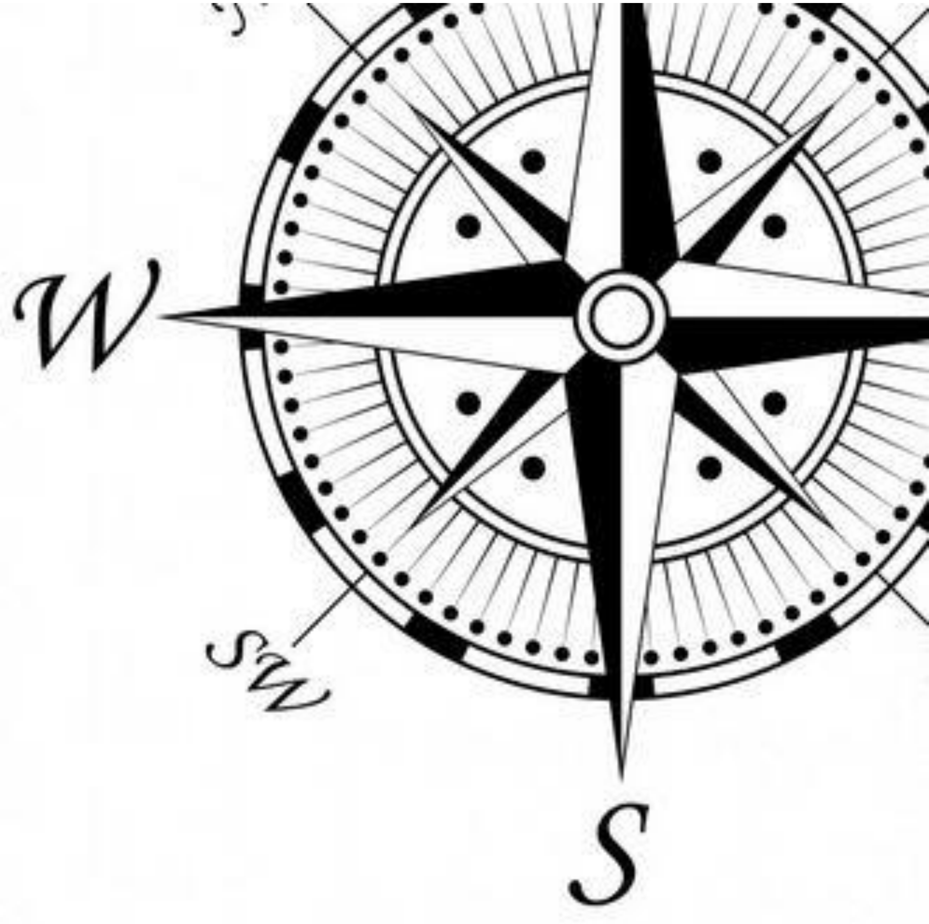
Intro Neural Network

Aftab Anjum

AG Information Profiling and Retrieval

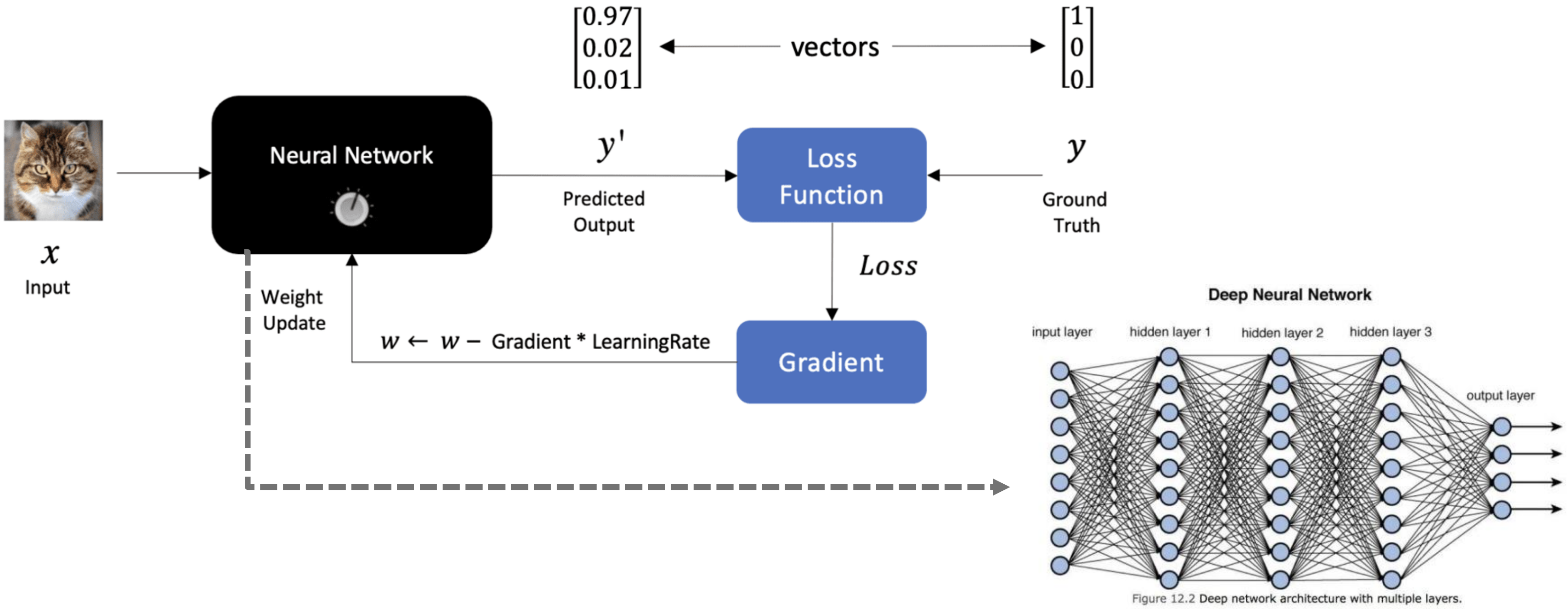
Topics Today

- Introduction to Neural network
- Frameworks & Libraries
- Jupyter Notebook & Google Colab

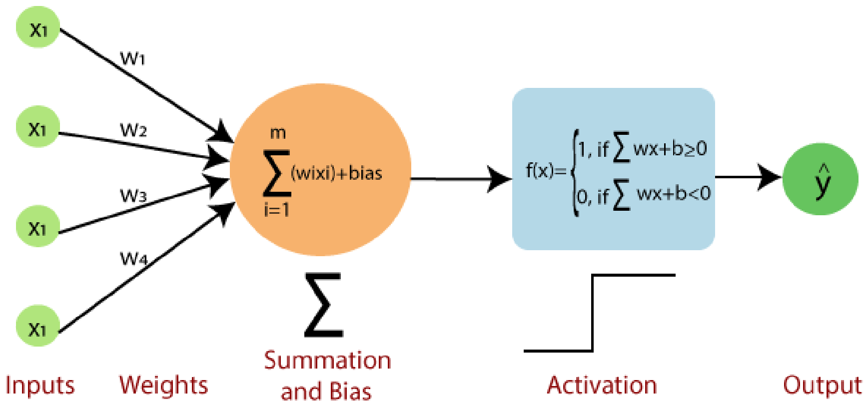


Neural Network (NN)

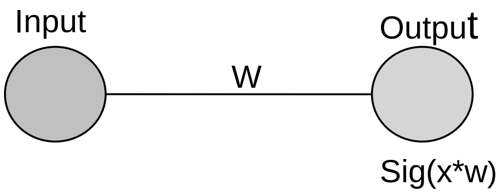
Lets see one naive example.



Neural Network



Why need bias and what its purpose?



$$sigmoid\ function = \frac{1}{1 + e^{-(w*x+b)}}$$

What is activation function and why we need it?

