

Statistical Modelling with Data

May 23 – June 02, 2023

Instructor: Qing (Leah) Li, Ph.D. Candidate at Cumming School of Medicine

qing.li2@ucalgary.ca

Thank you Dr. Thuntida Ngamkham for contributing the contents

Thank you Dr. Qingrun Zhang and Dr. Quan Long for contributing some slides

Statistical Modelling with Data

- Topic 1: Statistical Modelling
 - Lecture 1: First-order models with quantitative independent variables
- Topic 2: Statistical Modelling with interactions (Assignment 1)
 - Lecture 2: Interaction effects, quantitative and qualitative variables
 - Lecture 3: Interaction effects and second-order models
- Topic 3: Statistical Model selection (Assignment 2)
 - Lecture 4: Model selection: Stepwise regression, Forward selection and Backward Elimination
 - Lecture 5: Model selection: Evaluate the reliability of the model chosen
- Topic 4: Statistical model diagnostics
 - Lecture 6: Multiple regression diagnostics: verify linearity, independence, equal variance assumptions and normality assumptions.
 - Lecture 7: Multiple regression diagnostics: identify multicollinearity and outliers and data transformation.
- Topic 5: Transfer learning
 - Lecture 8: Deep learning basics
 - Lecture 9: Transfer-learning (Bonus): standing on the shoulders of giants.

Statistical Modelling with Data

- Topic 1: Statistical Modelling
 - Lecture 1: First-order models with quantitative independent variables
- Topic 2: Statistical Modelling with interactions (Assignment 1)
 - Lecture 2: Interaction effects, quantitative and qualitative variables
 - Lecture 3: Interaction effects and second-order models
- Topic 3: Statistical Model selection (Assignment 2)
 - Lecture 4: Model selection: Stepwise regression, Forward selection and Backward Elimination
 - Lecture 5: Model selection: Evaluate the reliability of the model chosen
- Topic 4: Statistical model diagnostics
 - Lecture 6: Multiple regression diagnostics: verify linearity, independence, equal variance assumptions, and normality assumptions.
 - Lecture 7: Multiple regression diagnostics: identify multicollinearity and outliers and data transformation.
- Topic 5: Transfer learning
 - Lecture 8: Deep learning basics
 - Lecture 9: Deep learning advances: Transfer-learning (Bonus).

Statistical Modelling with Data

Learning Outcomes: At the end of the course, participants will be able to

1. Model the multiple linear relationships between a response variable (Y) and all explanatory variables (both categorical and numerical variables) with interaction terms. Interpret model parameter estimates, construct confidence intervals for regression coefficients, evaluate model fits, and visualize correlations between a response variable (Y) and all explanatory variables (X) by graphs (scatter plot, residual plot) to assess model validity.
2. Predict the response variable at a certain level of the explanatory variables once the fit model exists.
3. Implement R-software and analyze statistical results for biomedical and other data.

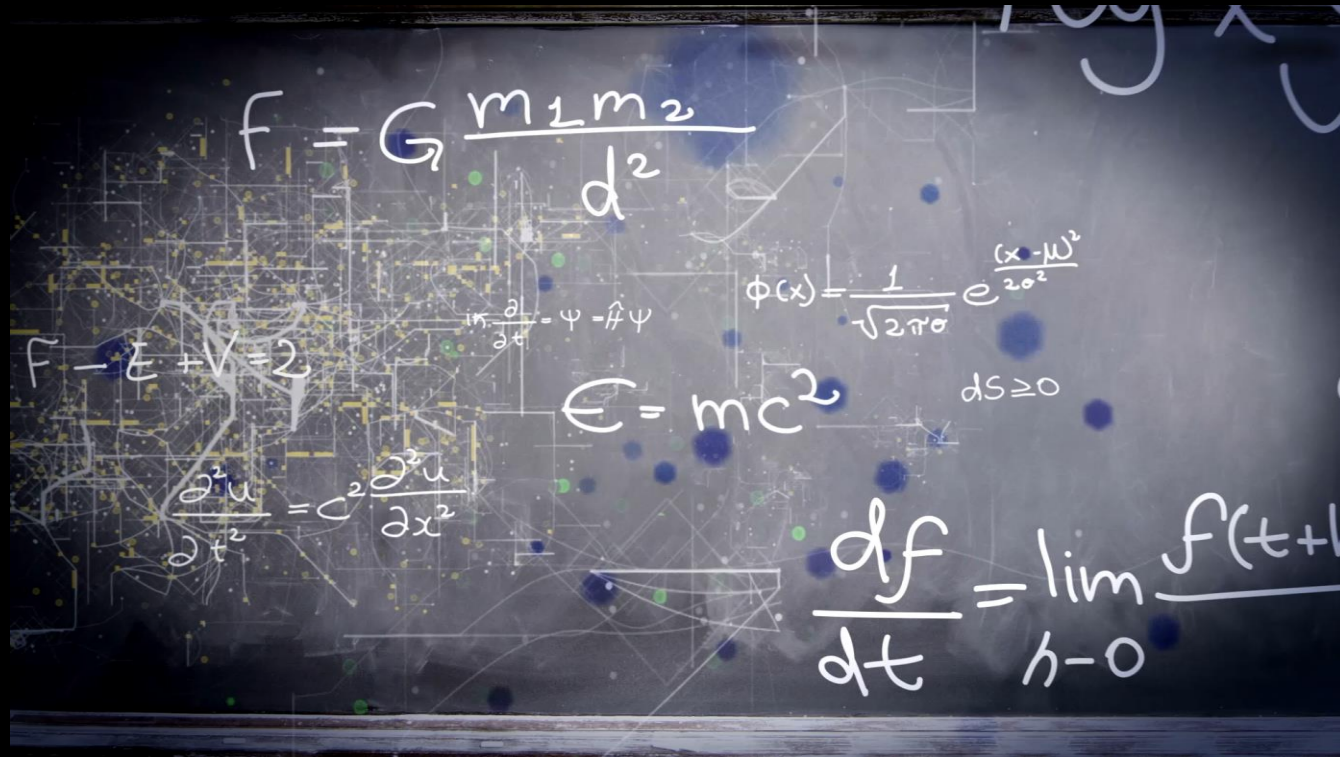
• **Evaluations**

1. Assignments will be posted on Slack (our communication tool with students).
2. Students must attend 70% (6/9) of the sessions in order to receive the certificate and are encouraged to work on the assignments progressively throughout the course as the relevant material is covered.

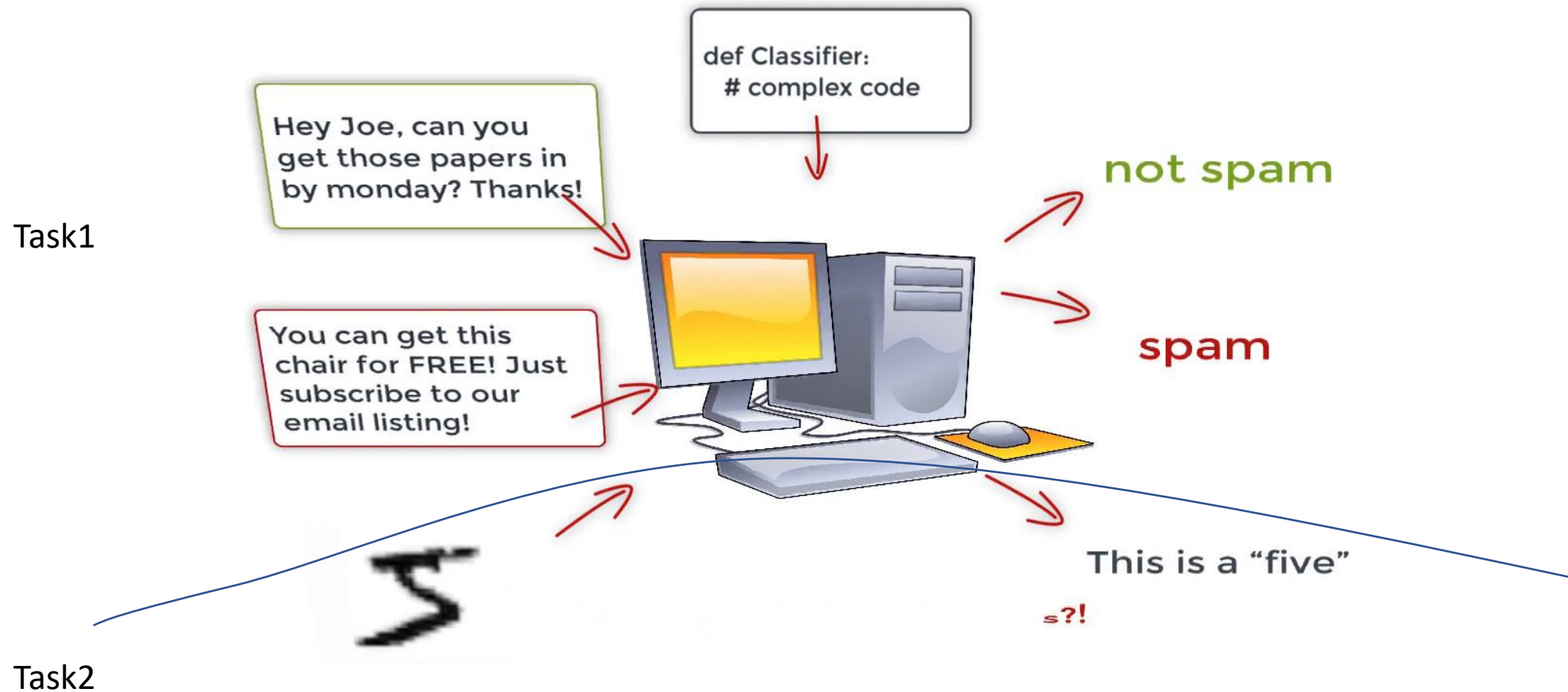
Statistical Modelling with Data

- Supportive materials
 - Lectures slides (2023)
 - R code scripts (2023)
 - PDF (dated 2022)
 - Two Assignments (dated 2022)
- Slack channels
 - Recoding videos
 - Exercises
 - Course-documents

