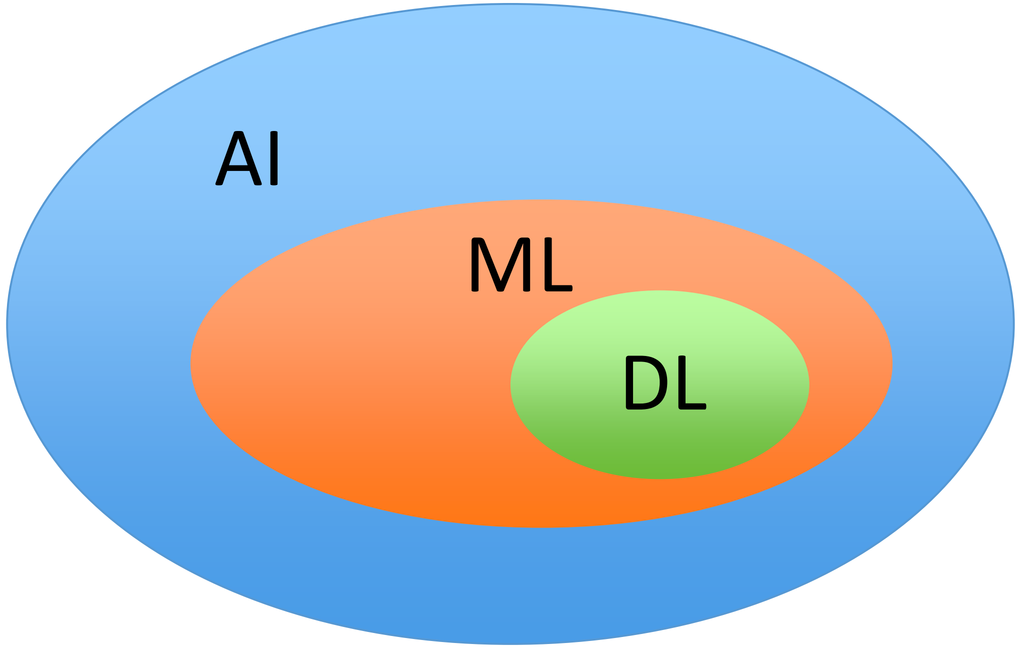
CMPT 732-G200. Practices for Visual Computing

Ali Mahdavi Amiri

AI, ML, DL

- Artificial Intelligence (AI)
- Machine Learning (ML)
- Deep Learning (DL)

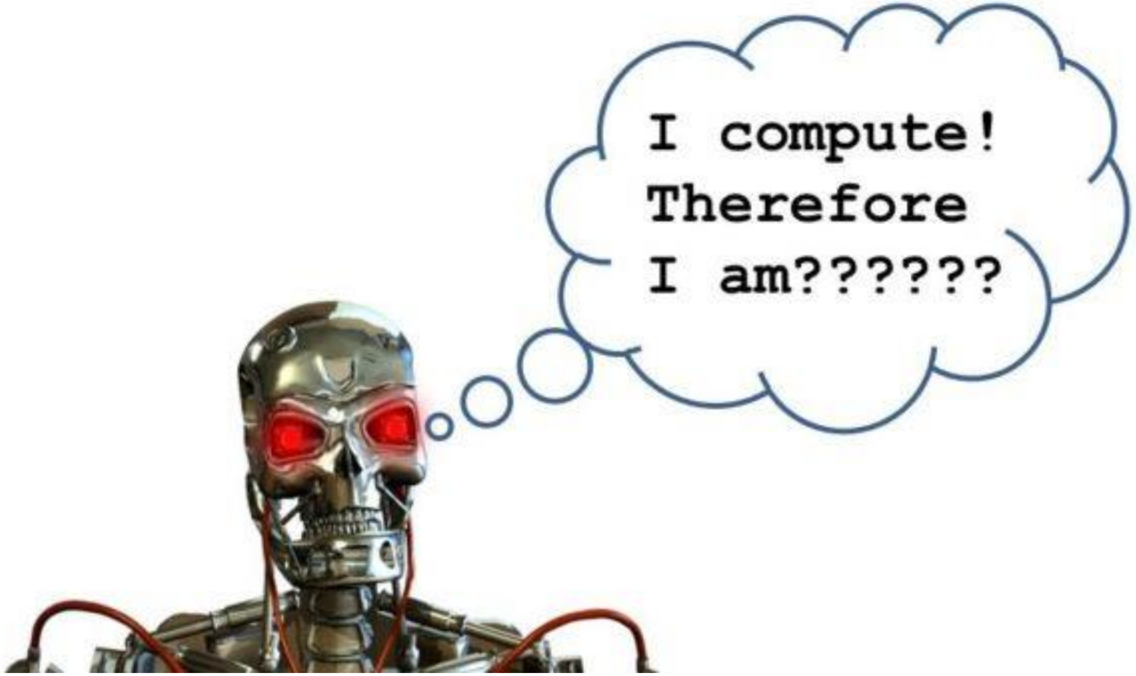


AI

- AI was born in 1950s.
- Can computers think?

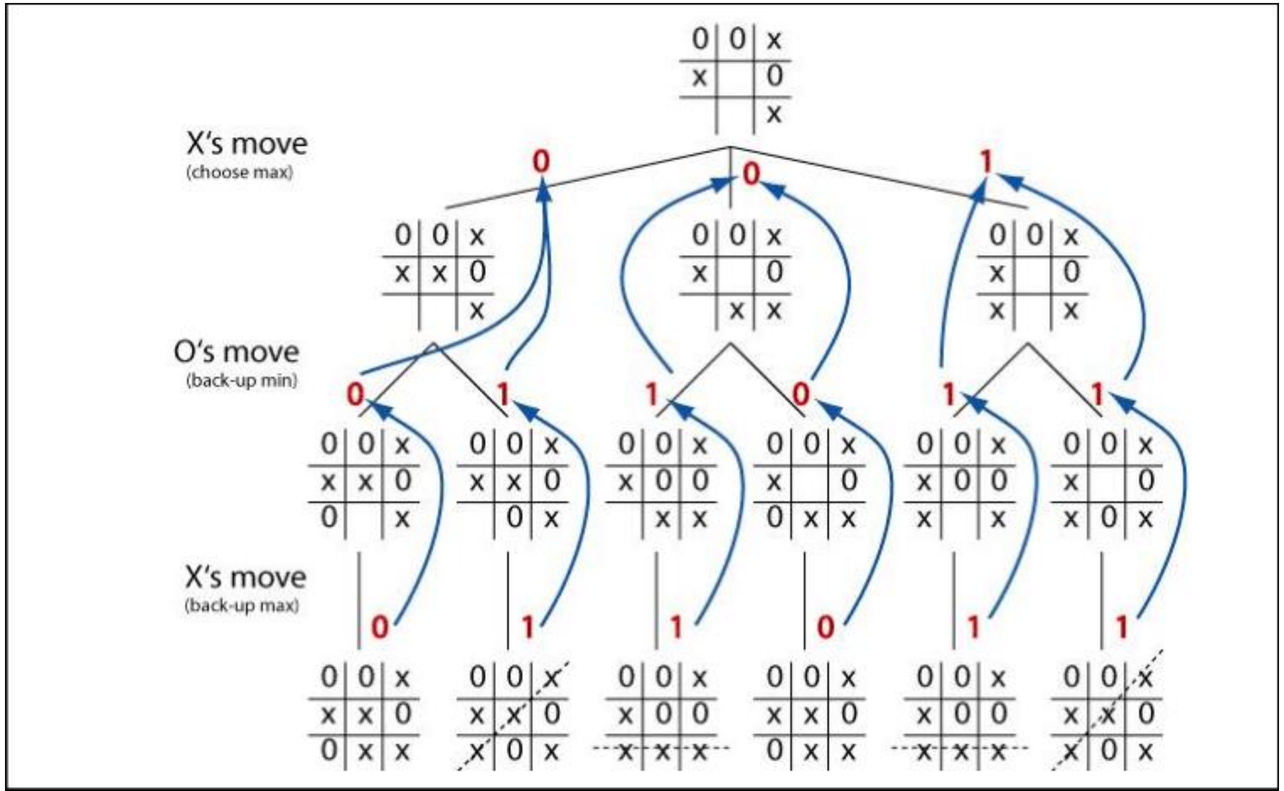
AI

- The effort to automate intellectual tasks normally performed by humans.



Symbolic AI

- For a long time, experts believed that human-level artificial intelligence could be achieved by having programmers handcrafted a sufficiently large set of explicit rules for manipulating knowledge.



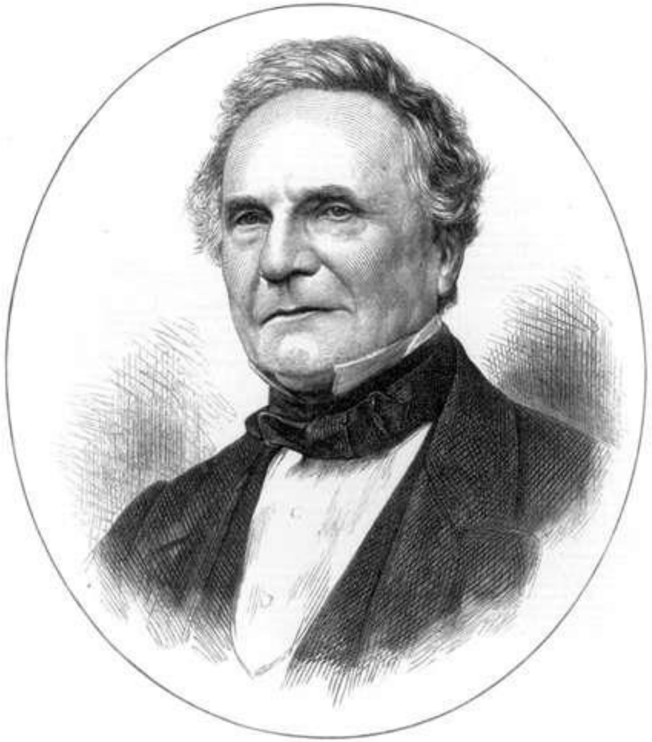
Symbolic AI

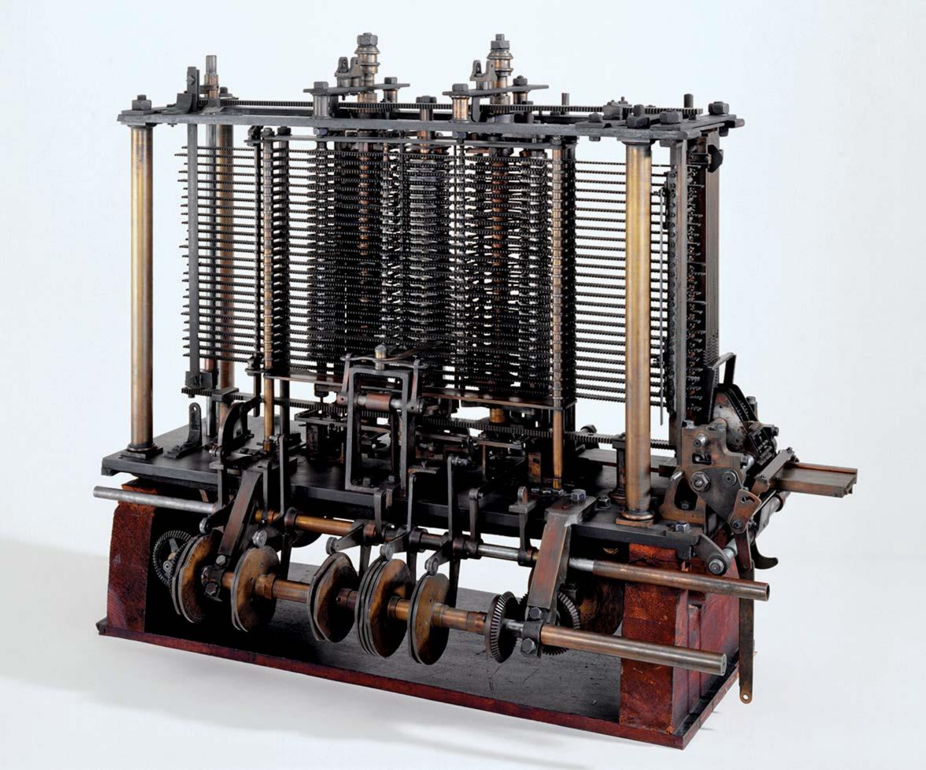
- For a long time, experts believed that human-level artificial intelligence could be achieved by having programmers handcrafted a sufficiently large set of explicit rules for manipulating knowledge.

- It gets intractable in many tasks (e.g., image classification)

Machine Learning

- Lady Ada Lovelace with Charles Babbage introduced Analytical Engine which was a mechanical computer.





Machine Learning

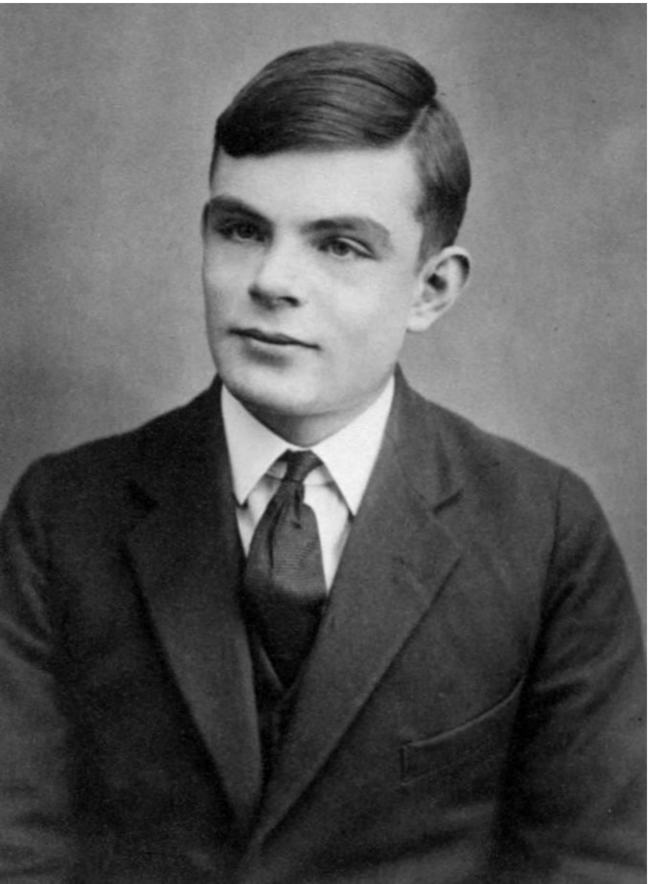
- As mentioned by Ada Lovelace, this machine was limited to the tasks that were instructed by human (programmers).

Machine Learning

- Can machines go beyond instructions?
- Can they become creative?

