



SINGAPORE UNIVERSITY OF  
TECHNOLOGY AND DESIGN

Established in collaboration with MIT

# Introduction to Deep Learning

*Soujanya Poria*

50.038 Computational data science

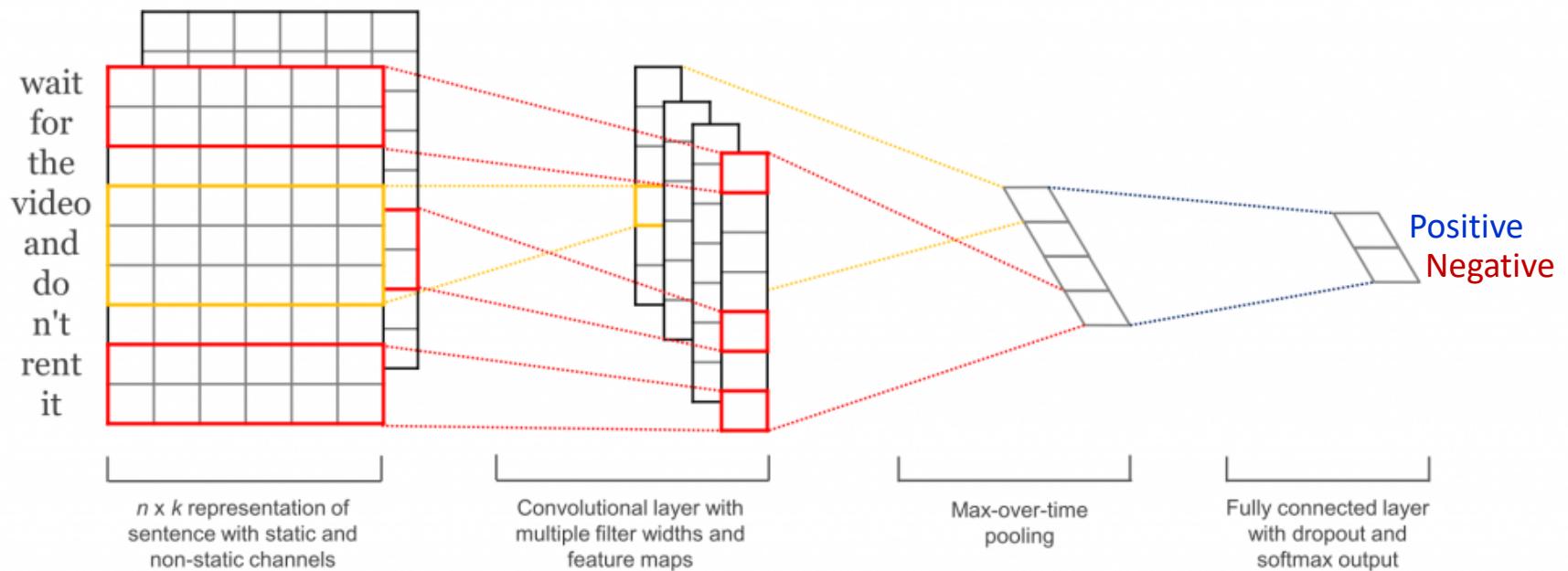
# Objectives

- Understand the basic structure of a neural network, particularly a Multi-layer Perceptron
- Understand the various steps in training a neural network

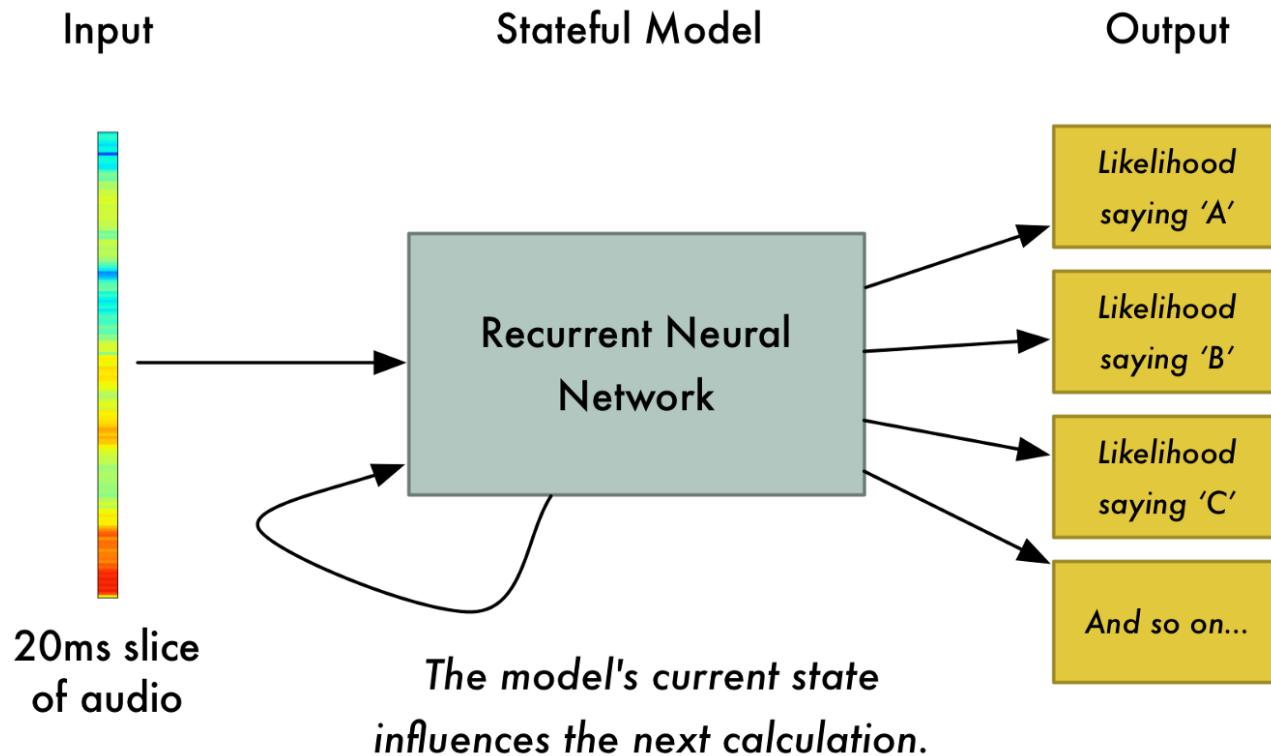
# Examples: Images



# Examples: Text

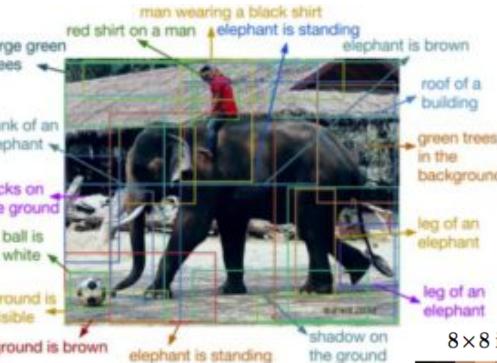


# Examples: Audio



<https://medium.com/@ageitgey/machine-learning-is-fun-part-6-how-to-do-speech-recognition-with-deep-learning-28293c162f7a>

# Various other examples

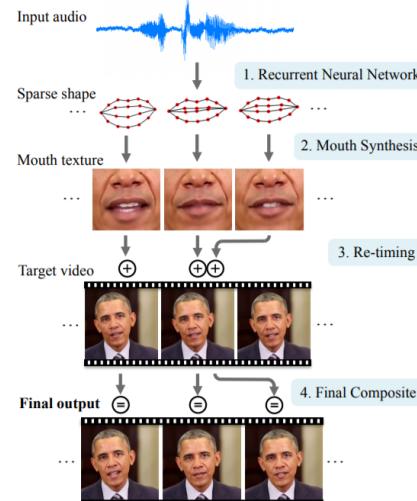


8 × 8 input    32 × 32 samples    ground truth

## Image Captioning



Colouring B&W Photos



Synthesizing Speech

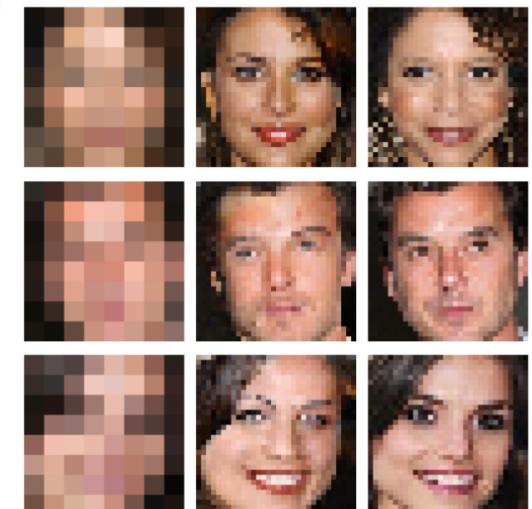
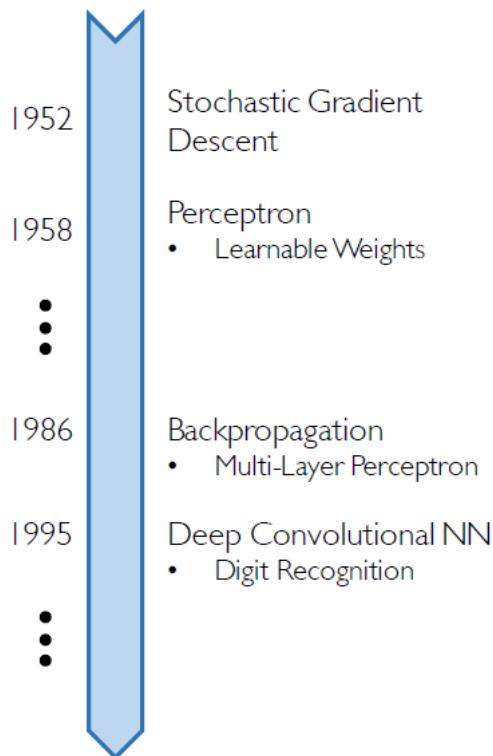


Image Super-resolution

# Why Deep Learning?



Neural Networks date back decades, so why the resurgence?

## I. Big Data

- Larger Datasets
- Easier Collection & Storage



## 2. Hardware

- Graphics Processing Units (GPUs)
- Massively Parallelizable

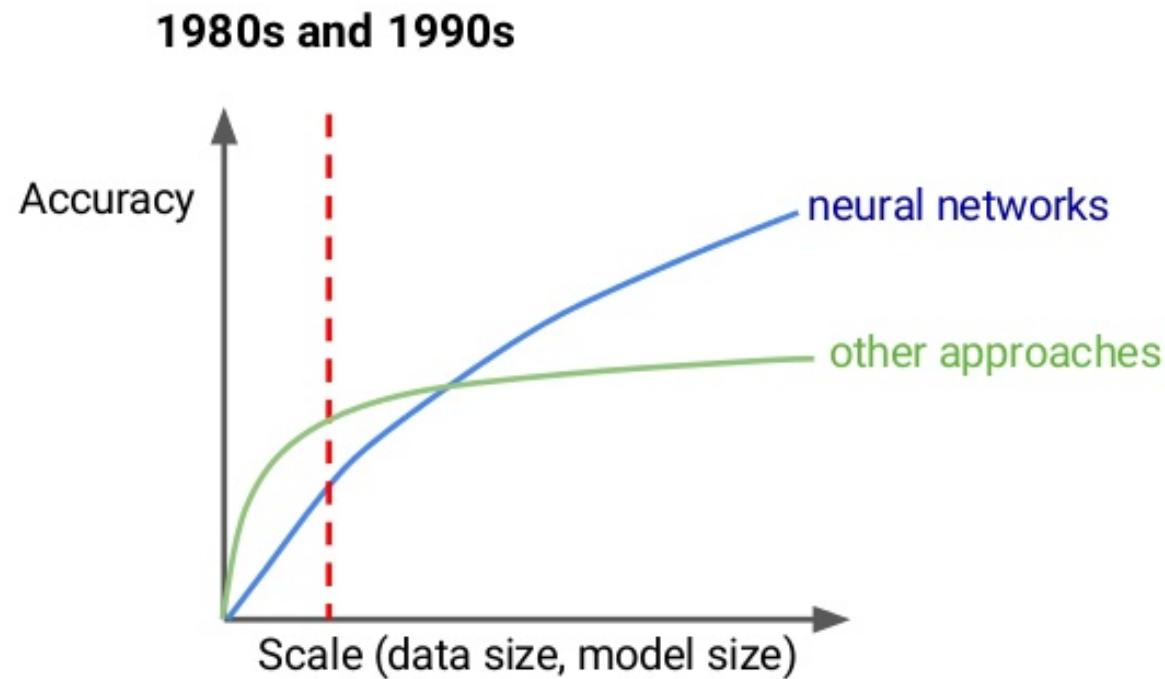


## 3. Software

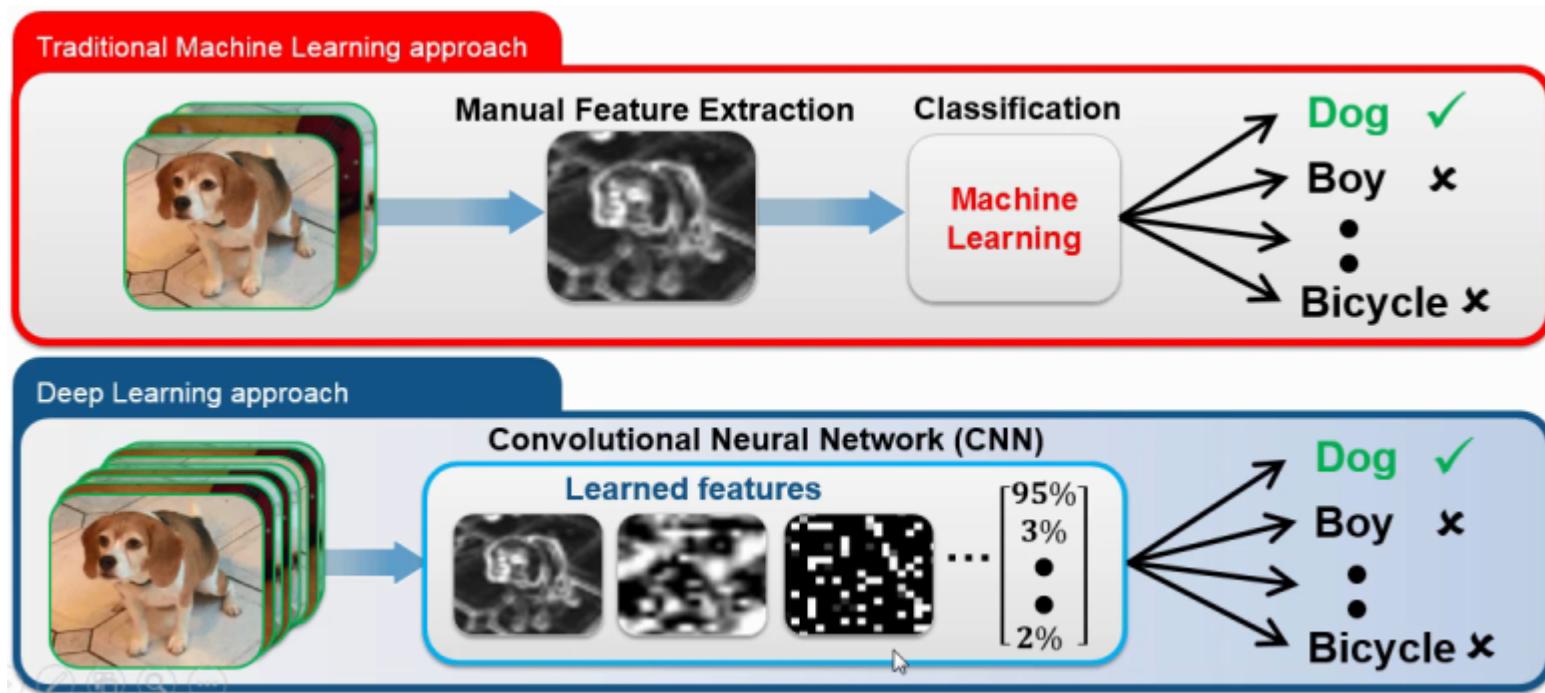
- Improved Techniques
- New Models
- Toolboxes



# Why Deep Learning?

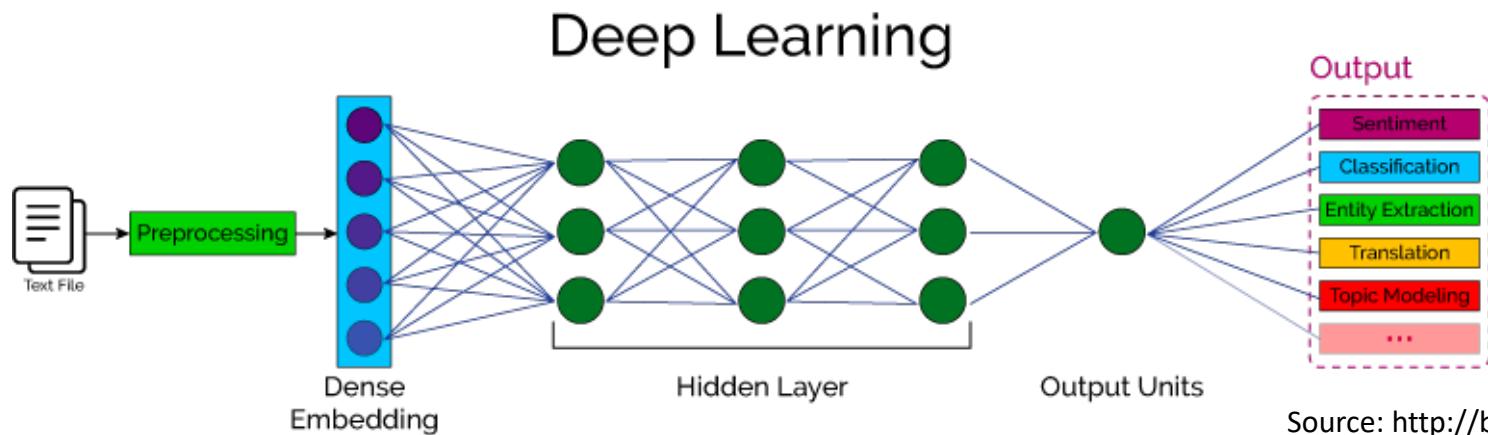
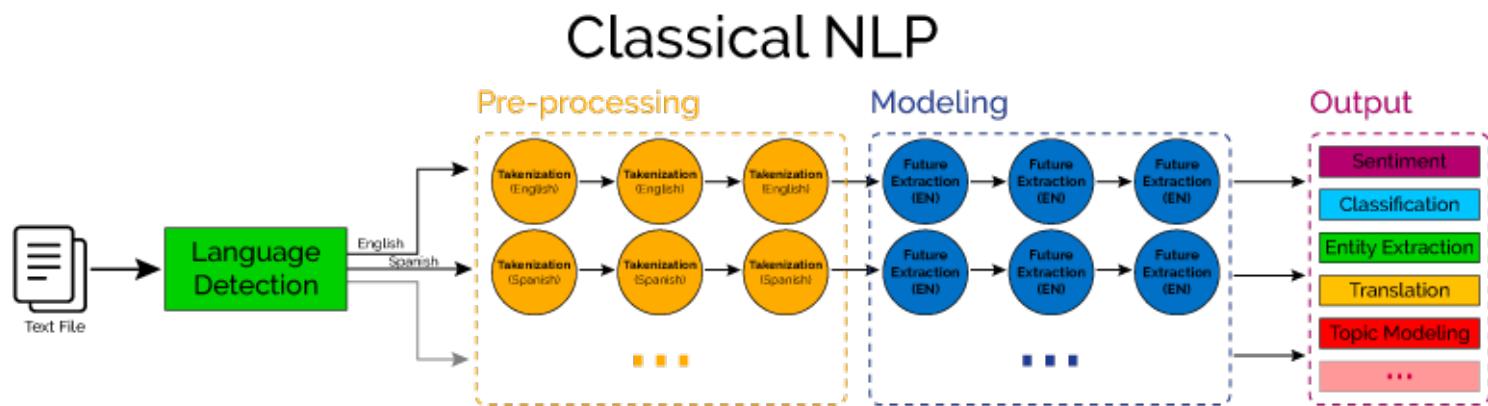