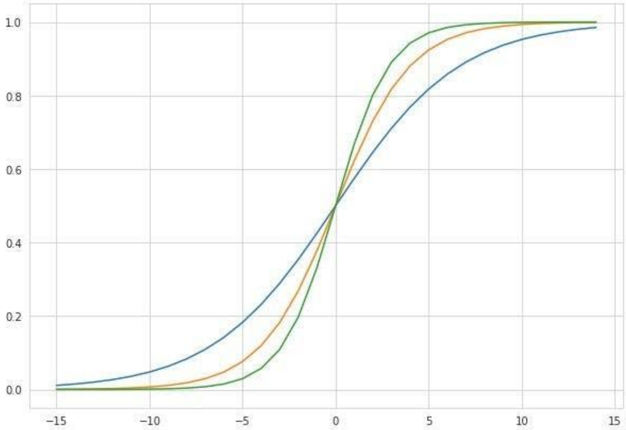
https://medium.com/@gurmeharkaur01/neural-networks-a-journey-into-the-weights-and-biases-39a00ac9265b

Neural Network

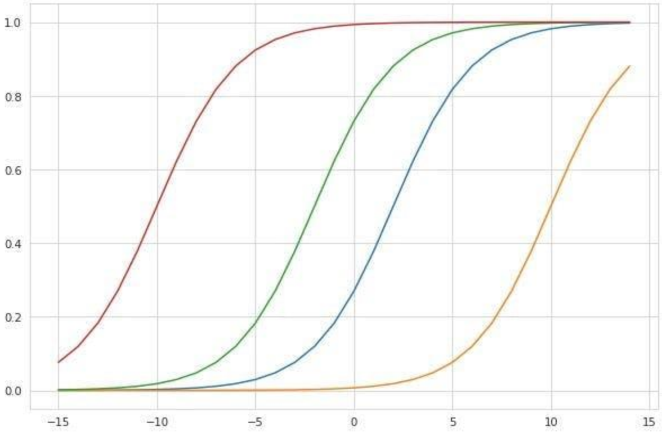
Why need bias and what its purpose?

- Neural network bias: A constant added to the product of features and weights.
- Purpose: Offset the result and shift the activation function.
- Importance: Essential for model flexibility and learning (helps to shift the activation function towards the positive or negative side.)

