Introduction to Deep Learning for Computer Vision

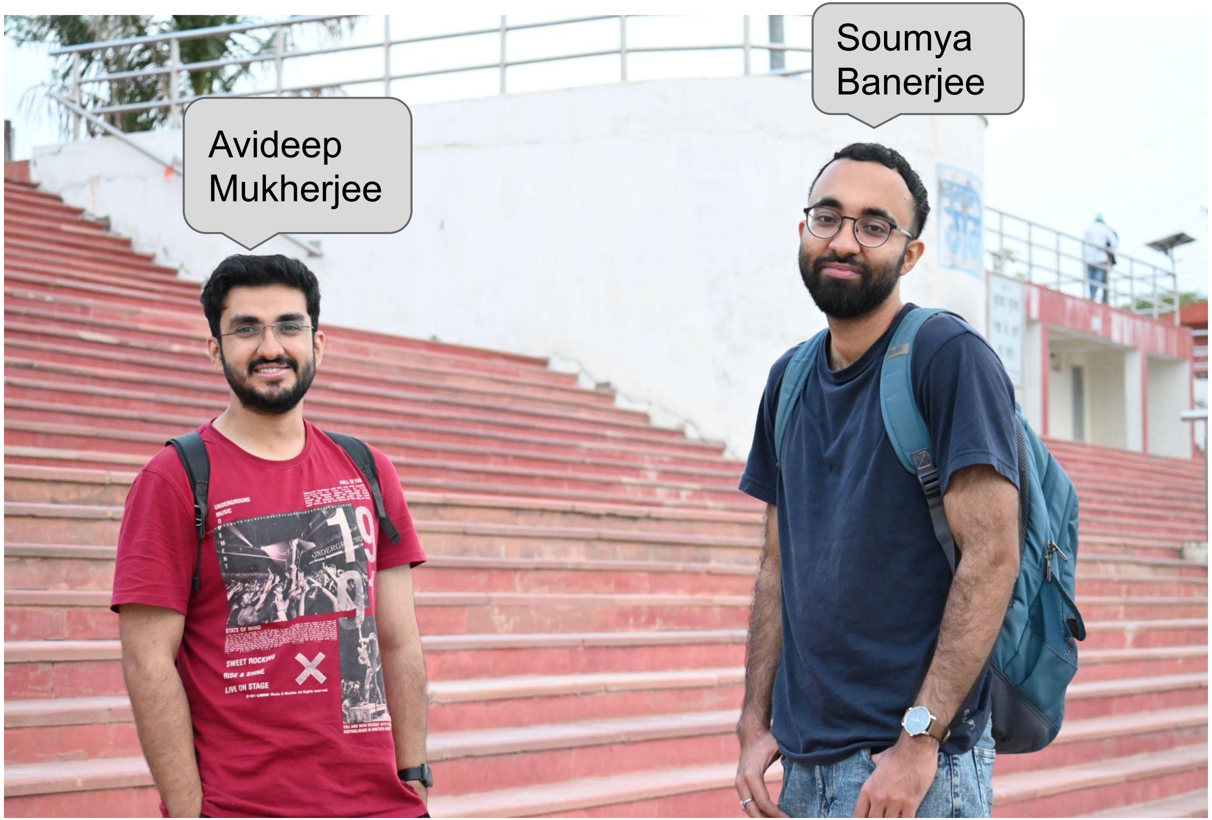
Adhyayan '23 - ACA Summer School
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Lecture 1

Instructors

Instructors

Instructors



Credits:

- http://introtodeeplearning.com/
- https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55
- https://0space.org/c/2098-machine-learning-vs-deep-learning-examples-and-use-case
- http://beamlab.org/deeplearning/2017/02/23/deep_learning_101_part1.html
- https://anjali-dl.blogspot.com/2020/03/importance-of-activation-functions.html

What is Deep Learning?

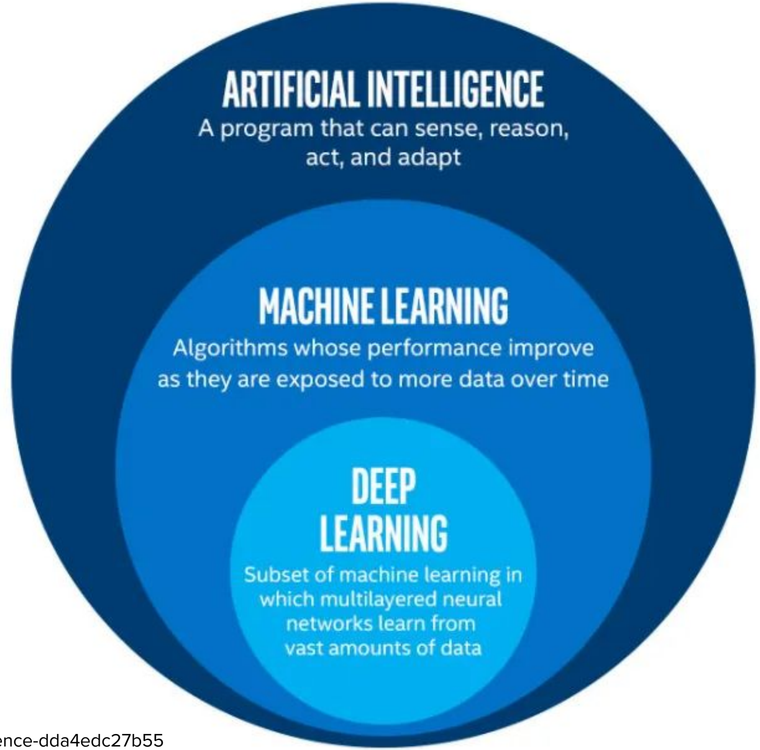


Image: https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55

Lecture Schedule

- Week 1:
  - Perceptron, Multi-layer Perceptron, Activation and Loss Functions. Python and Numpy hands-on demo.
  - Backpropagation, Batch Gradient Descent, SGD, Mini-batch SGD. Regularisation and Optimization. Introduction to PyTorch hands-on demo.
  - Convolutional Neural Networks
  - Popular CNN architectures. CNN Hands-on demo with PyTorch.
  - Training NNs: Weight Init, Dropout, Learning Rate Scheduling, Early Stopping, Weight Decay, Data Augmentation and Normalization, Batch Norm.

Lecture Schedule

- Week 1:
  - Perceptron, Multi-layer Perceptron, Activation and Loss Functions. Python and Numpy hands-on demo.
  - Backpropagation, Batch Gradient Descent, SGD, Mini-batch SGD. Regularisation and Optimization. Introduction to PyTorch hands-on demo.
  - Convolutional Neural Networks
  - Popular CNN architectures. CNN Hands-on demo with PyTorch.
  - Training NNs: Weight Init, Dropout, Learning Rate Scheduling, Early Stopping, Weight Decay, Data Augmentation and Normalization, Batch Norm.
- Week 2:
  - Object Detection (R-CNN, Yolov3), Image Segmentation (FCN, U-Net)
  - Unsupervised Learning and Generative Modelling: Autoencoder, VAE
  - Self-Attention & ViT
  - Adversarial Autoencoders, GANs, Diffusion (very brief overview)
  - Assorted Topics: Self Supervised Learning (SimSiam, Contrastive Learning, Rotation Loss), Active learning

Grading Policy

- 1 Quiz after Week 1 (comprising of Week 1 syllabus) - 50% weight.
- 1 Quiz after Week 2 (comprising of whole syllabus) - 50% weight.

Why Deep Learning? And Why Now?

Why Deep Learning?

- Handcrafted features are expensive to engineer, delicate and unsuitable for scaling.
- Deep Learning attempts to *learn* the fundamental features directly from data.

