LECTURER: Nghia Duong-Trung

NEURAL NETS AND DEEP LEARNING

- Course book: DLBDSNNDL01_Neural Nets and Deep Learning, provided by IU, myStudies
- Reading list DLBDSNNDL01, provided by IU, myStudies
- This slide is a summarization of important contents in the course book.
- Additional teaching materials:

https://github.com/duongtrung/IU-DLBDSNNDL01_Neural_Nets_and_Deep_Learning

UNIT 2

Feed-forward Networks

After completing this unit you will be able to ...

– understand what a feed-forward neural network is.

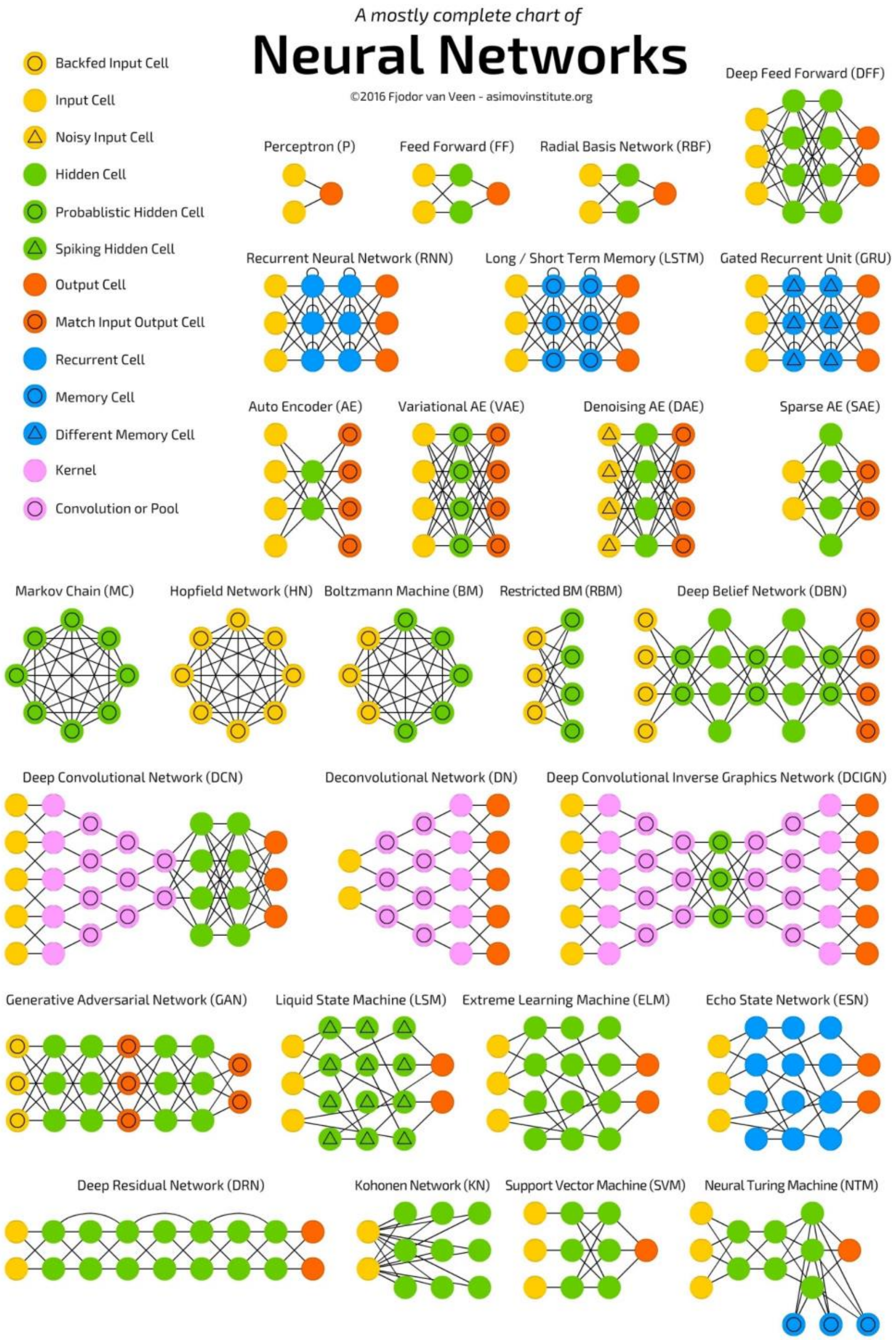
– distinguish between different network topologies and cost functions.

– explain how the backpropagation and gradient descent algorithm help a neural network to train a model.

– build a simple feed-forward neural network used for classifying image data.

– describe how batch normalization can improve the performance of a network

CHART OF NEURAL NETWORKS



https://www.asimovinstitute.org/neural-network-zoo/

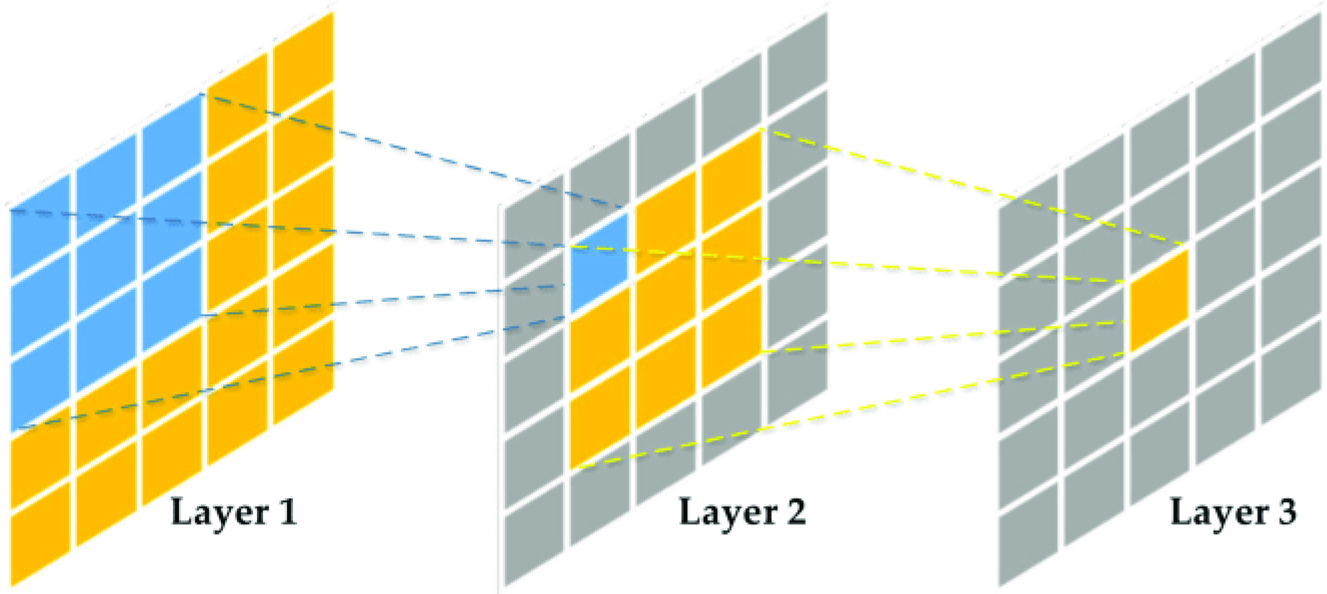
- We need 3 dimensions: height, width, color
- Color dimension always has size 3, corresponding to the RGB channels
- A(I,j,k) stores the value of the i'th, j'th column, k'th color
  - k = red, green, blue
- Physically, color is light, measured by light intensity
- It's a continuous value
- More precision -> Image takes up more space
- We've figured out that 8 bits (1 byte) is good enough
- $2^8 = 256$ possible values (0,1,2,…255)
  - This gives us 2^8 2^8 2^8 = 16.8 million possible colors
- How much space does a 500x500 image take up?
  - 500 x 500 x 3 x 8 = 6 million bits -> 750,000 bytes -> 732KB JPEG

