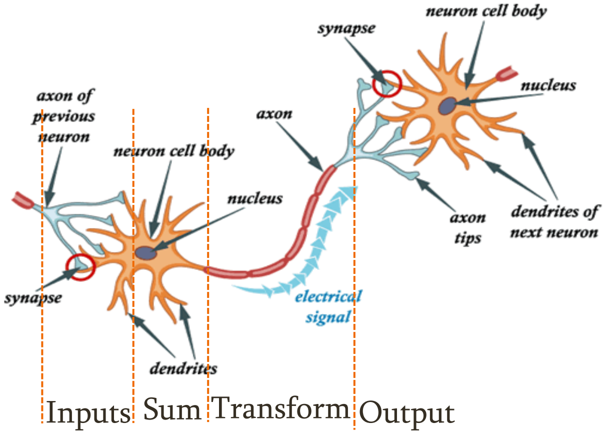
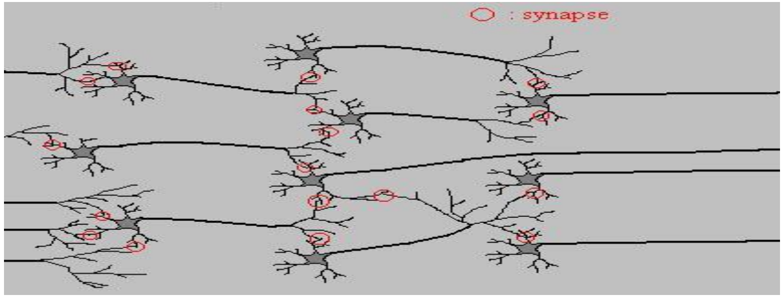
Artificial Neural Network

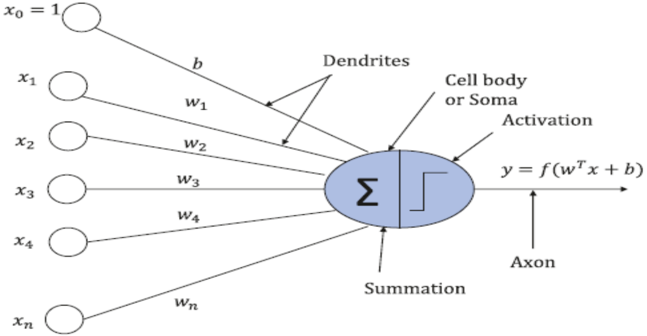
Abdessalam Bouchekif
abdessalam.bouchekif@epita.fr

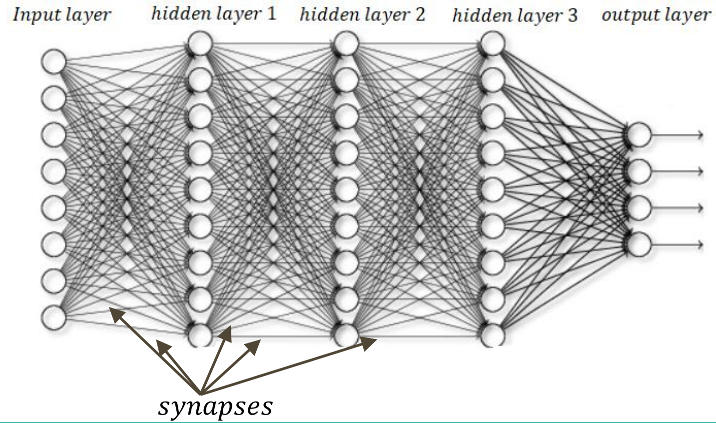
Artificial Neural Network

- An artificial neural network (or neural network for short) is a predictive model motivated by the way the brain operates.





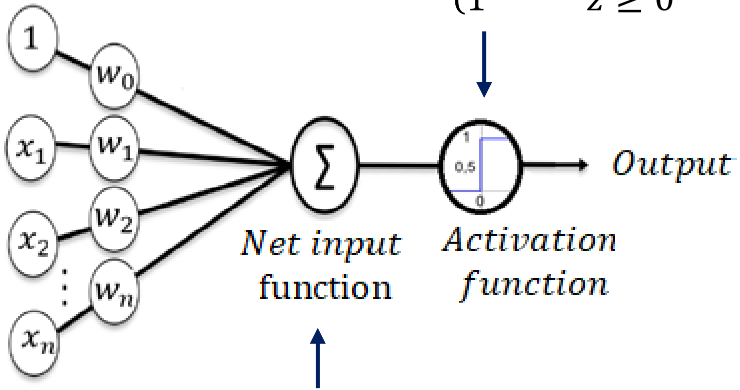




Perceptron

- Perceptron is a linear model used for binary classification

$$f(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$$



$$z = w_0 + w_1 x_1 + \cdots + w_n x_n = W^T . X$$

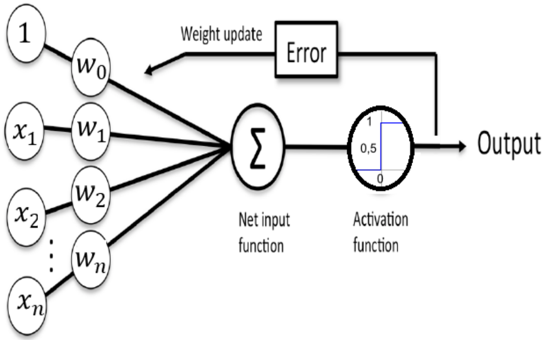
Perceptron Training

1. Initialize weights with random values.

2. Do

$$\boldsymbol{w_i = w_i + \eta(y_t - \hat{y}_i)x_i}$$

3. Repeat until no errors are made (or other convergence heuristics)



$\boldsymbol{w_i}$ is the connection weight between the i^{th} input neuron and the output neuron.

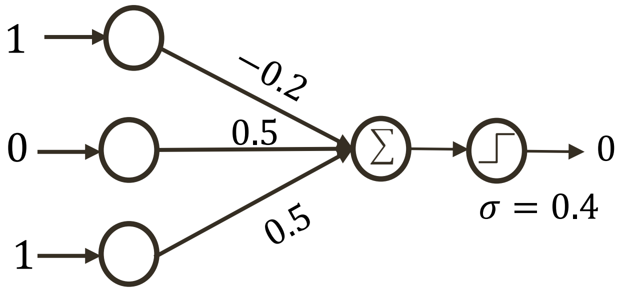
$\boldsymbol{x_i}$ is the i^{th} input value of the current training instance.

$\boldsymbol{\hat{y}}$ is the output of the j^{th} output neuron for the current training instance

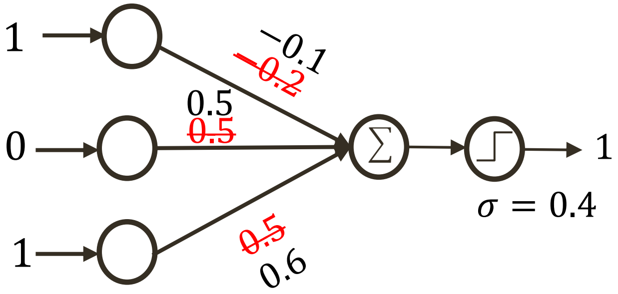
$\boldsymbol{y}$ is the target output of the j^{th} output neuron for the current training instance

$\boldsymbol{\eta}$ is the learning rate,

Example of processing one sample

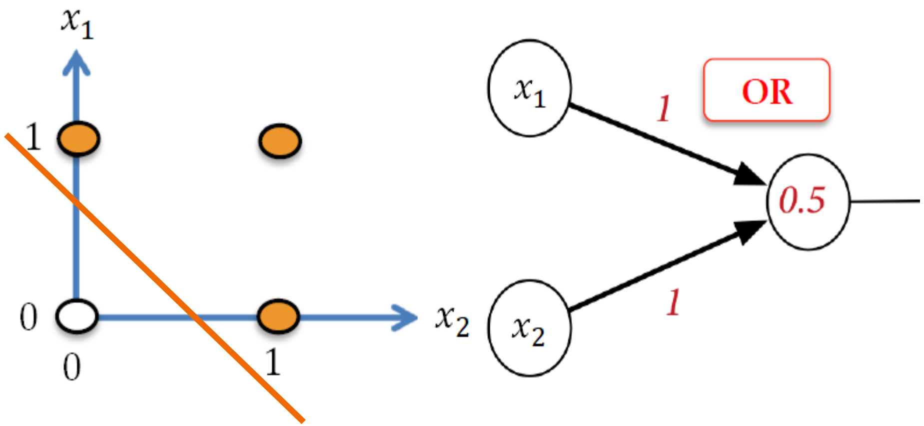


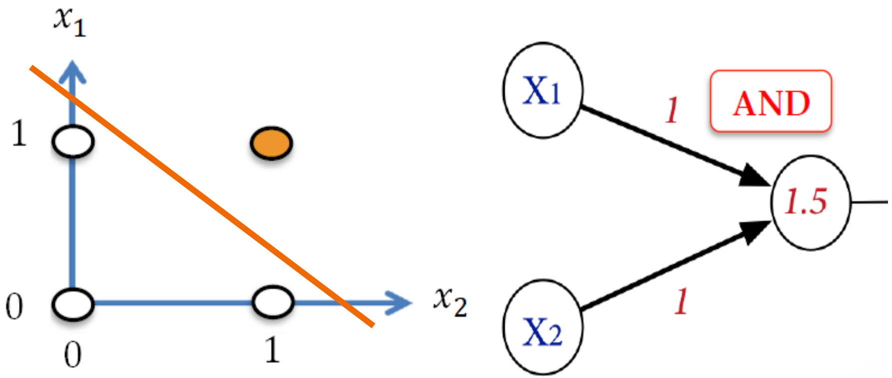
$\eta = 0.1$
$y_i - \hat{y}_i = 1$
$\eta(y_i - \hat{y}_i)x_{i1} = 0.1$
$\eta(y_i - \hat{y}_i)x_{i2} = 0.0$
$\eta(y_i - \hat{y}_i)x_{i3} = 0.1$