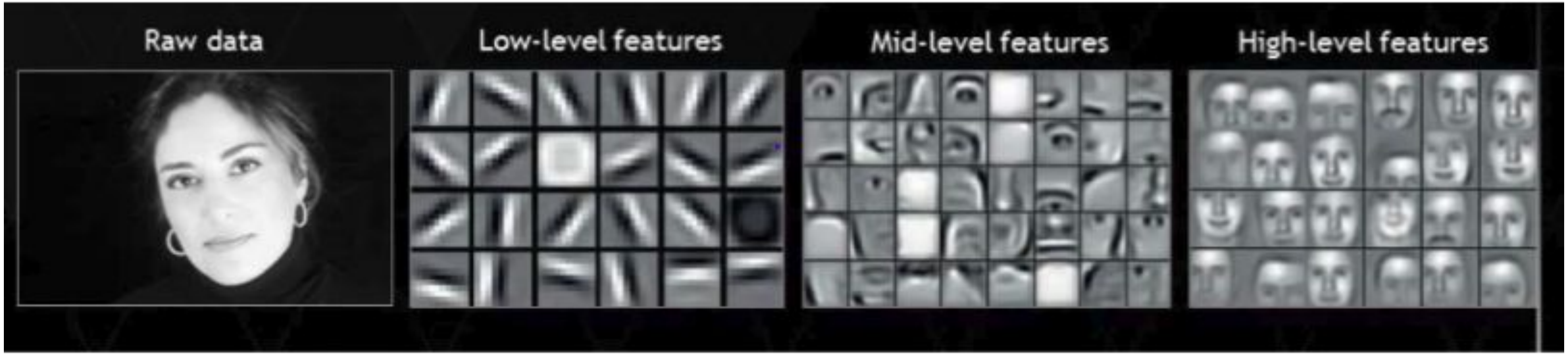


Image: https://0space.org/c/2098-machine-learning-vs-deep-learning-examples-and-use-case

Why Now?

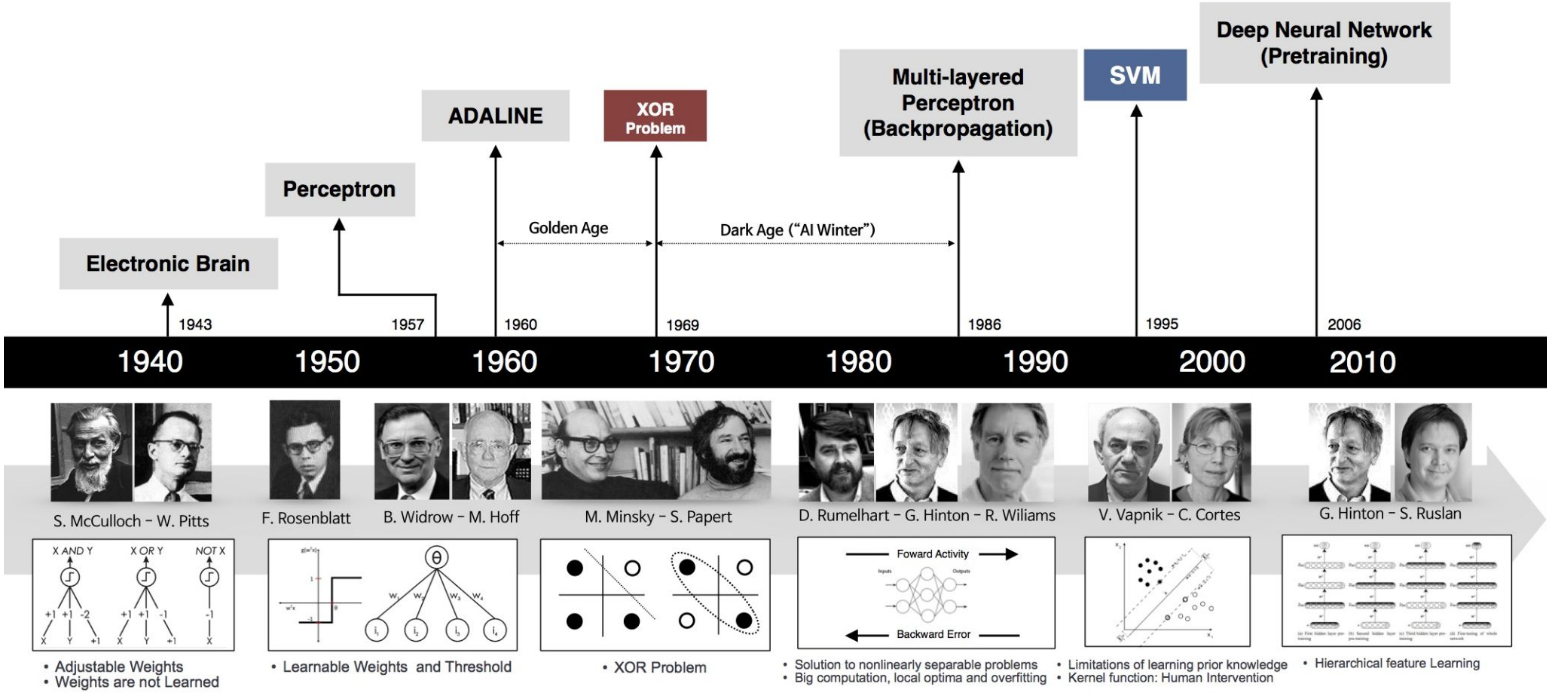


Image: http://beamlab.org/deeplearning/2017/02/23/deep_learning_101_part1.html

Why Now?

- Big Data:
  - Larger datasets.
  - Easier storage facility.

Why Now?

- Big Data:
  - Larger datasets.
  - Easier storage facility.
- Hardware:
  - GPUs!
  - Massively Parallelizable.

Why Now?

- Big Data:
  - Larger datasets.
  - Easier storage facility.
- Hardware:
  - GPUs!
  - Massively Parallelizable.
- Software:
  - Improved Techniques
  - Better Models
  - Better Frameworks

Perceptrons

The Perceptron

- Structural building block of deep learning.

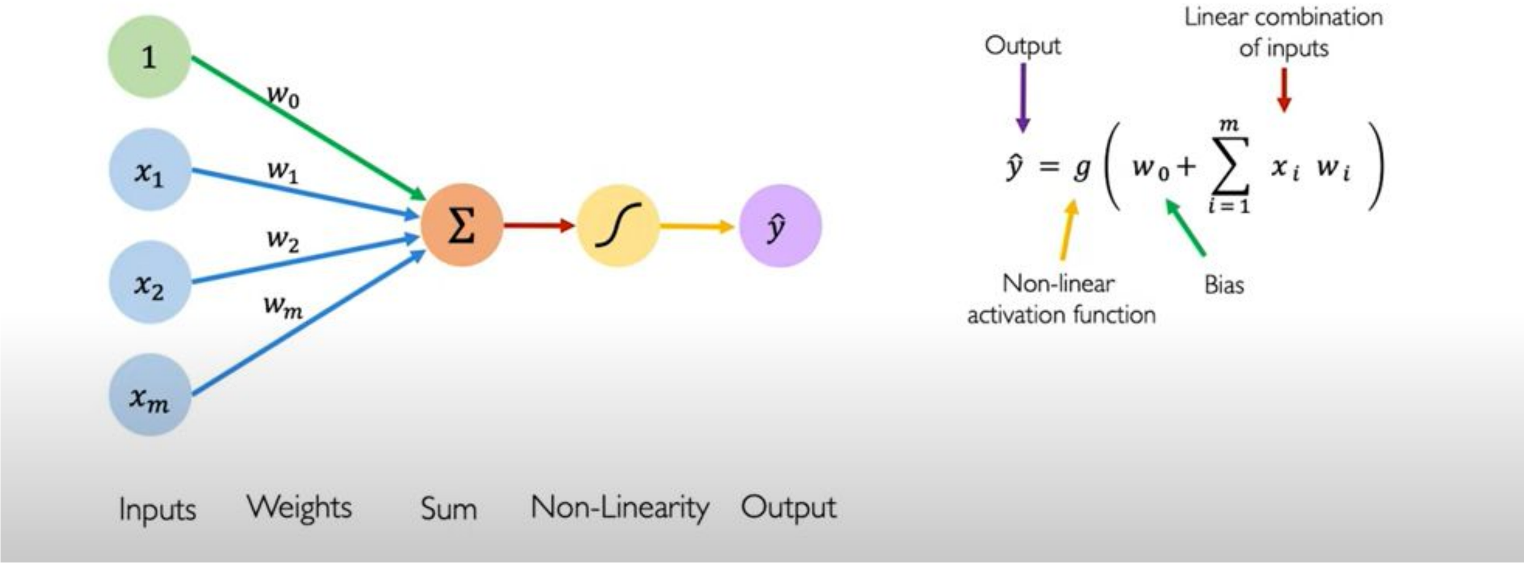
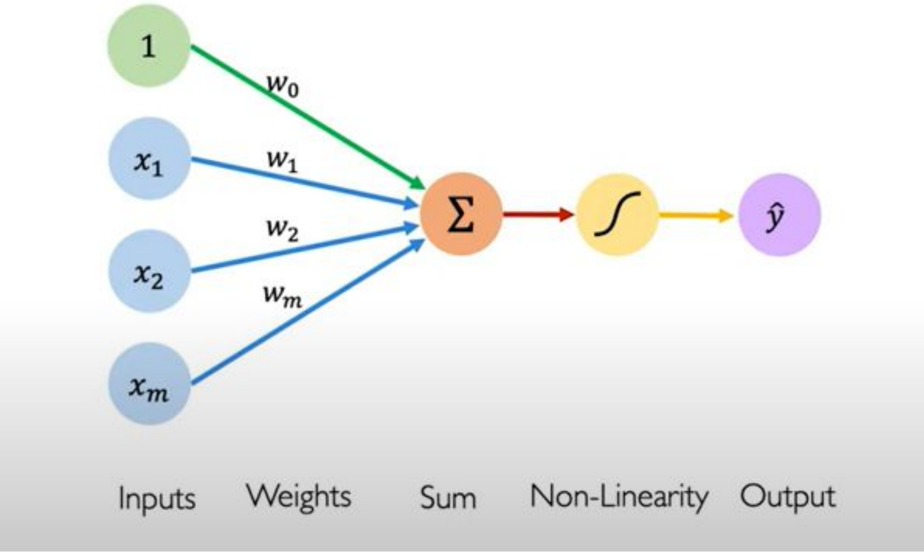


