SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN
Established in collaboration with MIT

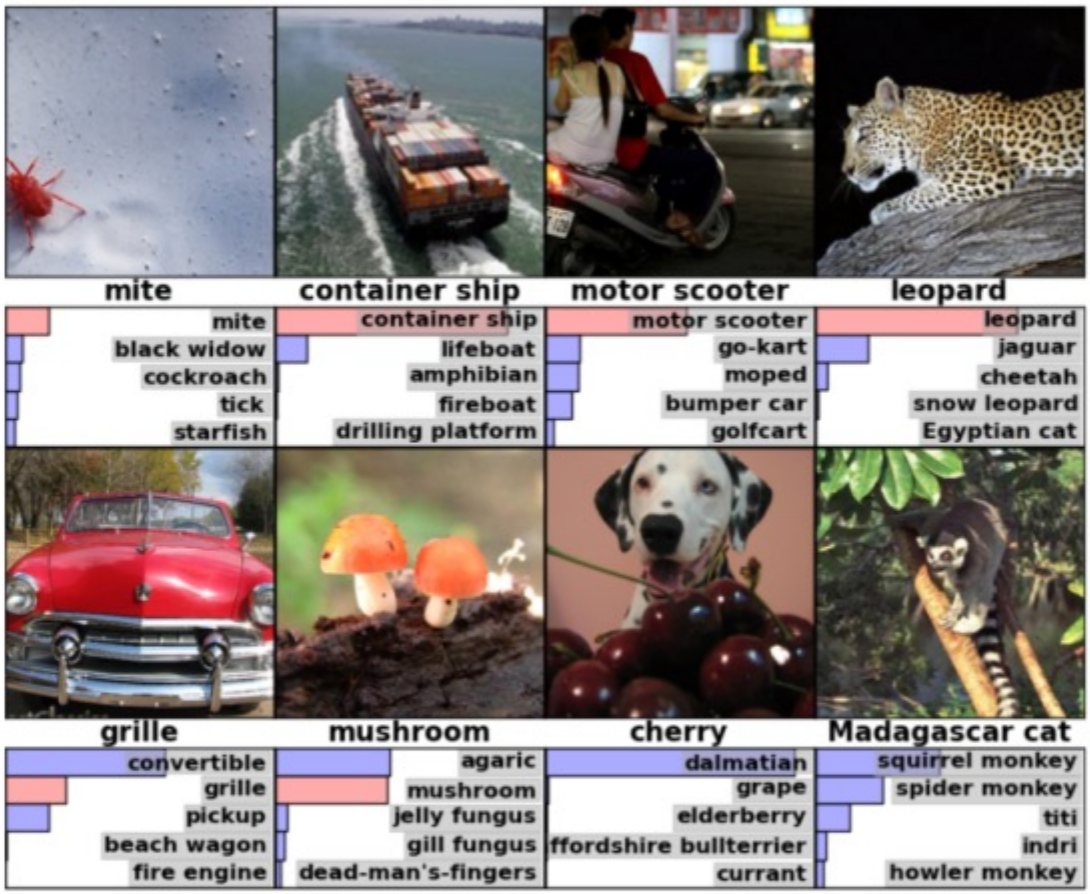
Introduction to Deep Learning

Soujanya Poria

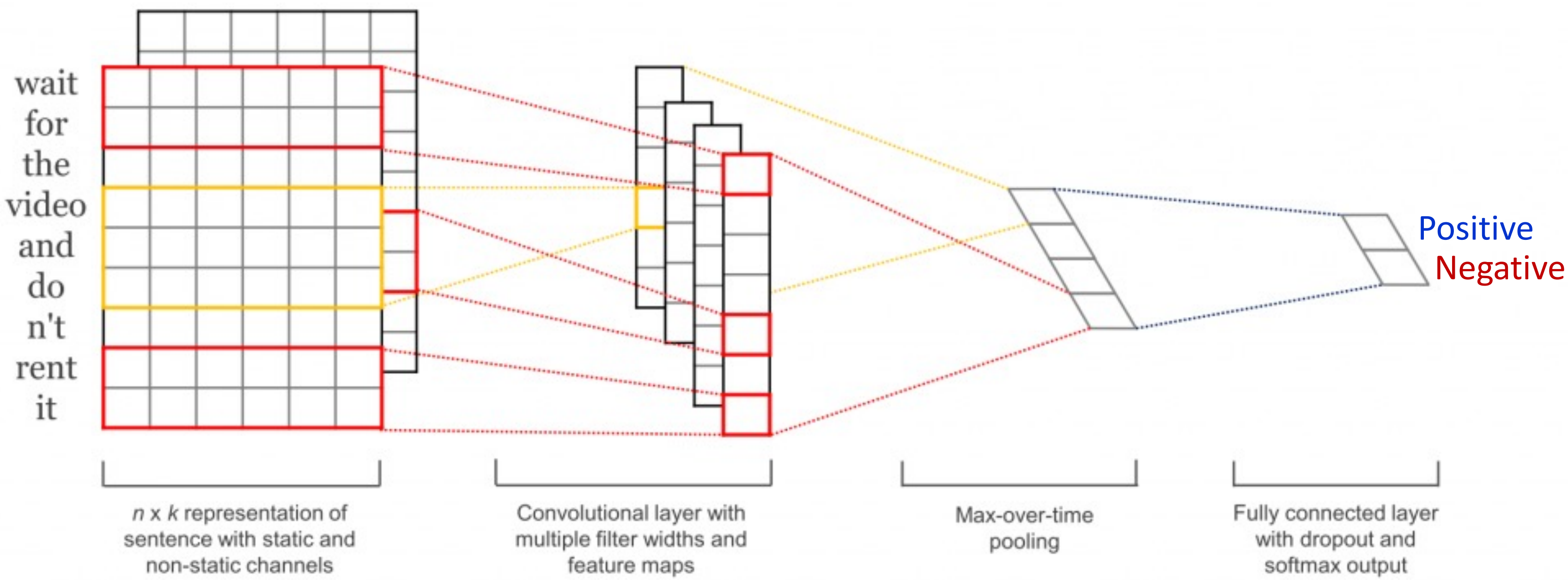
Objectives

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

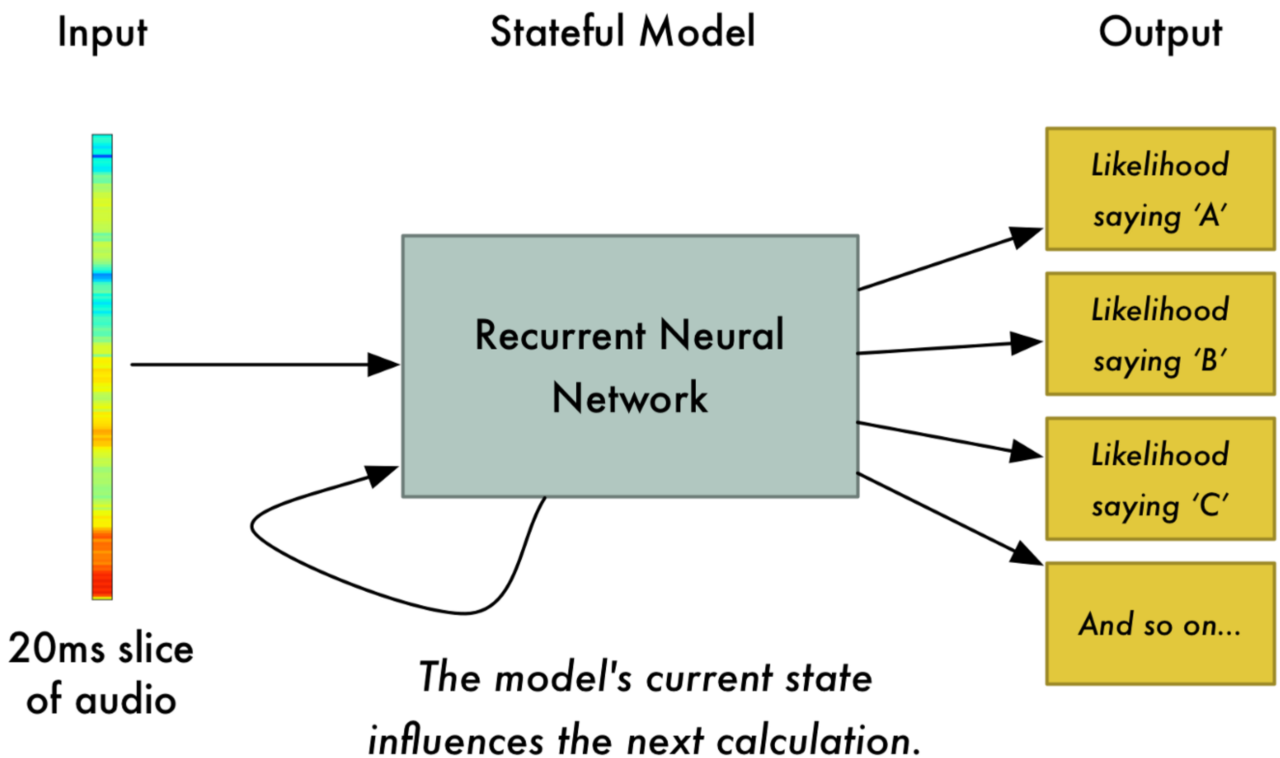
Examples: Images



Examples: Text



Examples: Audio

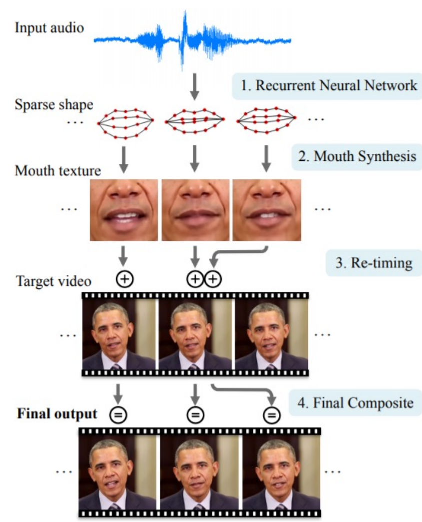


https://medium.com/@ageitgey/machine-learning-is-fun-part-6-how-to-do-speech-recognition-with-deep-learning-28293c162f7a

Various other examples

Image Captioning

Colouring B&W Photos



Synthesizing Speech