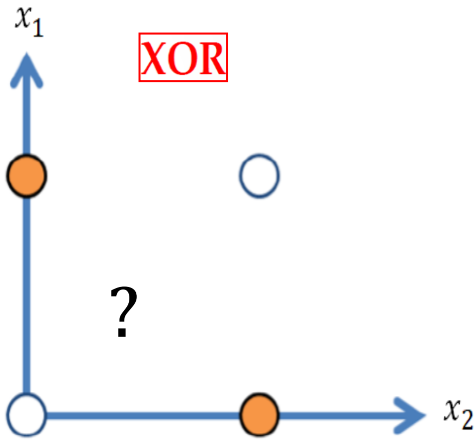
If	$y_i = \hat{y}_i$	-	***then***	no update
	$y_i - \hat{y}_i = 1$	x_i small positive		w_j increased by small amount
	$y_i - \hat{y}_i = 1$	x_i large negative		w_j decreased by large amount
	$y_i - \hat{y}_i = -1$	x_i large negative		w_j increased by large amount

XOR function





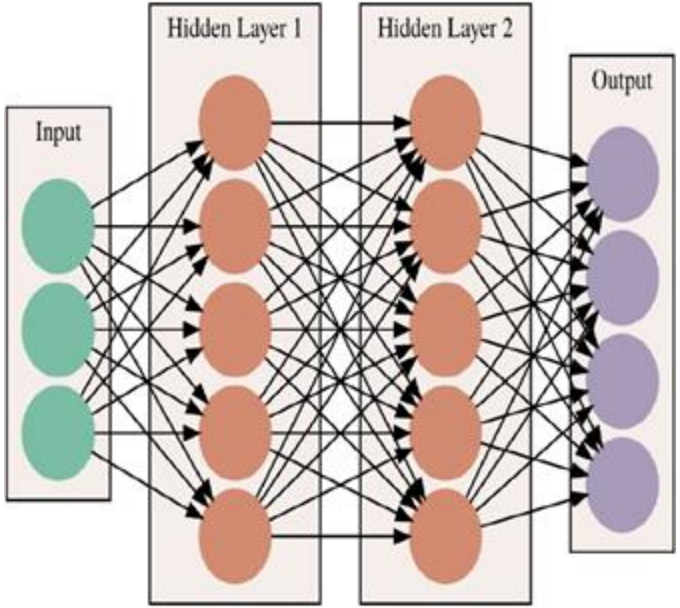
x_1	x_2	x_1 ***XOR*** x_2
0	0	0
0	1	1
1	0	1
1	1	0



Single neuron is only able to draw one single line through input space

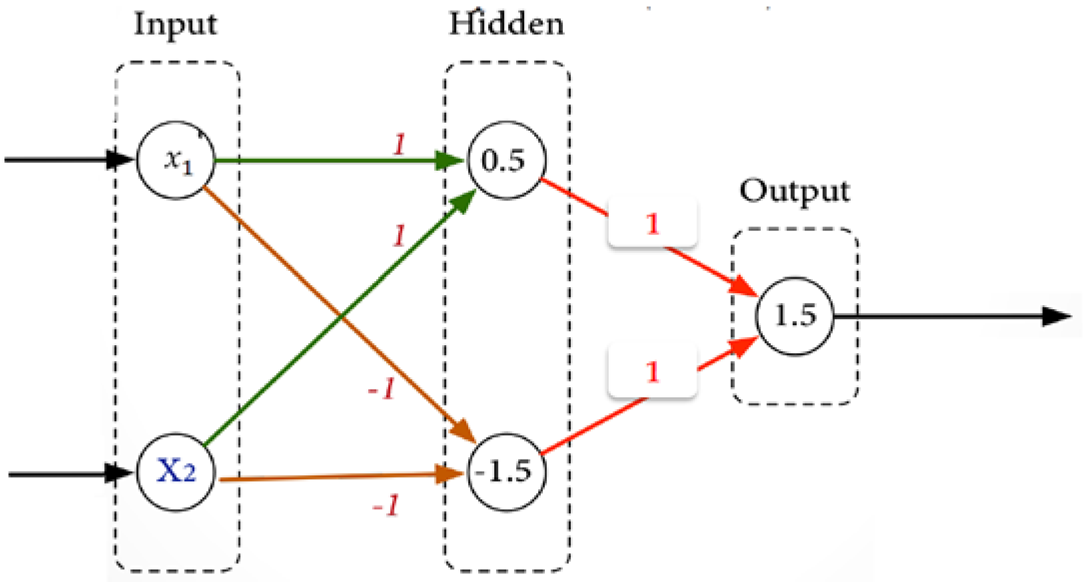
Multi Layer Perceptron (MLP)

- David Rumelhart, Geoffrey Hinton and Ronald Williams published a paper "*Learning representations by back-propagating errors*" (1986), which introduced:
  - **Hidden Layers**
  - **Backpropagation**



- MLPs are composed of the **input layer**, a number of **hidden layers**, and an **output layer.**

MLP for XOR function

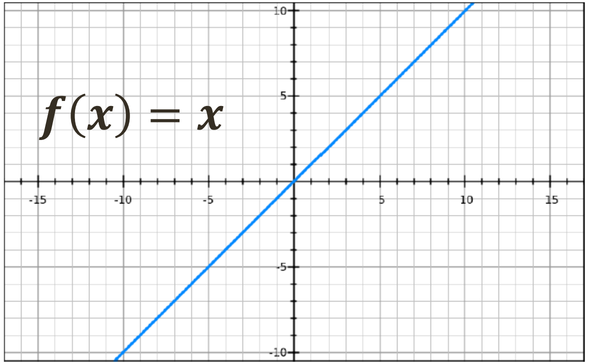


Activation fuction

- Determine the transformation that will be applied to the **weighted** sum of a **neuron** inputs.
- The **activation** of a **perceptron** is the threshold (step) function producing (0,1) or (-1,+1).
- In the case of modern networks, the **activation** is generally a *continuous* function
- Activation function will decide whether neuron should *activated or not*

Activation fuction

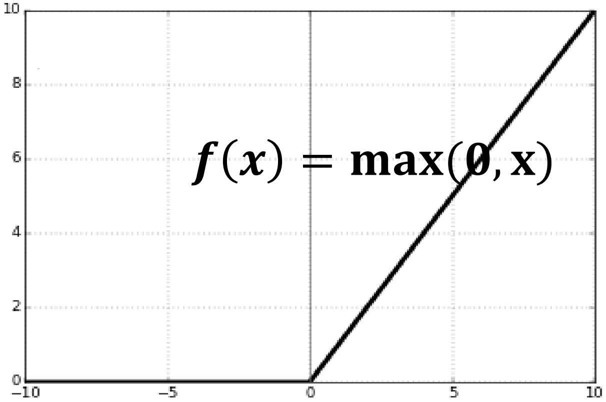
Linear (identity) function



$$f(x) = x$$

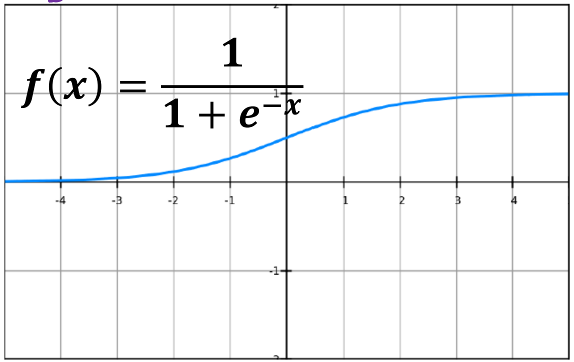
Relu

$f(x) \in [0, +\infty]$



$$f(x) = \max(0, x)$$

Sigmoïd



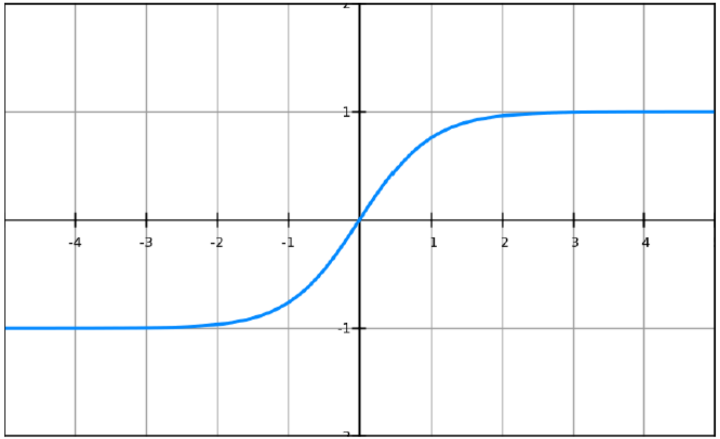
$$f(x) = \frac{1}{1 + e^{-x}}$$

$f(x) \in [0, 1]$

Activation fuction

Tanh

$$f(x) \in [-1, 1]$$



Hyperbolic sine $\quad \sinh x = \dfrac{e^x - e^{-x}}{2}$

Hyperbolic cosine $\quad \cosh x = \dfrac{e^x + e^{-x}}{2}$

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

Activation fuction

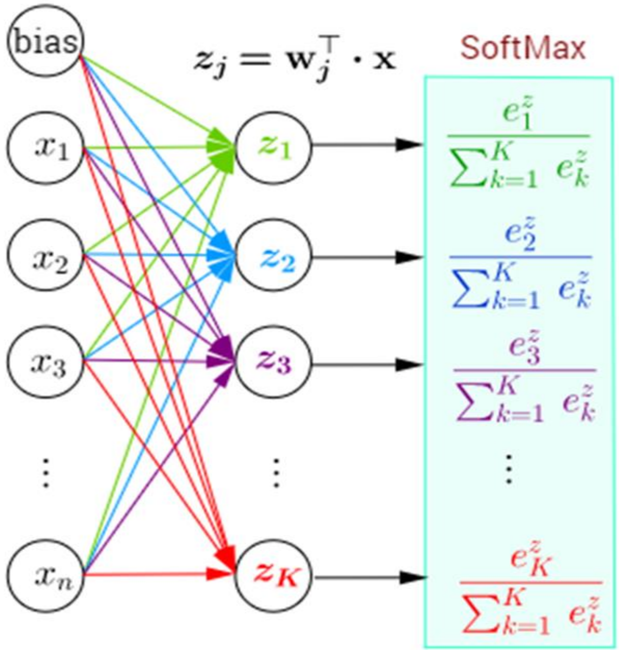
Softmax function

- Outputs interpretable as posterior probabilities for a categorical target variable