Image: https://vitalflux.com/perceptron-explained-using-python-example/

The Perceptron

- Structural building block of deep learning.



Output

Linear combination of inputs

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i \, w_i \right)$$

Non-linear activation function

Bias

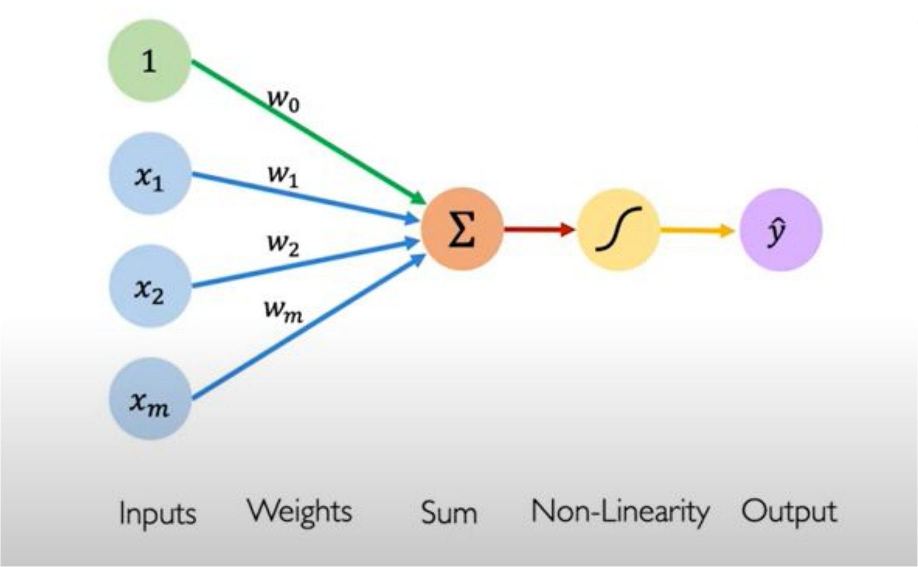
$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W} \right)$$

$$\text{where: } \boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \text{ and } \boldsymbol{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

Image: https://vitalflux.com/perceptron-explained-using-python-example/

The Perceptron

- Structural building block of deep learning.



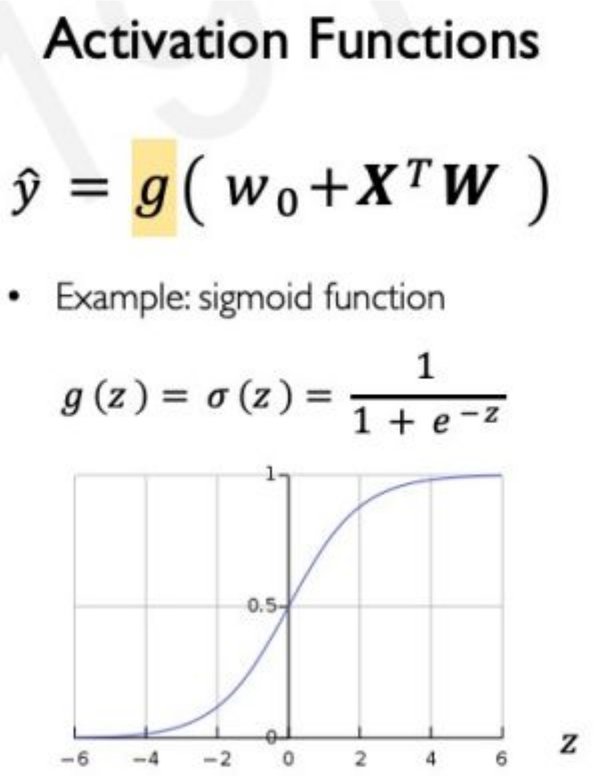
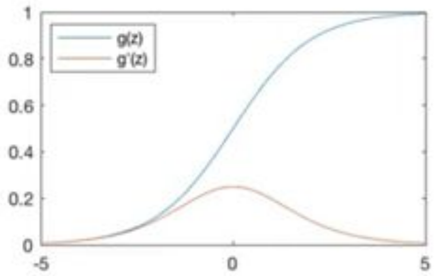


Image: https://vitalflux.com/perceptron-explained-using-python-example/

Common Activation Functions

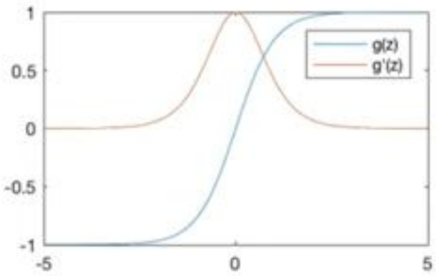
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

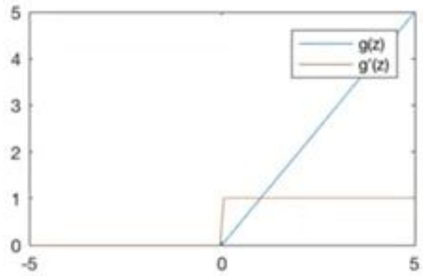
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)

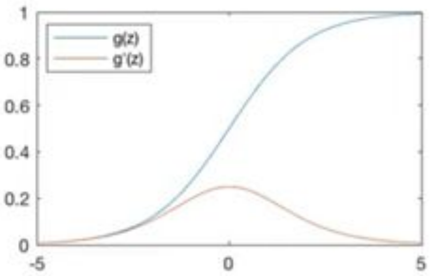


$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Common Activation Functions

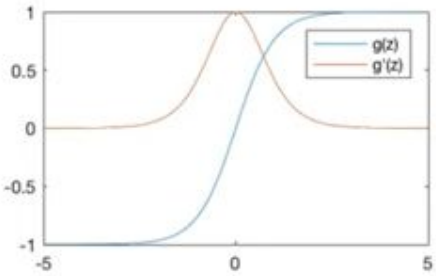
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

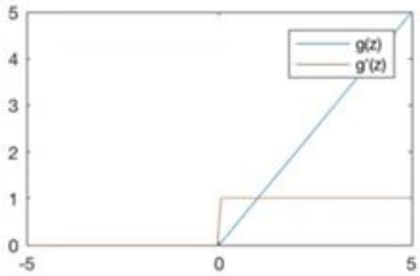
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)



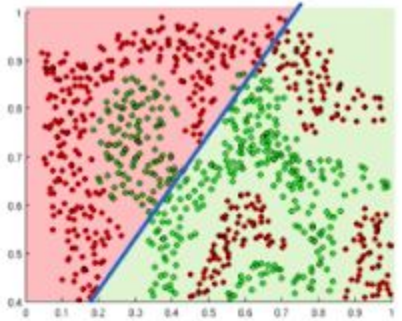
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

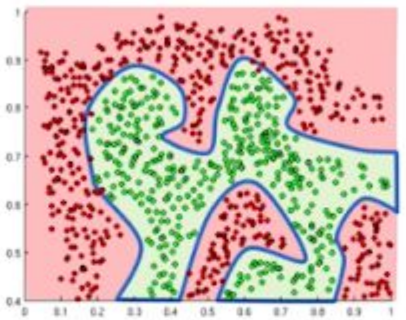
Note: All Activation functions are non-linear.