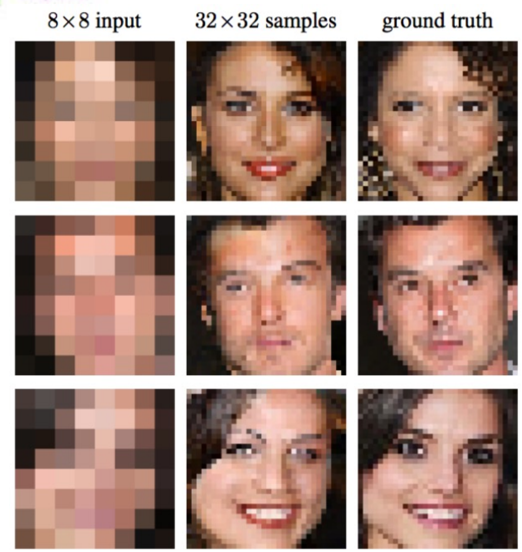


Image Super-resolution

Why Deep Learning?

1952 Stochastic Gradient Descent

1958 Perceptron
- Learnable Weights

⋮

1986 Backpropagation
- Multi-Layer Perceptron

1995 Deep Convolutional NN
- Digit Recognition

⋮

Neural Networks date back decades, so why the resurgence?

1. Big Data
- Larger Datasets
- Easier Collection & Storage

IMAGENET

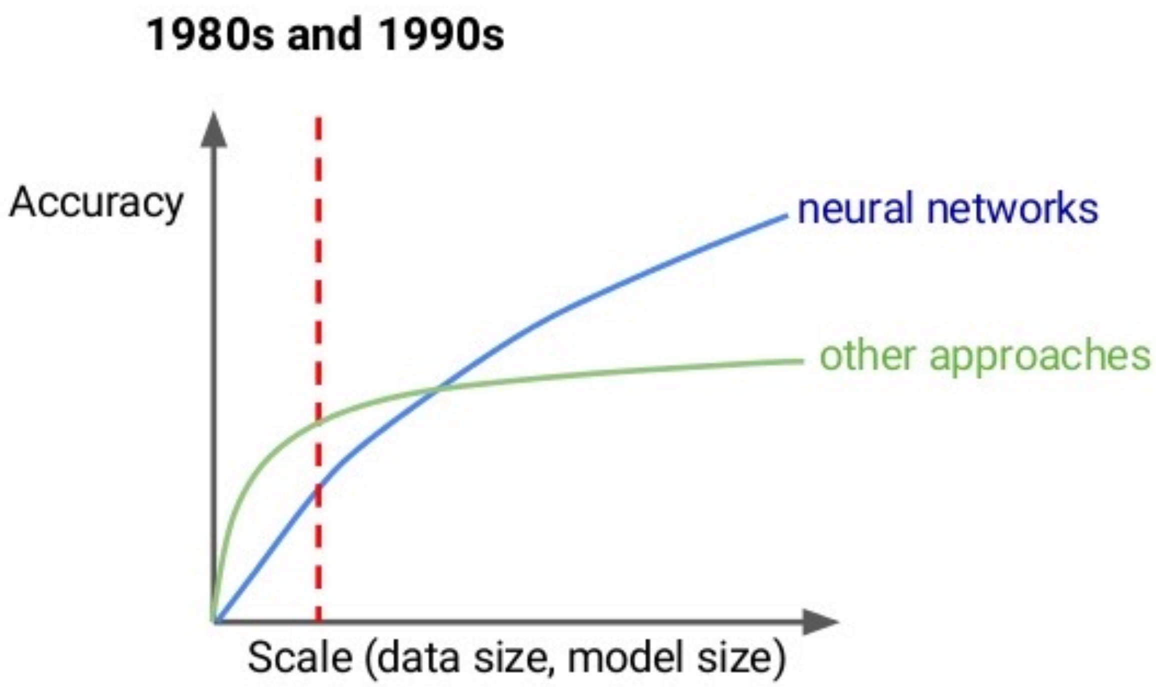
WIKIPEDIA
The Free Encyclopedia

2. Hardware
- Graphics Processing Units (GPUs)
- Massively Parallelizable

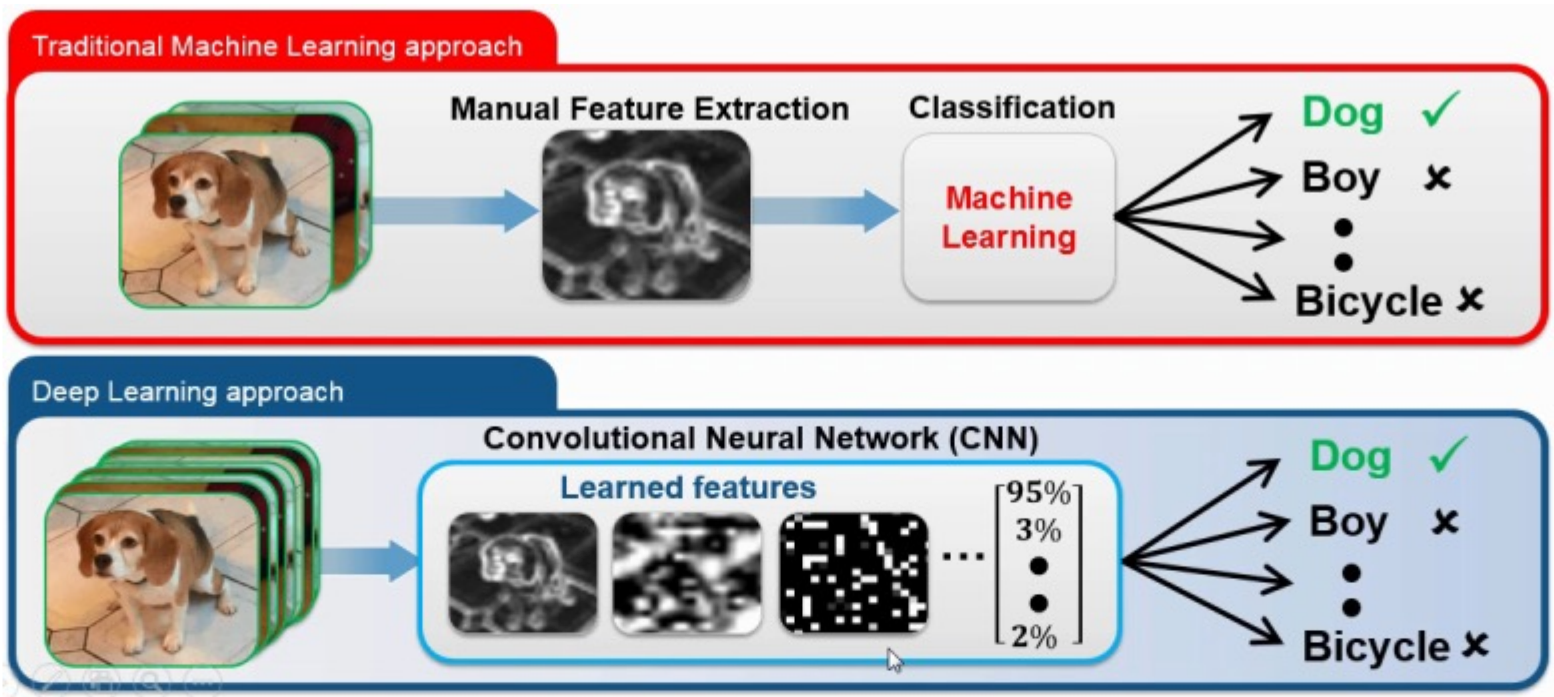
3. Software
- Improved Techniques
- New Models
- Toolboxes

TensorFlow

Why Deep Learning?