# Differences



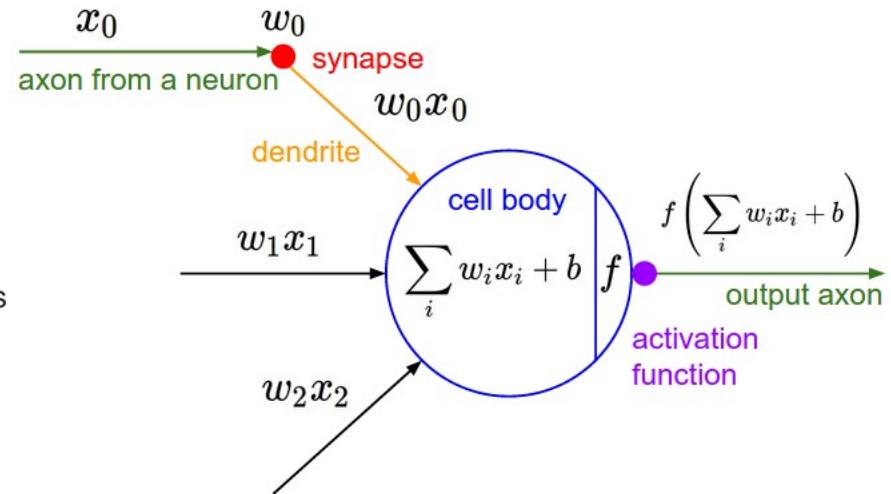
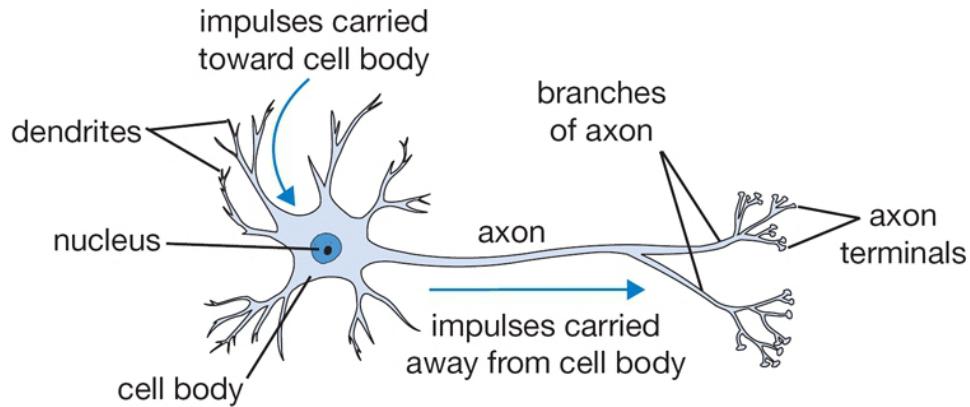
[https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep\\_learning.html](https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep_learning.html)

# Differences



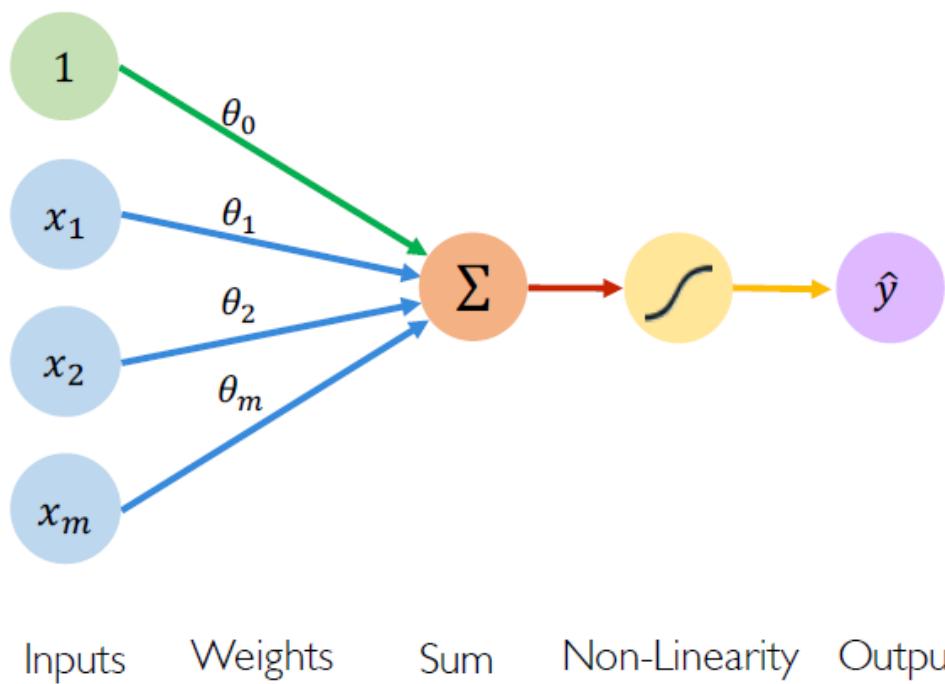
Source: <http://blog.aylien.com>

# Inspiration behind Neural Networks



<http://cs231n.github.io/neural-networks-1/>

# Perceptron



Linear combination of inputs  $\downarrow$

Output  $\downarrow$

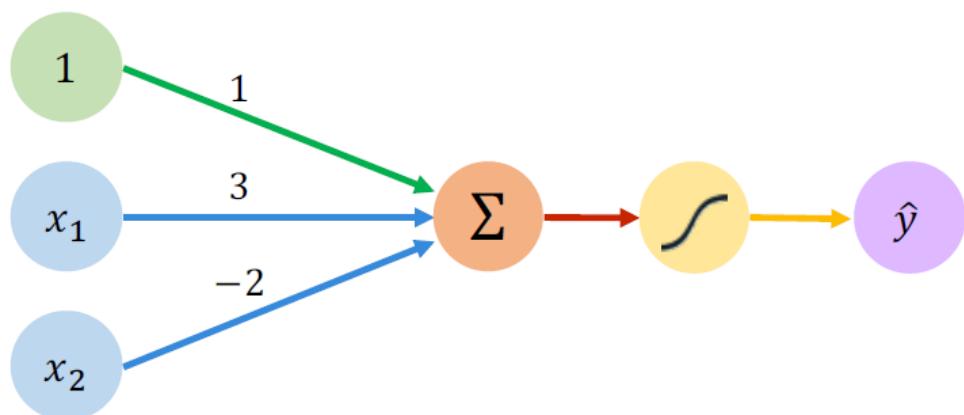
$$\hat{y} = g \left( \theta_0 + \sum_{i=1}^m x_i \theta_i \right)$$

Non-linear activation function  $\uparrow$

Bias  $\downarrow$

Diagram annotations explain the mathematical formula. A red arrow labeled "Linear combination of inputs" points to the summation term. A purple arrow labeled "Output" points to  $\hat{y}$ . A green arrow labeled "Bias" points to  $\theta_0$ . A yellow arrow labeled "Non-linear activation function" points to the sigmoid function  $g$ .

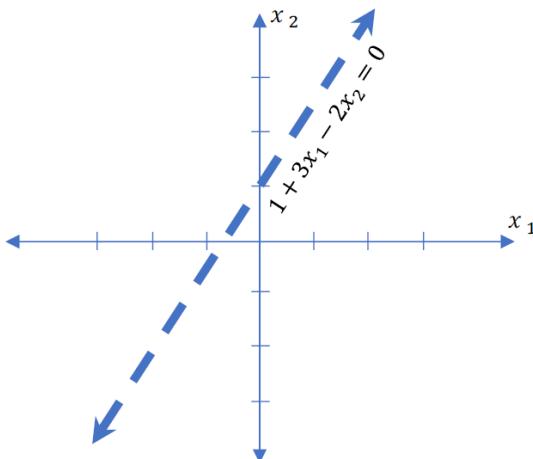
# Why Non-linear Activation?



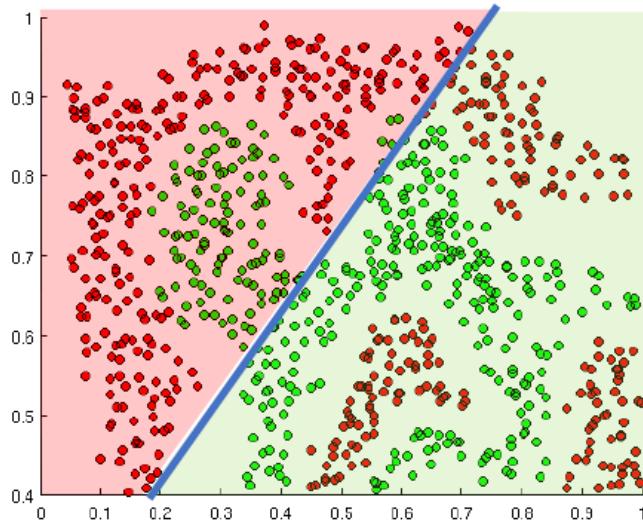
We have:  $\theta_0 = 1$  and  $\boldsymbol{\theta} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\begin{aligned}\hat{y} &= g(\theta_0 + \mathbf{X}^T \boldsymbol{\theta}) \\ &= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right) \\ \hat{y} &= g(1 + 3x_1 - 2x_2)\end{aligned}$$

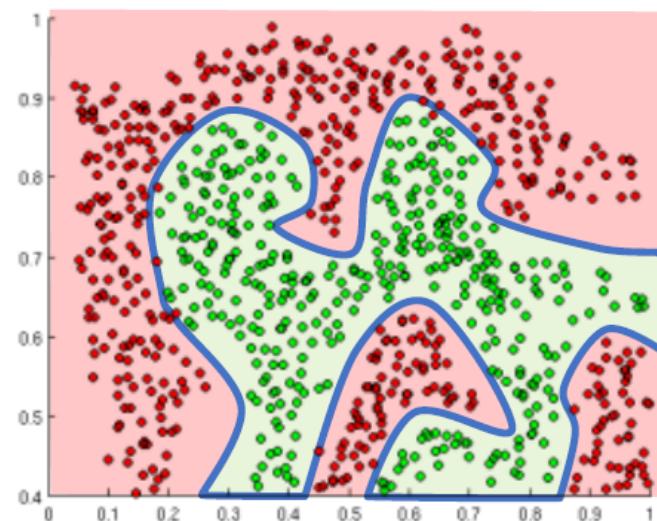
This is just a line in 2D!



# Why Non-linear Activation?



Linear Activation functions produce linear decisions no matter the network size

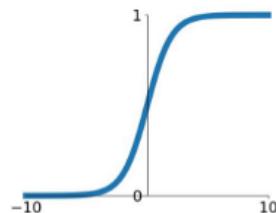


Non-linearities allow us to approximate arbitrarily complex functions

# Activation Functions

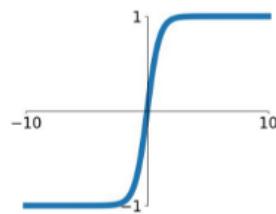
**Sigmoid**

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



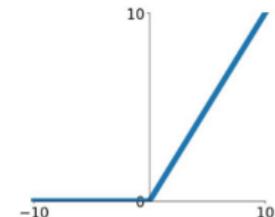
**tanh**

$$\tanh(x)$$



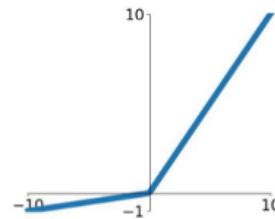
**ReLU**

$$\max(0, x)$$



**Leaky ReLU**

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



# Activation Functions

- Main problem with Sigmoid and tanh activation

## Vanishing gradient problem

Issue of exceeding small gradients when training neural networks

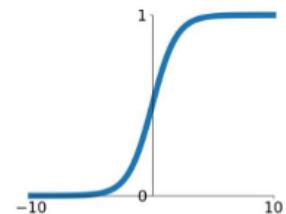
Worse with multiple layers in a NN

## Additional problem with Sigmoid

Slower convergence than tanh

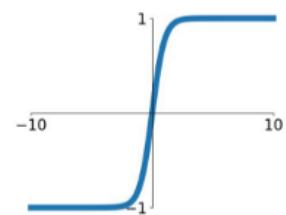
### Sigmoid

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



### tanh

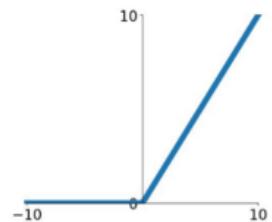
$$\tanh(x)$$



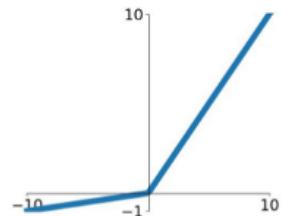
# Activation Functions

- Rectified Linear Units (ReLU)
  - Developed to overcome the vanishing gradient problem
  - However, some neuron may “die”
- Leaky ReLU
  - Prevents neurons from “dying” by using a small negative slope

