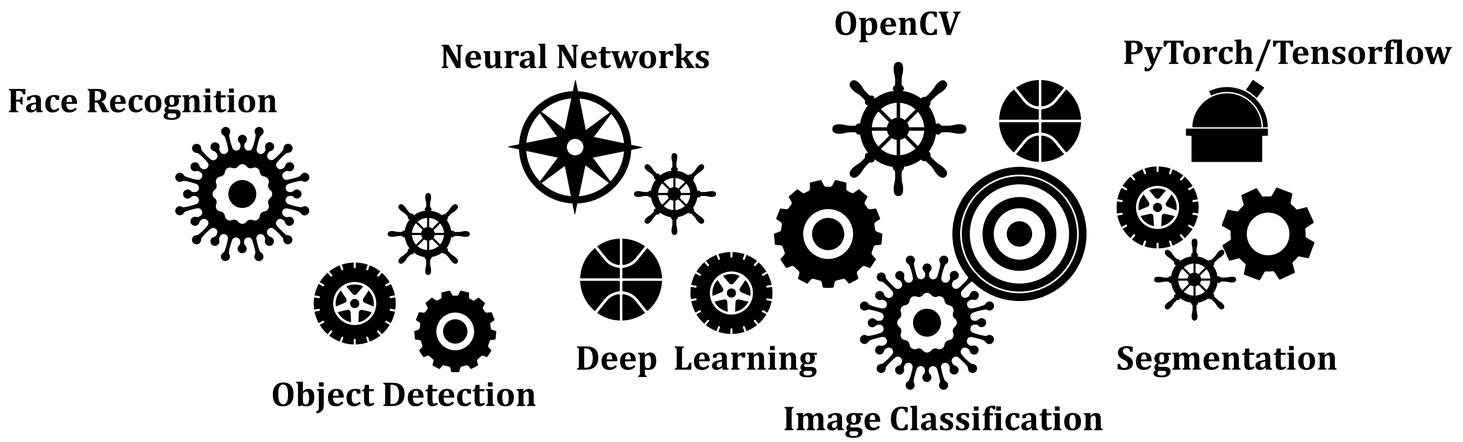
MPSSDEGB
कौशल से रोजगार, समृद्धि अपार

FOUNDATION FOR INNOVATION AND TECHNOLOGY TRANSFER
भारतीय प्रौद्योगिकी संस्थान दिल्ली
Indian Institute of Technology Delhi

Computer Vision



Artificial Neural Networks

- "Neural" is an adjective for neuron, and "network" denotes a graph like structure.
- Artificial Neural Networks are also referred to as "neural nets" , "artificial neural systems", "parallel distributed processing systems", "connectionist systems".

- For a computing systems to be called by these pretty names, it is necessary for the system to have a labeled directed graph structure where nodes performs some simple computations.
- "Directed Graph" consists of set of "nodes"(vertices) and a set of "connections"(edges/links/arcs) connecting pair of nodes.
- A graph is said to be "labeled graph" if each connection is associated with a label to identify some property of the connection

https://www.slideshare.net/DEEPASHRIHK/artificial-neural-network-18449085

Artificial Neural Networks

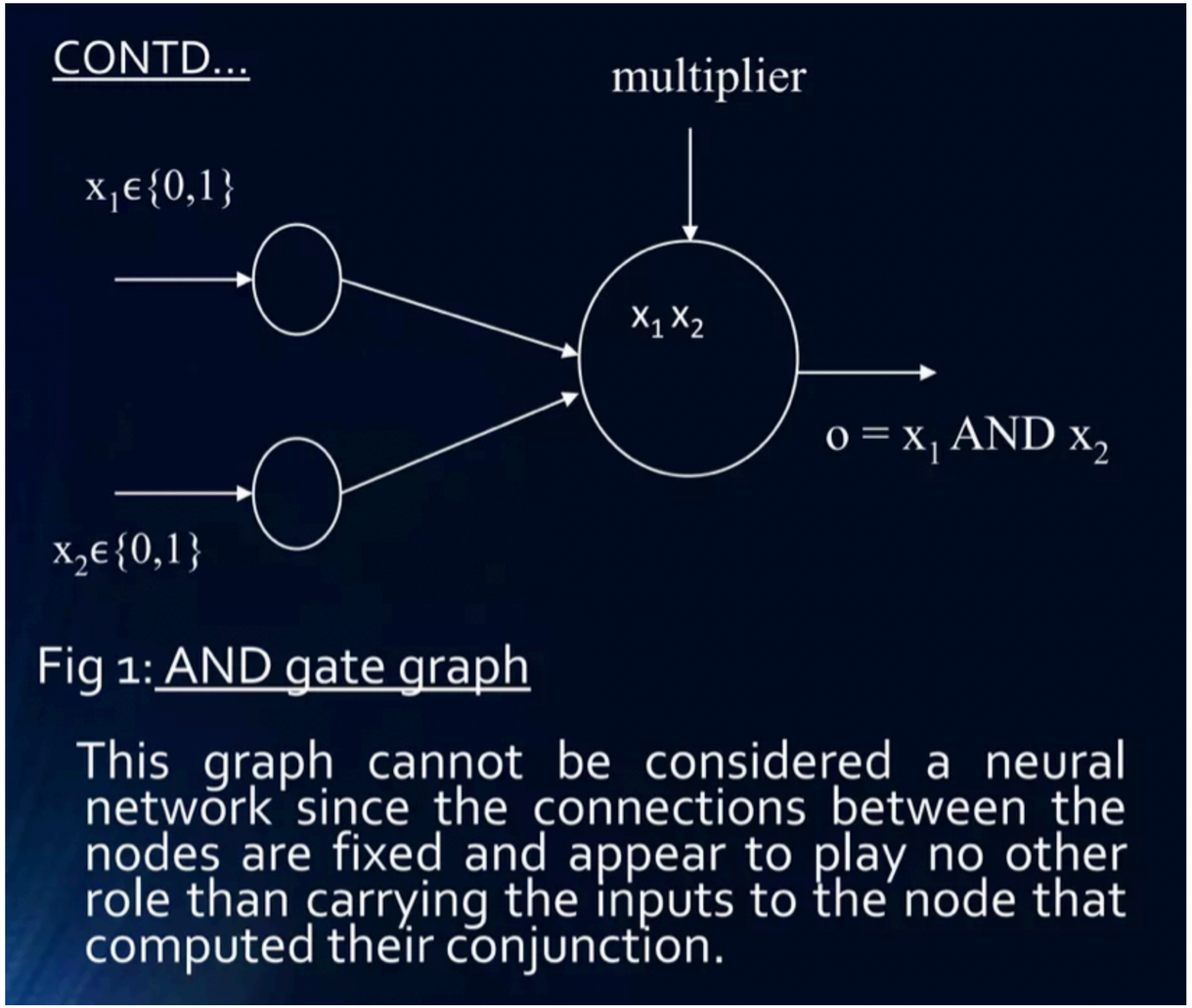


Fig 1: AND gate graph

This graph cannot be considered a neural network since the connections between the nodes are fixed and appear to play no other role than carrying the inputs to the node that computed their conjunction.

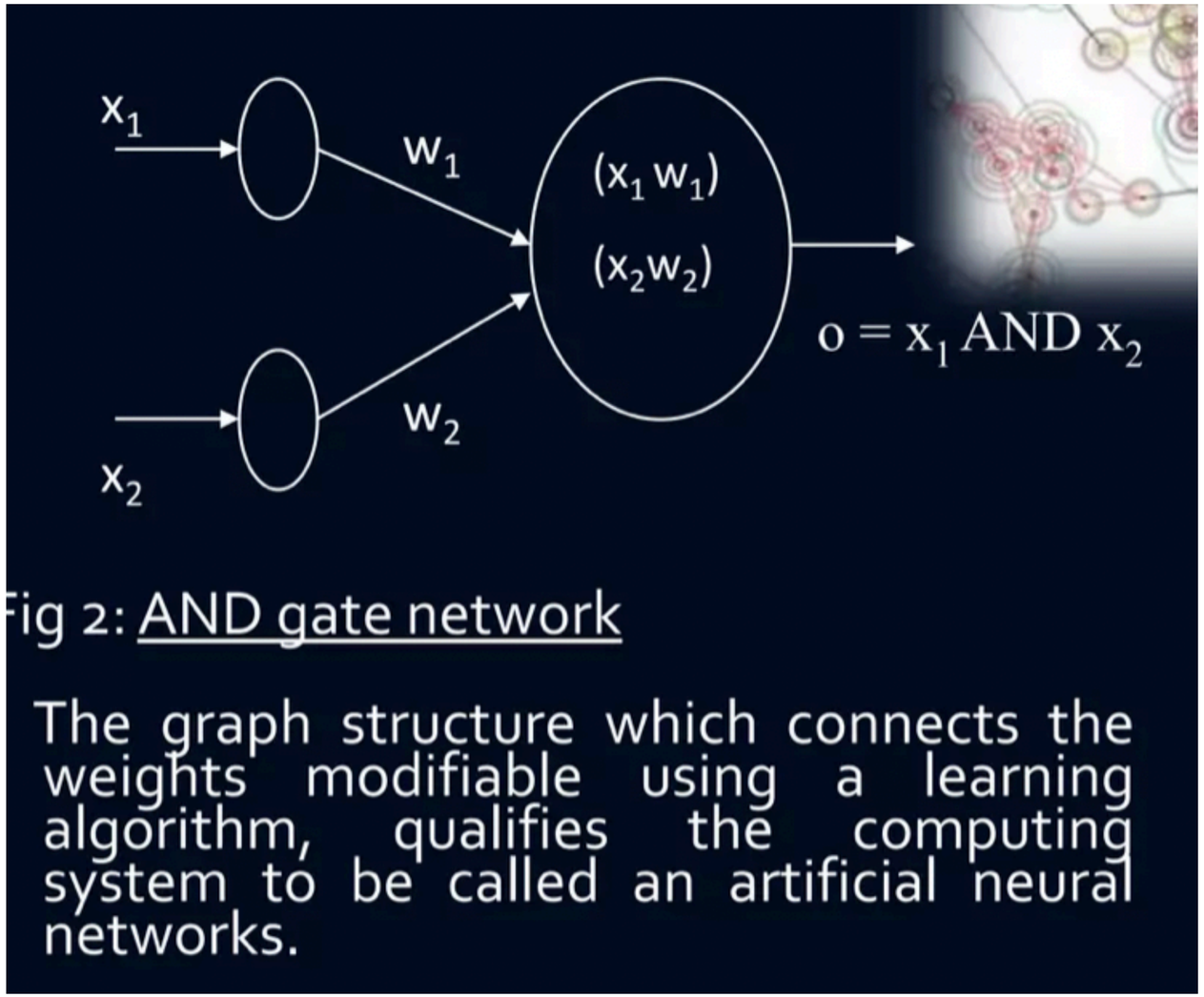


Fig 2: AND gate network

The graph structure which connects the weights modifiable using a learning algorithm, qualifies the computing system to be called an artificial neural networks.

