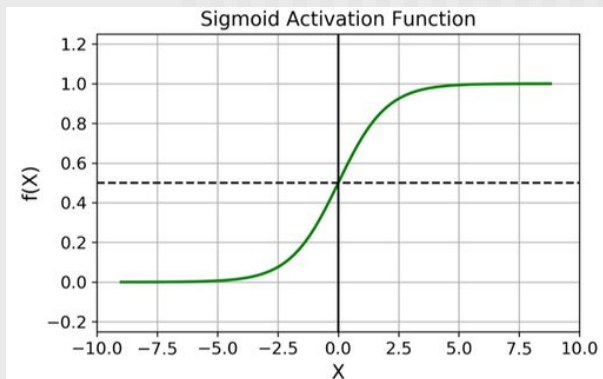
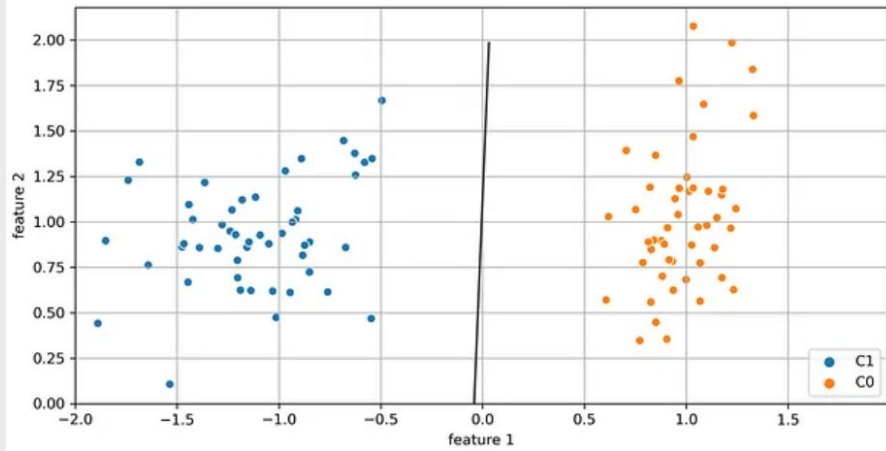


Lecture 5. Intro to Neural Networks

Suthep “Jogie” Madarasmi, Ph.D.

Linear Decision boundaries for Logistic Regression



$$x = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

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

$$y = 1 \text{ if } g(x) \geq 0.5$$

$$\text{if } \theta^T x \geq 0$$

$$\text{if } \theta_1 x_1 + \theta_2 x_2 \geq -$$

$$\theta_0$$

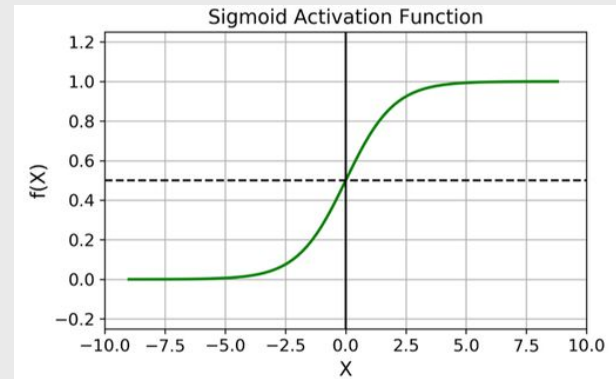
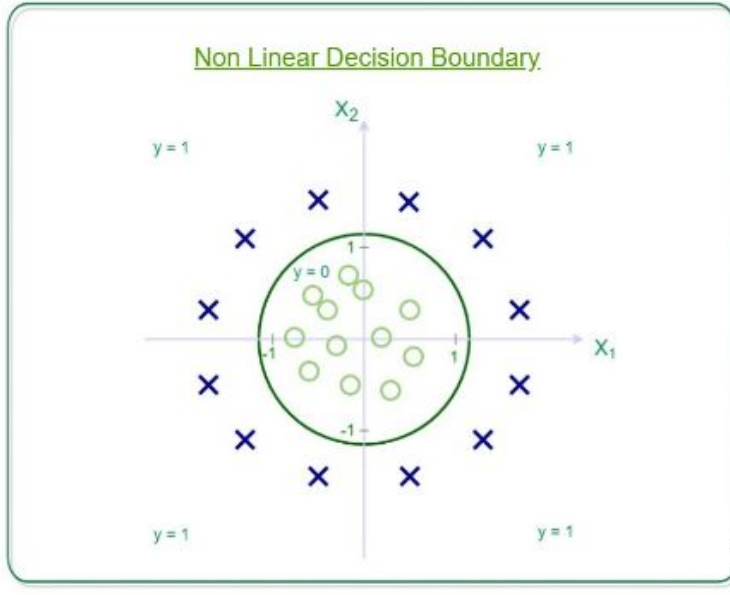
Circular Decision Boundary in Logistic Regression

$$h_{\Theta}(x) = g[\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_1^2 + \Theta_4 x_2^2]$$

$$h_{\Theta}(x) = g(\Theta^T x)$$

$$g(z) \geq 0.5 \rightarrow y = 1$$

$$\Theta^T x \geq 0$$



$$\Theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

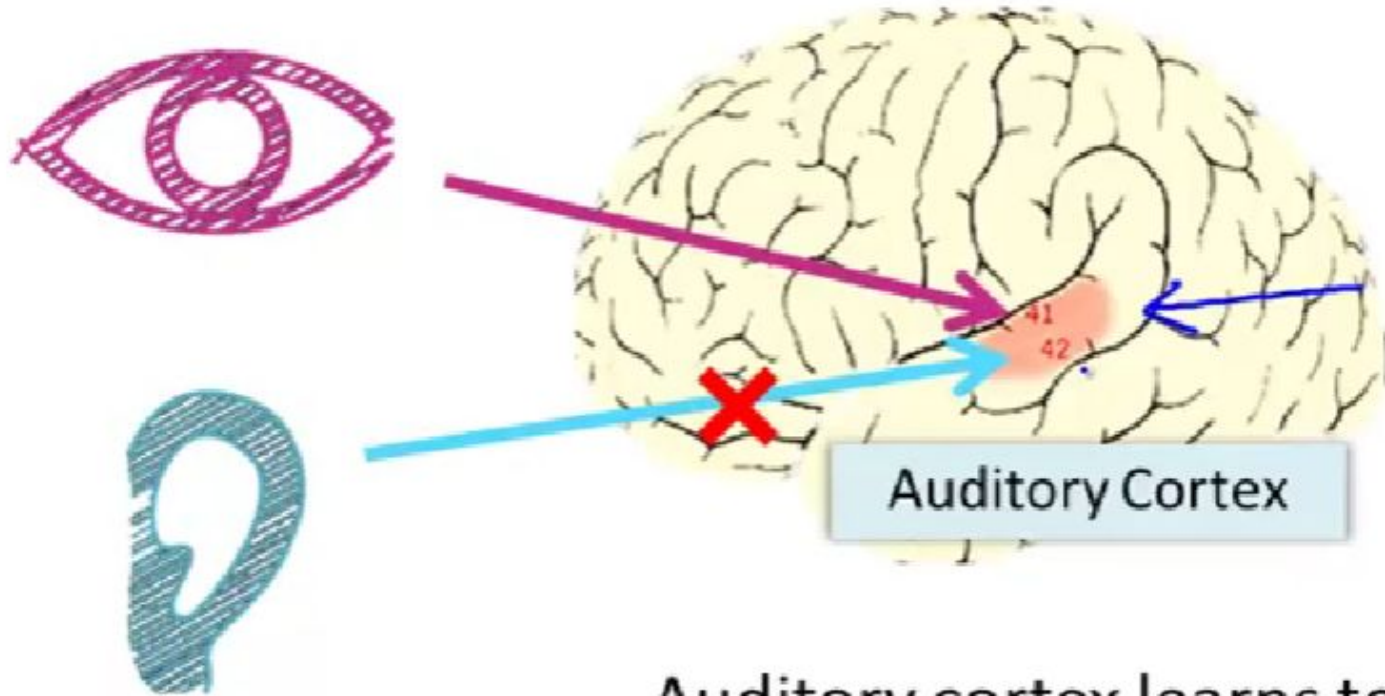
Above θ predicts $y = 1$ if:

$$\begin{aligned} -1 + x_1^2 + x_2^2 &\geq 0 \\ \Rightarrow x_1^2 + x_2^2 &\geq 1 \end{aligned}$$

Neural Networks

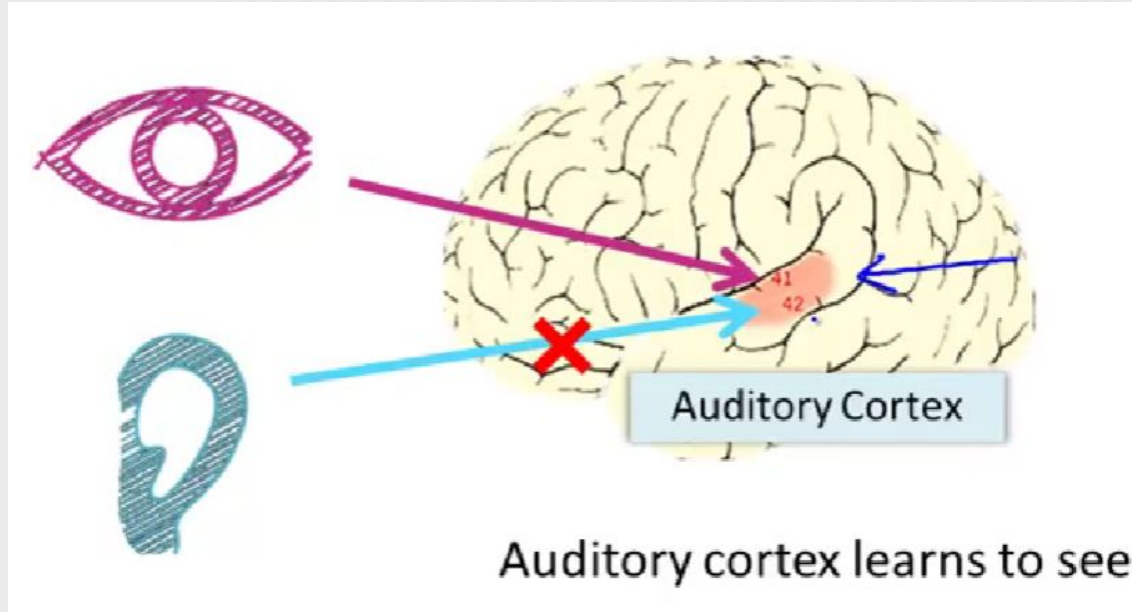
- Was popular in 70's. Then died due to limitation of the perceptron.
- Then came to backpropagation network (multi-layer perceptron). Interest rose again in 80's to 90's. Died again because of limitations.
- In 2010's gained popularity again with convolutional nets and deep learning.

Brain uses a singular learning algorithm



Auditory cortex learns to see

Brain uses a singular learning algorithm



Similar for sense of touch.

Experiments: Neural-rewiring.

Appears the brain uses a single learning algorithm for sight, sound, touch, etc.

Examples of brain relearning

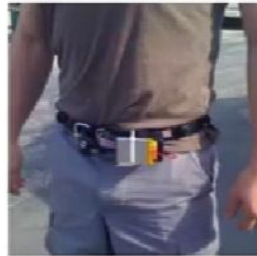
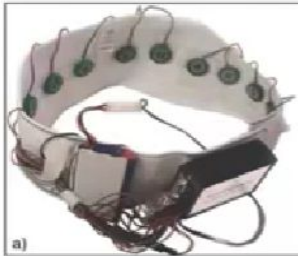
Sensor representations in the brain



Seeing with your tongue



Human echolocation (sonar)



Haptic belt: Direction sense

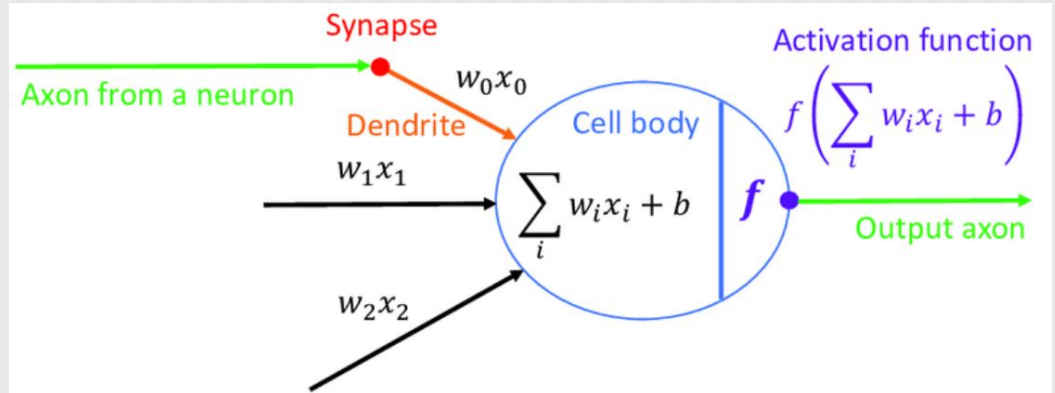
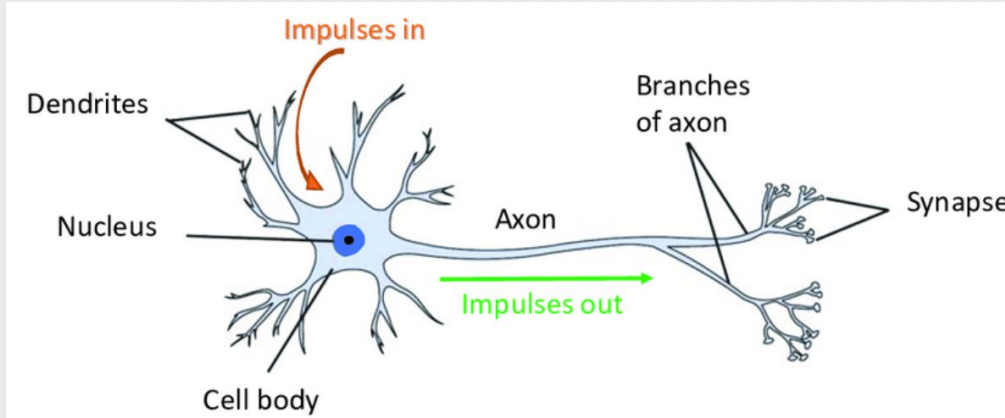