Changing weight **w**, keeping bias **b**=0



Weight **w** fixed and varying bias **b**



- Neural networks can approximate linear functions of the form y = mx + c.
- When c = 0, the neural network can only approximate functions passing through the origin; including the constant term c allows approximation of linear functions across the plane.

https://www.turing.com/kb/necessity-of-bias-in-neural-networks

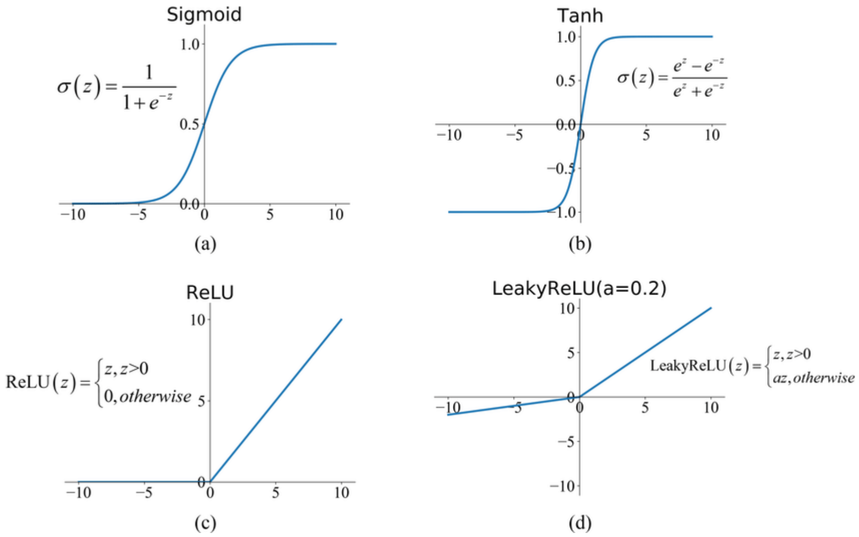
Neural Network

What is activation function and why we need it?

- Activation functions introduce non-linearity to neural networks.
- Non-linearity enables the network to learn complex patterns in data.

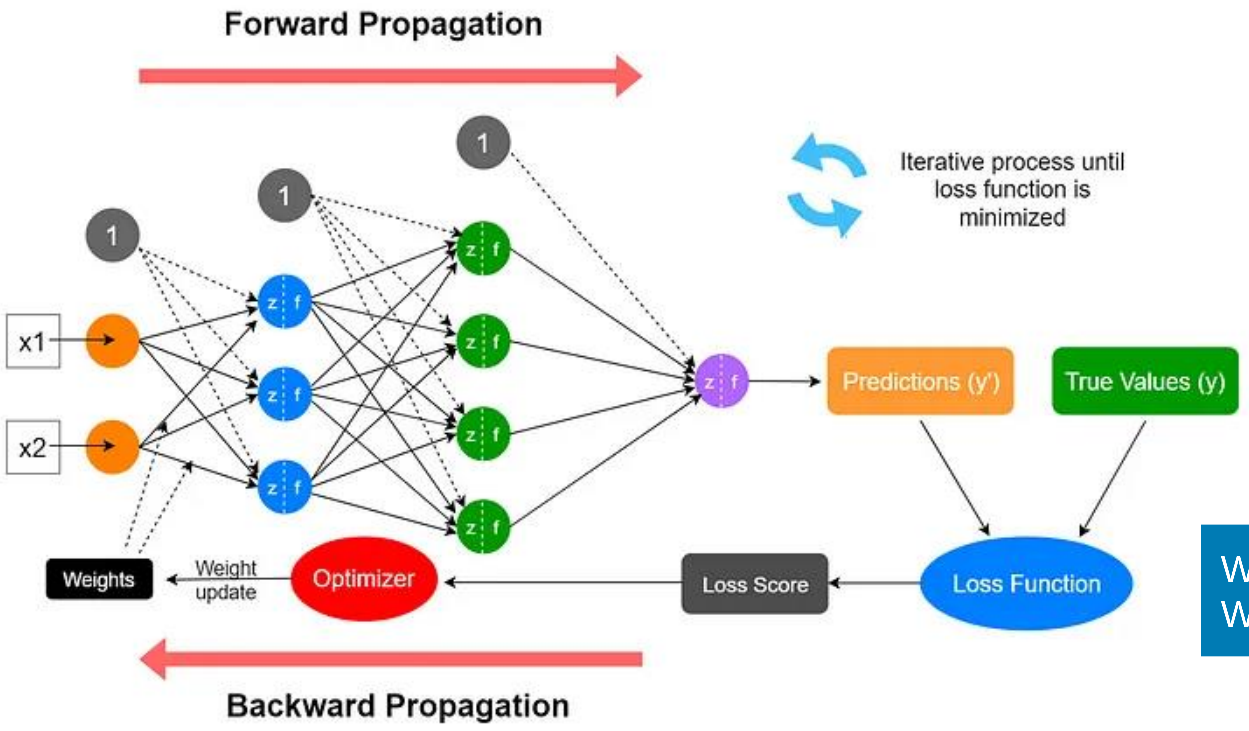
Importance of Activation Functions

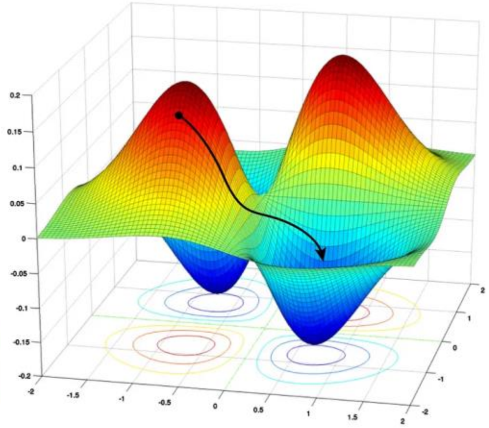
- Without activation functions, neural networks would only learn linear relationships.
- Non-linear activation functions are crucial for approximating arbitrary functions.
- They empower neural networks to solve complex tasks like image recognition and natural language processing.



https://www.turing.com/kb/necessity-of-bias-in-neural-networks

Neural network





What is Gradient Descent?
What is learning rate?

https://medium.com/data-science-365/overview-of-a-neural-networks-learning-process-61690a502fa

Neural Network

What is Gradient Descent?