https://www.slideshare.net/DEEPASHRIHK/artificial-neural-network-18449085

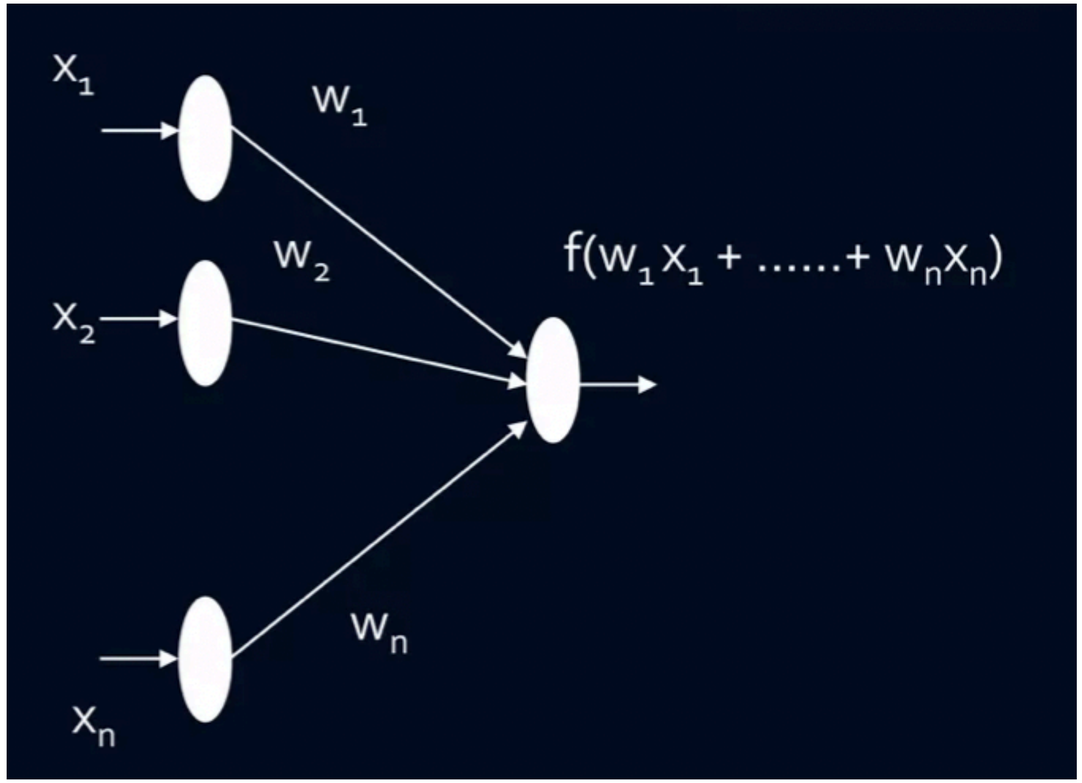
Artificial Neural Networks

ARTIFICIAL NEURON MODEL

- Inputs to the network are represented by the mathematical symbol, x_n
- Each of these inputs are multiplied by a connection weight , w_n

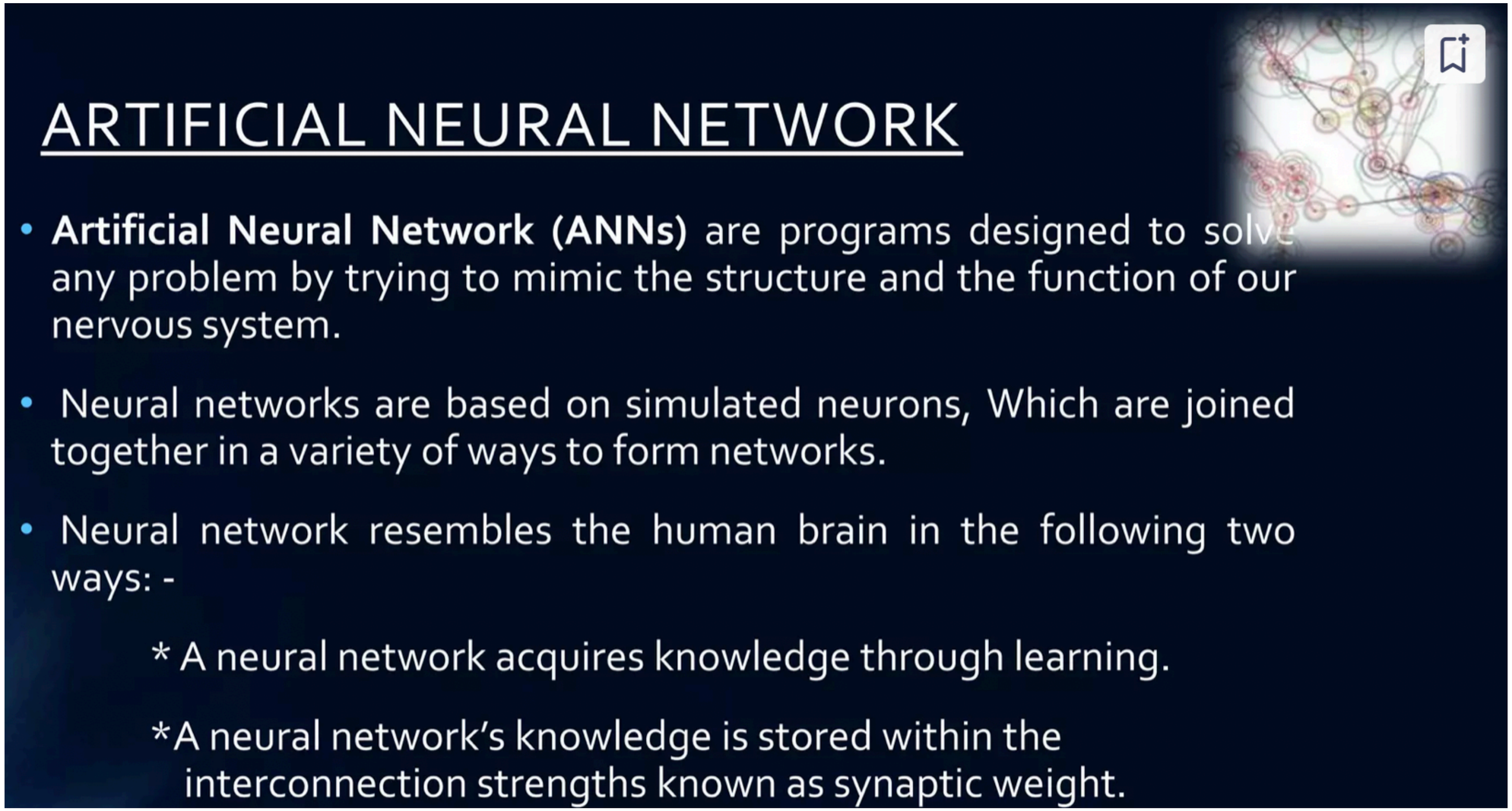
$$sum = w_1 x_1 + \ldots\ldots + w_n x_n$$

- These products are simply summed, fed through the transfer function, f( ) to generate a result and then output.



https://www.slideshare.net/DEEPASHRIHK/artificial-neural-network-18449085

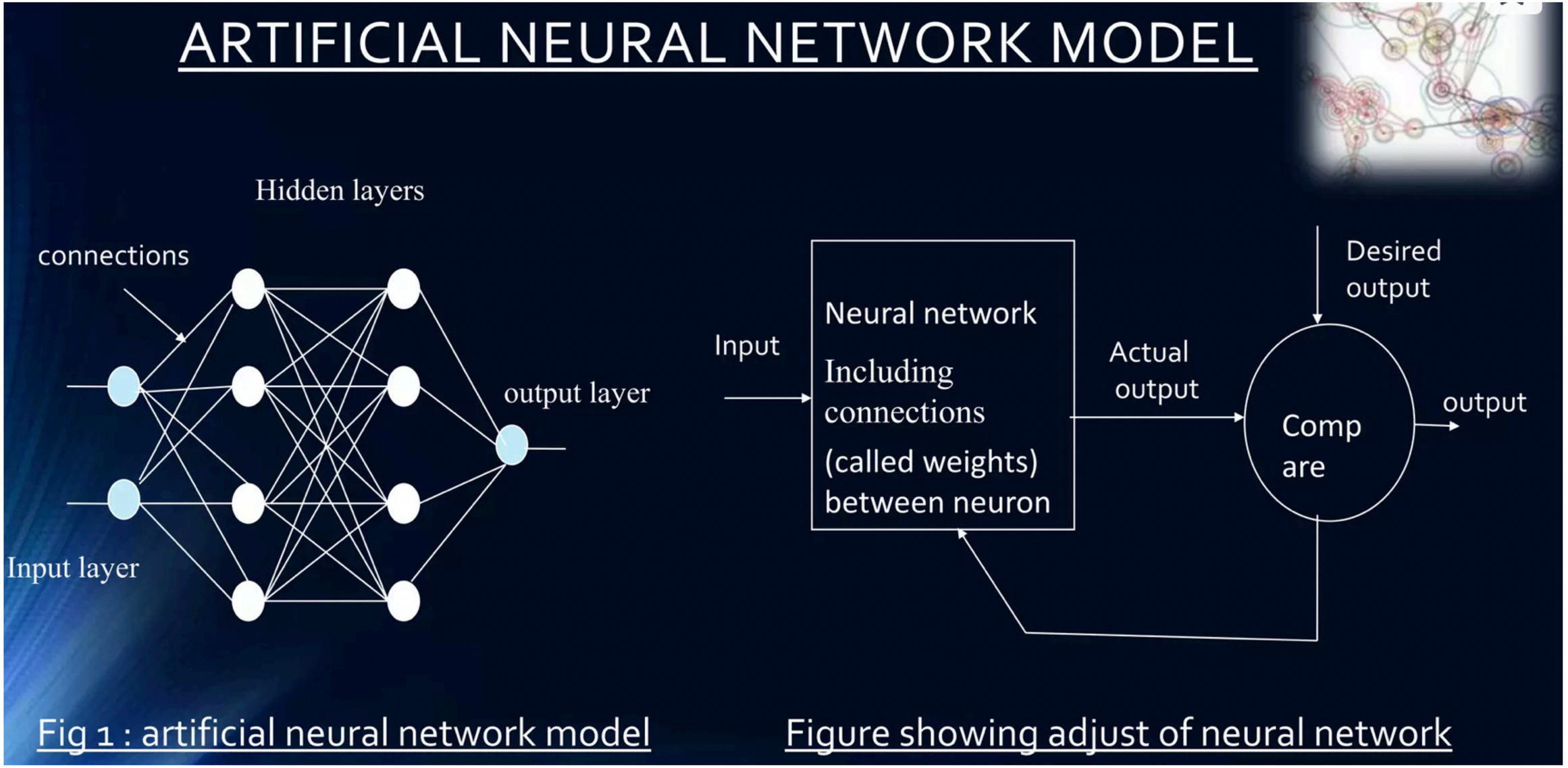
Artificial Neural Networks



https://www.slideshare.net/DEEPASHRIHK/artificial-neural-network-18449085

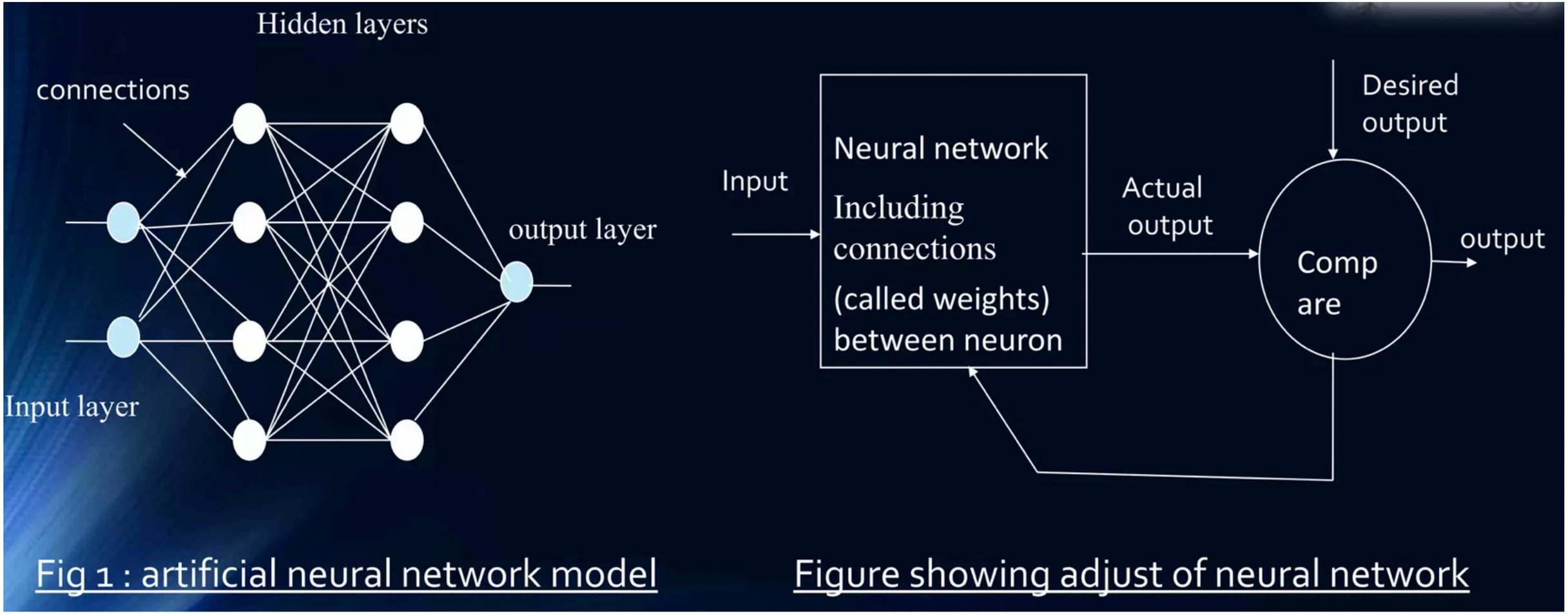
The content is added from different sources