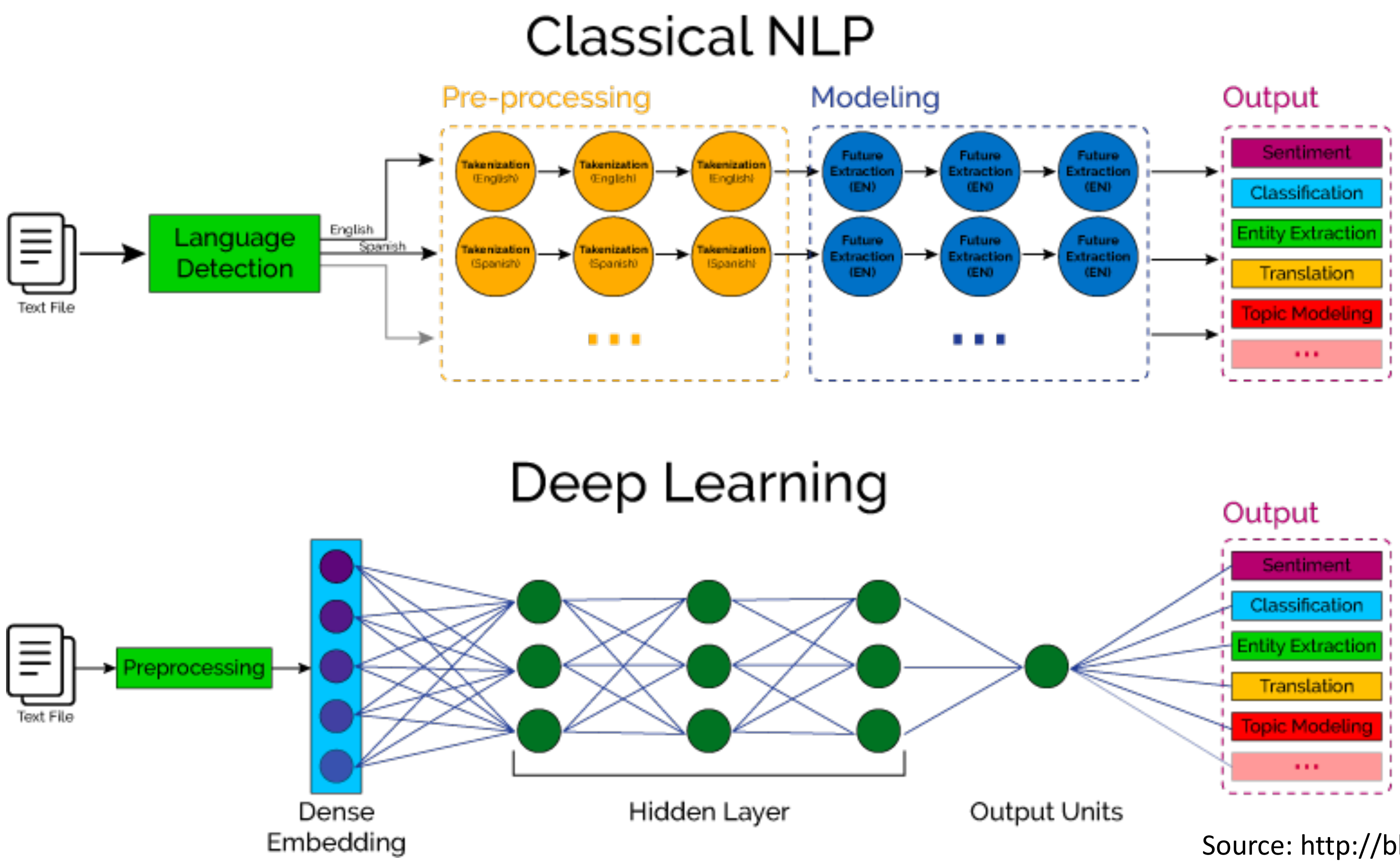


Differences



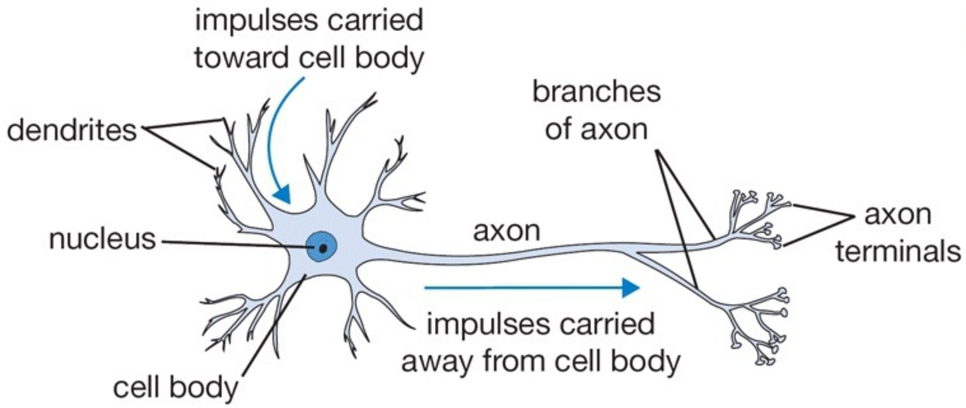
https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep_learning.html