- Network with K outputs

$$z_i = \sum(weight_{ji} * input) + bias_i$$

$$\boldsymbol{f(x) = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_o}}}$$



The Backprogation Algorithm

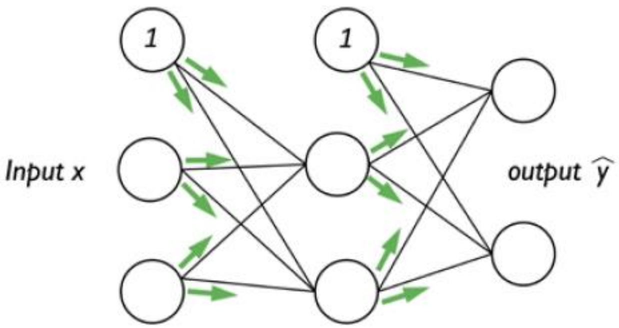
Random initialization of weights

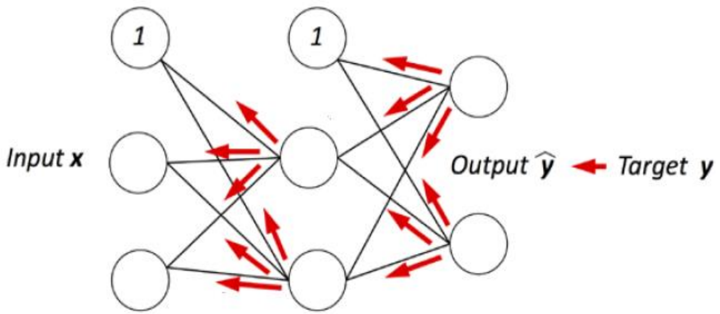
epoch

for each example in training do

- *forward stage*
- *error computation*
- *backward stage*

Input → Forward → Loss function → backpropagation of errors





The Backprogation Algorithm

- Propagating the inputs through each layer until the output layer

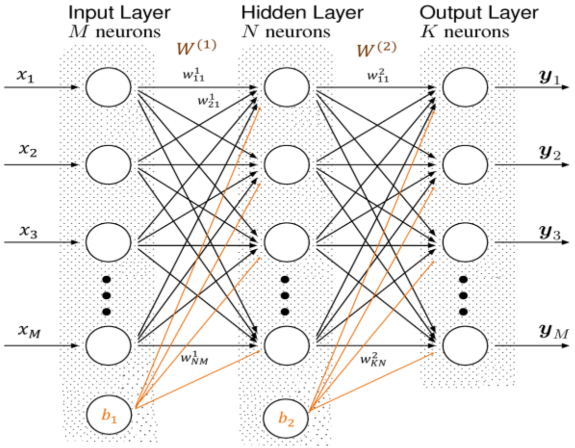
- Generate predictions during training that will need to calculate the loss

- Compute the gradient of this error as a function of the neuron's weights, and adjust its weights in the direction that most decreases the error

  For every weight w_{ij}^l and every bias b_i^l

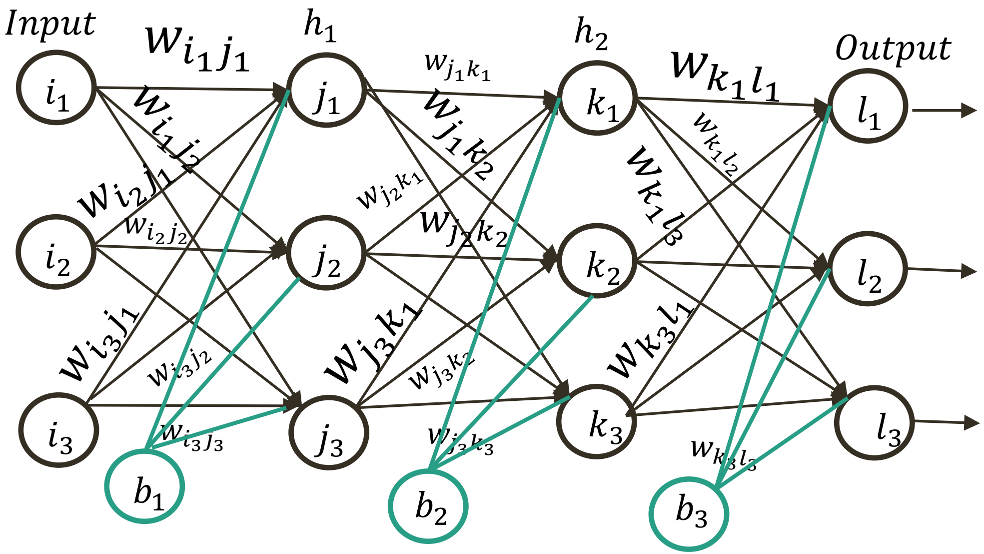
$$w_{ij}^l = w_{ij}^l - \alpha \frac{\partial J(w,b)}{\partial w_{ij}^l}$$

$$b_i^l = b_i^l - \alpha \frac{\partial J(w,b)}{\partial b_i^l}$$

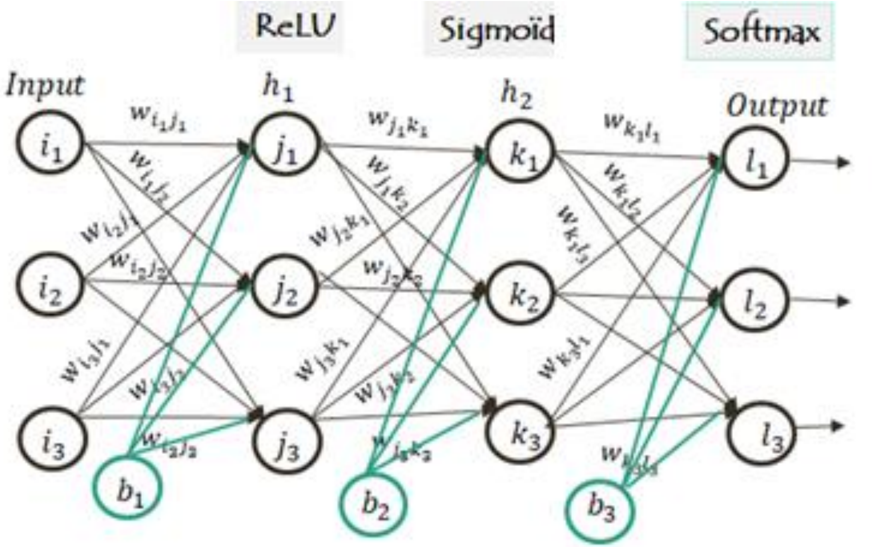


- Propagate these errors backward to infer errors for the hidden layer's

A Step by Step Backpropagation algorithm



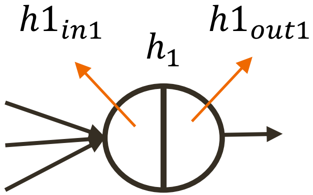
Forward pass: hidden layer n° 1



Hidden layer n°1

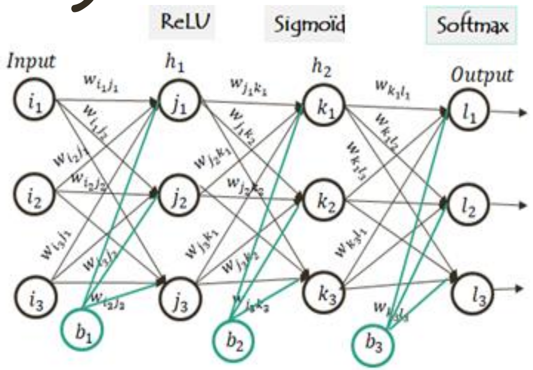
$f(x) = max(0, x)$

$$[\,h1_{in1}, h1_{in2}, h1_{in3}] = [i_1 \quad i_2 \quad i_3\,] \times \begin{bmatrix} w_{i_1j_1} & w_{i_1j_2} & w_{i_1j_3} \\ w_{i_2j_1} & w_{i_2j_2} & w_{i_2j_3} \\ w_{i_3j_1} & w_{i_3j_2} & w_{i_3j_3} \end{bmatrix} + [b_{j_1} \quad b_{j_2} \quad b_{j_3}]$$



$$[\,h1_{out1}, h1_{out2}, h1_{out3}] = [\max(0, h1_{in1}) \ \ \max(0, h1_{in2}) \ \ \max(0, h1_{in3})]$$

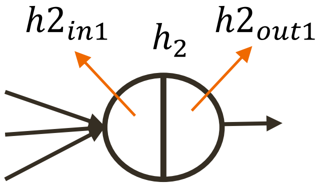
Forward pass: hidden layer n° 2 and output layer



Hidden layer n°2

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

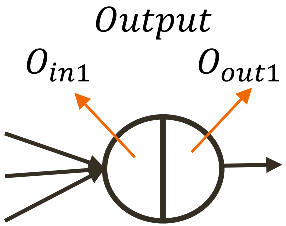
$$[\ h2_{in1},\ h2_{in2},\ h2_{in3}] \ = [j_1 \quad j_2 \quad j_3\ ] \times \begin{bmatrix} w_{j_1k_1} & w_{j_1k_2} & w_{j_1k_3} \\ w_{j_2k_1} & w_{j_2k_2} & w_{j_2k_3} \\ w_{j_3k_1} & w_{j_3k_2} & w_{j_3k_3} \end{bmatrix} + [b_{k_1} \quad b_{k_2} \quad b_{k_3}]$$



$$[\ h2_{out1},\ h2_{out2},\ h2_{out3}] = [1/\left(1+e^{-h2_{in1}}\right) \quad 1/\left(1+e^{-h2_{in2}}\right) \quad 1/(1+e^{-h2_{in1}})]$$