Artificial Neural Networks



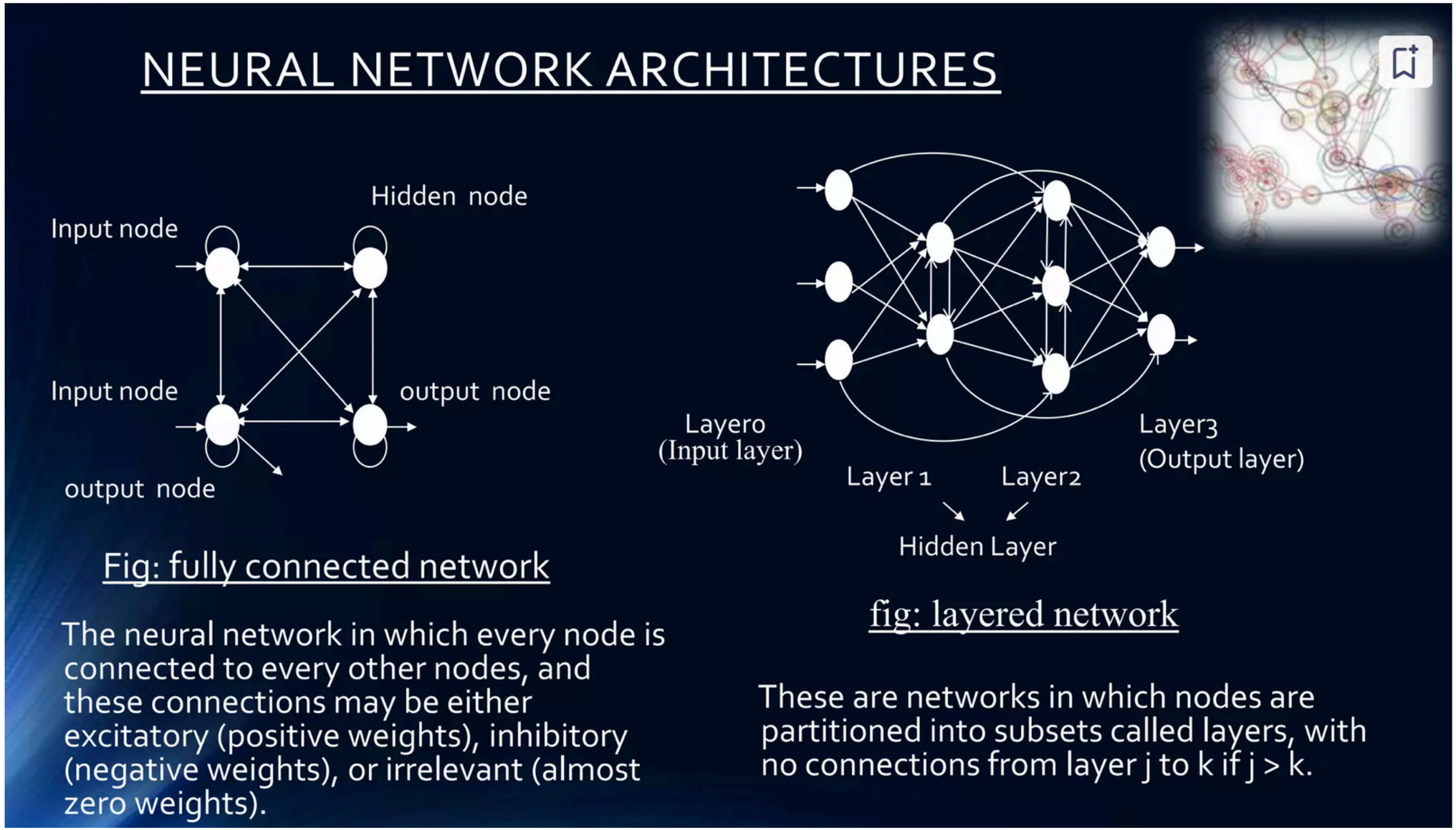
https://www.slideshare.net/DEEPASHRIHK/artificial-neural-network-18449085

Artificial Neural Networks



https://www.slideshare.net/DEEPASHRIHK/artificial-neural-network-18449085

Artificial Neural Networks



https://www.slideshare.net/DEEPASHRIHK/artificial-neural-network-18449085

The content is added from different sources

Artificial Neural Networks

CONTD...

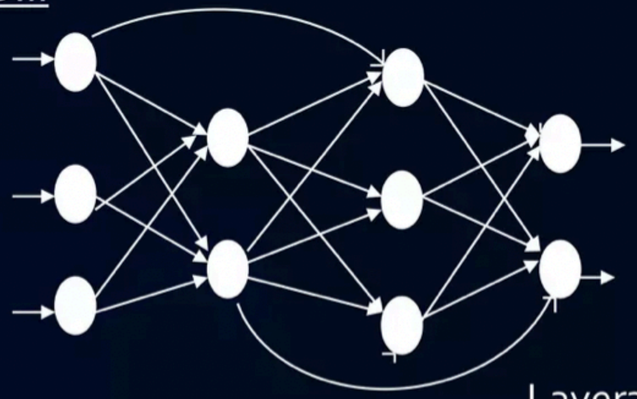


Fig : Acyclic network

This is the subclass of the layered networks in which there is no intra-layer connections. In other words, a connection may exist between any node in layer i and any node in layer j for i < j, but a connection is not allowed for i=j.

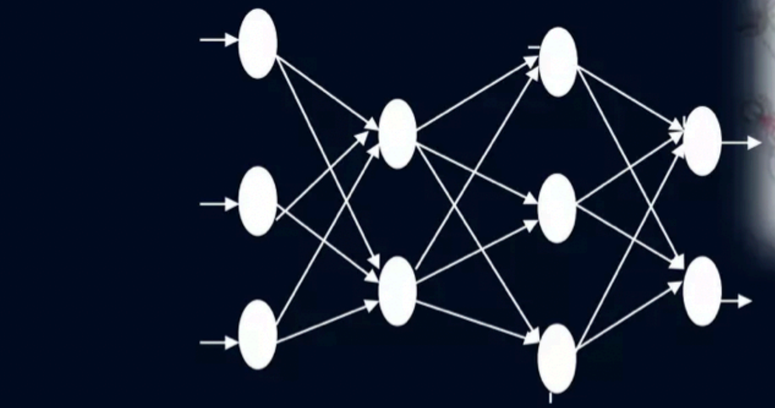


fig : Feedforward network

This is a subclass of acyclic networks in which a connection is allowed from a node in layer i only to nodes in layer i+1

https://www.slideshare.net/DEEPASHRIHK/artificial-neural-network-18449085

Artificial Neural Networks

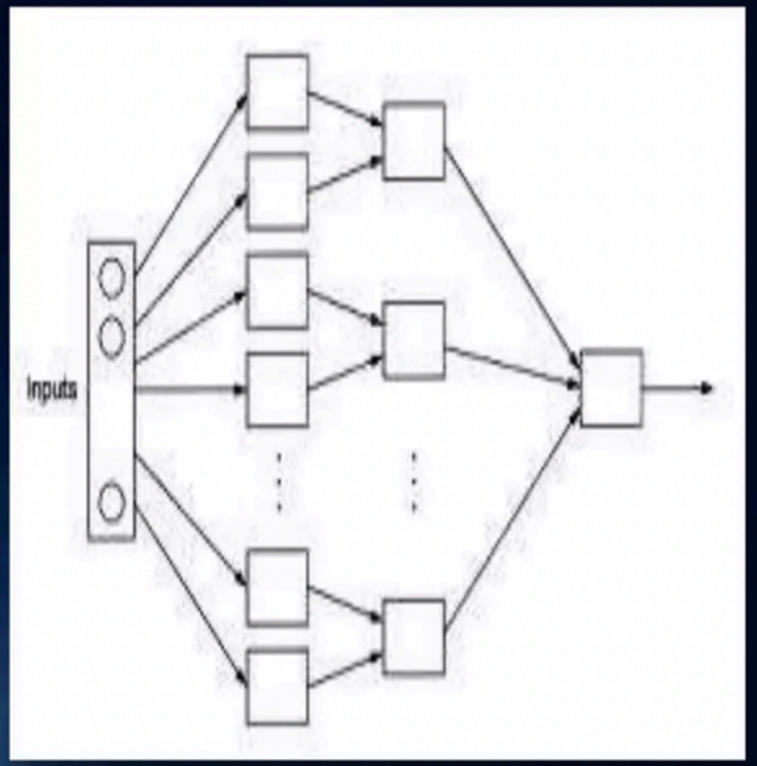


Fig : Modular neural network

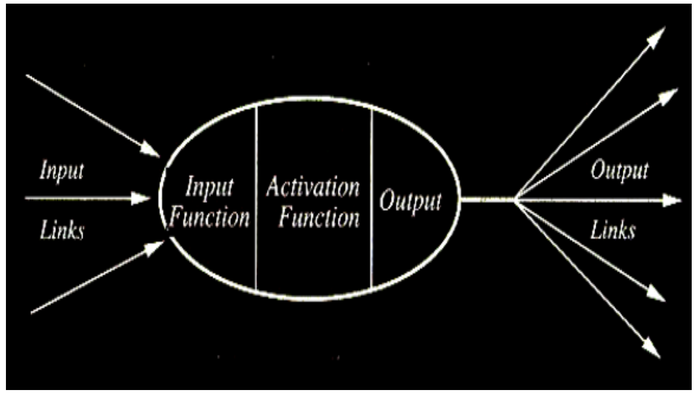
Many problems are best solved using neural networks whose architecture consists of several modules, with sparse interconnections between them. Modules can be organized in several different ways as Hierarchial organization, Successive refinement, Input modularity

https://www.slideshare.net/DEEPASHRIHK/artificial-neural-network-18449085

Key Concepts

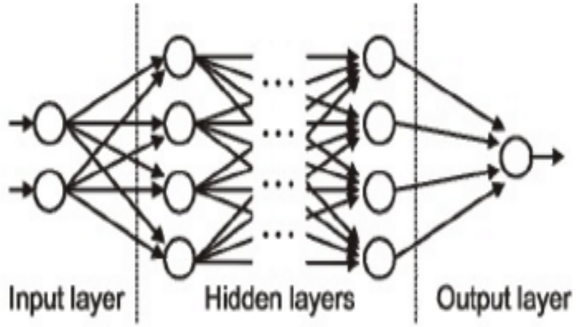
➢ **Neurons:** Neurons are the fundamental building blocks of a neural network.

- ➢Neurons are used for receiving input, processing input and generating output.
- ➢Artificial neural networks are made up of neurons.
- ➢Depending on the layer in which neurons are present they perform different functionalities.



➢ **Layers(Input/hidden/output):** The neurons are organized in the form of layers in an ANN.

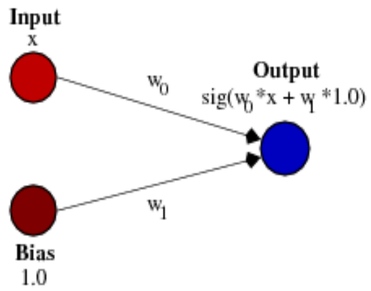
- ➢All layers in a neural network can be segregated as either a input layer, hidden layer or an output layer.
- ➢The input layer receives the input and is the first layer of the neural network.
- ➢The hidden layers are the processing layers that learn the features or patterns for mapping the input to the output layer.
- ➢The output layer is the final layer that generates the output.



Key Concepts

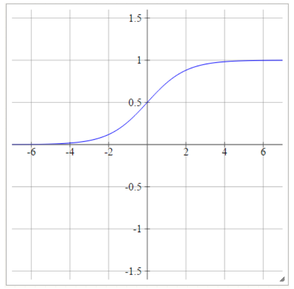
➢ **Weights and biases:** The neurons in a neural network are connected and each connection has an associated weight assigned to it. We initialize the weights randomly and these weights are updated during the model training process.