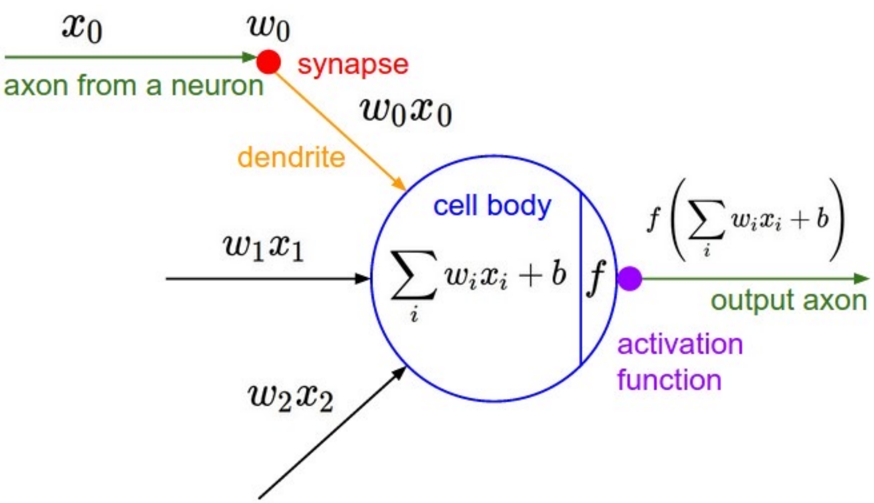
Differences



Source: http://blog.aylien.com

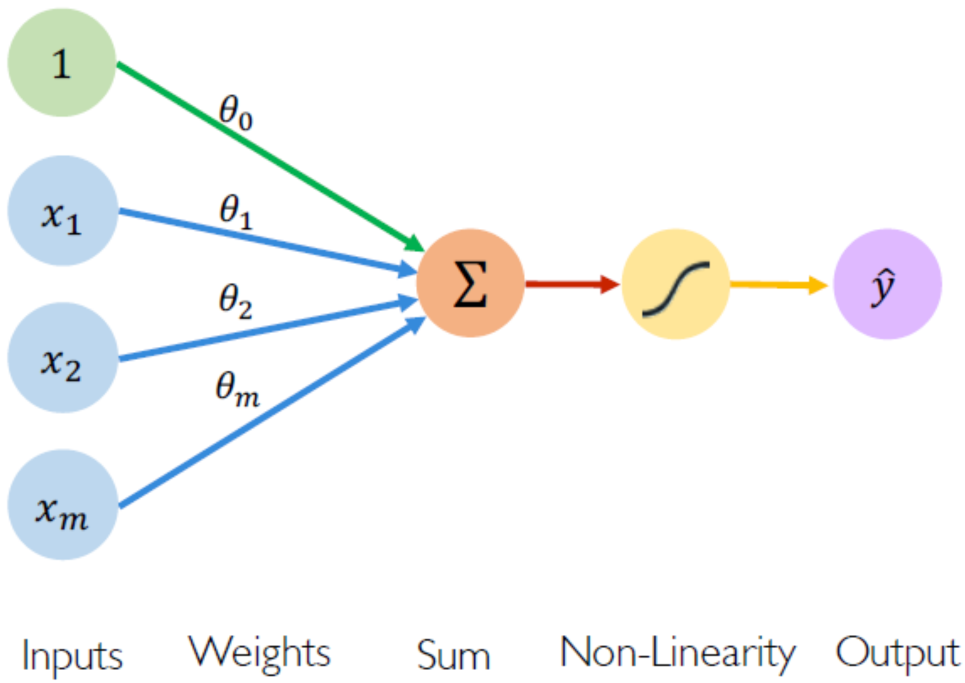
Inspiration behind Neural Networks





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

Perceptron



Output

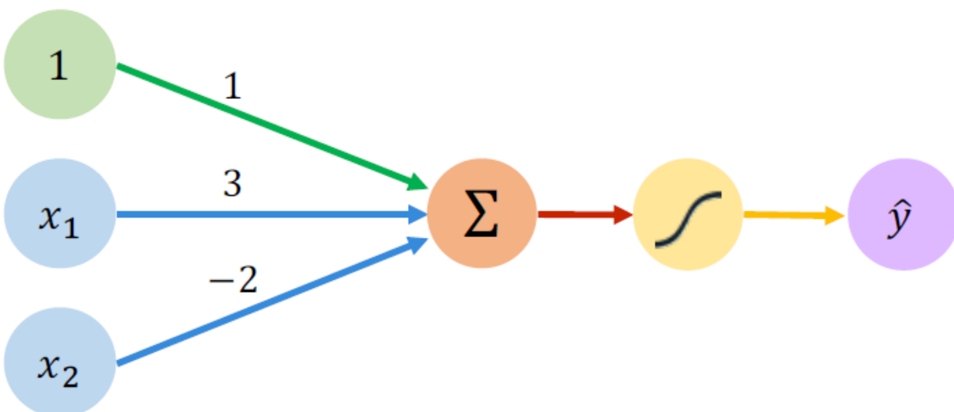
Linear combination of inputs

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

Non-linear activation function

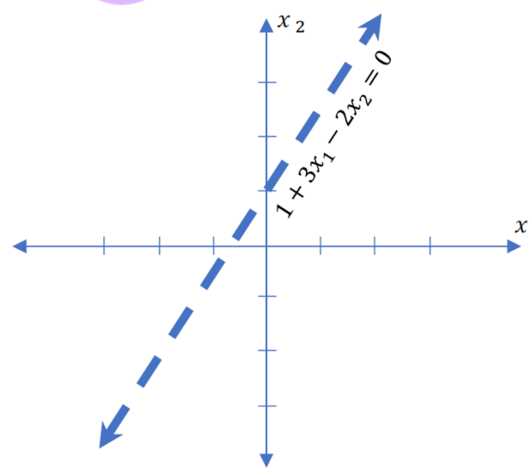
Bias

Why Non-linear Activation?

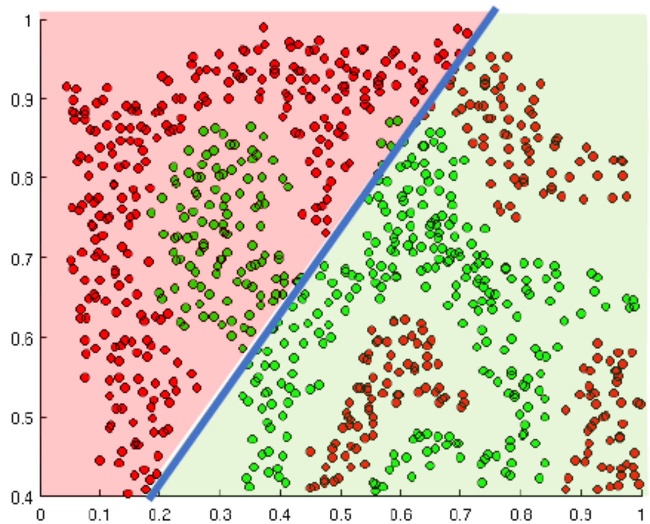


We have: $\theta_0 = 1$ and $\boldsymbol{\theta} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

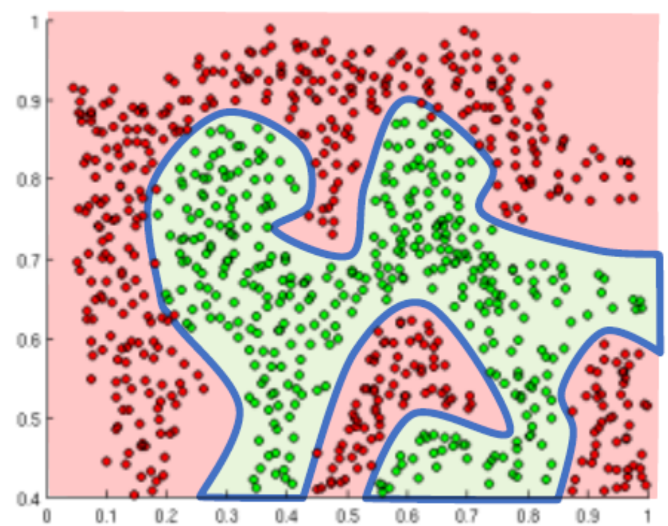
$$\hat{y} = g\left(\theta_0 + \boldsymbol{X}^T \boldsymbol{\theta}\right)$$
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$
$$\hat{y} = g\underbrace{\left(1 + 3x_1 - 2x_2\right)}_{\text{This is just a line in 2D!}}$$



Why Non-linear Activation?



Linear Activation functions produce linear decisions no matter the network size

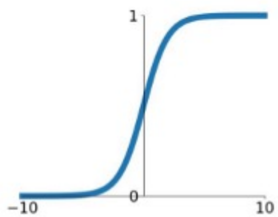


Non-linearities allow us to approximate arbitrarily complex functions

Activation Functions

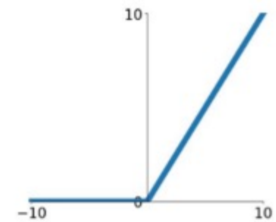
Sigmoid

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



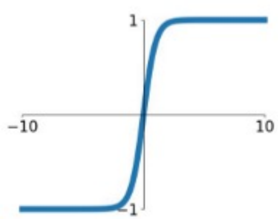
ReLU

$\max(0, x)$



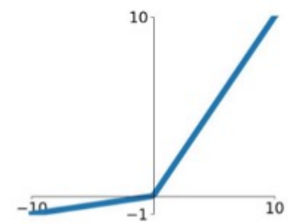
tanh

$\tanh(x)$



Leaky ReLU

$\max(0.1x, x)$



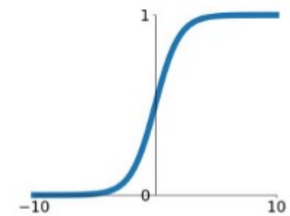
Activation Functions

- 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

Sigmoid

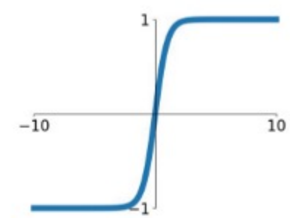
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



- Additional problem with Sigmoid
    - Slower convergence than tanh

tanh

$$\tanh(x)$$

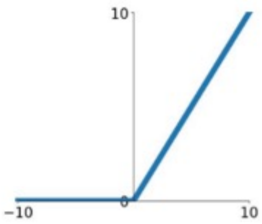


Activation Functions

- Rectified Linear Units (ReLU)
  - Developed to overcome the vanishing gradient problem
  - However, some neuron may "die"

ReLU

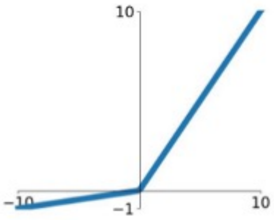
$\max(0, x)$



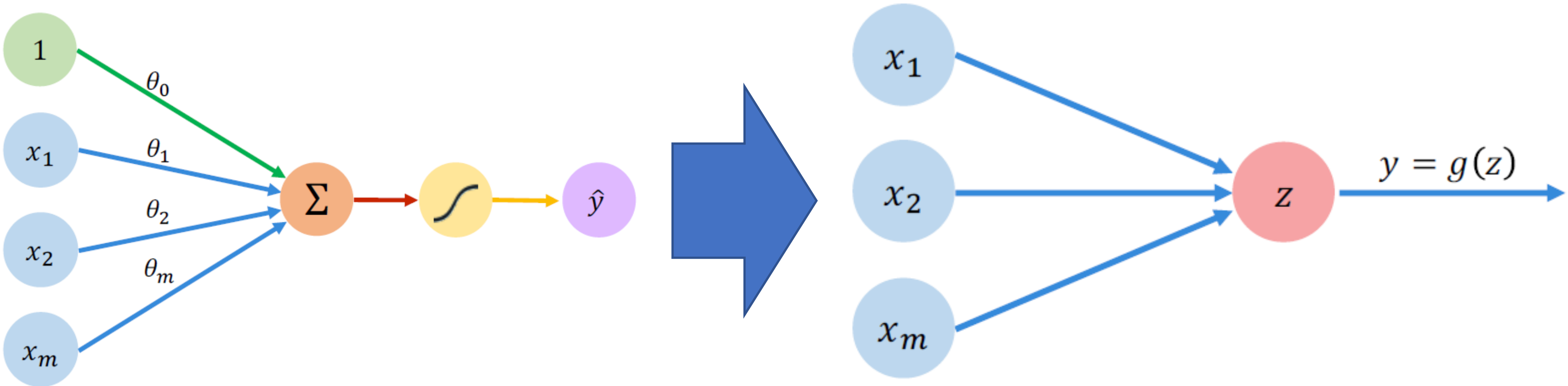
- Leaky ReLU
  - Prevents neurons from "dying" by using a small negative slope