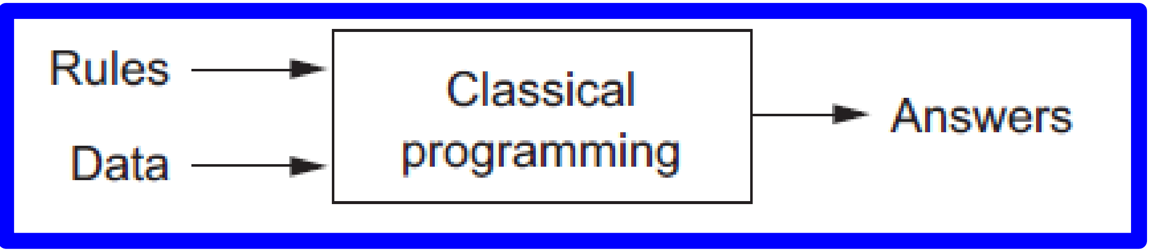
Machine Learning

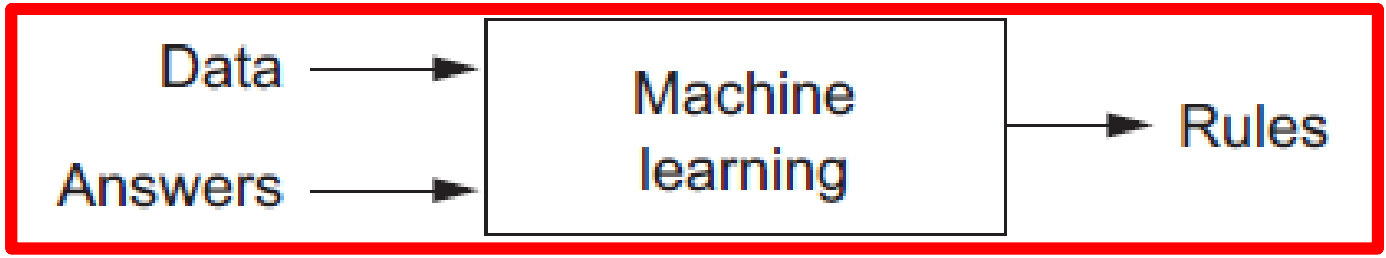
- These discussions lead to introducing Turing Test in 1950.



Machine Learning

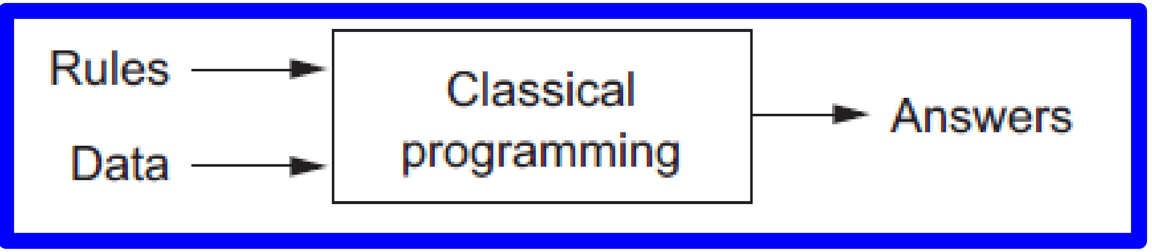
- As opposed to classical programming, in machine learning, a set of rules are learned.

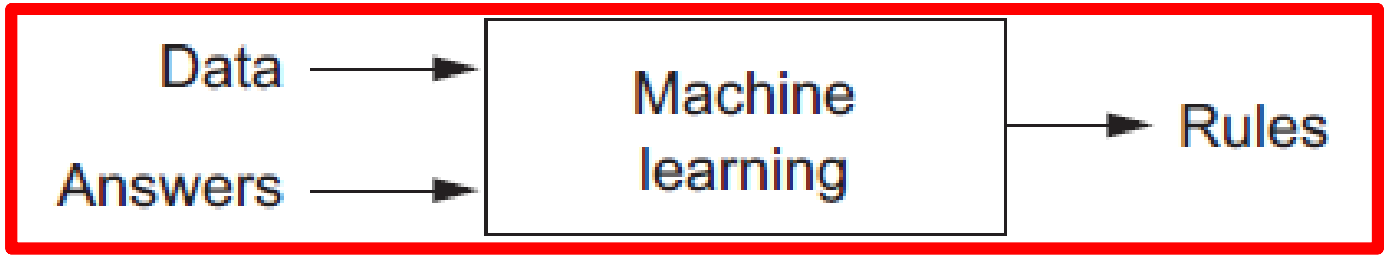




Machine Learning

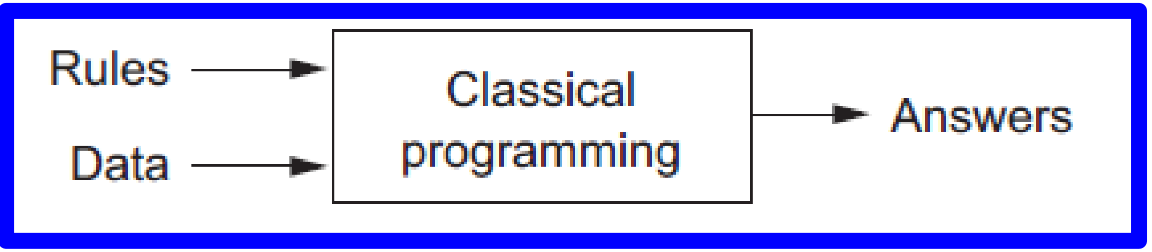
- As opposed to classical programming, in machine learning, a set of rules are learned.
- These rules can be applied to new data to produce original answers.

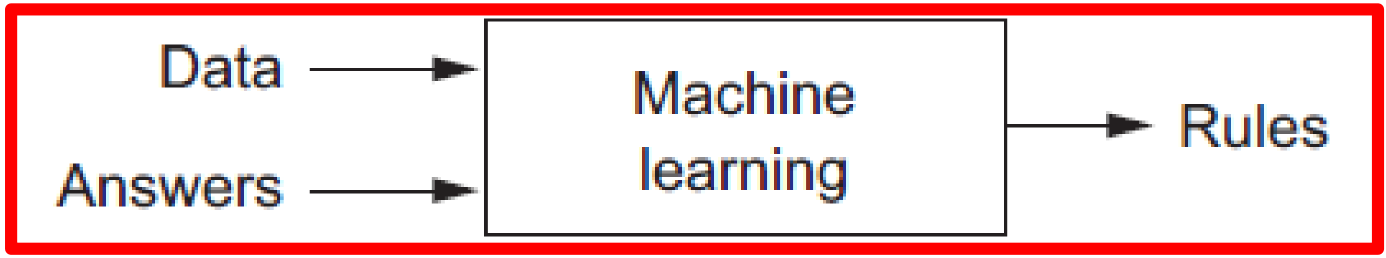




Machine Learning

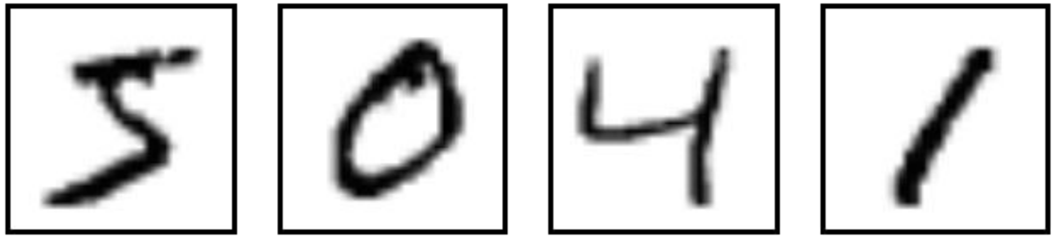
- As opposed to classical programming, in machine learning, a set of rules are learned.
- These rules can be applied to new data to produce original answers.
- In fact, machines are trained rather than directly programmed.





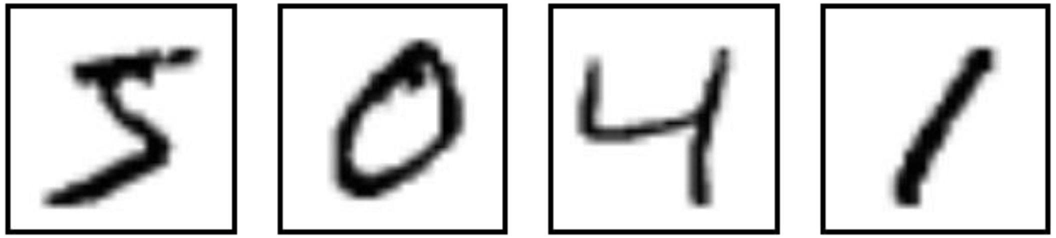
What is Learning?

- Components of Machine Learning:
    - Input data points
        - An image of a digit



What is Learning?

- Components of Machine Learning:
  - Input data points
    - An image of a digit

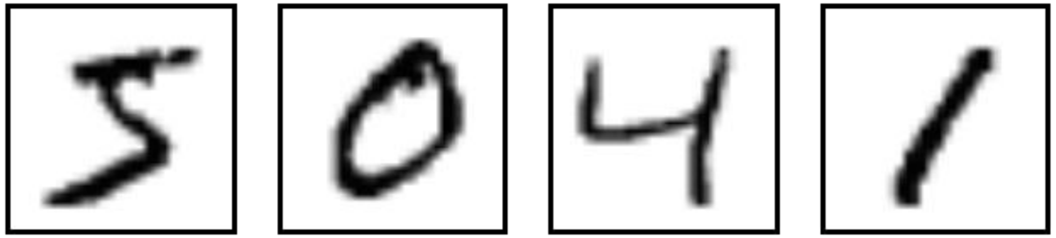


  - Examples of the expected output (Ground Truth)
    - A set of digits and their labels

5,0,4,1

What is Learning?

- Components of Machine Learning:
  - Input data points
    - An image of a digit
  - Examples of the expected output (Ground Truth)
    - A set of digits and their labels

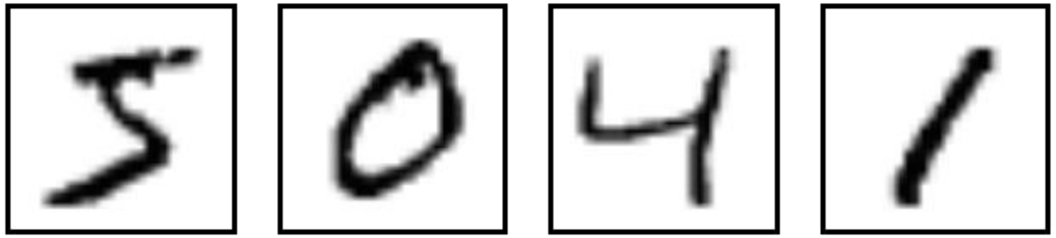


5,0,4,1

  - Measurement of the performance of the machine
    - If machine returns the correct label for a digit.

What is Learning?

- Components of Machine Learning:
  - Input data points
    - An image of a digit
  - Examples of the expected output (Ground Truth)
    - A set of digits and their labels



5,0,4,1

  - Measurement of the performance of the machine
    - If machine returns the correct label for a digit.

What is Learning?

- Measurement of the performance of the machine
  - If machine return the correct label for a digit.

- This process gives a feedback to adjust the way algorithm works.

What is Learning?

- Measurement of the performance of the machine
  - If machine return the correct label for a digit.

- This process gives a feedback to adjust the way algorithm works.
- This adjustment is learning.

Models

- Machine learning models can learn in two main forms
  - Unsupervised
  - Supervised

Unsupervised Learning

- Learning a function that learns a commonality in data that has not been classified/labeled/categorized.

Unsupervised Learning

- Learning a function that learns a commonality in data that has not been classified/labeled/categorized.

- We do not have ground truth data.

Unsupervised Learning

- Learning a function that learns a commonality in data that has not been classified/labeled/categorized.

- We do not have ground truth data.
    - K-means
    - PCA
    - SVD

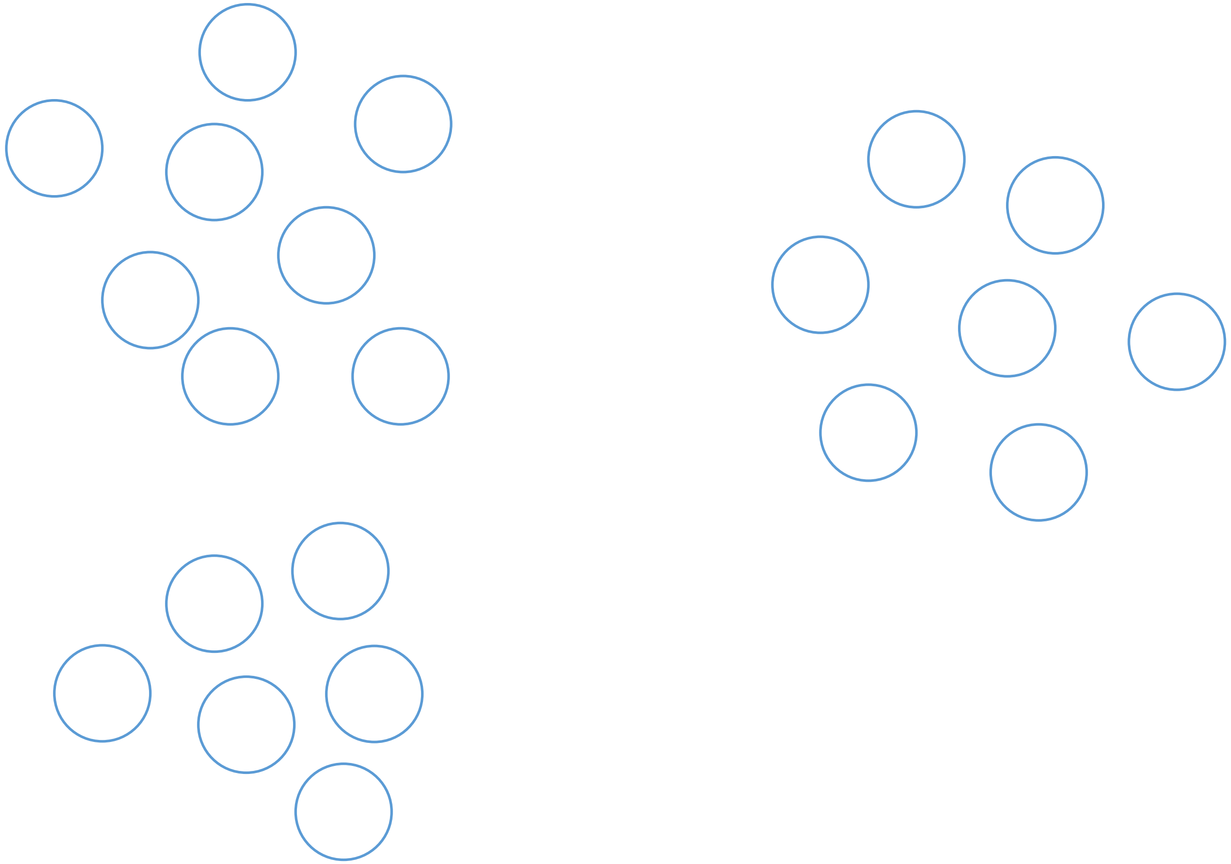
Unsupervised Learning

- Learning a function that learns a commonality in data that has not been classifies/labeled/categorized.