Biases are analogous to the intercept in a regression model.

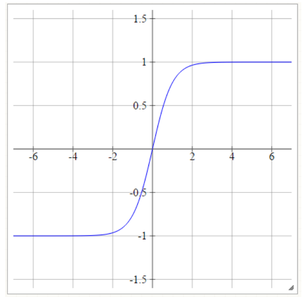


➢**Activation functions:** Activation functions introduce non-linearity into neural networks. Without these activation functions, neural networks will be very similar to that of a linear model. Commonly used activation functions include sigmoid, tanh, ReLu and softmax.

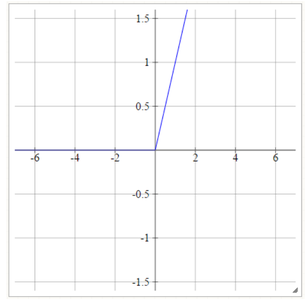
Sigmoid = 1/(1+e^(-x))



Tanh= (e^(x) – e^(-x))/((e^(x) + e^(-x)))

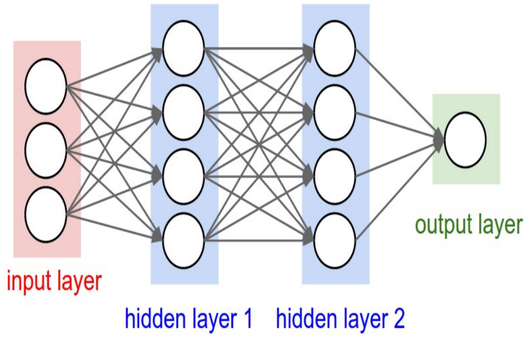


ReLu=max(0,x)



Key Concepts

➢ **Multi-Layer Perceptron's (MLP):** A single neuron would not be able to perform highly complex tasks. Therefore, we use stacks of neurons to generate the desired outputs. In the simplest network we would have an input layer, a hidden layer and an output layer. Each layer has multiple neurons and all the neurons in each layer are connected to all the neurons in the next layer.



➢ Training the neural network(Forward Pass/ Backward Pass): After configuring the neural network, the training samples are passed to the neural network with the input and target(i.e Forward pass) and the optimal weights and biases are learnt by the network(i.e. Backward pass).

Key Concepts

➢ **Cost function:** Cost function measures how close the model's prediction is to the actual value. A higher value indicates that the model's predictions are not accurate. The objective of the neural network is to minimize this cost function, thereby reducing the error.

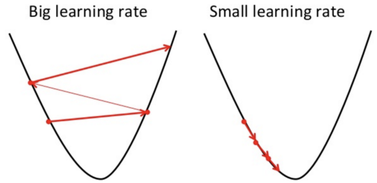
➢ **Backprorogation:** Backprorogation is a technique that is used for training a neural network. Initially, random weights and biases are assigned to the neural network. Once the training process starts, the weights and biases are updated based in the direction of the gradient.

➢ **Learning rate:** This parameter determines how fast or slow we will move towards the optimal weights. If the learning rate is very large we will skip the optimal solution. If it is too small we will need too many iterations to converge to the best values.

➢ **Optmizers:** Optmizers are the algorithms that are used for calculating the gradients and the parameter updates with the aim of reducing the cost function. Commonly used optimizers are stochastic gradient descent, adam, adagrad and rmsprop.

➢ **Regularizers:** Sometimes the neurons in the neural network learn the exact representation of the training data. When this happens the network will not perform well on unseen data. To avoid such an overfitting, regularizers are used. Popular regularizers include dropouts and batch normalization.

➢**Epochs:** One forward and backward pass of all training examples.



The content is added from different sources

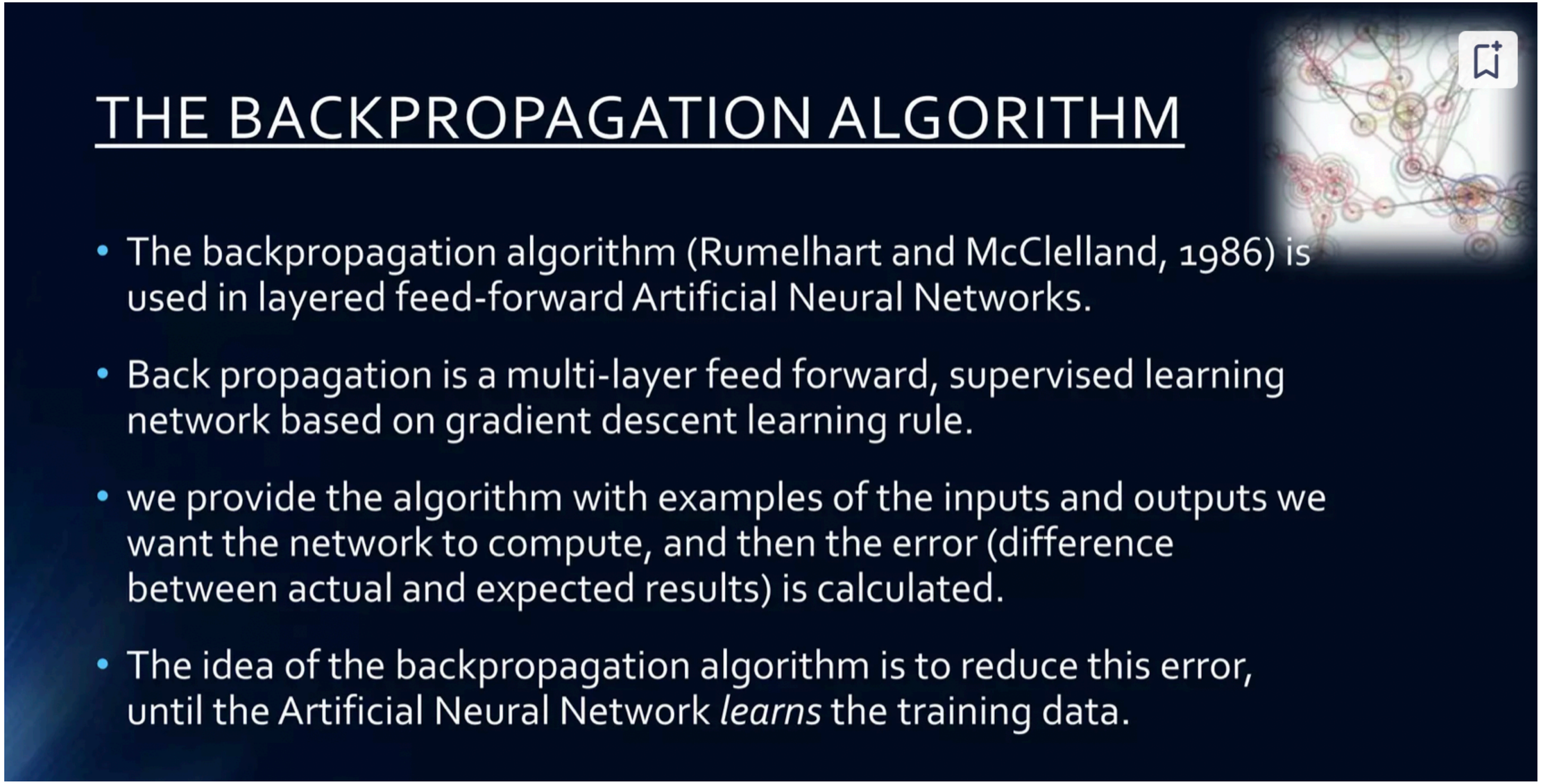
Artificial Neural Networks

LEARNING

- Neurons in an animal's brain are "hard wired". It is equally obvious tha animals, especially higher order animals, learn as they grow.
- How does this learning occur?
- What are possible mathematical models of learning?
- In artificial neural networks, learning refers to the method of modifying the weights of connections between the nodes of a specified network.
- The learning ability of a neural network is determined by its architecture and by the algorithmic method chosen for training.

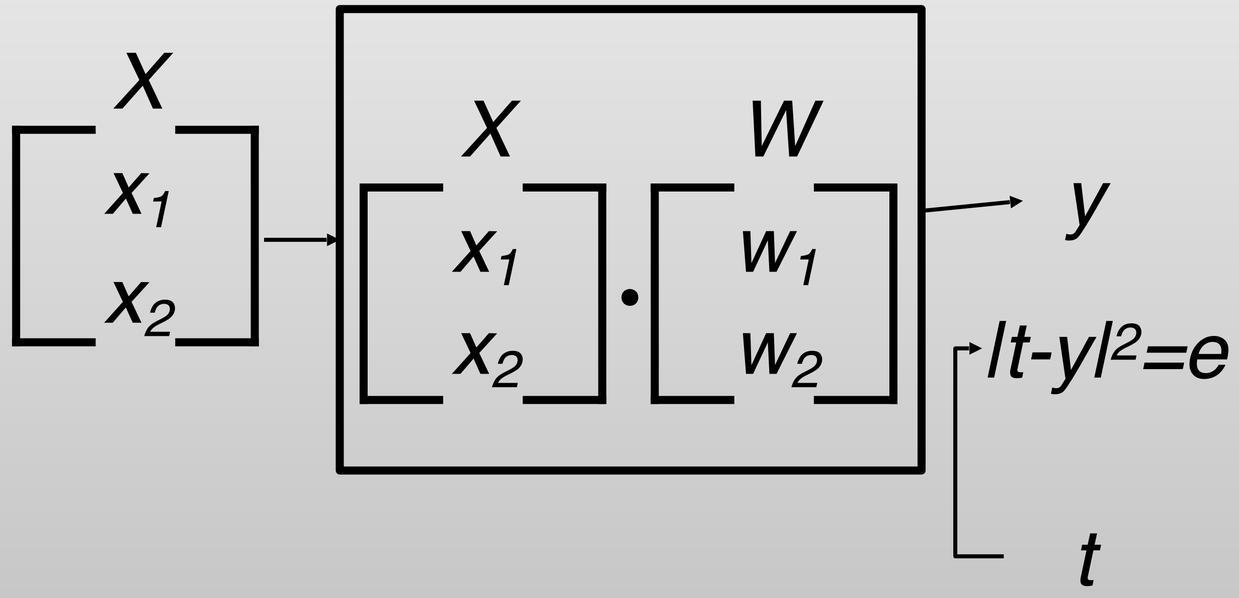
https://www.slideshare.net/DEEPASHRIHK/artificial-neural-network-18449085

