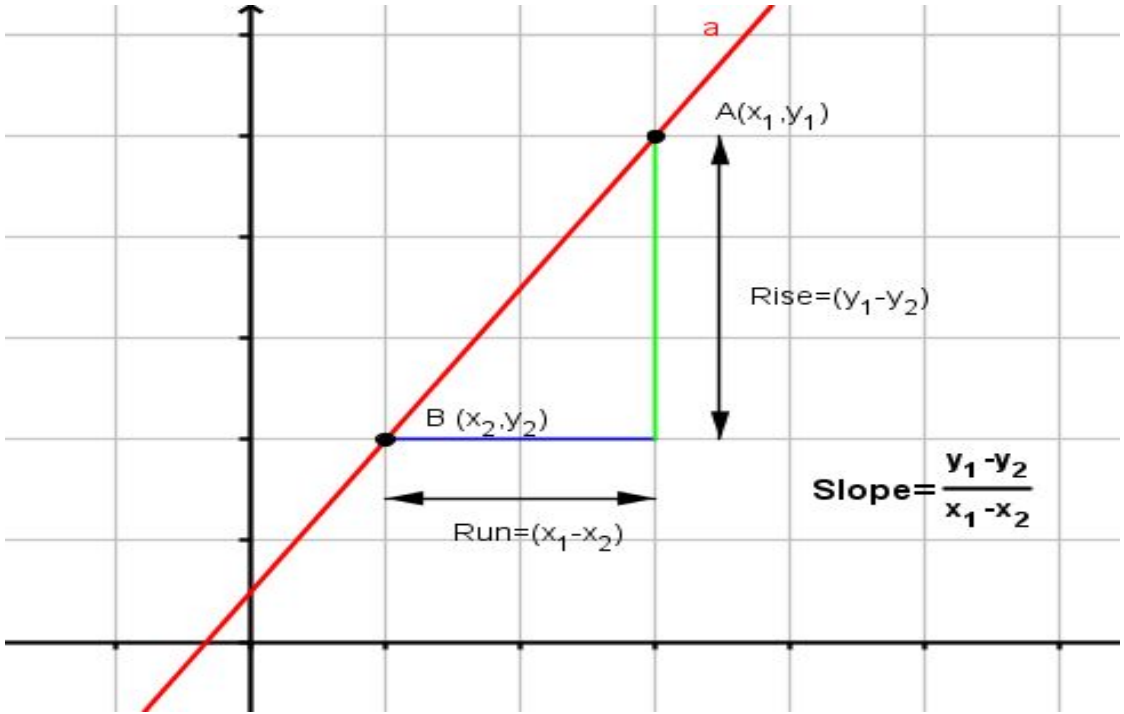
Quick Summary

- Function
- Model
- Model Training (Learning)
- Training Data, Test Data
- Loss, Cost, Loss Function, Training Accuracy, Test Accuracy

