Neural Networks - 2

Machine Learning and Big Data - DATA622

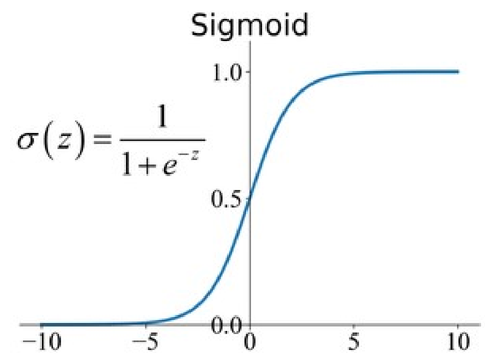
CUNY School of Professional Studies

Activation Functions

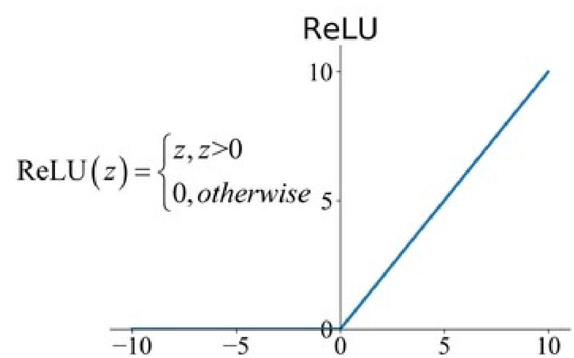
What do we do about non-linear decision boundaries?

Activation Functions

- Activation functions provide non-linear transformation (“reshape” data)
- Sigmoid is typically used for classification e.g. to predict the probability, as an output.
- Rectified Linear Unit (ReLU) is the ‘go-to’ choice for most purposes.
- There are a lot of choices of activation functions – these are only two of many – but they must always be non-linear.



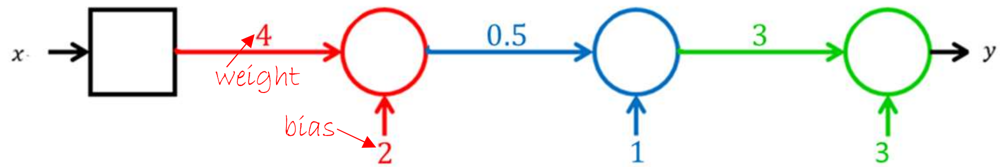
(a)



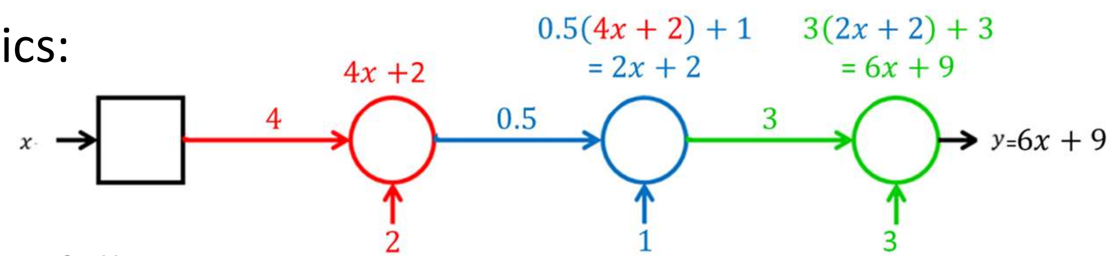
(c)

Do I really need activation functions?

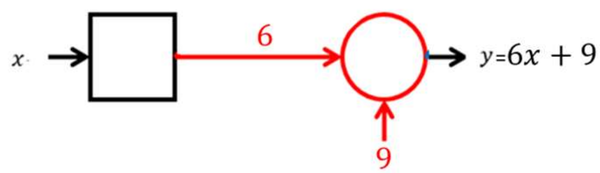
Let’s consider a simple neural network:



Let’s expand the mathematics:



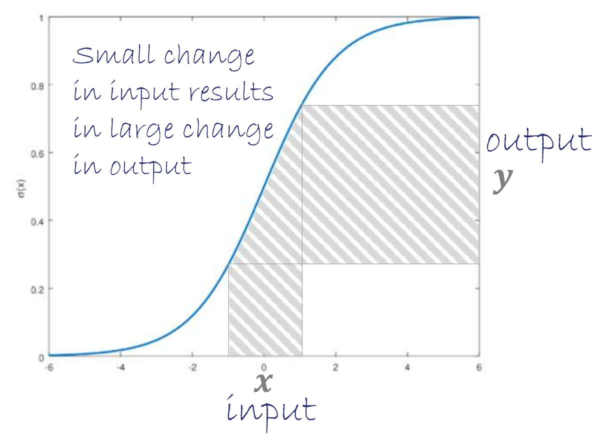
But isn’t that the same as the following?