- An optimization algorithm used to minimize the loss function.
- It iteratively adjusts the paramete

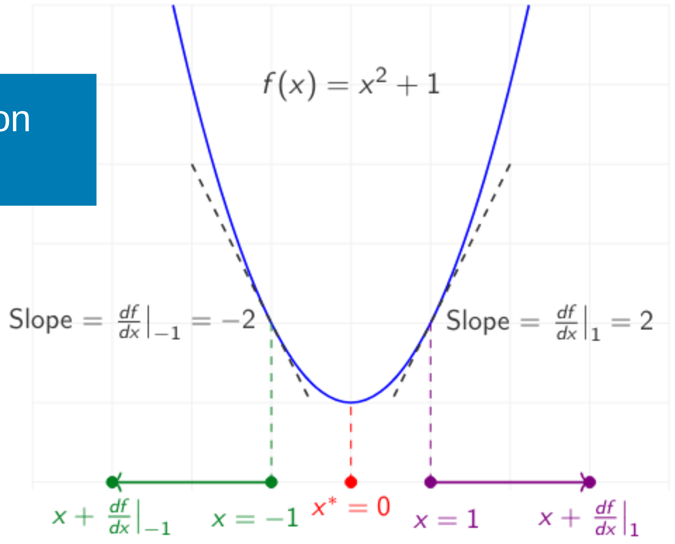
In which direction does function have max rate of increase

- Gradient is the direction, in input space, of maximum rate of increase of a function.

$$f\left(x+\frac{df}{dx}\right) \geq f(x)$$

- To minimize function f (x) with respect to x, move in negative gradient direction.

$$x^{\text{new}} = x^{\text{old}} - \left.\frac{df}{dx}\right|_{x^{\text{old}}}$$



https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21

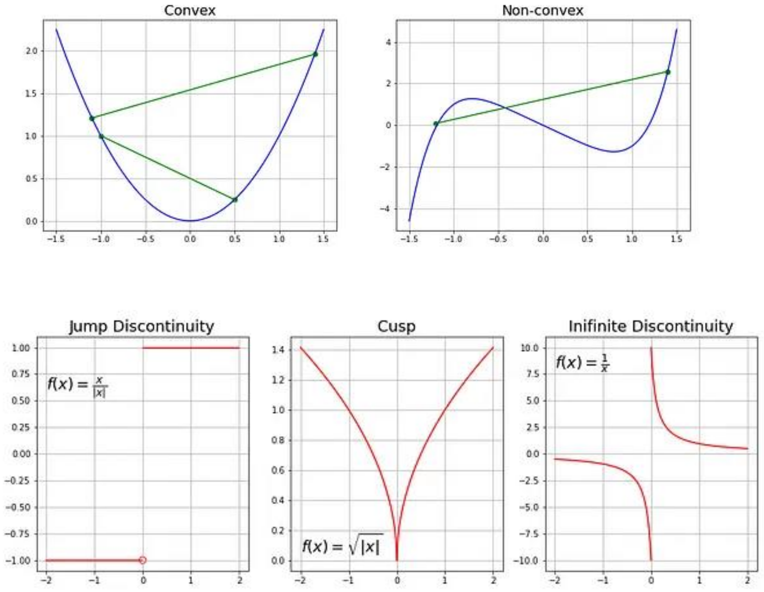
Neural Network

Steps of Gradient Descent

- Initialize the model parameters randomly.
- Compute the gradient of the loss function with respect to the parameters.
- Update the parameters in the direction opposite to the gradient.
- Repeat steps 2 and 3 until convergence or a predetermined number of iterations.

Gradient descent works only on **differentiable** and **convex functions**

What is the vanishing gradient problem in NN.



Examples of non-differentiable functions

https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21

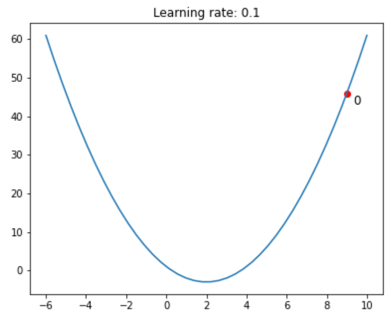
Neural Network

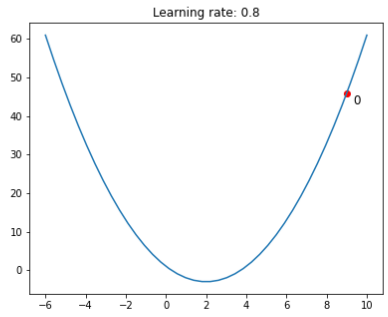
What is learning rate, how it can affect the training of NN.