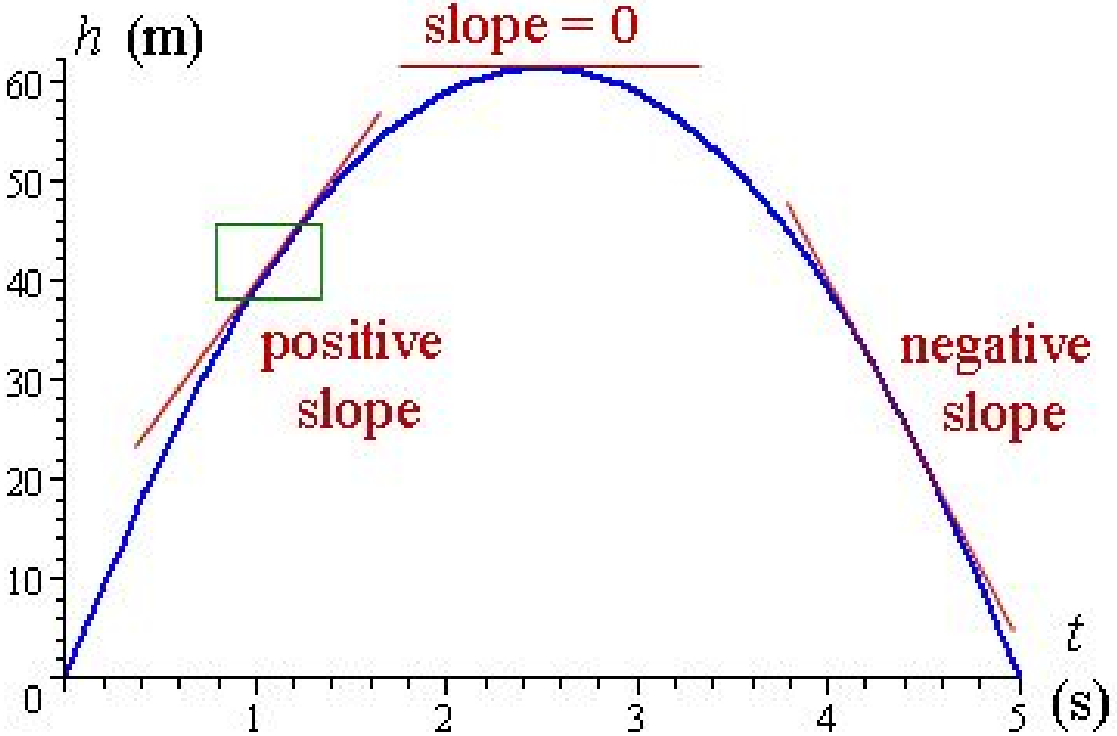
Slope of a Function
Gradients

Slope of a Function

- We know, how to visualize a function as curve.
- Steepness of the hill is called the Slope.
- Similarly, steepness of a curve / line is called Slope.
- Rate of change of value.
- It is just a number (scalar) representing the steepness.



Ref: http://barronstestprep.com/prep/sat/math/Coordinate-Geometry/Slope-of-a-Line.html