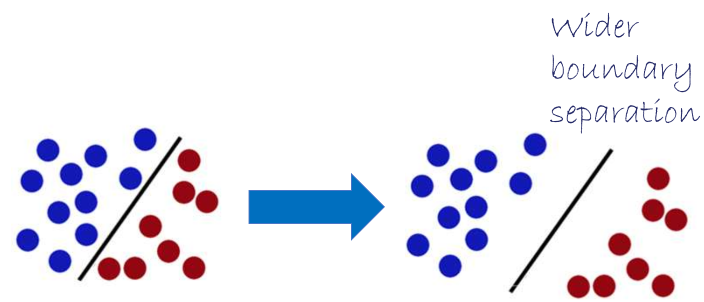


Takeaway: Without a <u>non-linear</u> activation the neural network will collapse

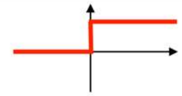
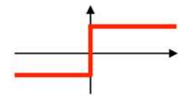
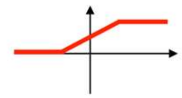
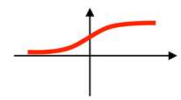
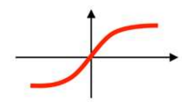
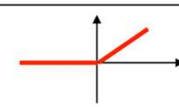
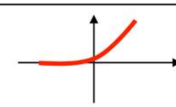
Sigmoid Activation Function

- Useful for output layer for classification problems
- A small change in input causes large change in output, at decision boundary
- Separates classes away from decision boundary
- Softmax performs a similar function for multi-class classification (3 or more classes)





More Activation Functions to choose from

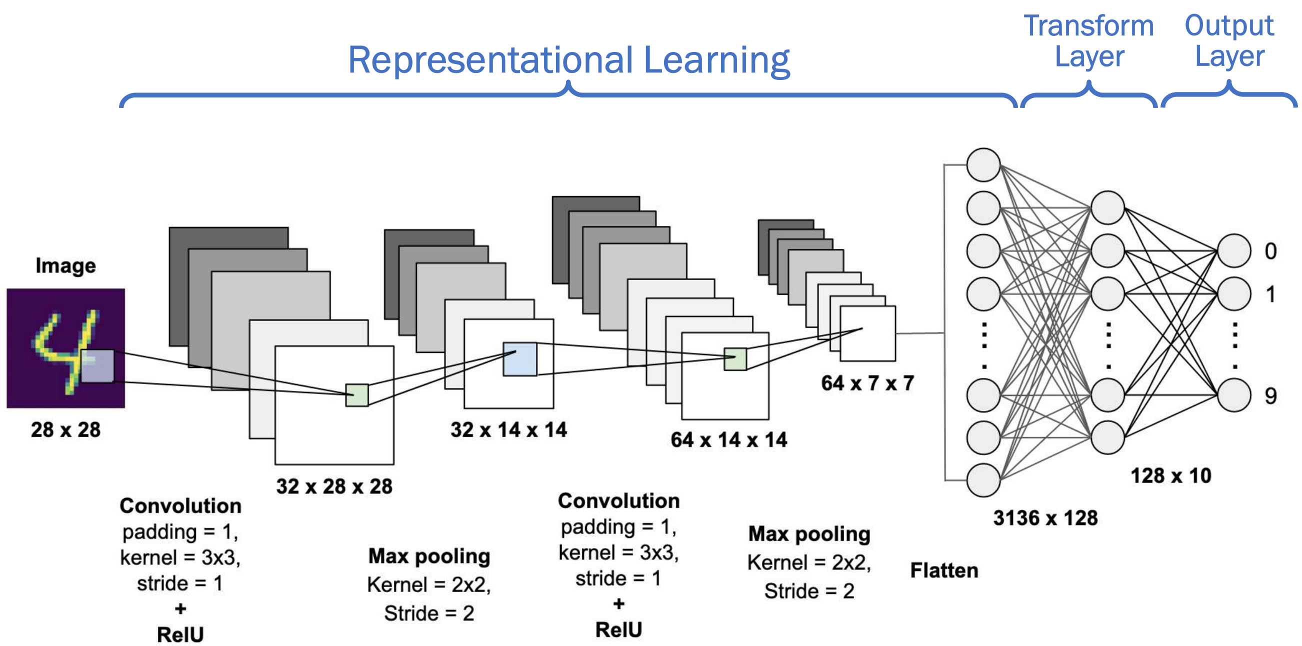
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Why do you need so many layers?