GRAYSCALE IMAGES

- Images that do not have color can be simplified.
- We call them grayscale because each pixel value can only be black, white, or some shade of gray
- Black = 0, White = 255
- Only requires a 2-D array (height, width)

THE RECEPTIVE FIELD

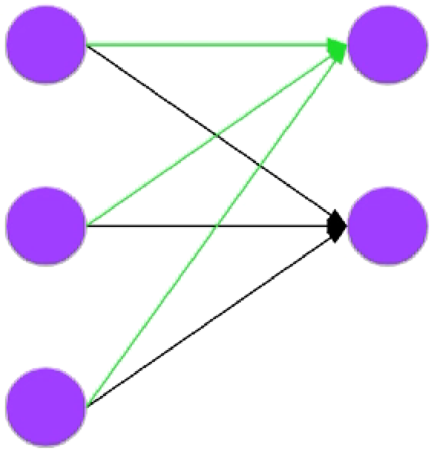
- The receptive field (of a biological neuron) is “the portion of the sensory space that can elicit neuronal responses, when stimulated”
- Based on the image, the entire area (the grid in the figure) an eye can see is called the field of view
- In a deep learning context, the Receptive Field is defined as the size of the region in the input that produces the feature

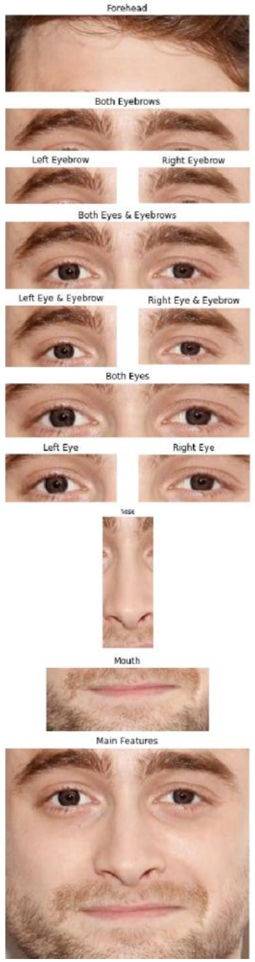


https://www.researchgate.net/figure/Schematic-diagram-of-the-receptive-field-in-CNNs_fig1_353545214

REPEATING THE SINGLE NEURON

- Each of these neurons may calculate something different
  - Via different weights
- E.g. input is a face
  - One neuron looks for the presence of an eye
  - One neuron looks for the presence of a nose
- They are looking for different features

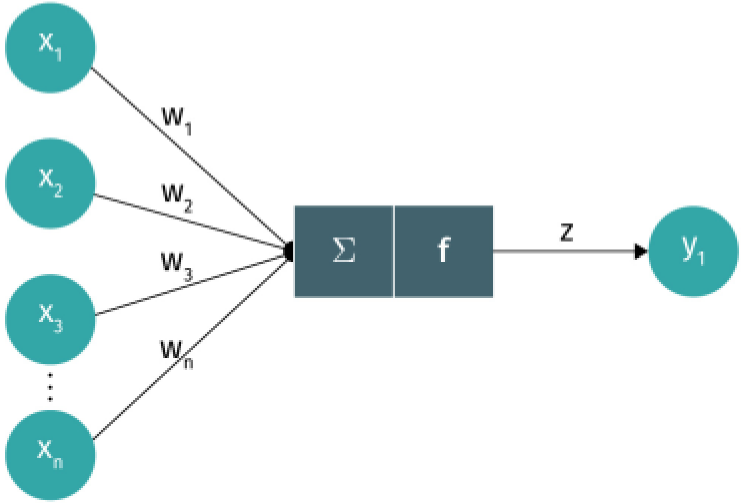




https://github.com/zementalist/Facial-Features-Extraction

PERCEPTRON

- Perceptron consists of binary input values, connection weights, a bias, and an activation function.
- $z = \sigma(w^T x + b) = \sigma(\sum_{i=1}^{n} w_i x_i + b)$ where x_i denotes an instance of the input data, w_i the respective weights for $i \in \{1, n\}$, b the bias and σ the activation function.