- We do not have ground truth data.
  - K-means
  - PCA
  - SVD

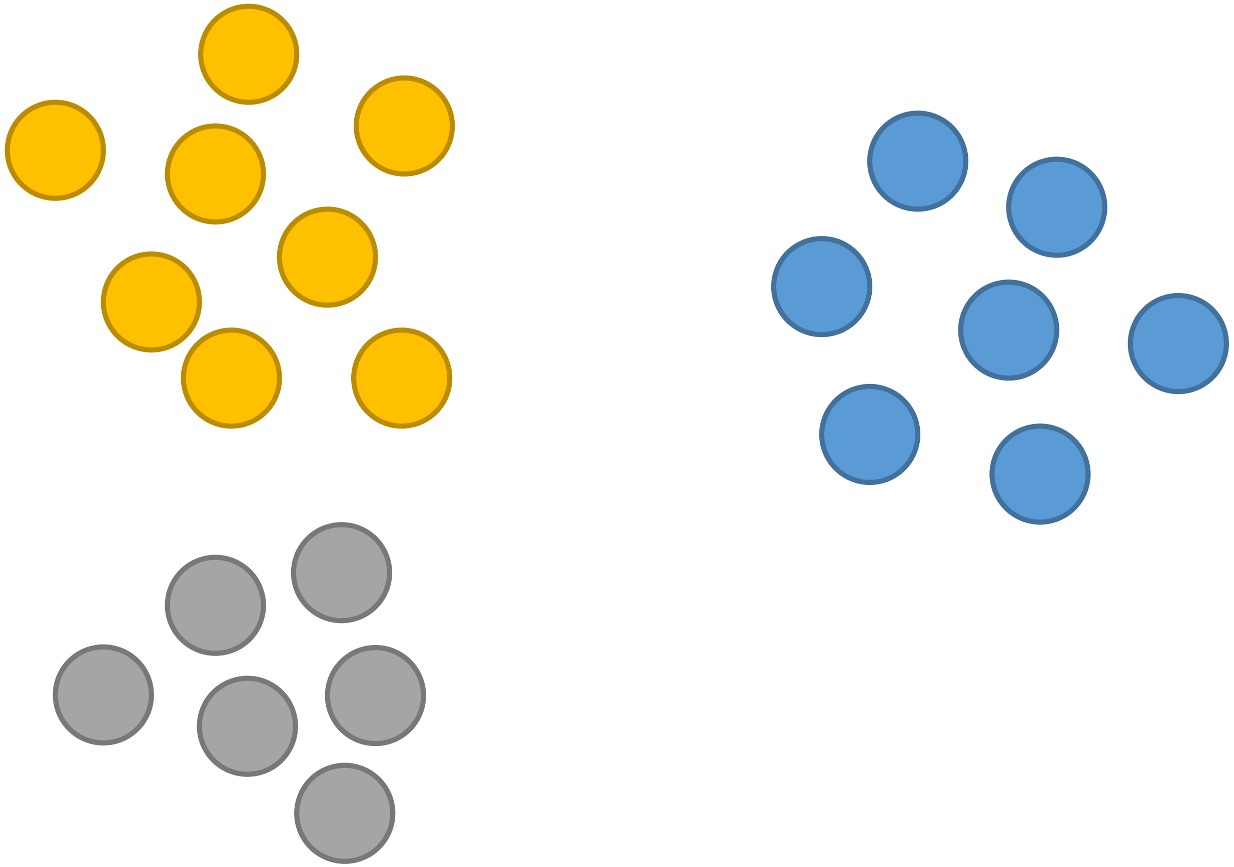
K-means

- Unsupervised model to cluster n data point



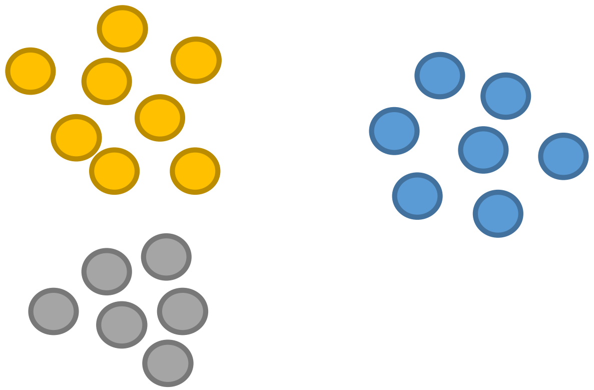
K-means

- Unsupervised model to cluster n data point into k clusters.



K-means

- Main objective is to minimize

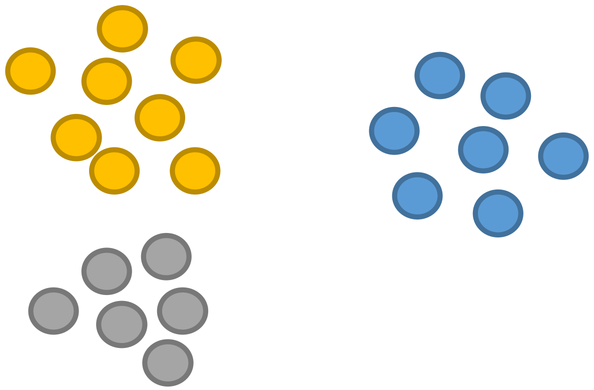


$$\sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

ith cluster

K-means

- Main objective is to minimize



$$\sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

ith cluster

Mean of points in S_i

Algorithm

- Assignment step: Assign each point x to the cluster S_i whose mean μ_i is closest to x.

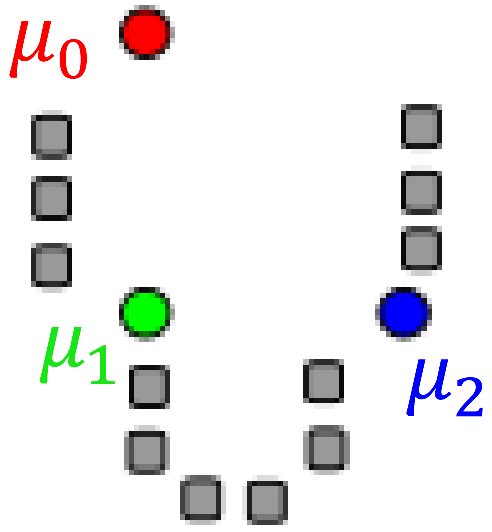
Algorithm

- Assignment step: Assign each point x to the cluster S_i whose mean μ_i is closest to x.

- Update step: Calculate new means to the centroids of data points in new S_i.

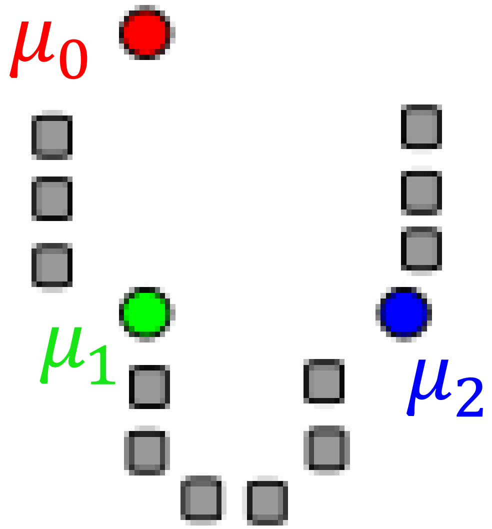
Algorithm

Assign initial centroids

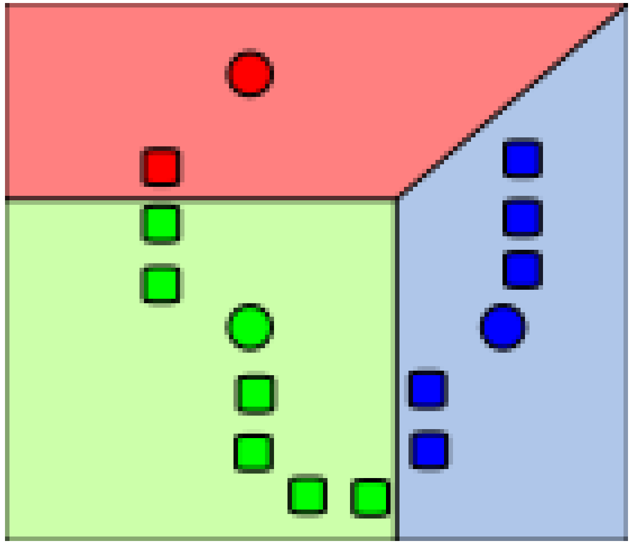


Algorithm

Assign initial centroids

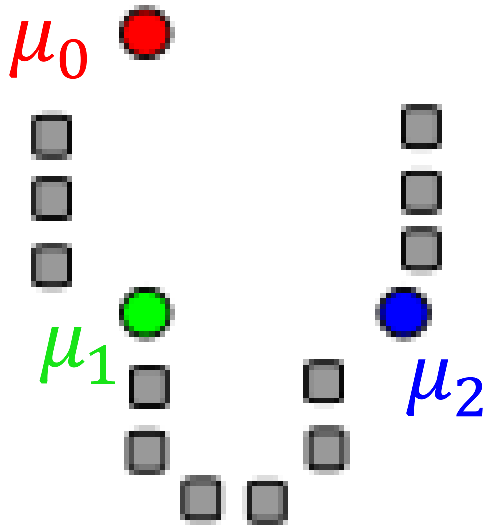


Assignment step

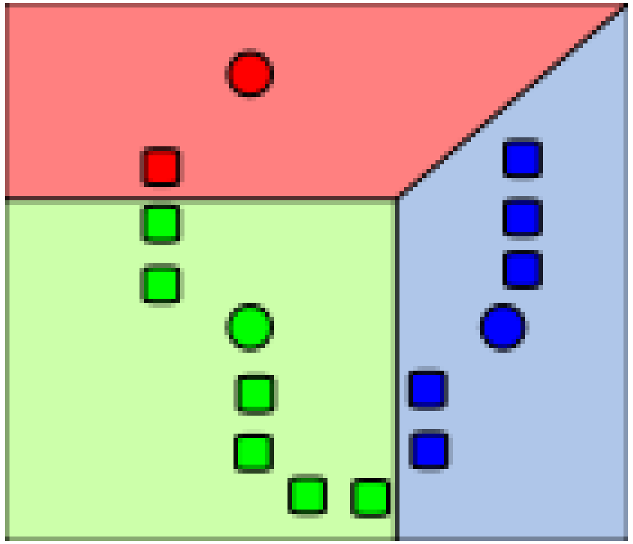


Algorithm

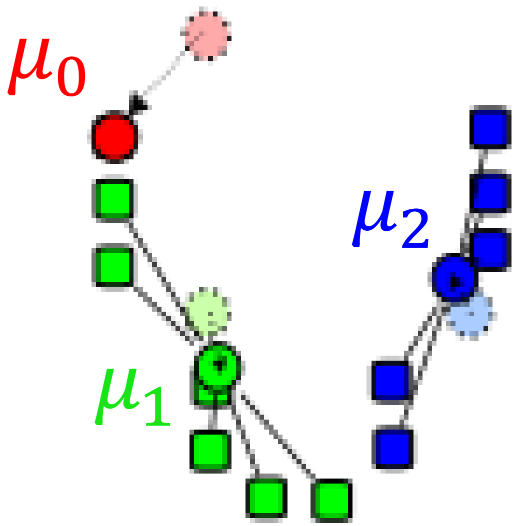
Assign initial centroids



Assignment step



Update step



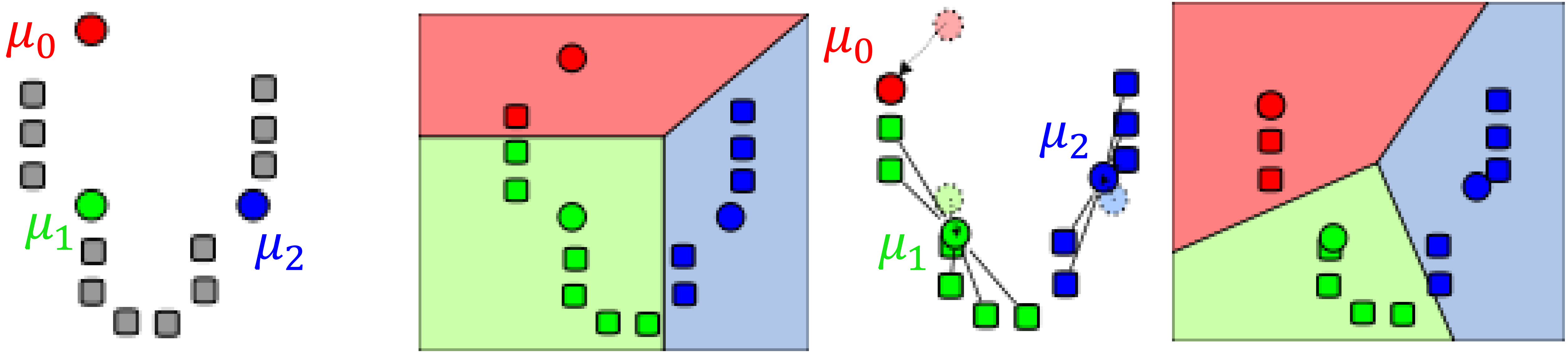
Algorithm

Assign initial centroids

Assignment step

Update step

Assignment Step



Supervised Learning

- Learning a function that maps an input to an output based on example input-output pairs.

Supervised Learning

- Learning a function that maps an input to an output based on example input-output pairs.

- We have ground-truth data.

Supervised Learning

- Learning a function that maps an input to an output based on example input-output pairs.

- We have ground-truth data.
  - Least square line
  - Linear regression
  - Deep Neural Network (supervised)

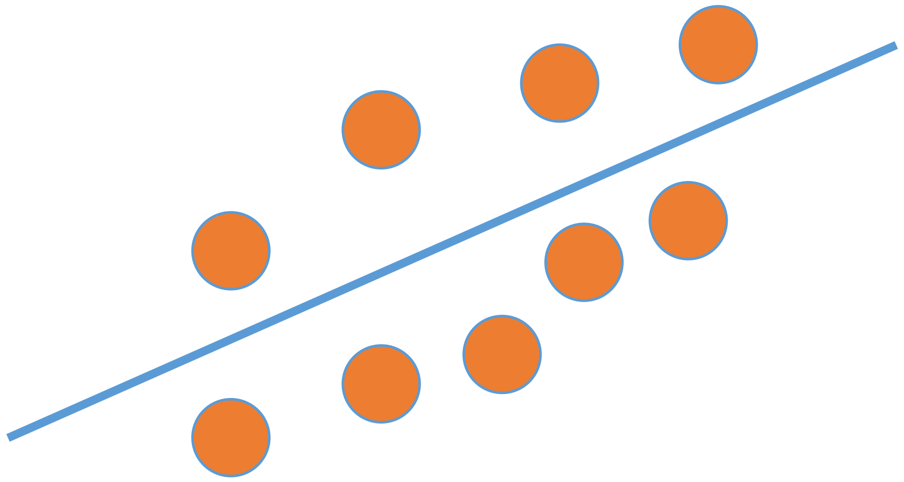
Supervised Learning

- Learning a function that maps an input to an output based on example input-output pairs.

- We have ground-truth data.
  - Least square line
  - Linear regression
  - Deep Neural Network (supervised)

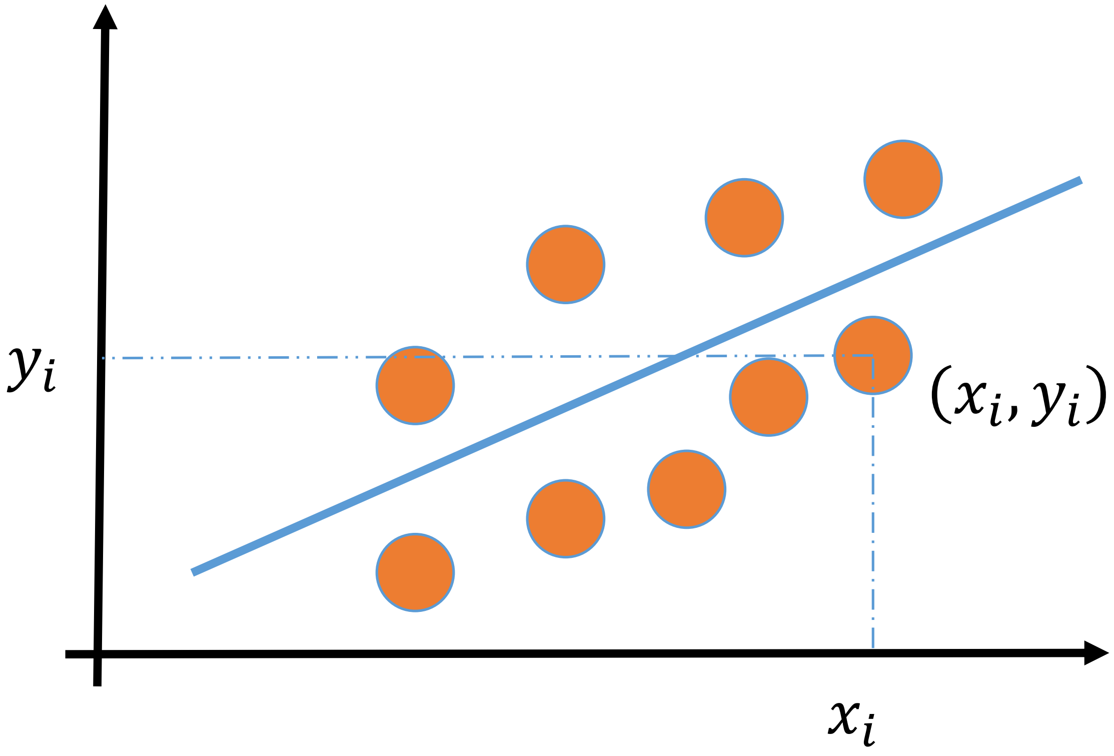
Least Square Lines

- Given data $\{(x_1, y_1), \dots, (x_N, y_N)\}$, we want to find a line $y = mx + b$ which is close to our data.