Feature Representation

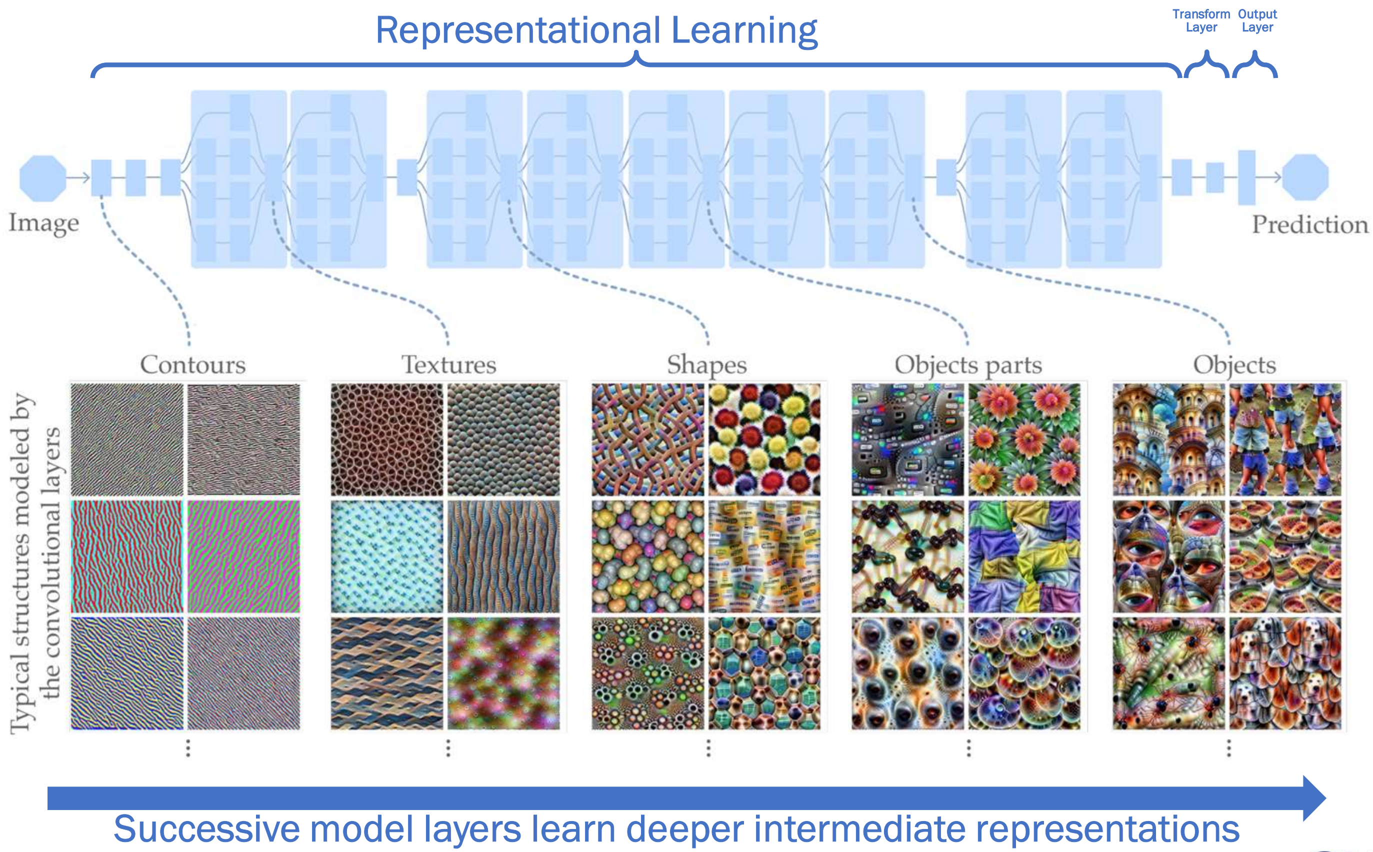
CUNY School of Professional Studies

Feature Representation



ConvNet=Convolutional Neural Network, used for images.

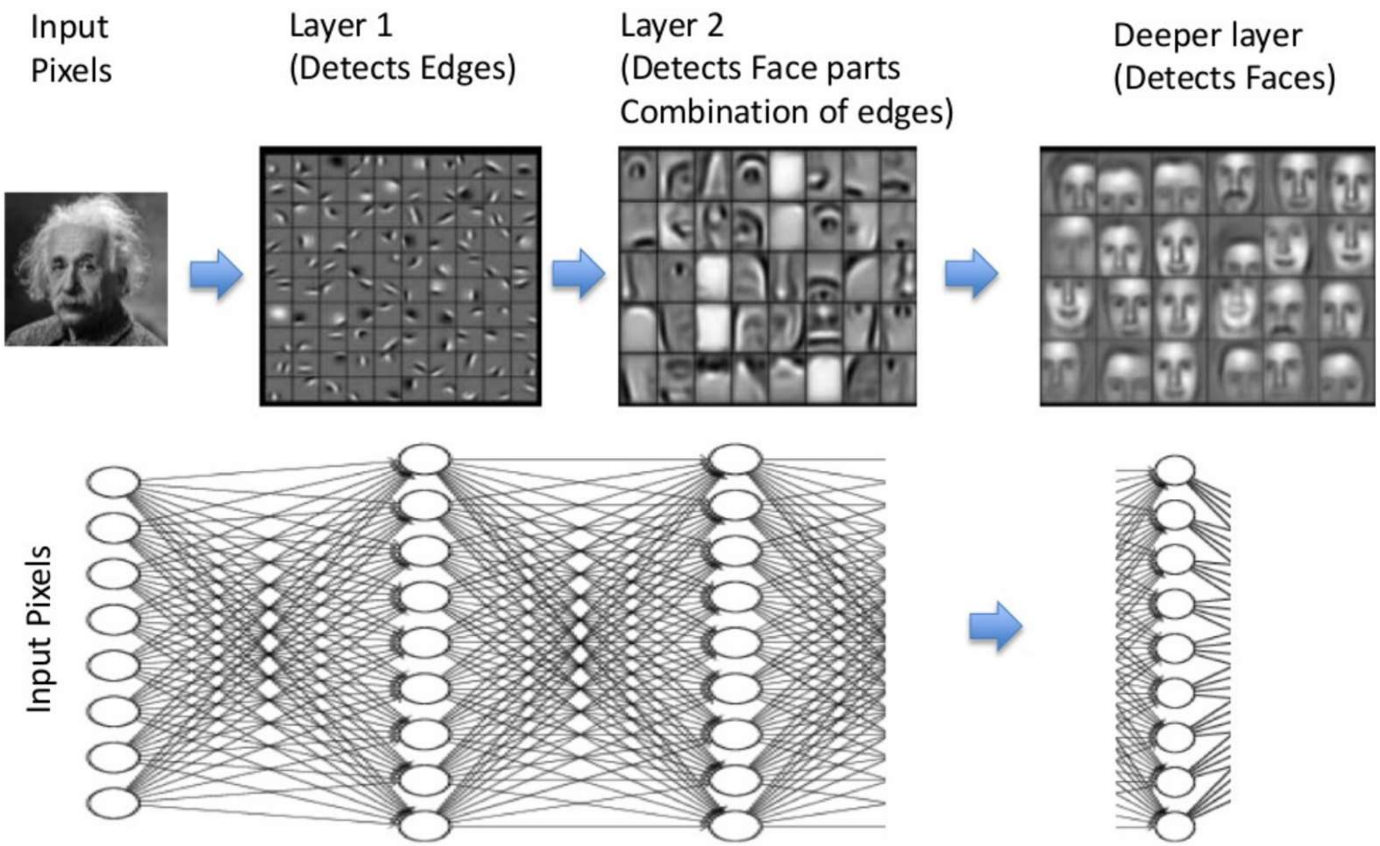
Feature Representation



Source: TLL
Source: Improving Latent Representations of ConvNets for Visual Understanding, Thomas Robert