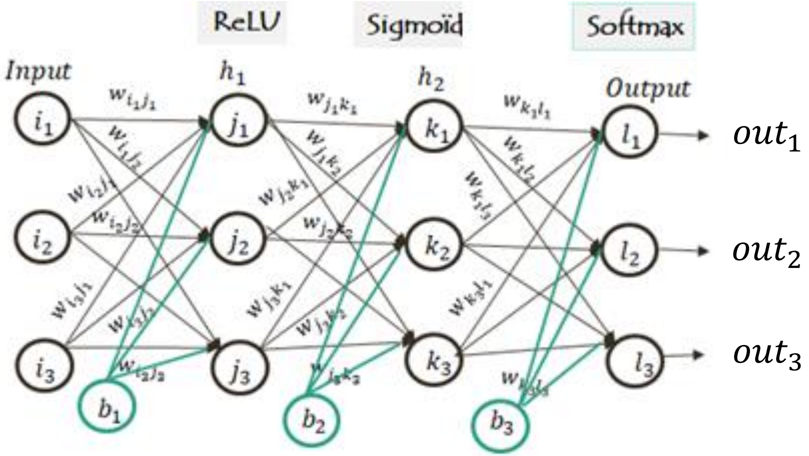
Output layer   *Softmax*

$$[O_{in1}O_{in2}O_{in3}] = [h2_{out1} \quad h2\ _{out2} \quad h2_{out3}] \times \begin{bmatrix} w_{k_1l_1} & w_{k_1l_2} & w_{k_1l_3} \\ w_{k_2l_1} & w_{k_2l_2} & w_{k_2l_3} \\ w_{k_3l_1} & w_{k_3l_2} & w_{k_3l_3} \end{bmatrix} + [b_{l_1} \quad b_{l_2} \quad b_{l_3}]$$



$$O_{out1}\ O_{out2}\ O_{out3} = [e^{O_{in1}}/(\textstyle\sum_{a=1}^{3} e^{O_{ina}}) \quad e^{O_{in2}}/(\sum_{a=1}^{3} e^{O_{ina}}) \quad e^{O_{in3}}/(\sum_{a=1}^{3} e^{O_{ina}})]$$

Error computation



$$Output = [1 \quad 0 \quad 0] \qquad \widehat{Output} = [out_1, out_2, out_3]$$

Cross-Entropy:

$$\boldsymbol{error} = -\frac{\mathbf{1}}{\mathbf{3}}\left(\sum_{i=1}^{3}\left(y_i \times \log\left(O_{out_i}\right)\right) + \left(\left(1 - y_i\right) \times \log\left(\left(1 - O_{out_i}\right)\right)\ \right)\right)$$

Backward pass

- Backpropagation require the use of the chain rule

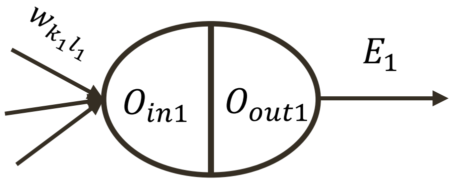
  - Let x be a real number
  - Let f and g be functions mapping from a real number to a real number

  - If $y = g(x)$ and $z = f\big(g(x)\big)$ Then the chain rule states that

$$\frac{dz}{dx} = \frac{dz}{dy} . \frac{dy}{dx}$$

- Backpropagation is obtained recursively by applying the chain rule

$$\frac{\partial E_1}{w_{k_1 l_1}} = \frac{\partial E_1}{\partial O_{out1}} \times \frac{\partial O_{out1}}{\partial O_{in1}} \times \frac{\partial O_{in1}}{\partial W_{k1l1}}$$

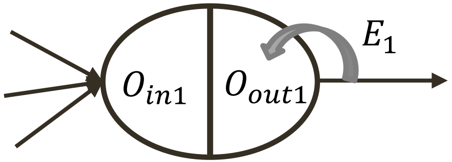


Backward pass

Applying the chain rule $$\frac{\partial E_1}{w_{k_1 l_1}} = \frac{\partial E_1}{\partial O_{out1}} \times \frac{\partial O_{out1}}{\partial O_{in1}} \times \frac{\partial O_{in1}}{\partial W_{k1l1}}$$

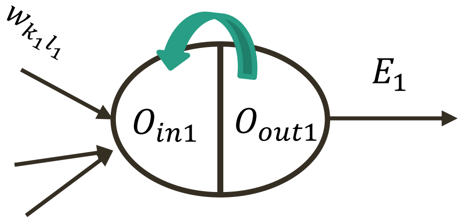
$$\boldsymbol{Cross\ Entropy} = -(\boldsymbol{y_i}\ \boldsymbol{log}(\boldsymbol{O_{out_i}}) + (\mathbf{1} - \boldsymbol{y_i})\boldsymbol{log}(\mathbf{1} - \boldsymbol{O_{out_i}}))$$

$$\begin{bmatrix} \frac{\partial E_1}{\partial O_{out1}} \\ \frac{\partial E_2}{\partial O_{out2}} \\ \frac{\partial E_3}{\partial O_{out3}} \end{bmatrix} = \begin{bmatrix} -((y_1 * 1/O_{out1}) + (1 - y_1) * (1/(1 - O_{out1}))) \\ -((y_1 * 1/O_{out2}) + (1 - y_2) * (1/(1 - O_{out2}))) \\ -((y_1 * 1/O_{out3}) + (1 - y_2) * (1/(1 - O_{out3}))) \end{bmatrix}$$



Backward pass

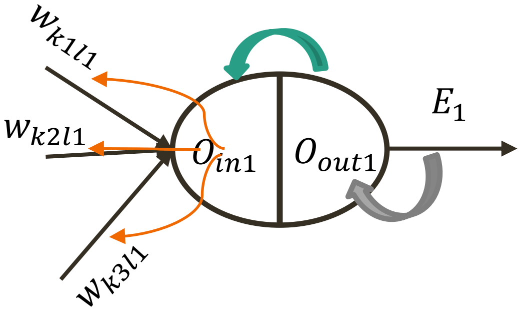
$$\frac{\partial E_1}{w_{k_1 l_1}} = \frac{\partial E_1}{\partial O_{out1}} \times \frac{\partial O_{out1}}{\partial O_{in1}} \times \frac{\partial O_{in1}}{\partial W_{k1l1}}$$



$$Softmax = \frac{\boldsymbol{e^{x_a}}}{\boldsymbol{\sum_{a=1}^{n} e^{x_a}}} \quad \rightarrow \quad \frac{\partial(softmax)}{\partial x_1} = \frac{(e^{x_1} \times (e^{x_2} + e^{x_3}))}{(e^{x_1} + e^{x_2} + e^{x_3})^2}$$

$$\begin{bmatrix} \dfrac{\partial O_{out1}}{\partial O_{in1}} \\ \dfrac{\partial O_{out2}}{\partial O_{in2}} \\ \dfrac{\partial O_{out3}}{\partial O_{in3}} \end{bmatrix} = \begin{bmatrix} e^{O_{in1}}(e^{O_{in2}} + e^{O_{in3}})/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}})^2 \\ e^{O_{in2}}(e^{O_{in1}} + e^{O_{in3}})/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}})^2 \\ e^{O_{in3}}(e^{O_{in1}} + e^{O_{in2}})/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}})^2 \end{bmatrix}$$

Backward pass



$$\frac{\partial E_1}{w_{k_1 l_1}} = \frac{\partial E_1}{\partial O_{out1}} \times \frac{\partial O_{out1}}{\partial O_{in1}} \times \boxed{\frac{\partial O_{in1}}{\partial W_{k1l1}}}$$

$$\frac{\partial O_{in1}}{\partial w_{k1l1}} = \frac{\partial((h2_{out1} * W_{k1l1}) + (h2_{out2} * w_{k2l1}) + (h2_{out3} * w_{k3l1}) + b_{l1})}{\partial w_{k1l1}} = h2_{out1}$$

$$\begin{bmatrix} \frac{\partial O_{in1}}{\partial w_{k1l1}} \\ \frac{\partial O_{in1}}{\partial w_{k2l1}} \\ \frac{\partial O_{in1}}{\partial w_{k3l1}} \end{bmatrix} = \begin{bmatrix} h2_{out1} \\ h2_{out2} \\ h2_{out3} \end{bmatrix} \qquad \begin{bmatrix} \frac{\partial O_{in1}}{\partial w_{k1l2}} \\ \frac{\partial O_{in1}}{\partial w_{k2l2}} \\ \frac{\partial O_{in1}}{\partial w_{k3l2}} \end{bmatrix} = \begin{bmatrix} h2_{out1} \\ h2_{out2} \\ h2_{out3} \end{bmatrix} \qquad \begin{bmatrix} \frac{\partial O_{in1}}{\partial w_{k1l3}} \\ \frac{\partial O_{in1}}{\partial w_{k2l3}} \\ \frac{\partial O_{in1}}{\partial w_{k3l3}} \end{bmatrix} = \begin{bmatrix} h2_{out1} \\ h2_{out2} \\ h2_{out3} \end{bmatrix}$$

Backward pass

$$w'_{k_i l_j} = w_{k_i l_j} - \alpha * \frac{\partial E}{\underbrace{k_i l_j}_{\delta w_{kl}\ ?}}$$