Implanting a 3rd eye

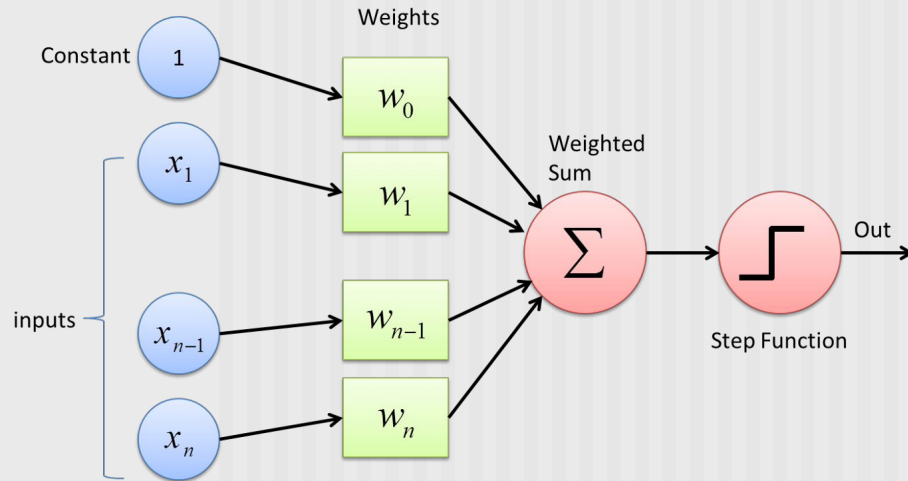
brain port. <https://www.youtube.com/watch?v=rklHXOwO83M>

echolocation. <https://www.youtube.com/watch?v=a05kgcI9D2Q>

Neural Network Computational Model



The Perceptron Model



$$y = \begin{cases} 1 & \text{if } w_1x_1 + \dots + w_nx_n \geq w_0 \\ -1 & \text{if } w_1x_1 + \dots + w_nx_n < w_0 \end{cases}$$

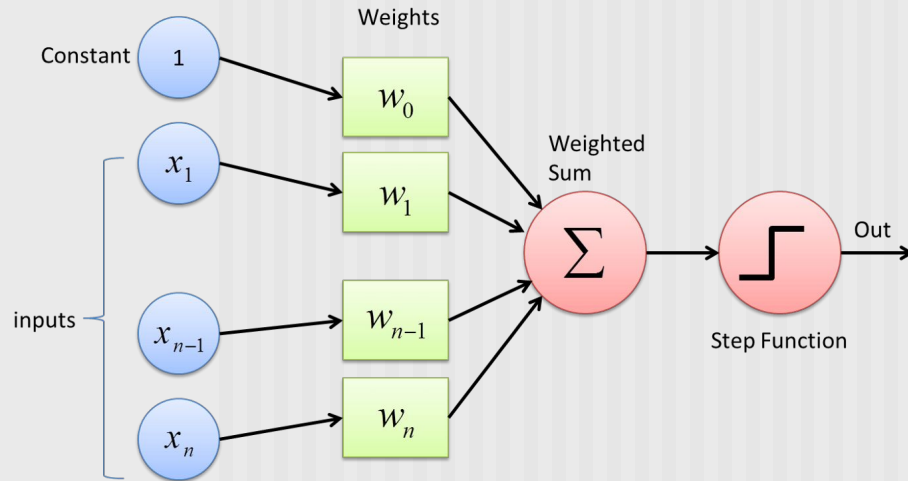
$$y = \begin{cases} 1 & \text{if } w_0x_0 + w_1x_1 + \dots + w_nx_n \geq 0; x_0 = 1 \\ -1 & \text{if } w_0x_0 + w_1x_1 + \dots + w_nx_n < 0 \end{cases}$$

$$\vec{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}; \vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

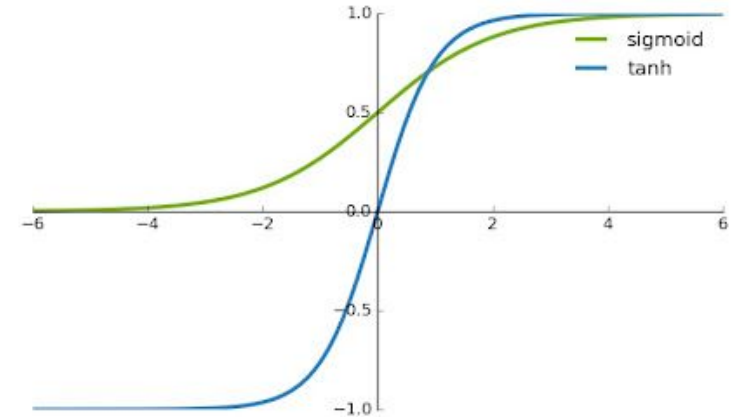
$$z = w_0x_0 + w_1x_1 + \dots + w_nx_n; \text{ where } x_0 = 1$$

$$= \sum_{i=0}^n w_i x_i = \vec{w}^T \vec{x}$$

The Step Function

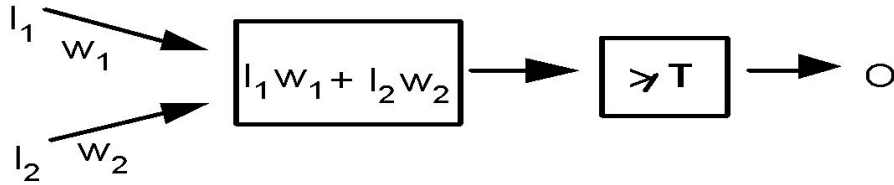


$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \quad \sigma(x) = \frac{1}{1 + e^{-x}}.$$



- Unlike Logistic Regression, the output of Perceptron is **more often** shown as -1 or 1.
- The step function can be changed tanh(x) which is the same as: $\tanh(x) = 2 * (\text{sigmoid}(2x) - 0.5) = 2 * \text{sigmoid}(2x) - 1$.

Desired Weights to Classify: And, Or, Not



I_1	I_2	AND	OR	NOT I_1
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0
w_1		1	1	-1
w_2		1	1	0
threshold		1.5	0.5	-0.5

$$I_1w_1 + I_2w_2 \geq T$$

$$I_1w_1 + I_2w_2 + 1 \cdot (-T) \geq 0$$

Goal: Find these weights through training.

Solving weights in AND problem

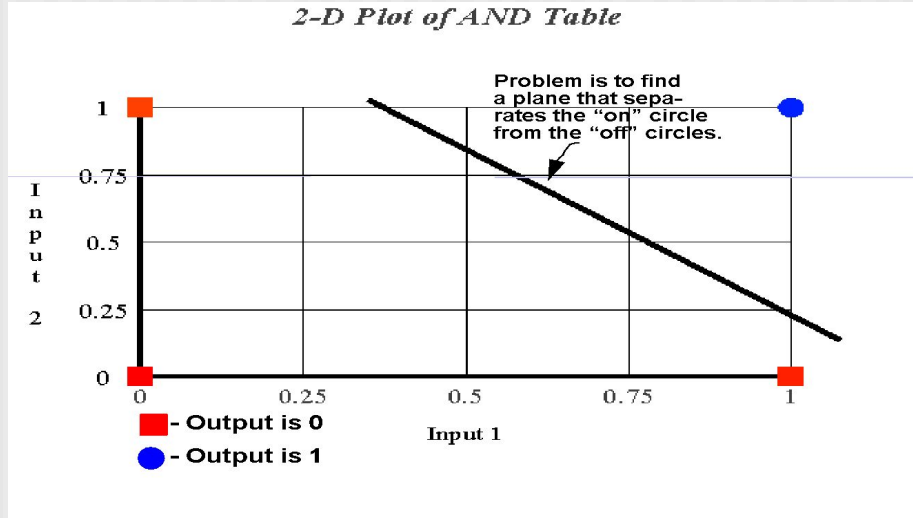
I_1	I_2	O	Eqn: $I_1 w_1 + I_2 w_2 + 1 \cdot w_3$
0	0	0	$0 \cdot w_1 + 0 \cdot w_2 + 1 \cdot w_3 < 0 \Rightarrow w_3 < 0$
0	1	0	$0 \cdot w_1 + 1 \cdot w_2 + 1 \cdot w_3 < 0 \Rightarrow w_2 + w_3 < 0$
1	0	0	$1 \cdot w_1 + 0 \cdot w_2 + 1 \cdot w_3 < 0 \Rightarrow w_1 + w_3 < 0$
1	1	1	$1 \cdot w_1 + 1 \cdot w_2 + 1 \cdot w_3 \geq 0 \Rightarrow w_1 + w_2 + w_3 \geq 0$

Learning is to find $\mathbf{w} = (w_1, w_2, w_3)$ which satisfy these 4 constraints.

$$\mathbf{w} = (1, 1, -1.5)$$

Decision boundary for AND Table

Scatter Plot for 2 Classes (0, 1):



A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Training Procedure: Find \mathbf{w} 's.

1. First assign any values to w_1 , w_2 and w_3
2. Using the current weight values w_1 , w_2 and w_3 and the next training data inputs I_1 and I_2 compute:

$$V = I_1 w_1 + I_2 w_2 + 1 \cdot w_3$$

3. If $V > 0$ set computed output C to 1 else set to 0.
4. If the computed output C is not the same as the training output O , **Adjust Weights.**
5. Repeat steps 2-4. If you run out of training items, start with the first training item. Stop repeating if no weight changes through 1 complete training cycle.

Adjust Weights using Gradient Descent

$$V = I_1 w_1 + I_2 w_2 + 1 \cdot w_3$$

$$C = \tau(V) = \tau(I_1 w_1 + I_2 w_2 + 1 \cdot w_3)$$

$$\begin{aligned} E &= \sum_{i=1}^4 (C_i - O_i)^2 \\ &= \sum_{i=1}^4 (\tau(I_{1i} w_1 + I_{2i} w_2 + w_3) - O_i)^2 \end{aligned}$$

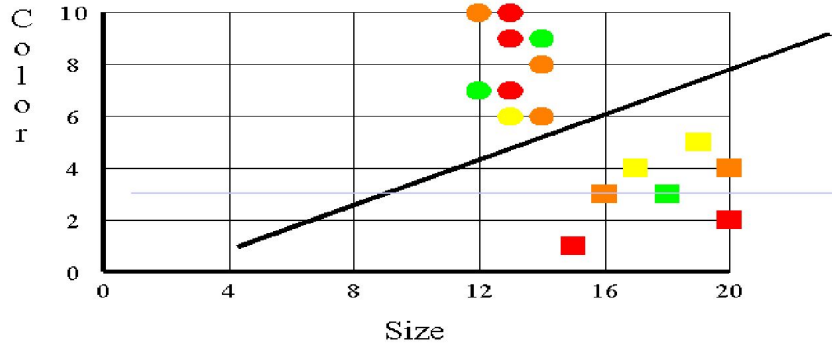
$$w_{1Next} = w_{1Current} - I_1(C - O)$$

$$w_{2Next} = w_{2Current} - I_2(C - O)$$

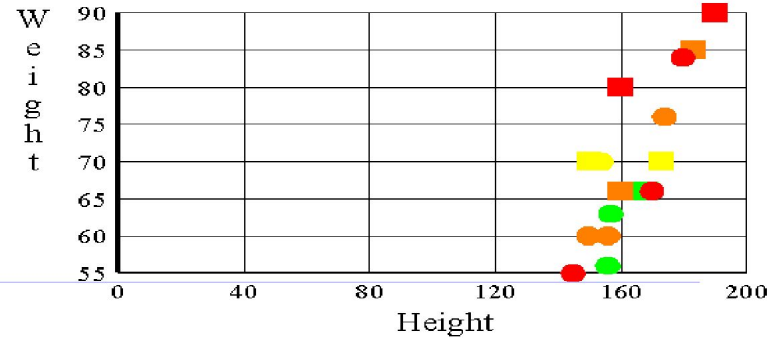
$$w_{3Next} = w_{3Current} - (C - O)$$

Linearly vs. Non-Linearly Separable

Linearly Separable: Basketball vs. Volleyball

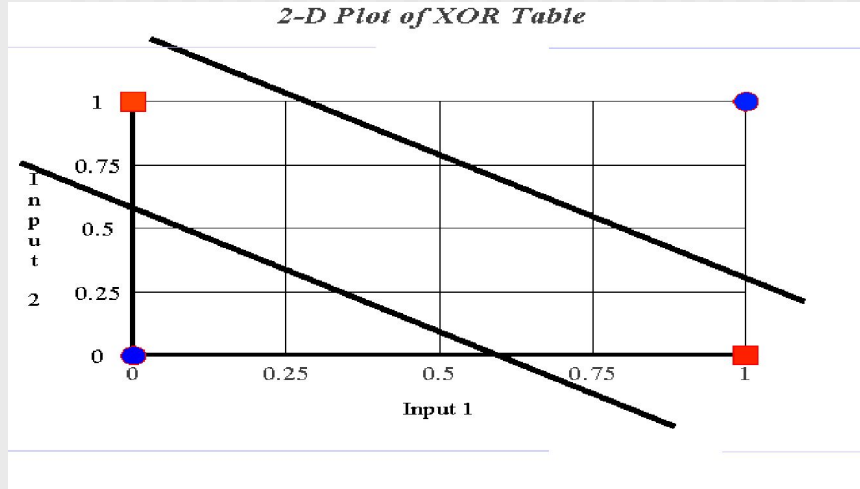


Linearly Non-Separable: Male vs. Female



Classification: Using features decide which classification. Often involves training such as N-net case.