Leaky ReLU

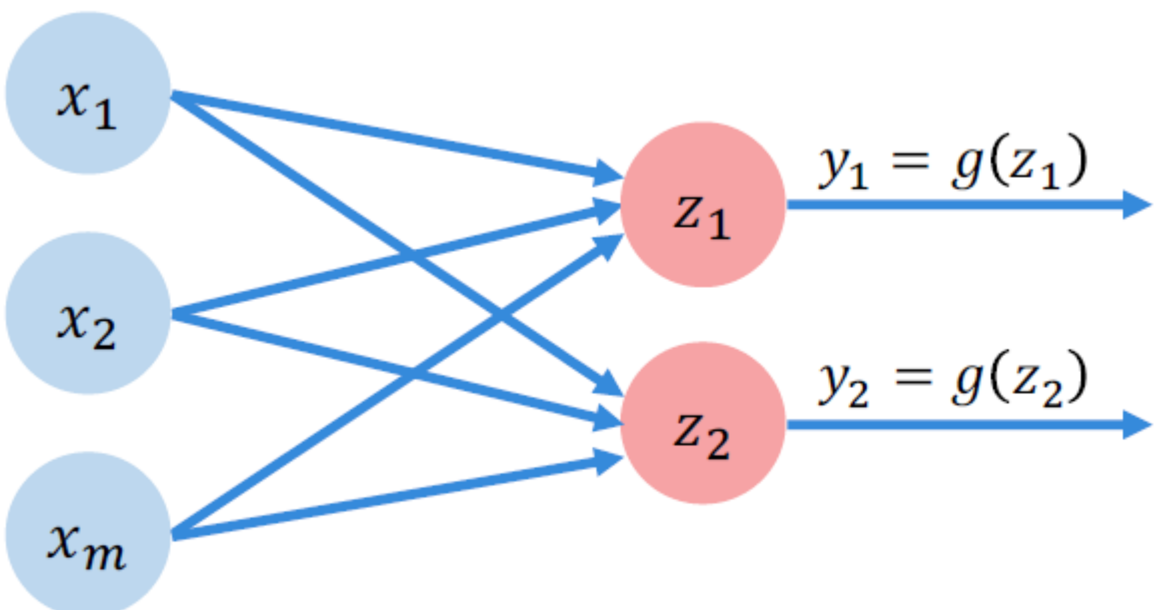
$\max(0.1x, x)$



Simplifying the Perceptron

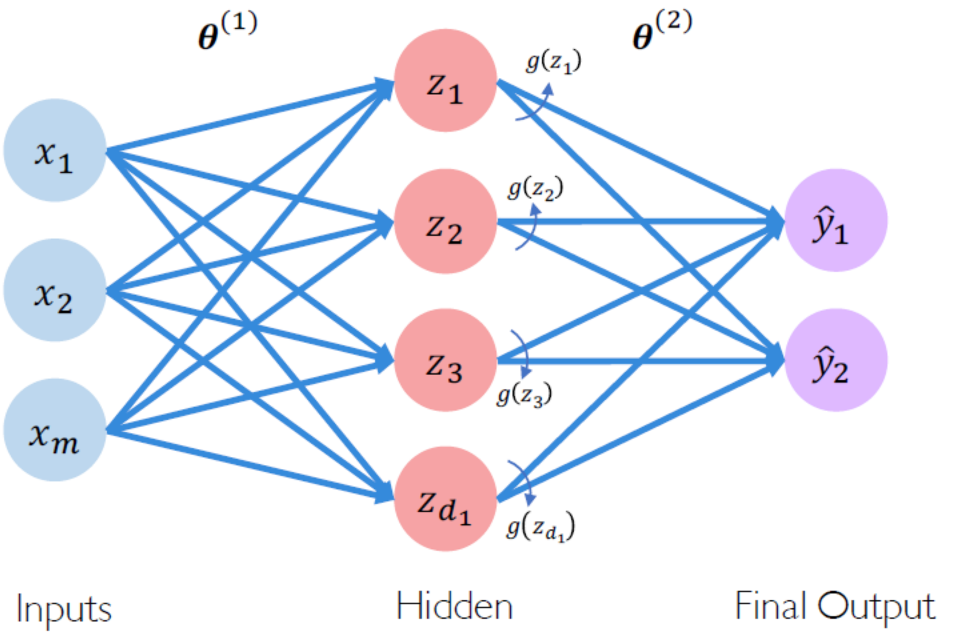


Multiple Perceptron



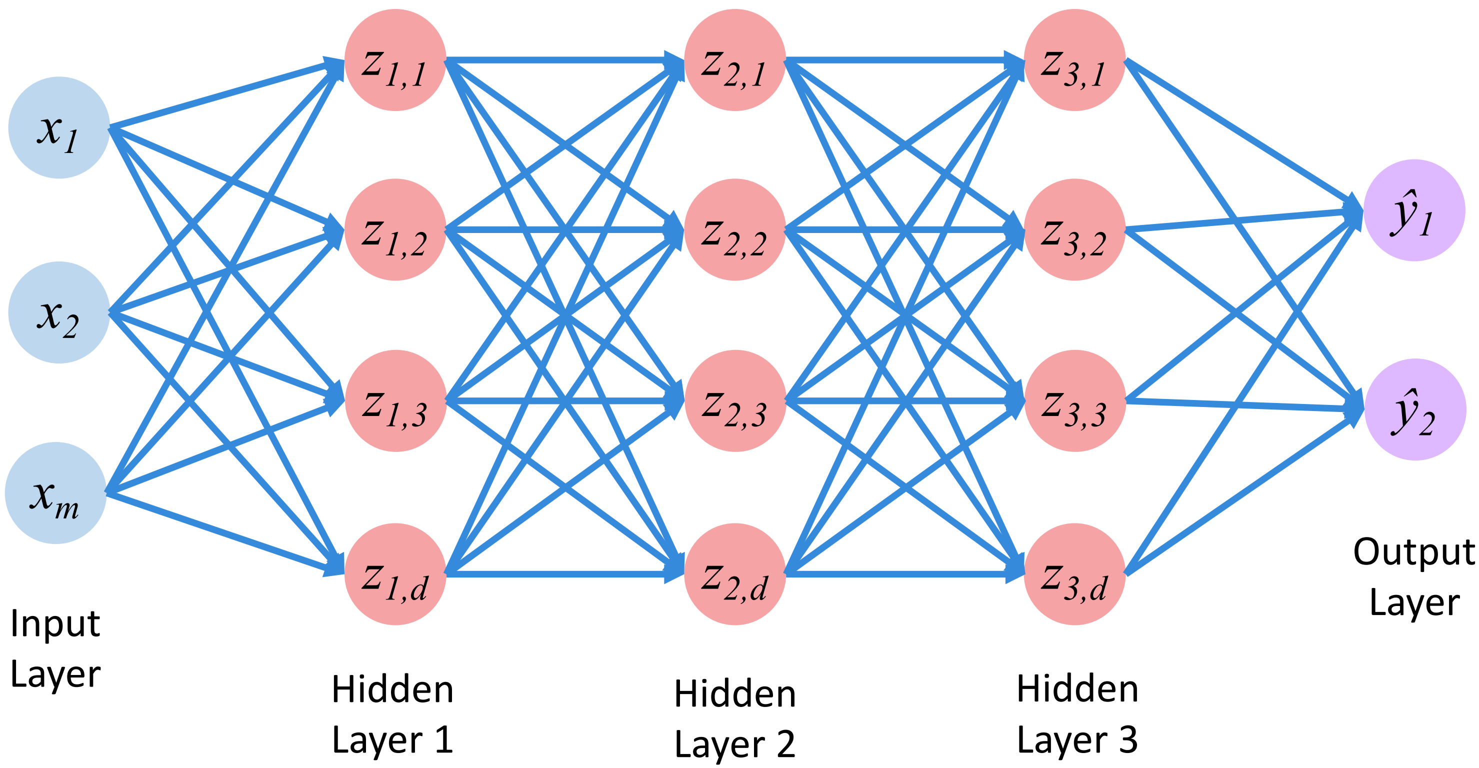
Neural Network

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

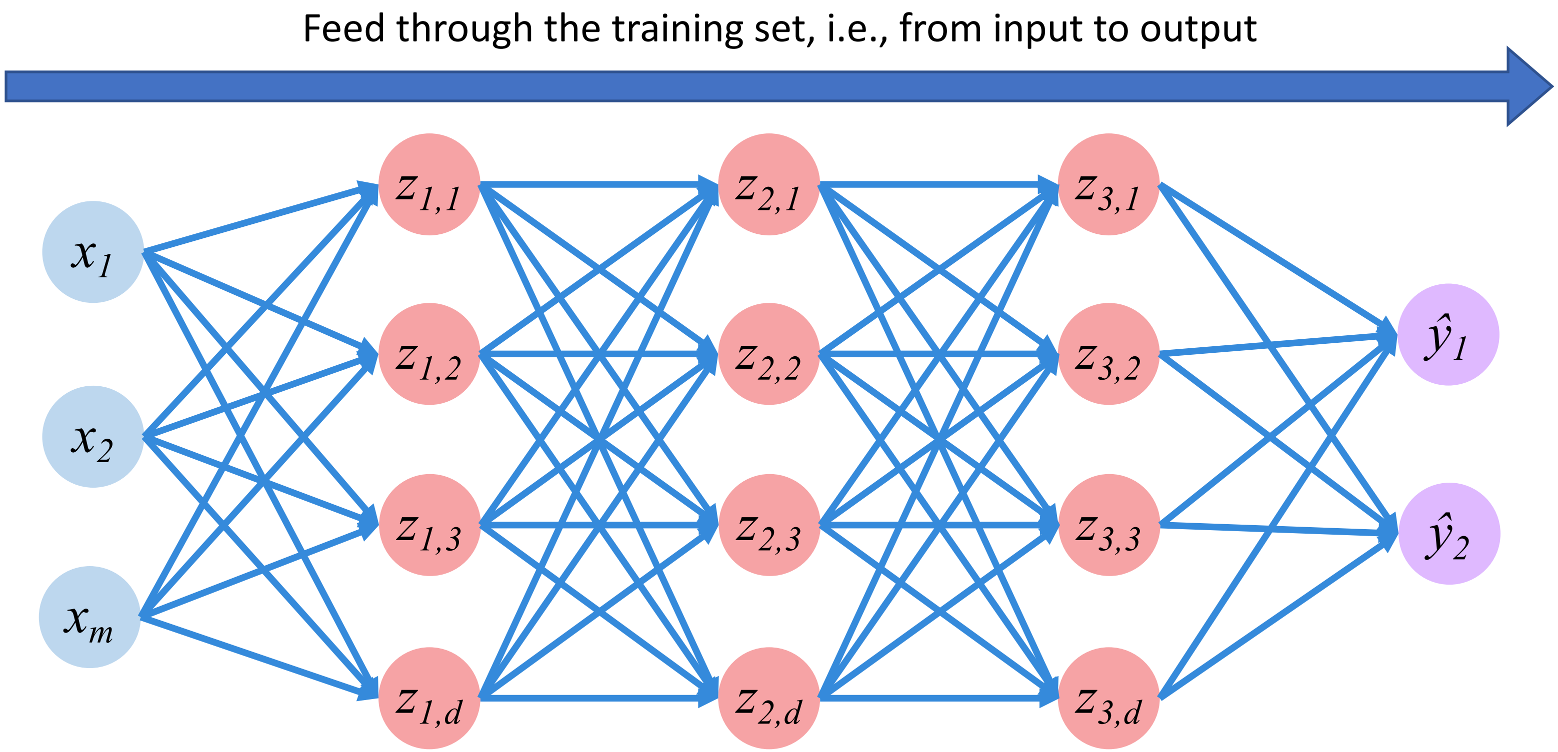


Deep Neural Network

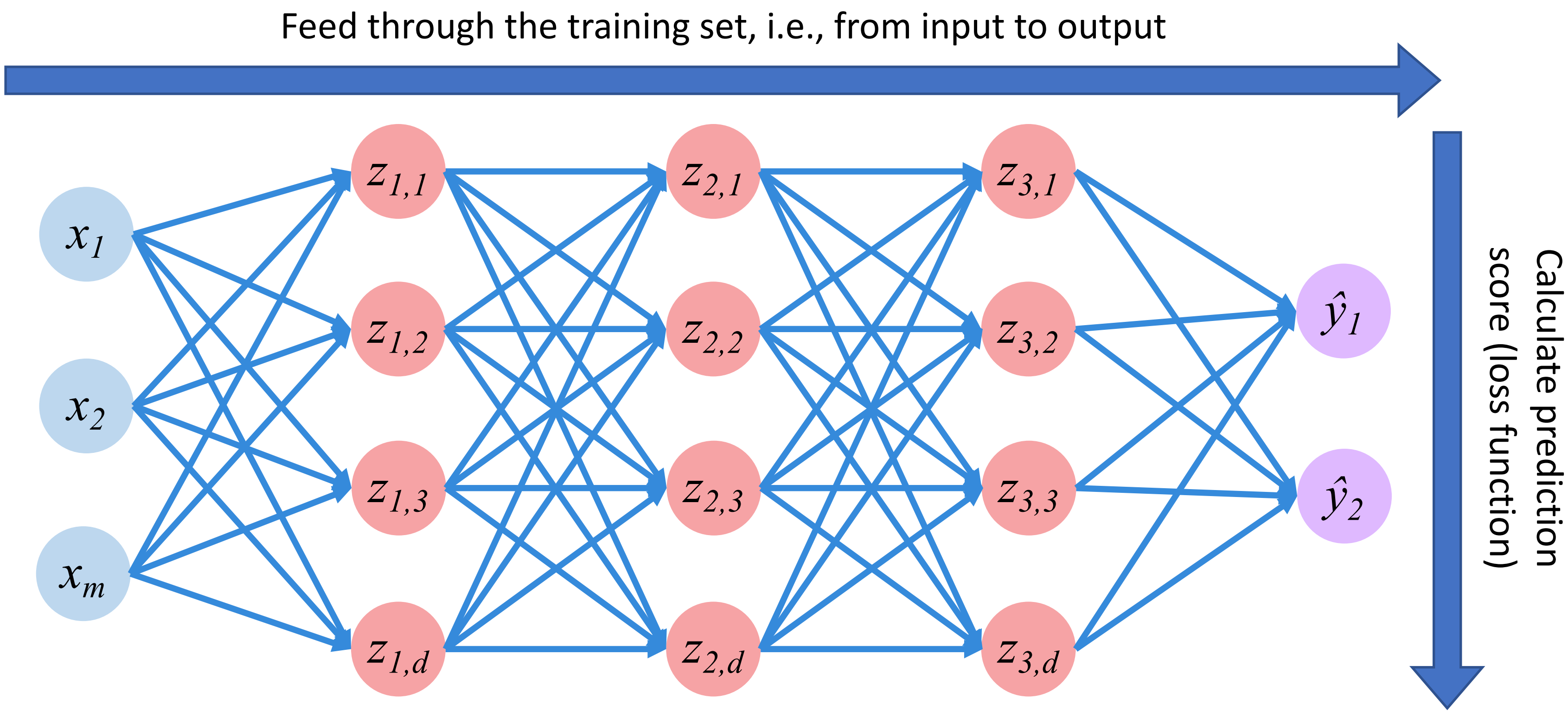