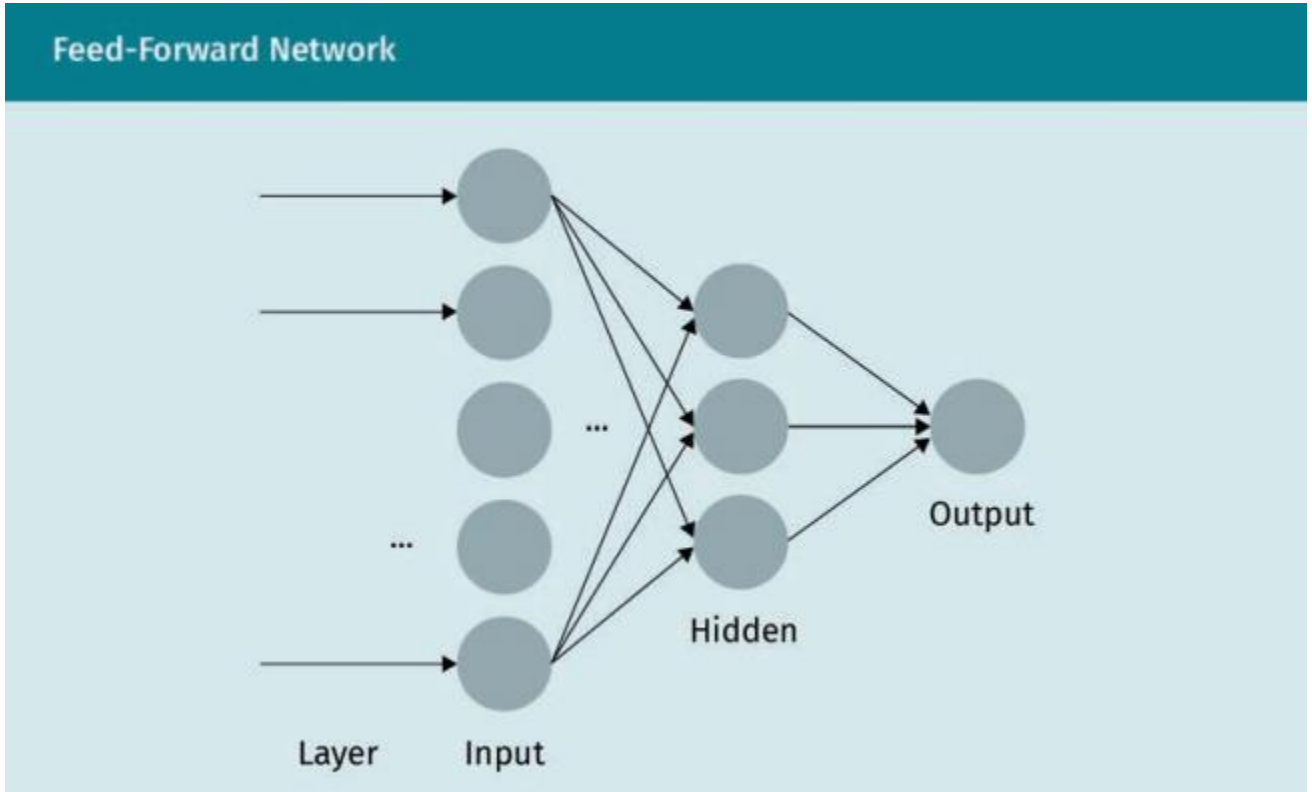
$$w_{i,\,j}^{\text{next step}} = w_{i,\,j} + \eta\left(y_j - \widehat{y_j}\right) \cdot x_i$$

where

- $w_{i,\,j}$ represents the weight of the connection between neuron i and neuron j.
- x_i is the i^th input value of the training data.
- $\widehat{y_j}$ is the output of the j^th output neuron of the training data.
- y_j is the target output of the j^th output neuron of the training data.
- η is the learning rate.

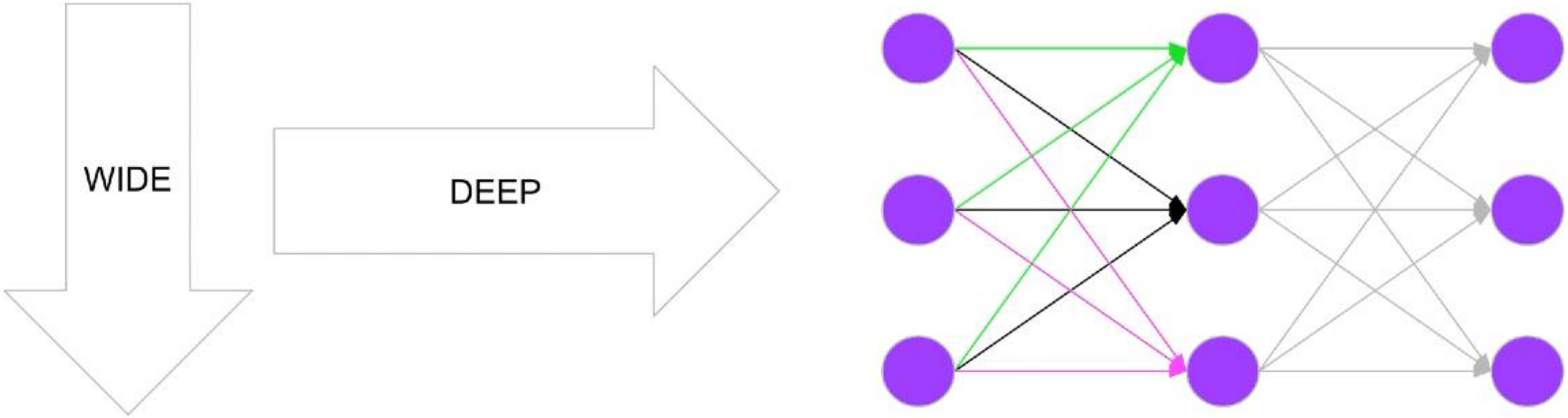
Source: Evis Plaku (2023), based on Géron (2019).

- A model is for making predictions.
- How do we make predictions with a neural network?
- We refer to this as “going in the forward direction”

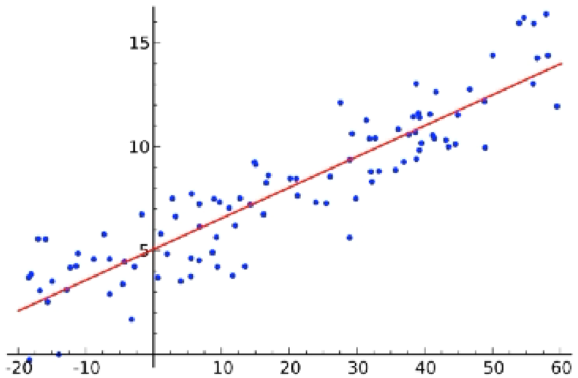


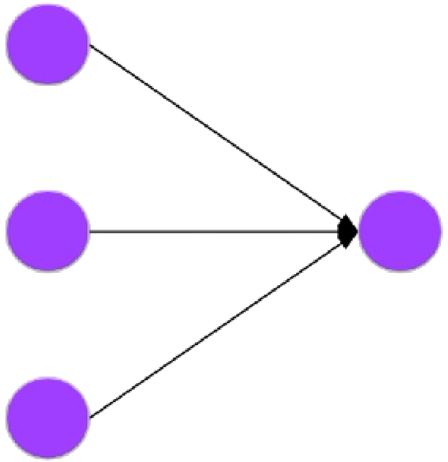
Source of the text: Zöller, 2022.