Importance of Activation Functions

- The purpose of Activation Functions is to introduce non-linearities in the network.



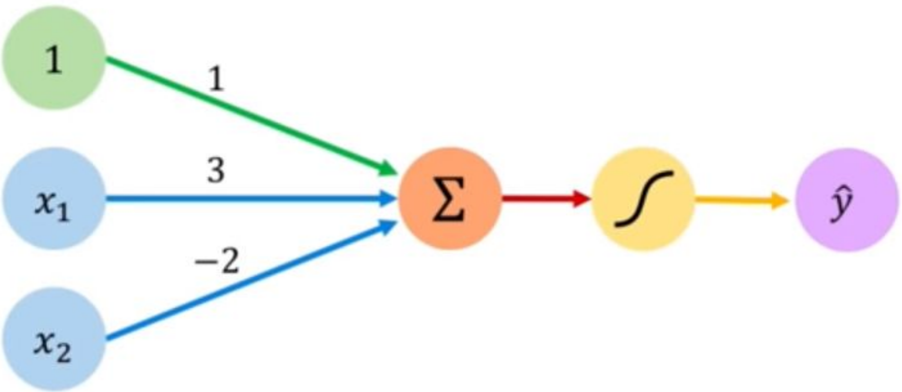
Linear activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

Image: https://anjali-dl.blogspot.com/2020/03/importance-of-activation-functions.html

Perceptron: Example

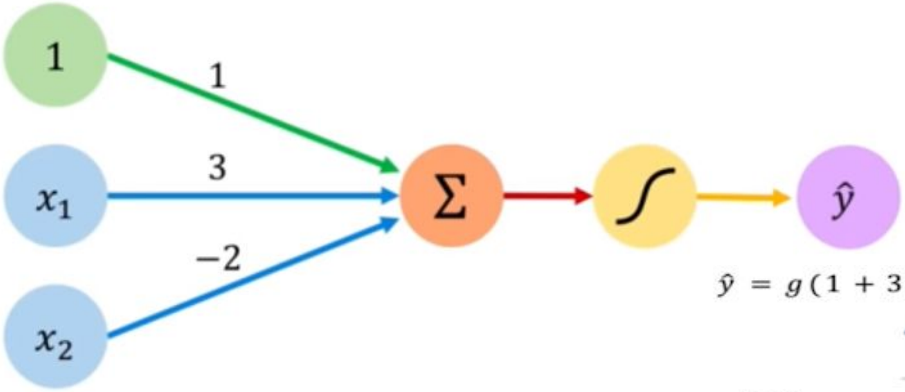


We have: $w_0 = 1$ and $\boldsymbol{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$
$$\hat{y} = g(\underbrace{1 + 3x_1 - 2x_2}_{\text{This is just a line in 2D!}})$$

Image: https://anjali-dl.blogspot.com/2020/03/importance-of-activation-functions.html

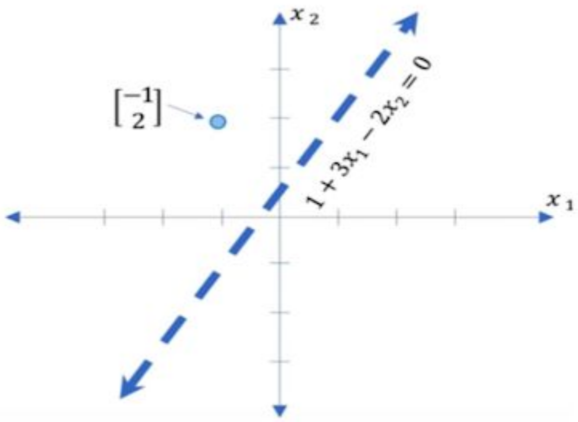
Perceptron: Example



We have: $w_0 = 1$ and $\boldsymbol{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W}\right)$$
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$

$$\hat{y} = g(1 + 3x_1 - 2x_2)$$

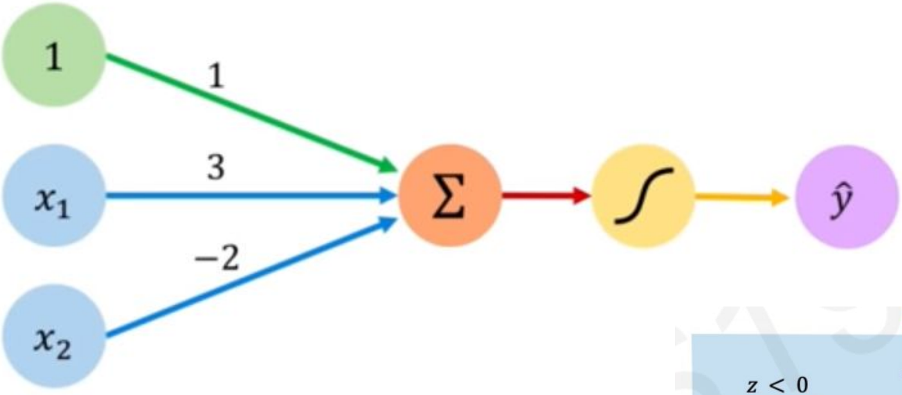


Assume we have input: $\boldsymbol{X} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

$$\hat{y} = g\left(1 + (3 * -1) - (2 * 2)\right)$$
$$= g(-6) \approx 0.002$$

Image: https://anjali-dl.blogspot.com/2020/03/importance-of-activation-functions.html

Perceptron: Example



We have: $w_0 = 1$ and $\boldsymbol{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W}\right)$$
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$
$$\hat{y} = g(\underbrace{1 + 3x_1 - 2x_2}_{\text{This is just a line in 2D!}})$$

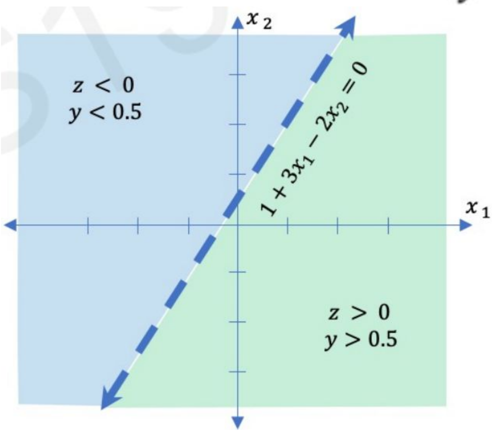
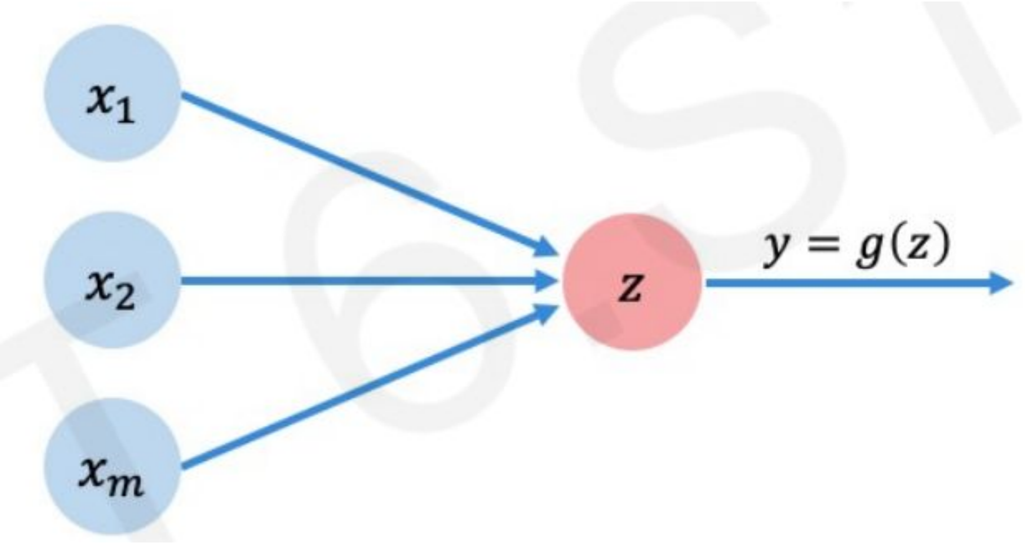


Image: https://anjali-dl.blogspot.com/2020/03/importance-of-activation-functions.html

Building Neural Networks with Perceptrons!