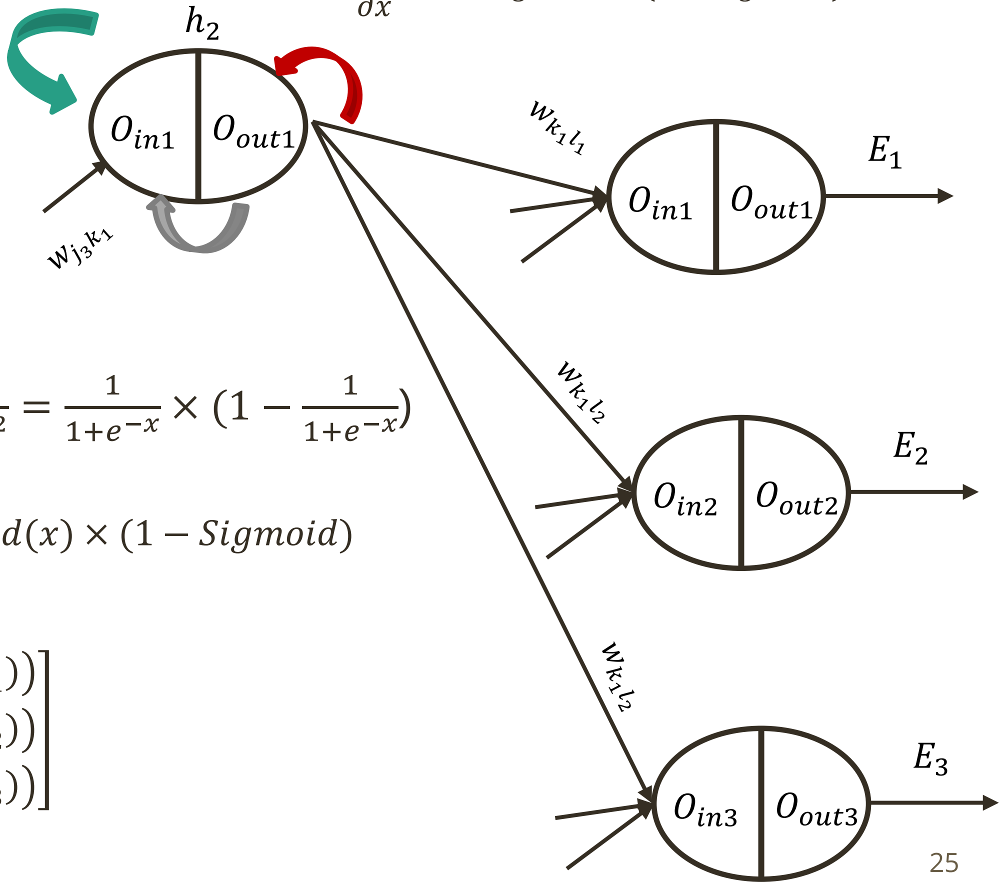
$$\frac{\partial E_j}{w_{k_i l_j}} = \frac{\partial E_j}{\partial O_{out_j}} \times \frac{\partial O_{out_j}}{\partial O_{in_j}} \times \frac{\partial O_{in_j}}{\partial W_{k_i l_j}}$$

$$\delta w_{kl} = \begin{bmatrix} \frac{\partial E_1}{\partial w_{k1l1}} & \frac{\partial E_2}{\partial w_{k1l2}} & \frac{\partial E_2}{\partial w_{k1l3}} \\ \frac{\partial E_1}{\partial w_{k2l1}} & \frac{\partial E_2}{\partial w_{k2l2}} & \frac{\partial E_3}{\partial w_{k2l3}} \\ \frac{\partial E_1}{\partial w_{k3l1}} & \frac{\partial E_2}{\partial w_{k3l2}} & \frac{\partial E_3}{\partial w_{k3l3}} \end{bmatrix} = \begin{bmatrix} \frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial W_{k1l1}} & \frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial W_{k1l2}} & \frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial W_{k1l3}} \\ \frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial W_{k2l1}} & \frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial W_{k2l2}} & \frac{\partial E_1}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial W_{k2l3}} \\ \frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial W_{k3l1}} & \frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial W_{k3l2}} & \frac{\partial E_1}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial W_{k2l3}} \end{bmatrix}$$

$$w'_{kl} = \begin{bmatrix} w_{k1l1} - (\alpha * \delta w_{k1l1}) & w_{k1l2} - (\alpha * \delta w_{k1l2}) & w_{k1l3} - (\alpha * \delta w_{k1l3}) \\ w_{k2l1} - (\alpha * \delta w_{k2l1}) & w_{k2l2} - (\alpha * \delta w_{k2l2}) & w_{k2l3} - (\alpha * \delta w_{k2l3}) \\ w_{k3l1} - (\alpha * \delta w_{k3l1}) & w_{k3l2} - (\alpha * \delta w_{k3l2}) & w_{k3l3} - (\alpha * \delta w_{k3l3}) \end{bmatrix}$$

Backward pass

$$\frac{\partial Sigmoid}{\partial x} = Sigmoid \times (1 - Sigmoid)$$



hidden layer 2 $\rightarrow$ *hidden layer 1*

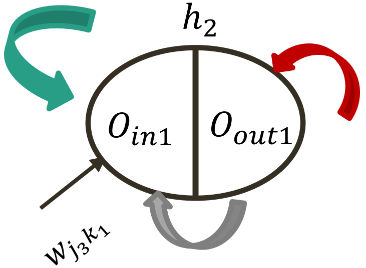
$$\frac{\partial E_{total}}{\partial W_{j3k1}} = \frac{\partial E_{total}}{\partial h2_{out1}} * \left(\frac{\partial h2_{out1}}{\partial h2_{in1}}\right) * \frac{\partial h2_{in1}}{\partial W_{j3k1}}$$

$$Sigmoid(x) = \frac{1}{(1+e^{-x})} \longrightarrow \frac{\partial Sigmoid}{\partial x} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}} \times \left(1 - \frac{1}{1+e^{-x}}\right)$$

$$\longrightarrow \frac{\partial Sigmoid}{\partial x} = Sigmoid(x) \times (1 - Sigmoid)$$

$$\begin{bmatrix} \frac{\partial h2_{out1}}{\partial h2_{in1}} \\ \frac{\partial h2_{out2}}{\partial h2_{in2}} \\ \frac{\partial h2_{out3}}{\partial h2_{in3}} \end{bmatrix} = \begin{bmatrix} Sigmoid(h2_{in1}) * \left(1 - Sigmoid(h2_{in1})\right) \\ Sigmoid(h2_{in2}) * \left(1 - Sigmoid(h2_{in2})\right) \\ Sigmoid(h2_{in3}) * \left(1 - Sigmoid(h2_{in3})\right) \end{bmatrix}$$

Backward pass



$$\frac{\partial E_{total}}{\partial W_{j3k1}} = \frac{\partial E_{total}}{\partial h2_{out1}} * \frac{\partial h2_{out1}}{\partial h2_{in1}} * \left(\frac{\partial h2_{in1}}{\partial W_{j3k1}}\right)$$

$$\frac{\partial h2_{in1}}{\partial W_{j_1k_1}} = \frac{\partial\left((h1_{out1} * W_{j_1k_1}) + (h1_{out2} * W_{j_2k_1}) + (h1_{out3} * W_{j_3k_1}) + b_{k1}\right)}{\partial W_{j1k1}} = h1_{out1}$$

$$\begin{bmatrix} \frac{\partial h2_{in1}}{\partial W_{j1k1}} \\ \frac{\partial h2_{in1}}{\partial W_{j2k1}} \\ \frac{\partial h2_{in1}}{\partial W_{j3k1}} \end{bmatrix} = \begin{bmatrix} h1_{out1} \\ h1_{out2} \\ h1_{out3} \end{bmatrix} \qquad \begin{bmatrix} \frac{\partial h2_{in2}}{\partial W_{j1k2}} \\ \frac{\partial h2_{in2}}{\partial W_{j2k2}} \\ \frac{\partial h2_{in2}}{\partial W_{j3k2}} \end{bmatrix} = \begin{bmatrix} h1_{out1} \\ h1_{out2} \\ h1_{out3} \end{bmatrix} \qquad \begin{bmatrix} \frac{\partial h2_{in3}}{\partial W_{j1k3}} \\ \frac{\partial h2_{in3}}{\partial W_{j2k3}} \\ \frac{\partial h2_{in3}}{\partial W_{j3k3}} \end{bmatrix} = \begin{bmatrix} h1_{out1} \\ h1_{out2} \\ h1_{out3} \end{bmatrix}$$

Backward pass

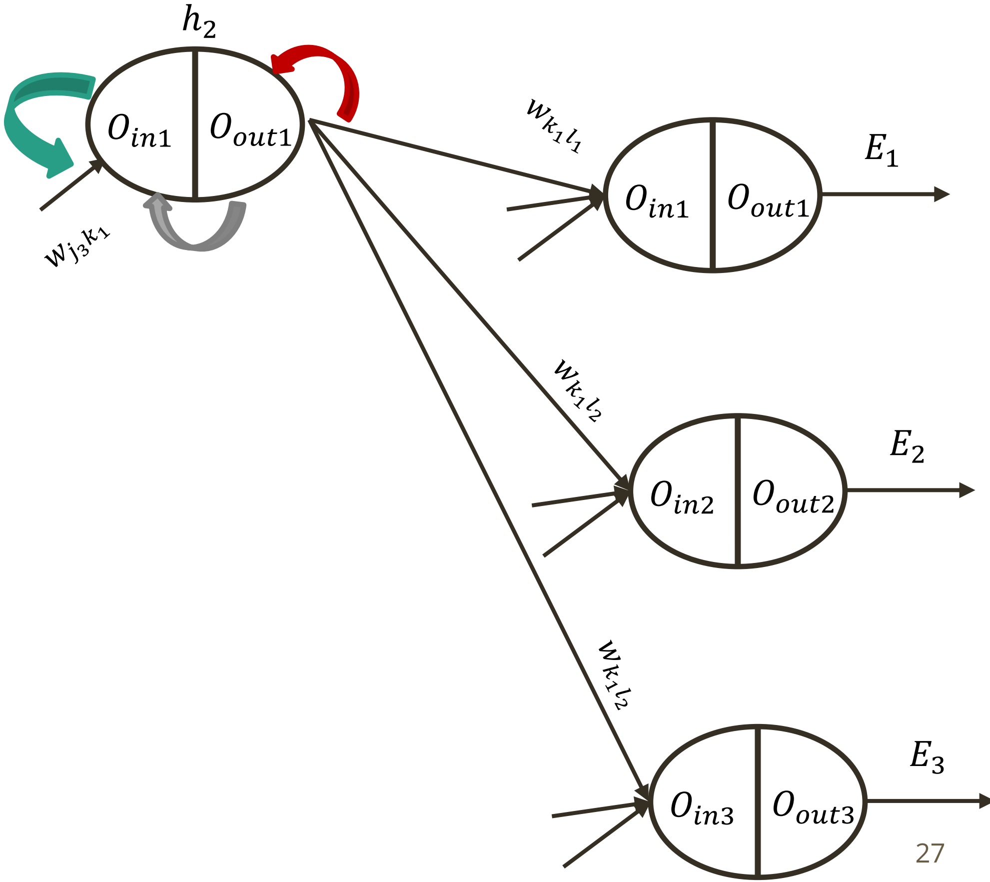
$$\frac{\partial E_{total}}{w_{j_n k_m}} = \left(\frac{\partial E_{total}}{\partial h2_{out_m}}\right) \times \frac{\partial h2_{out_m}}{\partial h2_{in_j}} \times \frac{\partial h2_{in_m}}{\partial W_{j_n k_m}}$$