Lecture 8: Deep learning basics



An example of machine learning: classification

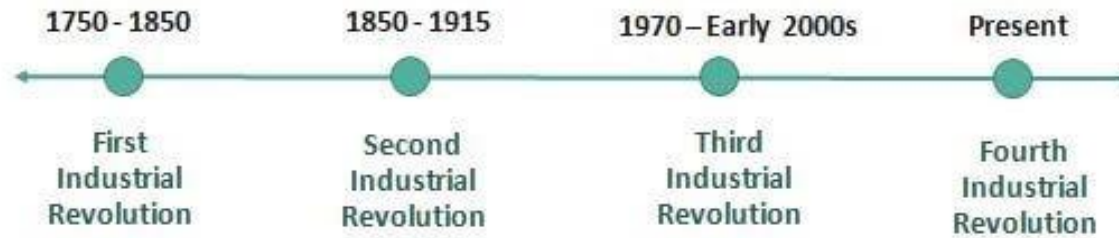


Outlines

1. What is deep learning and why?
2. Build a deep learning model:
 - Neural network
 - Activation function
 - Loss function
3. TensorFlow
4. Datasets
 - Split data into training and test dataset
5. Model training and evaluation
 - Training
 - Evaluation Metrics

Human intelligence

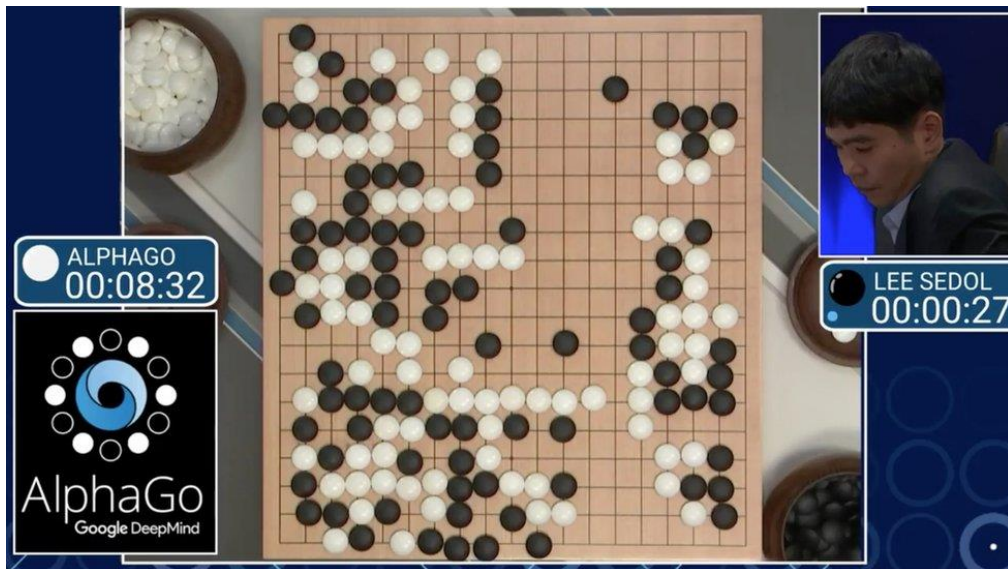
Industrial Revolution Timeline



Artificial intelligence

Artificial intelligence: Google's AlphaGo beats Go master Lee Se-dol

🕒 12 March 2016



ChatGPT



Developer(s) OpenAI

Initial release November 30, 2022; 6 months ago

Stable release May 24, 2023; 8 days ago^[1]

ChatGPT is an artificial intelligence (AI) chatbot developed by OpenAI and released in November 2022. The name "ChatGPT" combines "Chat", referring to its chatbot functionality, and "GPT", which stands for **Generative Pre-trained Transformer**, a type of large language model (LLM) (Wiki).

1. What is deep learning and why?



Q: Human brain is full of intelligence. Can we build machines that mimic human brain's activities so that they can have artificial intelligence?

Artificial Intelligence

Any technique which enables computers to mimic human behavior.

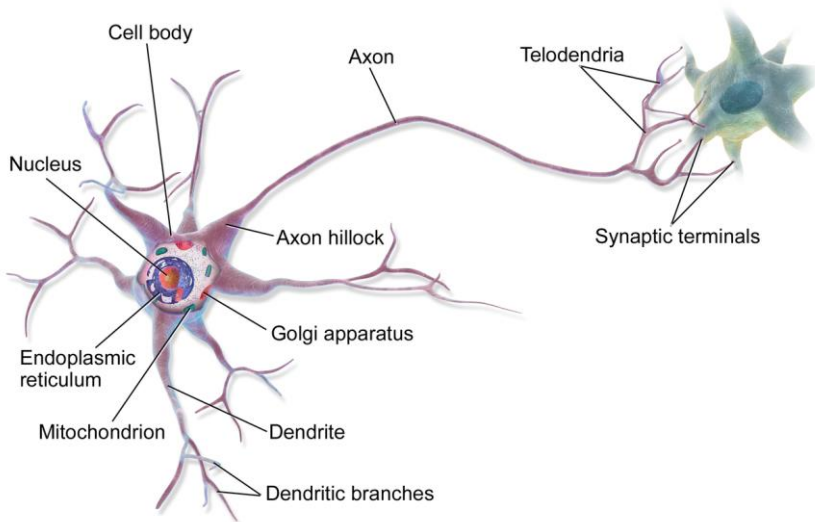
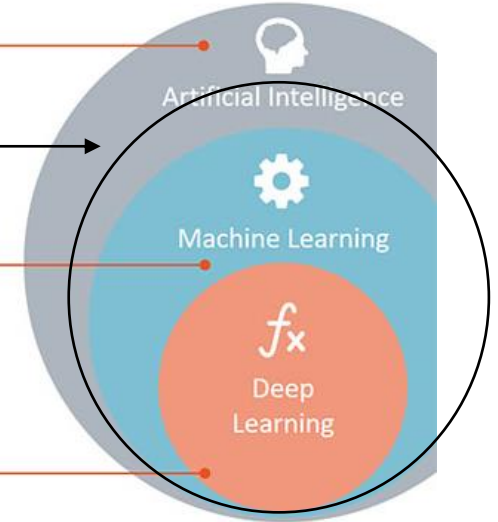
Statistical modelling

Machine Learning

Subset of AI techniques which use statistical methods to enable machines to improve with experiences.

Deep Learning

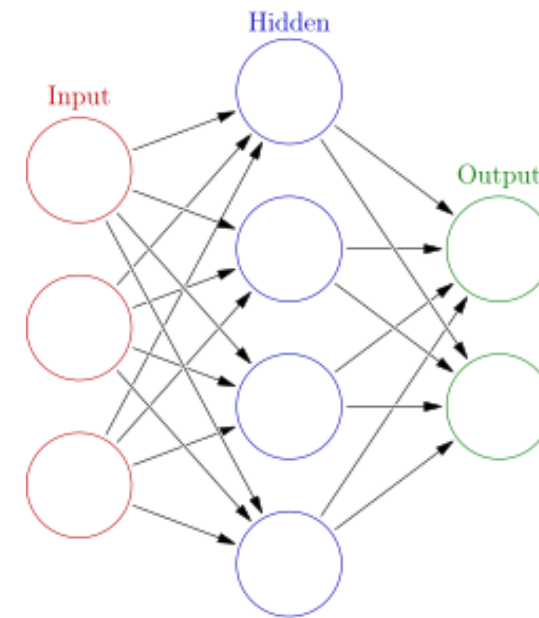
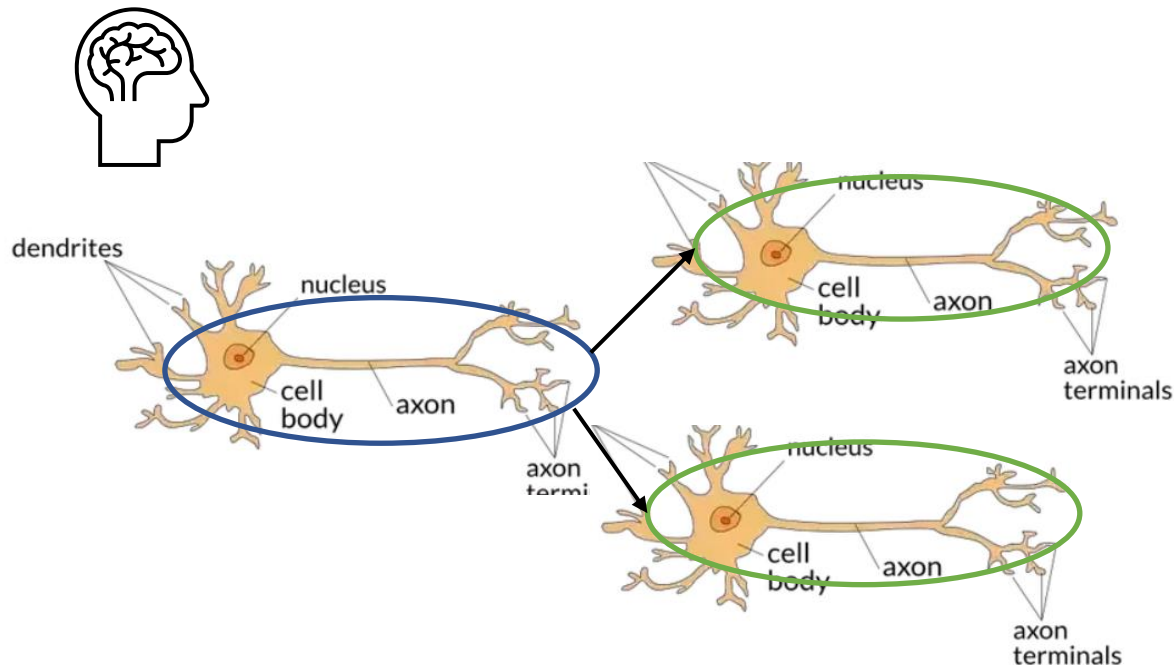
Subset of ML which make the computation of multi-layer neural networks feasible.