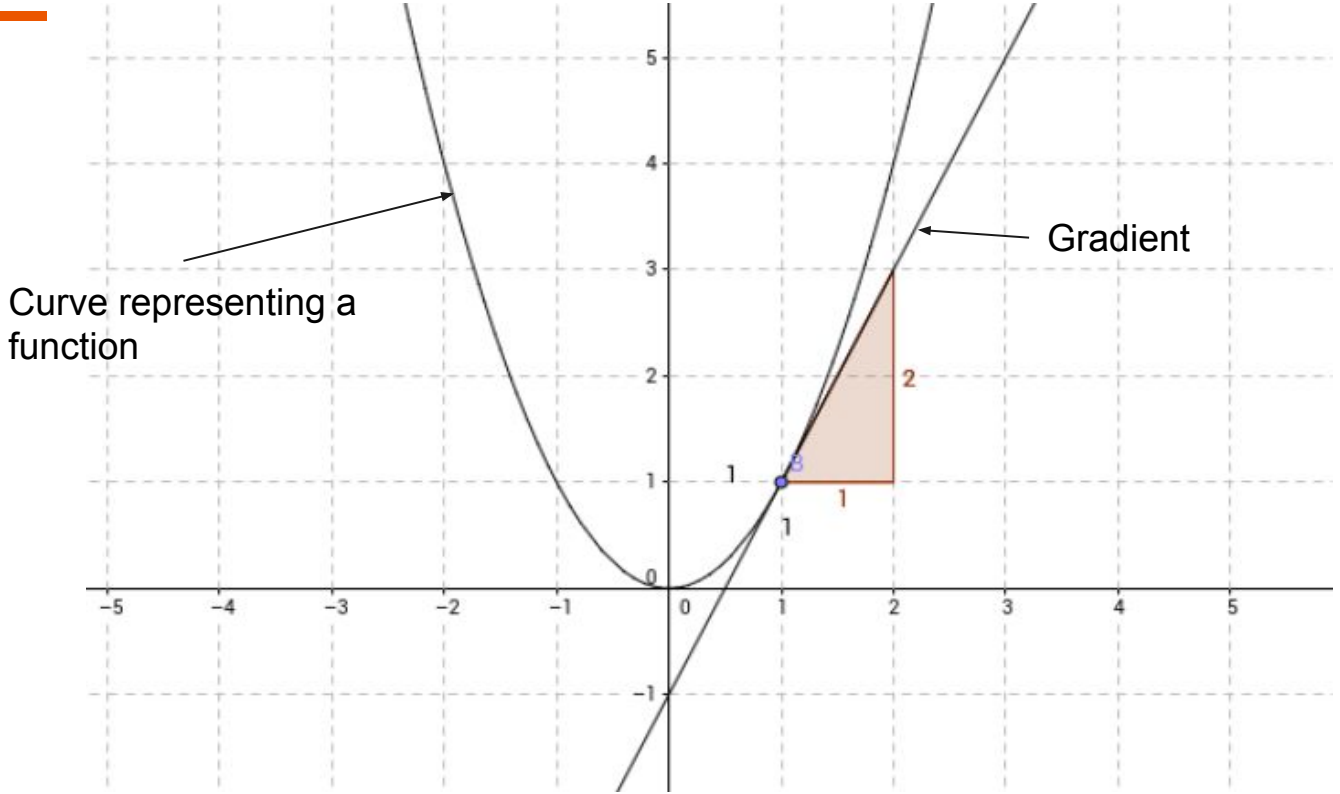


https://www.intmath.com/differentiation/differentiation-intro.php

Gradient

- Intuition behind Gradient is same as Slope.
- Gradient is also used to represent - “Rate of Change”.
- However, Gradient is also a Vector. It has direction.
- Positive slope -> Raising -> Positive Gradient with increasing slope direction.
- Negative slope -> Descending -> Negative Gradient with decreasing slope direction.

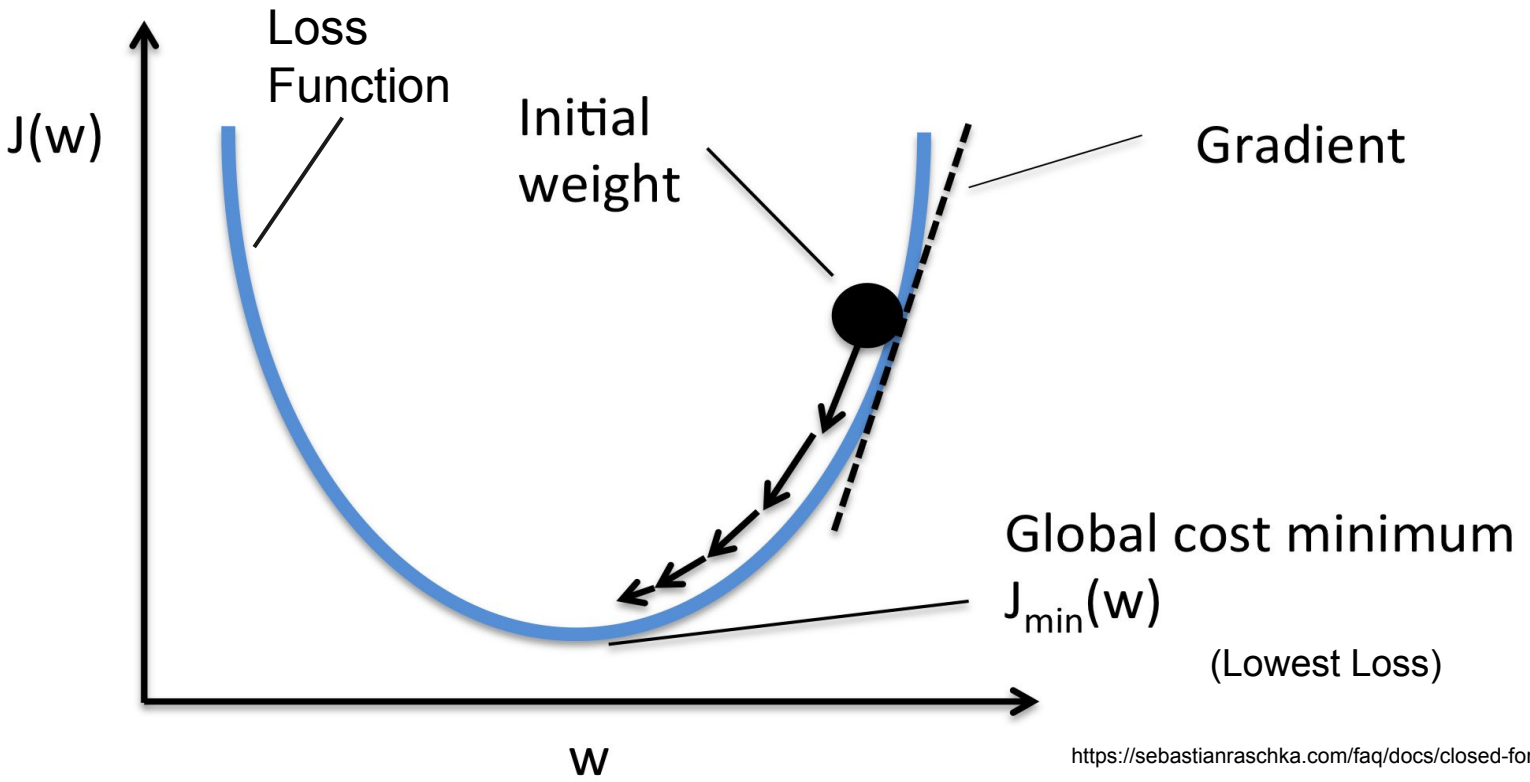
Mathematically Gradient is a vector of partial derivatives of a function representing the curve. (Do not worry, understand the above intuition)