Artificial Neural Networks

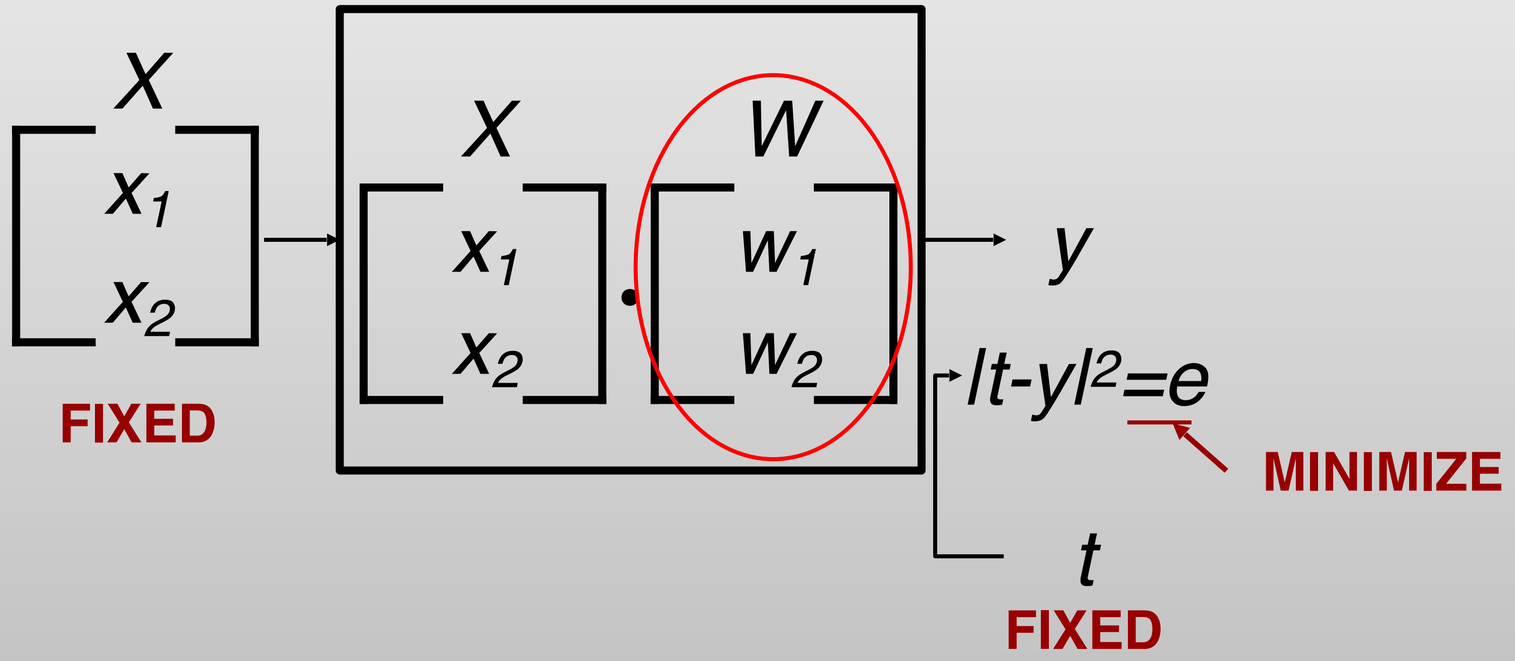


https://www.slideshare.net/DEEPASHRIHK/artificial-neural-network-18449085

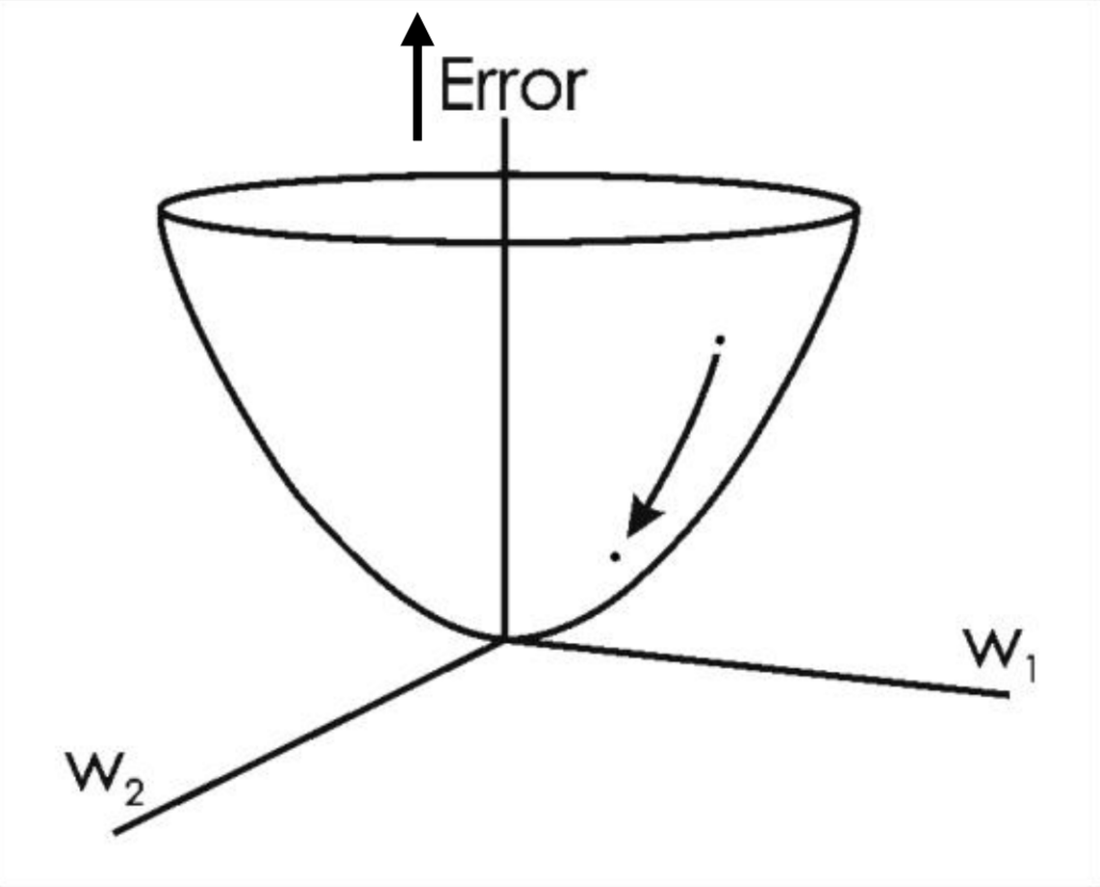
Learning Weights



Learning Weights



Learning Weights

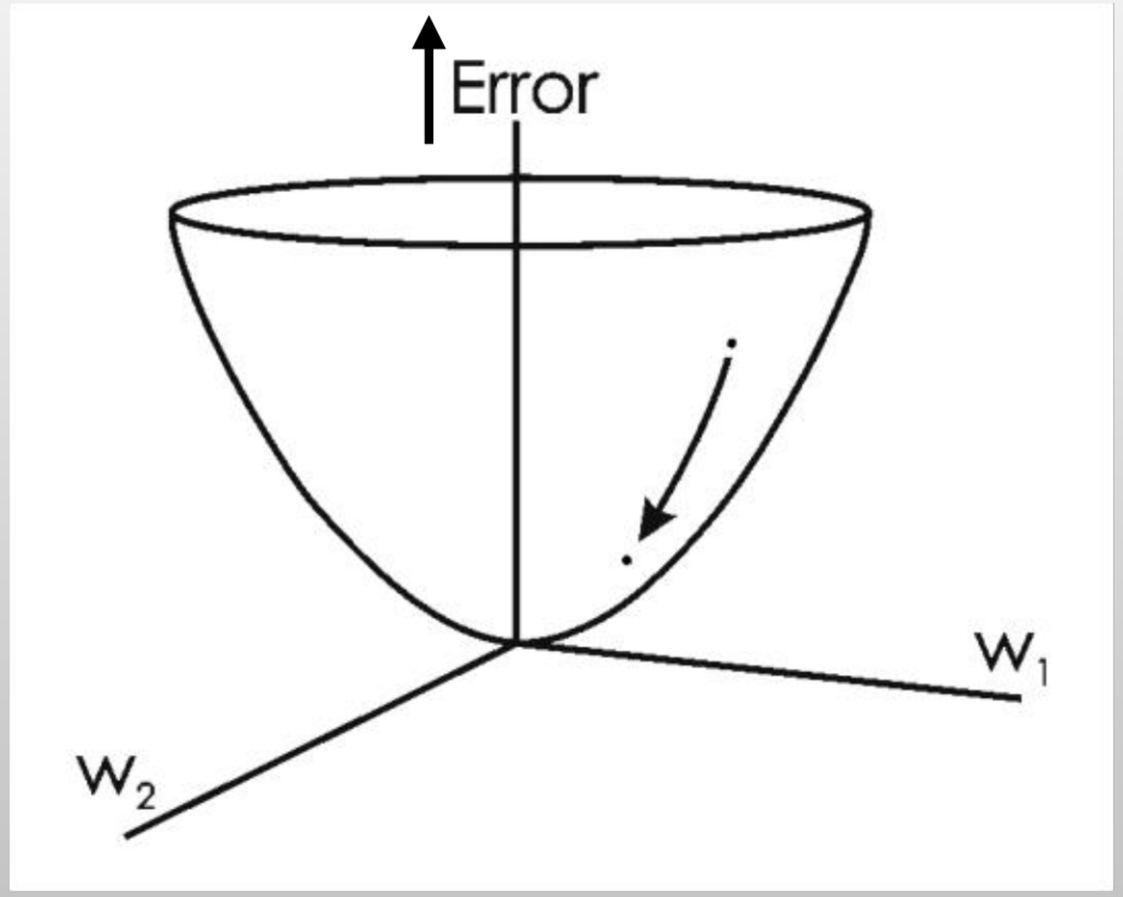


- Find minima of Error
  - optimal (w_1, w_2)
- Assign credit/blame

  $\frac{\partial e}{\partial w_1}$ $\frac{\partial e}{\partial w_2}$
- Change values

  w_1 *and* w_2

Learning Weights - Loss function



- w_1 *and* w_2 depend on error
- Estimate error
- *f(t,o)* function of output & target
- Loss Function

Learning Weights - Gradient descent

$$e=\tfrac{1}{2}\cdot\sum |t_i-y_i|^2$$

$$\frac{\partial e}{\partial w_1}=\sum \frac{\partial e}{\partial y_i}\cdot\left(\frac{\partial y_i}{\partial w_1}\right);\frac{\partial e}{\partial w_2}=\sum \frac{\partial e}{\partial y_i}\cdot\left(\frac{\partial y_i}{\partial w_2}\right)$$

CHAIN RULE

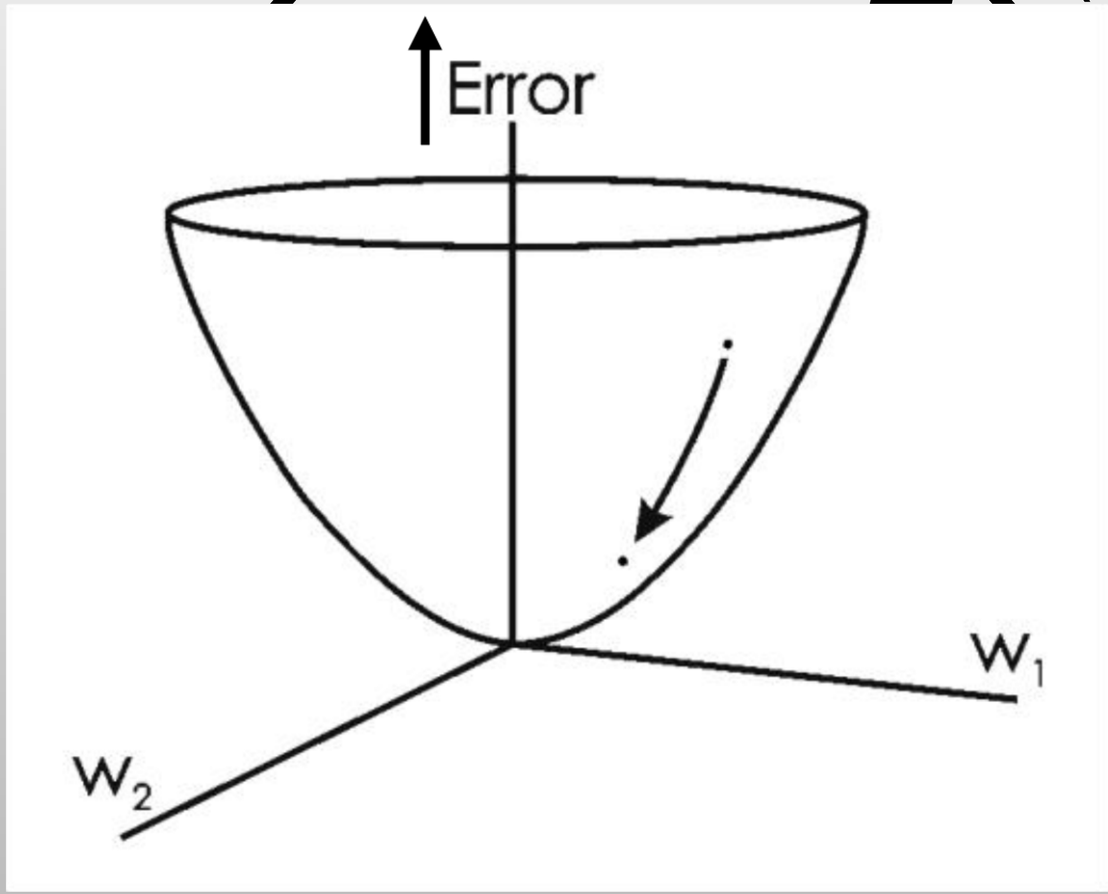
$$\frac{\partial e}{\partial y_i}=-(t_i-y_i);\frac{\partial y_i}{\partial w_1}=x_{1i};\frac{\partial y_i}{\partial w_2}=x_{2i}$$

$$\frac{\partial e}{\partial w_1}=\sum -(t_i-y_i)\cdot x_{1i};\frac{\partial e}{\partial w_2}=\sum -(t_i-y_i)\cdot x_{2i}$$