AI is the broader concept of developing intelligent systems, while statistical modeling, machine learning, and deep learning are **specific approaches** within AI. Statistical modeling focuses on analyzing data using **statistical techniques**, machine learning algorithms enable computers to learn from data and make predictions, and deep learning involves training **deep neural networks** to learn complex patterns and representations from data.

2. Build a deep learning model

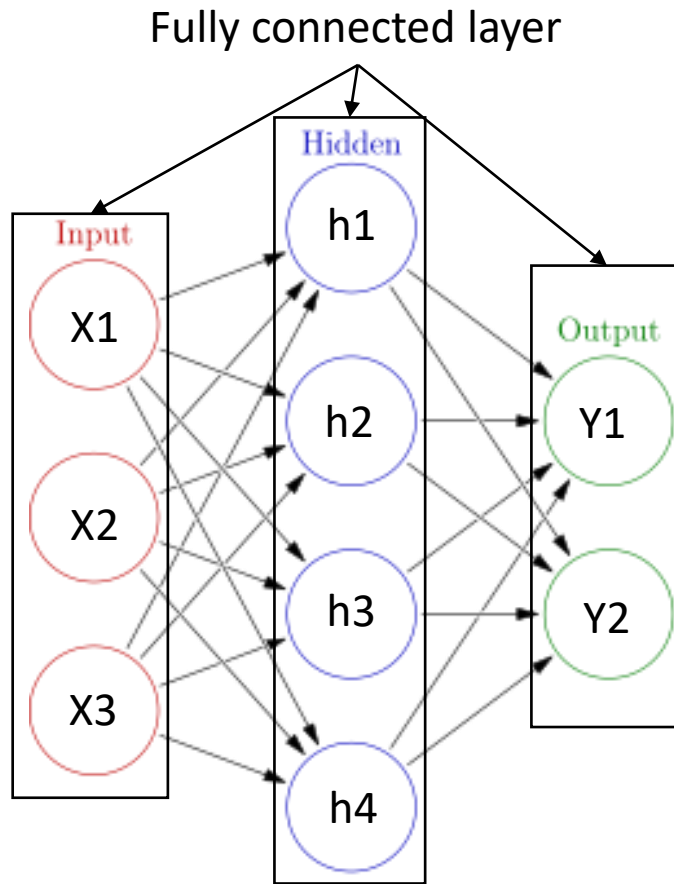
- Neural network (NN), are computing systems inspired by the biological neural networks that constitute animal brains (Wiki). The idea here is we want our network can perform tasks that human brain can do.
- The artificial general intelligence (AGI) is a type of hypothetical intelligent **agent**. The AGI concept is that it can learn to accomplish any intellectual task that human beings or other animals can perform.
- Human brain can perform way more complicated tasks than any machines.



Neural network (NN)

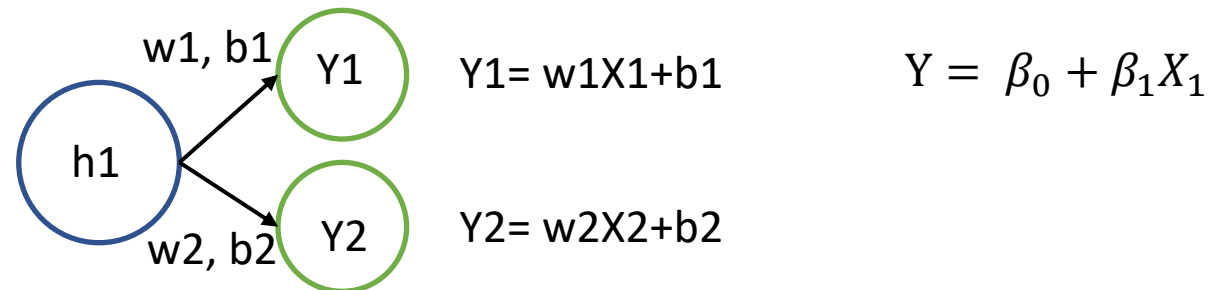
$$Y_1 = \omega_0 + \omega_1 X_1 + \omega_2 X_2 + \omega_3 X_3 + \varepsilon$$

$$Y_2 = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$$



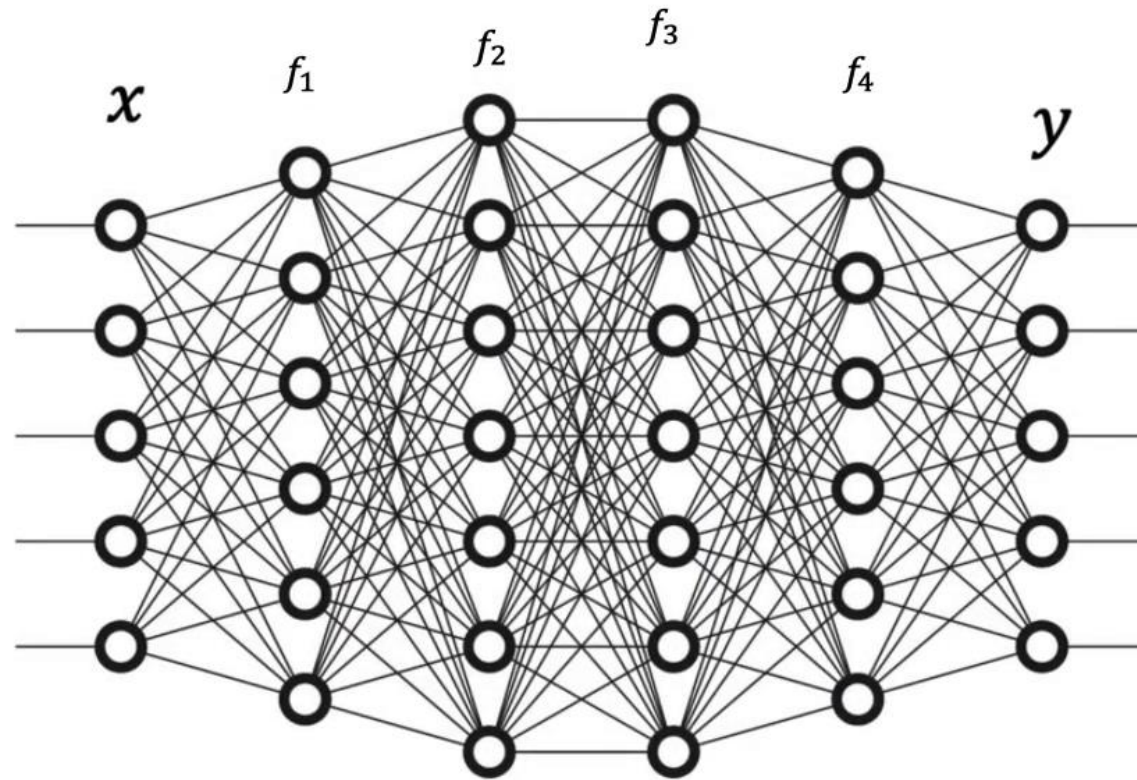
Fully connected network

- Node: mimic one human neuron
- Layer: one column of nodes. No connection between nodes in the same layer. The node in one layer is connected with nodes in the previous and next layers. If all nodes in one layer is fully connected with nodes in any layer, it's called fully connected layer. If a network consists only fully connected layers, it called fully connected neural network.
- Weights (w) and bias (b): represented by line arrows
- We aim to estimate values for all weights and bias, like what we did in statistical modelling.