- Adding on multiple hidden layers



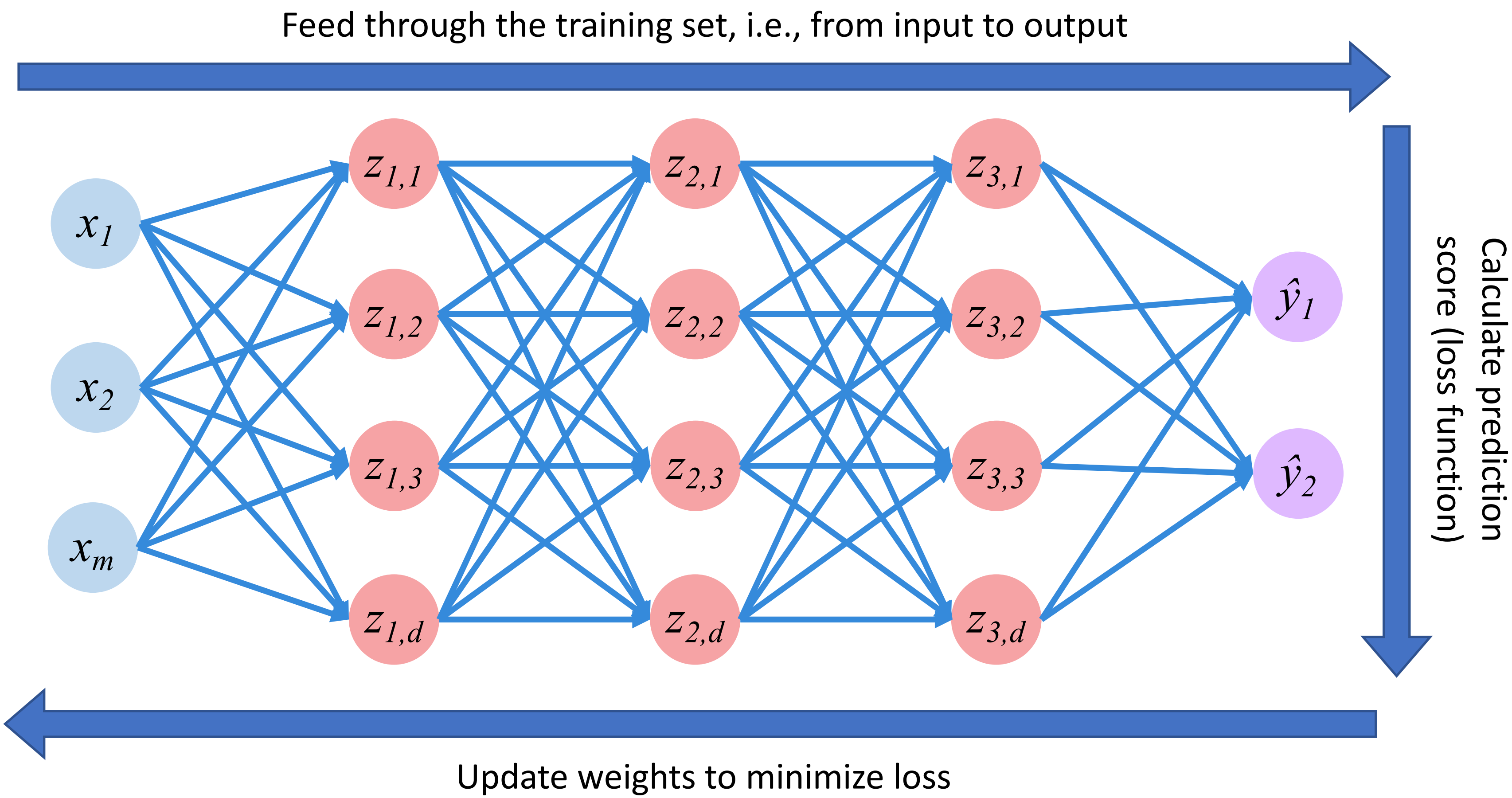
Training a Neural Network



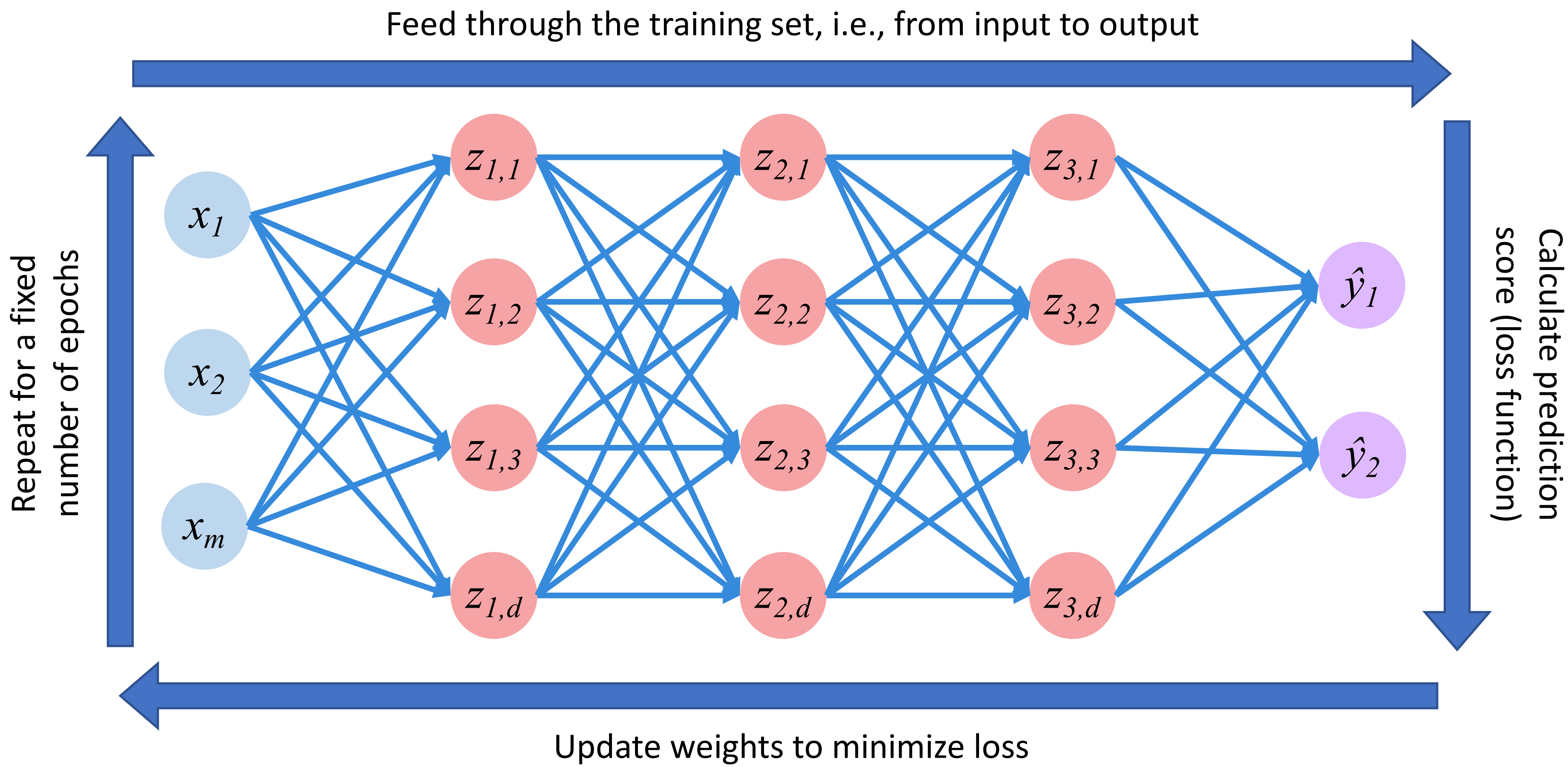
Training a Neural Network



Training a Neural Network



Training a Neural Network



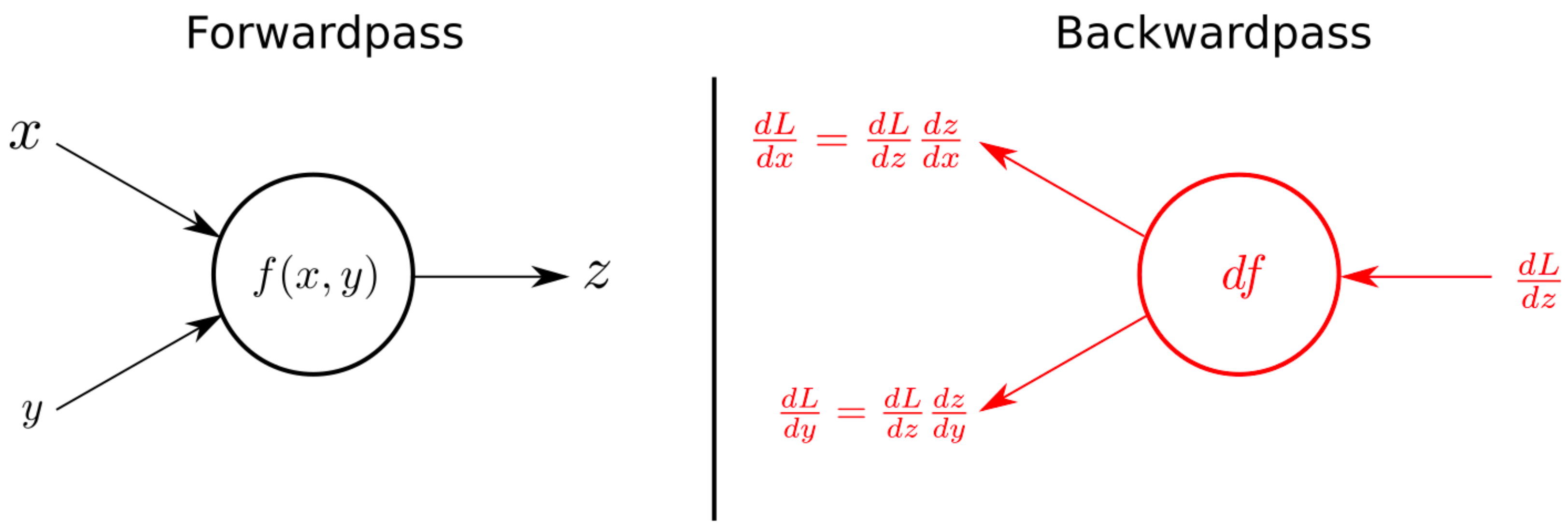
Training a Neural Network

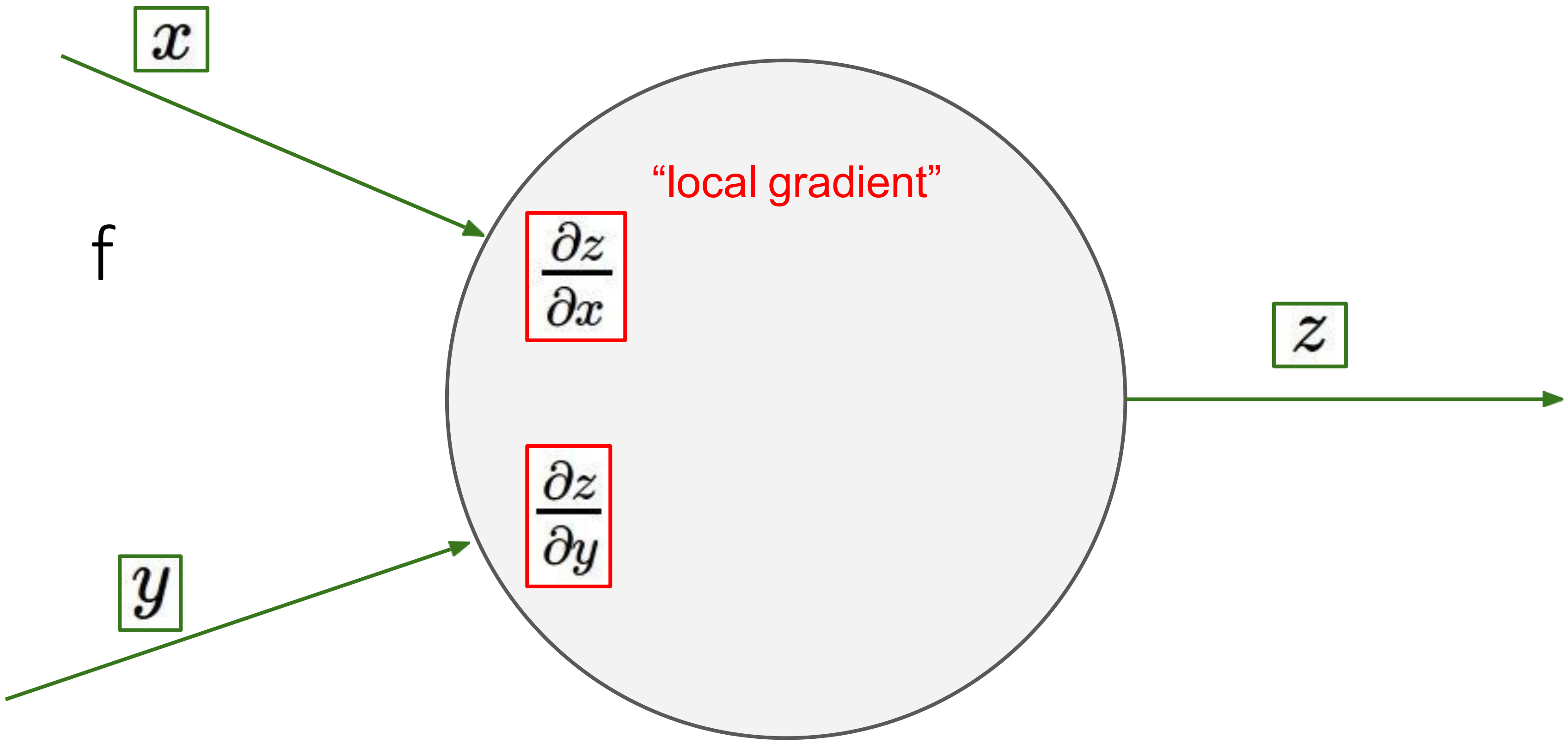
- What type of loss function?