$$\frac{\partial E_{total}}{\partial h2_{out1}} = \frac{\partial E_1}{\partial h2_{out1}} + \frac{\partial E_2}{\partial h2_{out1}} + \frac{\partial E_3}{\partial h2_{out1}}$$

$$\frac{\partial E_1}{\partial h2_{out1}} = \frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial h2_{out1}}$$

$$\frac{\partial E_2}{\partial h2_{out1}} = \frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial h2_{out1}}$$

$$\frac{\partial E_3}{\partial h2_{out1}} = \frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial h2_{out1}}$$



$$\frac{\partial E_{total}}{w_{j_n k_m}} = \frac{\partial E_{total}}{\partial h2_{out_m}} \times \frac{\partial h2_{out_m}}{\partial h2_{in_j}} \times \frac{\partial h2_{in_m}}{\partial W_{j_n k_m}}$$

$$\frac{\partial E_2}{\partial h2_{out1}} = \frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial h2_{out1}}$$

$$\frac{\partial E_{total}}{\partial h2_{out1}} = \frac{\partial E_1}{\partial h2_{out1}} + \frac{\partial E_2}{\partial h2_{out1}} + \frac{\partial E_3}{\partial h2_{out1}}$$

Backward pass

$$w'_{j_n k_m} = w_{j_n k_m} - \alpha * \frac{\partial E_{total}}{j_n k_m}$$

$$\delta w_{jk} = \begin{bmatrix} \frac{\partial E_{total}}{\partial w_{j1k1}} & \frac{\partial E_{total}}{\partial w_{j1k2}} & \frac{\partial E_{total}}{\partial w_{j1k3}} \\ \frac{\partial E_1}{\partial w_{j2k1}} & \frac{\partial E_2}{\partial w_{j2k2}} & \frac{\partial E_3}{\partial w_{j2k3}} \\ \frac{\partial E_1}{\partial w_{j3k1}} & \frac{\partial E_2}{\partial w_{j3k2}} & \frac{\partial E_3}{\partial w_{j3k3}} \end{bmatrix} = \begin{bmatrix} \frac{\partial E_{total}}{\partial h2_{out1}} * \frac{\partial h2_{out1}}{\partial h2_{in1}} * \frac{\partial h2_{in1}}{\partial W_{j1k1}} & \frac{\partial E_{total}}{\partial h2_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial W_{j1k2}} & \frac{\partial E_{total}}{\partial h2_{out3}} * \frac{\partial h2_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial W_{j1k3}} \\ \frac{\partial E_{total}}{\partial h2_{out1}} * \frac{\partial h2_{out1}}{\partial h2_{in1}} * \frac{\partial h2_{in1}}{\partial W_{j2k1}} & \frac{\partial E_{total}}{\partial h2_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial W_{j2k2}} & \frac{\partial E_{total}}{\partial h2_{out3}} * \frac{\partial h2_{out3}}{\partial h2_{in3}} * \frac{\partial h2_{in3}}{\partial W_{j2k3}} \\ \frac{\partial E_{total}}{\partial h2_{out1}} * \frac{\partial h2_{out1}}{\partial h2_{in1}} * \frac{\partial h2_{in1}}{\partial W_{j3k1}} & \frac{\partial E_{total}}{\partial h2_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial W_{j3k2}} & \frac{\partial E_{total}}{\partial h2_{out3}} * \frac{\partial h2_{out3}}{\partial h2_{in3}} * \frac{\partial h2_{in3}}{\partial W_{j3k3}} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial E_{total}}{\partial h2_{out1}} \\ \frac{\partial E_{total}}{\partial h2_{out2}} \\ \frac{\partial E_{total}}{\partial h2_{out3}} \end{bmatrix} = \begin{bmatrix} \left(\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial h2_{out1}}\right) + \left(\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial h2_{out1}}\right) + \left(\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial h2_{out1}}\right) \\ \left(\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial h2_{out2}}\right) + \left(\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial h2_{out2}}\right) + \left(\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial h2_{out2}}\right) \\ \left(\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial h2_{out3}}\right) + \left(\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * \frac{\partial O_{in2}}{\partial h2_{out3}}\right) + \left(\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * \frac{\partial O_{in3}}{\partial h2_{out3}}\right) \end{bmatrix}$$

Backward pass

$$w'_{j_n k_m} = w_{j_n k_m} - \alpha * \frac{\partial E_{total}}{j_n k_m}$$

$$\begin{bmatrix} \frac{\partial O_{in1}}{\partial h2_{out1}} & \frac{\partial O_{in2}}{\partial h2_{out1}} & \frac{\partial O_{in3}}{\partial h2_{out1}} \\ \frac{\partial O_{in1}}{\partial h2_{out2}} & \frac{\partial O_{in2}}{\partial h2_{out2}} & \frac{\partial O_{in3}}{\partial h2_{out2}} \\ \frac{\partial O_{in1}}{\partial h2_{out3}} & \frac{\partial O_{in2}}{\partial h2_{out3}} & \frac{\partial O_{in3}}{\partial h2_{out3}} \end{bmatrix} = \begin{bmatrix} W_{k1l1} & W_{k1l2} & W_{k1l3} \\ W_{k2l1} & W_{k2l2} & W_{k2l3} \\ W_{k3l1} & W_{k3l2} & W_{k3l3} \end{bmatrix}$$

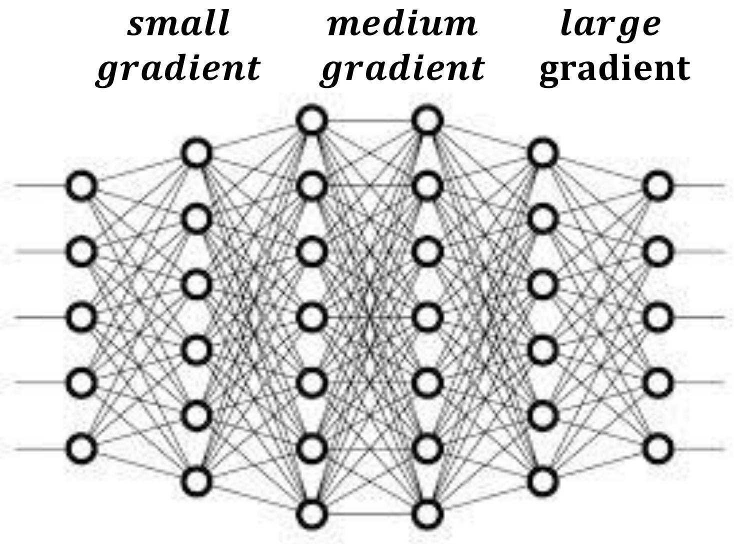
$$\begin{bmatrix} \frac{\partial E_{total}}{\partial h2_{out1}} \\ \frac{\partial E_{total}}{\partial h2_{out2}} \\ \frac{\partial E_{total}}{\partial h2_{out3}} \end{bmatrix} = \begin{bmatrix} (\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * W_{k1l1}) + (\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * W_{k1l2}) + (\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * W_{k1l3}) \\ (\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * W_{k2l1}) + (\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * W_{k2l2}) + (\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * W_{k2l3}) \\ (\frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * W_{k3l1}) + (\frac{\partial E_2}{\partial O_{out2}} * \frac{\partial O_{out2}}{\partial O_{in2}} * W_{k3l2}) + (\frac{\partial E_3}{\partial O_{out3}} * \frac{\partial O_{out3}}{\partial O_{in3}} * W_{k2l3}) \end{bmatrix}$$

... ...