https://www.geogebra.org/material/show/id/16356

Wait a minute, let us step back, I see a correlation..

- For training a model, my objective is to reduce the error (Loss).
- Gradients help me to find the curvature (direction and rate of change of value) of a function.
- If I can represent Loss function as curve and find the gradients, I can move towards lowest loss possible.
- This setting is called "**Optimization**".
- We are trying to **create a setting** where **model training** job is converted to a **optimization problem** with **objective of minimizing the loss**.



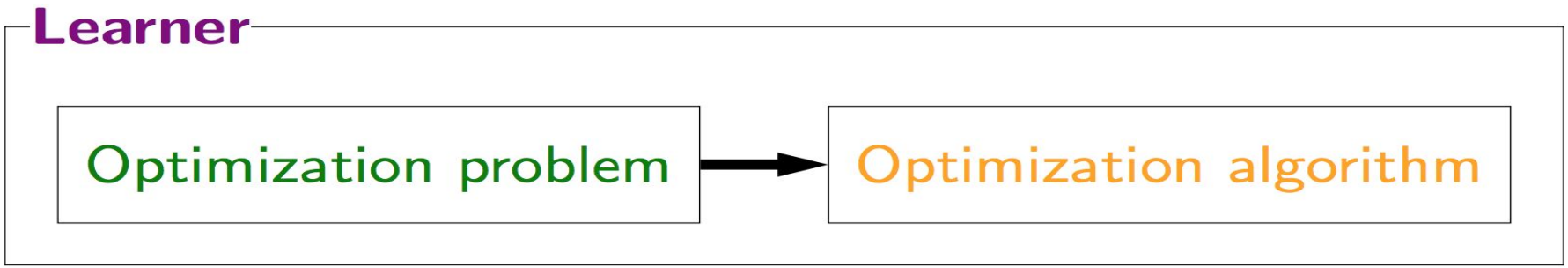
https://sebastianraschka.com/faq/docs/closed-form-vs-gd.html

Why Loss is a Function of Weights?

- All our learning process is about learning these Weights.
- From our previous example, y = 2*x, 2 is actually considered as a weight.
- Different weight you choose, you get different error (Loss).
- Hence, goal is to identify a set of weights that will minimize the loss.