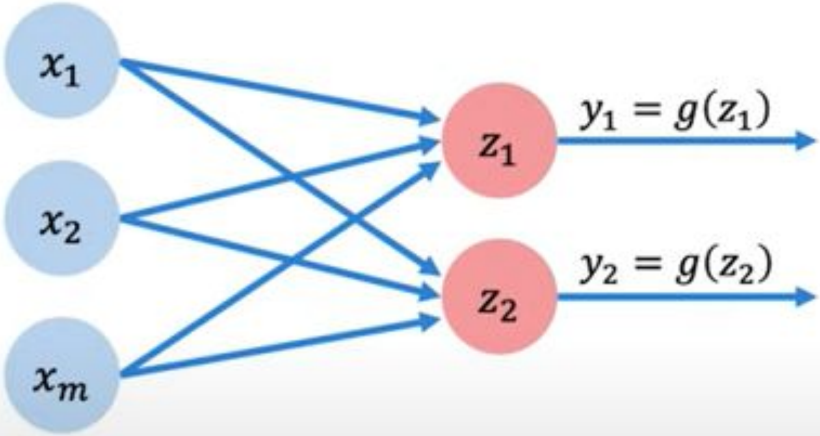
Perceptron: Simplified



$$z = w_0 + \sum_{j=1}^{m} x_j w_j$$

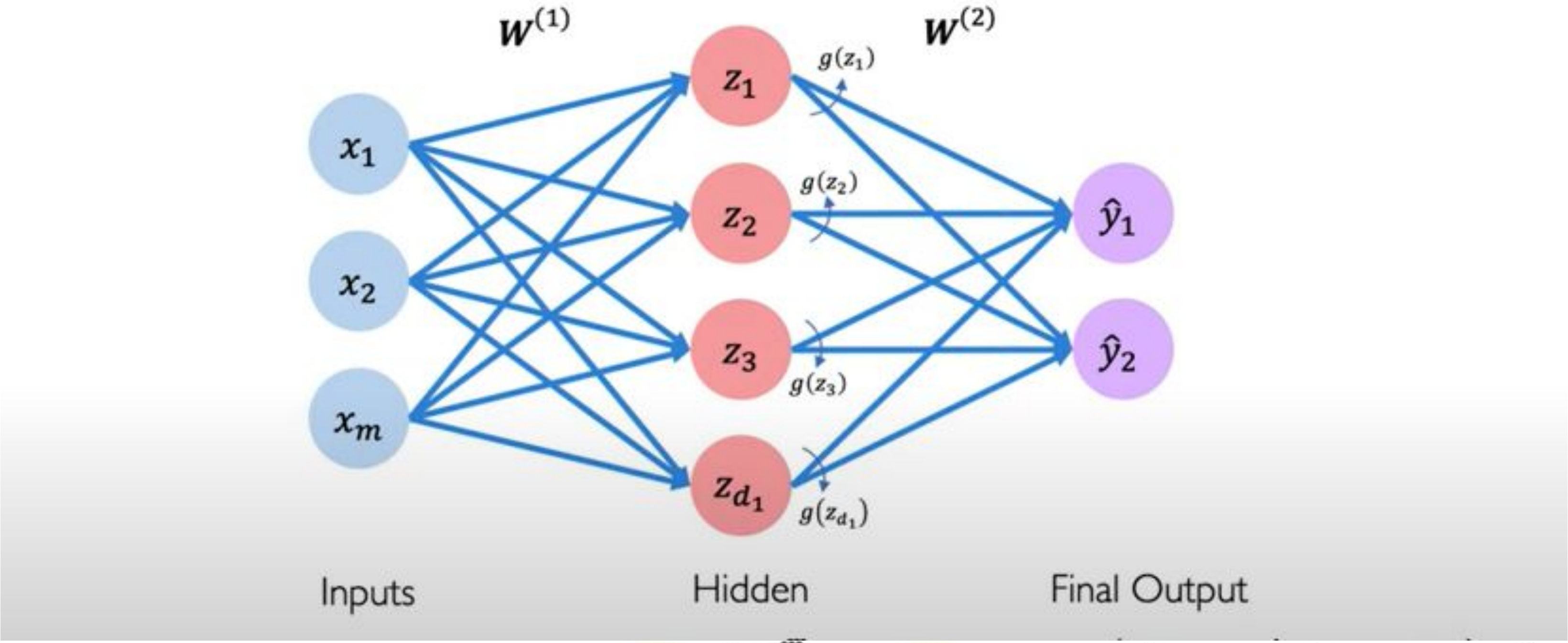
Multi Output Perceptron

Because all inputs are densely connected to all outputs, these layers are called **Dense** Layers.



$$z_i = w_{0,i} + \sum_{j=1}^{m} x_j \, w_{j,i}$$

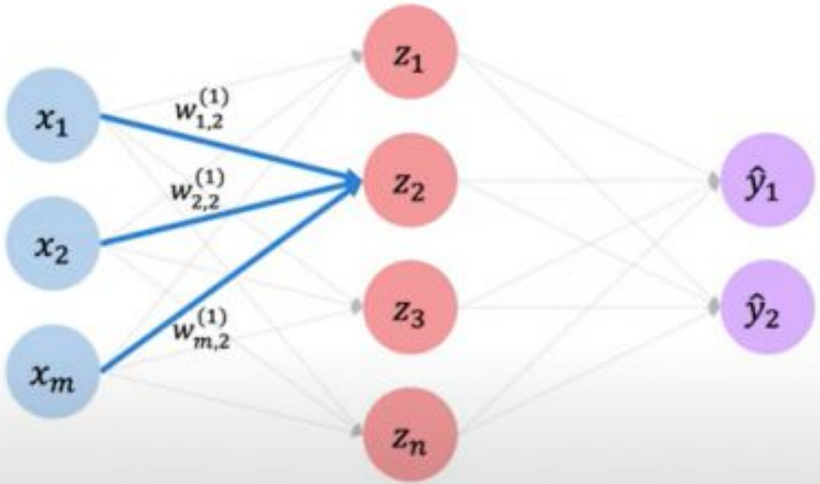
Single Layer Neural Network



$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^{m} x_j \, w_{j,i}^{(1)}$$

$$\hat{y}_i = g\left(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} g(z_j)\, w_{j,i}^{(2)}\right)$$