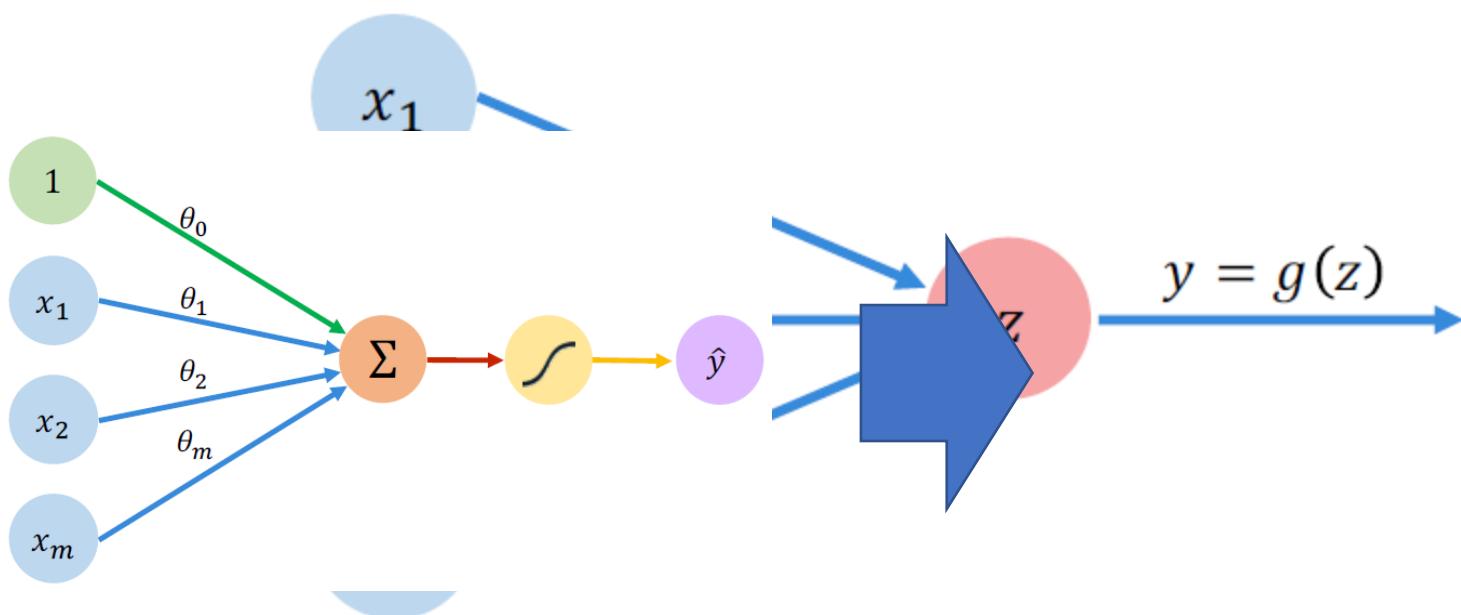
**ReLU**  
 $\max(0, x)$



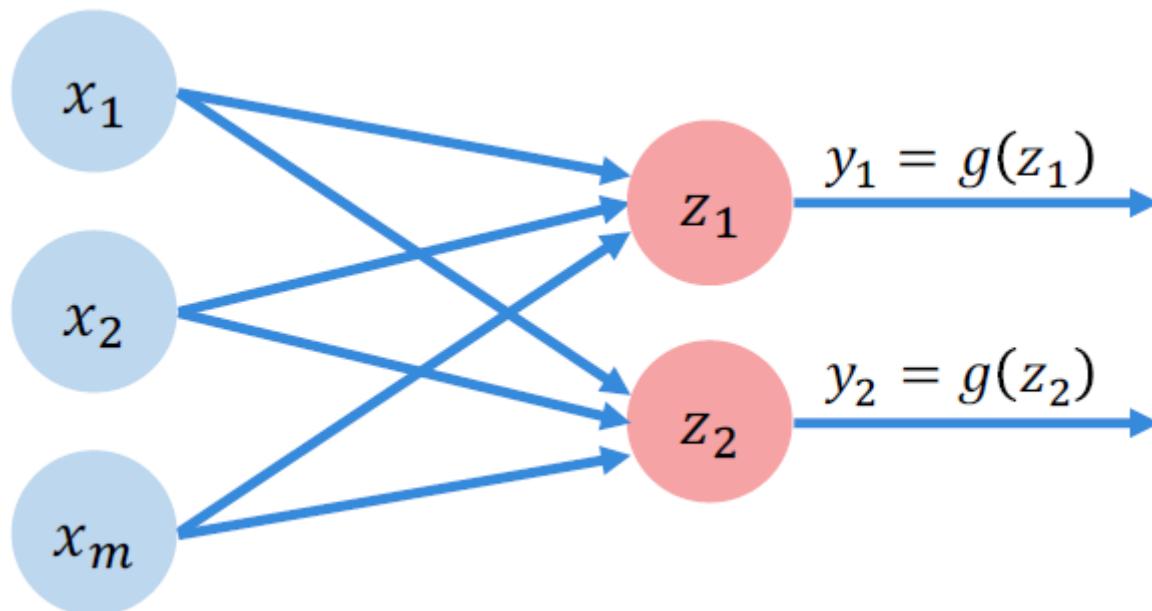
**Leaky ReLU**  
 $\max(0.1x, x)$



# Simplifying the Perceptron

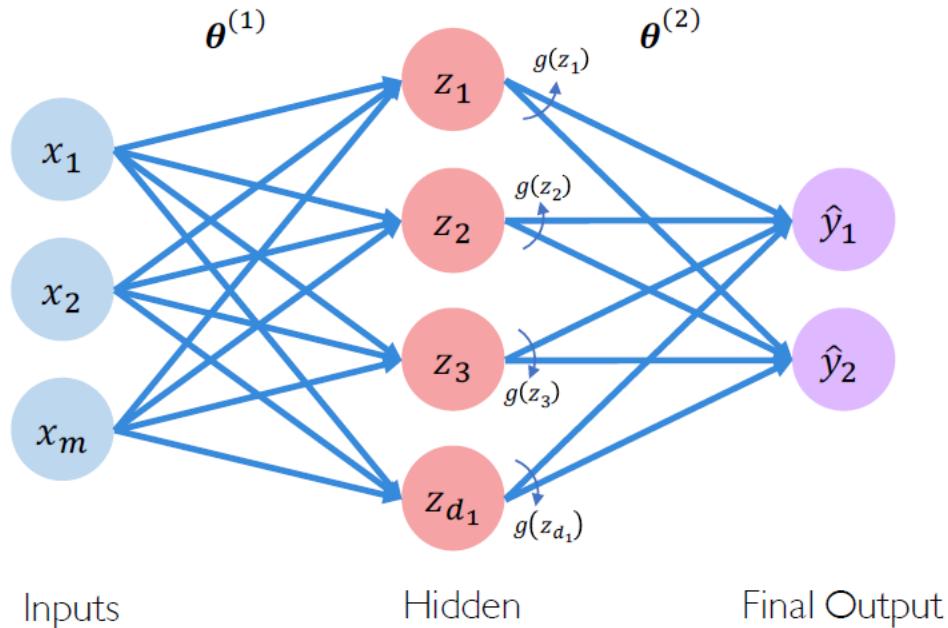


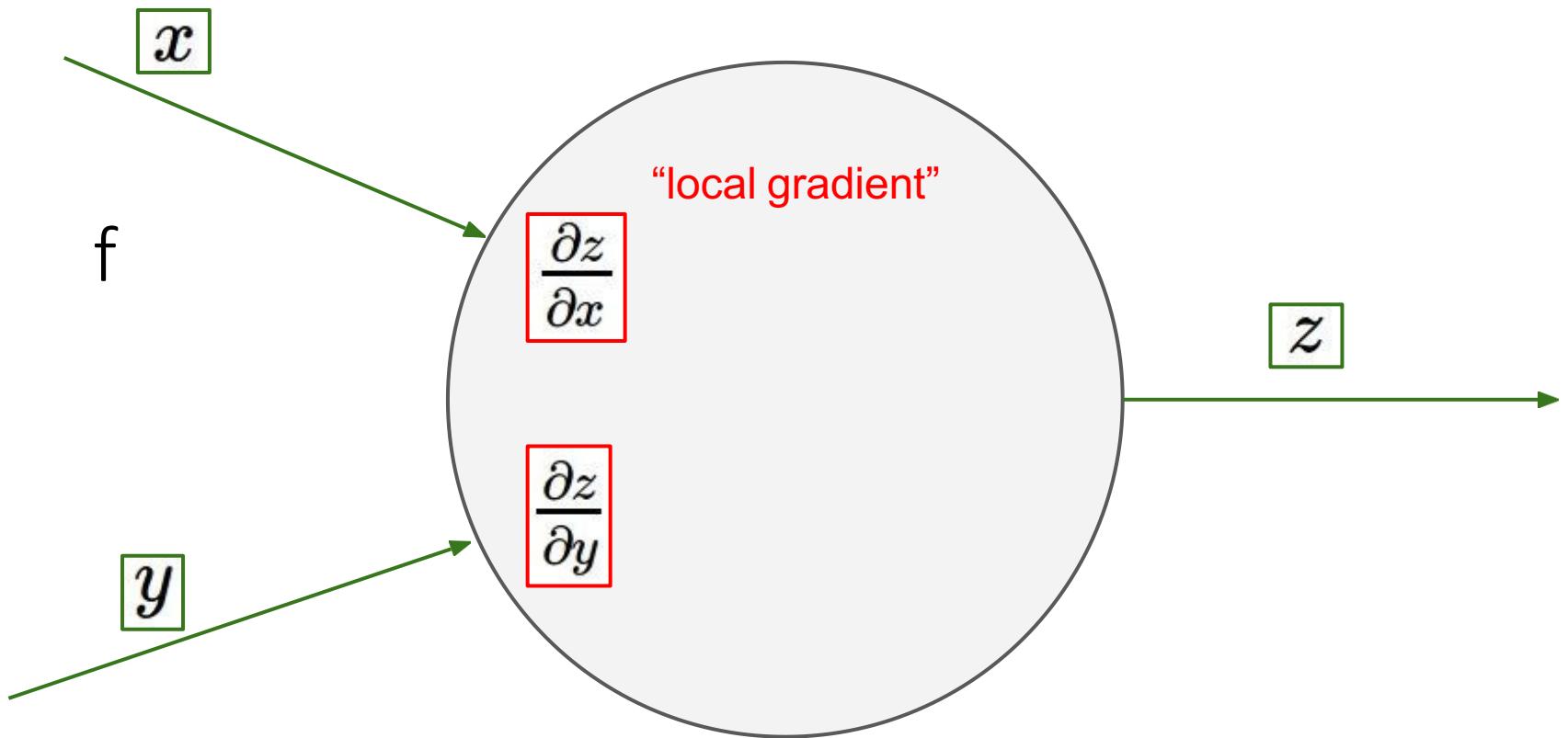
# Multiple Perceptron

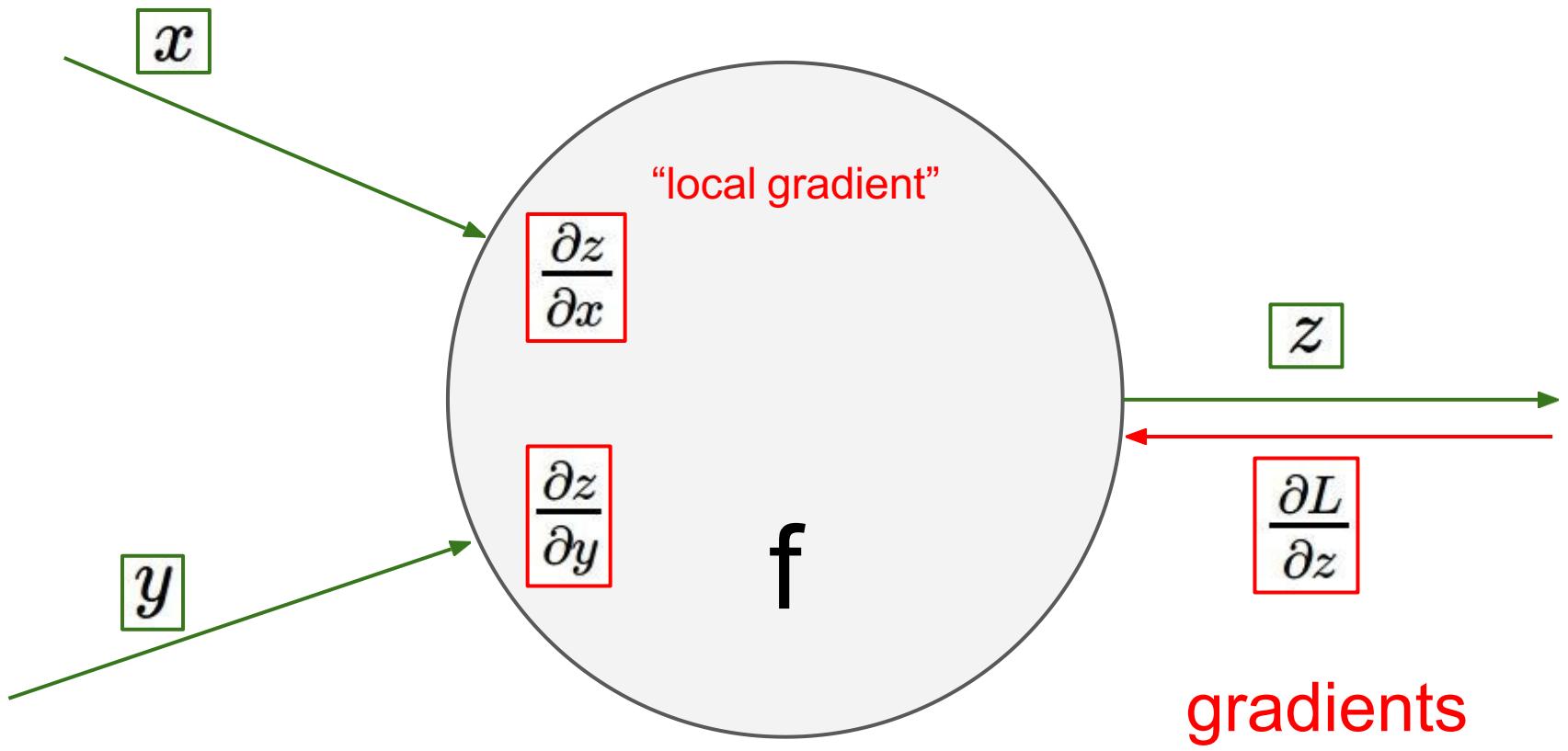


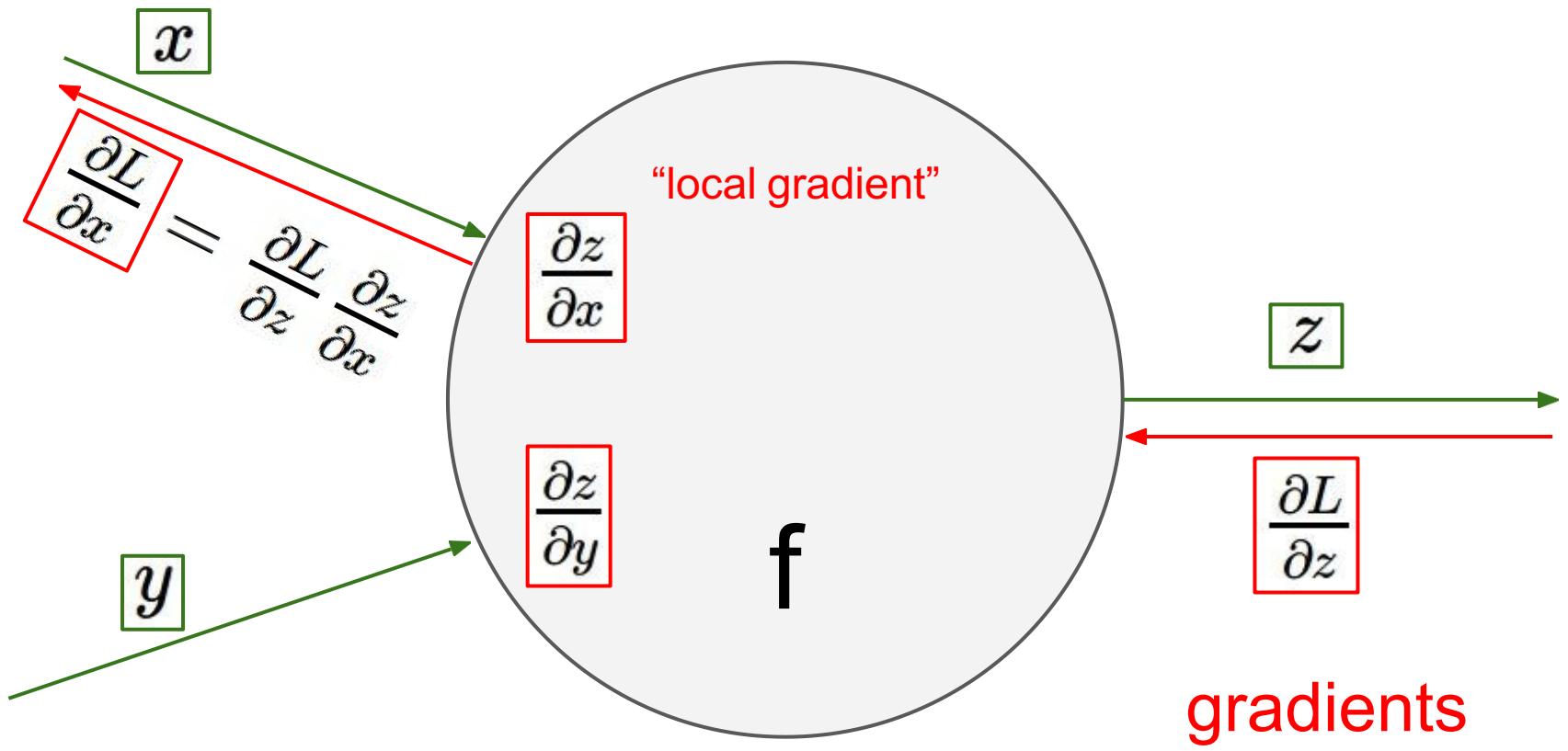
# Neural Network

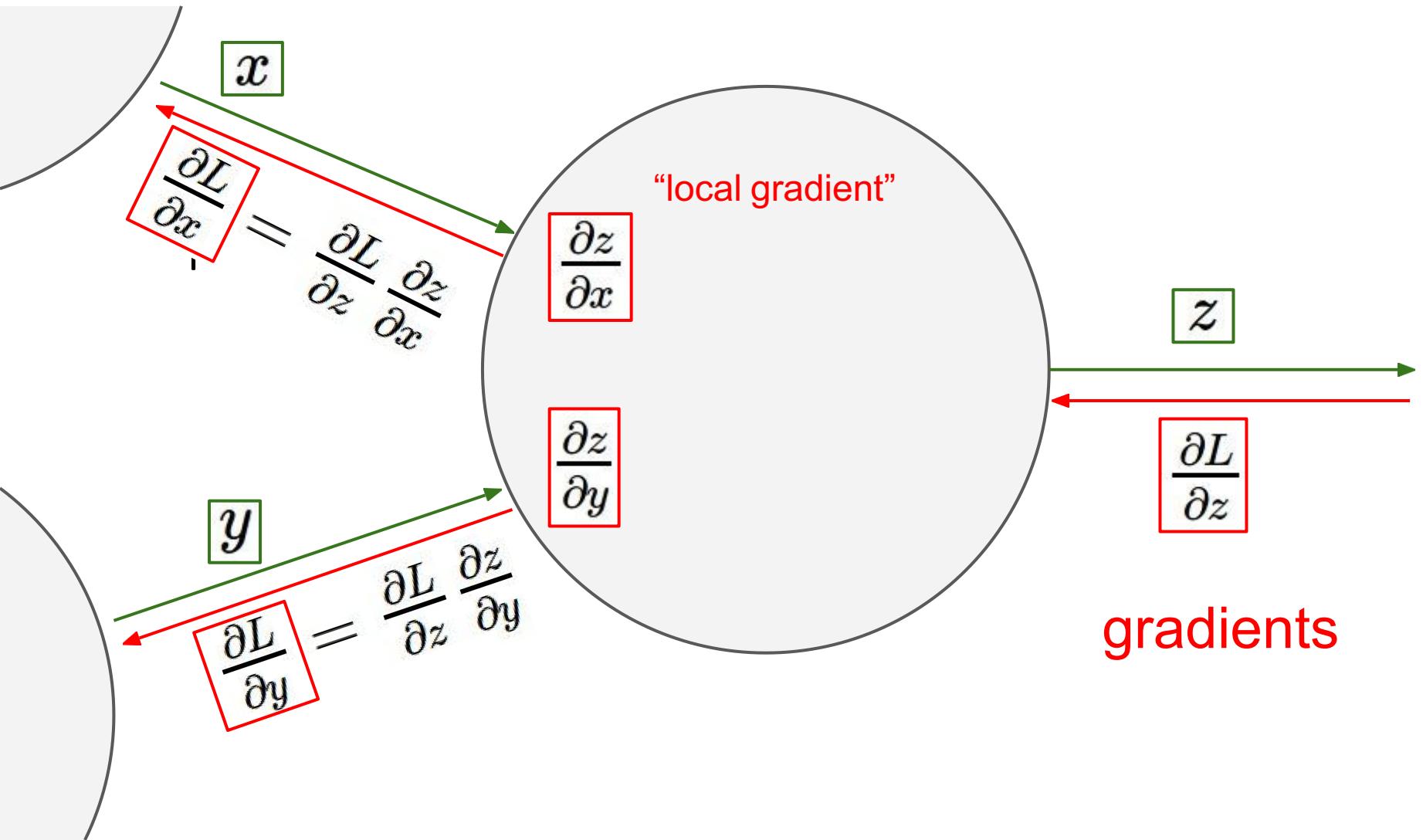
- Multiple perceptrons, aka a Multi-layer Perceptron (MLP)