- Learning rate is a hyper parameter, determines steps size during model training
- Optimal learning rate involves experimentation and tuning or by rate schedules and adaptive methods.

Effects of Learning Rate

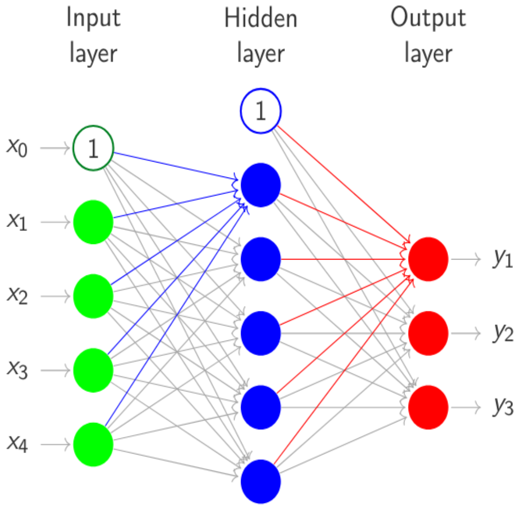
- Learning rate controls the convergence and stability of the training process.
- Too high learning rate can lead to overshooting, making the model fail to converge.
- Too low learning rate can result in slow convergence or getting stuck in local minima.





Neural Network: Forward-propagation

- Input layer neurons indexed by i
- Hidden layer neurons indexed by j
- Next hidden layer or output layer neurons will be indexed by k
- Weights of j-th hidden neuron will be denoted by the vector is $w_j^{(1)} \in \mathbb{R}^D$
- Weights between i-th input neuron and j-th hidden neuron is $w_{ji}^{(1)}$
- Weights of k-th output neuron will be denoted by the vector is $w_k^{(2)} \in \mathbb{R}^M$
- Weights between j-th hidden neuron and k-th output neuron is $w_{kj}^{(2)}$



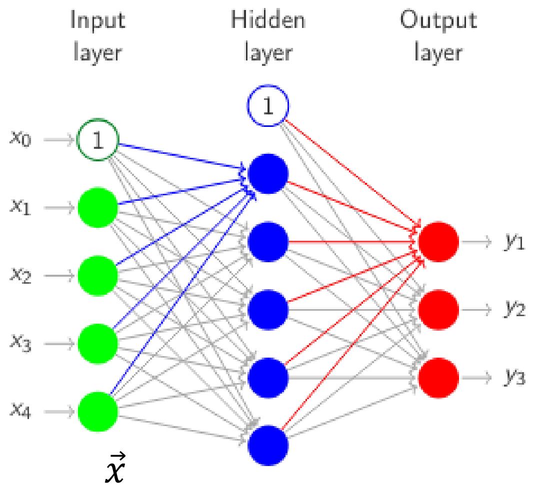
$w_{21}^{(2)}$

Neural Network: Forward-propagation

- For input x, denote out of hidden layer as the vector z(x) ∈ $\mathbb{R}^M$
- Model z_j(x) as non-linear function h (a_j) where pre-activation $a_j = w_j^{(1)\,T}$x with adjustable parameters $w_j^{(1)}$
- So the k-th output can be written as

$$y_k(\mathbf{x}) = f(a_k) = f(\mathbf{w}_k^{(2)T}\mathbf{z}(\mathbf{x}))$$
$$= f\left(\sum_{j=1}^{M} w_{kj}^{(2)} z_j(\mathbf{x}) + w_{k0}^{(2)}\right) = f\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right) + w_{k0}^{(2)}\right)$$

- where we have prepended x_0 = 1 to absorb bias input and $w_{j0}^{(1)}$and $w_{k0}^{(2)}$ represent biases.



$\vec{z}(\overrightarrow{x})$

z_j=h(a_j)

=h$(\vec{w}_j^{(1)T}\ \vec{x})$

= h$(\sum_{i=0}^{D} w_{ji}^{(1)}\ \ x_i)$

Neural Network: Forward-propagation

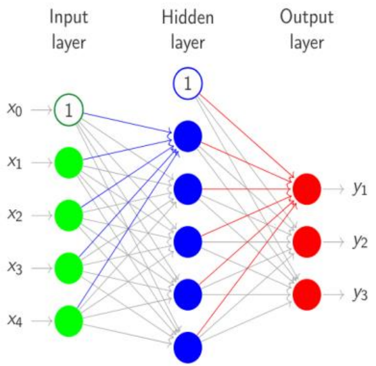
Which part is the hidden layer

- The computation

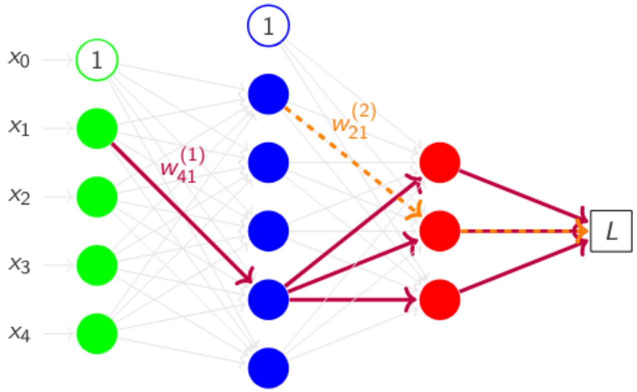
$$y_k(\mathbf{x}, \mathbf{W}) = f\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right) + w_{k0}^{(2)}\right)$$