9

Feature Representation



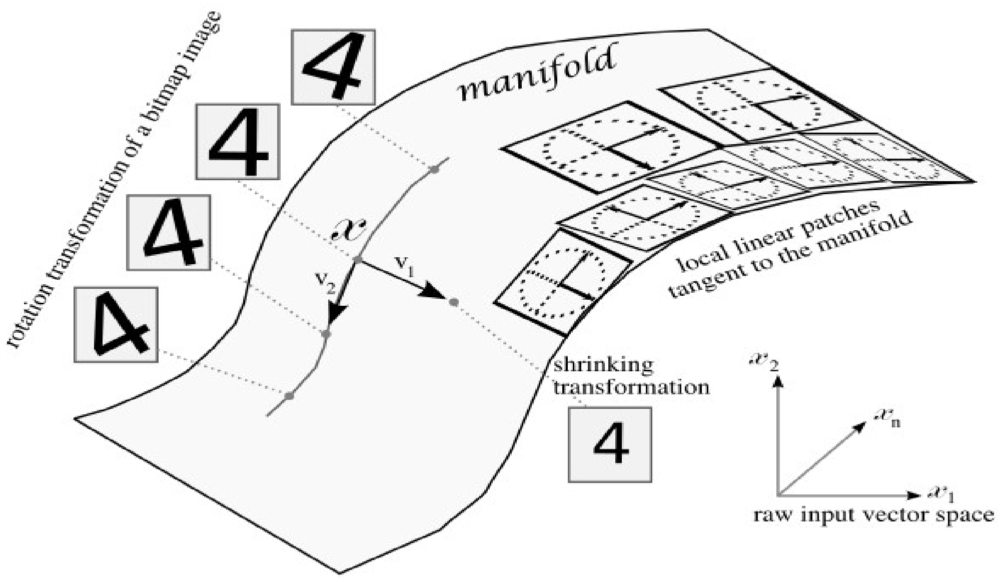
Manifold Hypothesis

Solution space of data

CUNY School of Professional Studies

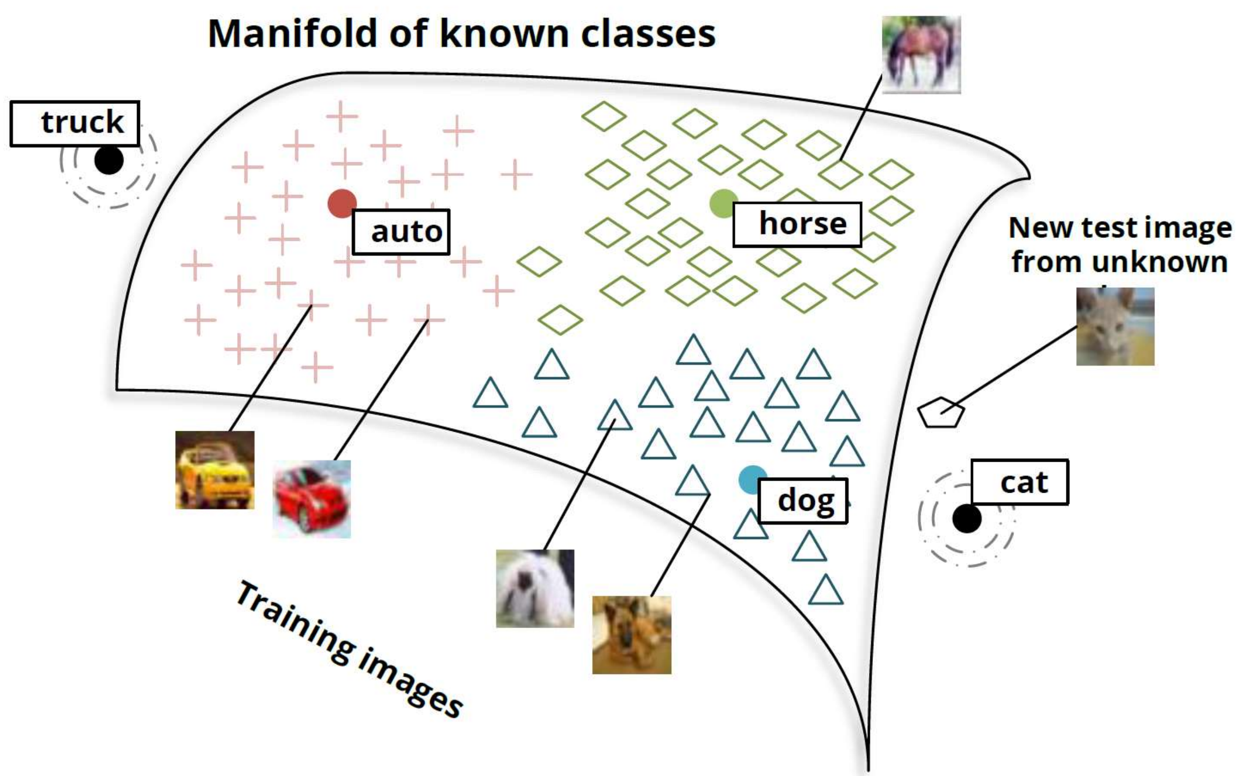
Manifold Hypothesis

- Manifold Hypothesis states that natural data forms lower dimensional manifolds in its embedding space.
- There are both theoretical and experimental reasons to suspect that the Manifold Hypothesis is true