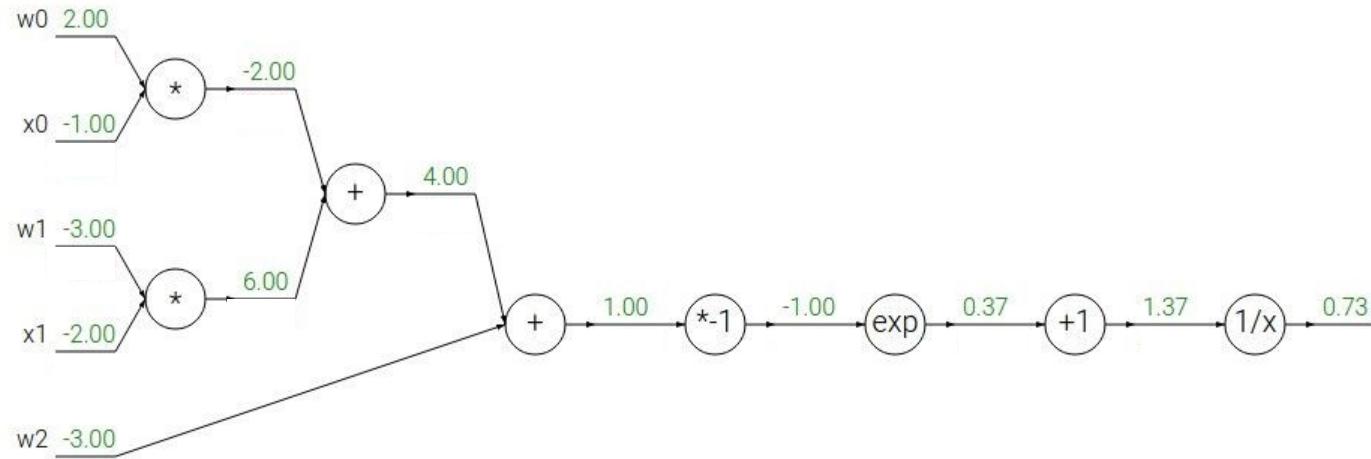






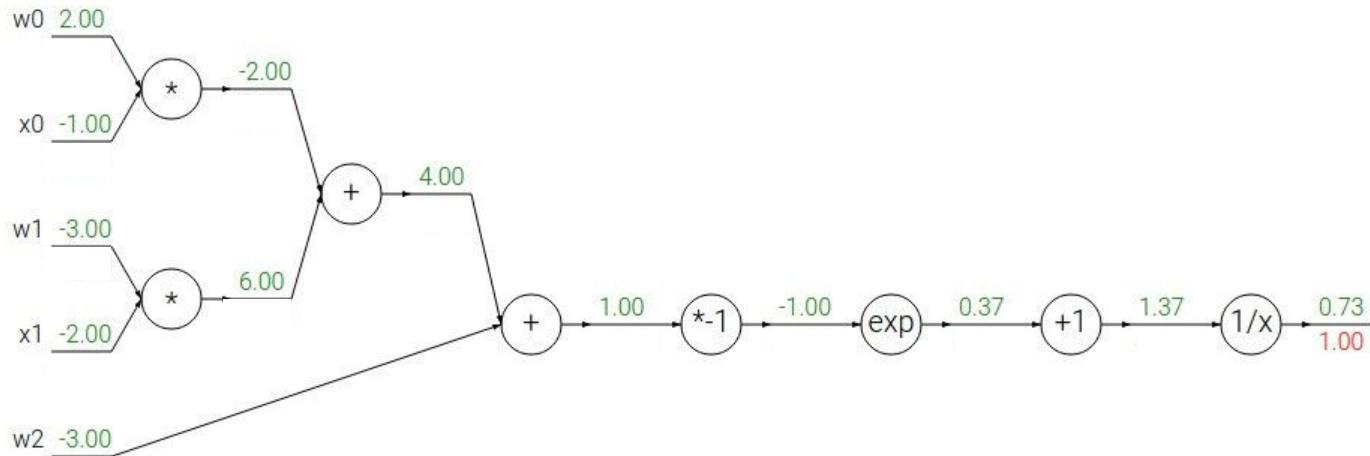
Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

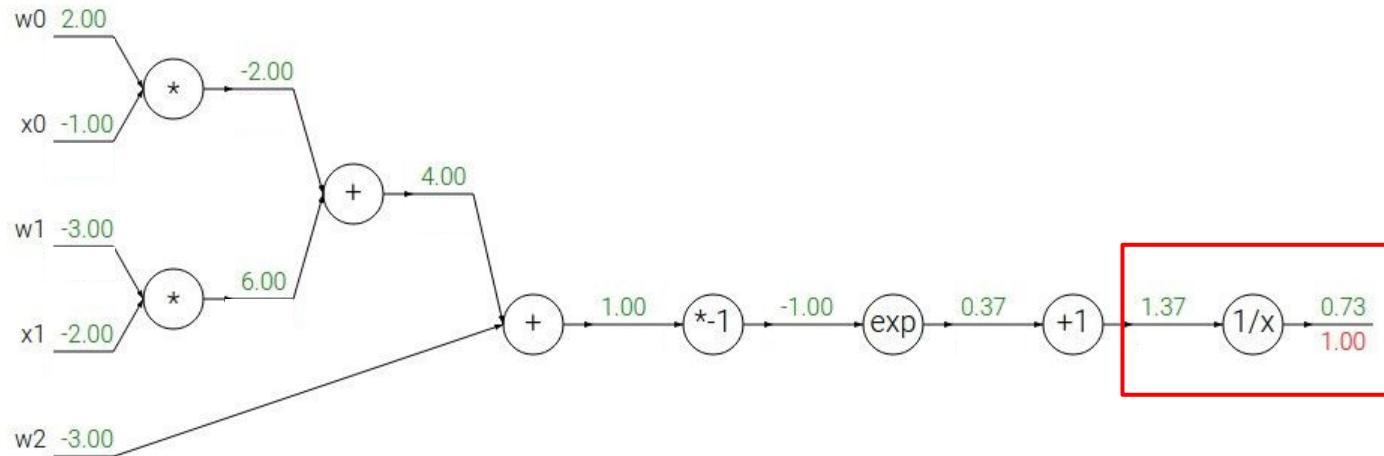
$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

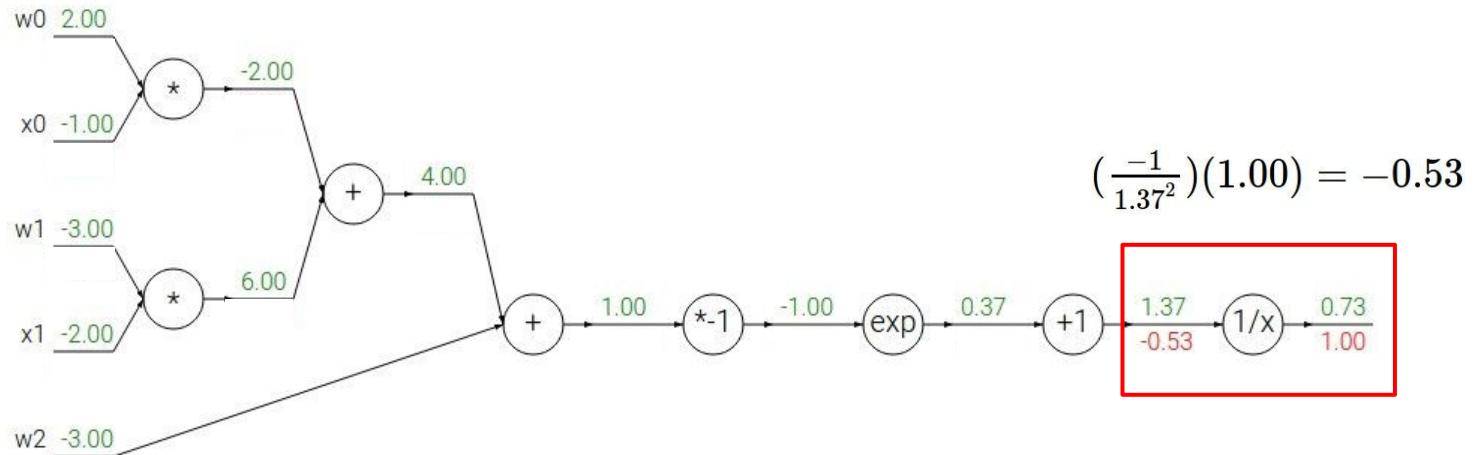
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

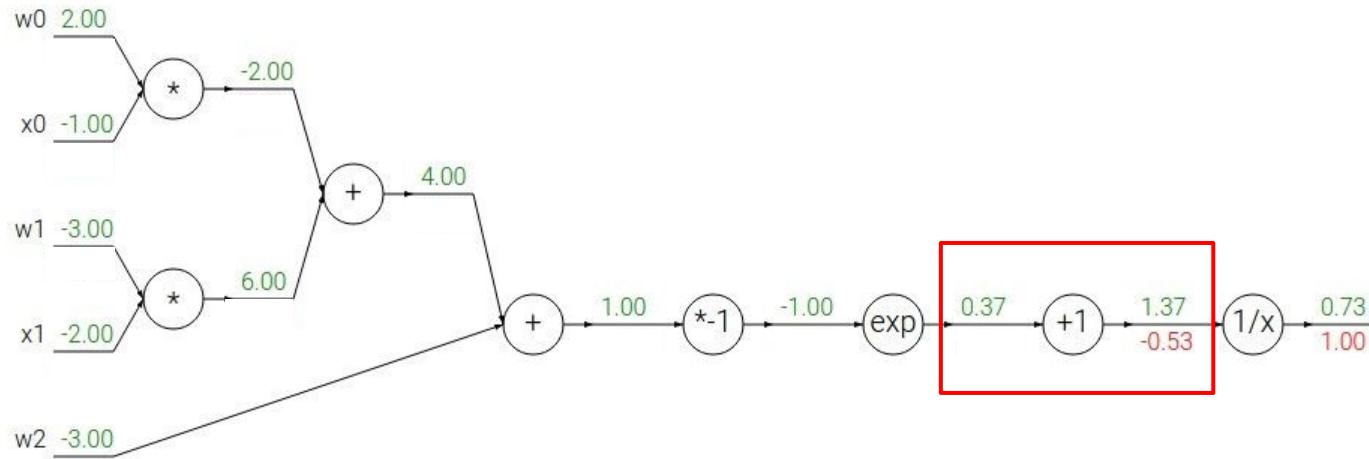
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

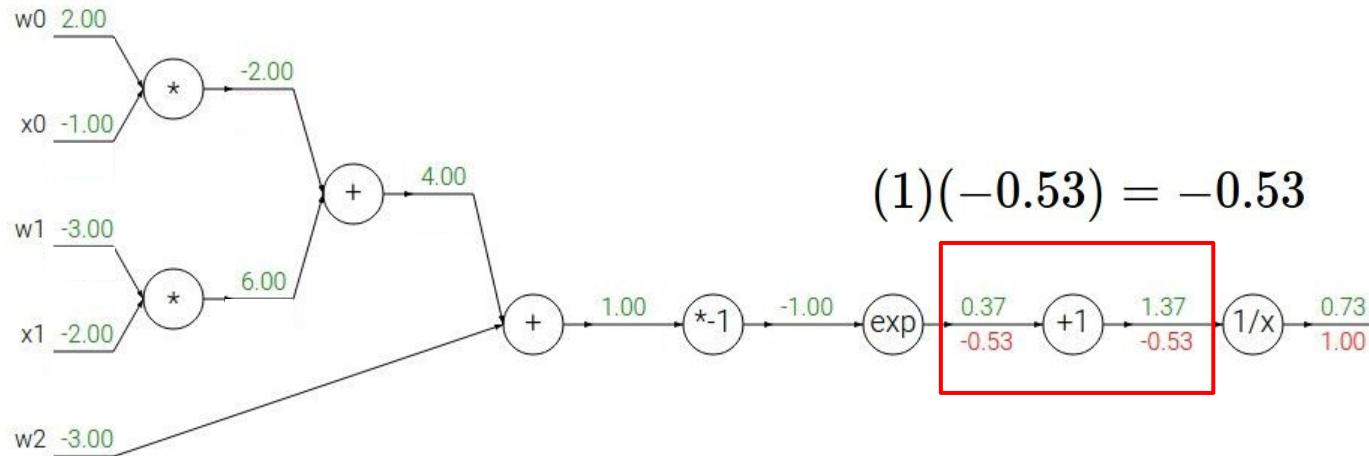
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

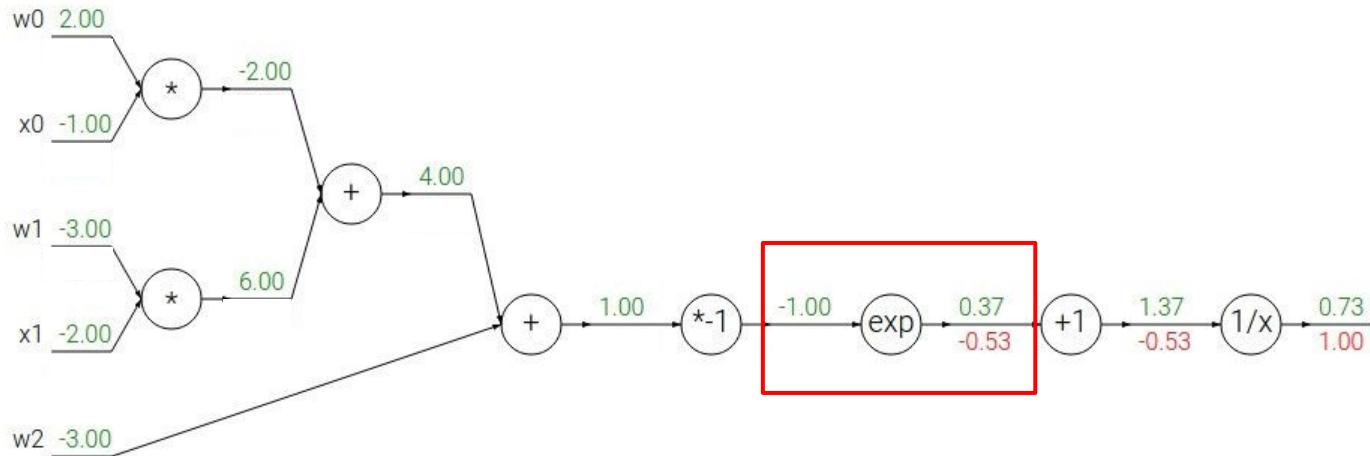
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

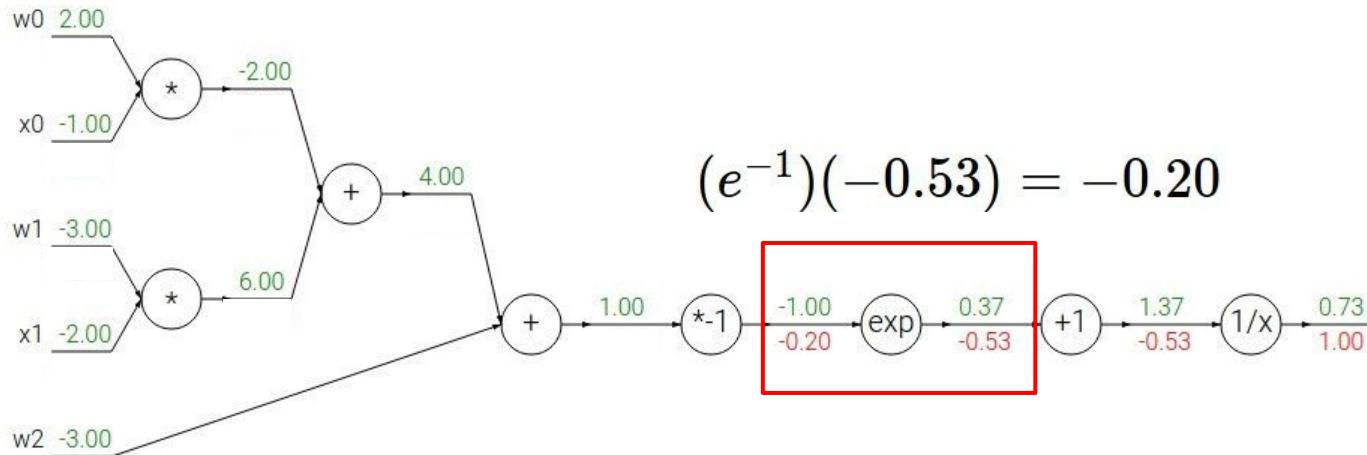
$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

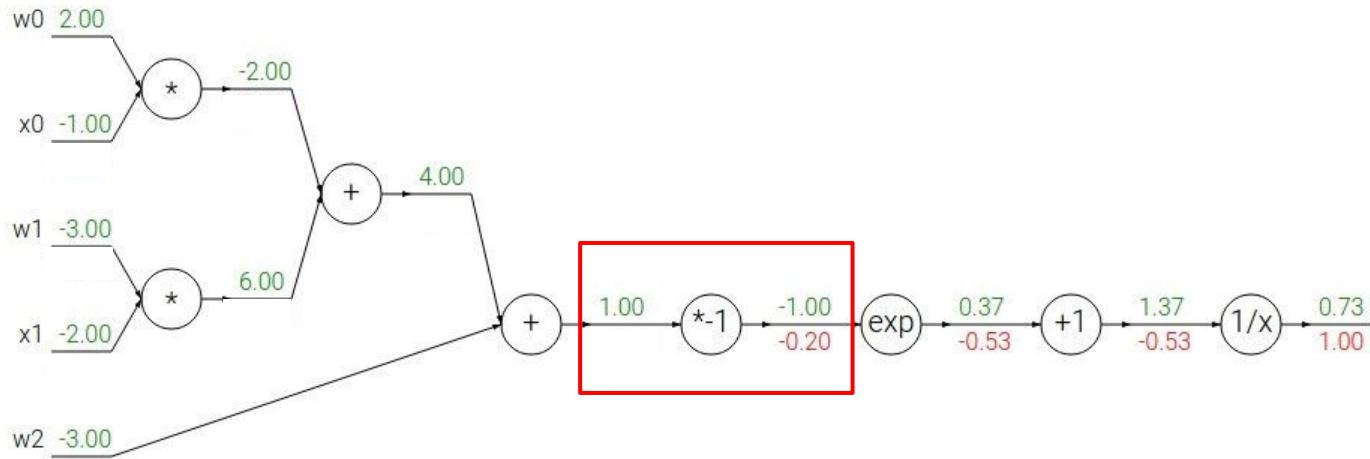
$$f_c(x) = c + x$$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

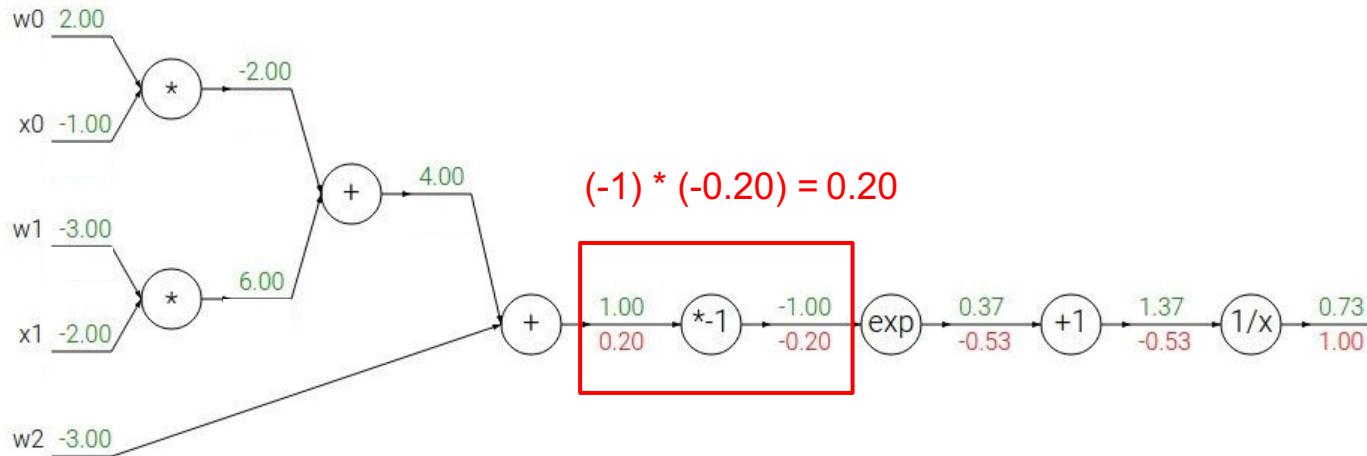
$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