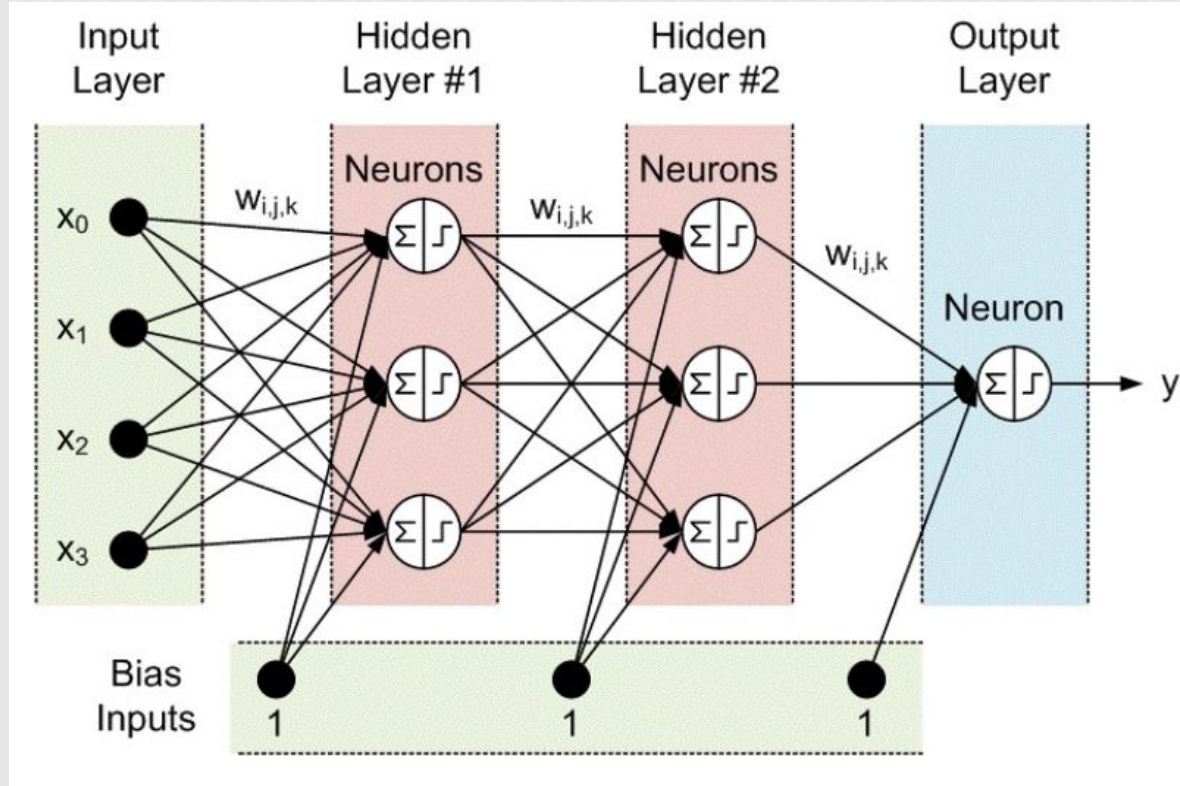
XOR is Not Linearly Separable



A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

The inability to solve this simple classification of XOR revealed the weakness of the perceptron model.

Example of Multi-layer Perceptron (MLP)



Also called **Feedforward** Neural Networks and **Back Propagation** Neural Networks.

Perceptron vs. MLP Models

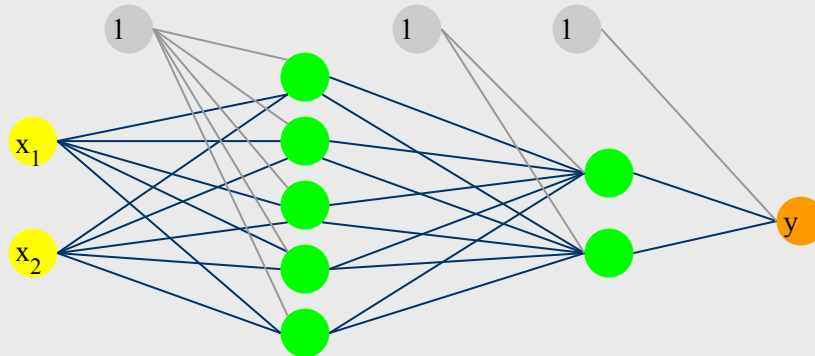
Perceptron (P)



Feed Forward (FF)



Yellow: Input layer | **Green:** Hidden layer | **Orange:** Output layer



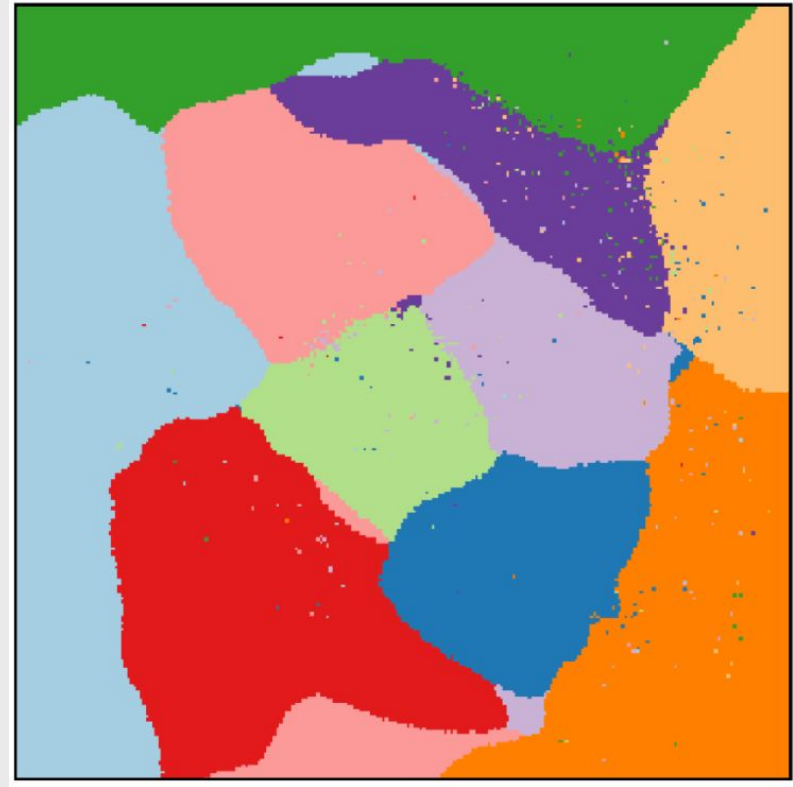
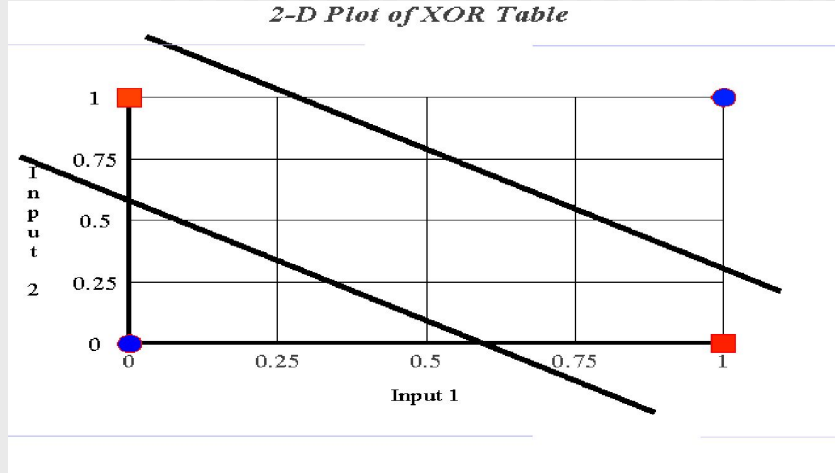
Shape of W: $(2, 5), (5, 2), (2, 1)$

Shape of Bias: $[(5,), (2,), (1,)]$

Alternative View:

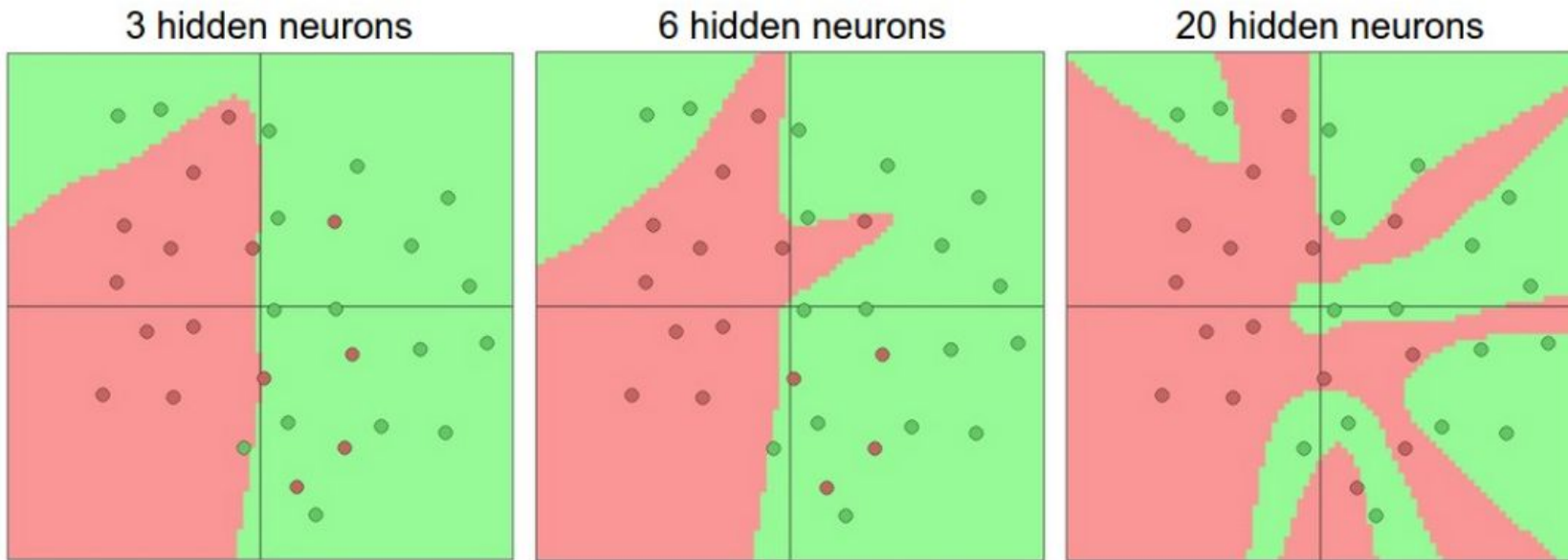
Shape of W: $(3, 5), (6, 2), (3, 1)$

Advantage of Backprop over Perceptron




Research proves that mathematically only 1 hidden layer is sufficient, but convergence (learning) is easier with more hidden layers or “deeper” nets.

More Hidden Neurons \Rightarrow Model More Complex Functions



Larger Neural Networks can represent more complicated functions. The data are shown as circles colored by their class, and the decision regions by a trained neural network are shown underneath. You can play with these examples in this [ConvNetsJS demo](#).