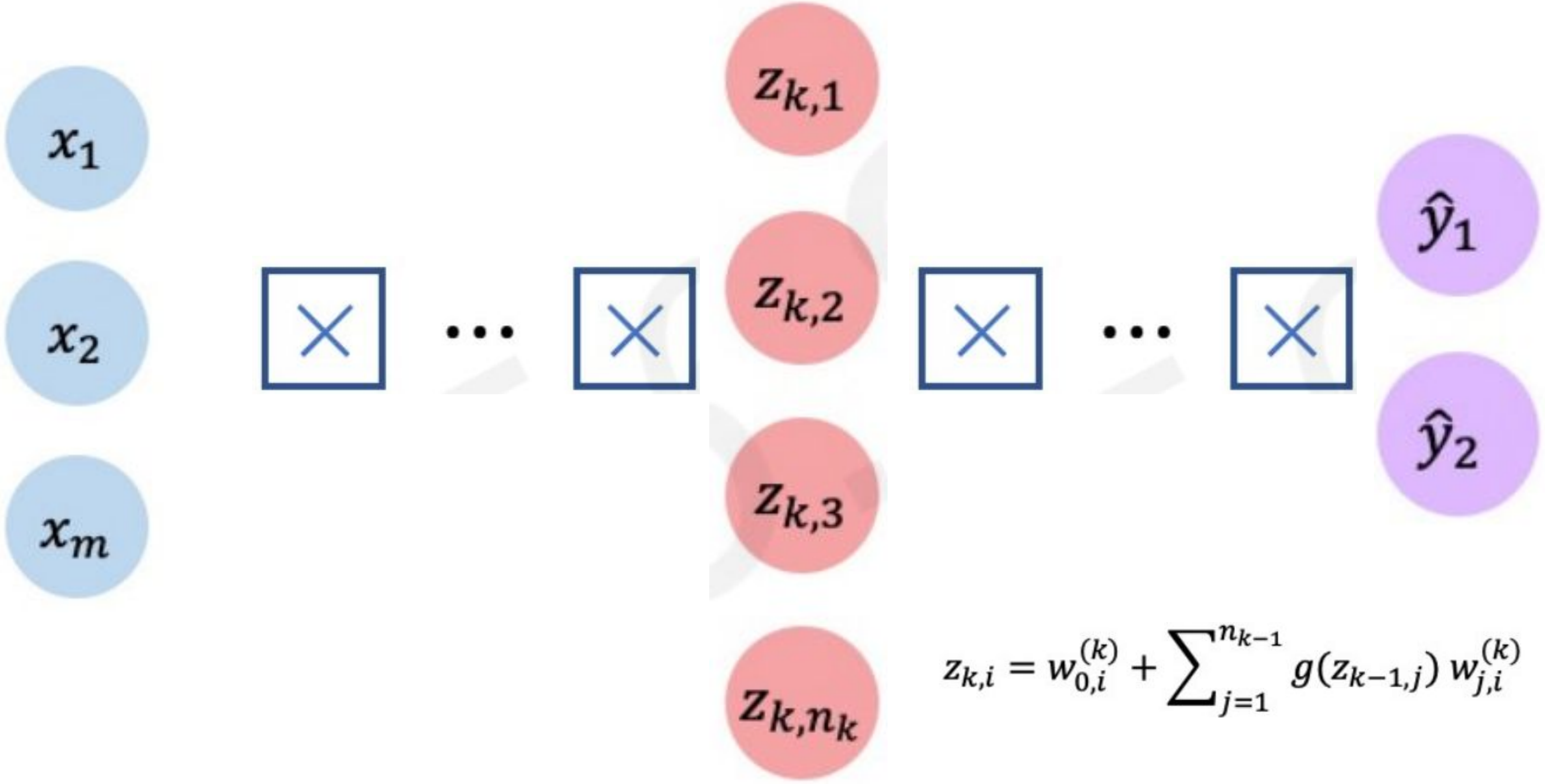
Single Layer Neural Network



$$z_2 = w_{0,2}^{(1)} + \sum_{j=1}^{m} x_j \, w_{j,2}^{(1)}$$

$$= w_{0,2}^{(1)} + x_1 \, w_{1,2}^{(1)} + x_2 \, w_{2,2}^{(1)} + x_m \, w_{m,2}^{(1)}$$

Deep Neural Network



$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{n_{k-1}} g(z_{k-1,j})\, w_{j,i}^{(k)}$$

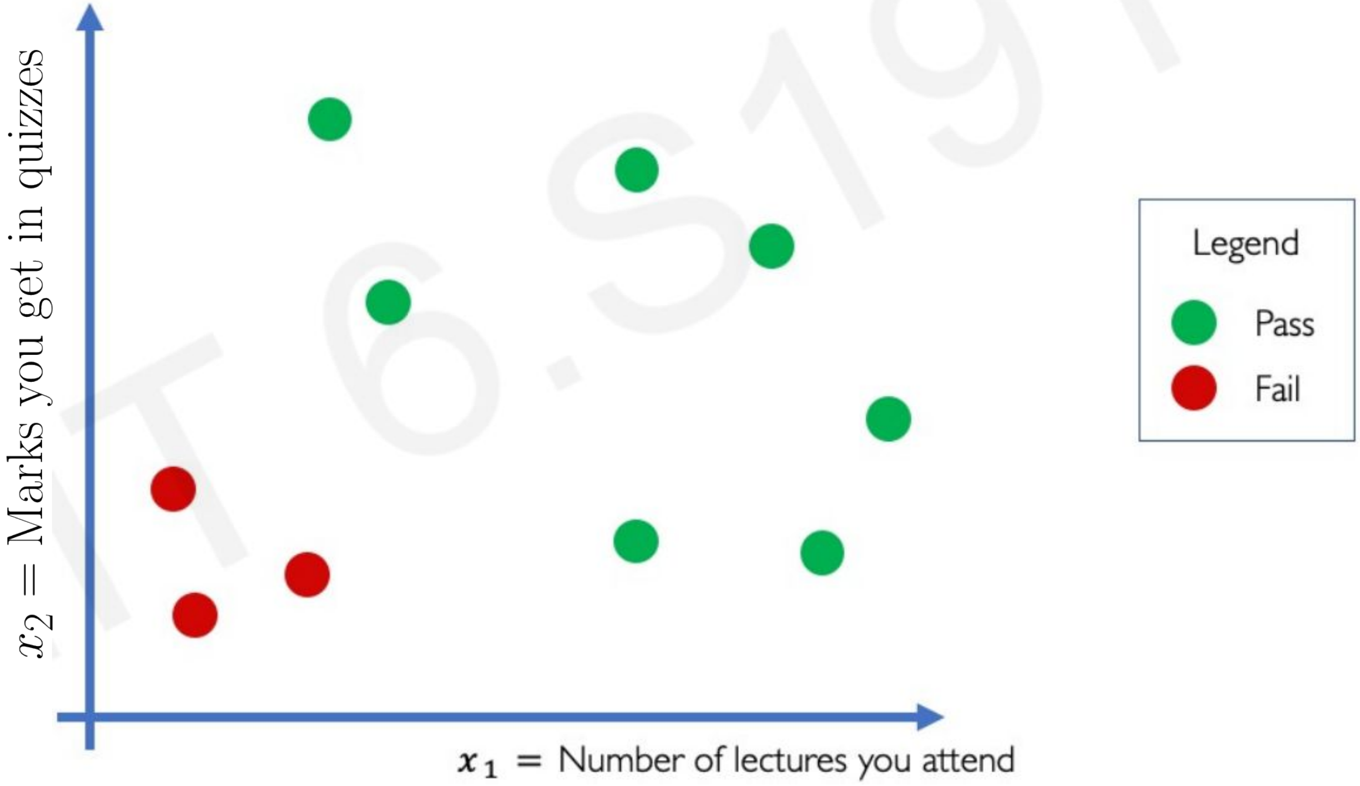
Applying Neural Networks

Example Problem

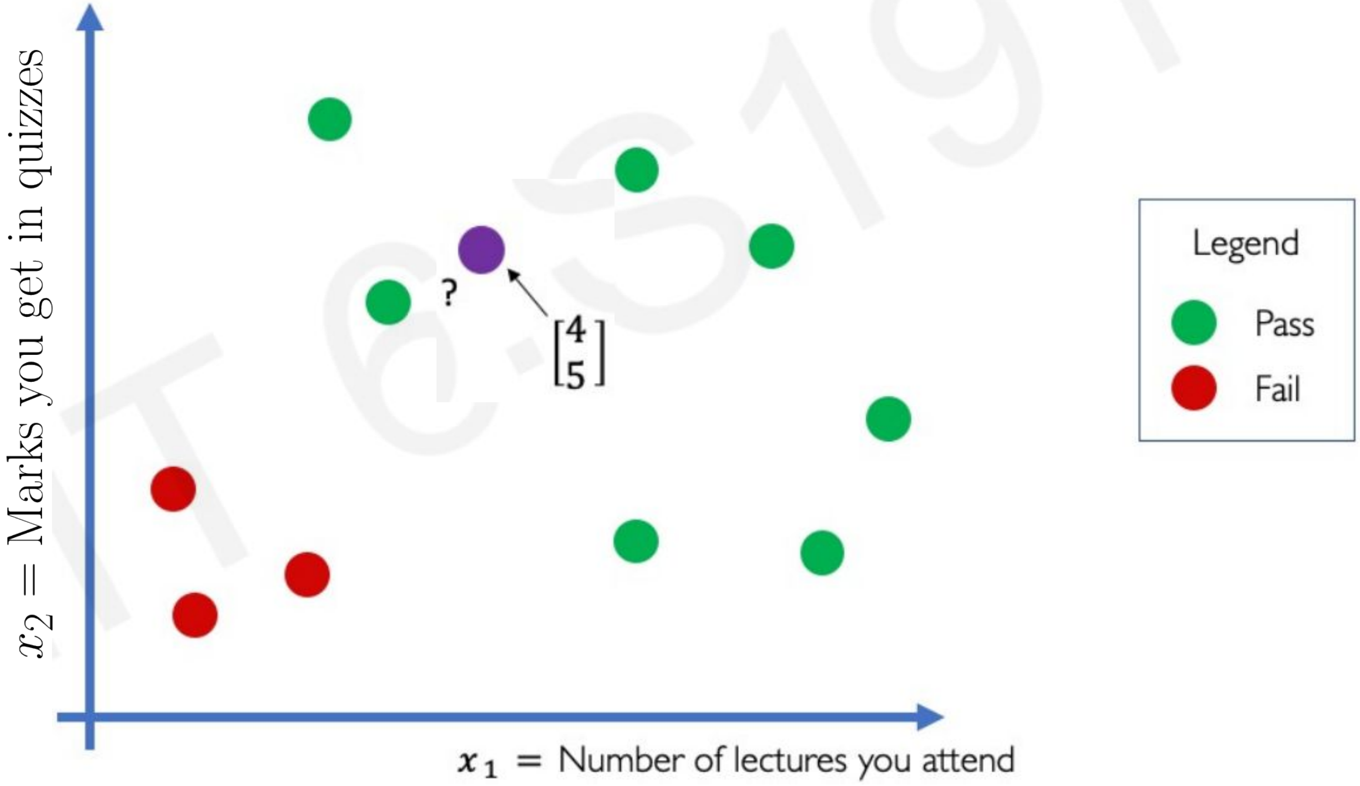
- Will I pass this course?
- Let us start with a simple two-feature model.