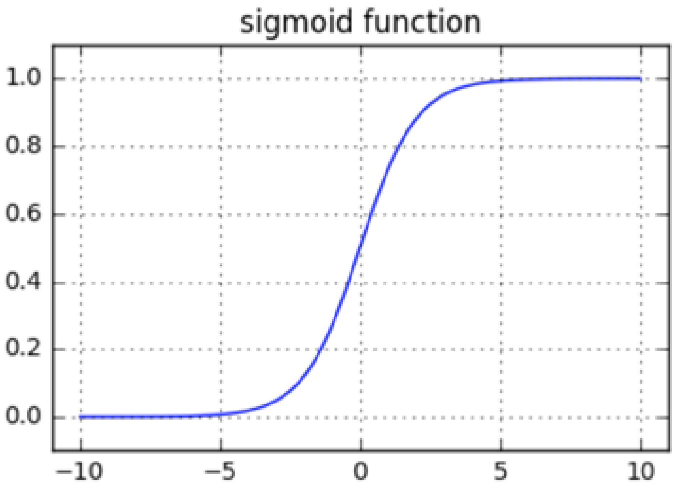
Exploding and Vanishing Gradients

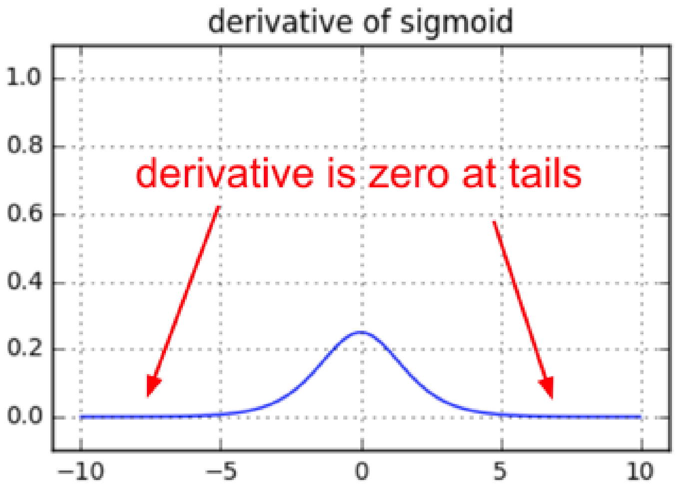
- Concerns neural networks with several hidden layers (deep neural networks)

- Gradients propagated over many stages tend to vanish.



Why Vanishing Gradients?

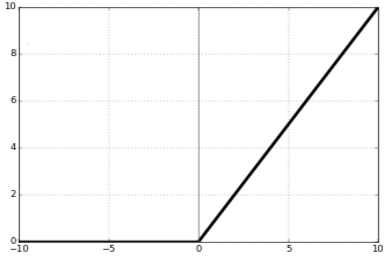




derivative close to 0 ➤ no gradient to propagate back through the network

Avoiding Vanishing Gradients

$f(x) = \max(0, x)$



- $ReLU$ have been shown to ameliorate the vanishing gradient problem

Problem during training, some neurons effectively die, meaning they stop outputting anything other than 0 (problem is known as the *dying ReLUs*.

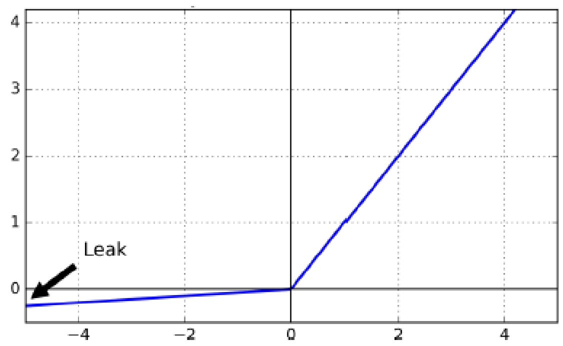
- Solution: Nonsaturating Activation Functions
  - $LeakyReLU$ (variant of the $ReLU$)
  - *Exponential Lineaire Units*
  - *Scaled ELU*

Leaky ReLUs

$$LeakyReLU_{\alpha}(x) = \max(\alpha x, x)$$

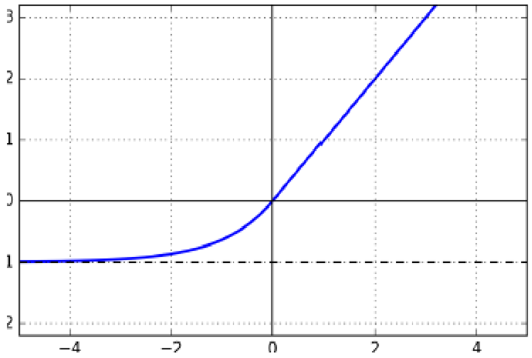
- small gradient when the unit is not active

With $\alpha = 0.01$ the neuron can go a long coma but never die.



Exponential Linear Units

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



Avoiding Overfitting

- Neural network with many hidden layers (**deep learning**) have a large number of parameters ➤ large amount of data, else overfitting

Data augmentation

Transfer learning

Early stopping

$\ell 1$ and $\ell 2$ Regularization

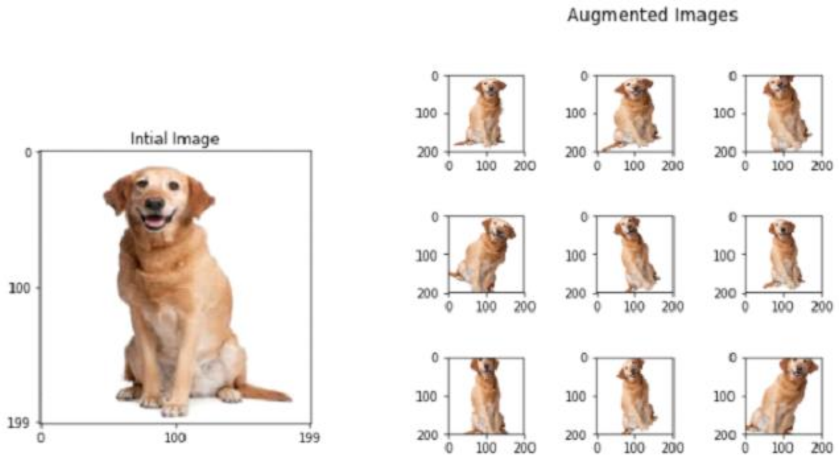
Dropout / Alpha Dropout

prevent overfitting

Data Augmentation

- Artificially boosting the size of the training set.

- Generating new training instances from existing ones (shifting, rotating, resizing, adjusting, contrast, …)

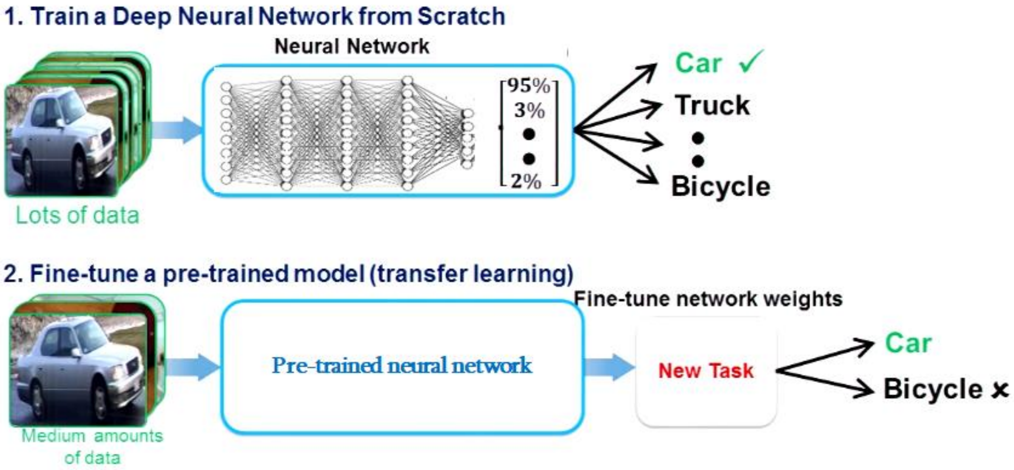


Transfer Learning

- Transferring the knowledge learned on one task (source task) to a second related task (target task).

Task 1

Task 2



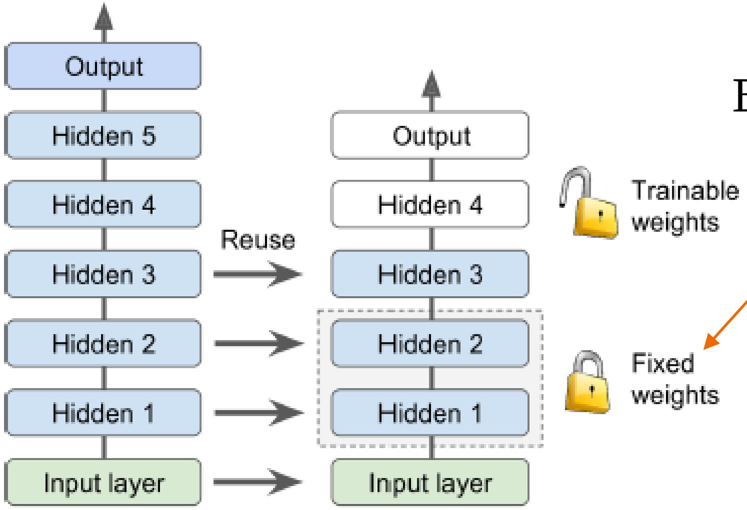
Task 1

Task 2

- Instead of training a deep network from scratch for your task

Transfer Learning

- Transfert learning = train a base network and then copy its first n layers to the first n layers of a target network



Bottom n layers can be frozen or fine tuned

- **Forzen** not updated during backprop

  Target task labels are scarce, number of parameters is large and we want to avoid overfetting

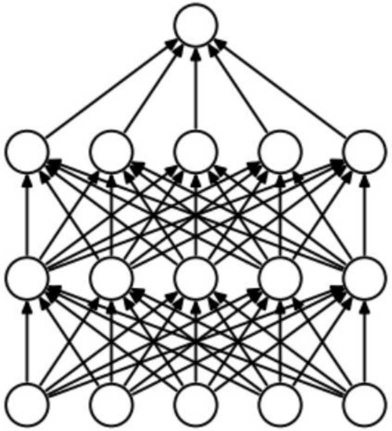
- **Fine-tuned** updated during backprop

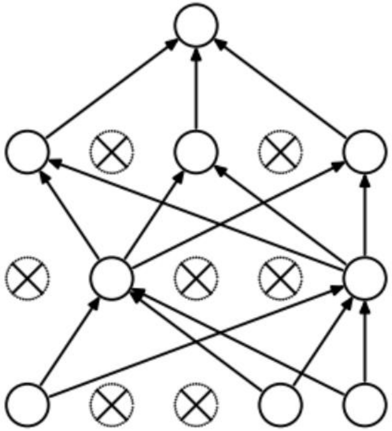
  Target task labels are more peintiful

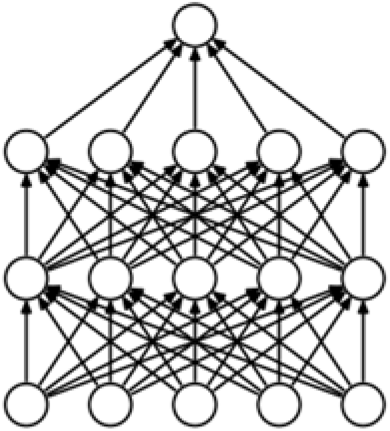
  If the target dataset is small, fine-tuning may result in overfitting

Dropout

- Randomly and temporary *drops* units from a layer on each training step


(a) Standard Neural Net


(b) After applying dropout.