Least Square Lines

- Given data $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, we want to find a line $y = mx + b$ which is close to our data.

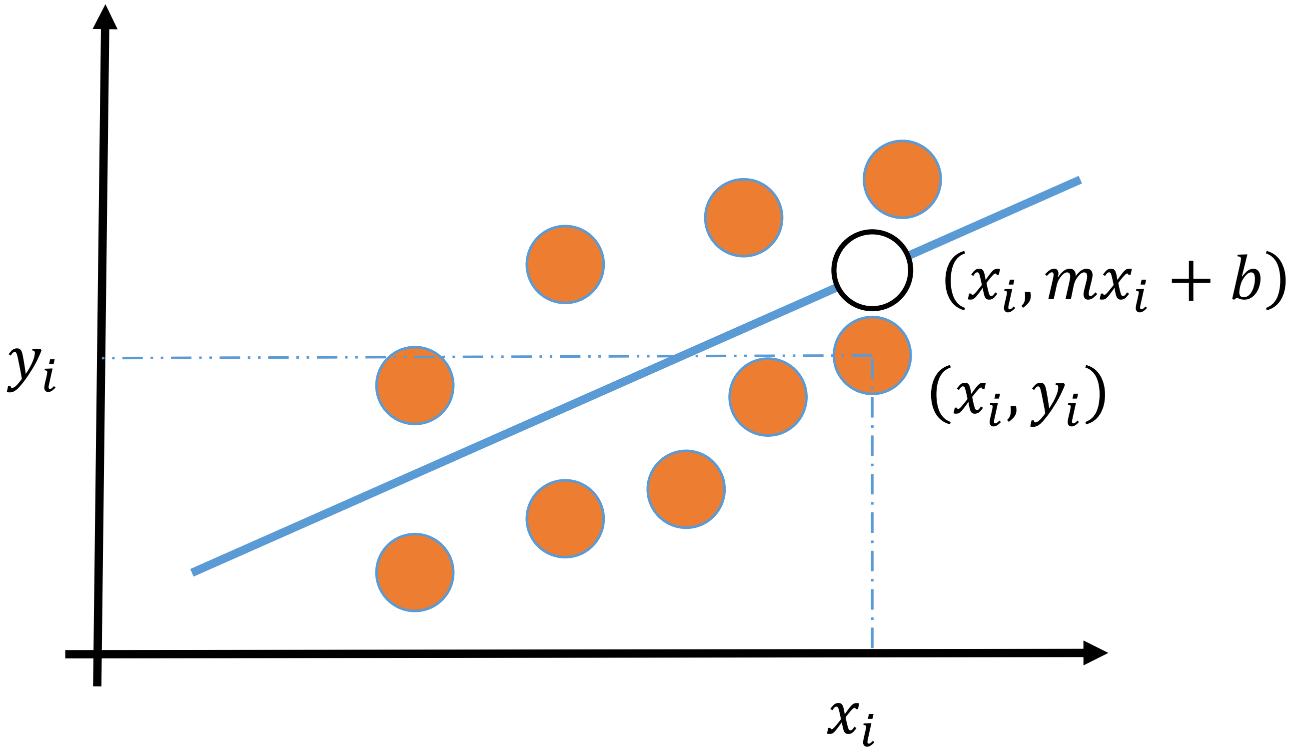


$$E(a, b) = \sum_{i=1}^{N} (y_i - (mx_i + b))^2$$

Least Square Lines

- Given data $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, we want to find a line $y = mx + b$ which is close to our data.

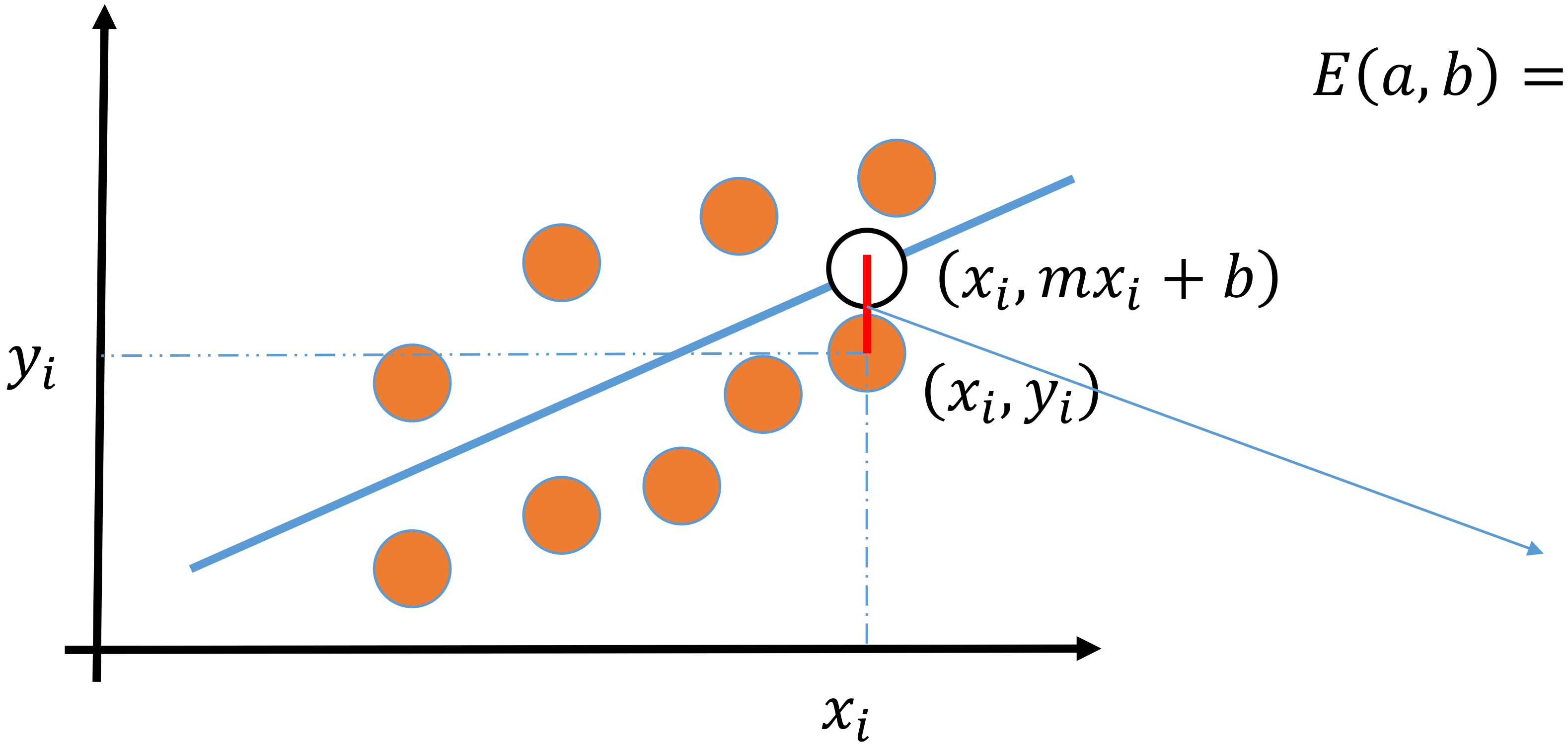
$$E(a, b) = \sum_{i=1}^{N} (y_i - (mx_i + b))^2$$



Least Square Lines

- Given data $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, we want to find a line $y = mx + b$ which is close to our data.

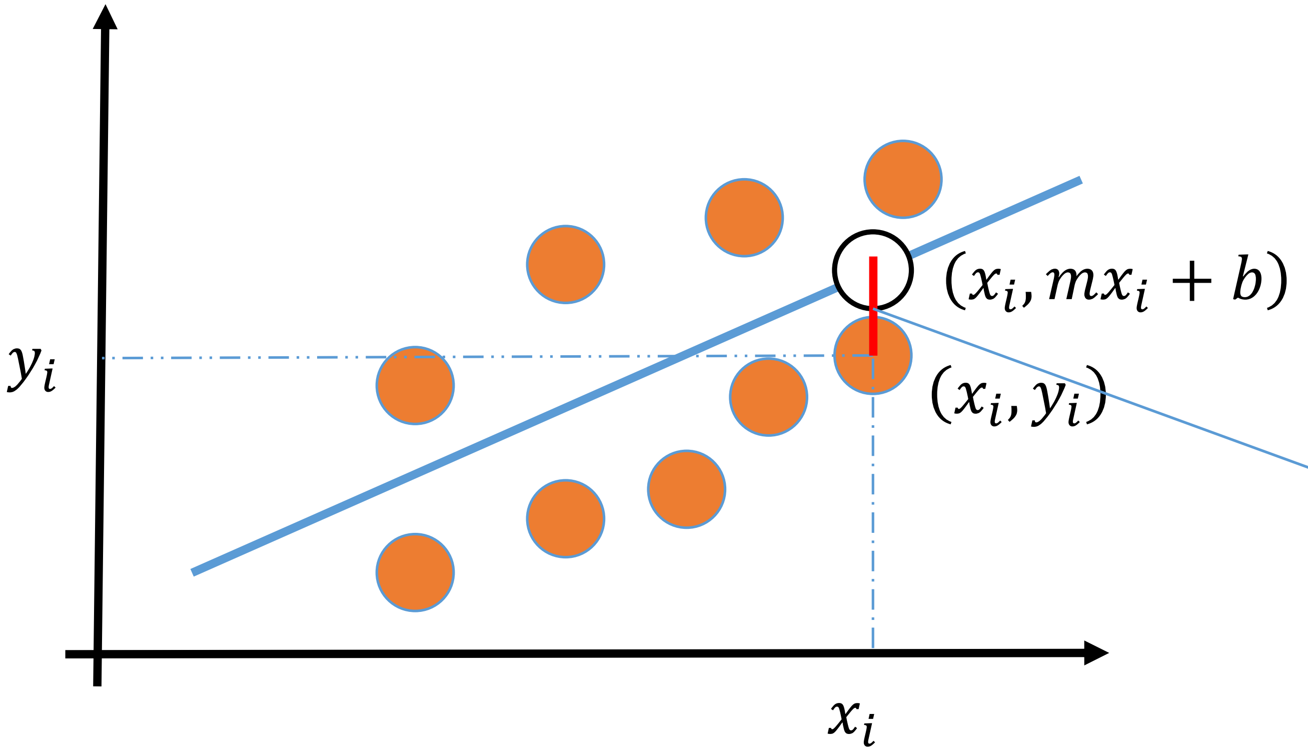
$$E(a, b) = \sum_{i=1}^{N} (y_i - (mx_i + b))^2$$



Minimize this squared distance

Least Square Lines

- What should we do?