- Key insight: we can stack more layers of neurons: a chain of neurons
- We can model the brain as uniform structure
- 2 important ways to extend the single neuron
  - The same inputs can be fed to multiple different neurons, each calculating something different (more neurons per layer)
  - Neurons in one layer can act as inputs to another layer

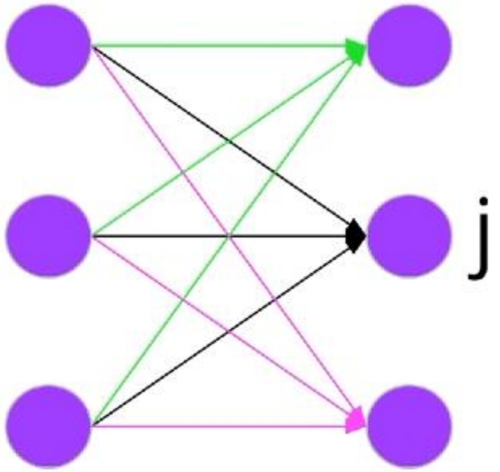


LINES TO NEURONS

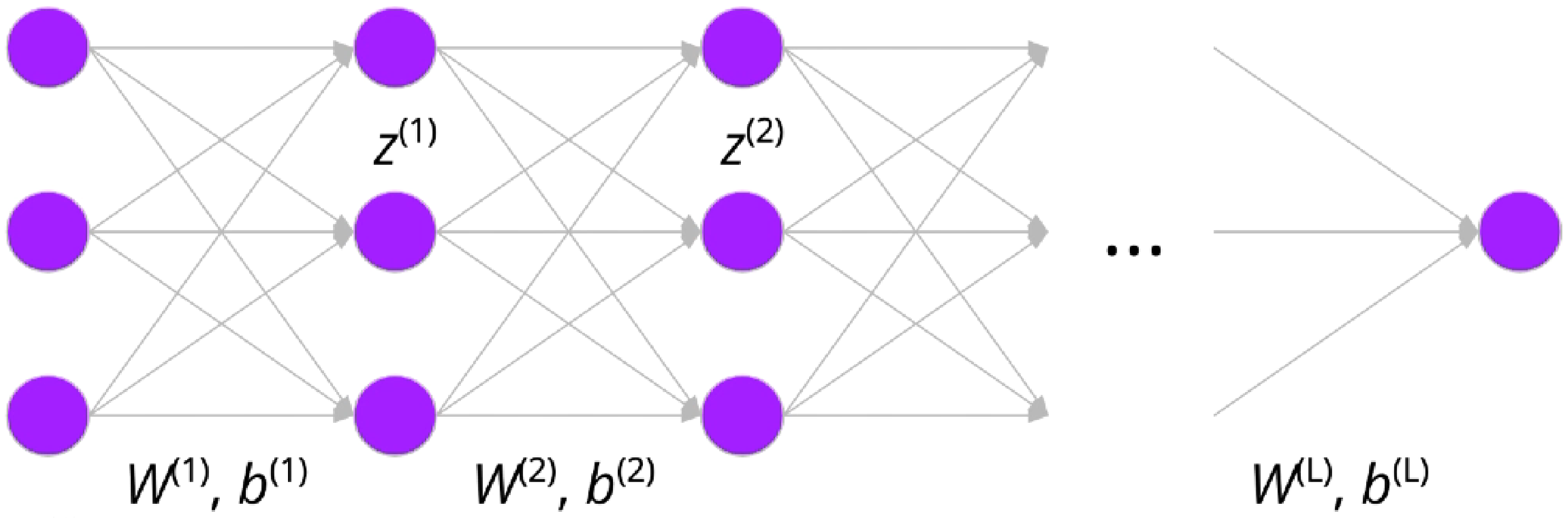




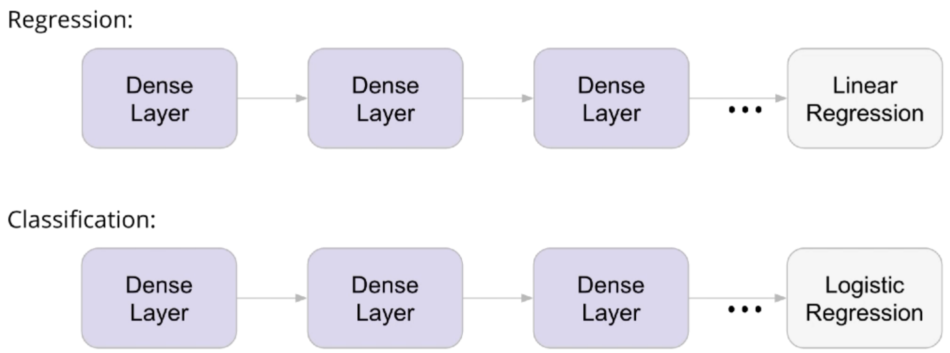
- A line: $ax + b$
- A neuron: $\sigma(w^T x + b)$
- Multiple neurons per layer
  - Call the output of the j_{th} neuron z_j: $z_j = \sigma(w^T x + b)$, for I = 1…M
  - Vectorize the neuron: $z = \sigma(W^T x + b)$
  - Shapes:
    - z is a vector of size M
    - x is a vector of size D
    - W is a matrix of size D x M
    - b is a vector of size M
    - $\sigma()$ is an element-wise operation



INPUT TO OUTPUT FOR AN L-LAYER NEURAL NETWORK



$$z^{(1)} = \sigma(W^{(1)T} x + b^{(1)})$$
$$z^{(2)} = \sigma(W^{(2)T} z^{(1)} + b^{(2)})$$
$$z^{(3)} = \sigma(W^{(3)T} z^{(2)} + b^{(3)})$$
$$\dots$$
$$p(y = 1 \mid x) = \sigma(W^{(L)T} z^{(L-1)} + b^{(L)})$$