Optimization Algorithm

- Ok, we now converted **Model Training Problem** as an **Optimization Problem** of **Minimizing the Loss Function** w.r.t Weights
- We now need an algorithm to run this optimization as a program.



https://web.stanford.edu/class/cs221/lectures/learning1.pdf

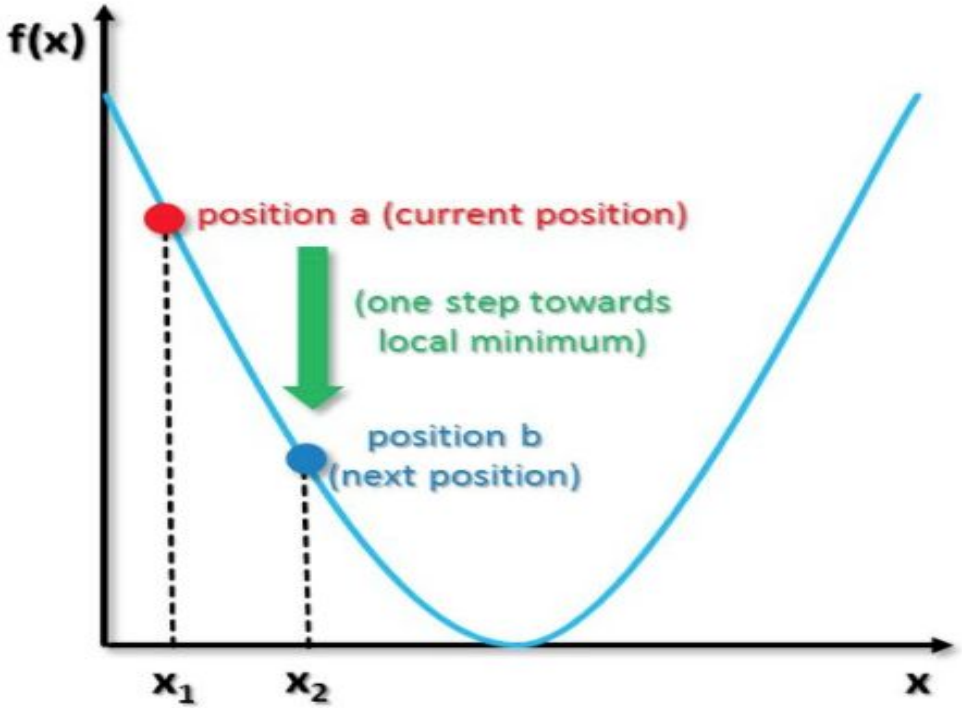
Algorithm: gradient descent

Initialize $\mathbf{w} = [0, \ldots, 0]$

For $t = 1, \ldots, T$:

$$\mathbf{w} \leftarrow \mathbf{w} - \underbrace{\eta}_{\text{step size}} \underbrace{\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})}_{\text{gradient}}$$

https://web.stanford.edu/class/cs221/lectures/learning1.pdf



http://www.big-data.tips/gradient-descent

Gradient Descent

1. Initialize the weights randomly.
2. For each input sample in Training Data, calculate the Loss(actual, predicted) with given weights.
3. Calculate the overall loss by aggregating the loss for each input sample.
4. Compute the gradient of loss w.r.t Weights.
5. Make changes to Weights based on a factor (Learning rate) of the calculated gradient.
6. Restart the loop from Step 2.
7. Run Step 2 to Step 5 for multiple times (epochs)

Issues:

1. Updates once per epoch i.e., 1 update after seeing all input data.
2. Very Slow. Not scalable. Takes long time to make changes and reach the minimum loss point (convergence)

Stochastic Gradient Descent

- Same as Gradient Descent.
- Difference is - **Updates** weights based on the gradients of loss (error) immediately after seeing **each input** sample.
- If your training data has 100 input samples, you update weights 100 times.

Issues:

- Too many updates. Not scalable for large data.
- May lead to issue where you are bouncing around the curve without being able to reach minimum loss point.