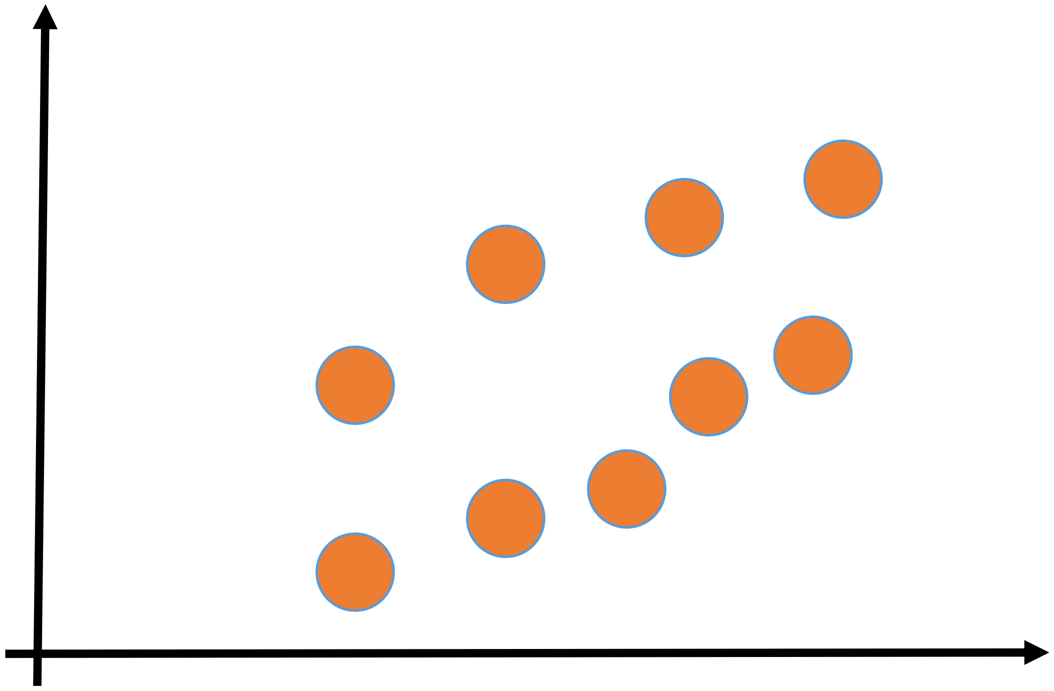


$$E(a,b) = \sum_{i=1}^{N} (y_i - (mx_i + b))^2$$

Minimize this squared distance

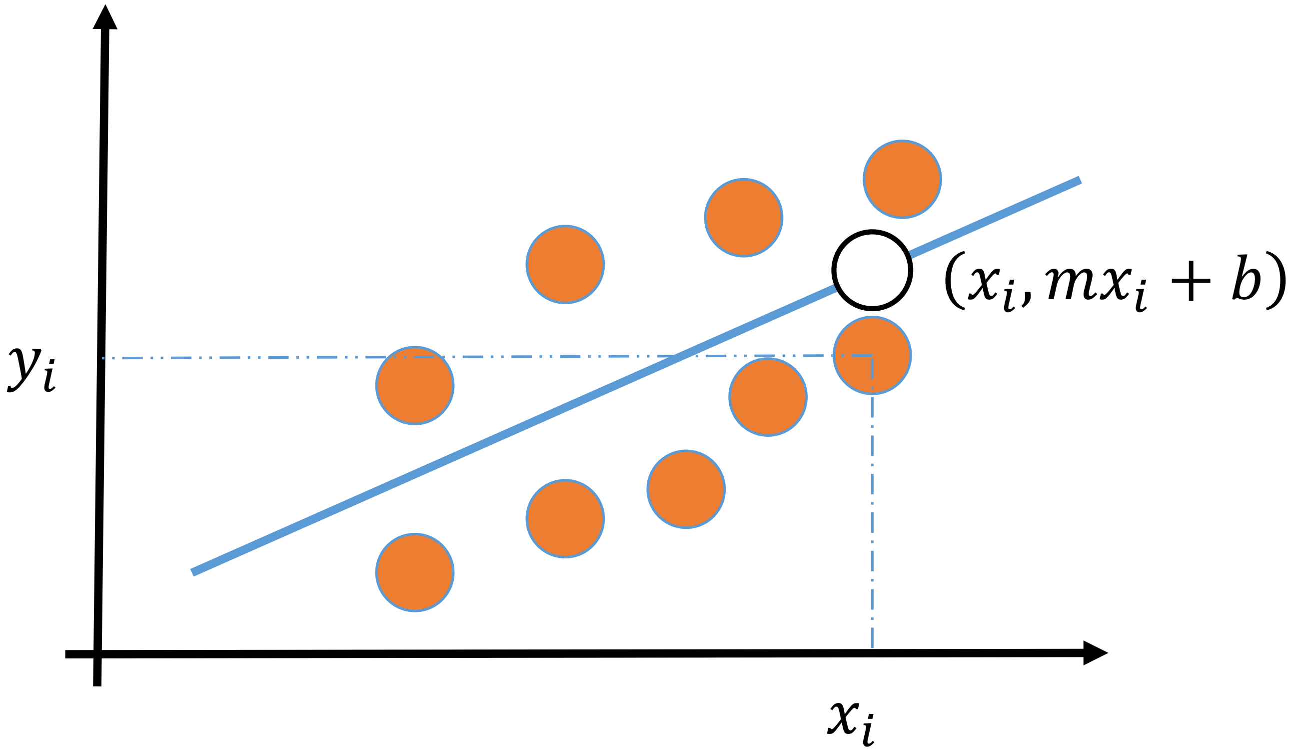
Necessities

- Data



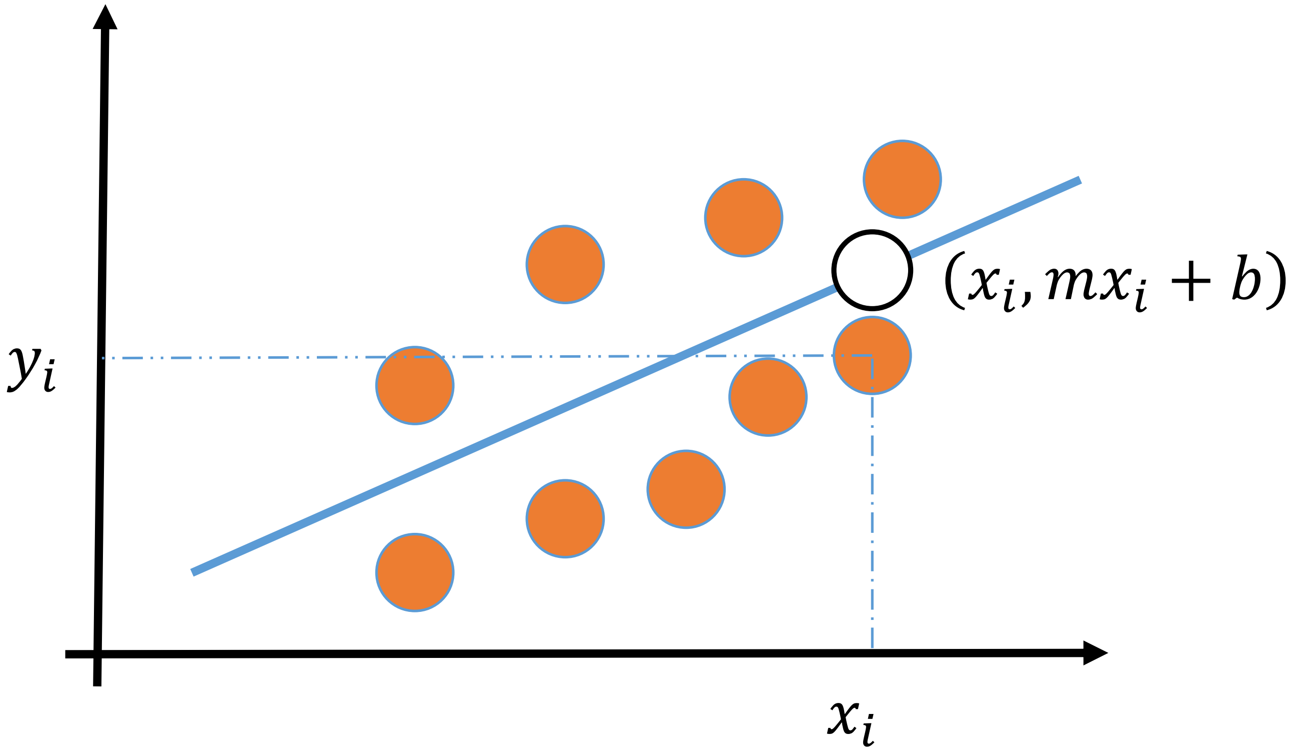
Necessities

- Model



Necessities

- Loss function (differentiable)



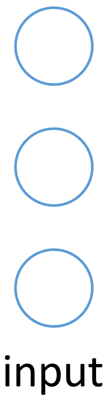
$$E(a,b) = \sum_{i=1}^{N} (y_i - (mx_i + b))^2$$

Neural Networks

- Single lines are too restrictive. We want a more general non-linear model.

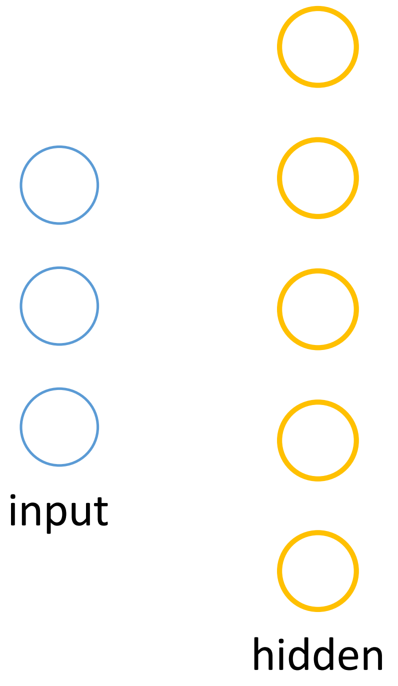
Neural Networks

- Trainable functions that can be used as a mapping between an input and a desired output.



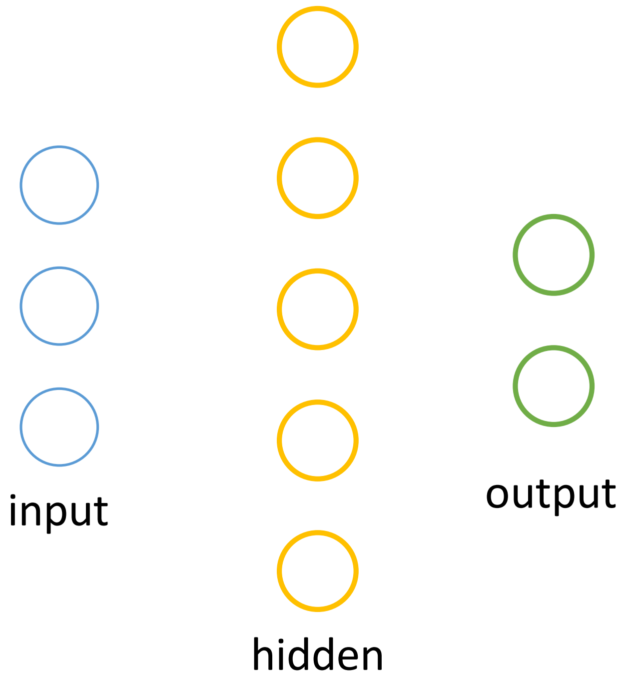
Neural Networks

- Trainable functions that can be used as a mapping between an input and a desired output.



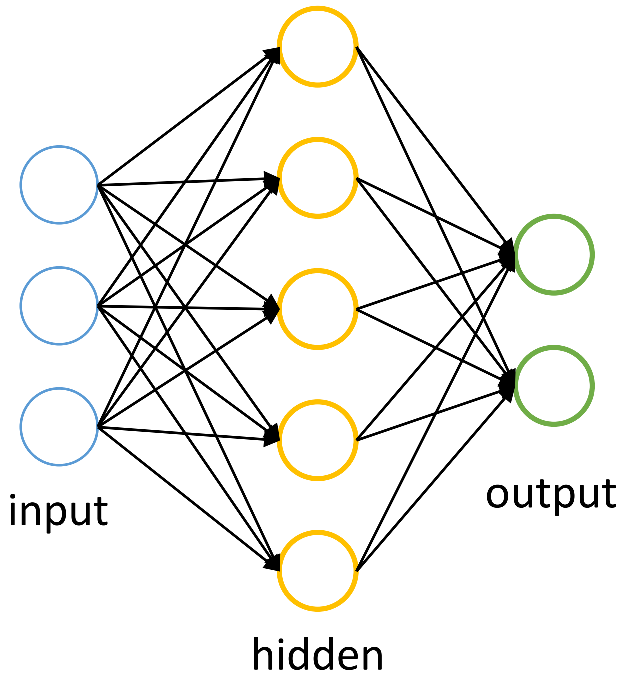
Neural Networks

- Trainable functions that can be used as a mapping between an input and a desired output.



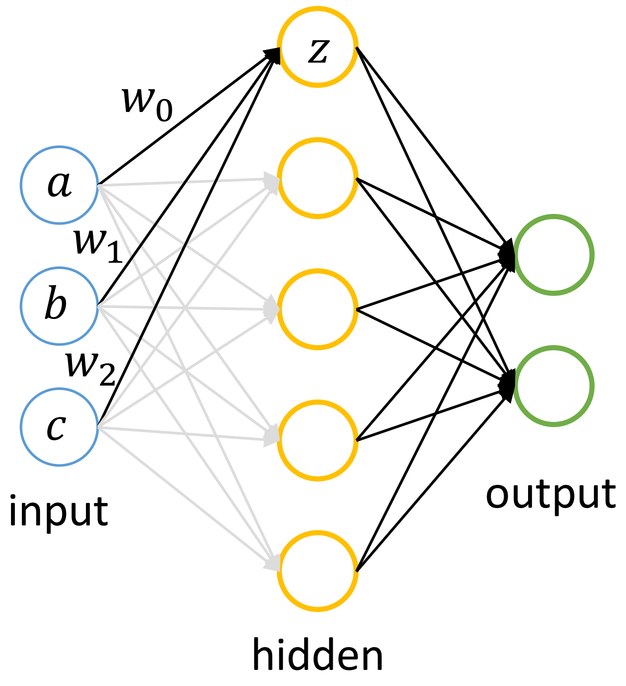
Neural Networks

- Neurons are connected to each other to define a parametric function.



Neural Networks

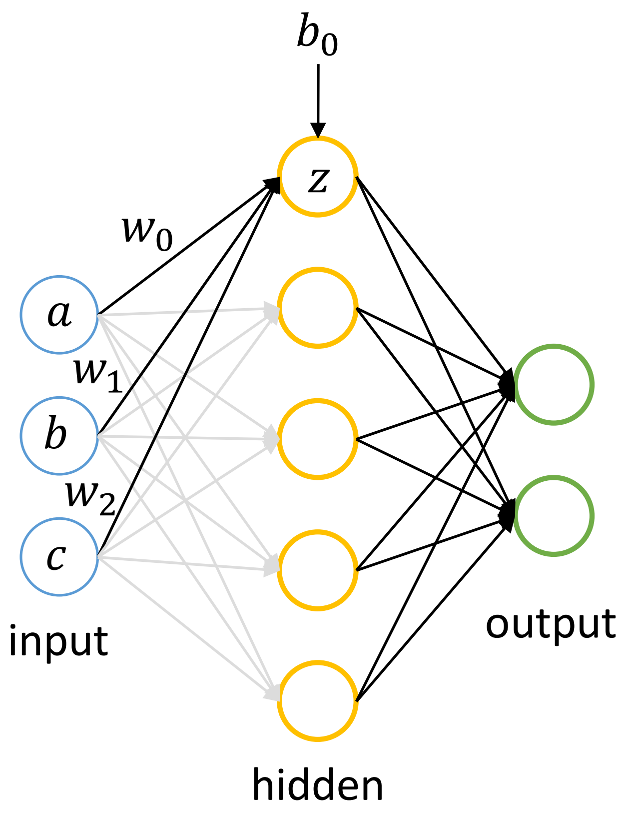
- The value input to each neuron is found by a linear combination of neuron values.



$$z = w_0 a + w_1 b + w_2 c$$

Neural Networks

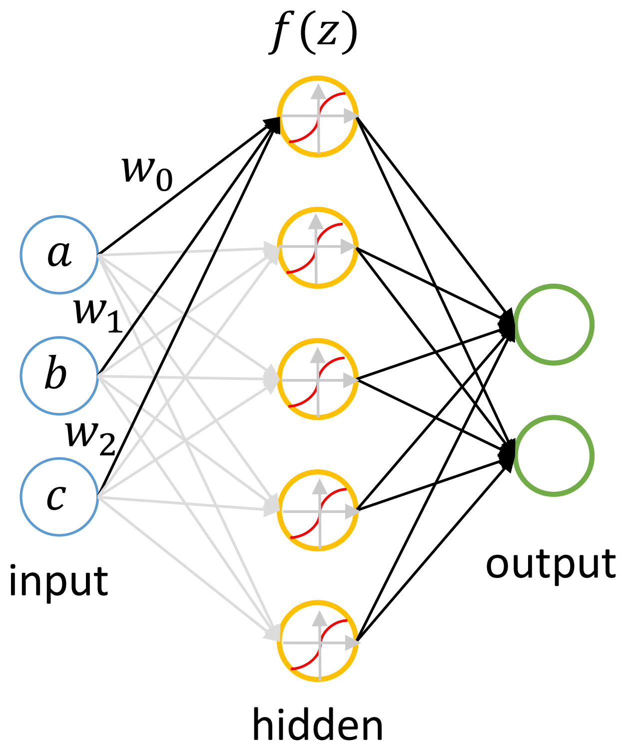
- You can also add a bias to each node



$$z = w_0 a + w_1 b + w_2 c + b_0$$

Neural Networks

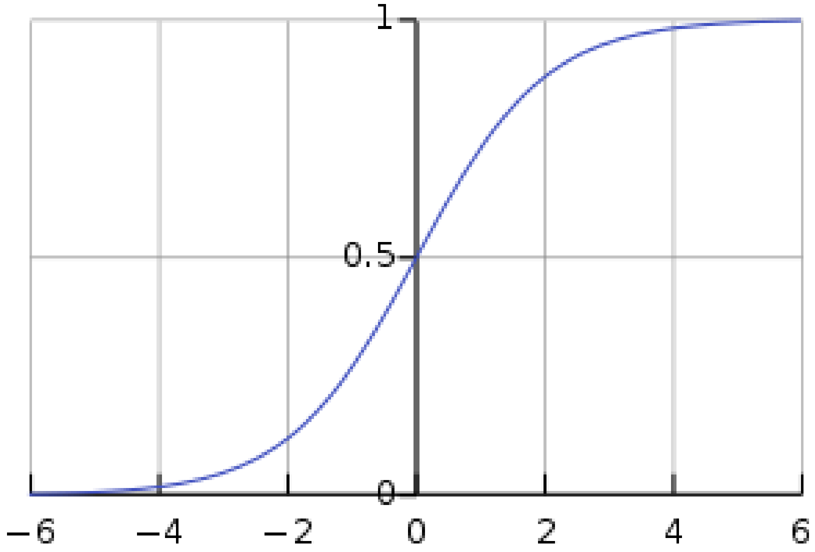
- Followed by a non-linear activation function.