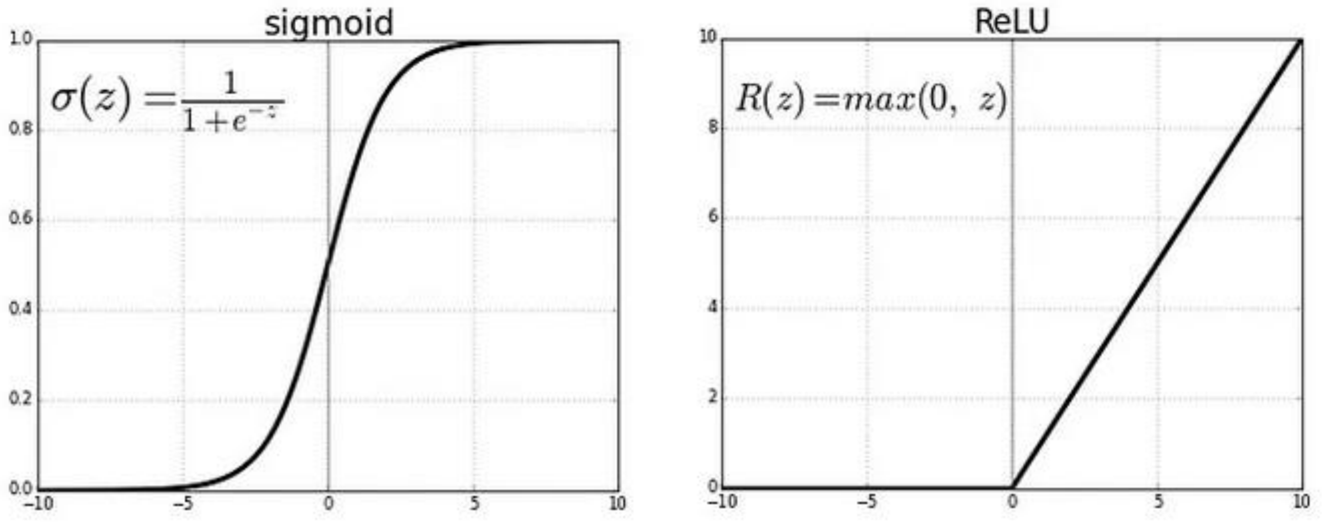
ACTIVATION FUNCTIONS

- It’s just a thing function that you use to get the output of node. It is also known as Transfer Function.
- It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).
- The Activation Functions can be basically divided into 2 types:
  - Linear activation function
  - Non-linear activation function
    - The Nonlinear Activation Functions are mainly divided on the basis of their range or curves
    - Sigmoid or Logistic Activation Function
    - Tanh or hyperbolic tangent Activation Function
    - ReLU (Rectified Linear Unit) Activation Function
    - Leaky ReLU

Source of the text: Zöller, 2022.

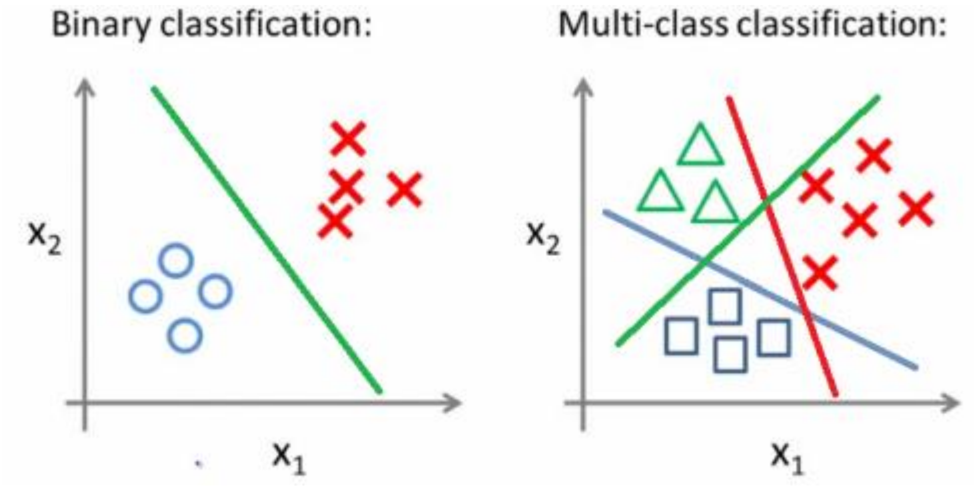
ACTIVATION FUNCTIONS

- Most people use ReLU as a reasonable default.
- Sometimes, you'll find ReLU and ELU benefit.
- ML is experimentation, not philosophy.
- Never use your mind to guess the output of a computer program.
-



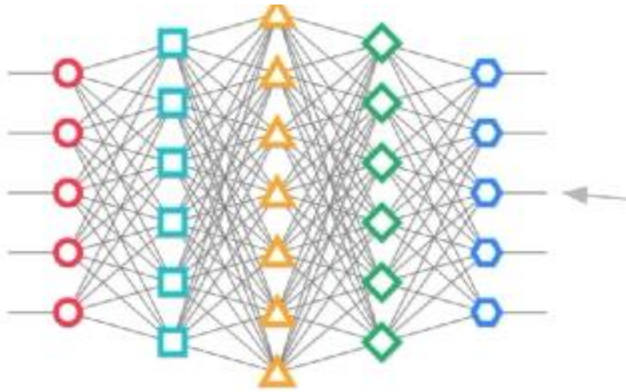
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

- For binary classification, we use a sigmoid at the output.
- We replaced sigmoids with ReLUs in the hidden layers.
- For output, sigmoid is still the right choice for binary classification.



https://utkuufuk.com/2018/06/03/one-vs-all-classification/

- Suppose we calculate the value right before applying the final activation function.
- $a^{(L)}$ is a vector of size K (for a K-class classifier)
- How do we turn this vector into a set of probabilities for each of the K classes?
  - $a^{(L)} = W^{(L)T} z^{(L-1)} + b^{(L)}$
- We need a probability distribution over K distinct values
  - Requirement 1: $p(y = k \mid x) \geq 0$
  - Requirement 2: $\sum_{k=1}^{K} p(y = k \mid x) = 1$



In this image, K=5

THE SOFTMAX FUNCTION

- Drop the superscript (L) for convenience.
- This function meets both of previous mentioned requirements.
  - Exp(any number) is positive
  - The denominator is the sum of all possible values of the numerator
    - $p(y = k \mid x) = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)}$
    - $\sum_{k=1}^{K} p(y = k \mid x) = \sum_{k=1}^{K} \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)} = 1$
  - The softmax is technically an activation function
  - CrossEntropyLoss already combines softmax + cross entropy

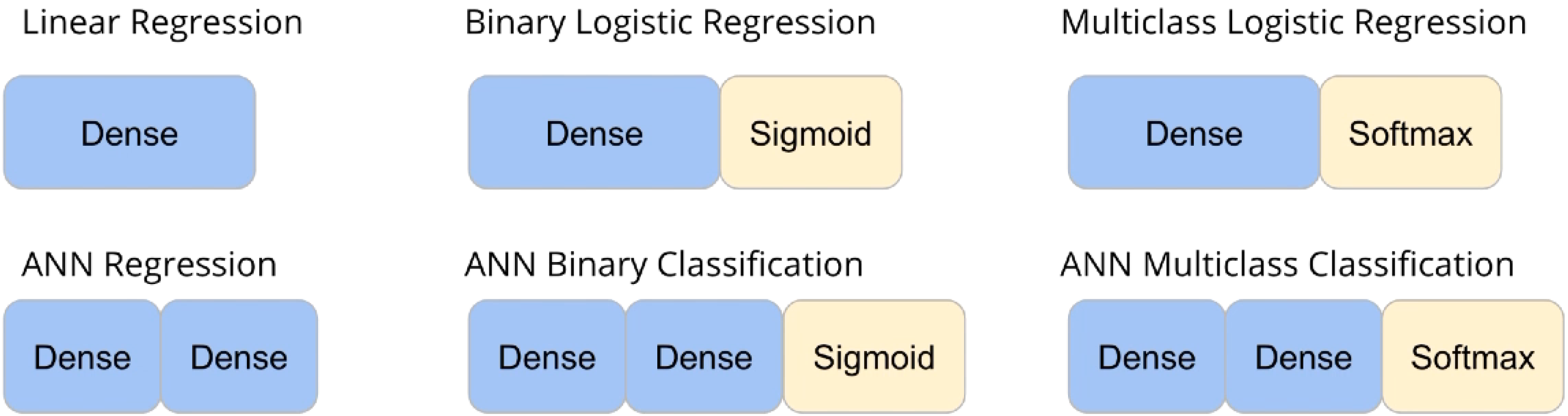
Task	Activation Function
Regression	None / Identity
Binary classification	Sigmoid
Multiclass classification	Softmax

```
model = nn.Linear(D, K)
criterion = nn.CrossEntropyLoss()
```

```
nn.Sequential(
  nn.Linear(D, M),
  nn.ReLU(),
  nn.Linear(M, K),
  nn.Softmax()
)
```