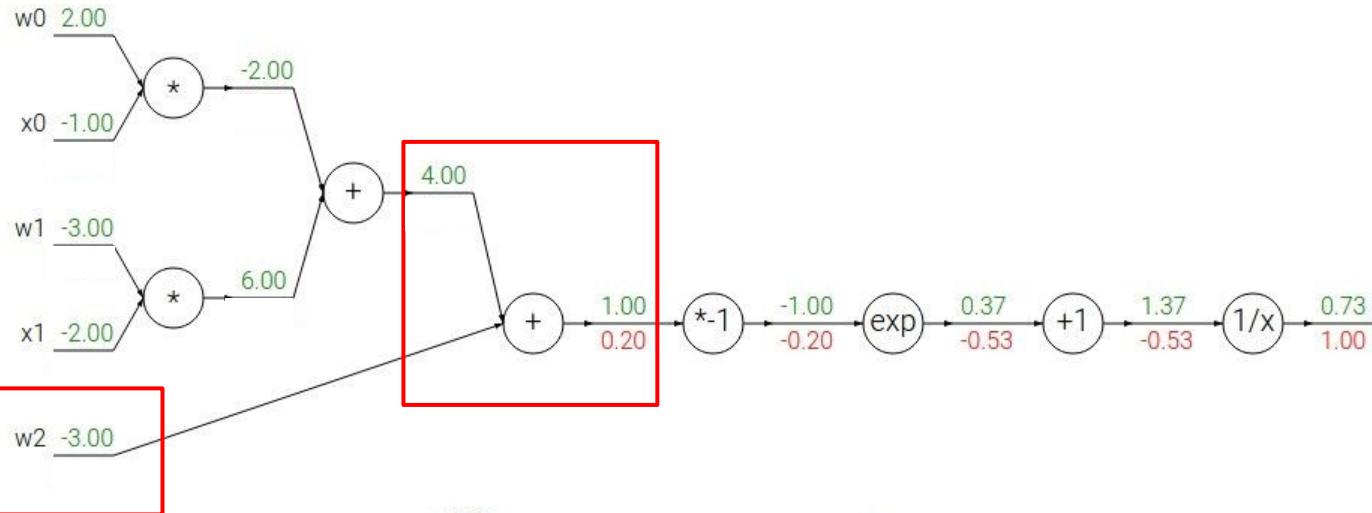
$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

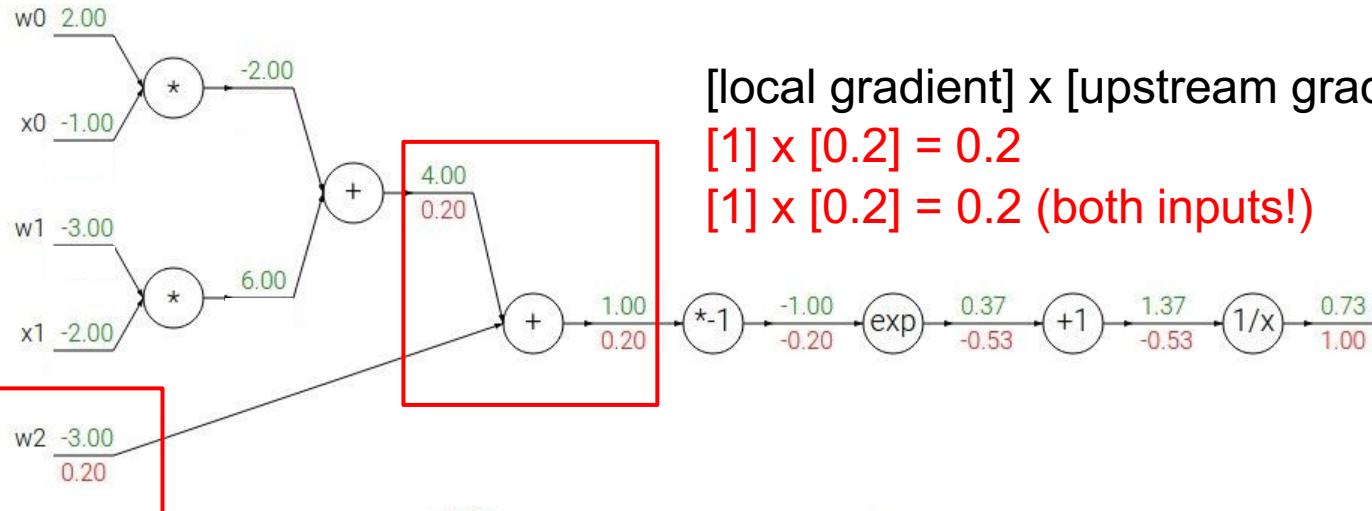
$$f_c(x) = c + x$$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

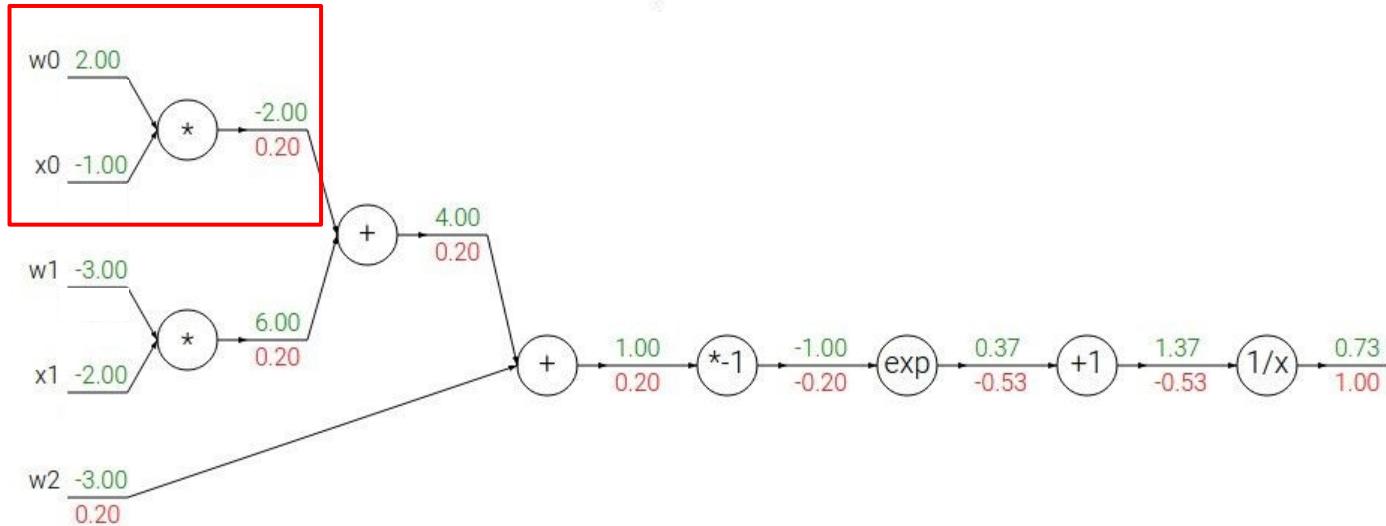
$$f_c(x) = c + x$$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

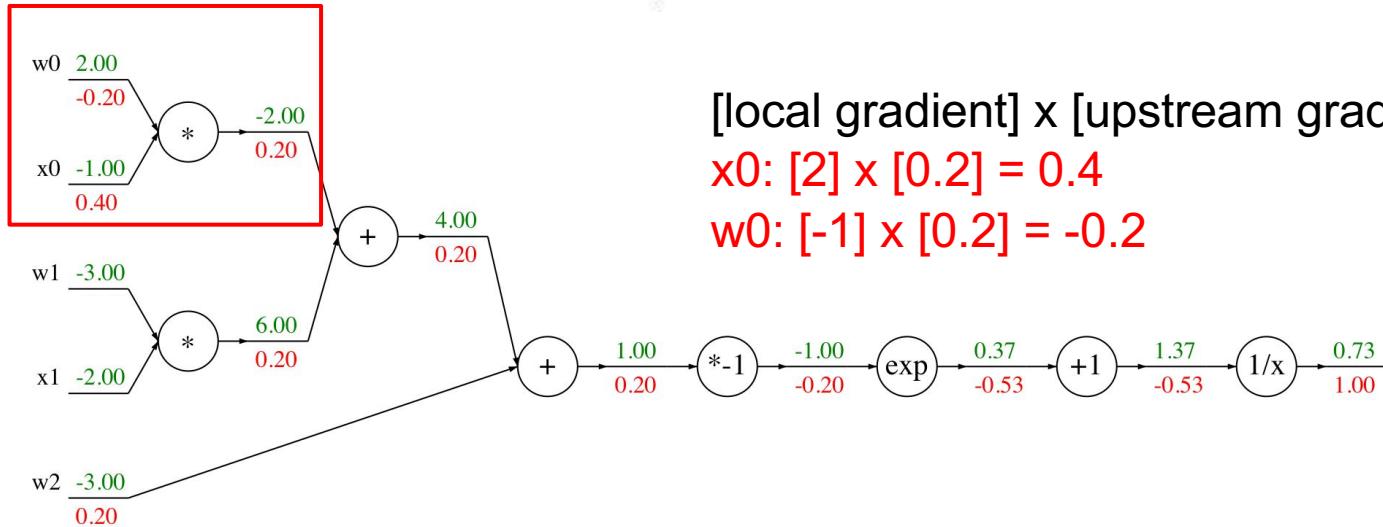
$$f_c(x) = c + x$$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Another example:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



[local gradient] x [upstream gradient]  
 $x_0: [2] \times [0.2] = 0.4$   
 $w_0: [-1] \times [0.2] = -0.2$

$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

$$\frac{df}{dx} = -1/x^2$$

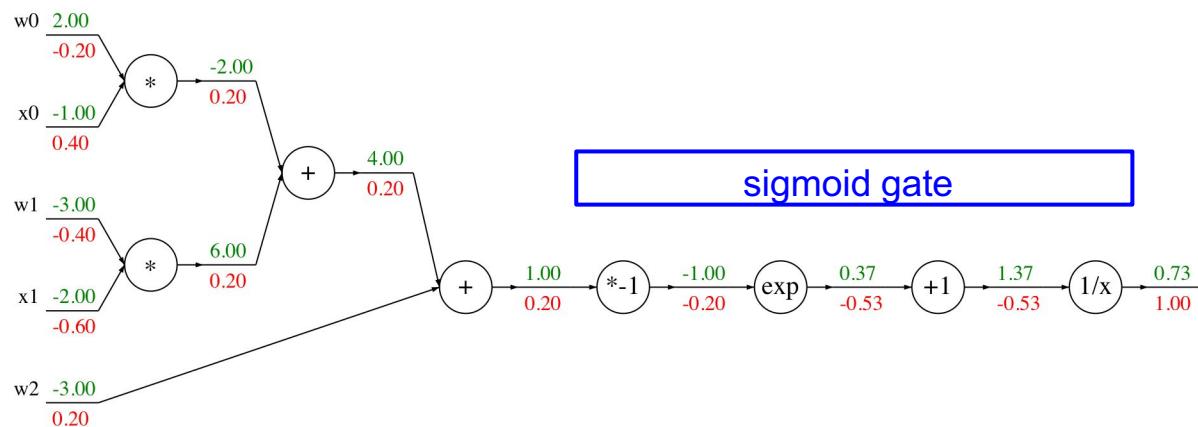
$$\frac{df}{dx} = 1$$

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

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

sigmoid function

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left( \frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left( \frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x))\sigma(x)$$

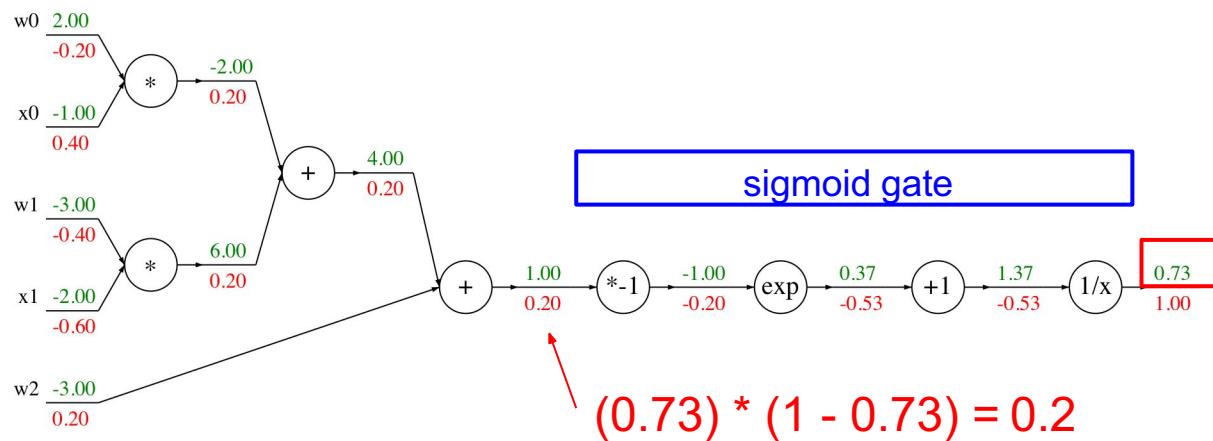


$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

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

sigmoid function

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left( \frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left( \frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x))\sigma(x)$$



# Logistic regression: binary classification

The loss function is -

$$: L(w) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))]$$

$x^{(i)}$  is a vector for all  $x_j$  ( $j=0,1, \dots, n$ ), and  $y^{(i)}$  is the target value for this example.

$$h(x) = \frac{1}{1 + e^{-w^T x}}$$

# Softmax

- Use softmax for multi-class classification
  - K is the number classes

$$P(y = j \mid z^{(i)}) = \phi_{\text{softmax}}(z^{(i)}) = \frac{e^{z^{(i)}}}{\sum_{j=0}^k e^{z_j^{(i)}}},$$

where we define the net input  $z$  as

$$z = w_0x_0 + w_1x_1 + \dots + w_mx_m = \sum_{l=0}^m w_lx_l = \mathbf{w}^T \mathbf{x}.$$

The loss function is  $H(y, p) = - \sum_i y_i \log(p_i)$

# Softmax vs Sigmoid function in Logistic classifier?

- In the two-class logistic regression, the predicted probabilities are as follows, using the sigmoid function:

$$\Pr(Y_i = 0) = \frac{e^{-\beta \cdot \mathbf{X}_i}}{1 + e^{-\beta_0 \cdot \mathbf{X}_i}}$$

$$\Pr(Y_i = 1) = 1 - \Pr(Y_i = 0) = \frac{1}{1 + e^{-\beta \cdot \mathbf{X}_i}}$$

In the multiclass logistic regression, with  $K$  classes, the predicted probabilities are as follows, using the softmax function:

$$\Pr(Y_i = k) = \frac{e^{\beta_k \cdot \mathbf{X}_i}}{\sum_{0 \leq c \leq K} e^{\beta_c \cdot \mathbf{X}_i}}$$

# Softmax vs Sigmoid function in Logistic classifier?

- One can observe that the softmax function is an extension of the sigmoid function to the multiclass case, as explained below. Let's look at the multiclass logistic regression, with K=2 classes:

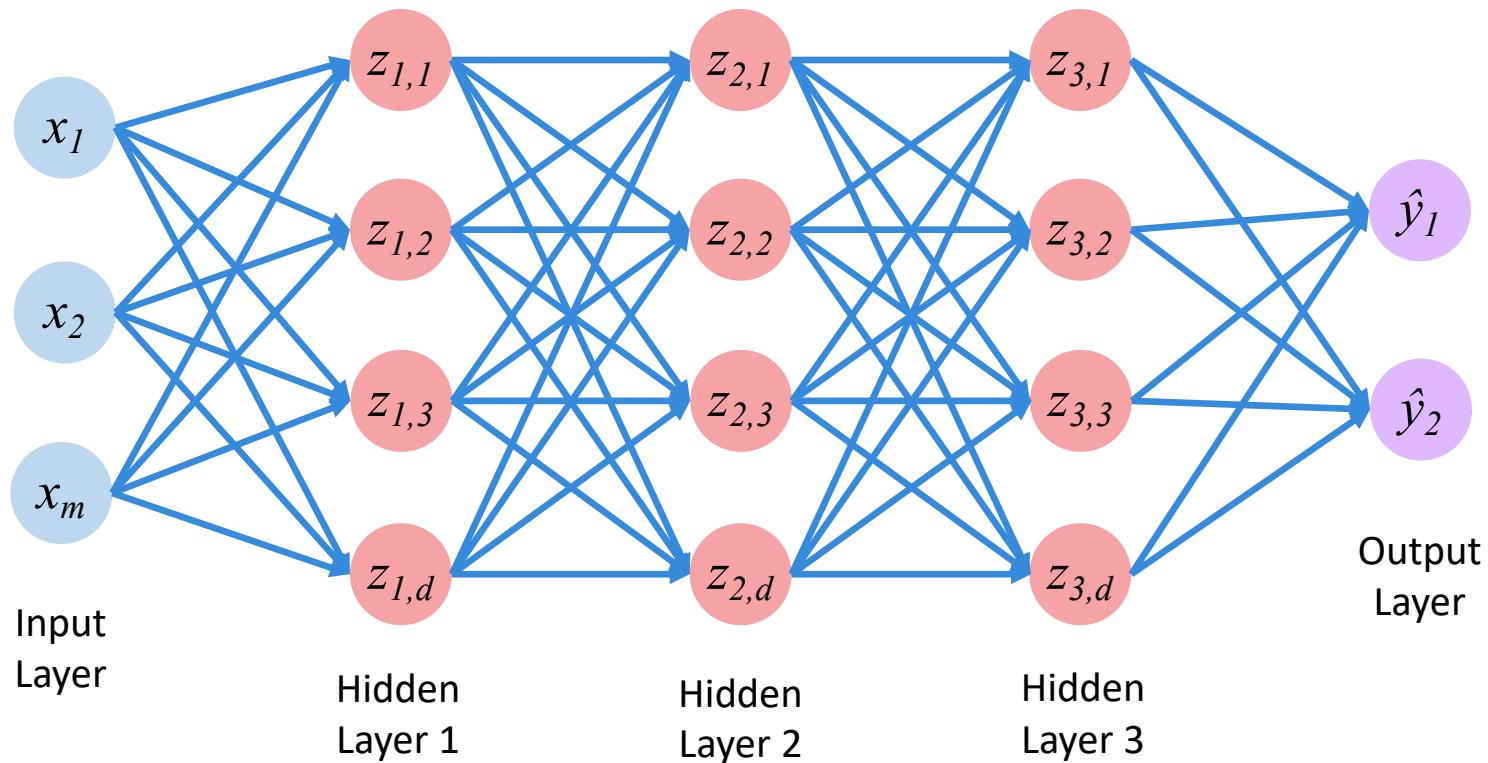
$$\Pr(Y_i = 0) = \frac{e^{\beta_0 \cdot \mathbf{X}_i}}{\sum_{0 \leq c \leq K} e^{\beta_c \cdot \mathbf{X}_i}} = \frac{e^{\beta_0 \cdot \mathbf{X}_i}}{e^{\beta_0 \cdot \mathbf{X}_i} + e^{\beta_1 \cdot \mathbf{X}_i}} = \frac{e^{(\beta_0 - \beta_1) \cdot \mathbf{X}_i}}{e^{(\beta_0 - \beta_1) \cdot \mathbf{X}_i} + 1} = \frac{e^{-\beta \cdot \mathbf{X}_i}}{1 + e^{-\beta \cdot \mathbf{X}_i}}$$

$$\Pr(Y_i = 1) = \frac{e^{\beta_1 \cdot \mathbf{X}_i}}{\sum_{0 \leq c \leq K} e^{\beta_c \cdot \mathbf{X}_i}} = \frac{e^{\beta_1 \cdot \mathbf{X}_i}}{e^{\beta_0 \cdot \mathbf{X}_i} + e^{\beta_1 \cdot \mathbf{X}_i}} = \frac{1}{e^{(\beta_0 - \beta_1) \cdot \mathbf{X}_i} + 1} = \frac{1}{1 + e^{-\beta \cdot \mathbf{X}_i}}$$