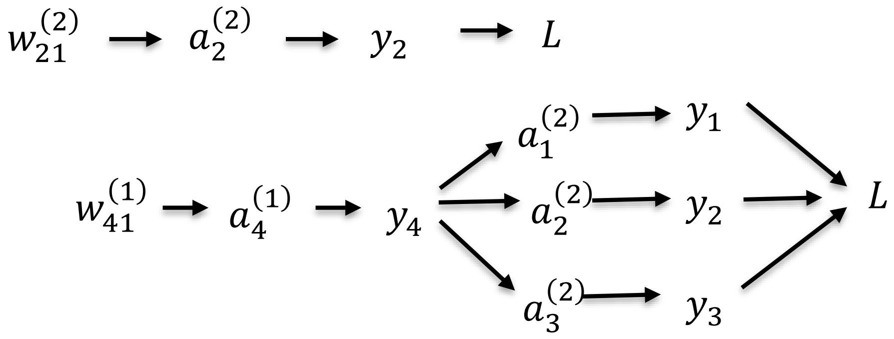
Compute forward pass for layer 1 neuron 1.

- can be viewed in two stages:
  - $z_j = \mathrm{h}\,(w_j^{(1)\,T}\mathrm{x})$ for j = 1, . . . , M.
  - $y_k = \mathrm{f}\,(w_k^{(2)\,T}\mathrm{z})$



Neural Network: Forward-propagation



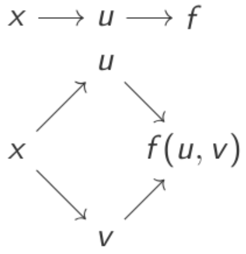


- Multivariate Chain Rule
  - The chain rule of differentiation states

$$\frac{df(u(x))}{dx} = \frac{df}{du}\frac{du}{dx}$$

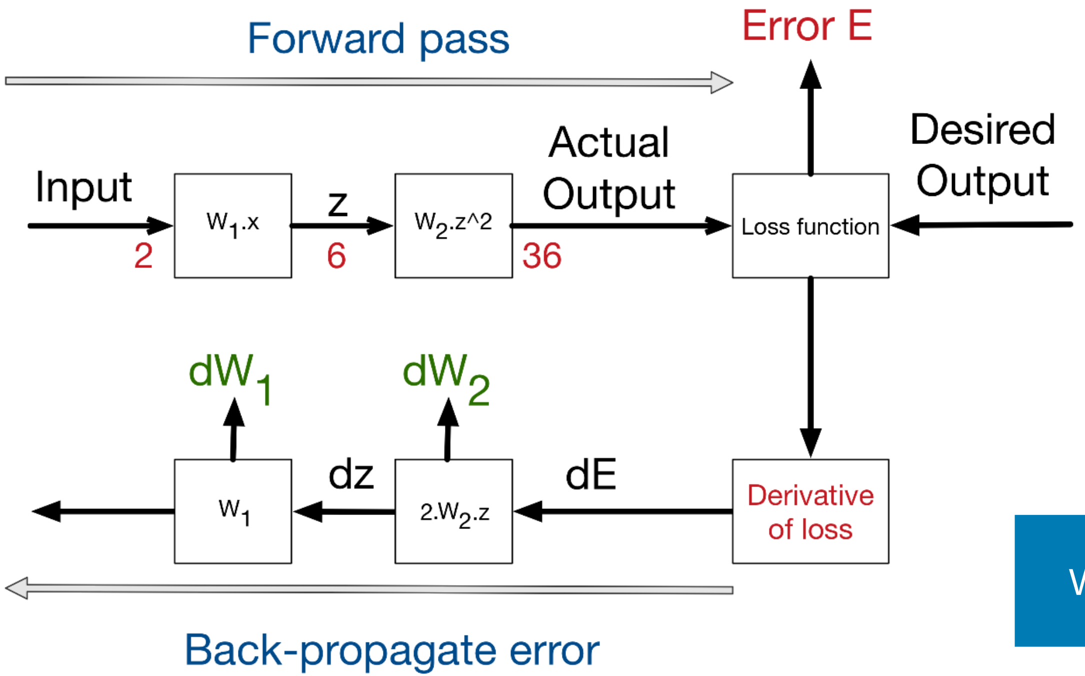
  - The multivariate chain rule of differentiation states

$$\frac{df(u(x),v(x))}{dx} = \frac{\partial f}{\partial u}\frac{du}{dx} + \frac{\partial f}{\partial v}\frac{dv}{dx}$$



- The multivariate chain rule applied to compute derivatives w.r.t weights of hidden layers has a special name – backpropagation.

Neural Network: Backpropagation



- Let compute forward pass and backpropagation with one example by hand.

What is overfitting and under-fitting?