NETWORK DESIGN PATTERNS

- Same pattern applies to CNNs, RNNs – the type of task corresponds only to the final activation function



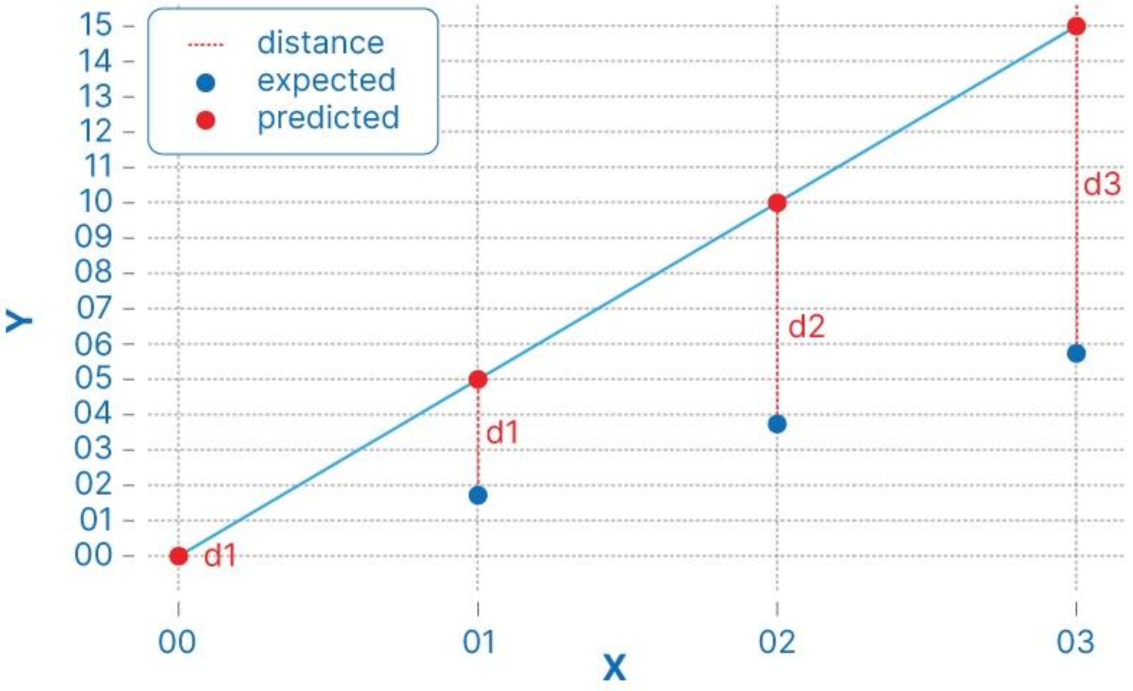
- Once a feed-forward neural network is trained, we are interested in investigating how well our model behaves when encountering input data that it have not been seen before. A cost function is a measure of “how well” the neural network does during the training phase.

$$C\left(W,\ B,\ S^{i},\ E^{i}\right)$$

where

- W denotes the set of connection weights of the network.
- B denotes the set of biases of the network.
- S^{i} denotes the input of a single training sample.
- E^{i} denotes the expected output of a sample i.

- Mean absolute error (MAE), Mean squared error (MSE), Root mean squared error (RMSE), Categorical cross-entropy, Binary cross-entropy



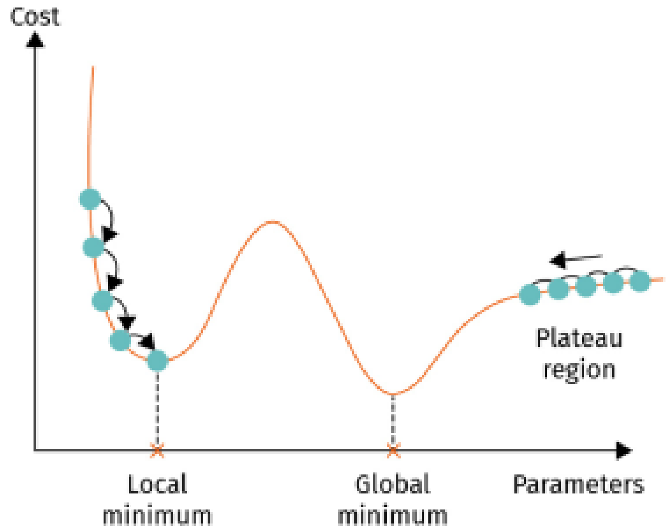
Source: https://www.shiksha.com/online-courses/articles/cost-function-in-machine-learning/