MLP Examples

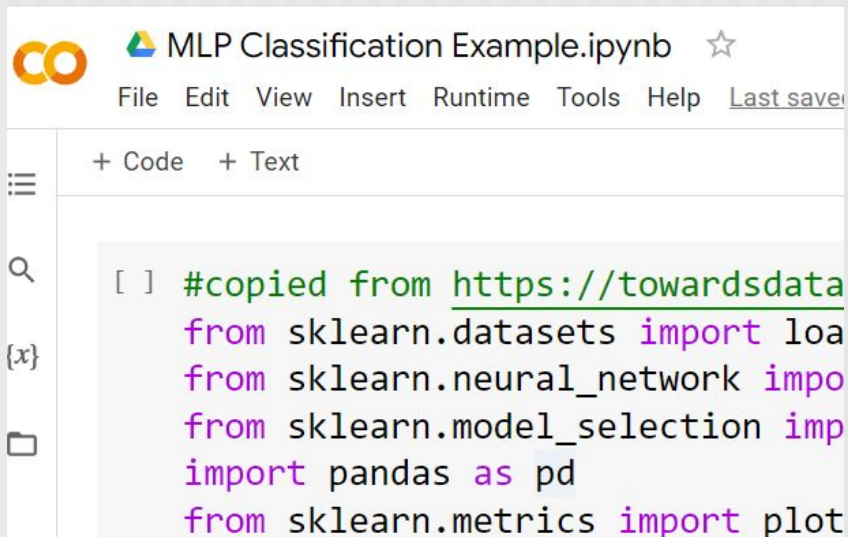

MLP Intro.ipynb ☆

File Edit View Insert Runtime Tools Help [Last saved at 1:35 PM](#)

+ Code + Text

[41] `import numpy as np`
`import matplotlib as plt`
`import math`


MLP Examples



The image shows a Google Colab interface for a file named "MLP Classification Example.ipynb". The interface includes a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help, and a "Last saved" status. Below the menu bar, there are tabs for "+ Code" and "+ Text". On the left side, there is a sidebar with icons for a menu, search, a variable "{x}", and a folder. The main code area contains the following Python code:

```
[ ] #copied from https://towardsdata
from sklearn.datasets import load
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.metrics import plot
```

MLP Examples



MLP Regression Example.ipynb ☆

File Edit View Insert Runtime Tools Help Li

+ Code + Text

```
[ ] #copied from https://toward  
from sklearn.neural_network  
from sklearn.model_selection  
from sklearn.datasets import
```


Quiz 6



King Mongkut's University of Technology

Machine Learning

Suthep Madarasmi, Ph.D.

Take Home Quiz 6 Due Sun Mar 10, 2024

Name: _____

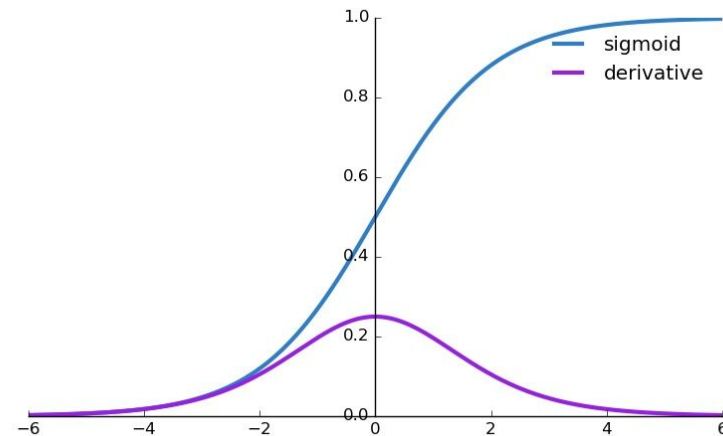
I.D. Number: _____

Score: _____ / 60

1. *15 points.* 1 hour. Using MLP Intro.ipynb, show that the OR, AND, NOT will work for a single layer perceptron, but the XOR will not be able to find a solution. For “NOT” your input will have 1 value, but for “OR”, “AND”, and ”XOR” it will have 2 values.
2. *15 points.* 1 hour. Use multilayer perceptron on sklearn's diabetes data meant for regression using 25% test data, find the R-square.

Derivative of Sigmoid Function

$$\begin{aligned}
 \frac{d}{dx} \sigma(x) &= \frac{d}{dx} \left[\frac{1}{1 + e^{-x}} \right] = \frac{d}{dx} (1 + e^{-x})^{-1} \\
 &= -1 * (1 + e^{-x})^{-2} (-e^{-x}) \\
 &= \frac{-e^{-x}}{-(1 + e^{-x})^2} \\
 &= \frac{e^{-x}}{(1 + e^{-x})^2} \\
 &= \frac{1}{1 + e^{-x}} \frac{e^{-x}}{1 + e^{-x}} \\
 &= \frac{1}{1 + e^{-x}} \frac{e^{-x} + (1 - 1)}{1 + e^{-x}} \\
 &= \frac{1}{1 + e^{-x}} \frac{(1 + e^{-x}) - 1}{1 + e^{-x}} \\
 &= \frac{1}{1 + e^{-x}} \left[\frac{(1 + e^{-x})}{1 + e^{-x}} - \frac{1}{1 + e^{-x}} \right] \\
 &= \frac{1}{1 + e^{-x}} \left[1 - \frac{1}{1 + e^{-x}} \right] \\
 &= \sigma(x)(1 - \sigma(x))
 \end{aligned}$$

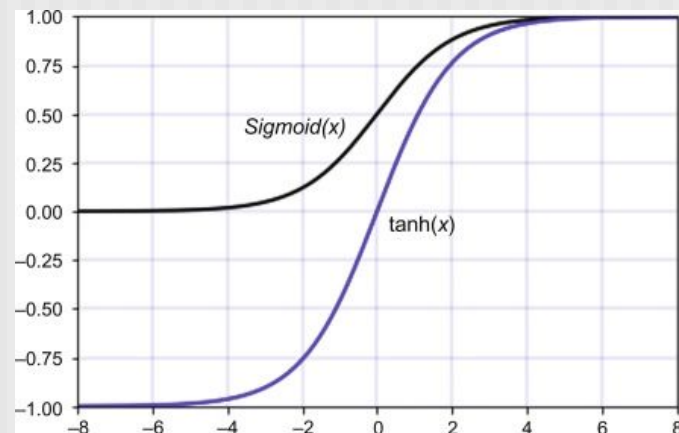


Derivative of Hyperbolic Tan

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad f(x) = \frac{2}{1 + e^{-2x}} - 1 = \tanh(x)$$

$$f'(x) = 1 - \tanh^2(x)$$



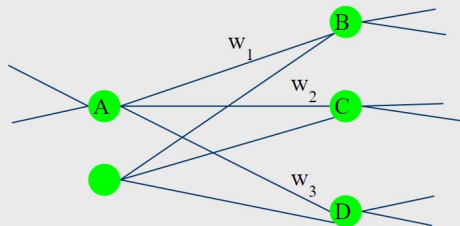
$$\begin{aligned} \frac{d}{dz} \left(\frac{e^z - e^{-z}}{e^z + e^{-z}} \right) &= \frac{e^z + e^{-z}}{(e^z + e^{-z})^2} d(e^z - e^{-z}) - \frac{e^z - e^{-z}}{(e^z + e^{-z})^2} d(e^z + e^{-z}) \\ &= \frac{(e^z + e^{-z})(e^z + e^{-z})}{(e^z + e^{-z})^2} - \frac{(e^z - e^{-z})(e^z - e^{-z})}{(e^z + e^{-z})^2} \\ &= \frac{(e^z + e^{-z})^2 - (e^z - e^{-z})^2}{(e^z + e^{-z})^2} \\ &= 1 - \left(\frac{e^z - e^{-z}}{e^z + e^{-z}} \right)^2 \\ &= 1 - \tanh^2(z) \end{aligned}$$

$$\tanh(x) = 2 \cdot \sigma(2x) - 1$$

$$\begin{aligned} \frac{d \tanh(x)}{dx} &= 2 \cdot \sigma(2x) \cdot (2 - 2 \cdot \sigma(2x)) \\ &= (\tanh(x) + 1) \cdot (1 - \tanh(x)) \\ &= 1 - \tanh^2(x) \end{aligned}$$

Backprop Learning Algorithm

1. Assign random values to all the weights
2. Choose a pattern from the training set (similar to perceptron).
3. Propagate the signal through to get final output (similar to perceptron).
4. Compute the error for the output layer (similar to the perceptron).
5. *Compute the errors in the preceding layers by propagating the error backwards.*
The error in hidden layer node A is computable by summing up the errors in B, C, and D weighted by the connection weights between A-B, A-C, and A-D, respectively.



6. Repeat step 2-5 for next training sample
7. Repeat steps 2-6 until not much change in weights for a run through **the entire training set**

Stochastic Gradient Descent (SGD)

$$w^{(\tau+1)} = w^{(\tau)} - \underbrace{\eta}_{\text{learning rate}} \cdot \nabla_w E(w)$$

$$E(w) = \sum_{n=1}^N E_n(w)$$

N denotes the size of dataset and $E_n(w)$ means the error evaluated when the algorithm processes the n -th data.

Descent through entire training set converges slower and is costly, so we break the training set into batch of n each, such as $n = 20$.

In **batch gradient descent** (or gradient descent). **n = size of training set.**

In **stochastic gradient descent**, **$n = 1$** for updating parameters per iteration.

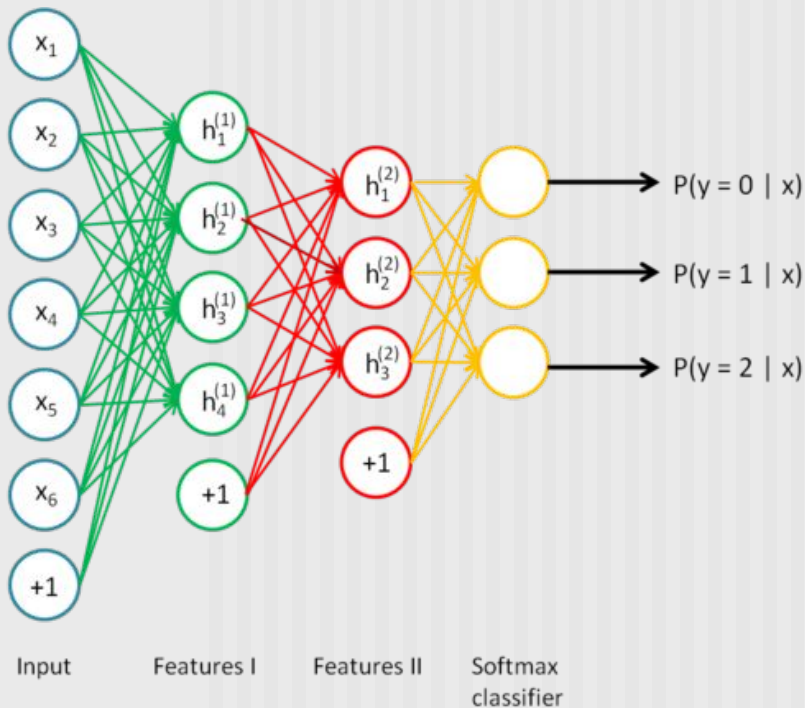
Use of subset of training dataset is called the **mini-batch SGD**. Strikes balance between GD and SGD.

Since we randomly select a subset of training samples from entire training set, this is Stochastic.

Gradient descent optimization uses all training samples per update.

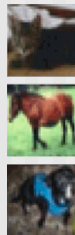
One Epoch is One complete pass through the entire training dataset.

Sigmoid at hidden nodes. Softmax at output node.



$$\text{Softmax}(z_j) = \frac{\sum e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$

Input pixels, x



Shape: (3, 32, 32)

Feedforward output, y_i

	cat	dog	horse
Forward propagation	5	4	2
	4	2	8
	4	4	1

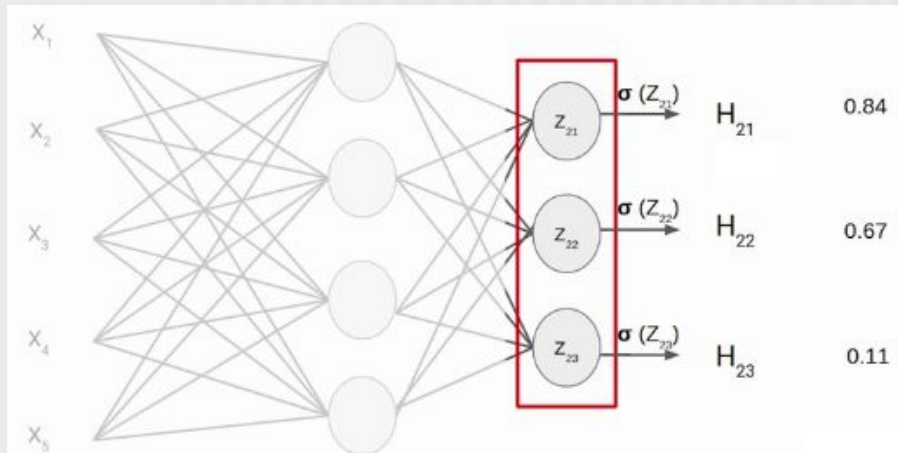
Shape: (3,)