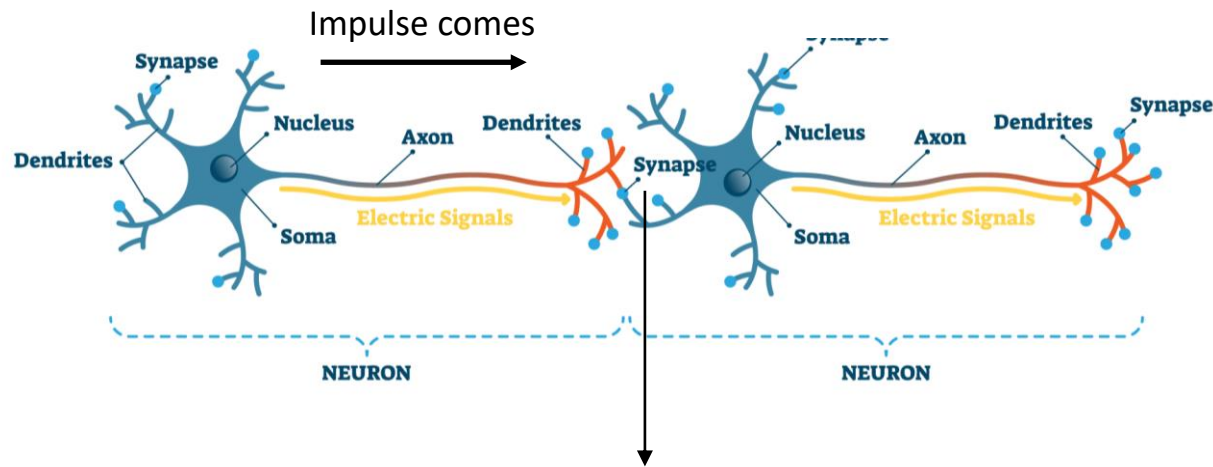
How many w and b in this fully connected network?

Deep Neural Network (DNN)

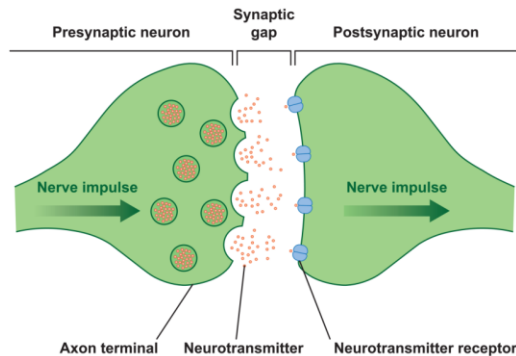


Source: Machine Learning, Deep Learning, Ai. Whats the difference (Datanami, 2017)

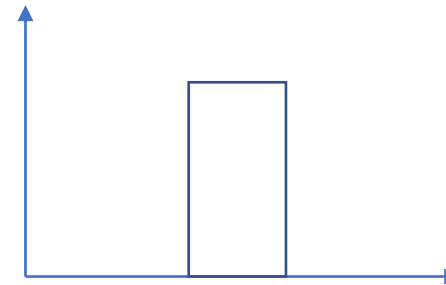
Activation function



Synaptic Transmission

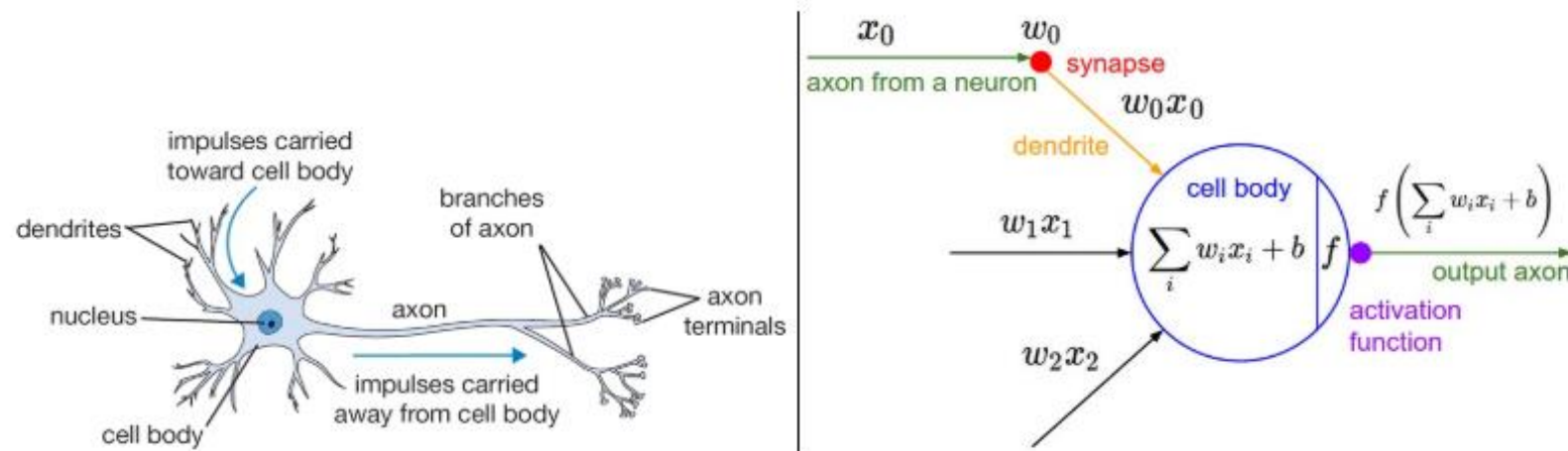


- Not all impulse will be passed from the front neuron to the back one. If the front neuron passed impulse successfully, then the back one will **fire** (activate) and continue to transfer impulse.
- We can use a function named “activation function” to describe the phenomenon whether the next neuron will be fired (active) or not.



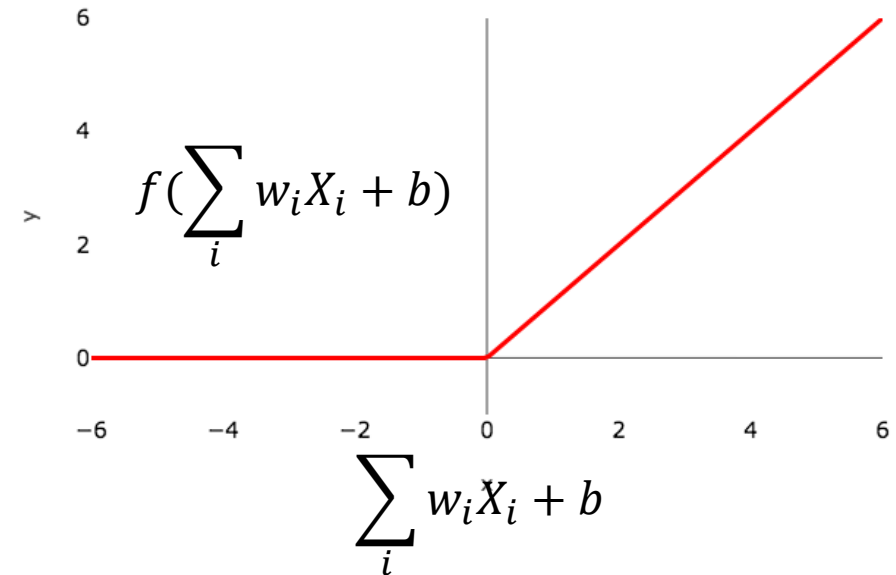
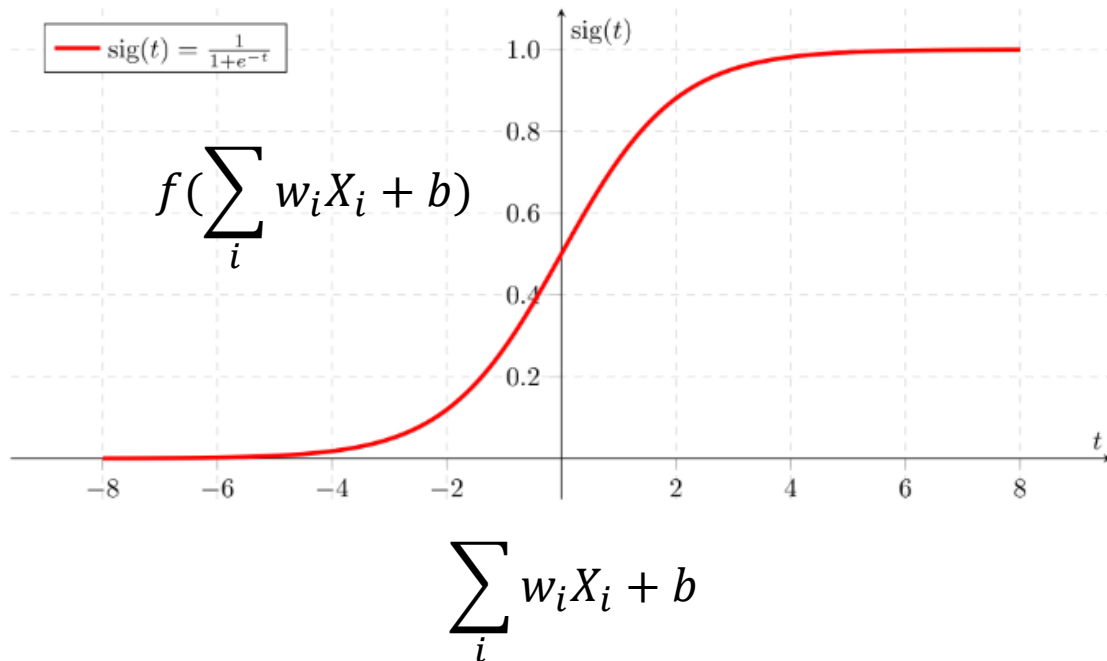
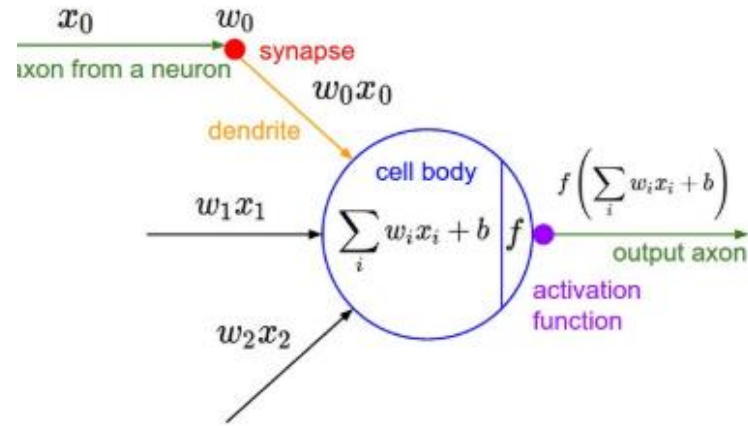
Activation function