Gradient descent

$$\frac{\partial e}{\partial w_1} = \sum -(t_i - y_i) \cdot x_{1i} \; ; \; \frac{\partial e}{\partial w_2} = \sum -(t_i - y_i) \cdot x_{2i}$$

← GRADIENT

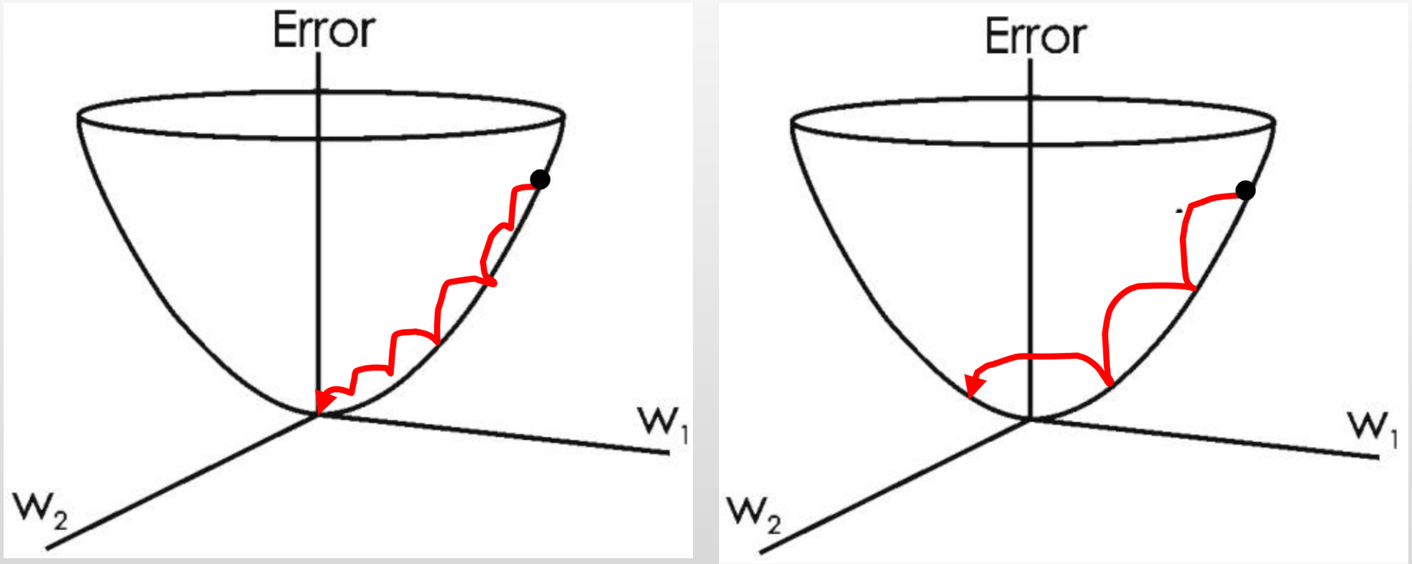


GRADIENT DESCENT

$$w_1' = w_1 \frac{\partial e}{\partial w_1}$$

$$w_2' = w_2 \frac{\partial e}{\partial w_2}$$

Gradient descent

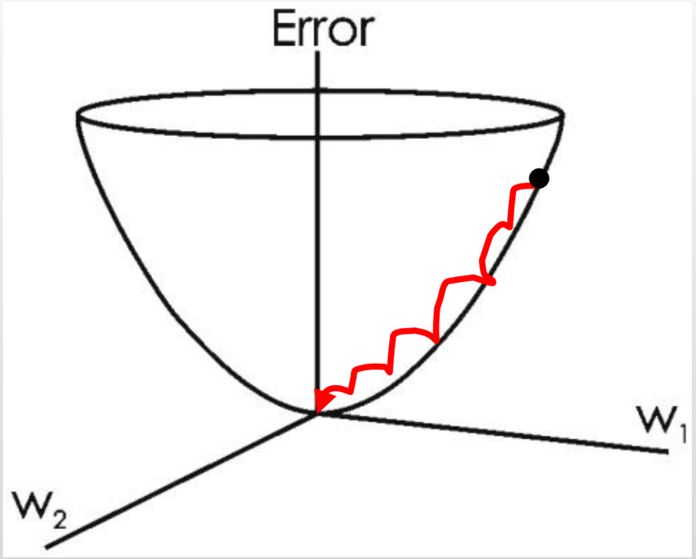


$$w_1' = w_1 \frac{\partial e}{\partial w_1} \cdot \eta$$

Learning rate

Step size

Gradient descent

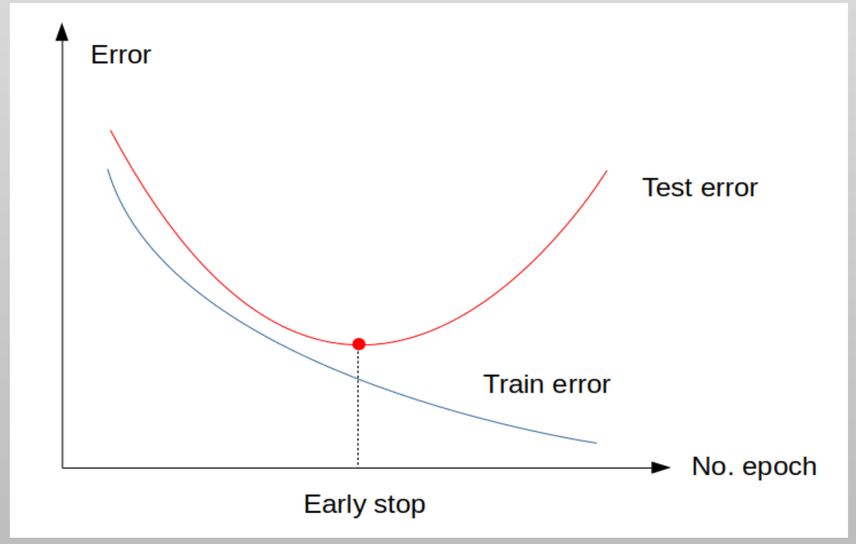


Repeat for n steps:

$$w_1' = w_1 \frac{\partial e}{\partial w_1} \cdot \eta$$

Get optimal weights

When to stop?



Limitations of Gradient descent

$$\frac{\partial e}{\partial w_1} = \sum -(t_i - y_i) \cdot x_{1i}$$

- Large number of samples
  - Computationally expensive
  - Redundant
- A uniform learning rate
- If new training samples are acquired start from scratch

Stochastic Gradient descent

$$\frac{\partial e}{\partial w_1} = -(t_i - y_i) \cdot x_{1i}$$

- Randomly select a small subset of samples in each iteration
- Start with a large learning rate keep reducing it after fixed number of steps
- As new samples come it, update weights using only the new samples
- Works really well for
  - large datasets
  - large networks

Multilayer perceptron

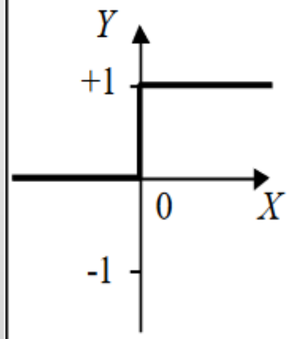
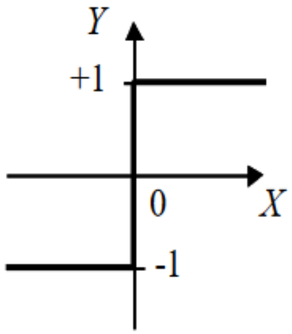
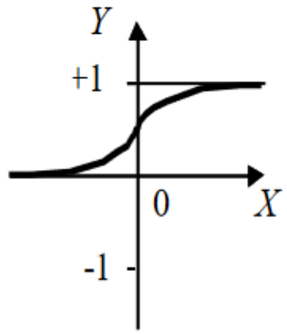
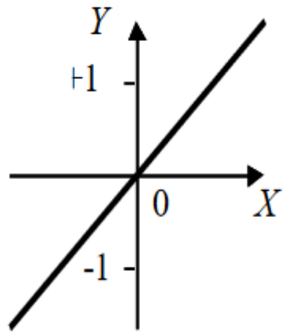
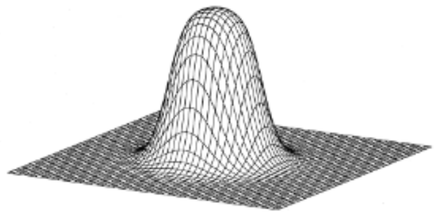
$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \cdot \begin{bmatrix} w_4 \\ w_5 \end{bmatrix} \cdot W_6 = y$$