$$z = w_0 a + w_1 b + w_2 c$$

Activation Function

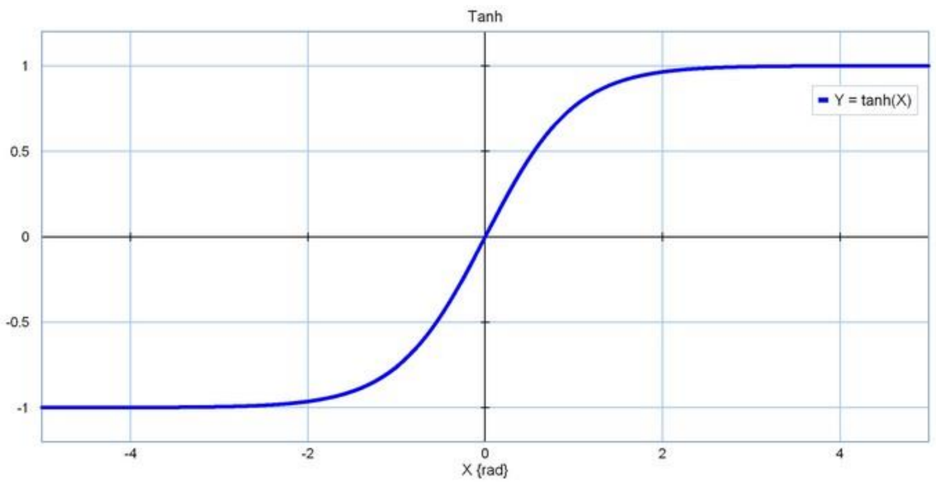
- Sigmoid



$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

Activation Function

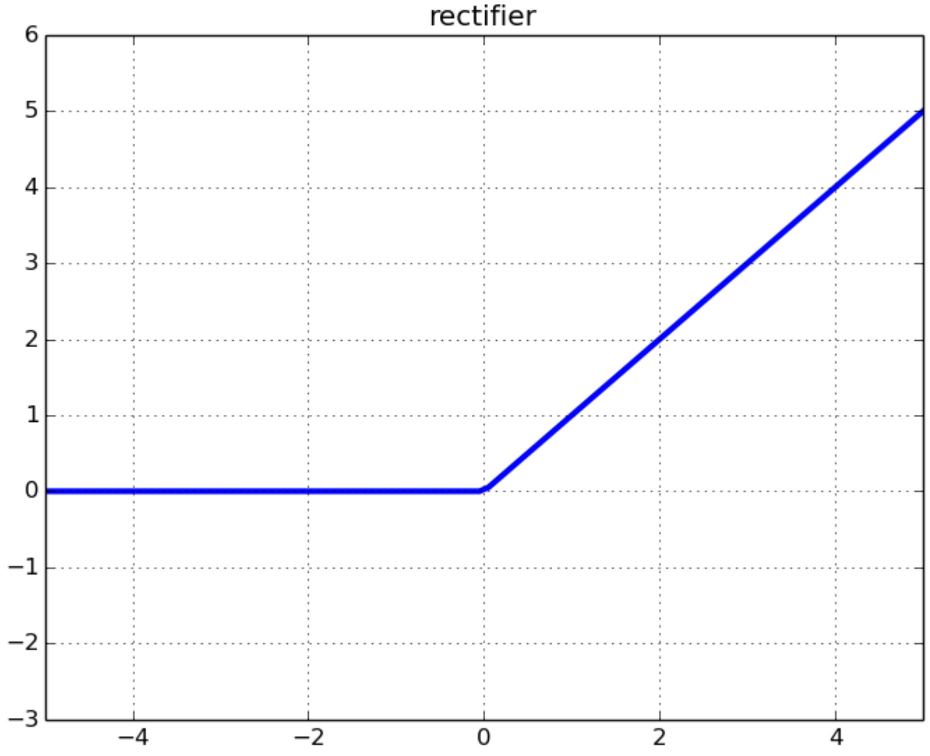
- Hyperbolic Tangent



$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} - e^{-x}}$$

Activation Function

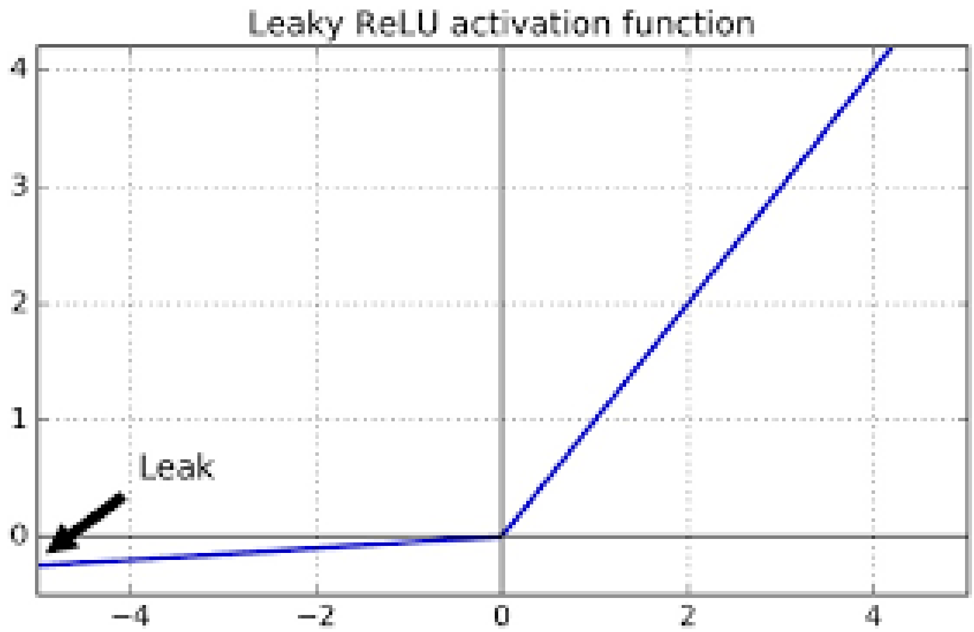
- Rectified Linear Unit (ReLU)



$$f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$$

Activation Function

- Leaky Rectified Linear Unit (ReLU)



$$f(x) = \begin{cases} x & for\ x > 0 \\ \alpha x & else \end{cases}$$

Training

$$Min \sum_{i=1}^{n} \mathcal{L}(f(x_i, W), y_i)$$

Training

$$Min \sum_{i=1}^{n} \boxed{\mathcal{L}}(f(x_i, W), y_i)$$

Loss function

Training

$$Min \sum_{i=1}^{n} \mathcal{L}(\boxed{f(x_i, W)}, y_i)$$

Output of neural network

Training

$$Min \sum_{i=1}^{n} \mathcal{L}(f(x_i, W), \boxed{y_i})$$

Expected result or ground truth

Training

- We use gradient descent to find the minimum

$$Min \sum_{i=1}^{n} \mathcal{L}(f(x_i, W), y_i)$$

Training

- We use gradient descent to find the minimum

$$Min \sum_{i=1}^{n} \mathcal{L}(f(x_i, W), y_i)$$

- We update the weights little by little based on partial derivatives

$$W_{t+1} = W_t - \boxed{\alpha} \nabla_w \mathcal{L}$$

Learning Rate

Training

- We use gradient descent to find the minimum

$$Min \sum_{i=1}^{n} \mathcal{L}(f(x_i, W), y_i)$$

- We update the weights little by little based on partial derivatives

$$W_{t+1} = W_t - \alpha \nabla_w \boxed{\mathcal{L}}$$

Defining a meaningful loss function is important

Training

- We have a set of images and we want to train a model on these images so that the model generalizes well.

Training

- We have a set of images and we want to train a model on these images so that the model generalizes well.

- It performs well on data sets that are not in the training set.

Training

- Having a data set, we split it into three groups:
  - Training set
  - Test set
  - Validation set

Why Three Sets? Why not only Training set and Test set?

Training

- You train your model using training set and fine tune the model (hyperparameters) using validation set. Each time you fine tune, a little bit of information leaks to the network. Although, you have not trained your model on validation set, the model overfits to the validation set.

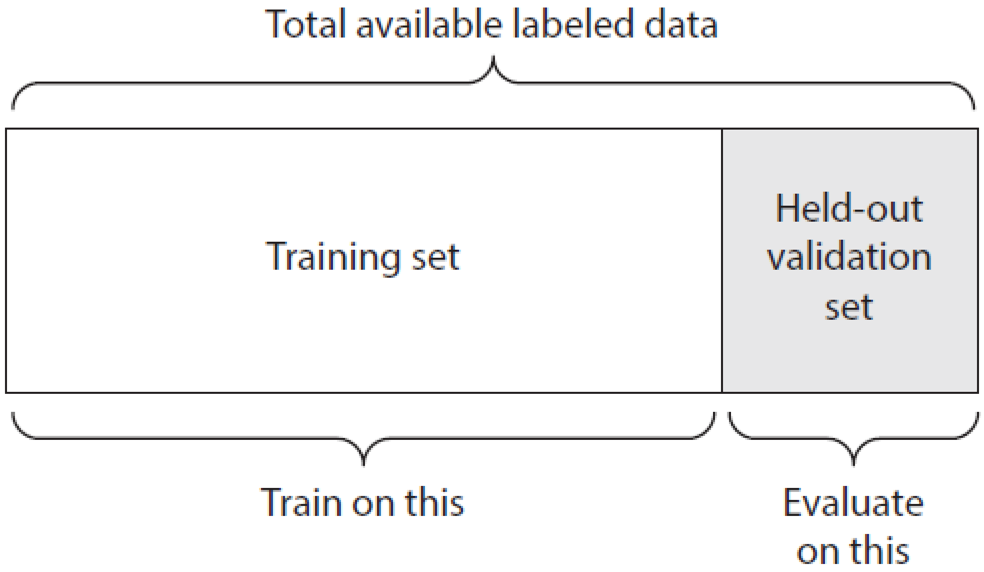
Training

- You train your model using training set and fine tune the model (hyperparameters) using validation set. Each time you fine tune, a little bit of information leaks to the network. Although, you have not trained your model on validation set, the model overfits to the validation set.

- You need an extra set to actually measure the performance of your network that has been never exposed to the network directly or indirectly.

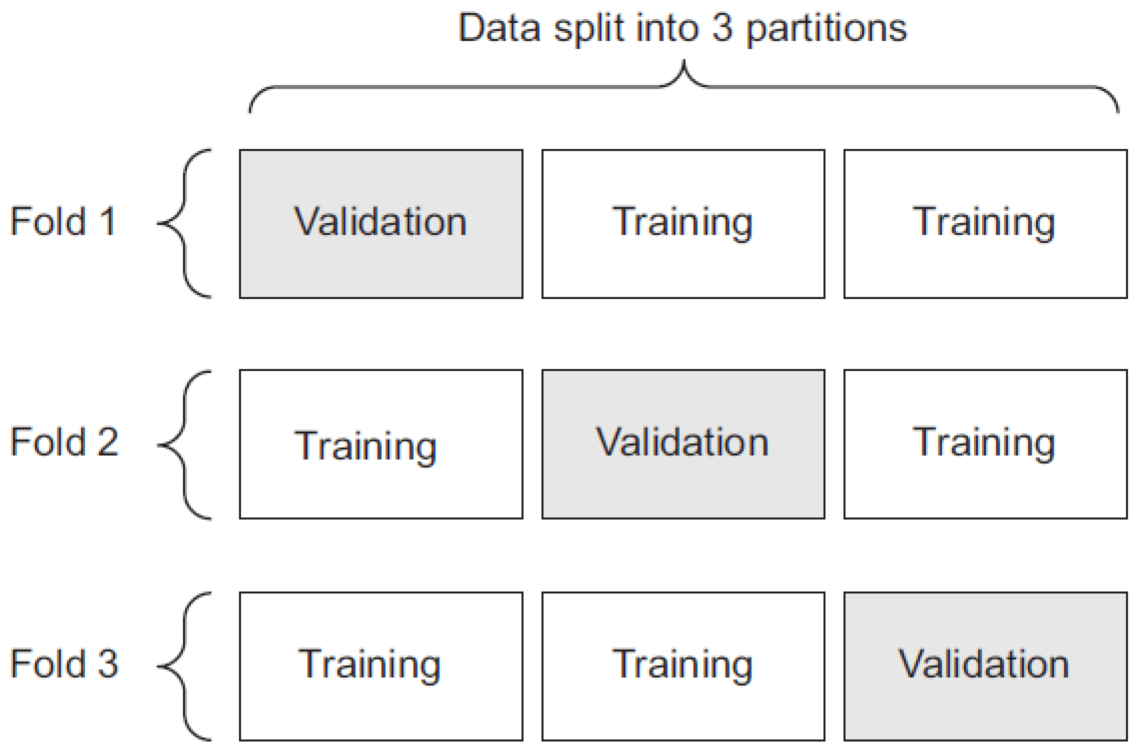
Validation

- Hold-out validation: you put aside a portion of your training data for validation.



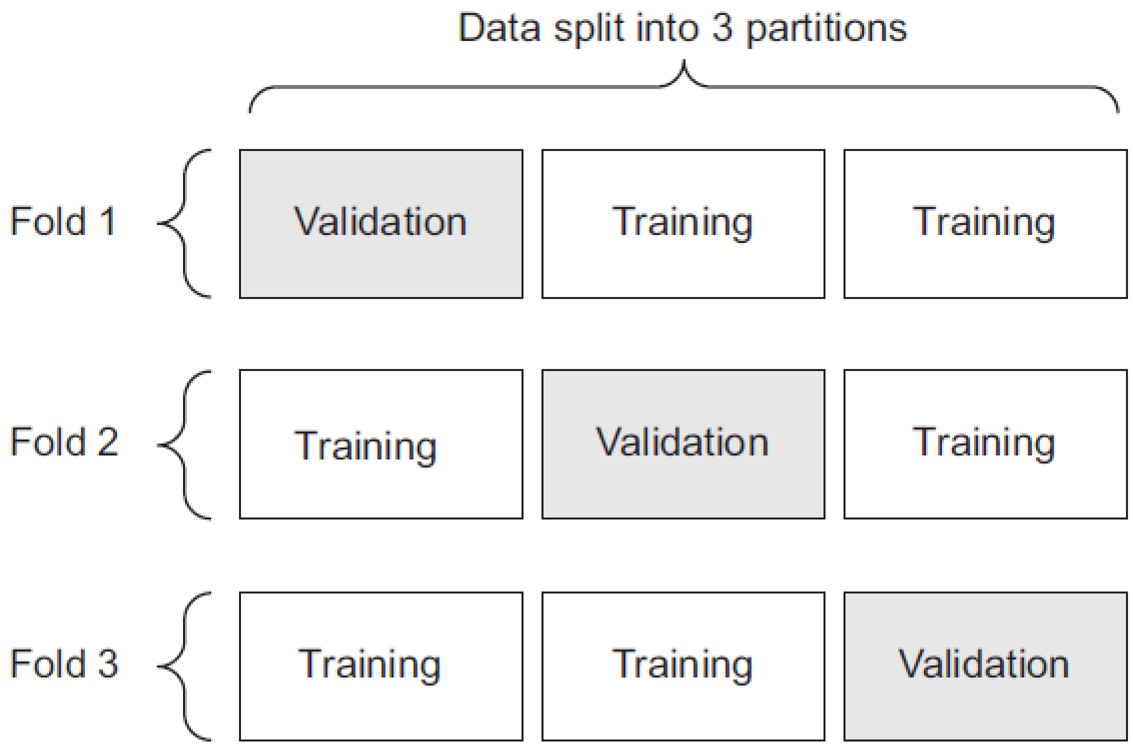
Validation

- K-Fold Validation: Split the data into k Partitions of equal size.