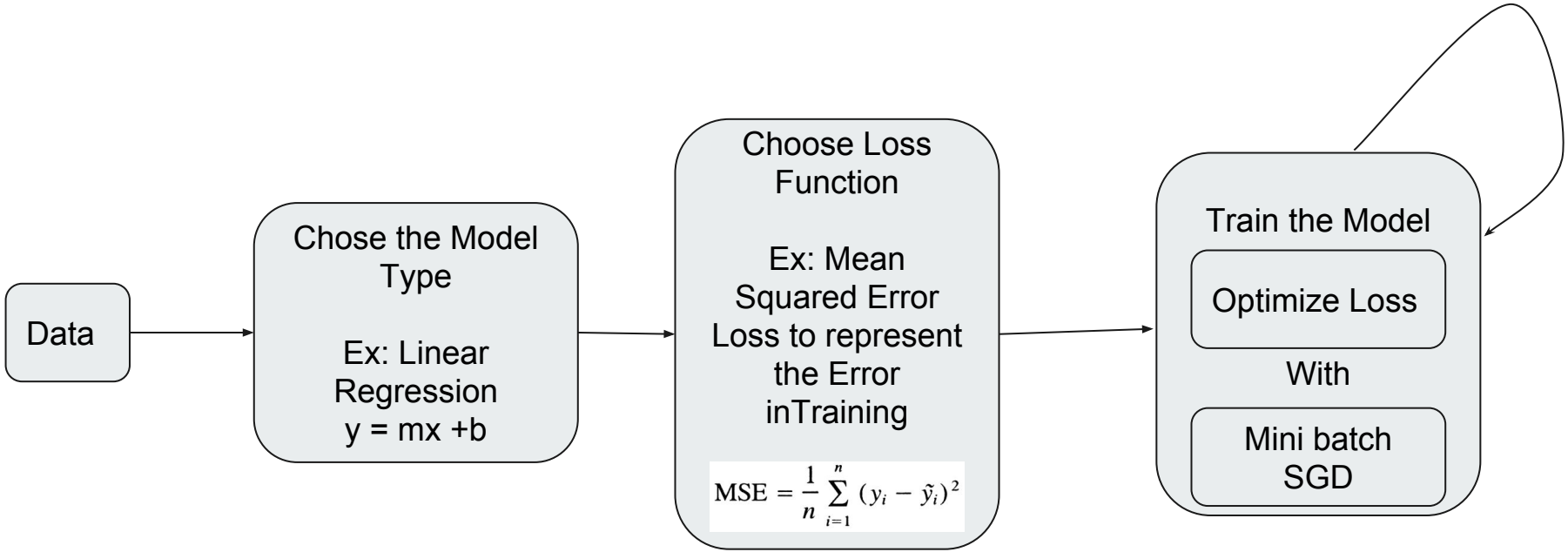
Mini-batch Stochastic Gradient Descent

- Updates weights **once after seeing a mini-batch of data** w.r.t gradients of the loss of mini-batch of data.
- For example, if we have 100 training samples and mini-batch size is 5. We will have 20 updates per epoch.
- Most commonly used in practice.
- Scalable.
- Overcomes issues with very few updates in Gradient Descent or too many updates in Stochastic Gradient Descent.
- If batch size is 1, then this will be same as SGD.

Hyperparameters

- We mentioned various parameters involved in the Algorithm - learning rate, number of epochs, batch-size.
- Each of these parameters of algorithms is called Hyperparameters.
- There are suggested patterns in choosing the hyperparameters. But, do not work always in all problem setting.
- Choosing the best hyperparameters is also one of the step in experimentation phase.
- Active research area.
- Let us ignore this for now.

To Summarize..



Demo

Handwritten Digit Recognition with Apache MXNet

Next Steps

- Linear Regression v/s Logistic Regression
- Backpropagation
- Learning in Neural Network => SGD + Backprop
- Overfitting, Regularization
- Convolutional Neural Networks - MNIST with CNN

Resources

- Read more about Deep Learning along with Learning MXNet - http://gluon.mxnet.io/index.html
- Slides and Jupyter Notebook available at - https://github.com/awsaiguru/resources/tree/master/presentations/awsaiguru_meetup_5_paloalto_Oct_2017
    - $ git clone https://github.com/awsaiguru/resources/ .