Softmax output, $S(y_i)$

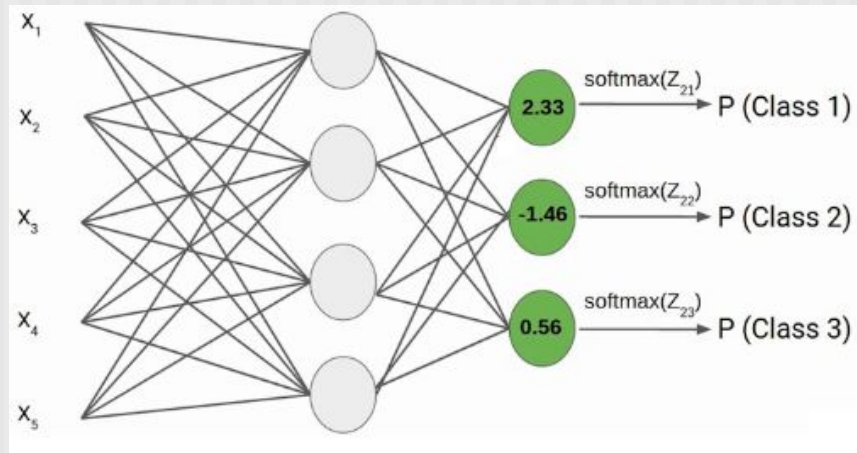
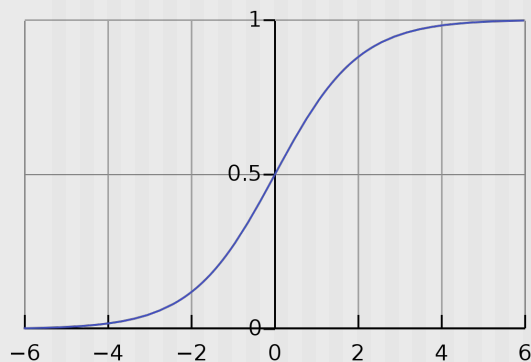
	cat	dog	horse
Softmax function	0.71	0.26	0.04
	0.02	0.00	0.98
	0.49	0.49	0.02

Shape: (3,)

Sigmoid function vs. Softmax function



sigmoid



$$2.33 \rightarrow P(\text{Class 1}) = \frac{\exp(2.33)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.83827314$$

$$-1.46 \rightarrow P(\text{Class 2}) = \frac{\exp(-1.46)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.01894129$$

$$0.56 \rightarrow P(\text{Class 3}) = \frac{\exp(0.56)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.14278557$$

Use Stable Softmax to Avoid Overflow in e^z

Since C is just an arbitrary constant, we can instead write:

$$S_j = \frac{e^{a_j + D}}{\sum_{k=1}^N e^{a_k + D}} \quad \text{eg.: 20, 40, 50.}$$

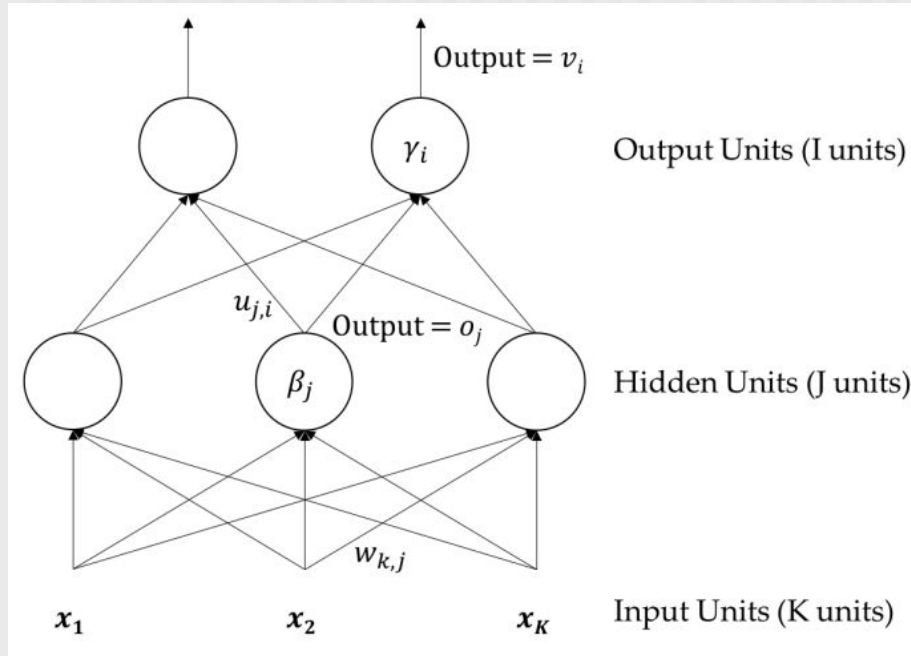
Where D is also an arbitrary constant. This formula is equivalent to the original S_j for any D , so we're free to choose a D that will make our computation better numerically. A good choice is the maximum between all inputs, negated:

$$D = -\max(a_1, a_2, \dots, a_N)$$

This will shift the inputs to a range close to zero, assuming the inputs themselves are not too far from each other. Crucially, it shifts them all to be negative (except the maximal a_j which turns into a zero). Negatives with large exponents "saturate" to zero rather than infinity, so we have a better chance of avoiding NaNs.

```
def stablesoftmax(x):
    """Compute the softmax of vector x in a numerically stable way."""
    shiftx = x - np.max(x)
    exps = np.exp(shiftx)
    return exps / np.sum(exps)
```


Minimizing the Cost Function



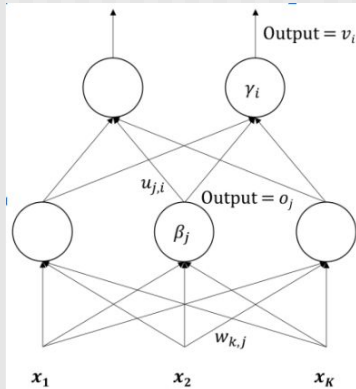
$$E(\theta) = \frac{1}{2} \sum_{i=1}^I (y_i - v_i)^2$$

$$o_j = \sigma \left(\sum_{k=1}^K x_k w_{k,j} + \beta_j \right)$$

$$v_i = \sigma \left(\sum_{j=1}^J o_j u_{j,i} + \gamma_i \right) = \sigma \left(\sum_{j=1}^J \sigma \left(\sum_{k=1}^K x_k w_{k,j} + \beta_j \right) u_{j,i} + \gamma_i \right)$$

$$o_j = \sigma \left(\sum_{k=1}^K x_k w_{k,j} + \beta_j \right)$$

I - no. output units
J - no. hidden units
K - no. input units



$$v_i = \sigma \left(\sum_{j=1}^J o_j u_{j,i} + \gamma_i \right) = \sigma \left(\sum_{j=1}^J \sigma \left(\sum_{k=1}^K x_k w_{k,j} + \beta_j \right) u_{j,i} + \gamma_i \right)$$

$$\frac{d}{dx} \sigma(x) = \frac{d}{dx} \left[\frac{1}{1 + e^{-x}} \right] = \frac{d}{dx} (1 + e^{-x})^{-1} = \sigma(x)(1 - \sigma(x))$$

$$E(\theta) = \frac{1}{2} \sum_{i=1}^I (y_i - v_i)^2$$

$$\begin{aligned} \frac{\delta E(\theta)}{\delta u_{j,i}} &= (y_i - v_i) \frac{\delta v_i}{\delta u_{j,i}} \\ &= (y_i - v_i) v_i (1 - v_i) o_j \end{aligned}$$

$$\begin{aligned} \frac{\delta E(\theta)}{\delta w_{k,j}} &= \sum_{i=1}^I (y_i - v_i) \frac{\delta v_i}{\delta w_{k,j}} \\ &= \sum_{i=1}^I (y_i - v_i) \frac{\delta v_i}{\delta o_j} \frac{\delta o_j}{\delta w_{k,j}} \\ &= \sum_{i=1}^I (y_i - v_i) (v_i (1 - v_i) u_{j,i}) (o_j (1 - o_j) x_k) \end{aligned}$$

$$o_j = \sigma \left(\sum_{k=1}^K x_k w_{k,j} + \beta_j \right)$$

$$v_i = \sigma \left(\sum_{j=1}^J o_j u_{j,i} + \gamma_i \right) = \sigma \left(\sum_{j=1}^J \sigma \left(\sum_{k=1}^K x_k w_{k,j} + \beta_j \right) u_{j,i} + \gamma_i \right)$$

Forward Propagation

(2) Hidden to Output

$$v_i = \sigma \left(\sum_{j=1}^J o_j u_{j,i} + \gamma_i \right)$$

I - no. output units

J - no. hidden units

K - no. input units

(1) Input to Hidden Layer

$$o_j = \sigma \left(\sum_{k=1}^K x_k w_{k,j} + \beta_j \right)$$

Backward Propagation

(3) Output to Hidden

$$\delta_i = (y_i - v_i) v_i (1 - v_i)$$

$$\Delta u_{j,i} = -\eta(t) \delta_i o_j$$

$$\Delta \gamma_i = -\eta(t) \delta_i$$

(4) Hidden to Input

$$\varphi_j = \sum_{i=1}^I \delta_i u_{j,i} o_j (1 - o_j)$$

$$\Delta w_{k,j} = -\eta(t) \varphi_j x_k$$

$$\Delta \beta_j = -\eta(t) \varphi_j$$

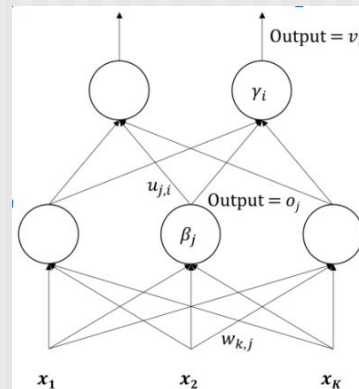
$$\frac{\delta E(\theta)}{\delta u_{j,i}} = (y_i - v_i) \frac{\delta v_i}{\delta u_{j,i}} = (y_i - v_i) v_i (1 - v_i) o_j$$

δ_i

$$\begin{aligned} \frac{\delta E(\theta)}{\delta w_{k,j}} &= \sum_{i=1}^I (y_i - v_i) \frac{\delta v_i}{\delta w_{k,j}} \\ &= \sum_{i=1}^I (y_i - v_i) \frac{\delta v_i}{\delta o_j} \frac{\delta o_j}{\delta w_{k,j}} \\ &= \sum_{i=1}^I (y_i - v_i) (v_i (1 - v_i) u_{j,i}) (o_j (1 - o_j) x_k) \end{aligned}$$

δ_i

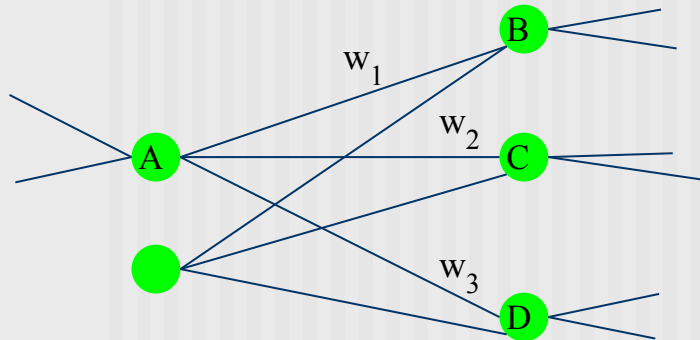
φ_j



Error Propagation (learning) Algorithm

If a hidden layer neuron A is connected to B, C, and D in the output layer, it is responsible for the errors observed in B, C, and D. Thus, the error in hidden layer node A is computable by summing up the errors in B, C, and D weighted by the connection weights between A-B, A-C, and A-D, respectively. We adjust weights w_1 , w_2 , w_3 in the hidden layer to reduce this error for A.

We then try to minimize this error by adjusting weights from output back to each hidden layer until the input layer.