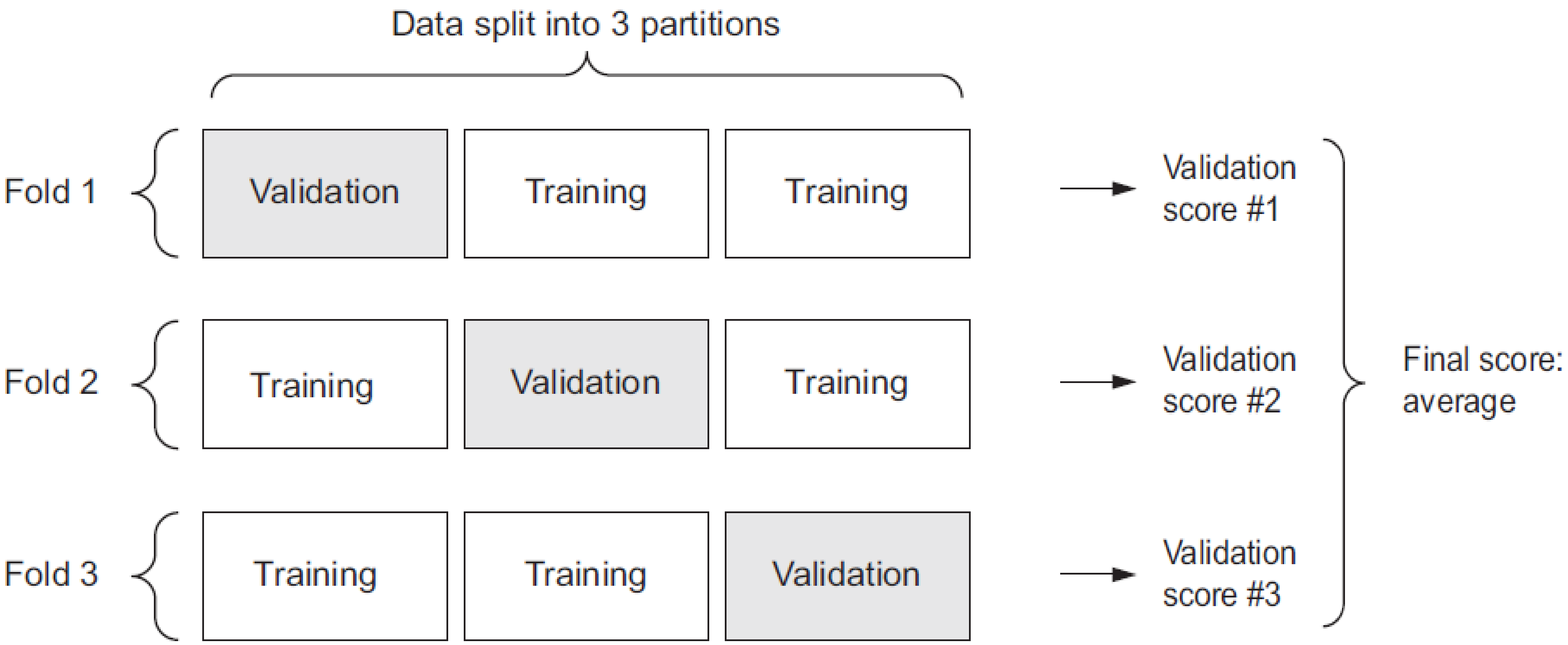
Validation

- K-Fold Validation: Validate on ith partition, train on the rest.



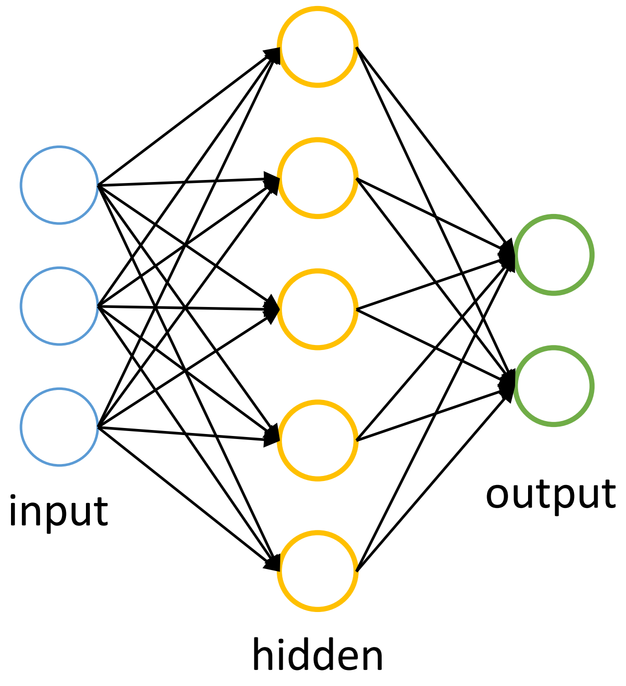
Validation

- K-Fold Validation: Final score is the average of all the scores.



Data Processing

- Vectorization: Neural networks usually accept vectors, you will always need to vectorize your input data.



Data Processing

- Data Normalization: You should avoid providing large data values or heterogeneous data values to your network.

Data Processing

- Data Normalization: You should avoid providing large data values or heterogeneous data values to your network.

- Large Values: You will end up having large gradient updates and the network does not converge.

Data Processing

- Data Normalization: You should avoid providing large data values or heterogeneous data values to your network.

- Large Values: You will end up having large gradient updates and the network does not converge.

- Heterogeneous Data: Some data have more effect on the network than the others.

Data Processing

- Rule of thumb:
  - Normalize each feature independently to have mean 0.
  - Normalize each feature independently to have standard deviation of 1.

Handling Missing Values

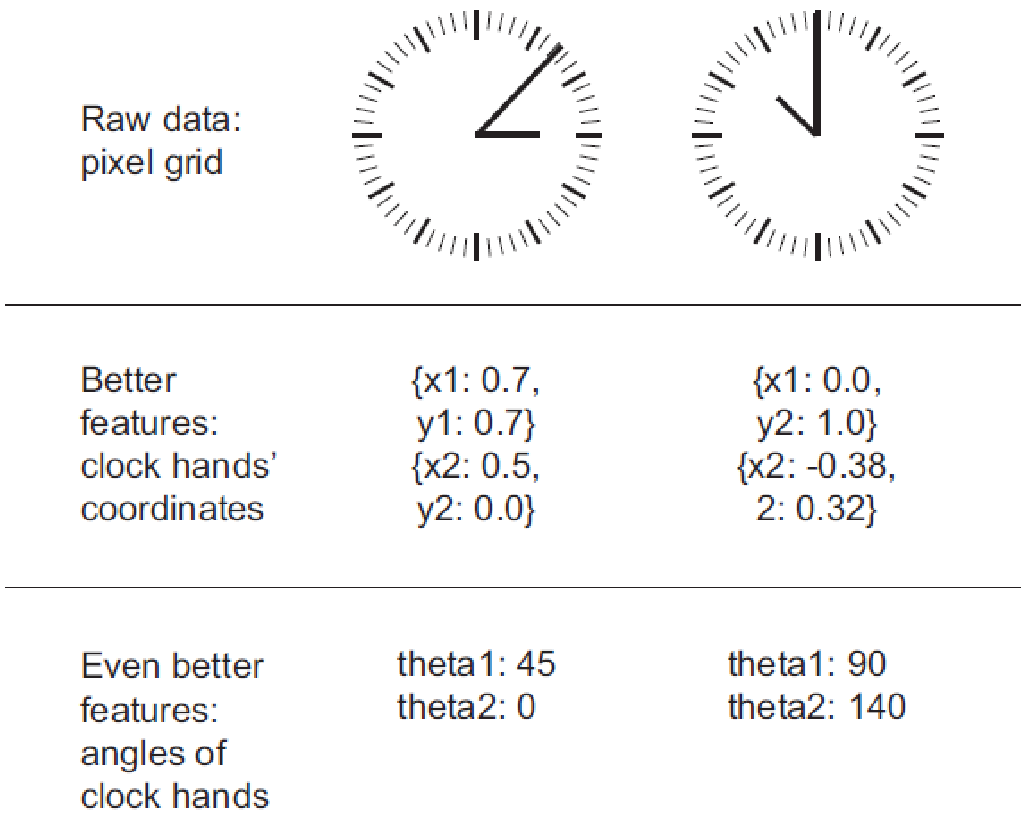
- If you have missing values in some features, default them to be zero. Network will learn to ignore such values.

Feature Engineering

- Try to provide data sets that are easier to understand by the network.

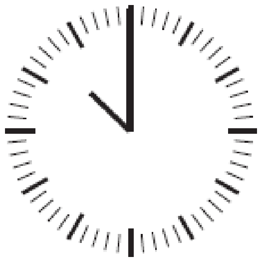
Feature Engineering

- Pixel values of a clock are a lot harder to understand for a network rather than a simple numeric representation.



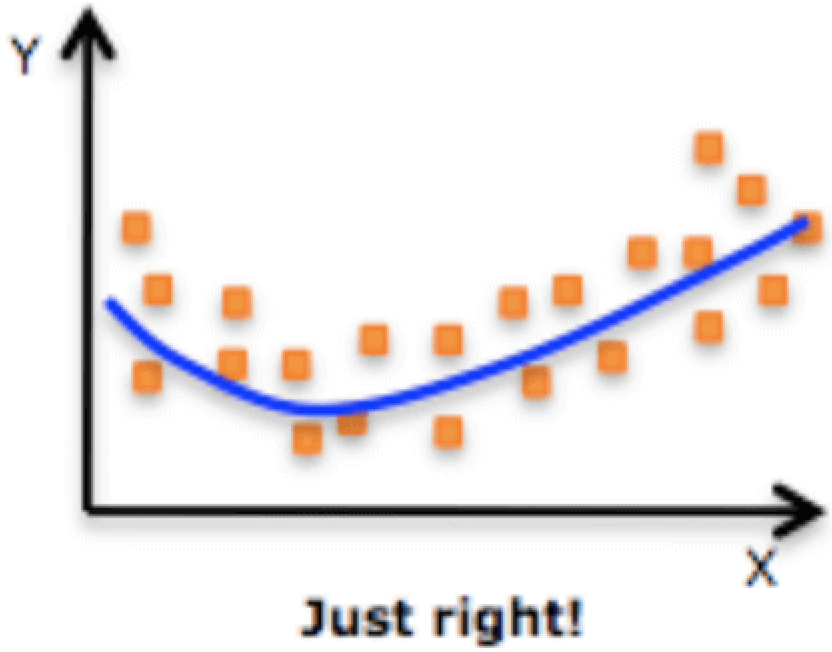
Feature Engineering

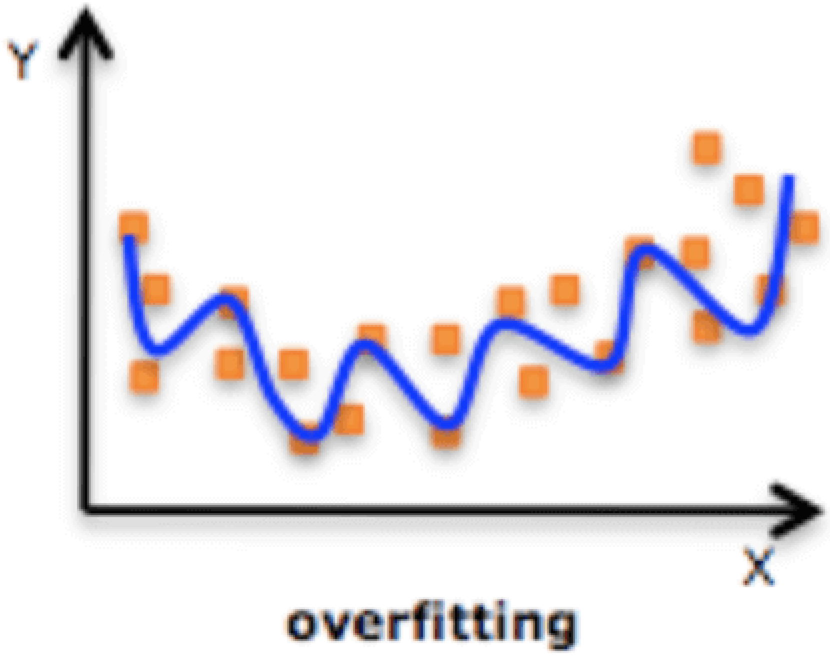
- With good features, you will need less data and simpler models.

Raw data: pixel grid		
Better features: clock hands' coordinates	{x1: 0.7, y1: 0.7} {x2: 0.5, y2: 0.0}	{x1: 0.0, y2: 1.0} {x2: -0.38, 2: 0.32}
Even better features: angles of clock hands	theta1: 45 theta2: 0	theta1: 90 theta2: 140

Overfitting and underfitting

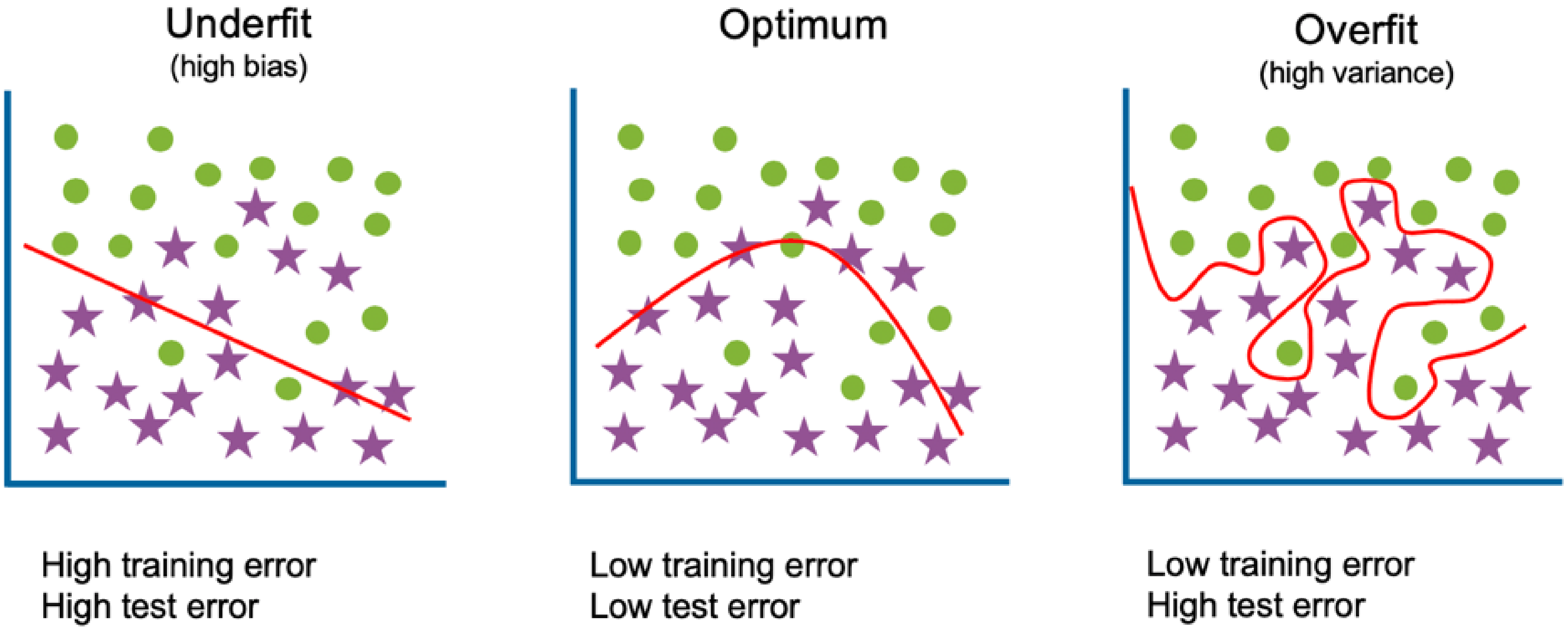
- Overfitting: network learns patterns and irregular behaviors of the training set that cannot be generalized to test/validation set. It is specific to the training set.