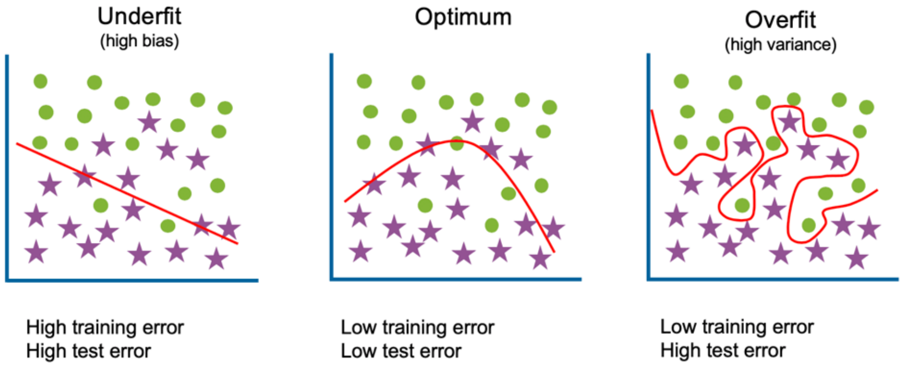
Neural Network

Underfitting

- Underfitting occurs when a model (too simple) not able to capture the underlying structure of the data
- Model have high bias and low variance.
- The model performs poorly both on the training and testing data.

Overfitting

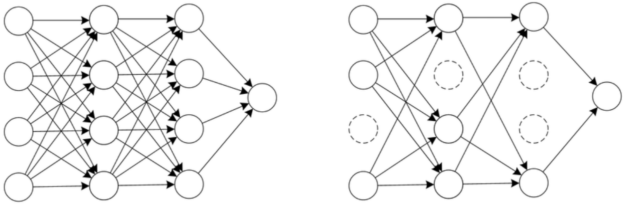
- Overfitting occurs when a model learns the training data too well, capturing noise and outliers.
- It results in low bias and high variance.
- Model performs well on the training data but fails to generalize to unseen data.



Neural Network: Dropout

- One of the most used regularization techniques in neural network.
- During training, a randomly selected subset of activations are set to zero within each layer.

- Dropout layer implementation is very simple.
- For each neuron (including inputs),
  - Generate a uniform random number between 0 and 1
  - If the number is greater than α, then neuron's output to 0
  - Otherwise, don't touch the neuron's output



(a) Standard Neural Network

(b) Network after Dropout

- Probably of dropping out is 1 − α

How to compute gradient in term of dropout?

Remember which neurons were dropped so that gradients are also zeroed out during backpropagation.

Is it bagging?

Exercise

What are the criticisms you think neural network might face?

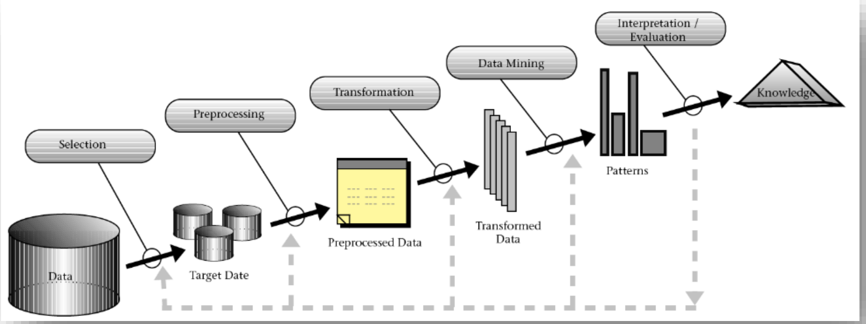
What are the advantages of neural network have over basic machine learning models (DT, RF)?

Motivation

- Nobody wants to reinvent the wheel everytime from scratch
  - Libraries
  - Frameworks

What‘s the difference?

U. Fayyad, G. P.-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. AI Magazine, 17(3):37-54, Fall 1996.



- Machine learning has many standardized procedures and methods/algorithms
  - Data preparation
  - Data splitting (train, val, test)
  - Training
  - Evaluation/Inference
  - Visualization
  - …
  - Decision trees
  - Support vector machines
  - Naive Bayes
  - Linear regression
  - Conditional random fields
  - …

Important Python Libraries for DM/ML

- Pandas
  - https://pandas.pydata.org/
  - Data analysis library for python, providing high-performance, easy-to-use data structures and data analysis tools.
- Numpy
  - https://numpy.org/
  - NumPy is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined.
- Scikit-learn:
  - https://scikit-learn.org/
  - Machine learning in python (built on numpy, scipy, matplotlib): preprocessing, classification, regression, dim reduction, clustering, model selection

Important Frameworks for Deep Learning

- Overview of deep learning software
  - https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

- Popular frameworks (majority written in C++ with Python interface)
  - TensorFlow (Goolge)
  - Theano (U Montreal)
  - PlaidML (Intel)
  - CNTK (Microsoft)
  - MXNet (Apache)
  - Torch
  - Deeplearning4j
  - Caffe (Berkeley)
  - Matlab+DL (MathWorks)