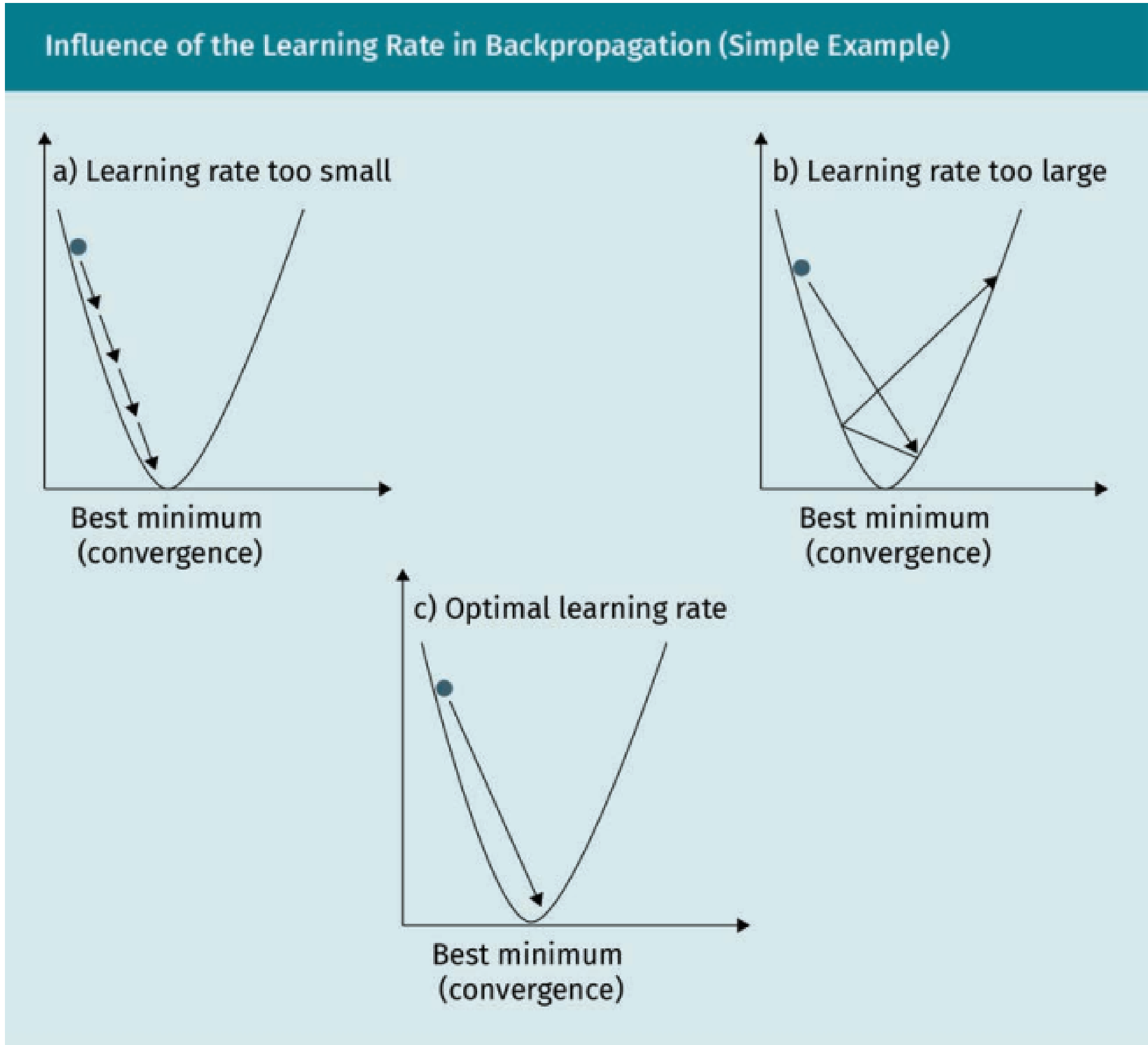
https://www.youtube.com/watch?app=desktop&v=gIx974WtVb4
https://www.youtube.com/watch?v=Md4b67HvmRo

- Recap of the steps
  - #1 Load in the data
  - #2 Build the model
  - #3 Train the model
  - #4 Evaluate the model
  - #5 Make predictions

OPTIMAL LEARNING RATE

- Learning rate α, which connects the derivatives calculated in the backward pass of the backpropagation algorithm to the actual change in the weights as the neural network is trained.
- Gradient descent pitfalls.

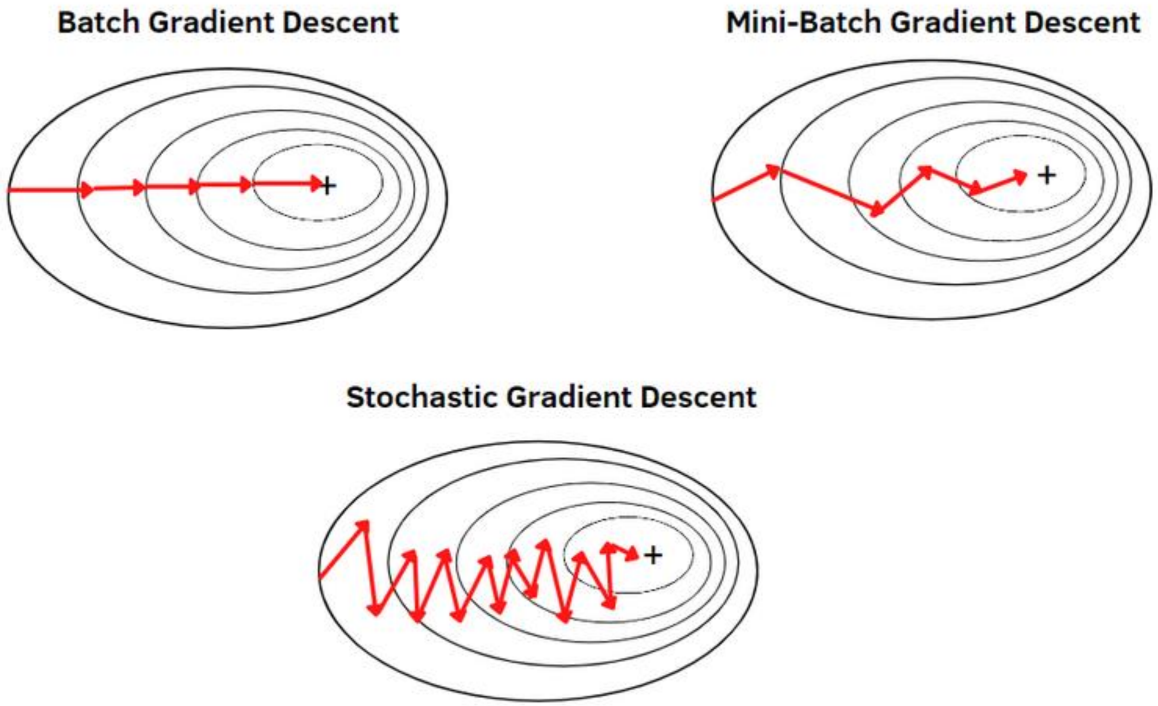




https://www.jeremyjordan.me/nn-learning-rate/
Source: Evis Plaku (2023), based on Géron (2019).

VARIANTS OF GRADIENT DESCENT ALGORITHM

- Batch gradient descent: computes the gradient of the cost function over the complete training set of input data, at every single step.
- Stochastic gradient descent: Instead of choosing the entire input dataset, only one instance is selected at random. Repeating this process long enough will allow the gradient to eventually get close to the minimum.
- Mini-batch gradient descent: A set of random instances known as mini-batch are used to compute the gradient.