- That is exactly what an activation function does in NN as well. It takes in the output signal from the previous cell and converts it into some form that can be taken as input to the next cell.
- Why activation function?
 - Apart from the biological similarity that was discussed earlier, they also help in keeping the value of the output from the neuron restricted to a certain limit as per our requirement.
 - The most important feature in an activation function is its ability to add non-linearity into a neural network.



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

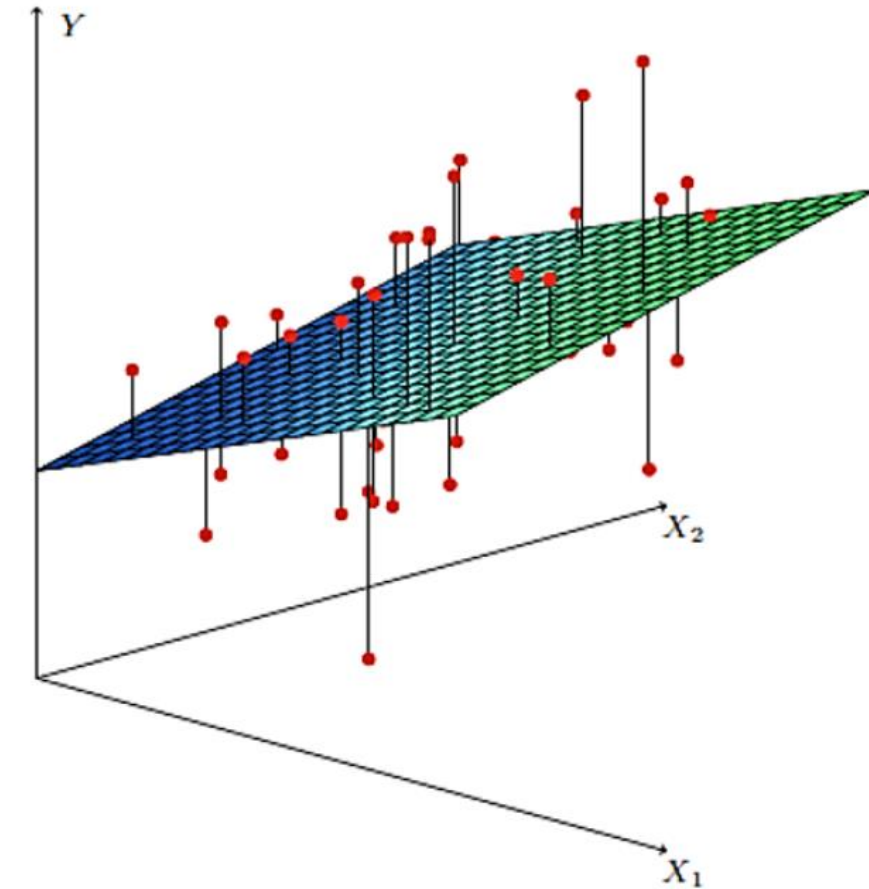
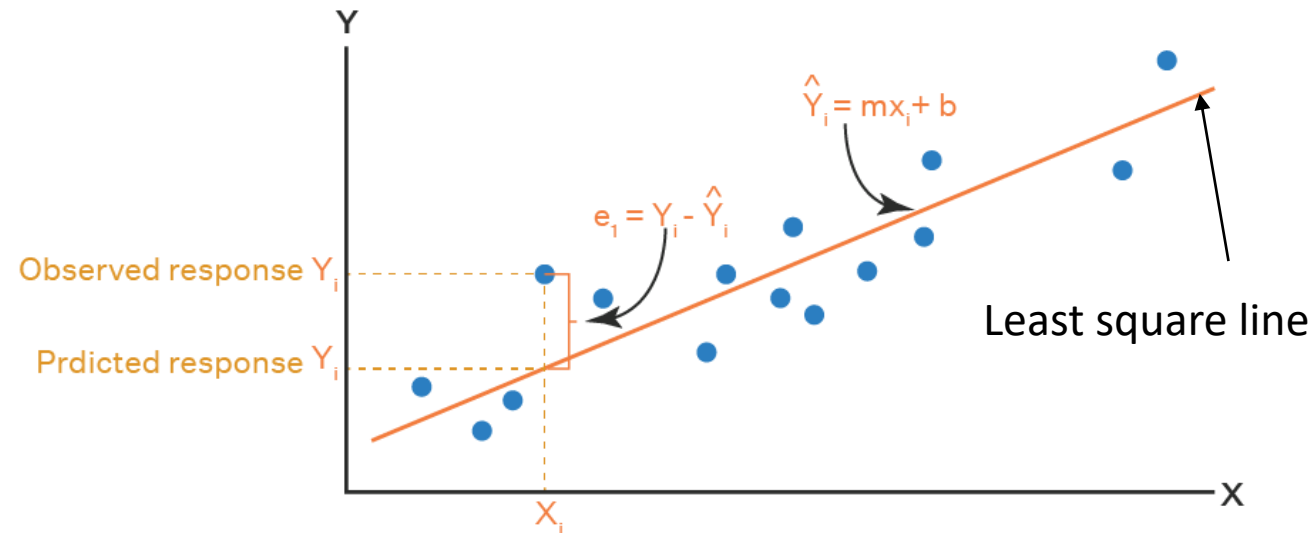
Activation functions examples



Loss function

The least squares method is a statistical procedure to find the best fit for a set of data points by minimizing the sum of the offsets or residuals of points from the plotted curve.

Parameter estimation is the process of computing a model's parameter values from measured data.



The least squares estimates (LSE) $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_p$ are obtained by minimizing the sum of the squared residuals:

$$SSE = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (\hat{\beta}_0 + \hat{\beta}_1 X_{1i} + \hat{\beta}_2 X_{2i} + \dots + \hat{\beta}_p X_{pi}))^2 \quad \leftarrow \text{Loss function as it's a function of estimated betas}$$

Loss function

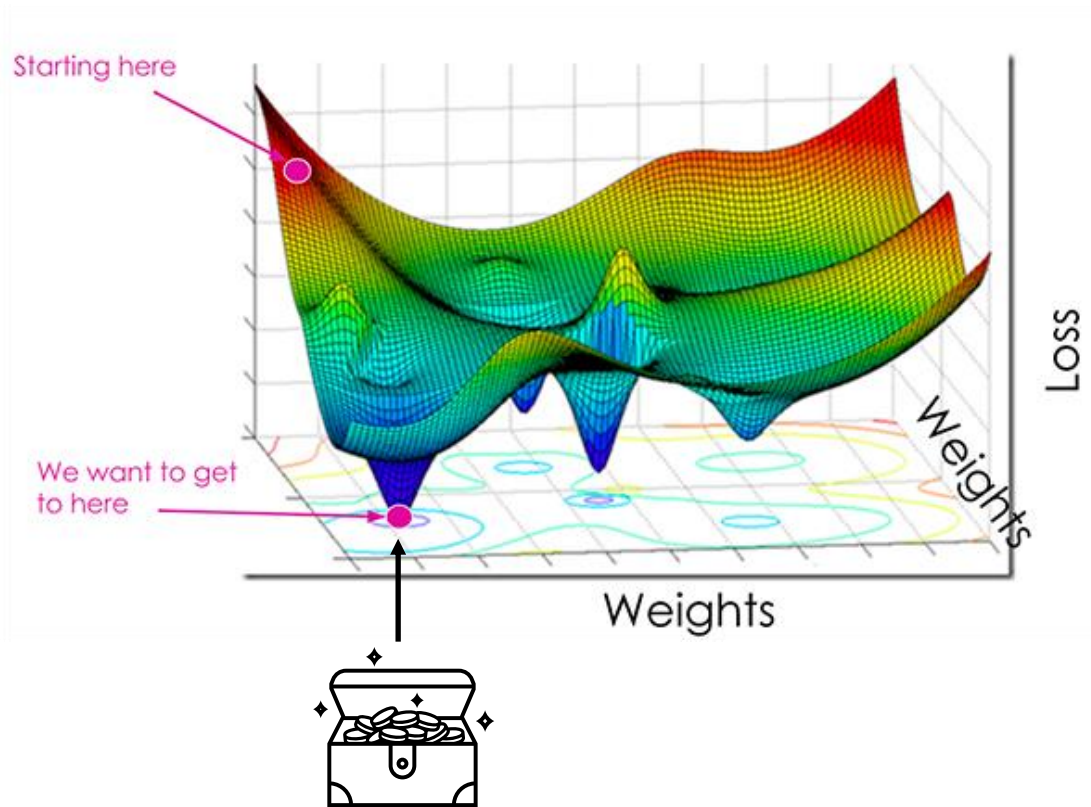
- The loss function is the function that computes the distance between the current output of the algorithm and the expected output. It's a method to evaluate how your algorithm models the data. It can be categorized into two groups. One for classification (discrete values, 0,1,2...) and the other for regression (continuous values).
- Mean square error (mse) is the average of the Sum of squared residual (SSE) over samples.
- We can regard MSE as the function of betas and it is commonly used as the loss function in regression problems.
- A lot of other loss function: Cross-entropy, Log loss, Exponential Loss, etc.