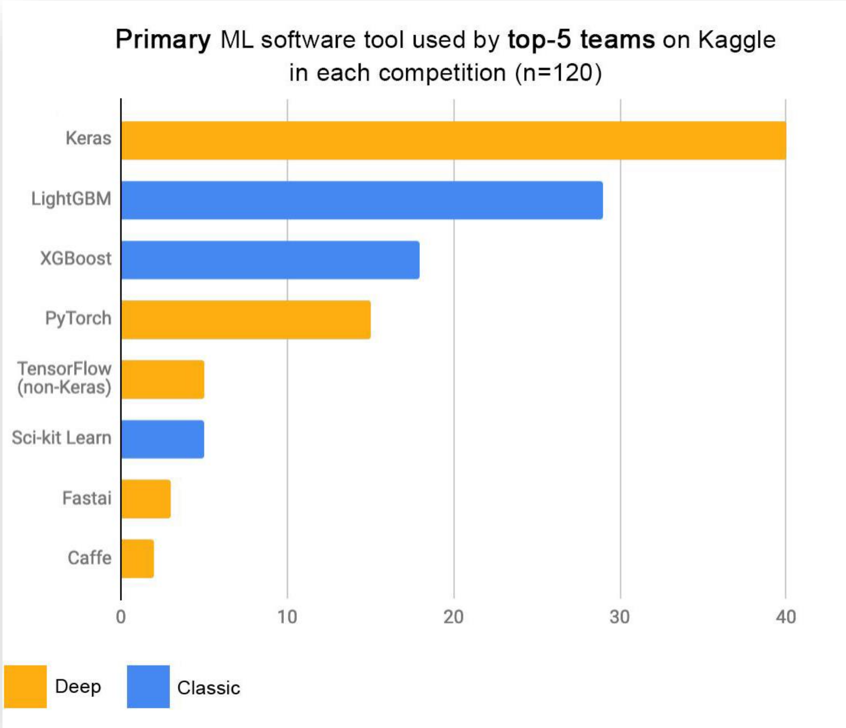
Kaggle

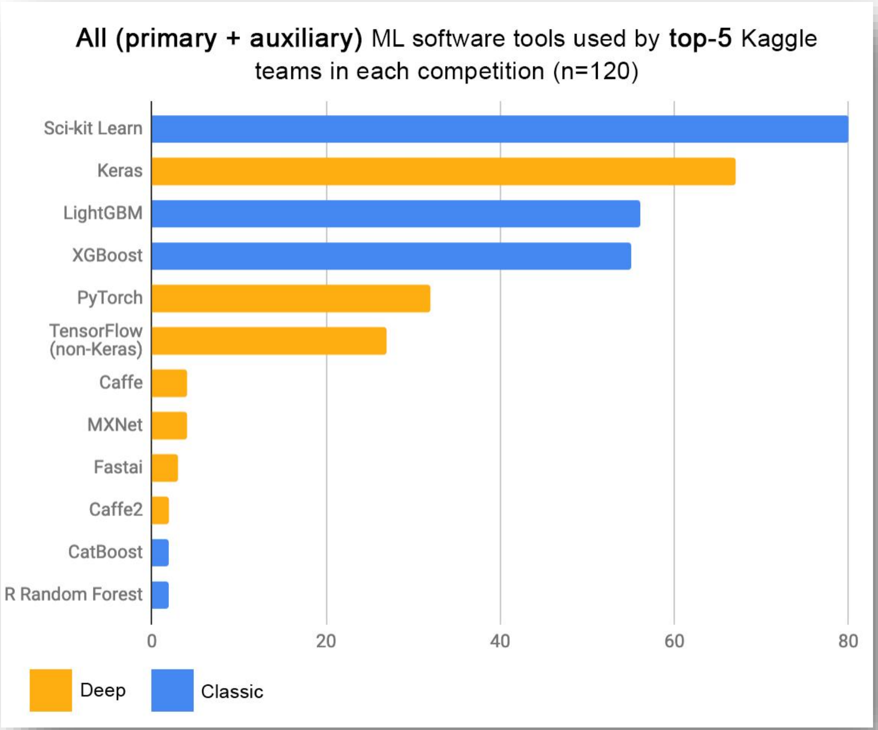
- Belongs to Google
- Online community of data scientists and machine learning practitioners
- Retrieve and publish data sets
- Explore, build, and share models

- Organizes competitions to solve data science challenges
  1. The competition host prepares the data and a description of the problem.
  2. Participants experiment with different techniques and compete against each other to produce the best models. Work is shared publicly through Kaggle Kernels to achieve a better benchmark and to inspire new ideas.
  3. After the deadline passes, the competition host pays the prize.

https://en.wikipedia.org/wiki/Kaggle

Software Used in Kaggle Competitions





https://keras.io/why_keras/

Jupyter Notebook

- https://jupyter.org/
- The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.
  - Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Google Colab

- https://colab.research.google.com/
- Colaboratory is a free Jupyter notebook environment that requires minimum or no setup and runs entirely in the cloud.
- With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.
- https://colab.research.google.com/notebooks/

Colab Demo

- Train a text classifier for the 20 newsgroups dataset using scikit.learn on Google Colab after exploring (understanding) the dataset
- Hints
  - The 20 newsgroups dataset is directly available within scikit.learn
  - It is easy to transform text to tfidf vectors

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

newsgroups_train = fetch_20newsgroups(subset='train',remove=('headers', 'footers', 'quotes'))
vectors_train = vectorizer.fit_transform(newsgroups_train.data)
newsgroups_test = fetch_20newsgroups(subset='test', remove=('headers', 'footers', 'quotes'))
vectors_test = vectorizer.transform(newsgroups_test.data)
```

Literature

- https://pytorch.org/
- https://docs.jupyter.org/en/latest/
- https://colab.research.google.com/
- https://docs.python.org/3/