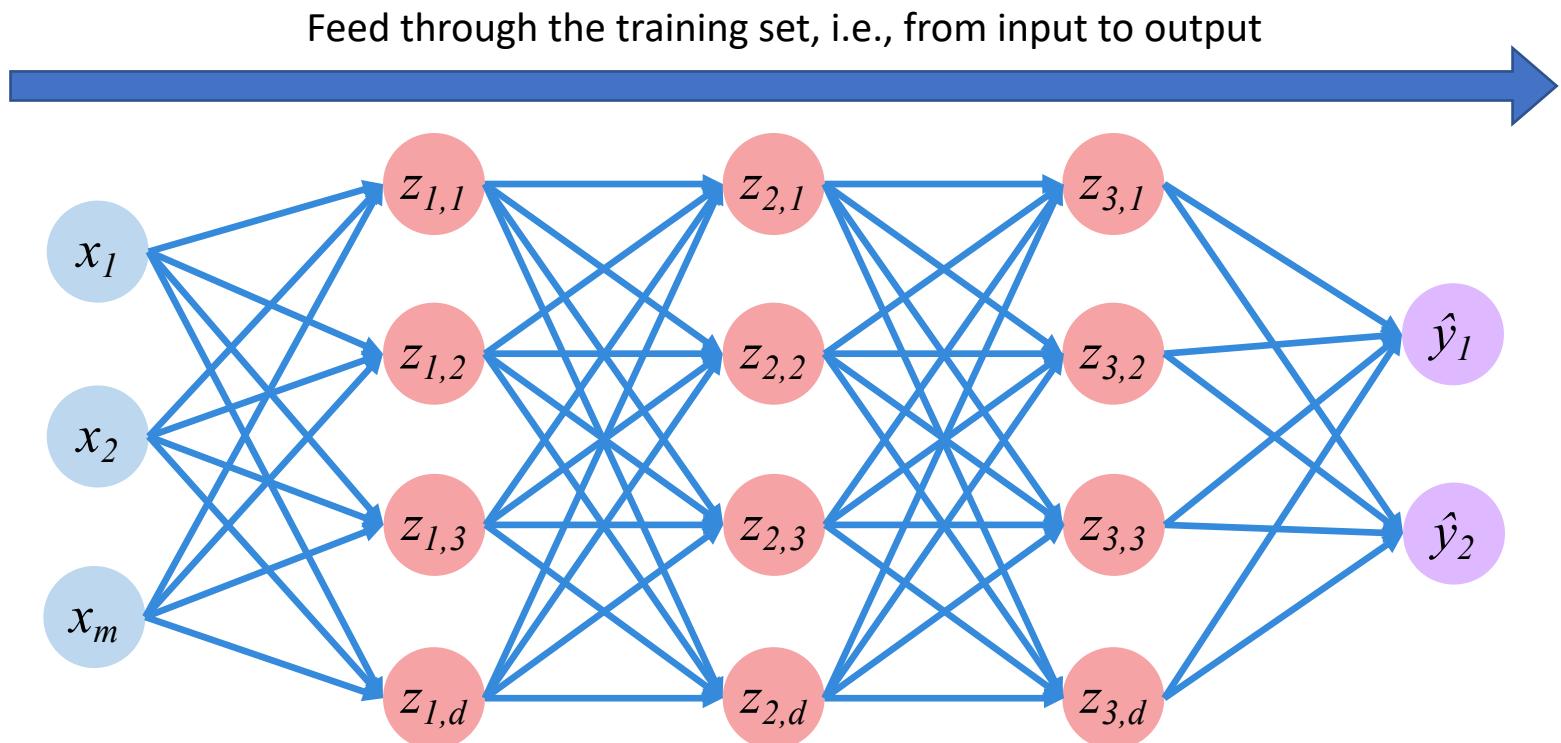
where  $\beta = -(\beta_0 - \beta_1)$

# Deep Neural Network

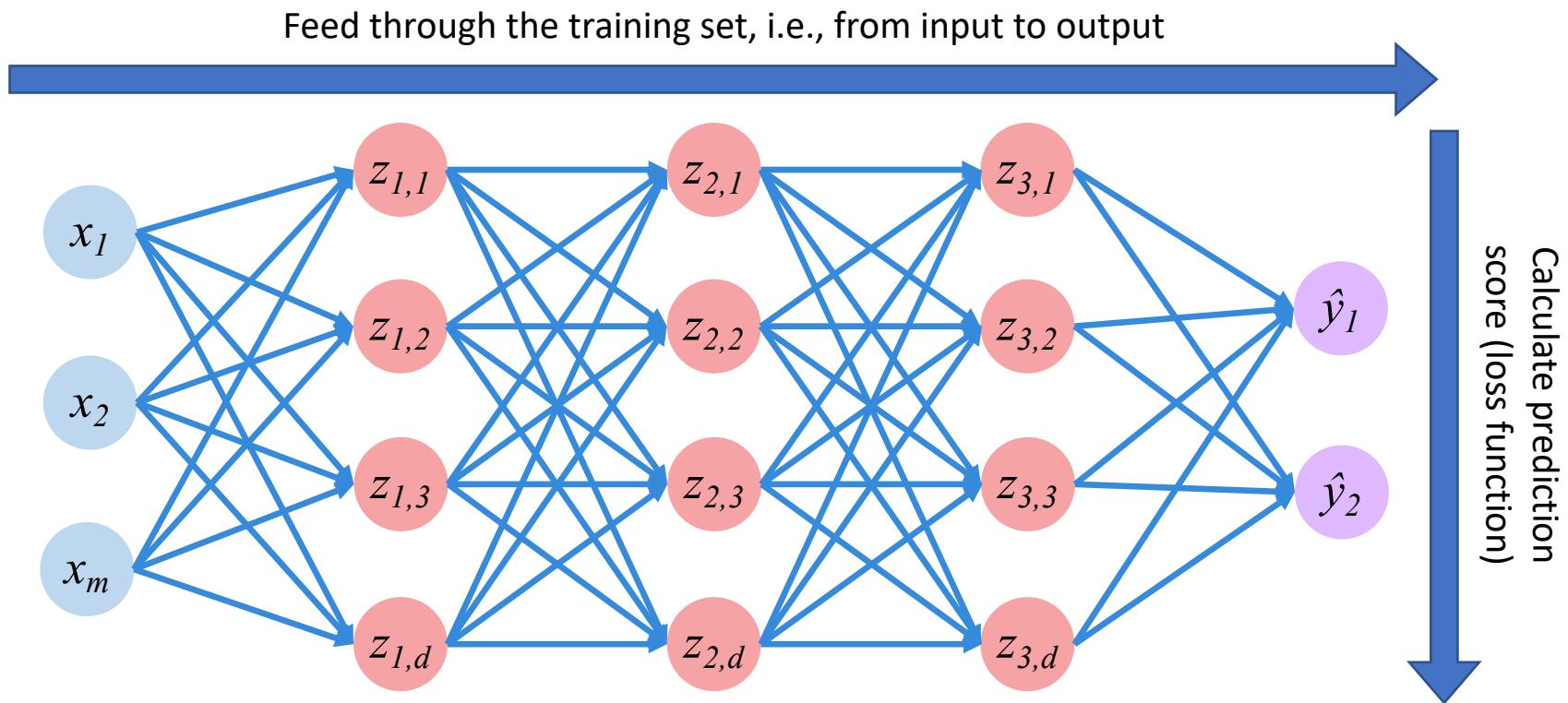
- Adding on multiple hidden layers



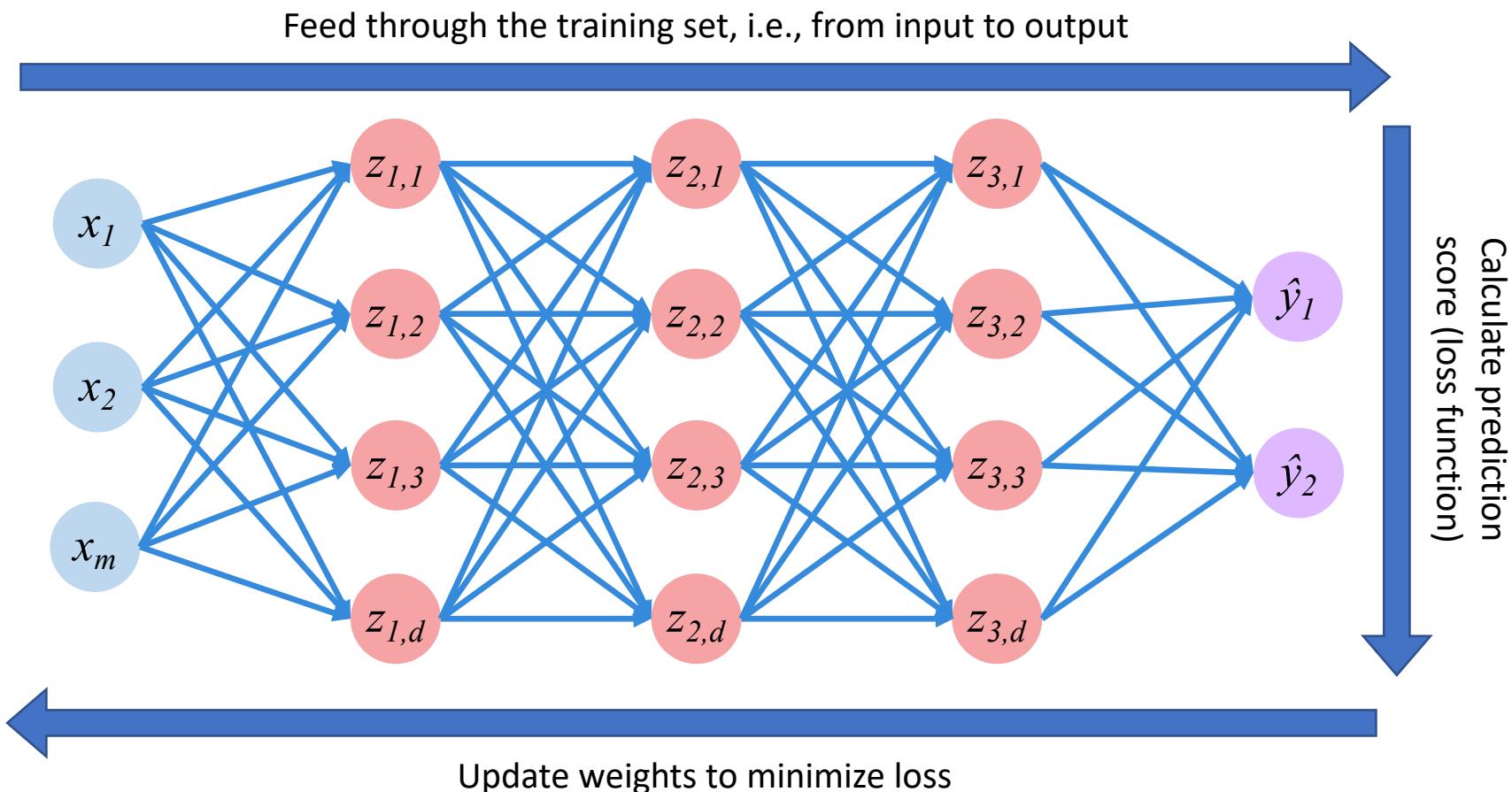
# Training a Neural Network



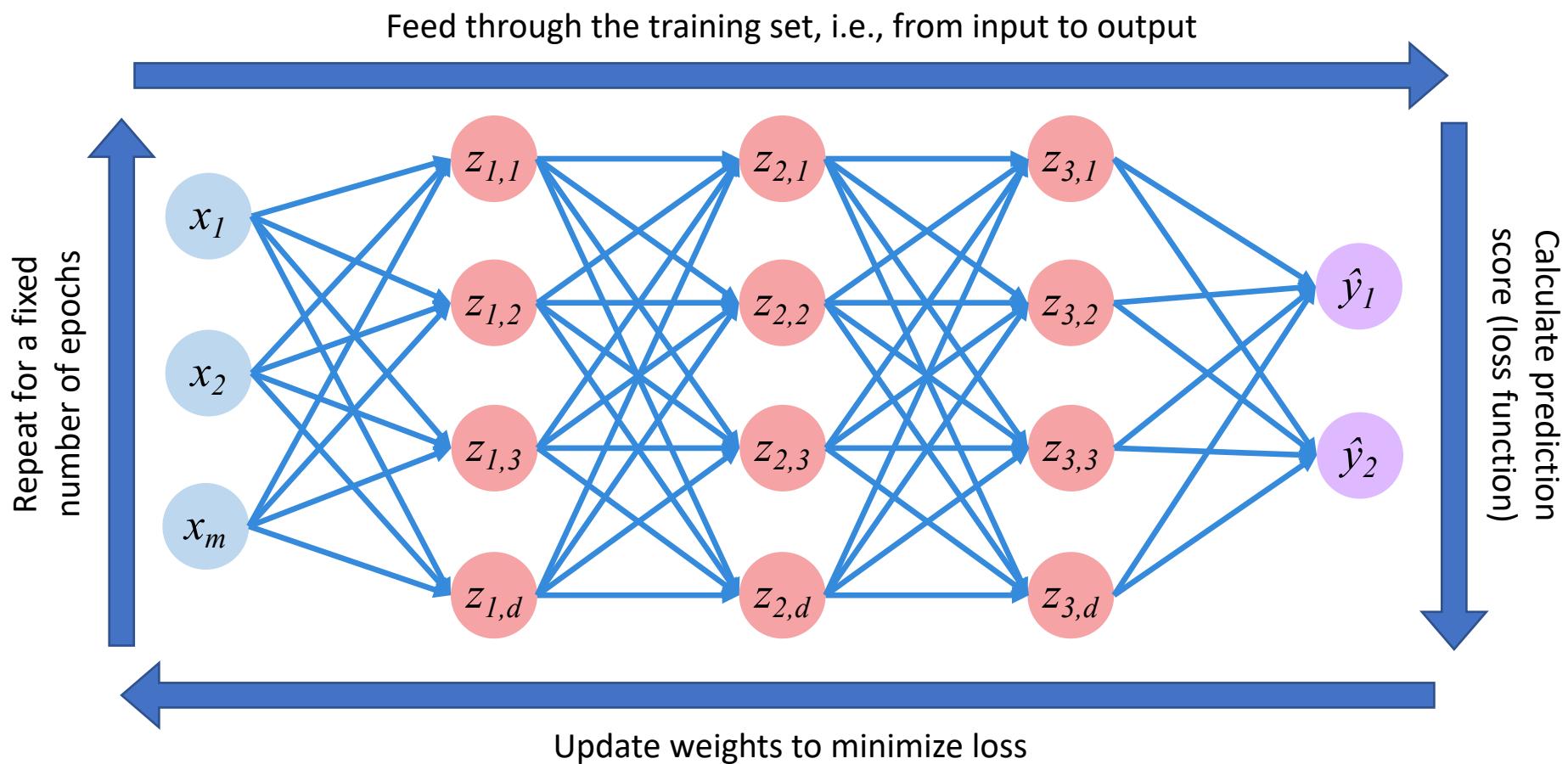
# Training a Neural Network



# Training a Neural Network



# Training a Neural Network



# Training a Neural Network

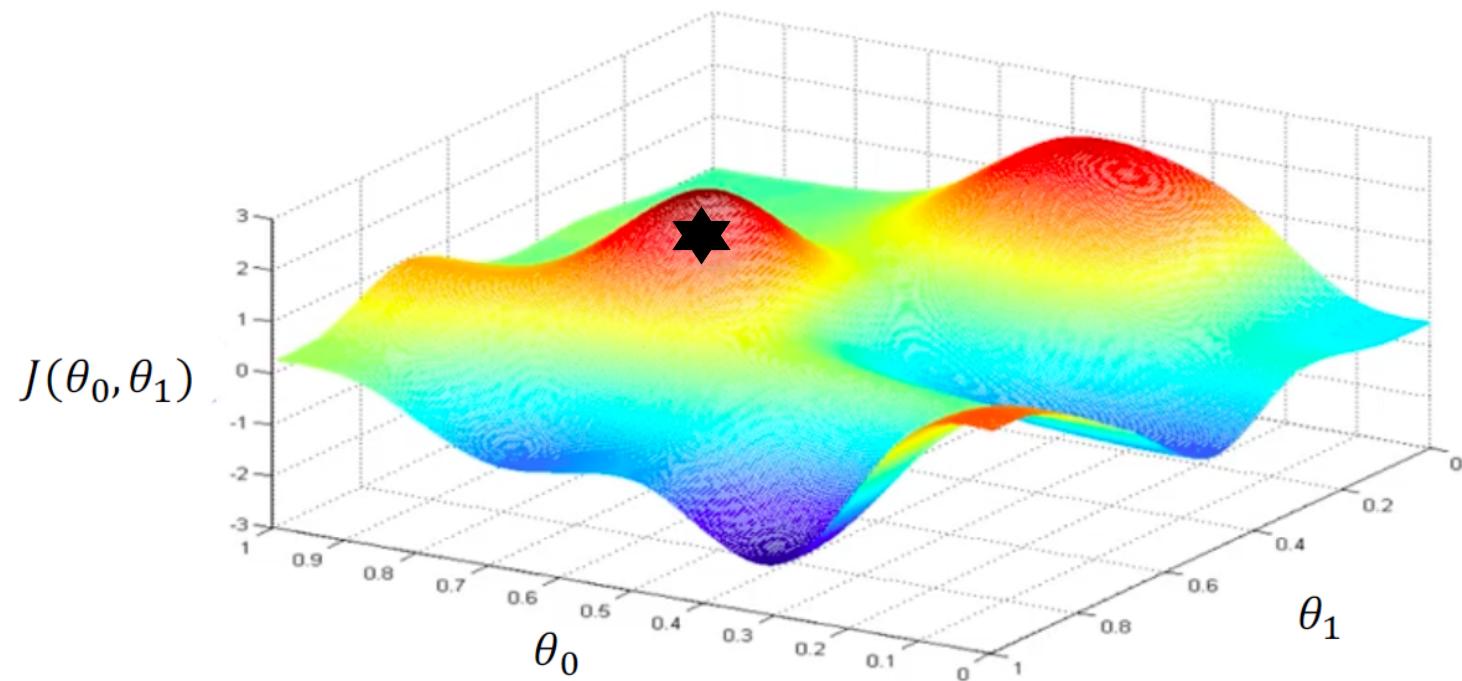
- What type of loss function?
- How to learn and update weights?
- How much training data to use?