$$\text{SSE} = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (\hat{\beta}_0 + \hat{\beta}_1 X_{1i} + \hat{\beta}_2 X_{2i} + \dots + \hat{\beta}_p X_{pi}))^2$$

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Loss function

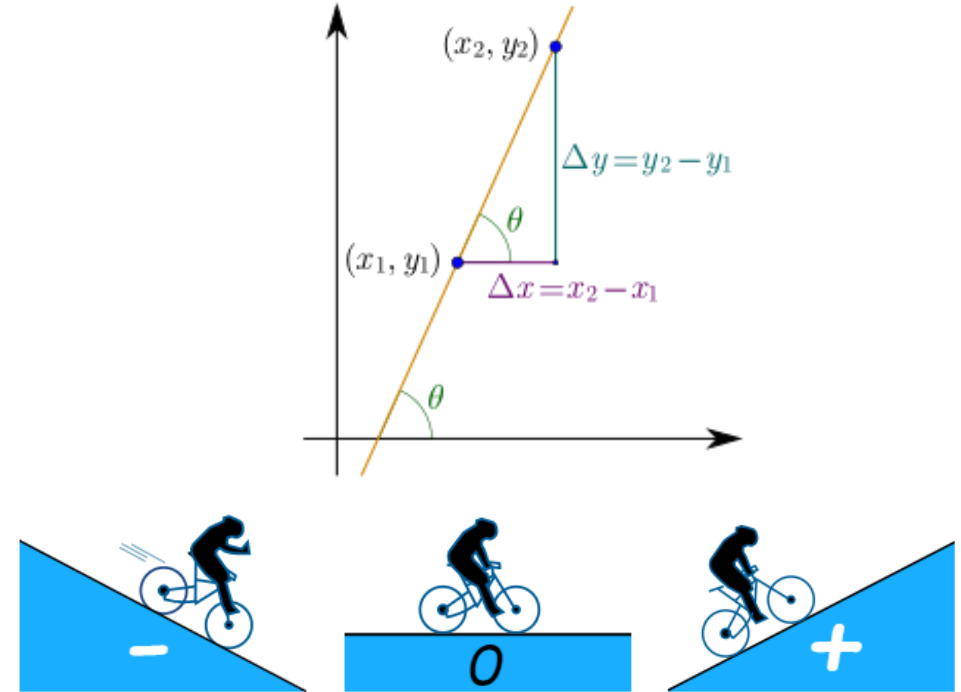
Player A, B, C



- At the most basic level, a loss function quantifies how “good” or “bad” a given predictor is at predicting the response in a dataset (regression).
- The smaller the loss, the better a job the regression model or NN is at modeling the relationship between the input data and the output targets.
- We, therefore, seek to:
 1. Bring down our loss, which will improve the accuracy of our model.
 2. Do it as quickly as possible and with as few changes or experiments to the hyperparameters as possible.
 3. All without making our network too good at what it does by fitting the training data too well.

Loss function

- **Optimizers** are algorithms or methods used to minimize an error function(loss function) or to maximize the efficiency of production.
- Optimizers help to know how to change weights and learning rate of neural network to reduce the losses.
- Gradient: In mathematics, the slope or gradient of a line is a number that describes both the direction and the steepness of the line (Wiki).



Loss function

- Types of optimizer:

Player A

- Gradient Descent

Player B

- Stochastic Gradient Descent (SGD)

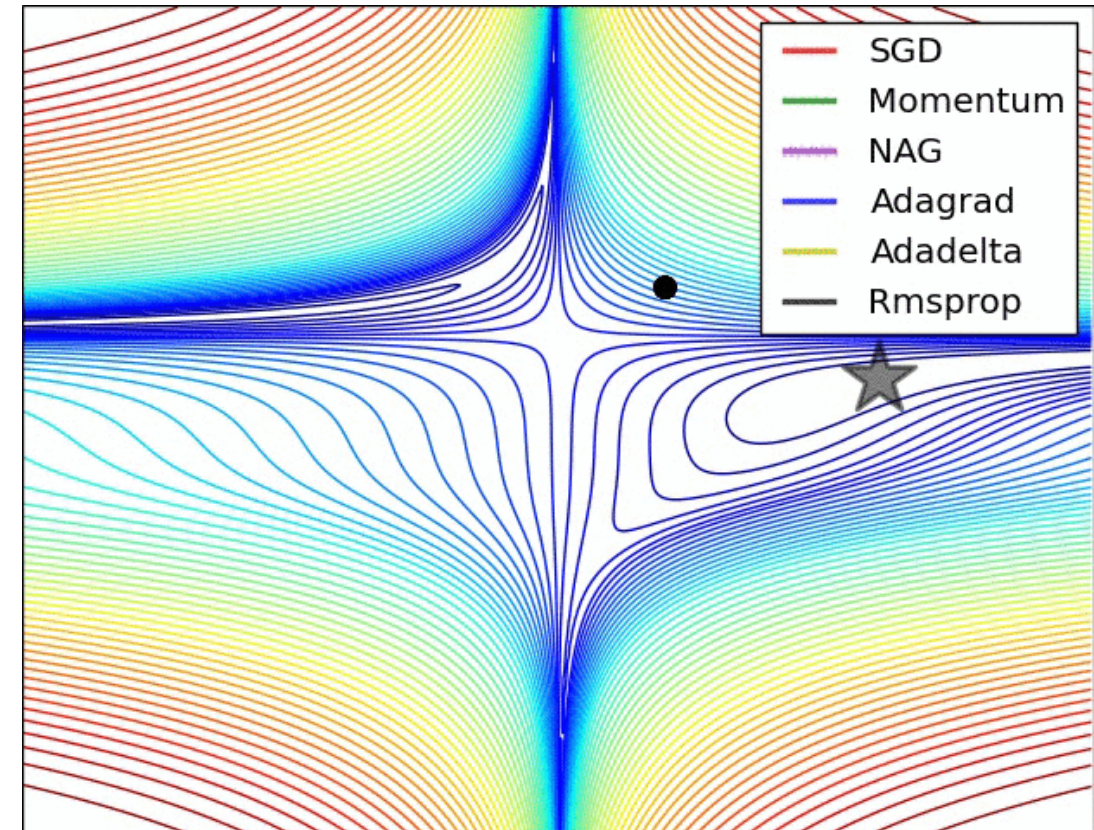
Player C

- Adam(Adaptive Moment Estimation)
Adam optimizer is one of the most popular and famous gradient descent optimization algorithms.

$$W_{new} = W_{old} - \boxed{\alpha} * \frac{\partial(Loss)}{\partial(W_{old})}$$

↑
★

α : can be a constant value, also can vary. Effect by optimizers.



3. TensorFlow


- TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks (Wiki).
- Python packages: a package is a folder containing modules and maybe other folders that themselves may contain more folders and modules.
- Python functions: A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.
- tf.keras (<https://www.tensorflow.org/guide/keras>) is the simplest way to build and train neural network models in TensorFlow. So, that's what we'll stick with in this tutorial, unless the models necessitate a lower-level API.
- Note that there's tf.keras comes with TensorFlow so you don't need to install anything extra, and it comes with powerful TensorFlow-specific features.

A lot of functions



tf.keras





Hands-on exercise 1: Build a fully connected neural network

- Building the neural network requires configuring the layers of the model, then compiling the model.
- First, we stack a few layers together using ``keras.Sequential``.
- Next, we configure the loss function, optimizer, and metrics to monitor. These are added during the model's compile step:
 - Loss function - measures how accurate the model is during training, we want to minimize this with the optimizer.
 - Optimizer - how the model is updated based on the data it sees and its loss function.
 - Metrics - used to monitor the training and testing steps.
- Let's build a network with 1 hidden layer of 20 neurons, and use mean squared error (MSE) as the loss function (most common one for regression problems).