CUNY School of Professional Studies

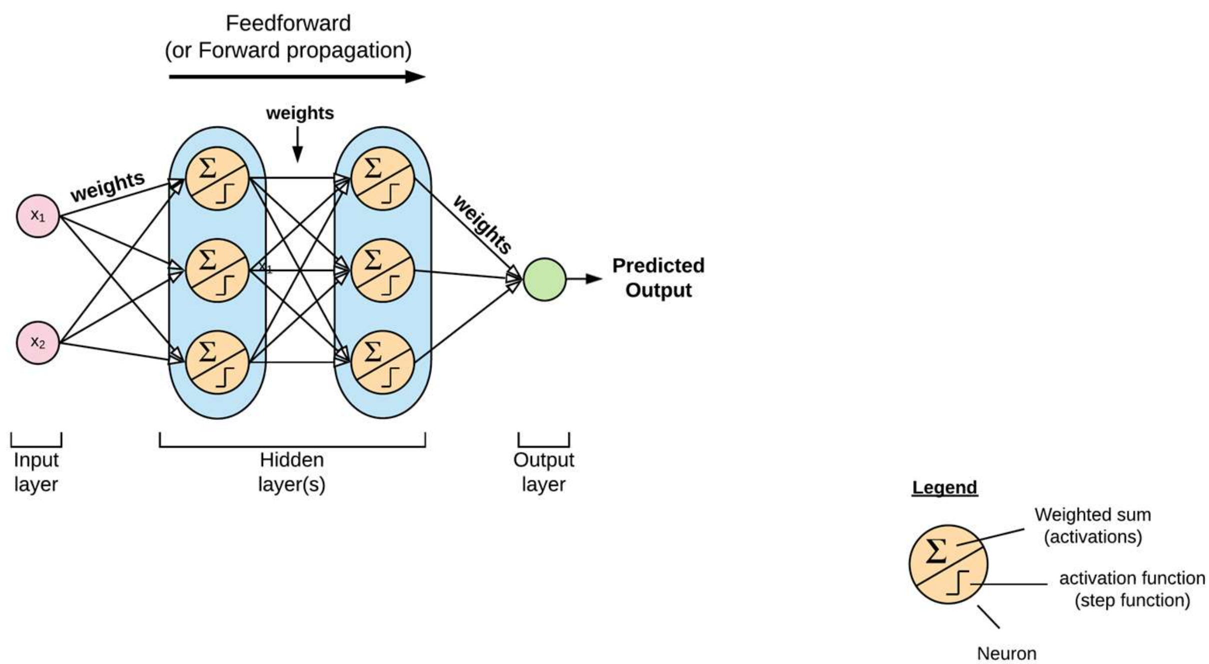
Another view of manifolds



Feedforward

How neural networks make predictions

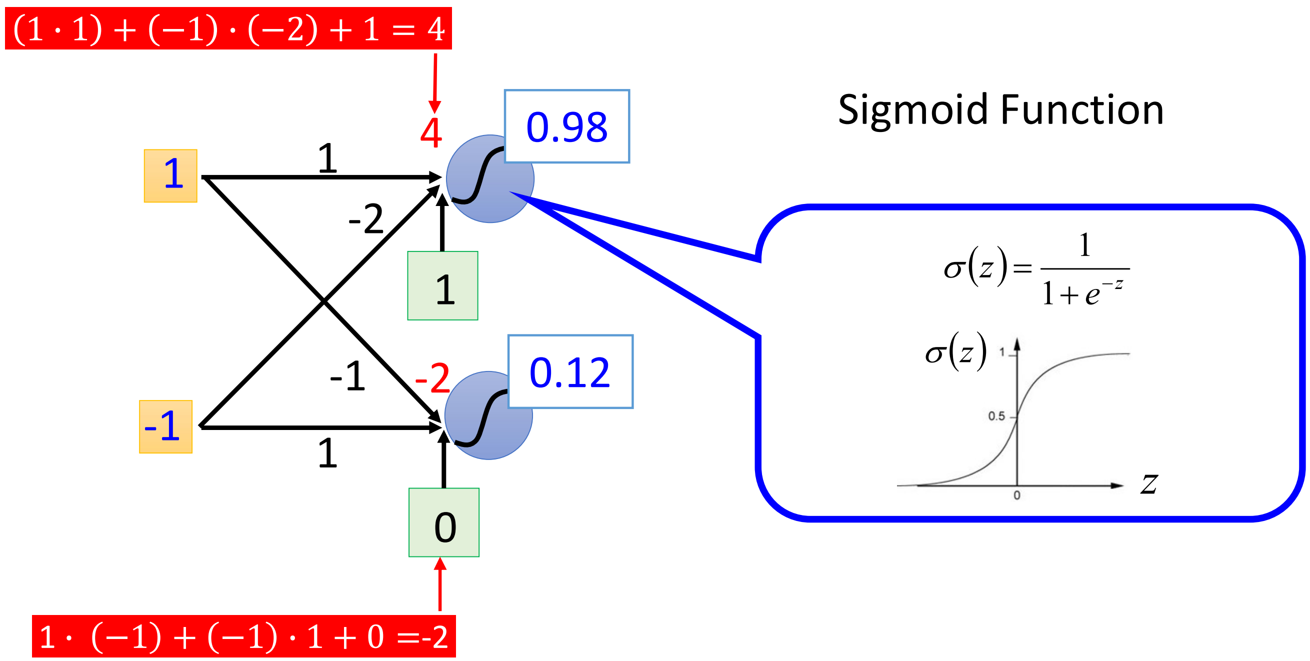
Feedforward



Source: https://ekababisong.org/gcp-ml-seminar/deep-learning/

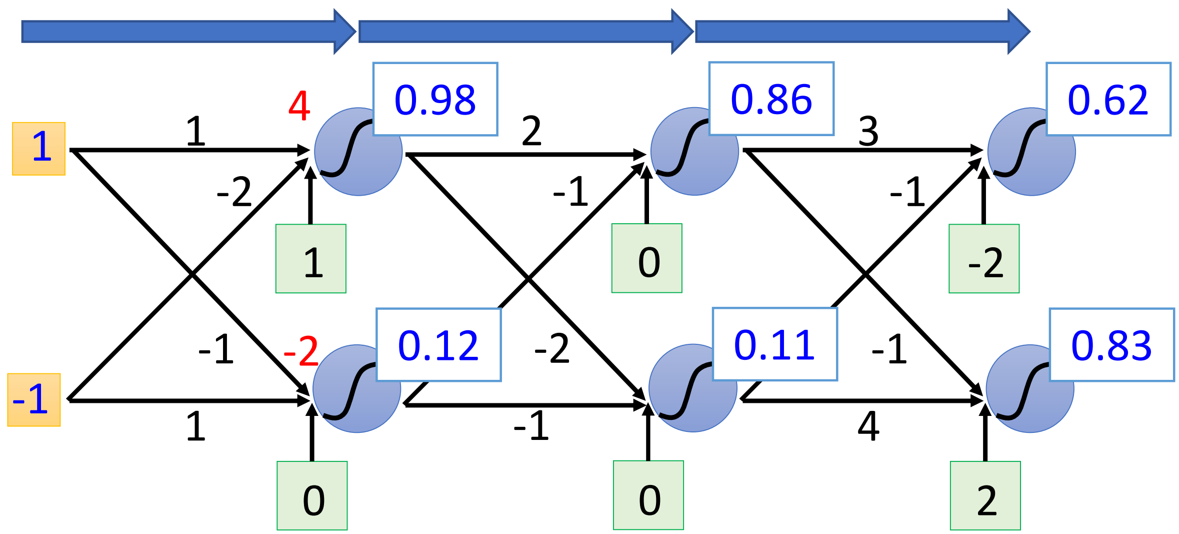
Feedforward

- A simple network, toy example



Source: Hung-yi Lee – Deep Learning Tutorial

Feedforward



$$f: R^2 \rightarrow R^2 \qquad f\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix}$$