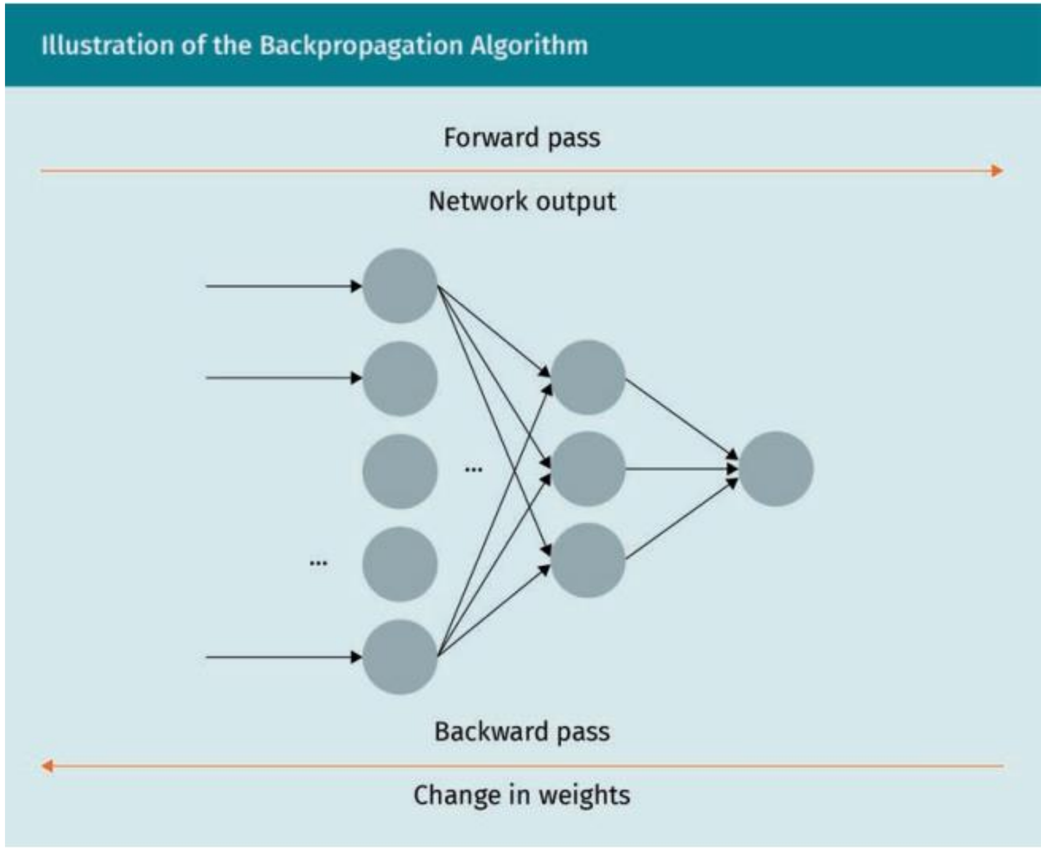


Source: https://www.analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/

BACKPROPAGATION AND GRADIENT DESCENT

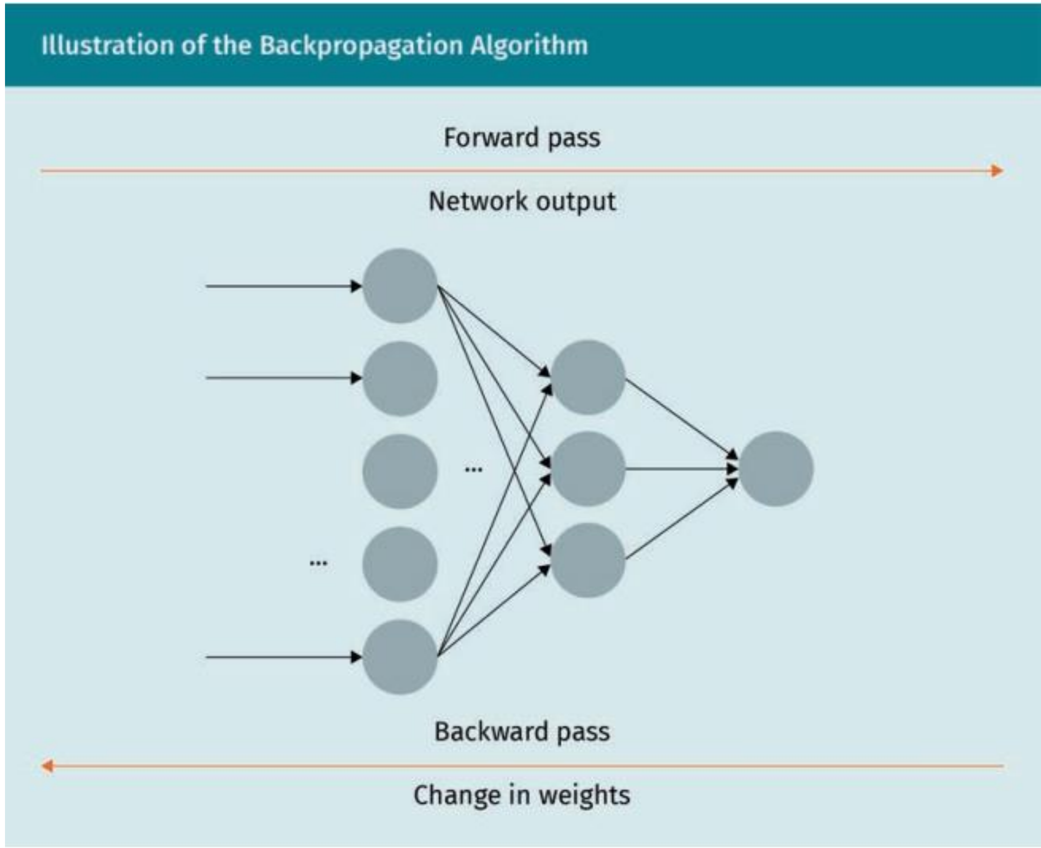
- The main idea of the backpropagation algorithm is to iteratively adjust the weights of the neural networks connecting the neurons in the various layers. The algorithm contains two main phases: a forward pass and a backward pass.

1. The algorithm starts by using one mini-batch at a time from the full set of training data. Weights are randomly initialized.
2. Computes outputs through layers.
3. The output error of the network is calculated using a loss function. Weights are then updated by an amount that is proportional to the calculated gradients.
4. This process is repeated until the cost function is minimized or a termination criterion is met.



BATCH NORMALIZATION

- The key objective of the training phase is to produce a model that generalizes well to new data. Training, however, can become highly sensitive to the initial distribution of connection weights.
- To address such issues, normalization is used as a vast category of techniques that aims to make samples that are fed to a machine learning model more similar to one another.
- Batch normalization is a normalization technique that continuously normalizes the input data during the training phase, even as the mean and variance of data changes during the training phase.



TRANSFER TASK

- Try the notebook 01.MLP_Regression.ipynb, 02. MLP_CIFAR10.ipynb

- After completing this unit you will be able to ...
  – understand what a feed-forward neural network is.
  – distinguish between different network topologies and cost functions.
  – explain how the backpropagation and gradient descent algorithm help a neural network to train a model.
  – build a simple feed-forward neural network used for classifying image data.
  – describe how batch normalization can improve the performance of a network

SESSION 1

Feed-forward Networks

DISCLAIMER

- This is the modified version of the IU slides.
- I used it for my lectures at IU only.