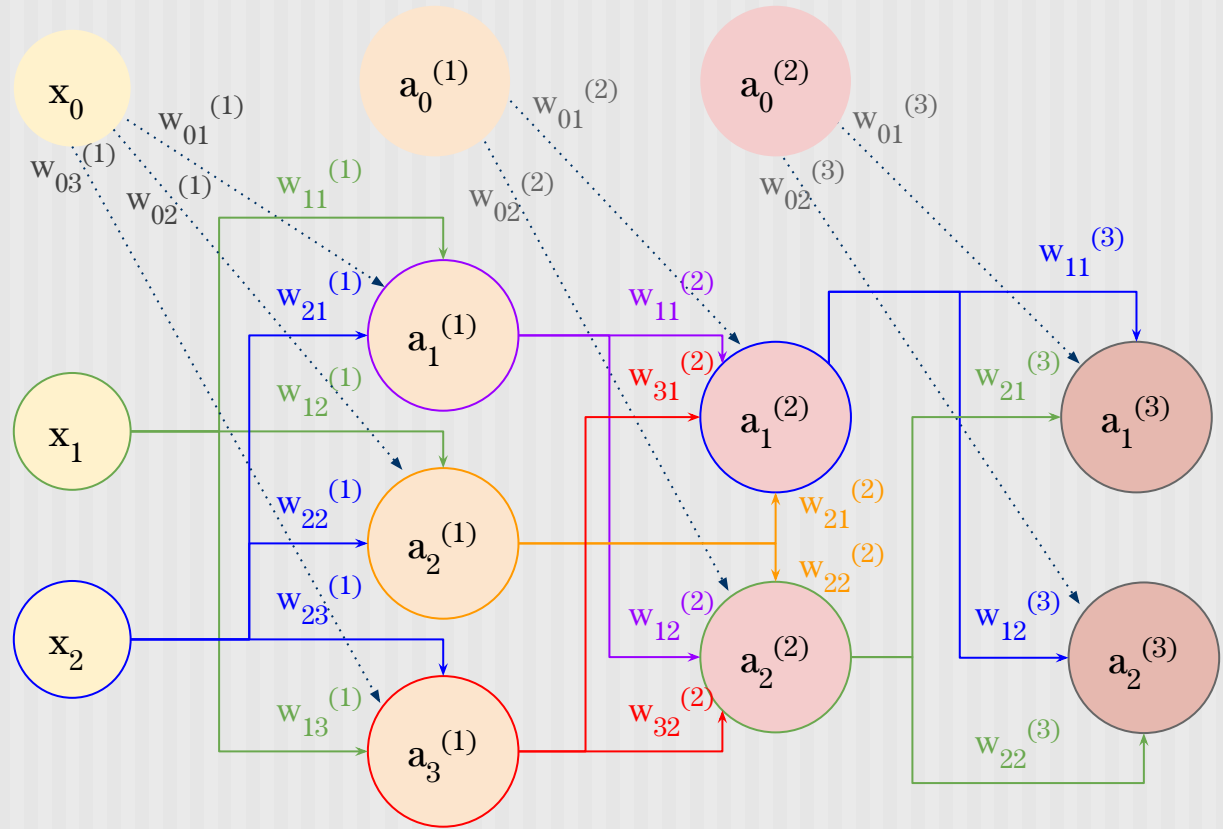


Descent Learning Step-by-step

$$a_{2 \times 1}^{(3)} = \sigma^{(3)} \left(W_{2 \times 2}^{(3)T} \cdot a_{2 \times 1}^{(2)} \right)$$

$$a_{2 \times 1}^{(2)} = \sigma^{(2)} \left(W_{3 \times 2}^{(2)T} \cdot a_{3 \times 1}^{(1)} \right)$$

$$a_{3 \times 1}^{(1)} = \sigma^{(1)} \left(W_{2 \times 3}^{(1)T} \cdot x_{2 \times 1} \right)$$



$$a^{(L)} = \sigma^{(L)} \left(z^{(L)} \right); \sigma^{(i)} - \text{layer } i \text{ activation function. } \sigma^{(L)} \text{ often softmax; } \sigma^{(i)}, i \neq L \text{ are often sigmoid.}$$

$$= \sigma^{(L)} \left(W^{(L)T} \sigma^{(L-1)} \left(W^{(L-1)T} \sigma^{(L-1)} \dots W^{(3)T} \sigma^{(3)} \left(\sigma^{(2)} \left(W^{(2)T} \sigma^{(1)} \left(W^{(1)T} x \right) \right) \dots \right) \right) \right)$$

$$= \sigma^{(4)} \left(W^{(4)T} \sigma^{(3)} \left(W^{(3)T} \sigma^{(2)} \left(W^{(2)T} \sigma^{(1)} \left(W^{(1)T} x \right) \right) \right) \right); \text{ case when } L = 4$$

Descent Learning Step-by-step

$$a_{3 \times 1}^{(1)} = \sigma^{(1)} \left(W_{3 \times 3}^{(1)T} \cdot x_{3 \times 1} \right); x_0 = 1$$

$$= \sigma^{(1)} \left(\begin{bmatrix} w_{01}^{(1)} & w_{11}^{(1)} & w_{21}^{(1)} \\ w_{02}^{(1)} & w_{12}^{(1)} & w_{22}^{(1)} \\ w_{03}^{(1)} & w_{13}^{(1)} & w_{23}^{(1)} \end{bmatrix} \begin{bmatrix} x_0 = 1 \\ x_1 \\ x_2 \end{bmatrix} \right)$$

$$= \begin{bmatrix} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) \\ \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) \\ \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \end{bmatrix} = \begin{bmatrix} \sigma^{(1)} z_1^{(1)} \\ \sigma^{(1)} z_2^{(1)} \\ \sigma^{(1)} z_3^{(1)} \end{bmatrix}$$

Descent Learning Step-by-step

$$a_{2 \times 1}^{(2)} = \sigma^{(2)} \left(W_{4 \times 2}^{(2)T} \cdot a_{4 \times 1}^{(1)} \right); a_0^{(1)} = 1$$

$$= \sigma^{(2)} \left(\begin{bmatrix} w_{01}^{(2)} & w_{11}^{(2)} & w_{21}^{(2)} & w_{31}^{(2)} \\ w_{02}^{(2)} & w_{12}^{(2)} & w_{22}^{(2)} & w_{32}^{(2)} \end{bmatrix} \begin{bmatrix} a_0^{(1)} = 1 \\ \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) \\ \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) \\ \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \end{bmatrix} \right)$$

$$\begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = \begin{bmatrix} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{21}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{31}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \\ \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{22}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{32}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \end{bmatrix}$$

$$\begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = \begin{bmatrix} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{21}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{31}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) \\ \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{22}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{32}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) \end{bmatrix} = \begin{bmatrix} \sigma^{(2)} \left(z_1^{(2)} \right) \\ \sigma^{(2)} \left(z_2^{(2)} \right) \end{bmatrix}$$

Descent Learning Step-by-step

$$\begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = \begin{bmatrix} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{21}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{31}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \\ \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{22}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{32}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \end{bmatrix}$$

$$a_{2 \times 1}^{(3)} = \sigma^{(3)} \left(W_{3 \times 2}^{(3)T} \cdot a_{3 \times 1}^{(2)} \right); a_0^{(2)} = 1$$

$$a_{2 \times 1}^{(3)} = \sigma^{(3)} \left(\begin{bmatrix} w_{01}^{(3)} & w_{11}^{(3)} & w_{21}^{(3)} \\ w_{02}^{(3)} & w_{12}^{(3)} & w_{22}^{(3)} \end{bmatrix} \begin{bmatrix} a_0^{(2)} = 1 \\ a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} \right) \quad \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = \begin{bmatrix} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{21}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{31}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) \\ \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{22}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{32}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) \end{bmatrix} = \begin{bmatrix} \sigma^{(2)} \left(z_1^{(2)} \right) \\ \sigma^{(2)} \left(z_2^{(2)} \right) \end{bmatrix}$$

$$a_1^{(3)} = \sigma^{(3)} \left[w_{01}^{(3)} + w_{11}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{21}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{31}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) + w_{21}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{22}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{32}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right]$$

$$a_2^{(3)} = \sigma^{(3)} \left[w_{02}^{(3)} + w_{12}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{21}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{31}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) + w_{22}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{22}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{32}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right]$$

Descent Learning Step-by-step

$$E = \sum_{i=1}^{80} \left[\left(y_{i1} - a_{i1}^{(3)}(x) \right)^2 + \left(y_{i2} - a_{i2}^{(3)}(x) \right)^2 \right]$$

$$\begin{aligned} a_1^{(3)} &= \sigma^{(3)} \left(z_1^{(3)} \right) \\ &= \sigma^{(3)} \left(w_{01}^{(3)} + w_{11}^{(3)} \sigma^{(2)} \left(z_1^{(2)} \right) + w_{21}^{(3)} \sigma^{(2)} \left(z_2^{(2)} \right) \right) \\ &= \sigma^{(3)} \left(w_{01}^{(3)} + w_{11}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{21}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{31}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) + w_{21}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{22}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{32}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) \right) \end{aligned}$$

$$\begin{aligned} a_1^{(3)} &= \sigma^{(3)} \left[w_{01}^{(3)} + w_{11}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{21}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{31}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right. \\ &\quad \left. + w_{21}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{22}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{32}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right] \end{aligned}$$

$$\begin{aligned} a_2^{(3)} &= \sigma^{(3)} \left(z_2^{(3)} \right) \\ &= \sigma^{(3)} \left(w_{02}^{(3)} + w_{12}^{(3)} \sigma^{(2)} \left(z_1^{(2)} \right) + w_{22}^{(3)} \sigma^{(2)} \left(z_2^{(2)} \right) \right) \\ &= \sigma^{(3)} \left(w_{02}^{(3)} + w_{12}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{21}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{31}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) + w_{22}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{22}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{32}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) \right) \end{aligned}$$

$$\begin{aligned} a_2^{(3)} &= \sigma^{(3)} \left[w_{02}^{(3)} + w_{12}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{21}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{31}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right. \\ &\quad \left. + w_{22}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{22}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{32}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right] \end{aligned}$$

$$\begin{aligned}
a_1^{(3)} &= \sigma^{(3)} \left[w_{01}^{(3)} + w_{11}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{21}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{31}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right. \\
&\quad \left. + w_{21}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{22}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{32}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right] \\
a_2^{(3)} &= \sigma^{(3)} \left[w_{02}^{(3)} + w_{12}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{21}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{31}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right. \\
&\quad \left. + w_{22}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{22}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{32}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right]
\end{aligned}$$

$z_2^{(1)}$

M = 2, the number of dimensions in the output.

N = 80, the number of training data.

K = 8, the batch size.

$$\begin{aligned}
E &= \frac{1}{2N} \sum_{i=1}^N \sum_{j=1}^M \left(y_{ij} - a_{ij}^{(L)}(x) \right)^2 = \frac{1}{2 \cdot 80} \sum_{i=1}^{80} \left[\left(y_{i1} - a_{i1}^{(3)}(x) \right)^2 + \left(y_{i2} - a_{i2}^{(3)}(x) \right)^2 \right] \\
&= \frac{1}{2 \cdot 80} \sum_{i=1}^{80} \left\| y_i - a_i^{(3)}(x) \right\|^2 = \frac{1}{80} \sum_{i=1}^{80} E_i
\end{aligned}$$

$$\begin{aligned}
E_i &= E_i \left(w_{01}^{(1)}, w_{02}^{(1)}, w_{03}^{(1)}, w_{11}^{(1)}, w_{12}^{(1)}, w_{13}^{(1)}, w_{21}^{(1)}, w_{22}^{(1)}, w_{23}^{(1)}, w_{01}^{(2)}, w_{02}^{(2)}, w_{11}^{(2)}, w_{12}^{(2)}, w_{21}^{(2)}, w_{22}^{(2)}, w_{31}^{(2)}, w_{32}^{(2)}, w_{01}^{(3)}, w_{02}^{(3)}, w_{11}^{(3)}, w_{12}^{(3)}, w_{21}^{(3)}, w_{22}^{(3)} \right) \\
&= \frac{1}{2} \left\| y_i - a_i^{(L)}(x) \right\|^2 = \frac{1}{2} \left\| y_i - a_i^{(3)}(x) \right\|^2 = \frac{1}{2} \left[\left(y_{i1} - a_{i1}^{(3)}(x) \right)^2 + \left(y_{i2} - a_{i2}^{(3)}(x) \right)^2 \right]
\end{aligned}$$

Descent Learning Step-by-step

$$\begin{aligned}
 a_1^{(3)} &= \sigma^{(3)} \left(z_1^{(3)} \right) \\
 &= \sigma^{(3)} \left(w_{01}^{(3)} + w_{11}^{(3)} \sigma^{(2)} \left(z_1^{(2)} \right) + w_{21}^{(3)} \sigma^{(2)} \left(z_2^{(2)} \right) \right) \\
 &= \sigma^{(3)} \left(w_{01}^{(3)} + w_{11}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{21}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{31}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) + w_{21}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{22}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{32}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) \right)
 \end{aligned}$$

$$\begin{aligned}
 a_2^{(3)} &= \sigma^{(3)} \left(z_2^{(3)} \right) \\
 &= \sigma^{(3)} \left(w_{02}^{(3)} + w_{12}^{(3)} \sigma^{(2)} \left(z_1^{(2)} \right) + w_{22}^{(3)} \sigma^{(2)} \left(z_2^{(2)} \right) \right) \\
 &= \sigma^{(3)} \left(w_{02}^{(3)} + w_{12}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{21}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{31}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) + w_{22}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(z_1^{(1)} \right) + w_{22}^{(2)} \sigma^{(1)} \left(z_2^{(1)} \right) + w_{32}^{(2)} \sigma^{(1)} \left(z_3^{(1)} \right) \right) \right)
 \end{aligned}$$

$$E = \sum_{i=1}^{80} \left[\left(y_{i1} - a_{i1}^{(3)}(x) \right)^2 + \left(y_{i2} - a_{i2}^{(3)}(x) \right)^2 \right]$$

$$\begin{aligned}
 \frac{dE_i}{dw_{mn}^{(l)}} &= -2 \left(y_1 - a_1^{(3)}(x) \right) \cdot \frac{da_1^{(3)}(x)}{dw_{mn}^{(l)}} - 2 \left(y_2 - a_2^{(3)}(x) \right) \cdot \frac{da_2^{(3)}(x)}{dw_{mn}^{(l)}} \\
 &= -2 \left(y_1 - a_1^{(3)}(x) \right) \cdot \frac{d\sigma^{(3)} \left(z_1^{(3)} \right)}{dz_1^{(3)}} \cdot \frac{dz_1^{(3)}}{dw_{mn}^{(l)}} - 2 \left(y_2 - a_2^{(3)}(x) \right) \cdot \frac{d\sigma^{(3)} \left(z_2^{(3)} \right)}{dz_2^{(3)}} \cdot \frac{dz_2^{(3)}}{dw_{mn}^{(l)}}
 \end{aligned}$$