Source: Hung-yi Lee – Deep Learning Tutorial

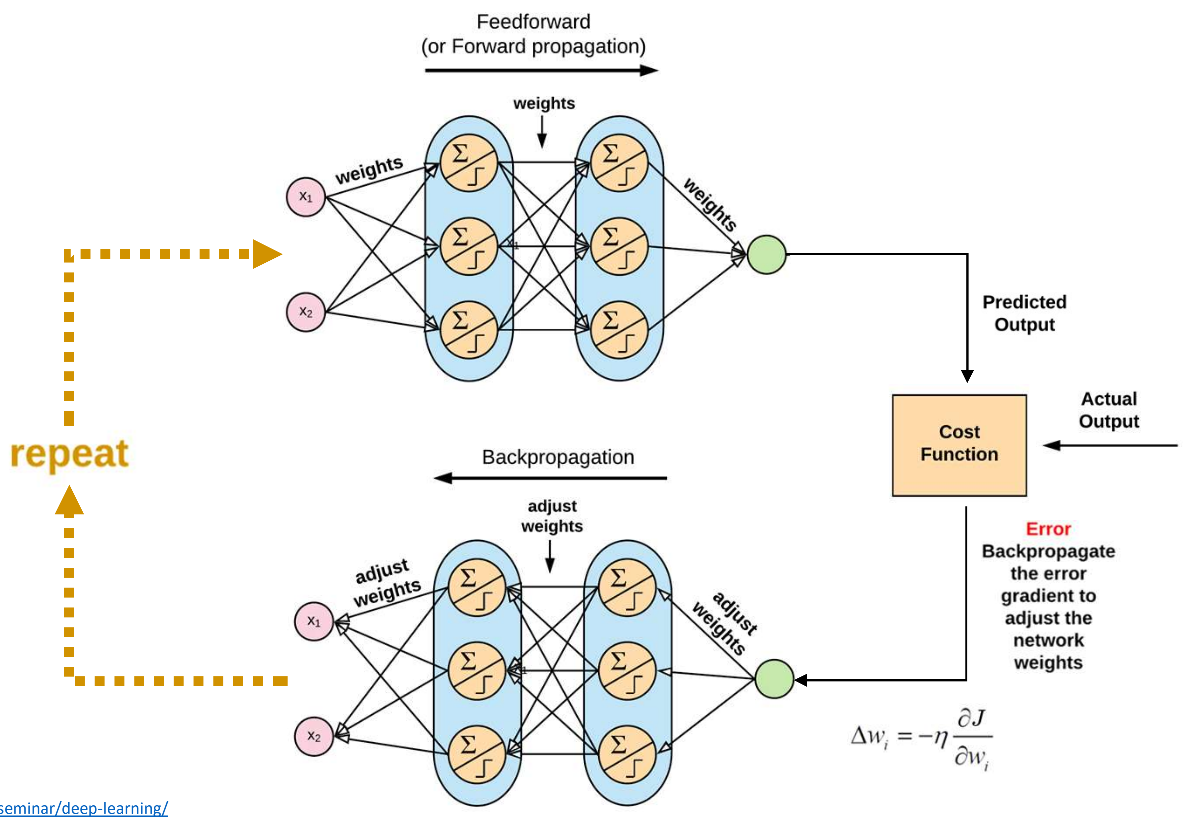
17

Backpropagation

How neural networks are trained

CUNY School of Professional Studies

Backpropagation



Source: https://ekababisong.org/gcp-ml-seminar/deep-learning/

CUNY School of Professional Studies

Backpropagation

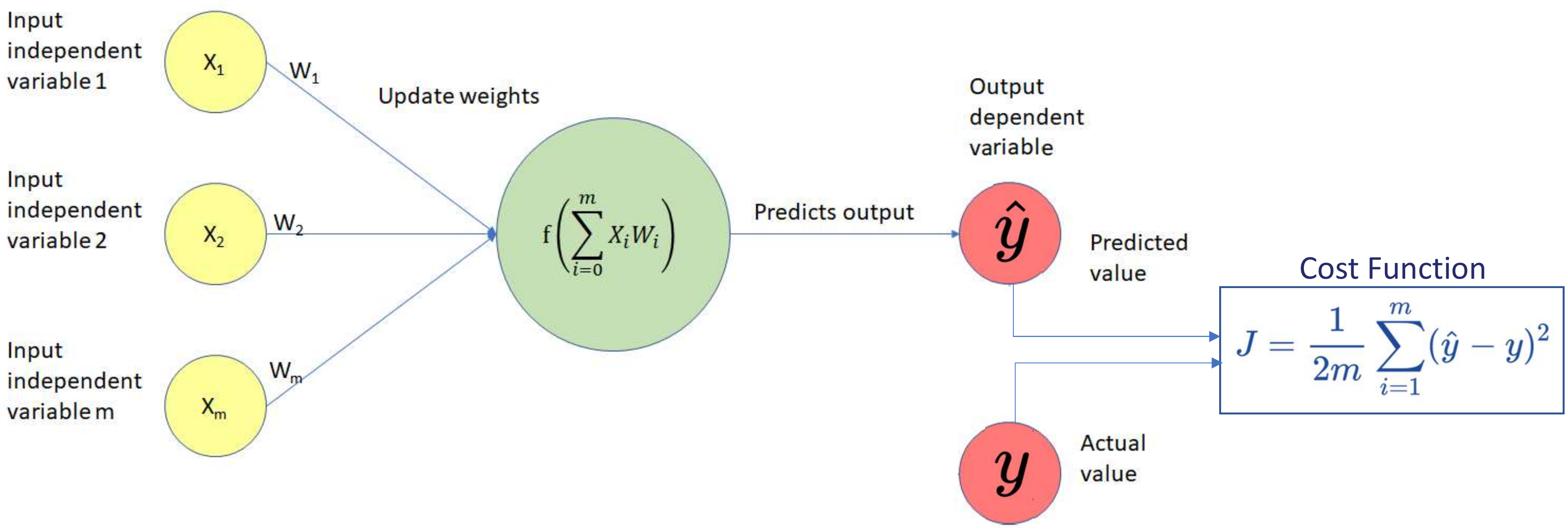
- Backpropagation is a type of gradient descent (terms used interchangeably)
- Gradient descent refers to the calculation of a gradient on each weight in the neural network for each training element.
  - Because the neural network will not output the expected value for a training element, the gradient of each weight will give you an indication about how to modify each weight to achieve the expected output.
  - If the neural network did output exactly what was expected, the gradient for each weight would be 0, indicating that no change to the weight is necessary.
- How it works:
  - Output of NN is evaluated against desired output
  - If results are not satisfactory, connection (weights) between layers are modified and process is repeated again and again until error is small enough.