Overfitting and underfitting

- Underfitting: you train the model more (more epochs), your model is still getting better results (lower loss) both on the training and validation sets.



Overfitting Solution

- Underfitting is not usually a problem if the network is not too big and time consuming to train.

- How to solve the problem of overfitting?

Overfitting Solution

- More data: not easy to find and collect.

Overfitting Solution

- More data: not easy to find and collect.

- Smaller network.

Overfitting Solution

- More data: not easy to find and collect.
- Smaller network.
- Add regularization.

Overfitting Solution

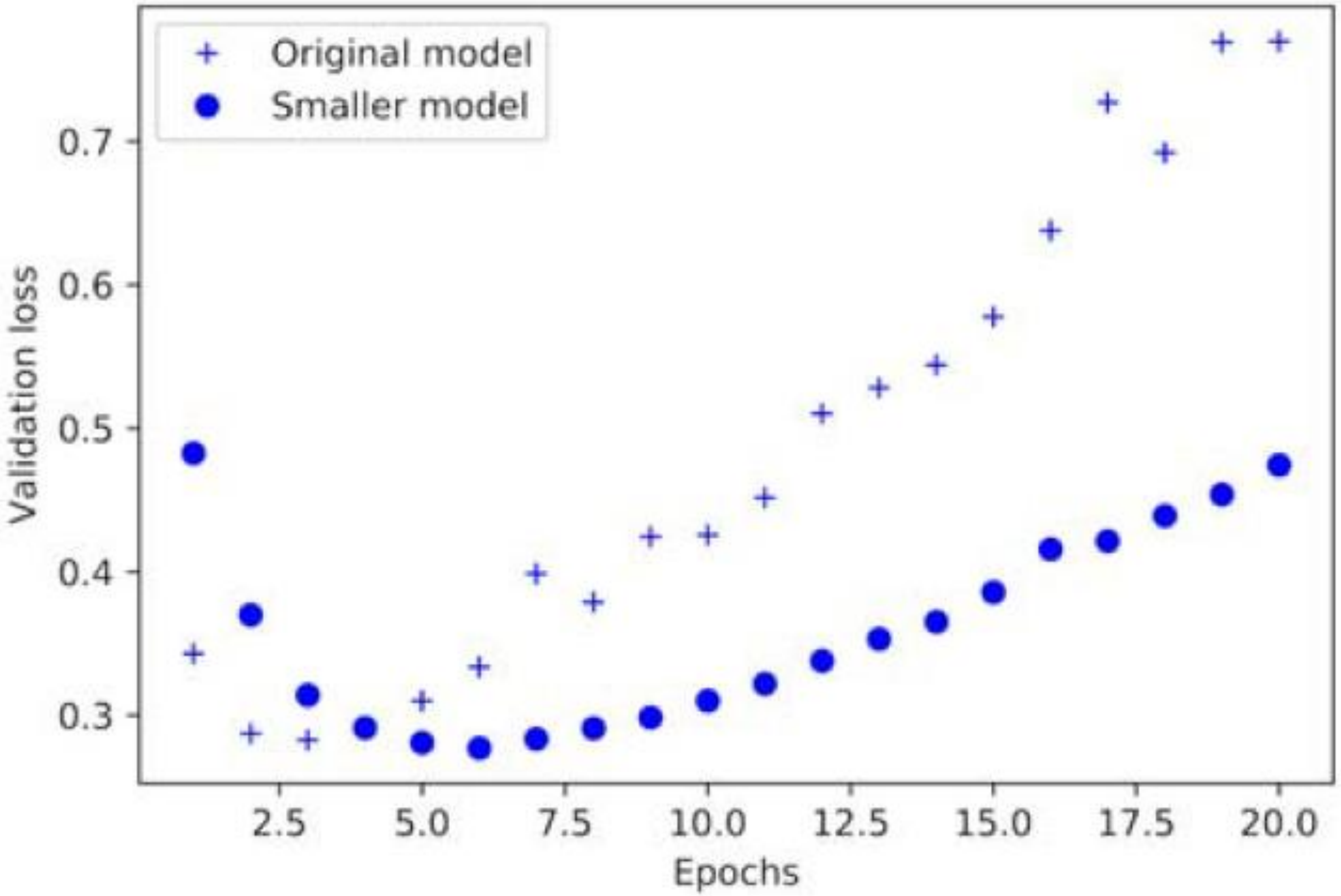
- More data: not easy to find and collect.
- Smaller network.
- Add regularization.
- Add dropout.

Smaller Network

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

```
model = models.Sequential()
model.add(layers.Dense(4, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(4, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

Smaller Network



Smaller Network

- By forcing values of a network to be smaller, you make the distribution more regular.

Weight Regularization

- *Large weights tend to cause sharp transitions in the node functions and thus large changes in output for small changes in the inputs.*

Weight Regularization

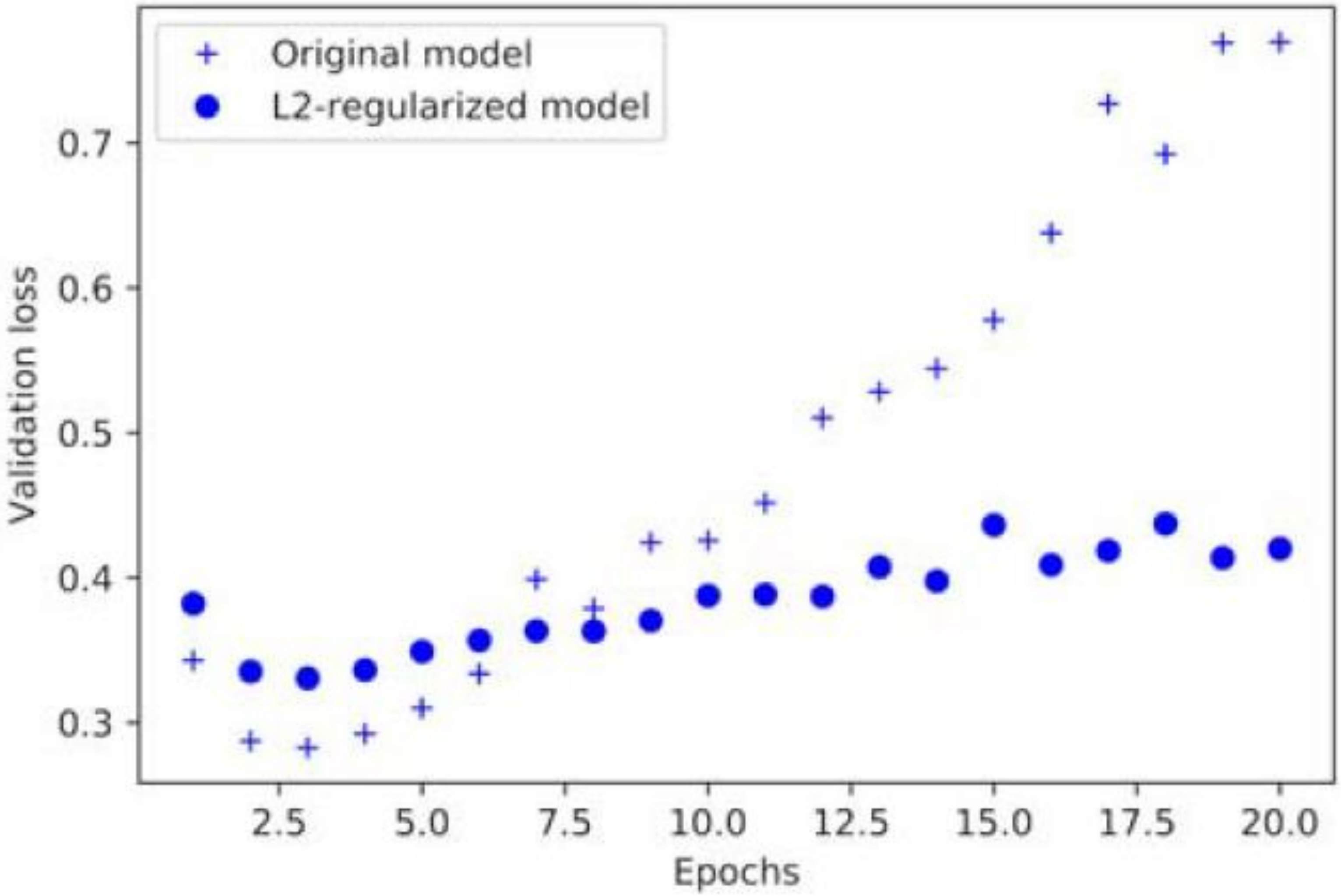
- L1 Regularization: The cost added is proportional to the absolute value of the weight coefficients. (L1 norm).

- L2 Regularization: The cost is added proportional to the square value of the weight coefficients. (AKA., weight decay).

Weight Regularization

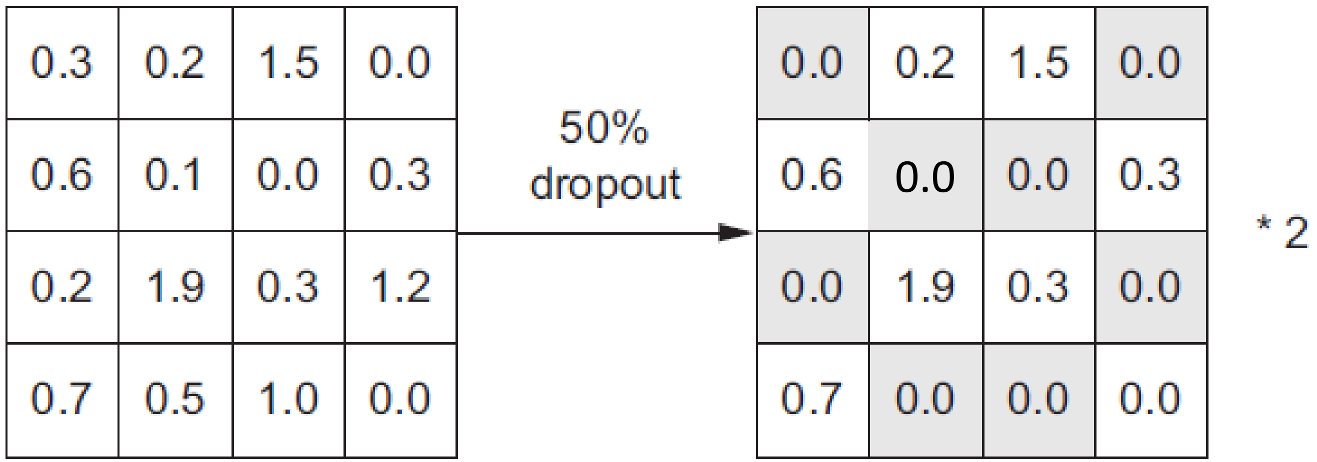
```
from keras import regularizers

model = models.Sequential()
model.add(layers.Dense(16, kernel_regularizer=regularizers.l2(0.001),
                       activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, kernel_regularizer=regularizers.l2(0.001),
                       activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

Weight Regularization



Dropout

- Remove X-percent of the data (dropout rate)
- Compromise by a scalar proportional to dropout rate.

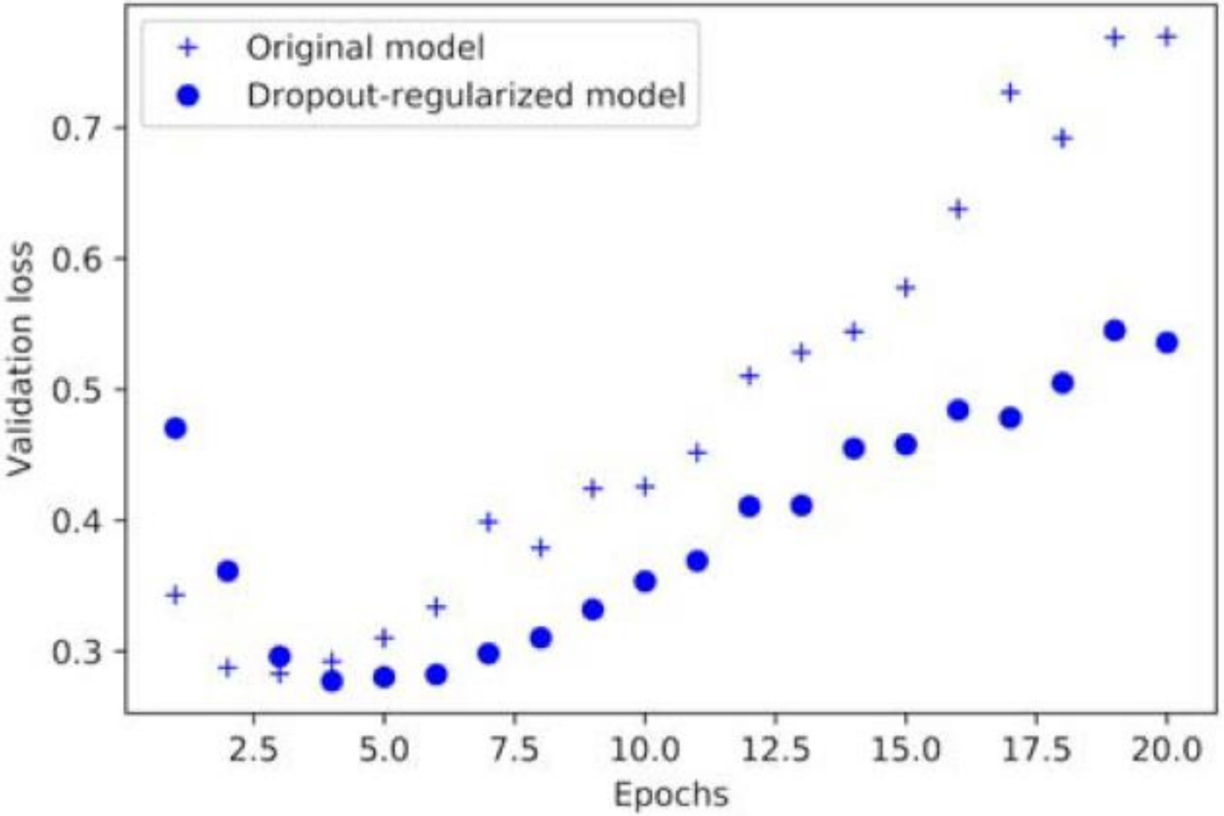


Dropout

```
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

Dropout rate

Dropout



Original model
Dropout-regularized model
Validation loss
0.7
0.6
0.5
0.4
0.3
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs

Summary

- What is your problem? What is your data? Why deep learning?
    - Image Classification

Summary

- What is your problem? What is your data? Why deep learning?
  - Image Classification

- How do you evaluate? (what is your loss?)
  - Miss-classification

Summary

- What is your problem? What is your data? Why deep learning?
  - Image Classification

- How do you evaluate? (what is your loss?)
  - Miss-classification
- K-fold Validation?

Summary

- What is your problem? What is your data? Why deep learning?
  - Image Classification

- How do you evaluate? (what is your loss?)
  - Miss-classification
- K-fold Validation?
- Make/choose a simple base-line.

Summary

- Make a model better than base-line.

Summary

- Make a model better than base-line.
- Develop a model that overfits.
  - Make sure you can learn the training data set well and you can converge.

Summary

- Make a model better than base-line.

- Develop a model that overfits.
  - Make sure you can learn the training data set well and you can converge.
- Use drop-out, weight regularization, change hyperparameters to get the best results.