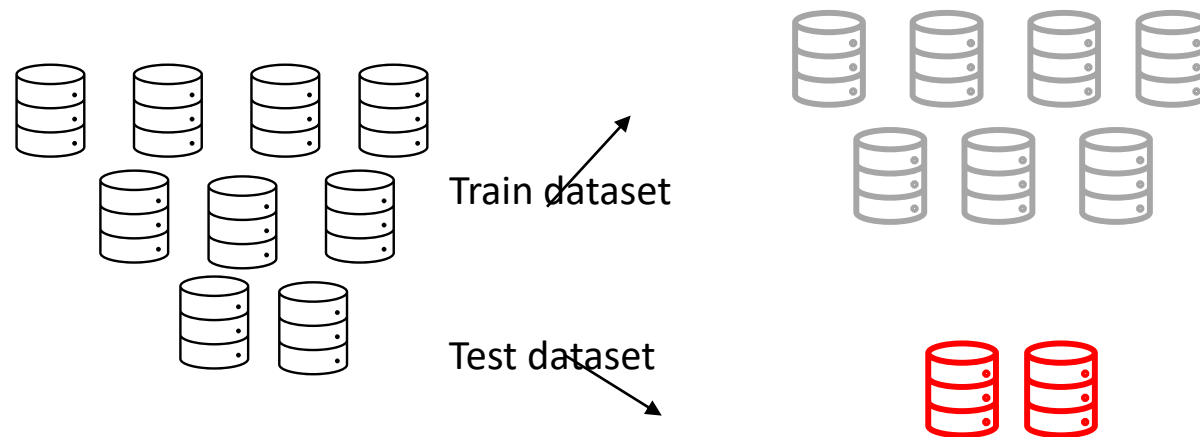
Coffee break

15:10

4. Datasets

Why split dataset into training and test?

A very common issue when training a model is **overfitting**. This phenomenon occurs when a model performs really well on the data that we used to train it, but it fails to generalise well to new, unseen data points. There are numerous reasons why this can happen — it could be due to the noise in data or it could be that the model learned to predict specific inputs rather than the predictive parameters that could help it make correct predictions. Typically, the higher the complexity of a model the higher the chance that it will be overfitted.



Split data into training and test dataset

- Training set for training our model and then treat the testing set as a collection of data points that will help us evaluate whether the model can generalise well to new, unseen data.
- A common practice is to assign 8/10 data points as training dataset and 2/10 data points as test dataset.
- In case when the training accuracy is extremely high while the testing accuracy is poor then this is a good indicator that the model is probably overfitted.

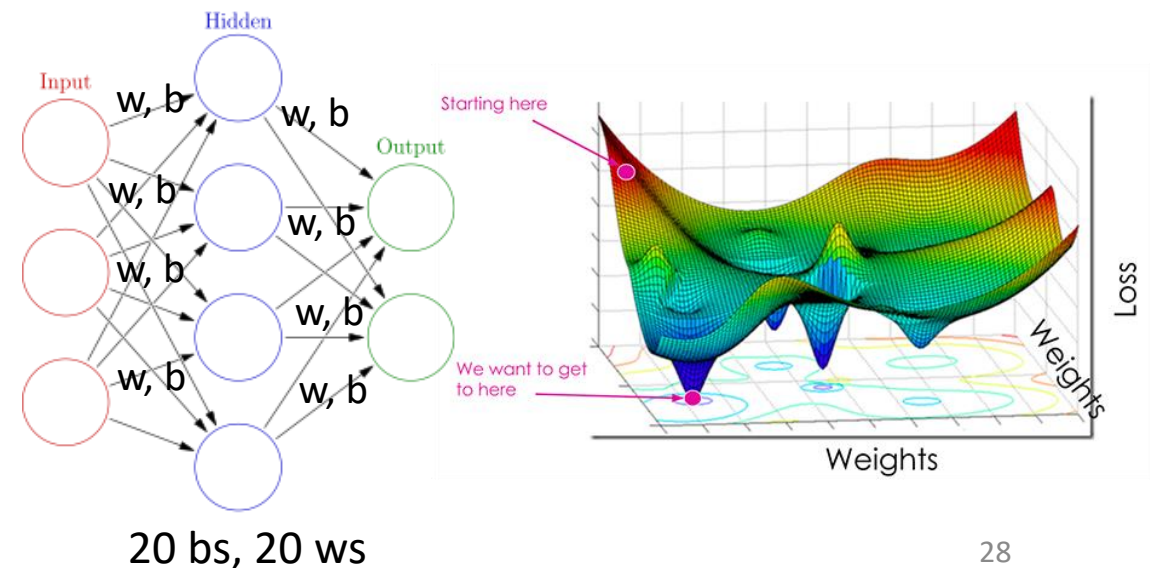
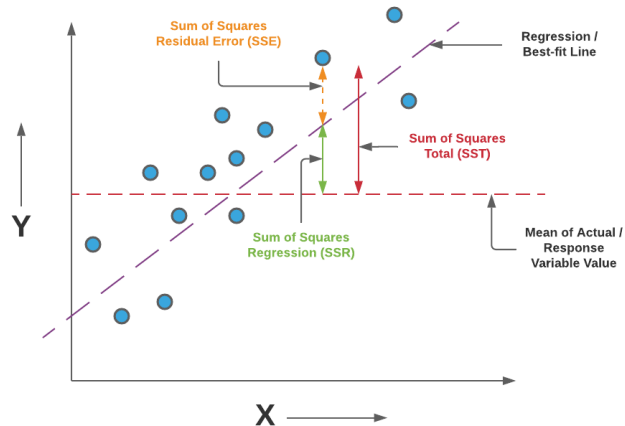
5. Model training and evaluation

A machine learning training model is a process in which a machine learning (ML) algorithm is fed with sufficient training data to learn from.

Parameter estimation is the process of computing a model's parameter values from measured data.

response, **coefficients of correlation**, predictors

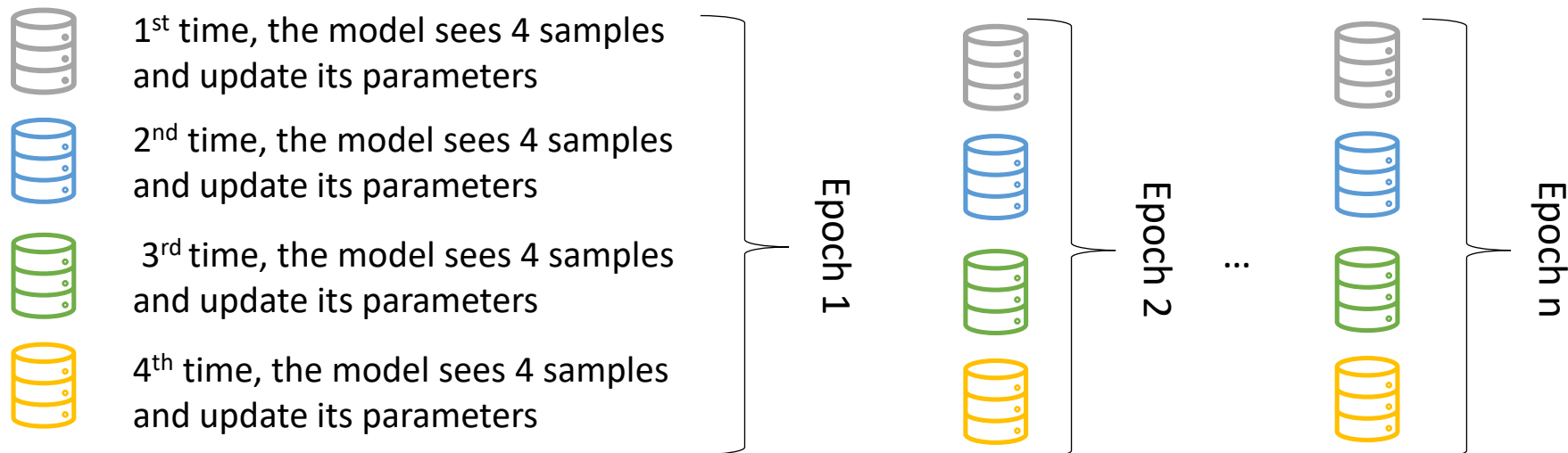
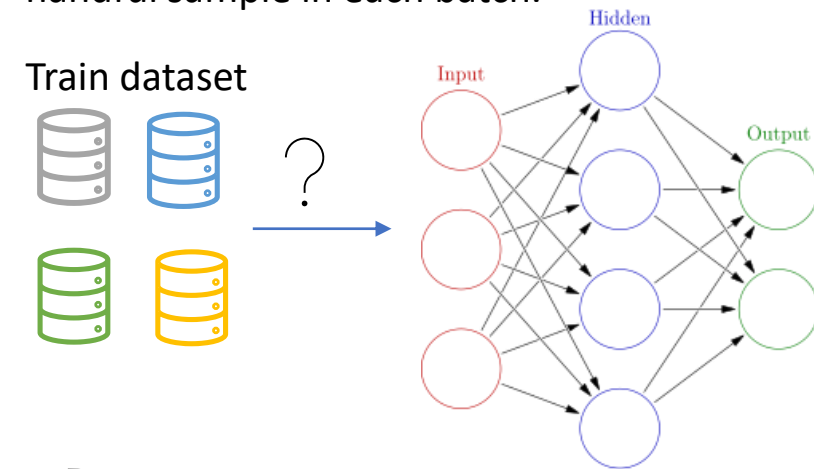
$$g(y_i) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$$



5. Model training and evaluation

- Epoch is when an entire train dataset is passed forward and backward through the neural network only once.
- Batch sizes. A total number of training examples used to update the model's parameters one time. A reasonable number neither too small nor too large. A typical value is 2^n , $n=3, 4, 5, 6$.
- Iterations is the number of batches need to complete one epoch.