Cost function

Tracking how your training is performing

CUNY School of Professional Studies

Cost Function (J)



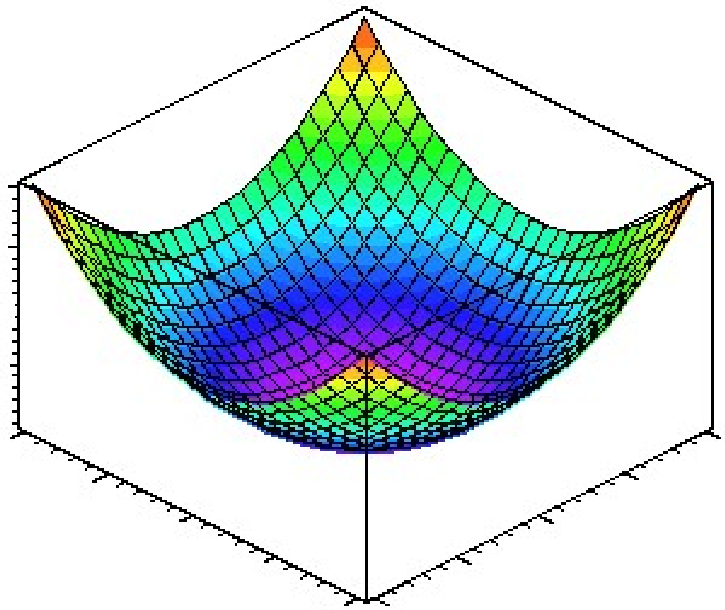
Cost Function

- Cost functions determine how well a model performs for a given dataset
- Cost Functions measure just how wrong the model is in finding a relation between the input and output.
- Comparing other functions:
  - Accuracy functions tell you how well your model does, not how to improve it
  - Error functions measure the difference between the target and the actual values e.g. $(\hat{y} - y)$
  - Loss functions quantify the cost for a single training example e.g. $(\hat{y} - y)^2$
  - Cost functions quantify the average across the entire dataset
    e.g.
    $$J = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y} - y)^2$$
  - Loss functions quantify the impact of the error i.e. error is objective while loss is subjective e.g. we might adopt a non-symmetric loss function if we may be more negatively affected by an error in a particular direction (e.g., false positive vs. false negative)