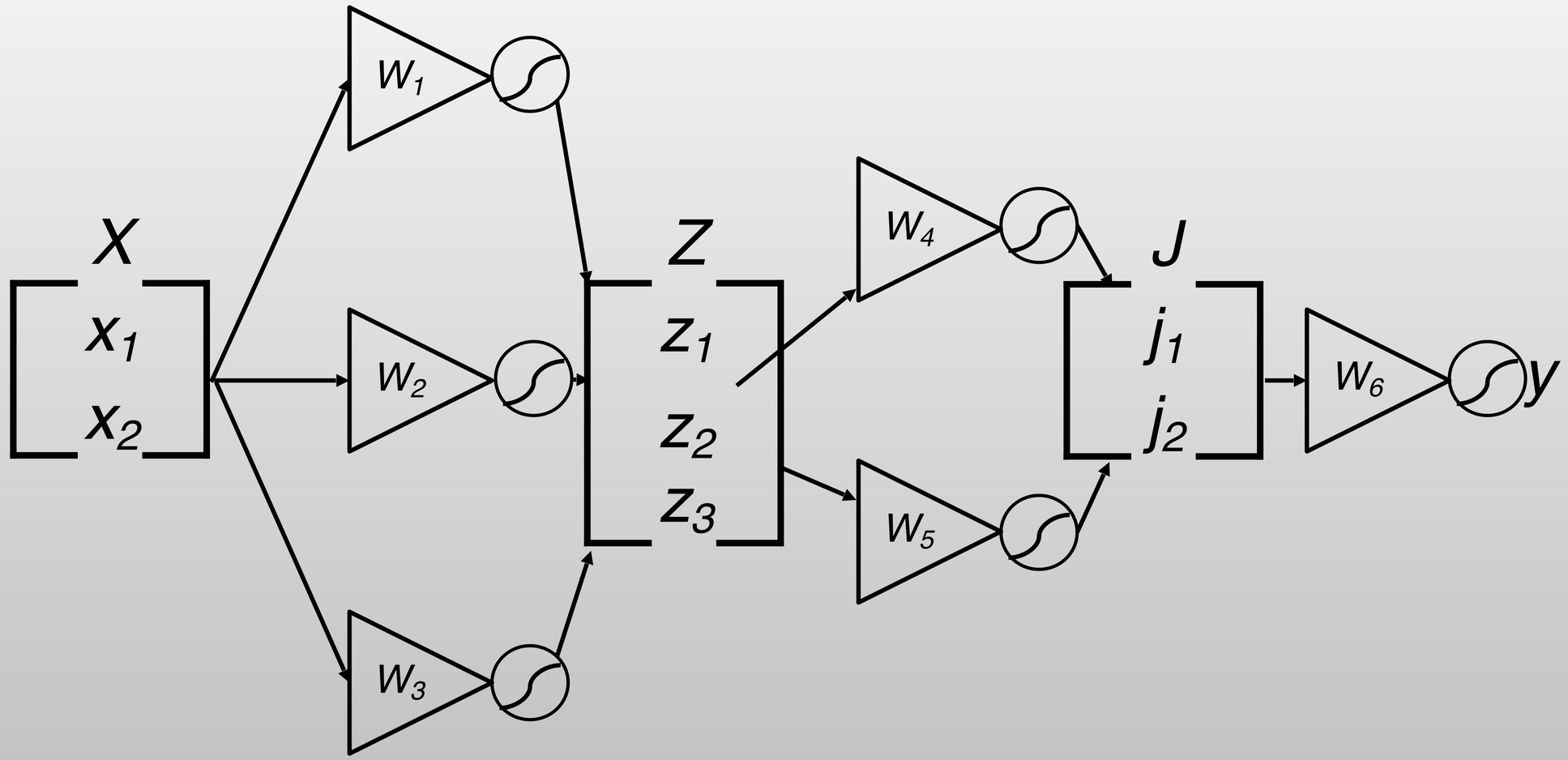
SERIES OF LINEAR OPERATIONS

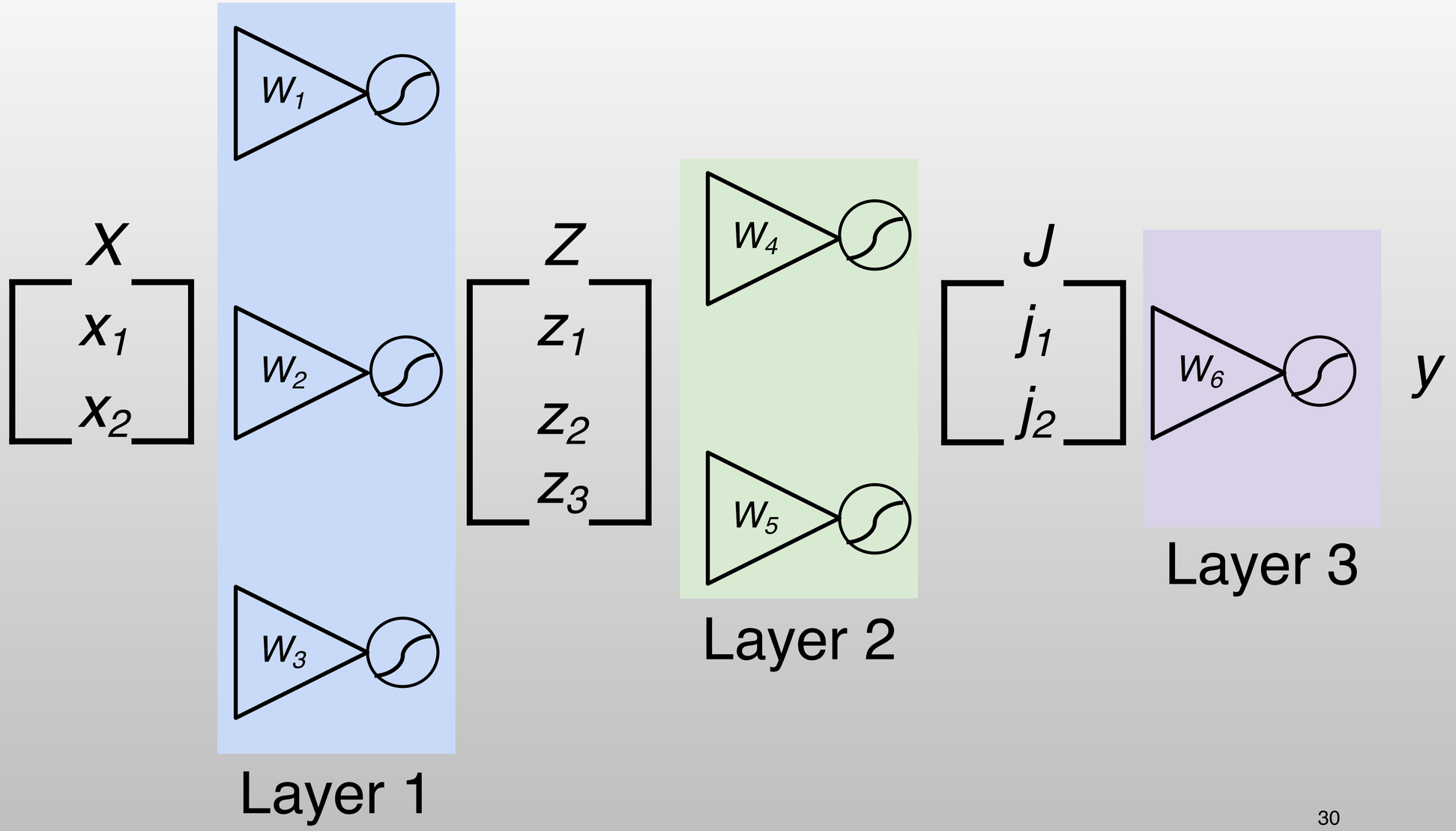
STILL A LINEAR OPERATION

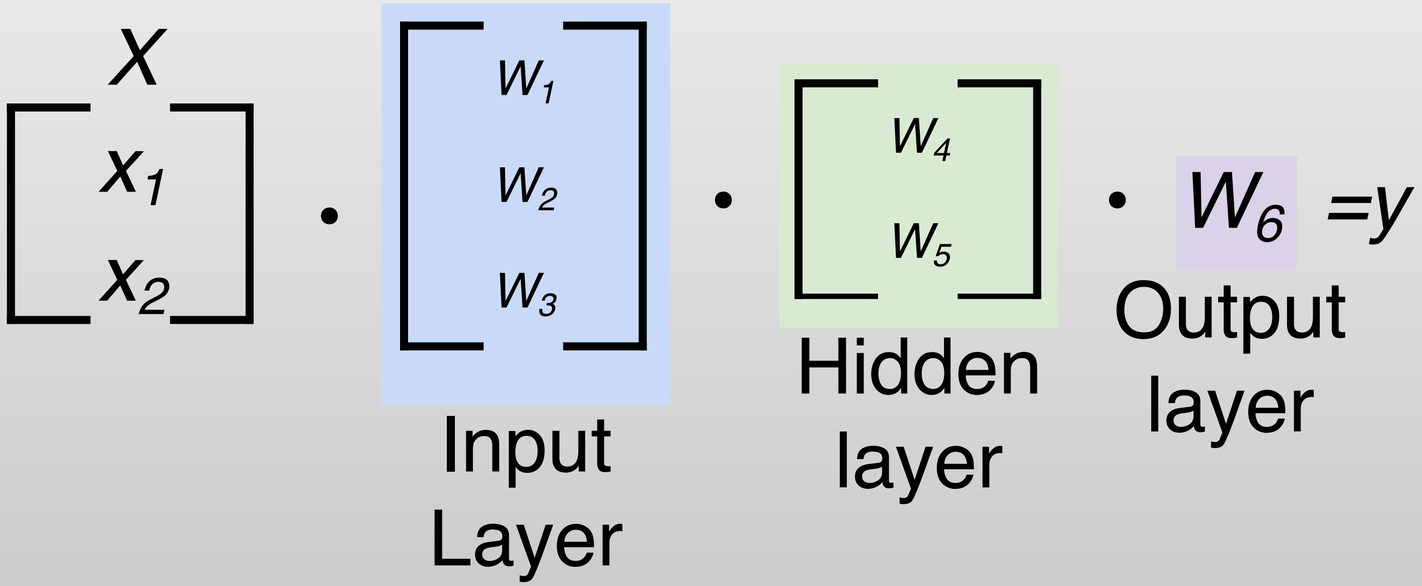
INTRODUCE NON-LINEAR OPERATIONS

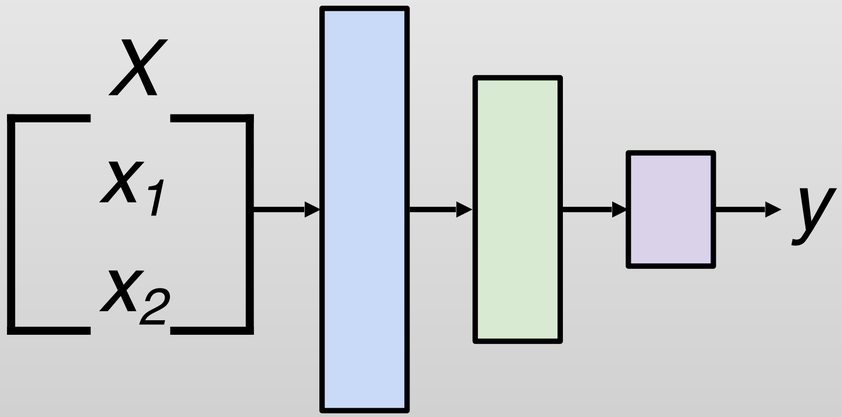
Activation functions

Step function	*Sign function*	*Sigmoid function*	*Linear function*	Radial basis function
				
$Y^{step} = \begin{cases} 1, \text{ if } X \geq 0 \\ 0, \text{if } X < 0 \end{cases}$	$Y^{sign} = \begin{cases} +1, \text{if } X \geq 0 \\ -1, \text{if } X < 0 \end{cases}$	$Y^{sigmoid} = \frac{1}{1+e^{-X}}$	$Y^{linear} = X$	$Y^{RBF} = e^{-\frac{(X-c)^2}{2\sigma^2}}$


X
x1
x2
w1
w2
w3
Z
z1
z2
z3
w4
w5
J
j1
j2
w6
y


X
x1
x2
w1
w2
w3
Layer 1
Z
z1
z2
z3
w4
w5
Layer 2
J
j1
j2
w6
y
Layer 3

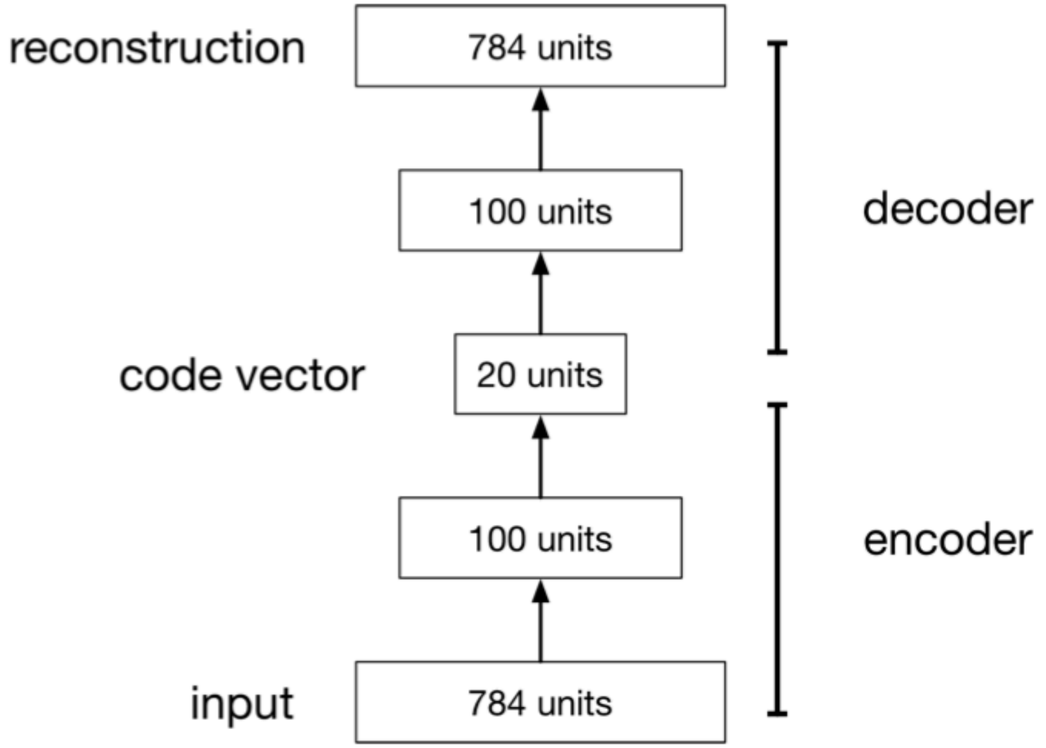

X
x1
x2
w1
w2
w3
w4
w5
W6 =y
Input Layer
Hidden layer
Output layer



- Chain rule can be extended across layers
- Weights learnt from data
- Hyperparameters
  - no. of layers
  - no. of neurons
  - activations
  - loss function

Autoencoder

- An autoencoder is a feed-forward neural net whose job it is to take an input **x** and predict **x**.
- To make this non-trivial, we need to add a bottleneck layer whose dimension is much smaller than the input.



https://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec20.pdf

Autoencoder

Why autoencoders?