$$\begin{aligned}
 \frac{dE_i}{dw_{01}^{(3)}} &= -2 \left(y_1 - a_1^{(3)}(x) \right) \cdot \frac{d\sigma^{(3)} \left(z_1^{(3)} \right)}{dz_1^{(3)}} \cdot \frac{dz_1^{(3)}}{dw_{01}^{(3)}} \\
 &= -2 \left(y_1 - a_1^{(3)}(x) \right) \cdot \sigma^{(3)} \left(z_1^{(3)} \right) \cdot \left(1 - \sigma^{(3)} \left(z_1^{(3)} \right) \right) \cdot 1 \\
 &= -2 \left(y_1 - a_1^{(3)}(x) \right) \cdot a_1^{(3)} \cdot \left(1 - a_1^{(3)} \right)
 \end{aligned}$$

$$\begin{aligned}
 \frac{\delta E(\theta)}{\delta u_{j,i}} &= (y_i - v_i) \frac{\delta v_i}{\delta u_{j,i}} \\
 &= (y_i - v_i) v_i (1 - v_i) o_j
 \end{aligned}$$

$$\begin{aligned}
 \frac{dE_i}{dw_{21}^{(3)}} &= -2 \left(y_1 - a_1^{(3)}(x) \right) \cdot \frac{d\sigma^{(3)} \left(z_1^{(3)} \right)}{dz_1^{(3)}} \cdot \frac{dz_1^{(3)}}{dw_{21}^{(3)}} \\
 &= -2 \left(y_1 - a_1^{(3)}(x) \right) \cdot \sigma^{(3)} \left(z_1^{(3)} \right) \cdot \left(1 - \sigma^{(3)} \left(z_1^{(3)} \right) \right) \cdot \sigma^{(2)} \left(z_2^{(2)} \right) \\
 &= -2 \left(y_1 - a_1^{(3)}(x) \right) \cdot a_1^{(3)} \cdot \left(1 - a_1^{(3)} \right) \cdot a_2^{(2)}
 \end{aligned}$$

Descent Learning Step-by-step

$$\begin{aligned}
 a_1^{(3)} &= \sigma^{(3)}\left(z_1^{(3)}\right) \\
 &= \sigma^{(3)}\left(w_{01}^{(3)} + w_{11}^{(3)}\sigma^{(2)}\left(z_1^{(2)}\right) + w_{21}^{(3)}\sigma^{(2)}\left(z_2^{(2)}\right)\right) \quad \mathbf{z}_1^{(2)} \\
 &= \sigma^{(3)}\left(w_{01}^{(3)} + w_{11}^{(3)}\sigma^{(2)}\left(w_{01}^{(2)} + w_{11}^{(2)}\sigma^{(1)}\left(z_1^{(1)}\right) + w_{21}^{(2)}\sigma^{(1)}\left(z_2^{(1)}\right) + w_{31}^{(2)}\sigma^{(1)}\left(z_3^{(1)}\right)\right) + w_{21}^{(3)}\sigma^{(2)}\left(w_{02}^{(2)} + w_{12}^{(2)}\sigma^{(1)}\left(z_1^{(1)}\right) + w_{22}^{(2)}\sigma^{(1)}\left(z_2^{(1)}\right) + w_{32}^{(2)}\sigma^{(1)}\left(z_3^{(1)}\right)\right)\right) \quad \mathbf{z}_2^{(2)} \\
 a_2^{(3)} &= \sigma^{(3)}\left(z_2^{(3)}\right) \\
 &= \sigma^{(3)}\left(w_{02}^{(3)} + w_{12}^{(3)}\sigma^{(2)}\left(z_1^{(2)}\right) + w_{22}^{(3)}\sigma^{(2)}\left(z_2^{(2)}\right)\right) \quad \mathbf{z}_1^{(2)} \\
 &= \sigma^{(3)}\left(w_{02}^{(3)} + w_{12}^{(3)}\sigma^{(2)}\left(w_{01}^{(2)} + w_{11}^{(2)}\sigma^{(1)}\left(z_1^{(1)}\right) + w_{21}^{(2)}\sigma^{(1)}\left(z_2^{(1)}\right) + w_{31}^{(2)}\sigma^{(1)}\left(z_3^{(1)}\right)\right) + w_{22}^{(3)}\sigma^{(2)}\left(w_{02}^{(2)} + w_{12}^{(2)}\sigma^{(1)}\left(z_1^{(1)}\right) + w_{22}^{(2)}\sigma^{(1)}\left(z_2^{(1)}\right) + w_{32}^{(2)}\sigma^{(1)}\left(z_3^{(1)}\right)\right)\right) \quad \mathbf{z}_2^{(2)}
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial E_i}{\partial w_{31}^{(2)}} &= -2\left(y_1 - a_1^{(3)}(x)\right) \cdot \frac{\partial \sigma^{(3)}\left(z_1^{(3)}\right)}{\partial z_1^{(3)}} \cdot \frac{\partial z_1^{(3)}}{\partial w_{31}^{(2)}} - 2\left(y_1 - a_2^{(3)}(x)\right) \cdot \frac{\partial \sigma^{(3)}\left(z_2^{(3)}\right)}{\partial z_2^{(3)}} \cdot \frac{\partial z_2^{(3)}}{\partial w_{31}^{(2)}} \\
 &= -2\left(y_1 - a_1^{(3)}(x)\right) \cdot \frac{\partial \sigma^{(3)}\left(z_1^{(3)}\right)}{\partial z_1^{(3)}} \cdot w_{11}^{(3)} \cdot \frac{\partial \sigma^{(2)}\left(z_1^{(2)}\right)}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{31}^{(2)}} - 2\left(y_1 - a_2^{(3)}(x)\right) \cdot \frac{\partial \sigma^{(3)}\left(z_2^{(3)}\right)}{\partial z_2^{(3)}} \cdot w_{12}^{(3)} \cdot \frac{\partial \sigma^{(2)}\left(z_1^{(2)}\right)}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{31}^{(2)}} \\
 &= -2\left(y_1 - a_1^{(3)}(x)\right) \cdot \sigma^{(3)}\left(z_1^{(3)}\right) \cdot \left(1 - \sigma^{(3)}\left(z_1^{(3)}\right)\right) \cdot w_{11}^{(3)} \cdot \sigma^{(2)}\left(z_1^{(2)}\right) \cdot \left(1 - \sigma^{(2)}\left(z_1^{(2)}\right)\right) \cdot \sigma^{(1)}\left(z_3^{(1)}\right) \\
 &\quad - 2\left(y_1 - a_2^{(3)}(x)\right) \cdot \sigma^{(3)}\left(z_2^{(3)}\right) \cdot \left(1 - \sigma^{(3)}\left(z_2^{(3)}\right)\right) \cdot w_{12}^{(3)} \cdot \sigma^{(2)}\left(z_1^{(2)}\right) \cdot \left(1 - \sigma^{(2)}\left(z_1^{(2)}\right)\right) \cdot \sigma^{(1)}\left(z_3^{(1)}\right)
 \end{aligned}$$

Descent Learning Step-by-step

$$\begin{aligned}
 a_1^{(3)} &= \sigma^{(3)}(z_1^{(3)}) \\
 &= \sigma^{(3)}\left(w_{01}^{(3)} + w_{11}^{(3)}\sigma^{(2)}\left(z_1^{(2)}\right) + w_{21}^{(3)}\sigma^{(2)}\left(z_2^{(2)}\right)\right) \quad \mathbf{z}_1^{(2)} \\
 &= \sigma^{(3)}\left(w_{01}^{(3)} + w_{11}^{(3)}\sigma^{(2)}\left(w_{01}^{(2)} + w_{11}^{(2)}\sigma^{(1)}\left(z_1^{(1)}\right) + w_{21}^{(2)}\sigma^{(1)}\left(z_2^{(1)}\right) + w_{31}^{(2)}\sigma^{(1)}\left(z_3^{(1)}\right)\right) + w_{21}^{(3)}\sigma^{(2)}\left(w_{02}^{(2)} + w_{12}^{(2)}\sigma^{(1)}\left(z_1^{(1)}\right) + w_{22}^{(2)}\sigma^{(1)}\left(z_2^{(1)}\right) + w_{32}^{(2)}\sigma^{(1)}\left(z_3^{(1)}\right)\right)\right) \quad \mathbf{z}_2^{(2)} \\
 a_2^{(3)} &= \sigma^{(3)}(z_2^{(3)}) \\
 &= \sigma^{(3)}\left(w_{02}^{(3)} + w_{12}^{(3)}\sigma^{(2)}\left(z_1^{(2)}\right) + w_{22}^{(3)}\sigma^{(2)}\left(z_2^{(2)}\right)\right) \quad \mathbf{z}_1^{(2)} \\
 &= \sigma^{(3)}\left(w_{02}^{(3)} + w_{12}^{(3)}\sigma^{(2)}\left(w_{01}^{(2)} + w_{11}^{(2)}\sigma^{(1)}\left(z_1^{(1)}\right) + w_{21}^{(2)}\sigma^{(1)}\left(z_2^{(1)}\right) + w_{31}^{(2)}\sigma^{(1)}\left(z_3^{(1)}\right)\right) + w_{22}^{(3)}\sigma^{(2)}\left(w_{02}^{(2)} + w_{12}^{(2)}\sigma^{(1)}\left(z_1^{(1)}\right) + w_{22}^{(2)}\sigma^{(1)}\left(z_2^{(1)}\right) + w_{32}^{(2)}\sigma^{(1)}\left(z_3^{(1)}\right)\right)\right) \quad \mathbf{z}_2^{(2)}
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial E_i}{\partial w_{01}^{(2)}} &= -2\left(y_1 - a_1^{(3)}(x)\right) \cdot \frac{\partial \sigma^{(3)}\left(z_1^{(3)}\right)}{\partial z_1^{(3)}} \cdot w_{11}^{(3)} \cdot \frac{\partial \sigma^{(2)}\left(z_1^{(2)}\right)}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{01}^{(2)}} - 2\left(y_1 - a_2^{(3)}(x)\right) \cdot \frac{\partial \sigma^{(3)}\left(z_2^{(3)}\right)}{\partial z_2^{(3)}} \cdot w_{12}^{(3)} \cdot \frac{\partial \sigma^{(2)}\left(z_1^{(2)}\right)}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{01}^{(2)}} \\
 &= -2\left(y_1 - a_1^{(3)}(x)\right) \cdot \sigma^{(3)}\left(z_1^{(3)}\right) \cdot \left(1 - \sigma^{(3)}\left(z_1^{(3)}\right)\right) \cdot w_{11}^{(3)} \cdot \sigma^{(2)}\left(z_1^{(2)}\right) \cdot \left(1 - \sigma^{(2)}\left(z_1^{(2)}\right)\right) \cdot 1 \\
 &\quad - 2\left(y_1 - a_2^{(3)}(x)\right) \cdot \sigma^{(3)}\left(z_2^{(3)}\right) \cdot \left(1 - \sigma^{(3)}\left(z_2^{(3)}\right)\right) \cdot w_{12}^{(3)} \cdot \sigma^{(2)}\left(z_1^{(2)}\right) \cdot \left(1 - \sigma^{(2)}\left(z_1^{(2)}\right)\right) \cdot 1
 \end{aligned}$$

$$z_1^{(2)} = w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)}(z_1^{(1)}) + w_{21}^{(2)} \sigma^{(1)}(z_2^{(1)}) + w_{31}^{(2)} \sigma^{(1)}(z_3^{(1)}); z_2^{(2)} = w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)}(z_1^{(1)}) + w_{22}^{(2)} \sigma^{(1)}(z_2^{(1)}) + w_{32}^{(2)} \sigma^{(1)}(z_3^{(1)})$$

$$z_2^{(1)} = w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2$$

$z_1^{(3)}$

$z_2^{(1)}$

$z_1^{(2)}$

$$a_1^{(3)} = \sigma^{(3)} \left[w_{01}^{(3)} + w_{11}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{21}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{31}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right. \\ \left. + w_{21}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{22}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{32}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right]$$

$$a_2^{(3)} = \sigma^{(3)} \left[w_{02}^{(3)} + w_{12}^{(3)} \sigma^{(2)} \left(w_{01}^{(2)} + w_{11}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{21}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{31}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right. \\ \left. + w_{22}^{(3)} \sigma^{(2)} \left(w_{02}^{(2)} + w_{12}^{(2)} \sigma^{(1)} \left(w_{01}^{(1)} + w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 \right) + w_{22}^{(2)} \sigma^{(1)} \left(w_{02}^{(1)} + w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 \right) + w_{32}^{(2)} \sigma^{(1)} \left(w_{03}^{(1)} + w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 \right) \right) \right]$$