Cost Function

- MSE Cost function will always be parabolic (by definition)
  - So it has only one global minimum

$$J = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y} - y)^2$$



- SSE – Similar to MSE but not an average

$$J = \frac{1}{2} \sum_{i=1}^{m} (\hat{y} - y)^2$$

Note: The 2 we added in the denominator makes the mathematics cleaner.

Gradient Descent Rule

Gradient descent rule:

$$w_i \leftarrow w_i + \Delta w_i$$

Where:

$$\Delta w_i = -\eta \frac{\partial J}{\partial w_i}$$

η is a positive constant called the *learning rate*, and determines step size of gradient descent search

Cost Function

$$SSE = \frac{1}{2}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2$$

$$SSE = \frac{1}{2}\sum_{j=1}^{n}\left(y_j - \left(\sum_{i=0}^{m}\phi\left(w_i^T \times x_{j,i}\right)\right)\right)^2$$

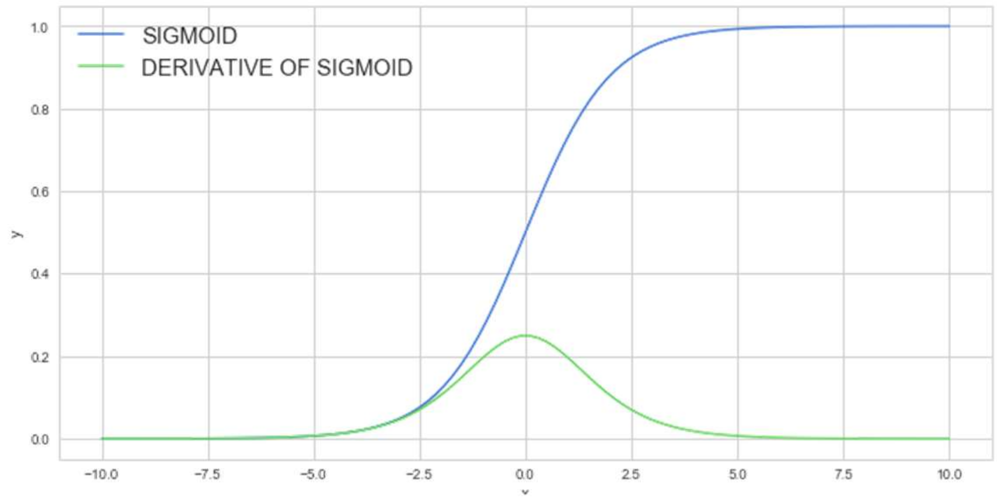
$$\frac{\partial\, SSE}{\partial\, w_i} = \sum_{j=1}^{n}\left(\underbrace{\left(y_j - \hat{y}_j\right)}_{\substack{\textit{error of the}\\ \textit{output of the}\\ \textit{weighted sum}}} \times \underbrace{-x_{j,i}}_{\substack{\textit{rate of change of}\\ \textit{weighted sum}\\ \textit{with respect to}\\ \textit{change in } w_i}}\right)$$

Derivative of the Sigmoid Function

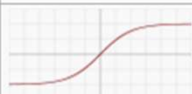
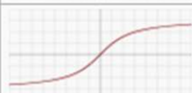
$$y = \frac{1}{1+e^{-x}}$$

$$\frac{dy}{dx} = -\frac{1}{(1+e^{-x})^2}(-e^{-x}) = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right) = y(1-y)$$



Derivatives of other Activation Functions

Name	Plot	Equation	Derivative
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Cost Function

$$\mathrm{SSE} = \frac{1}{2}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2$$

$$\mathrm{SSE} = \frac{1}{2}\sum_{j=1}^{n}\left(y_j - \left(\sum_{i=0}^{m}\phi\left(w_i^T \times x_{j,i}\right)\right)\right)^2$$

$$\frac{\partial\,\mathrm{SSE}}{\partial\,w_i} = \sum_{j=1}^{n}\left(\underbrace{\left(y_j - \hat{y}_j\right)}_{\substack{\textit{error of the}\\ \textit{output of the}\\ \textit{weighted sum}}} \times \underbrace{-x_{j,i}}_{\substack{\textit{rate of change of}\\ \textit{weighted sum}\\ \textit{with respect to}\\ \textit{change in } w_i}}\right)$$

CUNY School of Professional Studies

Gradient Descent Rule or Sigmoid

$$w_i^{t+1} = w_i^t + \left(\eta \times \underbrace{\sum_{j=1}^{n} \left((y_j^t - \hat{y}_j^t) \times \underbrace{(\hat{y}_j^t \times (1 - \hat{y}_j^t))}_{\text{Derivative of the sigmoid activation function with respect to the weighted sum}} \times x_{j,i}^t \right)}_{\text{Error gradient for } w_i} \right)$$

CUNY School of Professional Studies

30

Cost Function

$$w_i^{t+1} = w_i^t + \left(\eta \times \underbrace{\sum_{j=1}^{n} \left(\left(y_j^t - \hat{y}_j^t \right) \times x_{j,i}^t \right)}_{\text{error gradient for } w_i} \right)$$

31