# Types of Loss Functions

- For Regression tasks (continuous values)
  - Mean Absolute Error
  - Mean Squared Error
- For Classification tasks (discrete categories)
  - Categorical Cross Entropy
  - Binary Cross Entropy
- And various others in Keras

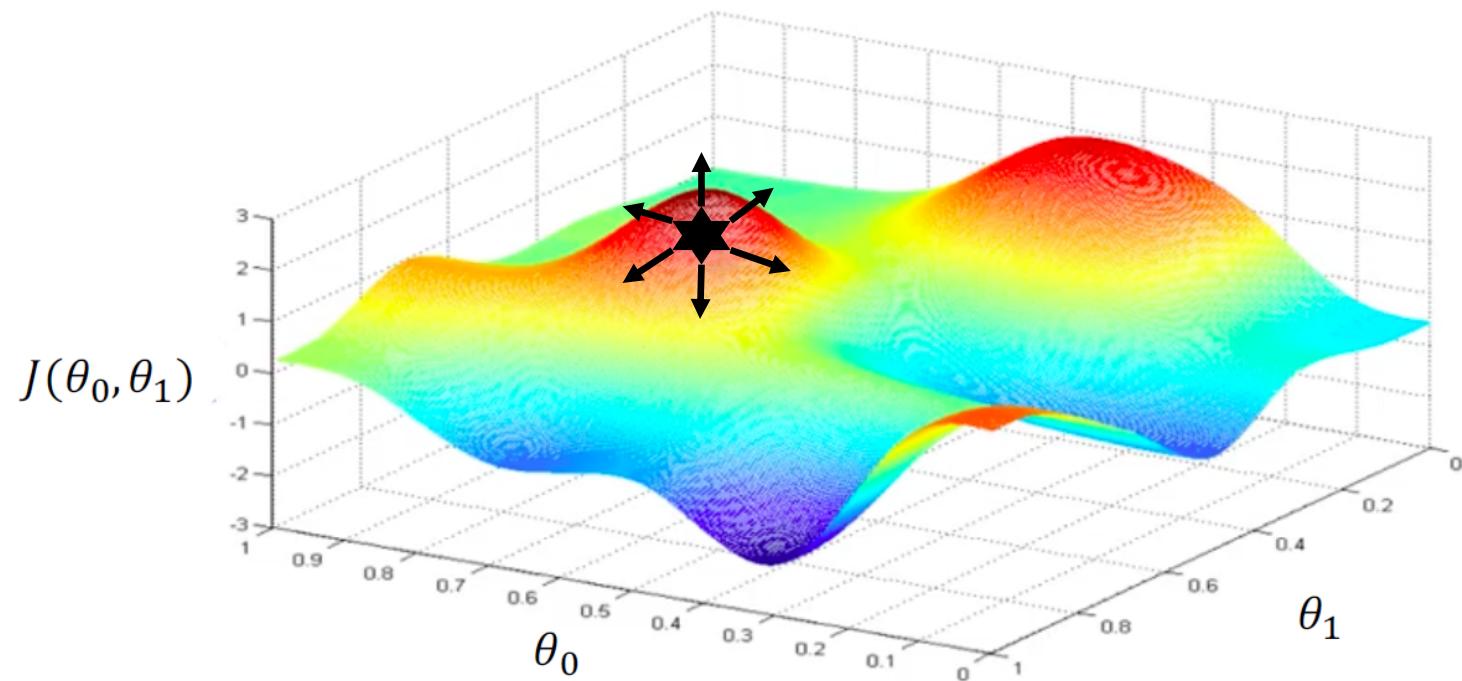
# Gradient Descent

- Standard approach for learning weights



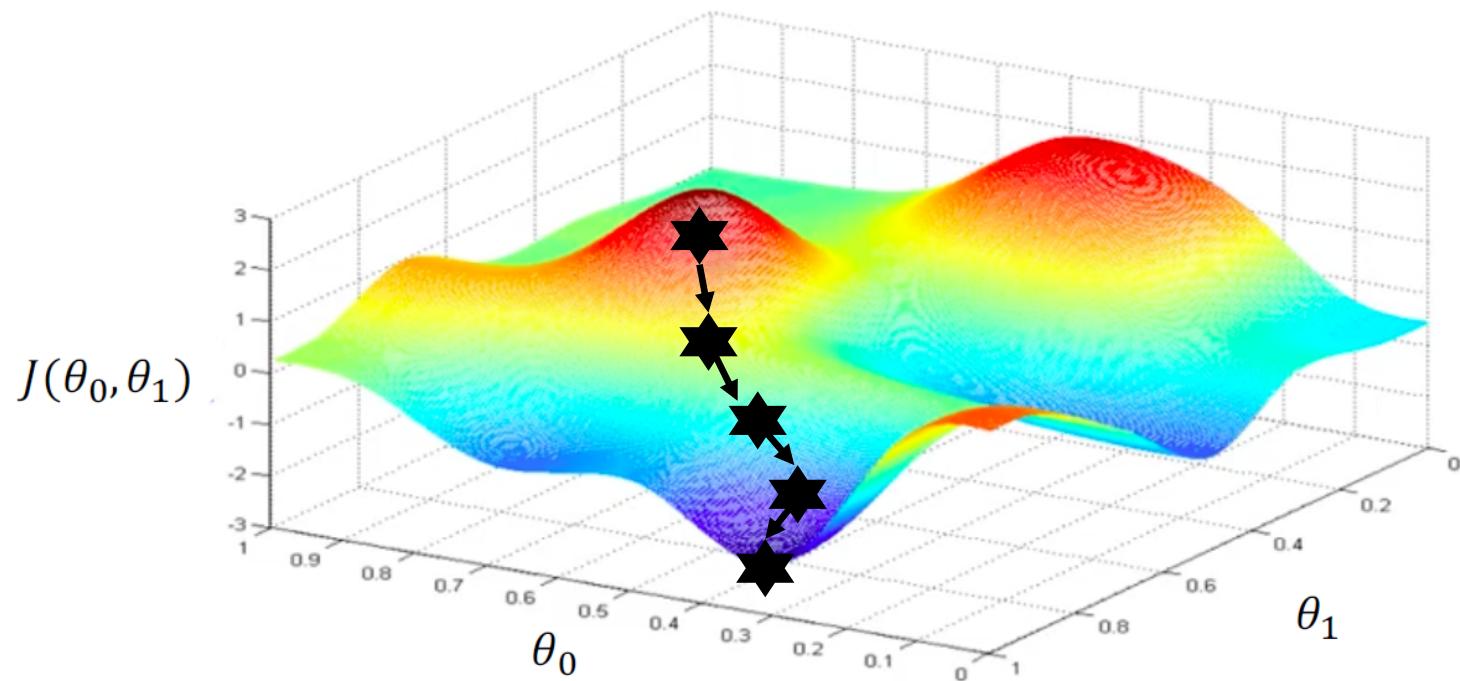
# Gradient Descent

- Standard approach for learning weights



# Gradient Descent

- Standard approach for learning weights
  - What about the learning rate?



# Exercise 1: Gradient Descent

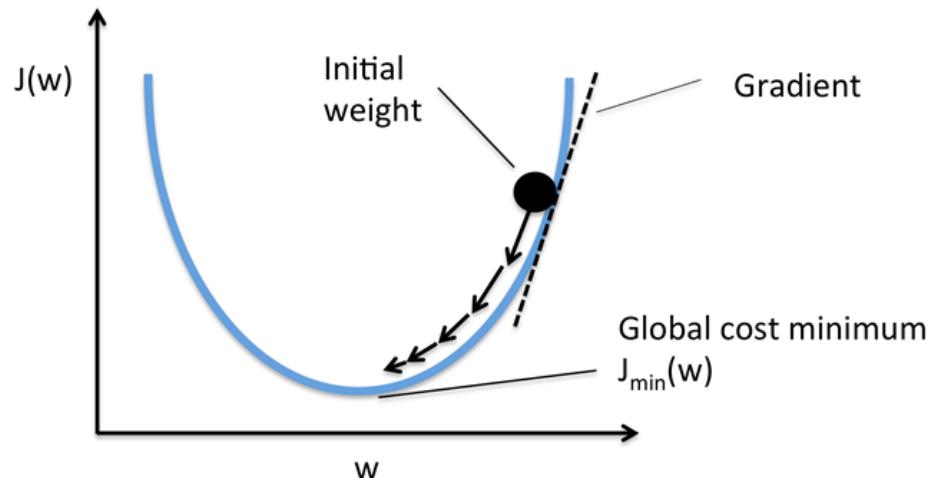
- What are the effects of learning rates on gradient descent?

# Mini-batch Training

- Batch gradient descent (BGD)
  - Compute weights using entire training set
- Stochastic gradient descent (SGD)
  - Compute weights using an instance at a time
- Mini-batch gradient descent
  - Compute weights using small batches (e.g., 32) from training set
- Typically, mini-batch is used in training neural networks
  - More robust convergence (avoid local minima) than BGD
  - Computationally more efficient than SGD
  - Parallelization to speed up computation

# Adaptive Learning Rates

- Instead of fixed learning rates, use learning rates that change based on:
  - Value of gradient
  - Speed of learning
  - Sizes of weights

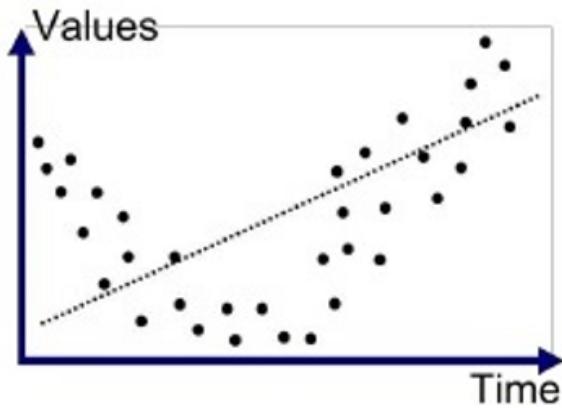


<https://wiki.tum.de/display/Ifdv/Adaptive+Learning+Rate+Method>

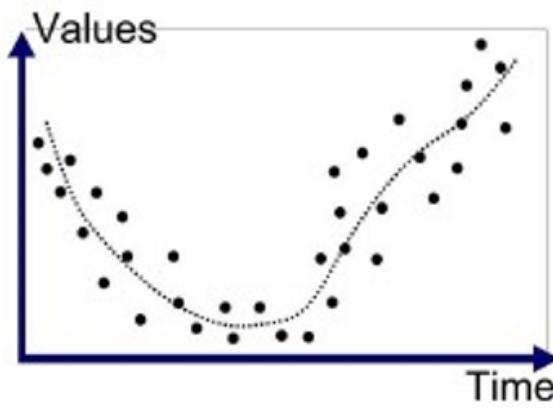
# Adaptive Learning Rates

- Various approaches available in Keras
  - Adagrad
  - Adadelta
  - Adam
  - Nadam
  - RMSprop

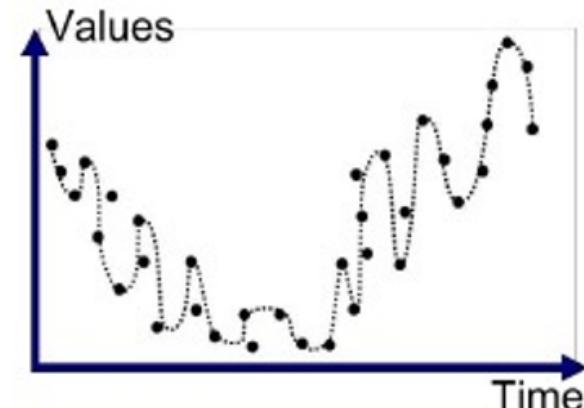
# Under/Over-fitting Problem



Underfitted



Good Fit/R robust



Overfitted

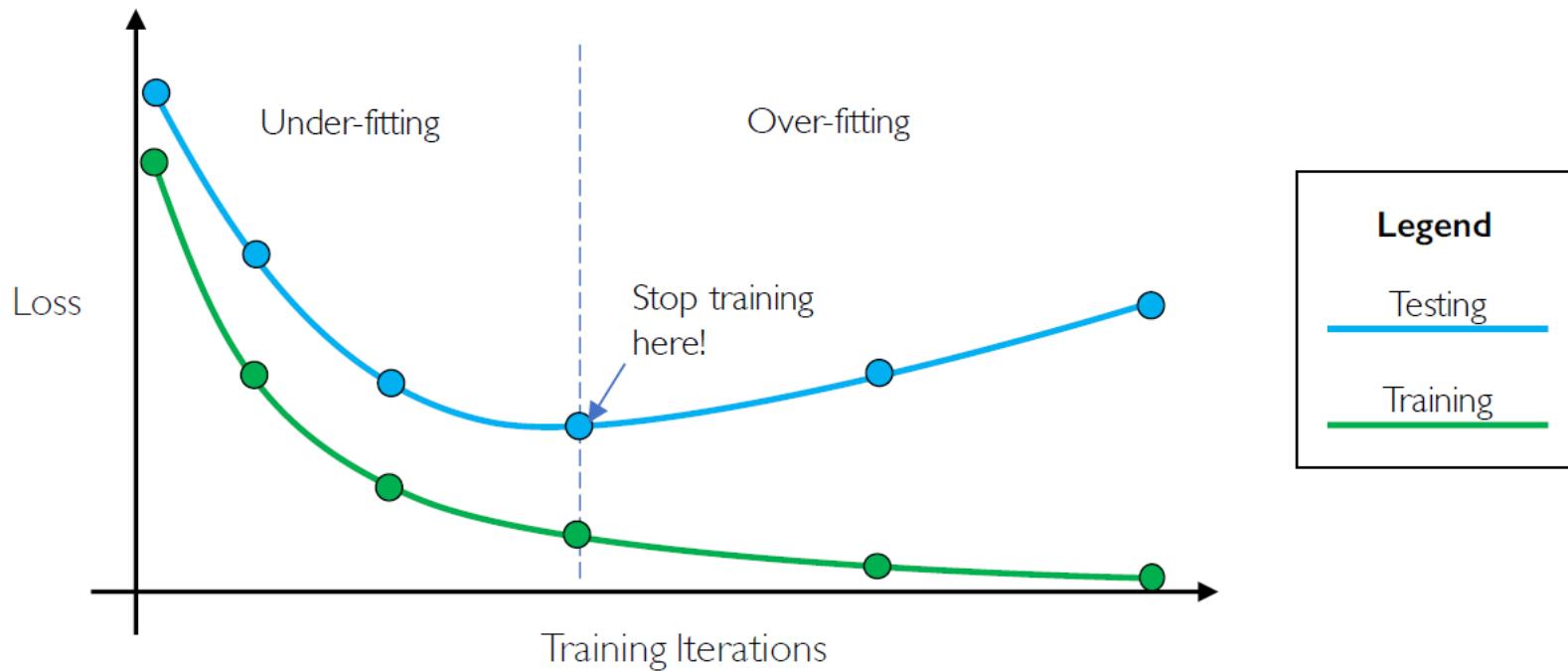
Source: <https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76>

# Regularization

- Technique to help prevent overfitting on training data
  - Helps to generalize our model on unseen (testing) data
- Use
  - Early Stopping

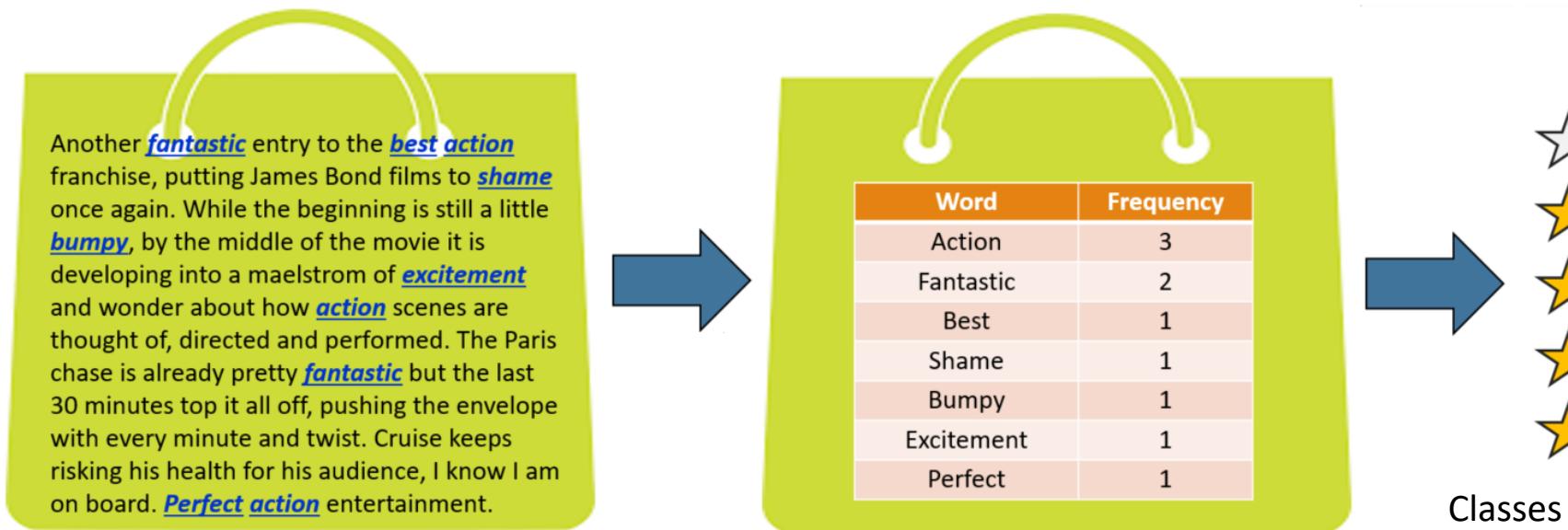
# Early Stopping

- Stop training of model before overfitting



## Exercise 2: Application to NLP

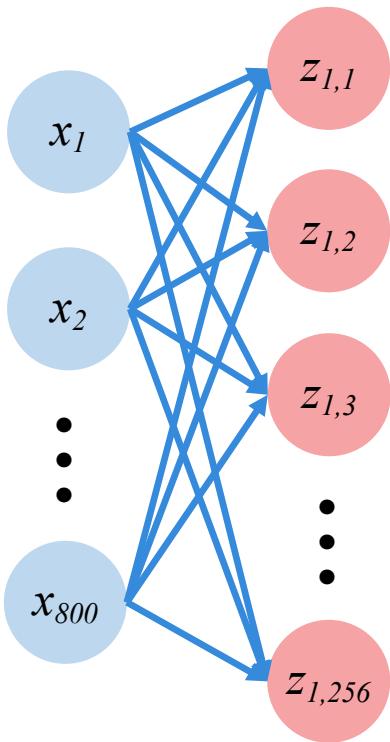
- What would your neural network architecture look like for the following sentiment analysis task?