$$x_1 = \text{Number of Lectures you attend}$$
$$x_2 = \text{Marks you get in quizzes}$$

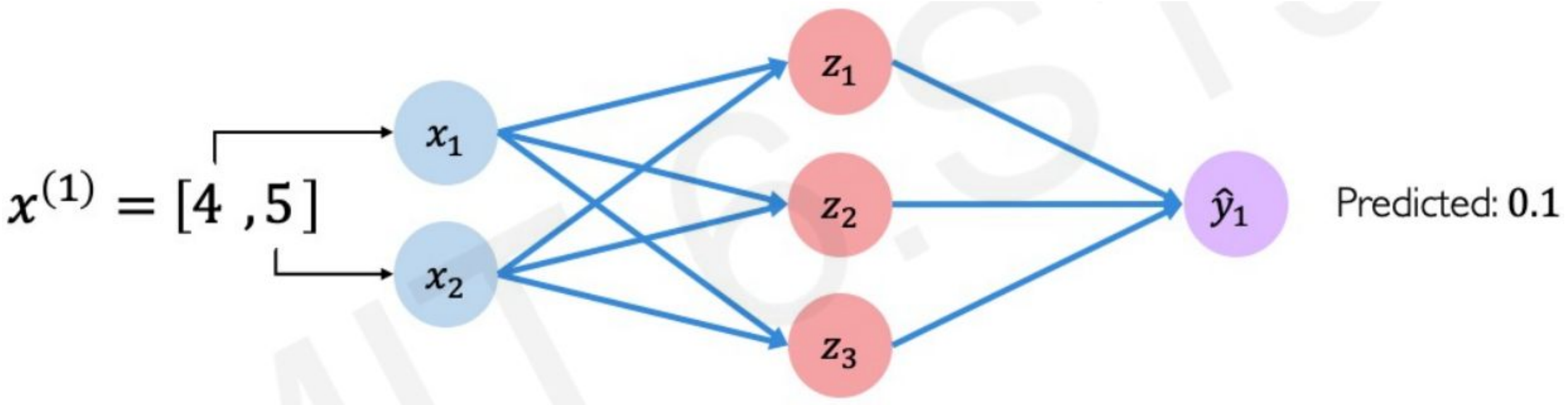
Example Problem: Will I pass this course?



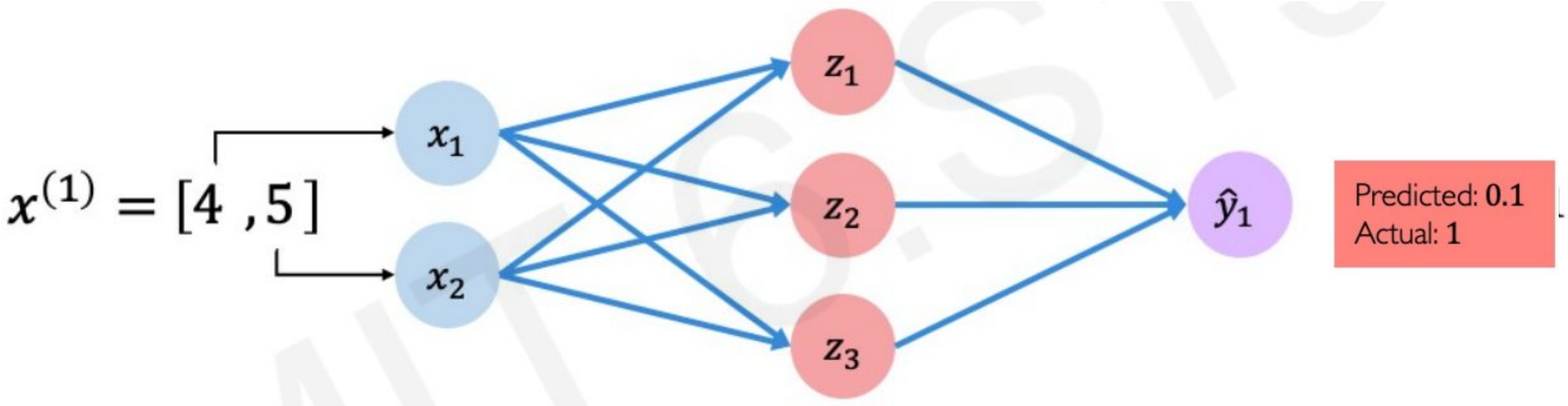
Example Problem: Will I pass this course?



Example Problem: Will I pass this course?

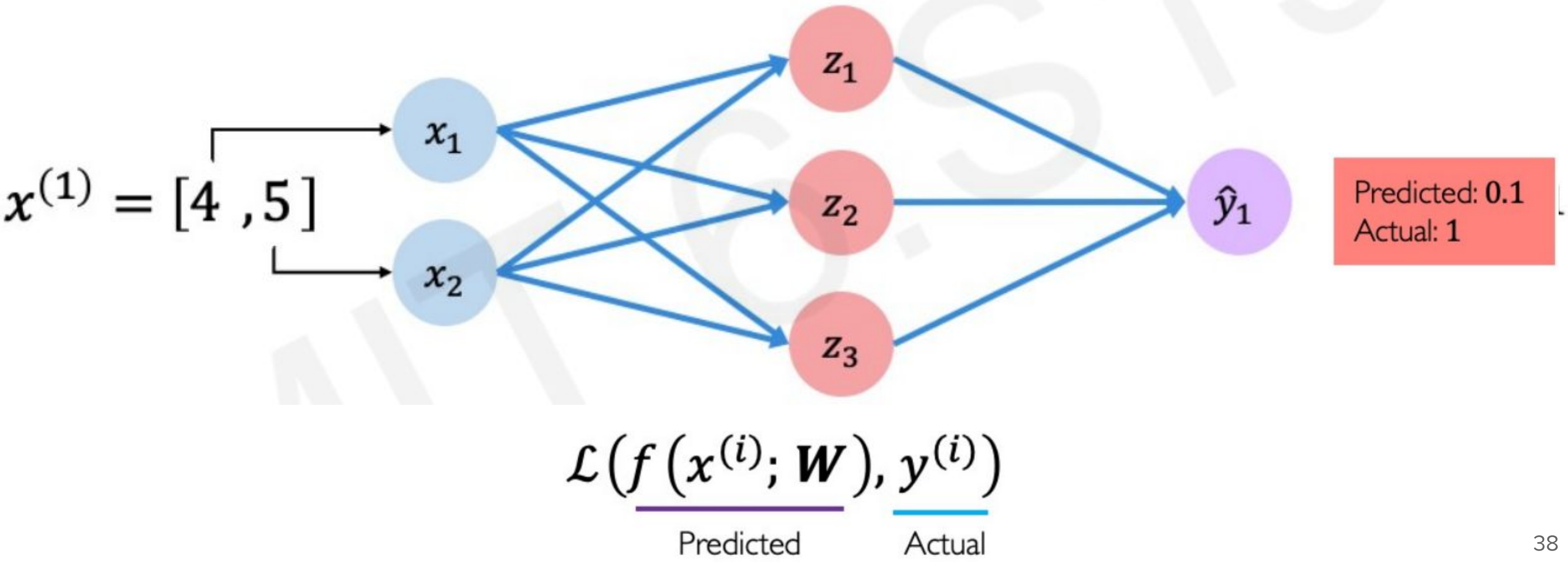


Example Problem: Will I pass this course?