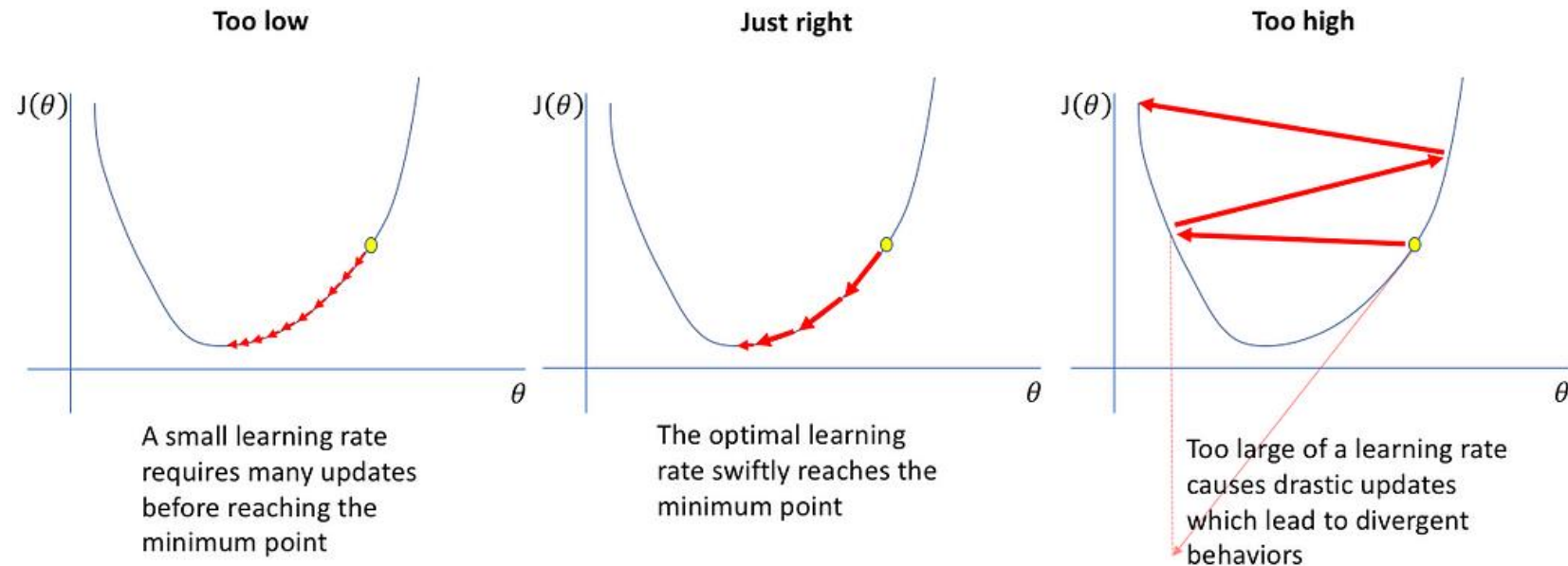
Q: Are we passing all of them to NN at once?
A: It's better to pass them in batches with a handful sample in each batch.



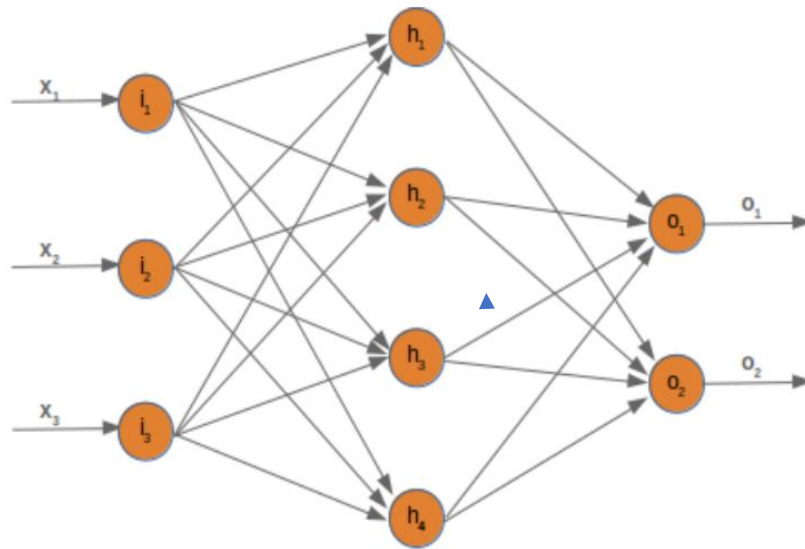
Learning rate

$$W_{new} = W_{old} - \alpha * \frac{\partial(Loss)}{\partial(W_{old})}$$

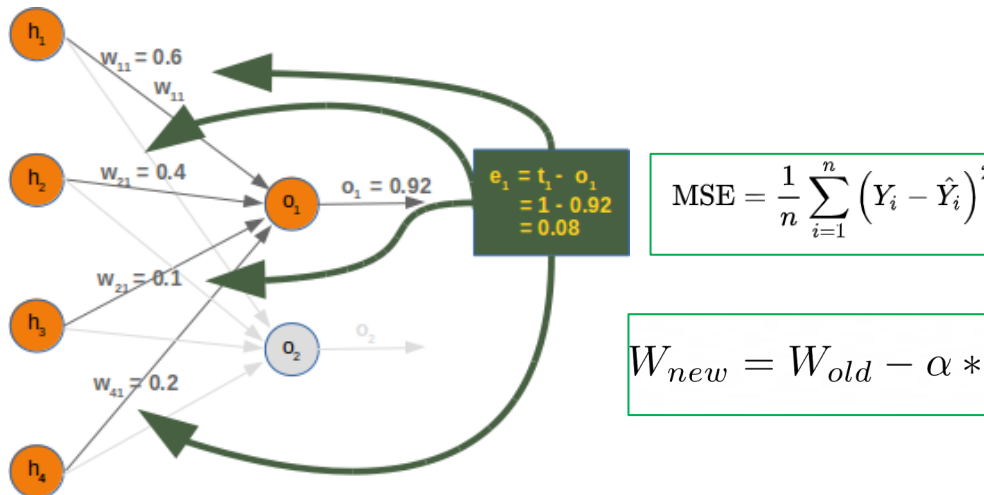
- How big/small the steps are gradient descent takes into the direction of the local minimum are determined by the learning rate, which figures out how fast or slow we will move towards the optimal weights.



Model training



- Feed forward
 - input training data at the input layer, and it travels from input to hidden and from hidden to the output layer
- Back propagation
 - a widely used algorithm for training feedforward artificial neural networks
 - feed forward the values
 - calculate the error and propagate it back to the earlier layers.



Model evaluation: Metrics

Does that sound like our criteria combos 5?

- Metrics help in evaluating deep learning models on **test dataset**. Metrics are used to monitor and measure the performance of model. Metrics are not differentiable but in some cases is differentiable as can be used both as loss function(Regularization added) and a metric.
- Regression:
 - Mean squared values (MSE),
 - Root Mean Squared Error (RMSE), the units of the RMSE are the same as the original units of the target value that is being predicted.
 - Mean absolute error (MAE), the units of the error score match the units of the target value that is being predicted.
- Classification: a lot, precision, accuracy, F1 score, Recall, ROC curve, etc.

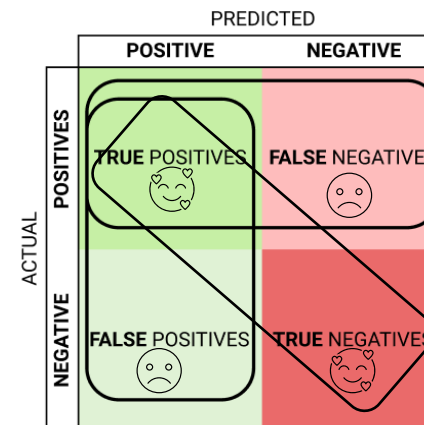


Regression


$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$

Classification



Precision
$\frac{\text{TRUE POSITIVES}}{\text{TRUE POSITIVES} + \text{FALSE POSITIVES}}$
Recall / TPR
$\frac{\text{TRUE POSITIVES}}{\text{TRUE POSITIVES} + \text{FALSE NEGATIVES}}$
F1 score
$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} = \frac{2tp}{2tp + fp + fn}$
Accuracy
$\frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$




Hands-on exercise2 : Split and load datasets

Use `iris_data` from `sklearn.datasets` package and `iris_load` function to load iris data

Check the head of the `iris_data`

Split `iris_data` into 80% training dataset, 20% as test dataset using `train_test_split()` function from `sklearn.model_selection` package.



Hands-on exercise 3: Training and evaluate NN models

Use `keras.datasets.boston_housing_load_data()` to load predefined training and test dataset for house price.
Train the pre-build neural network with one layer and 20 nodes on the training sample.



Thank you

- Questions OR Comments?
- Slack channel: section2-course-documents
- Email: qing.li2@uclagary.ca