Vocabulary = 800 Words

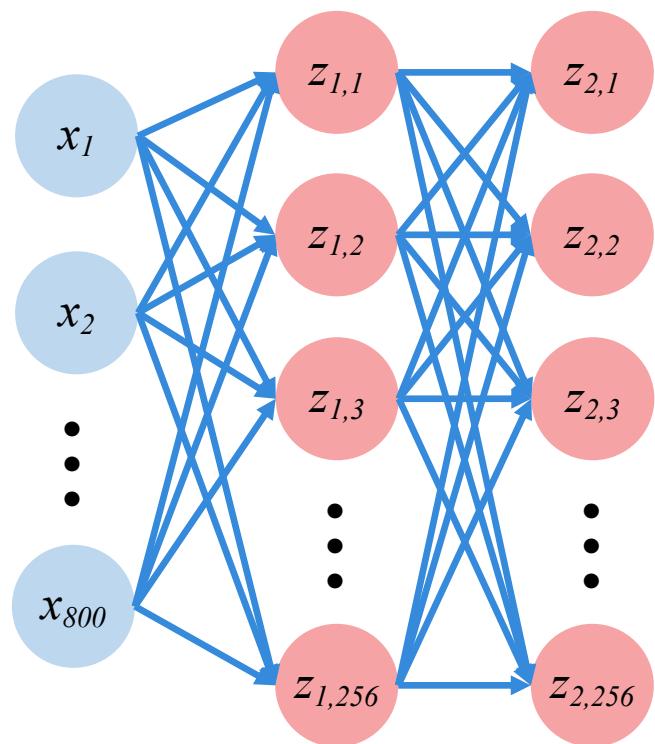
Classes =  
{V.Good, Good,  
Neutral, Bad,  
V.Bad}

# Keras Example: NLP Application



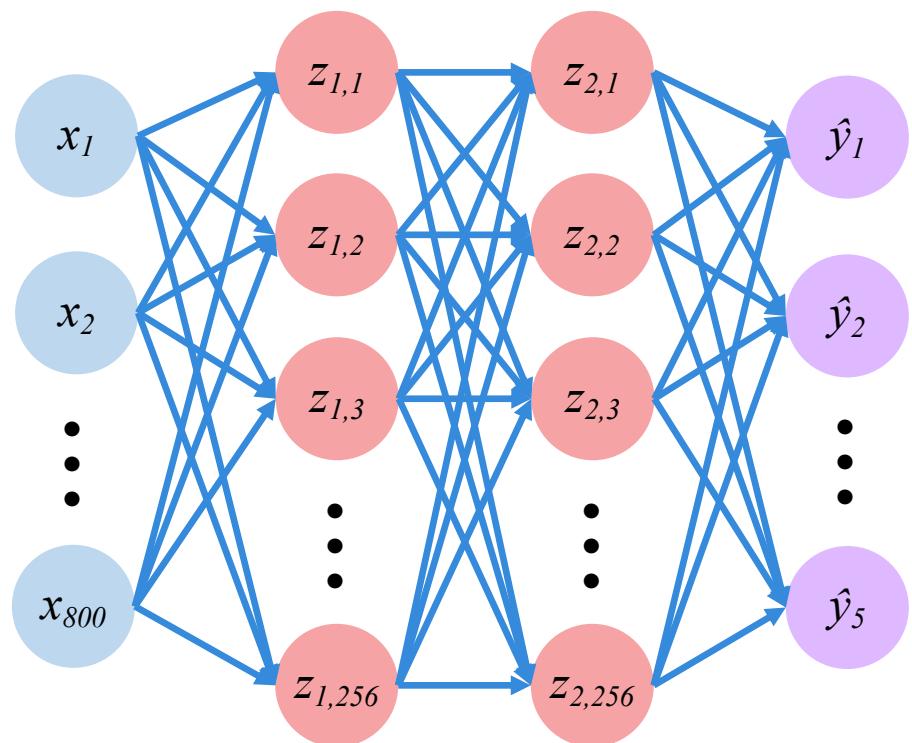
```
1 model = Sequential()  
2  
3 # input layer and hidden layer 1  
4 model.add(Dense(256, activation='relu', input_dim=800))  
5 model.add(Dropout(0.5))
```

# Keras Example: NLP Application



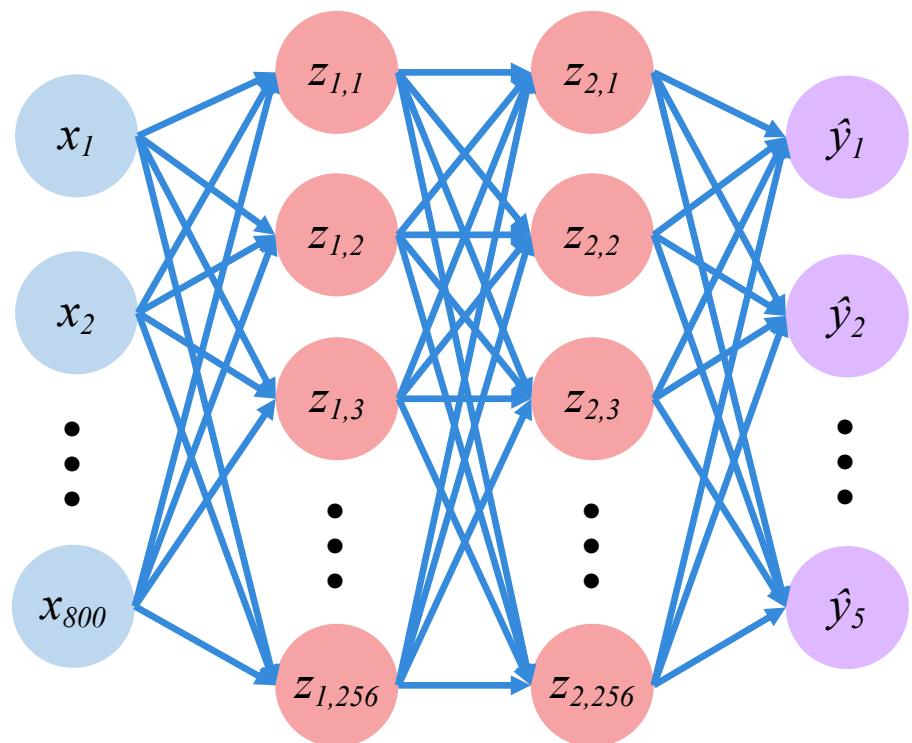
```
1 model = Sequential()
2
3 # input layer and hidden layer 1
4 model.add(Dense(256, activation='relu', input_dim=800))
5 model.add(Dropout(0.5))
6
7 # hidden layer 2
8 model.add(Dense(256, activation='relu'))
9 model.add(Dropout(0.5))
```

# Keras Example: NLP Application



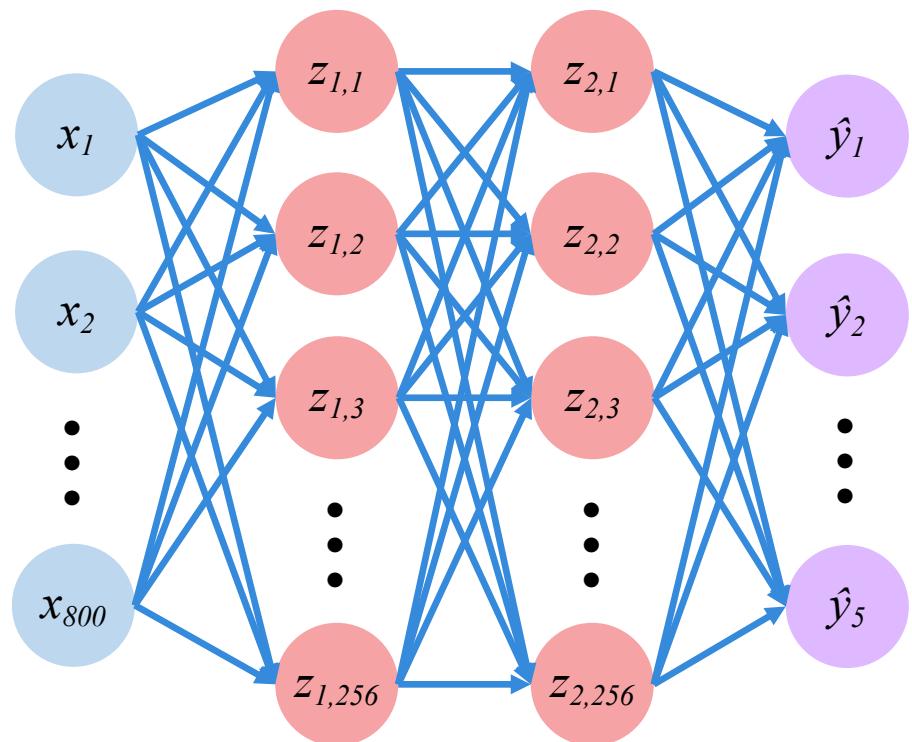
```
1 model = Sequential()
2
3 # input layer and hidden layer 1
4 model.add(Dense(256, activation='relu', input_dim=800))
5 model.add(Dropout(0.5))
6
7 # hidden layer 2
8 model.add(Dense(256, activation='relu'))
9 model.add(Dropout(0.5))
10
11 # output layer
12 model.add(Dense(5, activation='softmax'))
```

# Keras Example: NLP Application



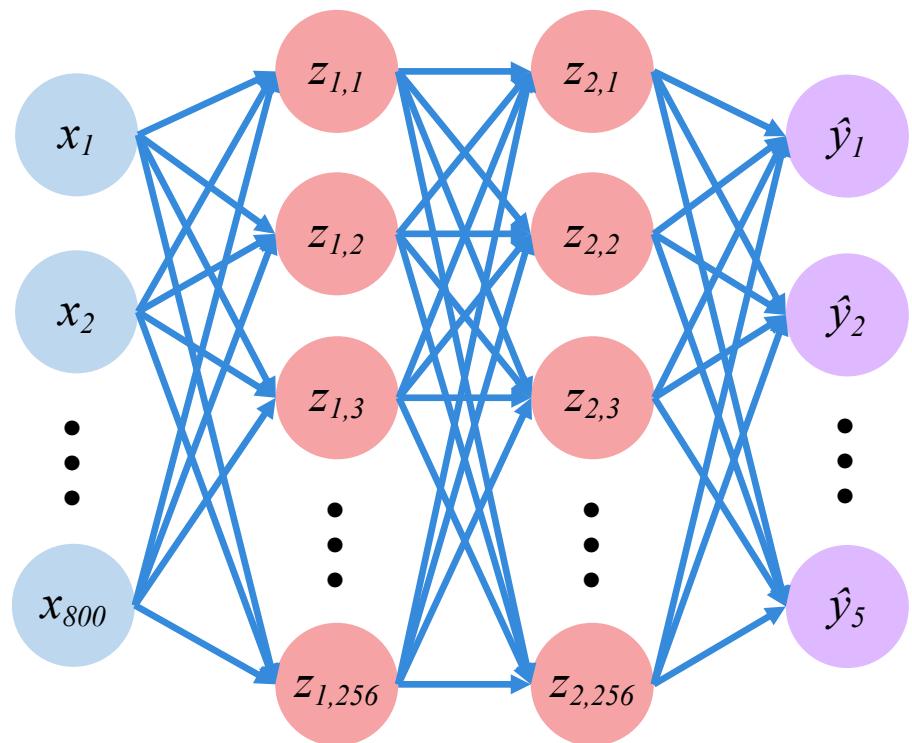
```
1 model = Sequential()
2
3 # input layer and hidden layer 1
4 model.add(Dense(256, activation='relu', input_dim=800))
5 model.add(Dropout(0.5))
6
7 # hidden layer 2
8 model.add(Dense(256, activation='relu'))
9 model.add(Dropout(0.5))
10
11 # output layer
12 model.add(Dense(5, activation='softmax'))
13
14 # configure Learning process
15 model.compile(loss='categorical_crossentropy',
16                 optimizer='adam',
17                 metrics=['accuracy'])
```

# Keras Example: NLP Application



```
1 model = Sequential()
2
3 # input layer and hidden layer 1
4 model.add(Dense(256, activation='relu', input_dim=800))
5 model.add(Dropout(0.5))
6
7 # hidden layer 2
8 model.add(Dense(256, activation='relu'))
9 model.add(Dropout(0.5))
10
11 # output layer
12 model.add(Dense(5, activation='softmax'))
13
14 # configure Learning process
15 model.compile(loss='categorical_crossentropy',
16                 optimizer='adam',
17                 metrics=['accuracy'])
18
19 # train our model
20 model.fit(x_train, y_train,
21             epochs=50,
22             batch_size=128)
```

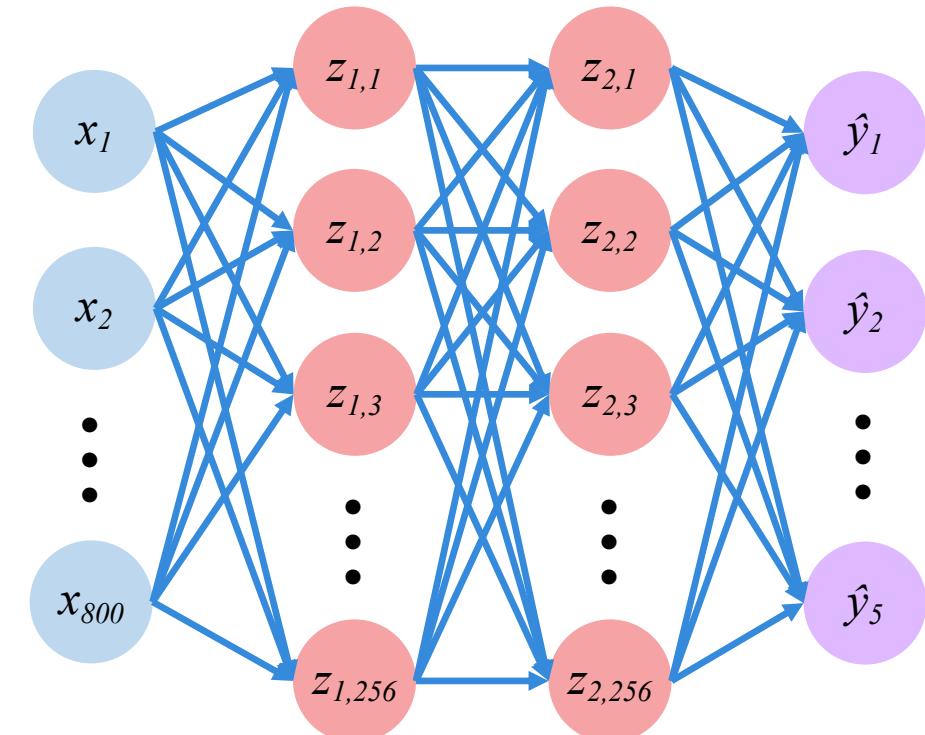
# Keras Example: NLP Application



```
1 model = Sequential()
2
3 # input layer and hidden layer 1
4 model.add(Dense(256, activation='relu', input_dim=800))
5 model.add(Dropout(0.5))
6
7 # hidden layer 2
8 model.add(Dense(256, activation='relu'))
9 model.add(Dropout(0.5))
10
11 # output layer
12 model.add(Dense(5, activation='softmax'))
13
14 # configure Learning process
15 model.compile(loss='categorical_crossentropy',
16                 optimizer='adam',
17                 metrics=['accuracy'])
18
19 # train our model
20 model.fit(x_train, y_train,
21             epochs=50,
22             batch_size=128)
23
24 # test our model
25 score = model.evaluate(x_test, y_test, batch_size=128)
```

# Exercise 3: Discussion of Shortcomings

- What are the possible shortcomings of our previous Naïve Bayes and this MLP approach to sentiment classification?



A visualization showing the relationship between word frequency and sentiment. A green card displays a table of word frequencies, with a blue arrow pointing to a vertical stack of five yellow stars, representing a positive sentiment rating.

Word	Frequency
Action	3
Fantastic	2
Best	1
Shame	1
Bumpy	1
Excitement	1
Perfect	1



# How deep learning helps

- Suppose we have L input sentences each having M words. Each of these words have N number of features. Hence, the input matrix X has dimension of  $L \times M \times N$ .
- Now say, we want to classify sentiment of each sentence.
- Can we automatically learn a function that learns and tells us the role of each word in the classification?
- We call this **attention mechanism**.
- The idea is simple.
- We apply a dense layer or FC network on the input matrix X. FC of dimension  $N \times 1$ . We get  $X'$  of dimension  $L \times M \times 1$ .
- We add some non-linearity to  $X'$  so we get  $X'' = \tanh(X')$
- Now, apply softmax to the last dimension to get the importance probability of each word. Say, we store this to  $X''' = \text{softmax}(X'', \text{axis}=-1)$  of dimension  $L \times M \times 1$

# How deep learning helps

- We multiply  $X'''$  with the input  $X$  to find an abstract representation of our inputs that encode the role of each word along with the original semantic of the words.
- Hence we have  $X'''' = X''' * X$  to get the abstract features of dimension  $L * N$
- These features can be sent to a softmax layer for classification
- Using the backpropagation method, the deep network can learn the weights and function required to leverage the role of each word in the classification.

# Things to do

- For the upcoming lab, please install Keras
  - See <https://keras.io/#installation>

# References

- [http://introtodeeplearning.com/materials/2018\\_6S191\\_Lecture1.pdf](http://introtodeeplearning.com/materials/2018_6S191_Lecture1.pdf)
- <http://www.yaronhadad.com/deep-learning-most-amazing-applications/>
- <https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092>
- <https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f>