(c) Test *dropout rate* = 1

- Choise of units to drop is random, determined by a probability p (*dropout rate*)

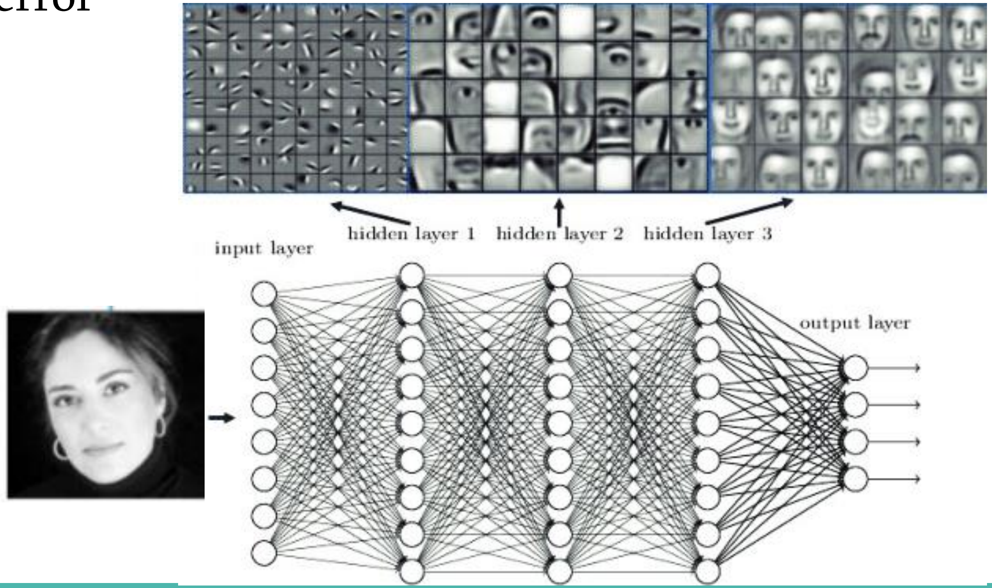
➡ Creating sub-architectures within the model

Hyperparameters in Deep Neural Network

- Variables which determines the network structure and the variables which determine how the network is trained.

- Number of Hidden Layers

  Adding layers until the test error does not improve anymore.



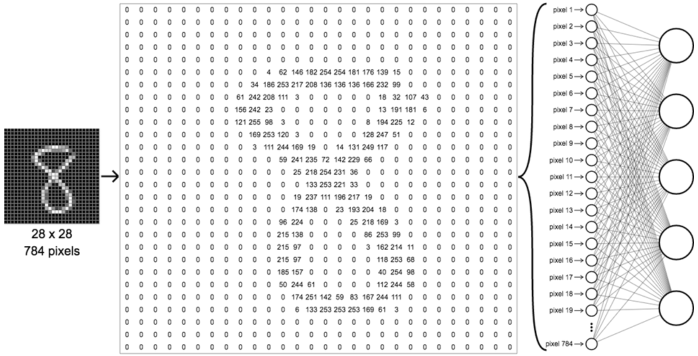
Hyperparameters in Neural Network

- **Number of neurons**

  - ***Input and output layers*** determined by the type of input and output

    For MNIST handwritten digit recognition task (0 to 9)

    - **784** input neurons (each image in MNIST has 28 × 28 pixels= 784 features)
    - **10** output neurons (one neuron per class)



  - ***Hidden layers***

    - Increasing the number of neurones gradually until the network starts overfitting.

Hyperparameters in Neural Network

- ❑ **Number of hidden layers**

  - Keep adding layers until the test error does not improve anymore.

- ❑ **Dropout**

  - Overfitting ➡ increase the $dropout\ rate$

  - Underfitting ➡ decrease the $dropout\ rate$

  - Dropout rate = 0.5 usually works well

- ❑ **Activation fuction**

  - Hidden layers: no saturating function (ELU, $Leaky\ ReLU$, …)

  - Output layer: Softmax (multi-class predictions) or Sigmoid (binary predictions)

Hyperparameters in Neural Network

- Learning Rate
  - **Low learning rate** slows down the learning process but converges smoothly.
  - **Larger learning rate** speeds up the learning but may not converge.
  - Usually a decaying Learning rate is preferred.
- Loss function
- Number of epochs
  - 1 epoch = one forward pass and one backward pass off all the training examples
  - Increase the number of epochs until the validation accuracy starts decreasing even when training accuracy is increasing(overfitting).

Hyperparameters in Neural Network

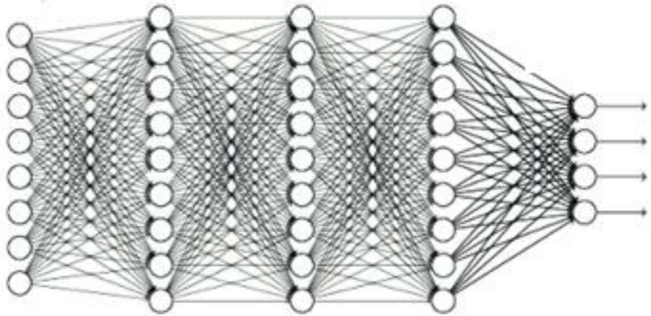
- **Batch size**
  - Total number of training examples in one forward/backward
  - Typically chosen between 1 and a few hundreds.

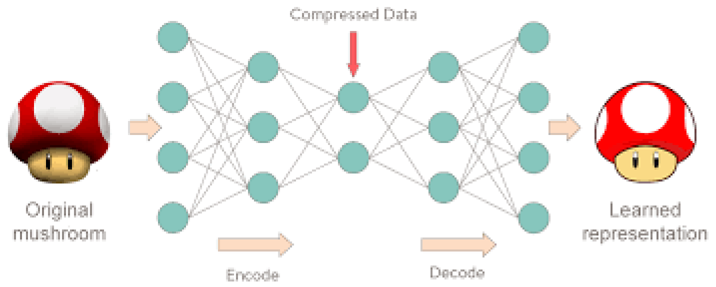
- **Number of iterations**
  - Number of batches needed to complete one epoch.

  Example: with 55000 training data, 100 batch size ➡ 1 epoch = 550 iterations

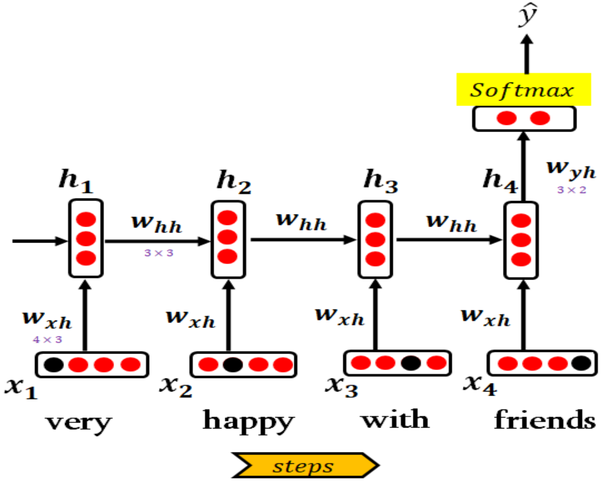
- **Momentum**
  - Helps accelerate gradients vectors in the right directions, thus leading to faster converging

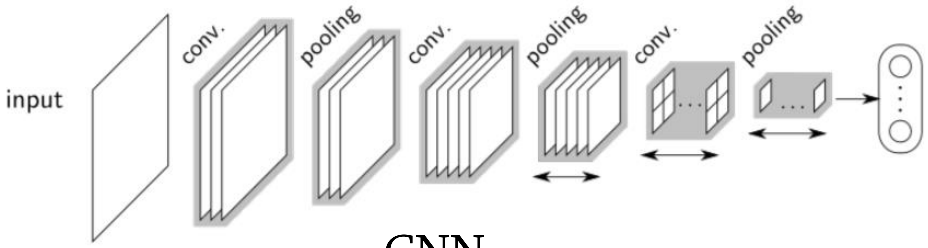
Deep learning algorithms





- RNN
- LSTM
- GRU



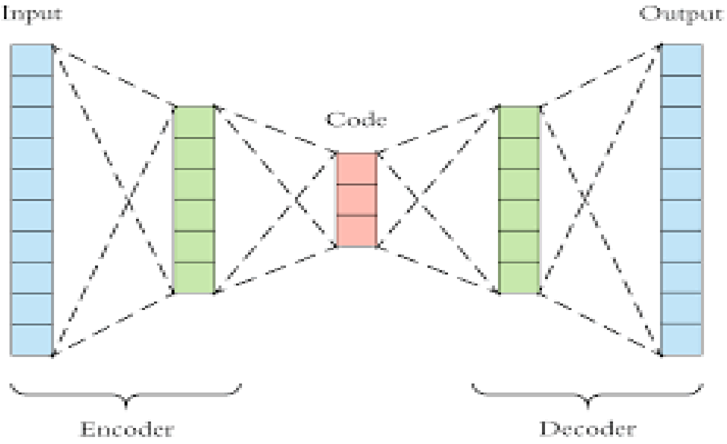


CNN

Autoencoders

- 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.



Why Autoencoders

- ❑ Dimensionality reduction
- ❑ Feature extraction (Unsupervised pretraining)