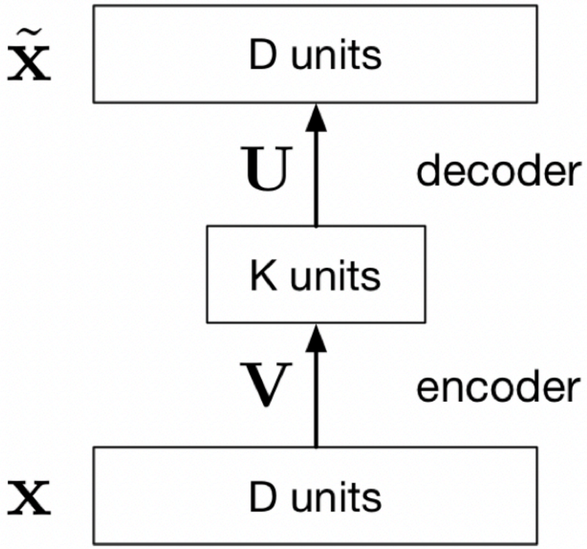
- Map high-dimensional data to two dimensions for visualization
- Compression (i.e. reducing the file size)
  - Note: autoencoders don't do this for free — it requires other ideas as well.
- Learn abstract features in an unsupervised way so you can apply them to a supervised task
  - Unlabled data can be much more plentiful than labeled data

https://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec20.pdf

Principal Component Analysis

- The simplest kind of autoencoder has one hidden layer, linear activations, and squared error loss.

$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$$

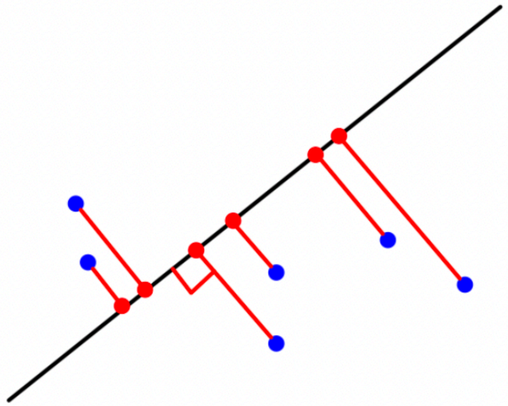


- This network computes $\tilde{\mathbf{x}} = \mathbf{UVx}$, which is a linear function.
- If $K \geq D$, we can choose $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{UV}$ is the identity. This isn't very interesting.
- But suppose $K < D$:
  - $\mathbf{V}$ maps $\mathbf{x}$ to a K-dimensional space, so it's doing dimensionality reduction.
  - The output must lie in a K-dimensional subspace, namely the column space of $\mathbf{U}$.

https://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec20.pdf

Principal Component Analysis

- We just saw that a linear autoencoder has to map D-dimensional inputs to a K-dimensional subspace $\mathcal{S}$.
- Knowing this, what is the best possible mapping it can choose?
  - By definition, the projection of $\mathbf{x}$ onto $\mathcal{S}$ is the point in $\mathcal{S}$ which minimizes the distance to $\mathbf{x}$.

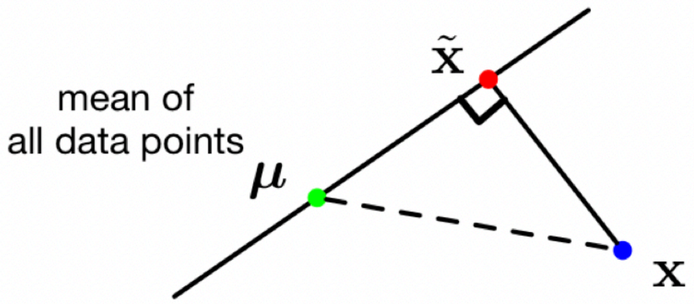


- Fortunately, the linear autoencoder can represent projection onto $\mathcal{S}$: pick $\mathbf{U} = \mathbf{Q}$ and $\mathbf{V} = \mathbf{Q}^\top$, where $\mathbf{Q}$ is an orthonormal basis for $\mathcal{S}$.

https://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec20.pdf

Principal Component Analysis

- The autoencoder should learn to choose the subspace which minimizes the squared distance from the data to the projections.
- This is equivalent to the subspace which maximizes the variance of the projections.



By the Pythagorean Theorem,

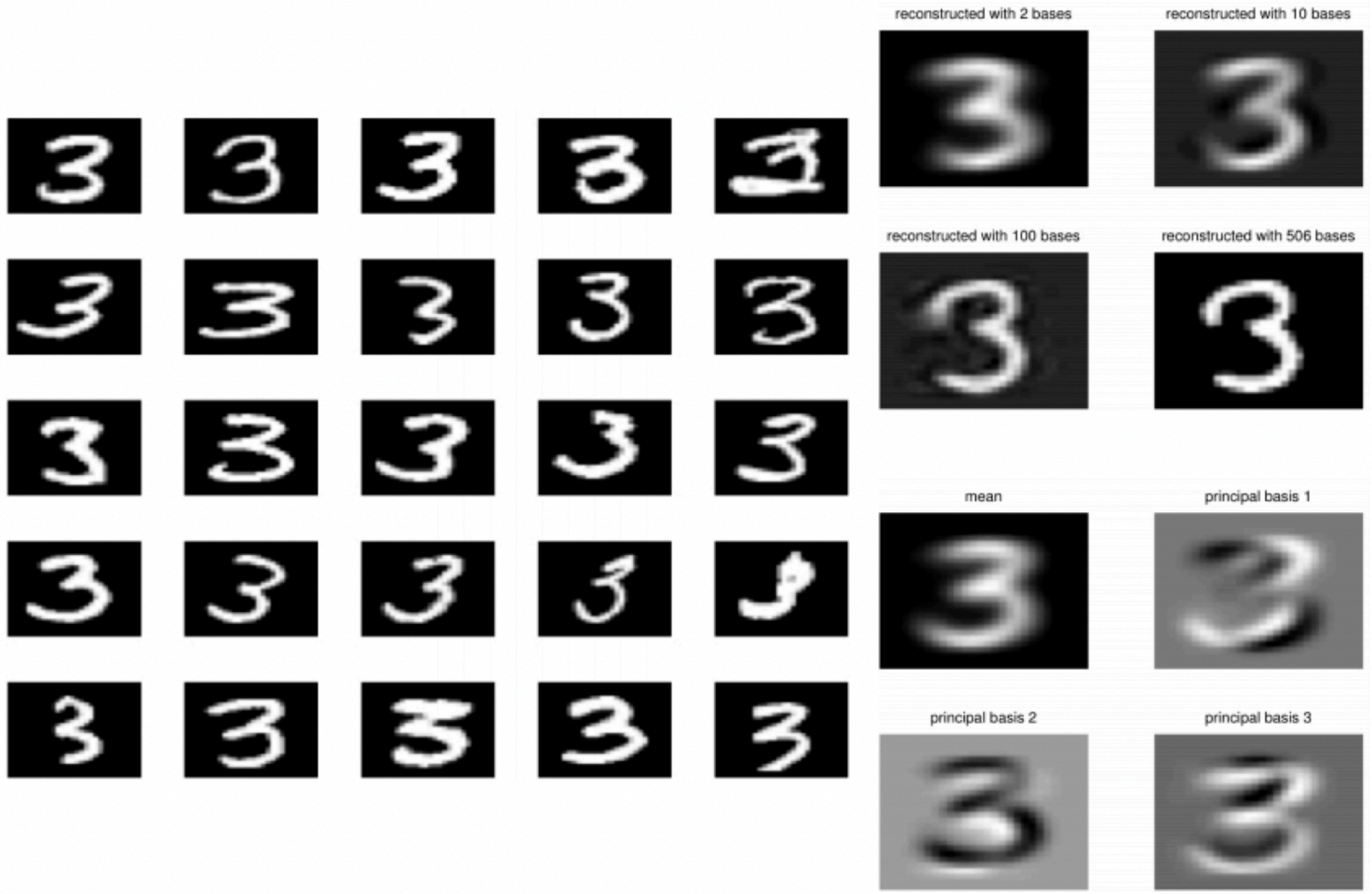
$$\underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\tilde{\mathbf{x}}^{(i)}-\boldsymbol{\mu}\|^2}_{\text{projected variance}}+\underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}^{(i)}-\tilde{\mathbf{x}}^{(i)}\|^2}_{\text{reconstruction error}}$$

$$=\underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}^{(i)}-\boldsymbol{\mu}\|^2}_{\text{constant}}$$

- You wouldn't actually sove this problem by training a neural net. There's a closed-form solution, which you learn about in CSC 411.
- The algorithm is called principal component analysis (PCA).

Principal Component Analysis

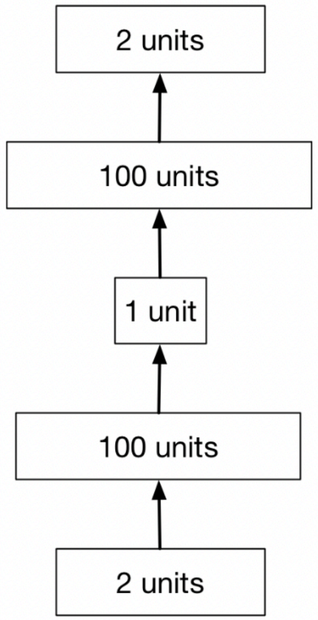
PCA for digits

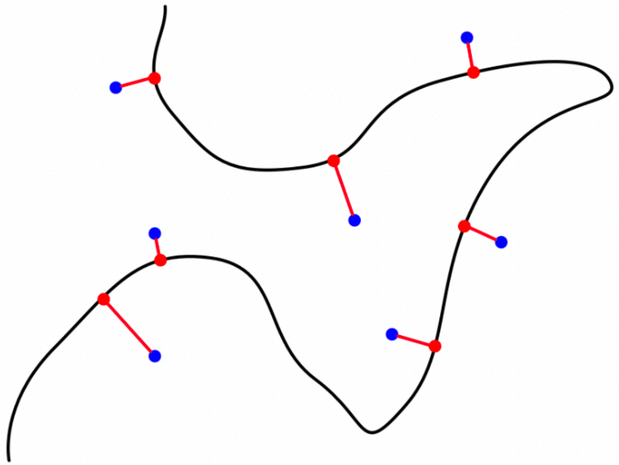


https://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec20.pdf

Deep Autoencoder

- Deep nonlinear autoencoders learn to project the data, not onto a subspace, but onto a nonlinear manifold
- This manifold is the image of the decoder.
- This is a kind of nonlinear dimensionality reduction.

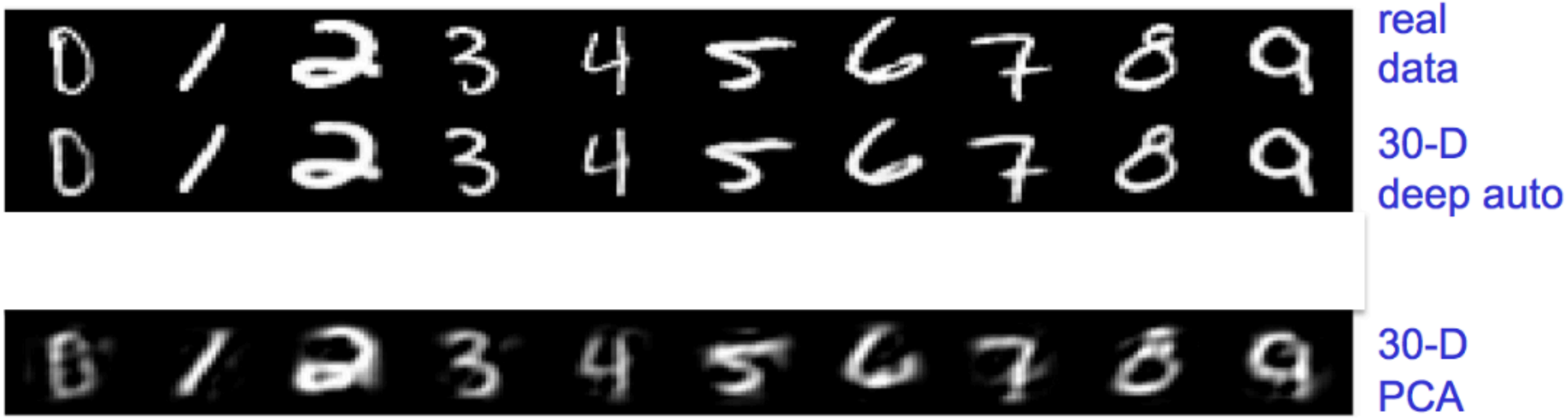




https://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec20.pdf

Deep Autoencoder

- Nonlinear autoencoders can learn more powerful codes for a given dimensionality, compared with linear autoencoders (PCA)



https://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec20.pdf