$z_2^{(2)}$

$$\frac{\partial E_i}{\partial w_{12}^{(1)}} = -2 \left(y_1 - a_1^{(3)}(x) \right) \cdot \frac{\partial \sigma^{(3)}(z_1^{(3)})}{\partial z_1^{(3)}} \cdot \frac{\partial z_1^{(3)}}{\partial w_{12}^{(1)}} - 2 \left(y_1 - a_2^{(3)}(x) \right) \cdot \frac{\partial \sigma^{(3)}(z_2^{(3)})}{\partial z_2^{(3)}} \cdot \frac{\partial z_2^{(3)}}{\partial w_{12}^{(1)}}$$

$$\frac{\partial z_1^{(2)}}{\partial w_{12}^{(1)}} = w_{21}^{(2)} \cdot \frac{\partial \sigma^{(1)}}{\partial w_{12}^{(1)}} \cdot \frac{\partial z_2^{(1)}}{\partial w_{12}^{(1)}} = w_{21}^{(2)} \cdot \frac{\partial \sigma^{(1)}}{\partial w_{12}^{(1)}} \cdot x_1$$

$$\frac{\partial z_2^{(2)}}{\partial w_{12}^{(1)}} = w_{22}^{(2)} \cdot \frac{\partial \sigma^{(1)}}{\partial w_{12}^{(1)}} \cdot \frac{\partial z_2^{(1)}}{\partial w_{12}^{(1)}} = w_{22}^{(2)} \cdot \frac{\partial \sigma^{(1)}}{\partial w_{12}^{(1)}} \cdot x_1$$

$$= -2 \left(y_1 - a_1^{(3)}(x) \right) \cdot \frac{\partial \sigma^{(3)}(z_1^{(3)})}{\partial z_1^{(3)}} \cdot \left(w_{11}^{(3)} \cdot \frac{\partial \sigma^{(2)}(z_1^{(2)})}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{12}^{(1)}} + w_{21}^{(3)} \cdot \frac{\partial \sigma^{(2)}(z_2^{(2)})}{\partial z_2^{(2)}} \cdot \frac{\partial z_2^{(2)}}{\partial w_{12}^{(1)}} \right) \\ - 2 \left(y_1 - a_2^{(3)}(x) \right) \cdot \frac{\partial \sigma^{(3)}(z_2^{(3)})}{\partial z_2^{(3)}} \cdot \left(w_{12}^{(3)} \cdot \frac{\partial \sigma^{(2)}(z_1^{(2)})}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{12}^{(1)}} + w_{22}^{(3)} \cdot \frac{\partial \sigma^{(2)}(z_2^{(2)})}{\partial z_2^{(2)}} \cdot \frac{\partial z_2^{(2)}}{\partial w_{12}^{(1)}} \right)$$

$$\frac{\partial E_i}{\partial w_{12}^{(1)}} = -2\left(y_1 - a_1^{(3)}(x)\right) \cdot \frac{\partial \sigma^{(3)}\left(z_1^{(3)}\right)}{\partial z_1^{(3)}} \cdot \frac{\partial z_1^{(3)}}{\partial w_{12}^{(1)}} - 2\left(y_1 - a_2^{(3)}(x)\right) \cdot \frac{\partial \sigma^{(3)}\left(z_2^{(3)}\right)}{\partial z_2^{(3)}} \cdot \frac{\partial z_2^{(3)}}{\partial w_{12}^{(1)}}$$

$$= -2\left(y_1 - a_1^{(3)}(x)\right) \cdot \frac{\partial \sigma^{(3)}\left(z_1^{(3)}\right)}{\partial z_1^{(3)}} \cdot \left(w_{11}^{(3)} \cdot \frac{\partial \sigma^{(2)}\left(z_1^{(2)}\right)}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{12}^{(1)}} + w_{21}^{(3)} \cdot \frac{\partial \sigma^{(2)}\left(z_2^{(2)}\right)}{\partial z_2^{(2)}} \cdot \frac{\partial z_2^{(2)}}{\partial w_{12}^{(1)}} \right) \\ - 2\left(y_1 - a_2^{(3)}(x)\right) \cdot \frac{\partial \sigma^{(3)}\left(z_2^{(3)}\right)}{\partial z_2^{(3)}} \cdot \left(w_{12}^{(3)} \cdot \frac{\partial \sigma^{(2)}\left(z_1^{(2)}\right)}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{12}^{(1)}} + w_{22}^{(3)} \cdot \frac{\partial \sigma^{(2)}\left(z_2^{(2)}\right)}{\partial z_2^{(2)}} \cdot \frac{\partial z_2^{(2)}}{\partial w_{12}^{(1)}} \right)$$

$$\frac{\partial z_1^{(2)}}{\partial w_{12}^{(1)}} = w_{21}^{(2)} \cdot \frac{\partial \sigma^{(1)}}{\partial w_{12}^{(1)}} \cdot \frac{\partial z_2^{(1)}}{\partial w_{12}^{(1)}} = w_{21}^{(2)} \cdot \frac{\partial \sigma^{(1)}}{\partial w_{12}^{(1)}} \cdot x_1$$

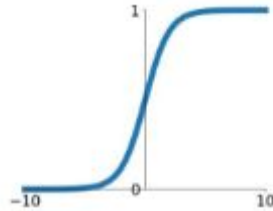
$$\frac{\partial z_2^{(2)}}{\partial w_{12}^{(1)}} = w_{22}^{(2)} \cdot \frac{\partial \sigma^{(1)}}{\partial w_{12}^{(1)}} \cdot \frac{\partial z_2^{(1)}}{\partial w_{12}^{(1)}} = w_{22}^{(2)} \cdot \frac{\partial \sigma^{(1)}}{\partial w_{12}^{(1)}} \cdot x_1$$

$$\begin{aligned} \frac{\delta E(\theta)}{\delta w_{k,j}} &= \sum_{i=1}^I (y_i - v_i) \frac{\delta v_i}{\delta w_{k,j}} \\ &= \sum_{i=1}^I (y_i - v_i) \frac{\delta v_i}{\delta o_j} \frac{\delta o_j}{\delta w_{k,j}} \\ &= \sum_{i=1}^I (y_i - v_i) (v_i (1 - v_i) u_{j,i}) (o_j (1 - o_j) x_k) \end{aligned}$$

Activation Functions

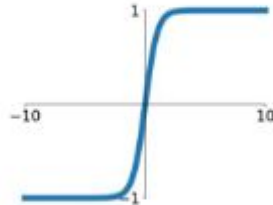
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



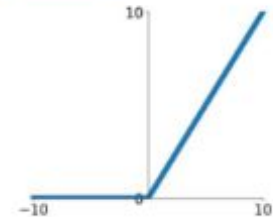
tanh

$$\tanh(x)$$



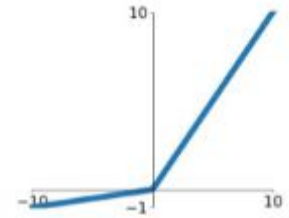
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

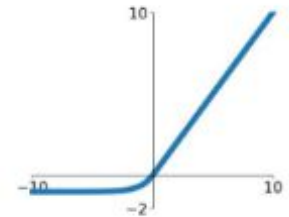


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

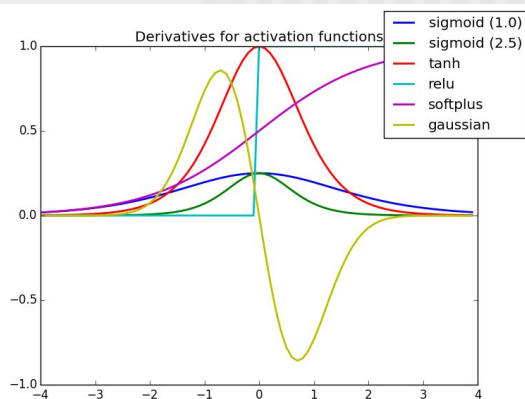
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$





Popular Activation Functions



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
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		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

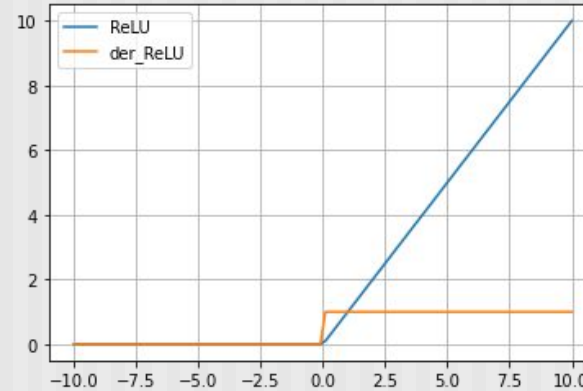
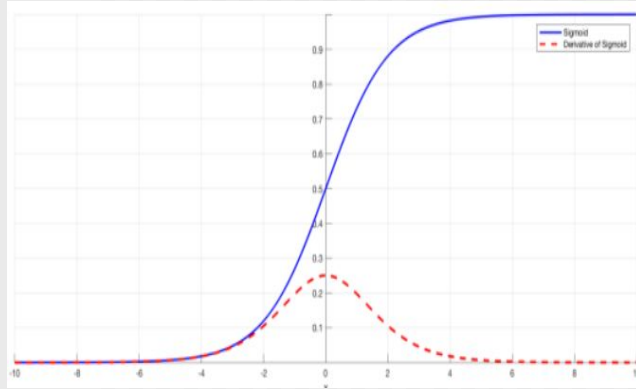
See <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

Activation Functions

<https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/>

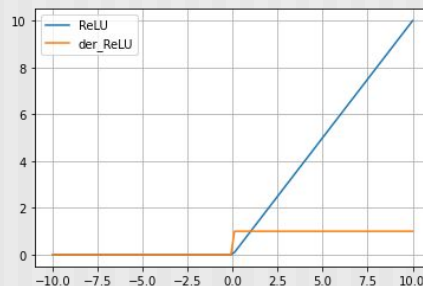
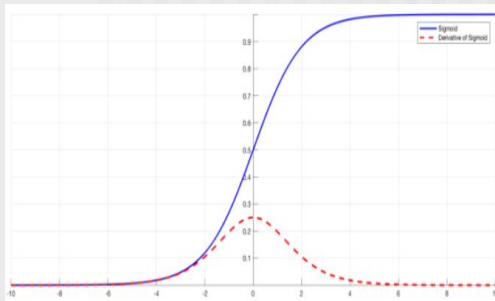
- Sigmoid functions and their combinations generally work better in the case of classifiers
- Sigmoids and tanh functions are sometimes avoided due to the vanishing gradient problem
- **ReLU function is a general activation function and is commonly used today**
- If we encounter a case of dead neurons in our networks the leaky ReLU function is the best choice
- **Always keep in mind that ReLU function should only be used in the hidden layers**
- As a rule of thumb, you can begin with using ReLU function and then move over to other activation functions in case ReLU does not provide with optimum results.

The Vanishing Gradient Problem



- When the inputs of the sigmoid function becomes larger or smaller (when $|x|$ becomes bigger), the derivative is close to 0.
- In backprop via the chain rule, the derivatives of each layer are multiplied down the network (from the final layer to the initial) to compute the derivatives of the initial layers.
- The gradient decreases exponentially as we propagate error gradients down to the initial layers.
- A small gradient means that the weights and biases of the earlier layers will not be updated effectively with each training session. **The effect in large networks is not sent down** to them.
- Solution 1: use ReLU or Rectified Linear Unit activation function: $y = \max(0, x)$. ReLU has a constant gradient.
- Solution 2: Batch normalize inputs so $|x|$ falls within -4 to 4, so that the sigmoidal derivative is not near 0.

Avoiding the Vanishing Gradient Problem



1. **Initialization.** Initialize the weights randomly, but with a mean of zero and a small variance.
2. **Non-saturating activation functions.** Use activation functions that do not become very flat for large or small inputs (big $|x|$) to help prevent the gradients from vanishing. Examples: ReLU with constant gradient and its variants.
3. **Batch normalization.** Normalizing the inputs to each layer of the network, since it reduces the dependence of the gradients on the scale of the input values.
4. **Gradient clipping.** Clipping the gradients to a certain (min, max) range prevent from becoming too small or too large.
5. **Residual connections.** Adding residual connections that skip over one or more layers of the network allows more direct flow of gradients to the lower layers.

Batch Normalization

1. Given a mini-batch of inputs, calculate the mean and variance of the input activations over the batch.
2. Normalize the input activations (x_1, \dots, x_N) using the batch mean μ and variance σ to get (x'_1, \dots, x'_N) .
3. Scale and shift the normalized activations using learnable parameters (added unknowns) gamma γ and beta β : $(x''_1, \dots, x''_N) = (\gamma_1 x'_1 + \beta_1, \dots, \gamma_N x'_N + \beta_N)$.
4. Pass the scaled and shifted activations through a nonlinear activation function.
5. During training, update the N gamma and N beta parameters using backpropagation and gradient descent to minimize the loss function.
6. During inference, first normalize the input activations by the N mean μ and N variance σ of the entire training set. Also transform input using the learned N gamma and N beta parameters.

Regularization on the Weights

MLP trains using **Stochastic Gradient Descent**, **Adam**, or **L-BFGS**. Stochastic Gradient Descent (SGD) updates parameters using the gradient of the loss function with respect to a parameter that needs adaptation, i.e.

$$w \leftarrow w - \eta \left(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial Loss}{\partial w} \right) \quad \boxed{Loss(\hat{y}, y, W) = \frac{1}{2} \|\hat{y} - y\|_2^2 + \frac{\alpha}{2} \|W\|_2^2}$$

where η is the learning rate which controls the step-size in the parameter space search. *Loss* is the loss function used for the network.