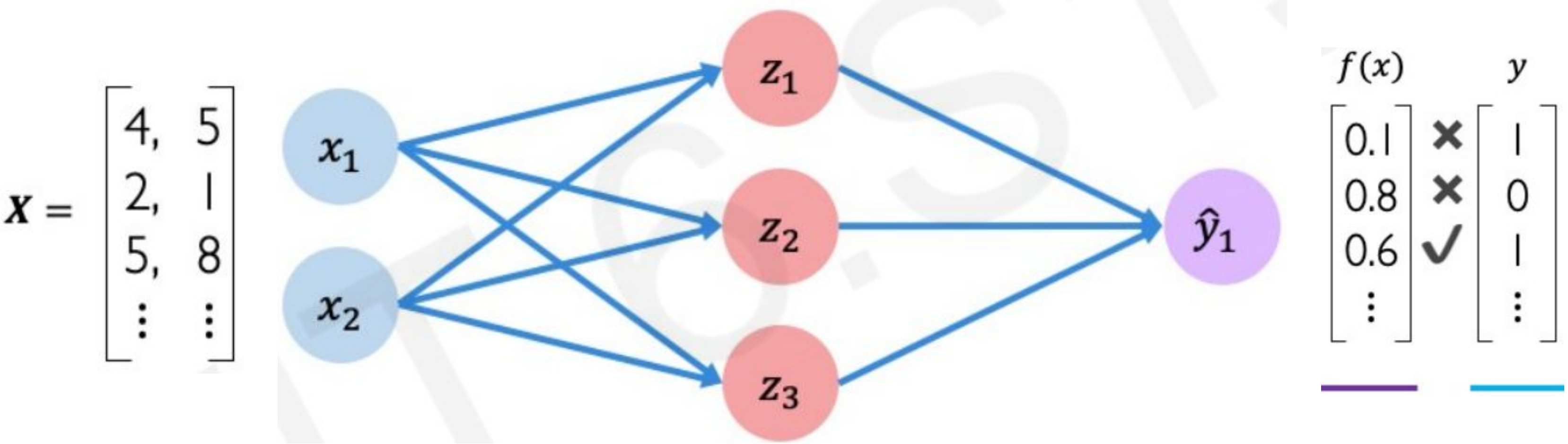
Quantifying the Loss

- The **Loss** of our Network measures the cost incurred from incorrect predictions.



Empirical Loss

- The **Empirical Loss** measures the total loss over our entire dataset.



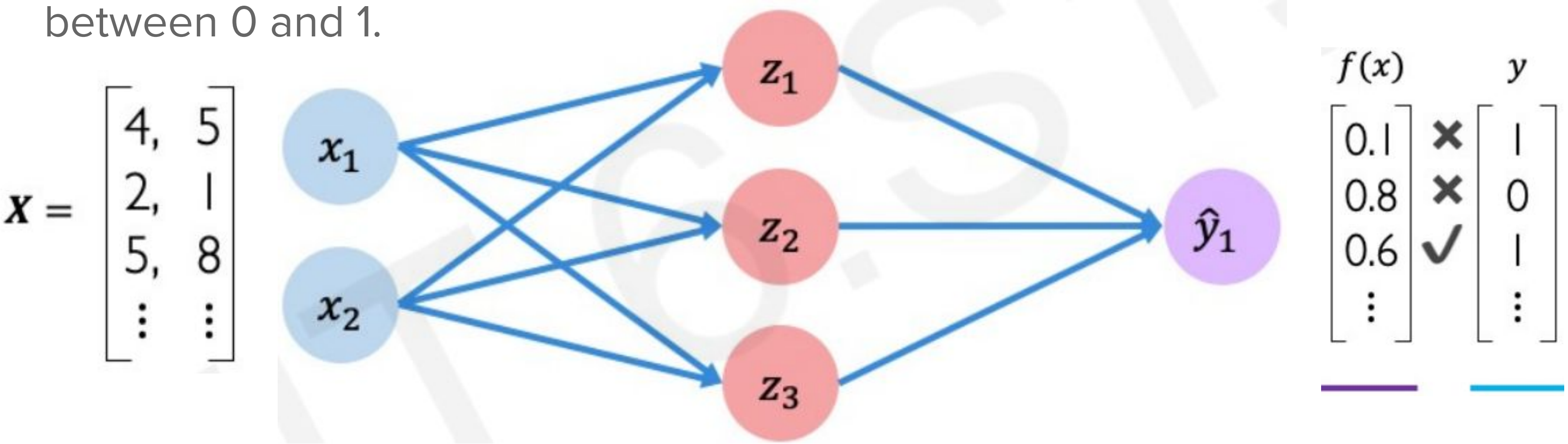
$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}},\underbrace{y^{(i)}}_{\text{Actual}}\big)$$

Also known as:

- Objective Function
- Cost Function
- Empirical Risk

Binary Cross Entropy Loss

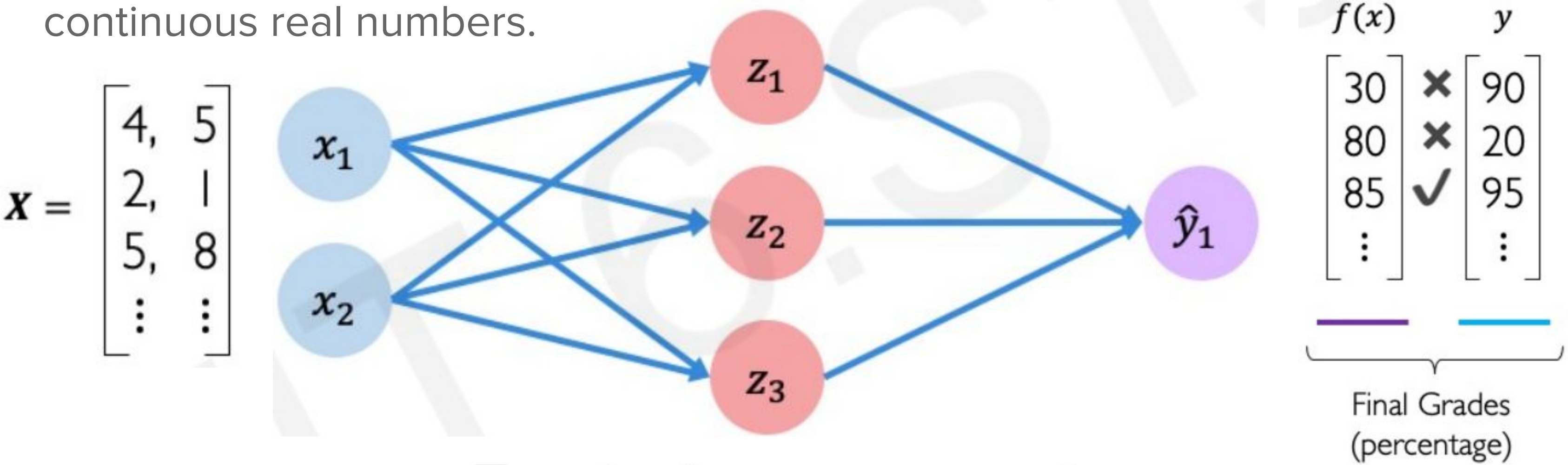
- **Cross Entropy Loss** can be used with Models that output a probability between 0 and 1.



$$J(\boldsymbol{W}) = -\frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\Big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big) + (1 - \underbrace{y^{(i)}}_{\text{Actual}}) \log\Big(1 - \underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big)$$

Mean Squared Error Loss

- **Mean squared error loss** can be used with regression models that output continuous real numbers.



$$J(W) = \frac{1}{n}\sum_{i=1}^{n}\Big(\underbrace{y^{(i)}}_{\text{Actual}} - \underbrace{f\big(x^{(i)}; W\big)}_{\text{Predicted}}\Big)^2$$

Next Lecture: Training Neural Networks!