- How to learn and update weights?
- How much training data to use?

Forward propagation and Backpropagation

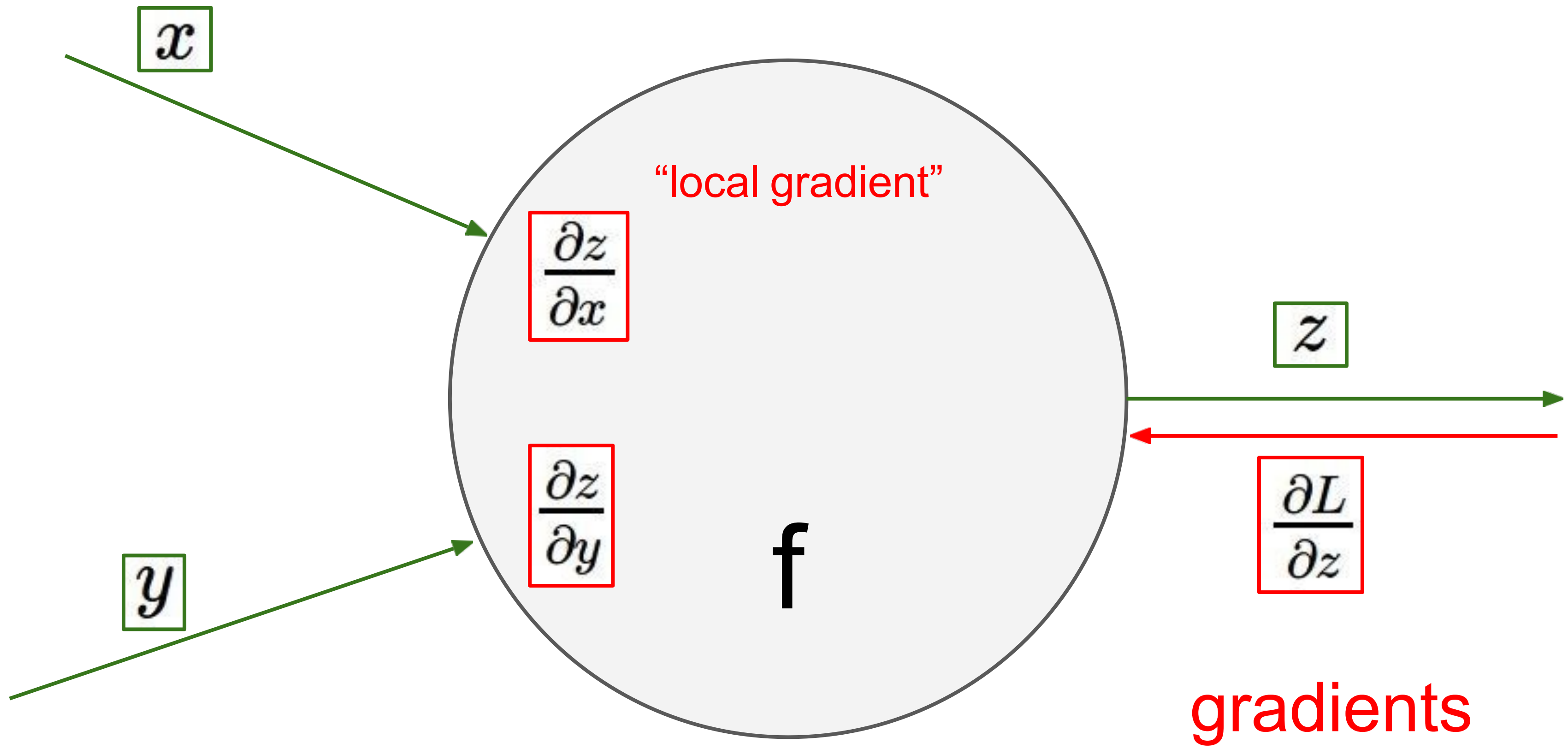
1. Given inputs and the labels first do forward propagation through the network. It is done by feeding the input to the network. Finally, we will get an output from the last layer. This process is called forward propagation.
2. Calculate the error/loss and gradients of the loss with respect to each of the parameters in the network. The error/loss is the difference between the network generated output and the actual output in the training dataset.
3. Backpropagate the gradients and update/tune the weights using gradient descent technique. This process is called backpropagation.

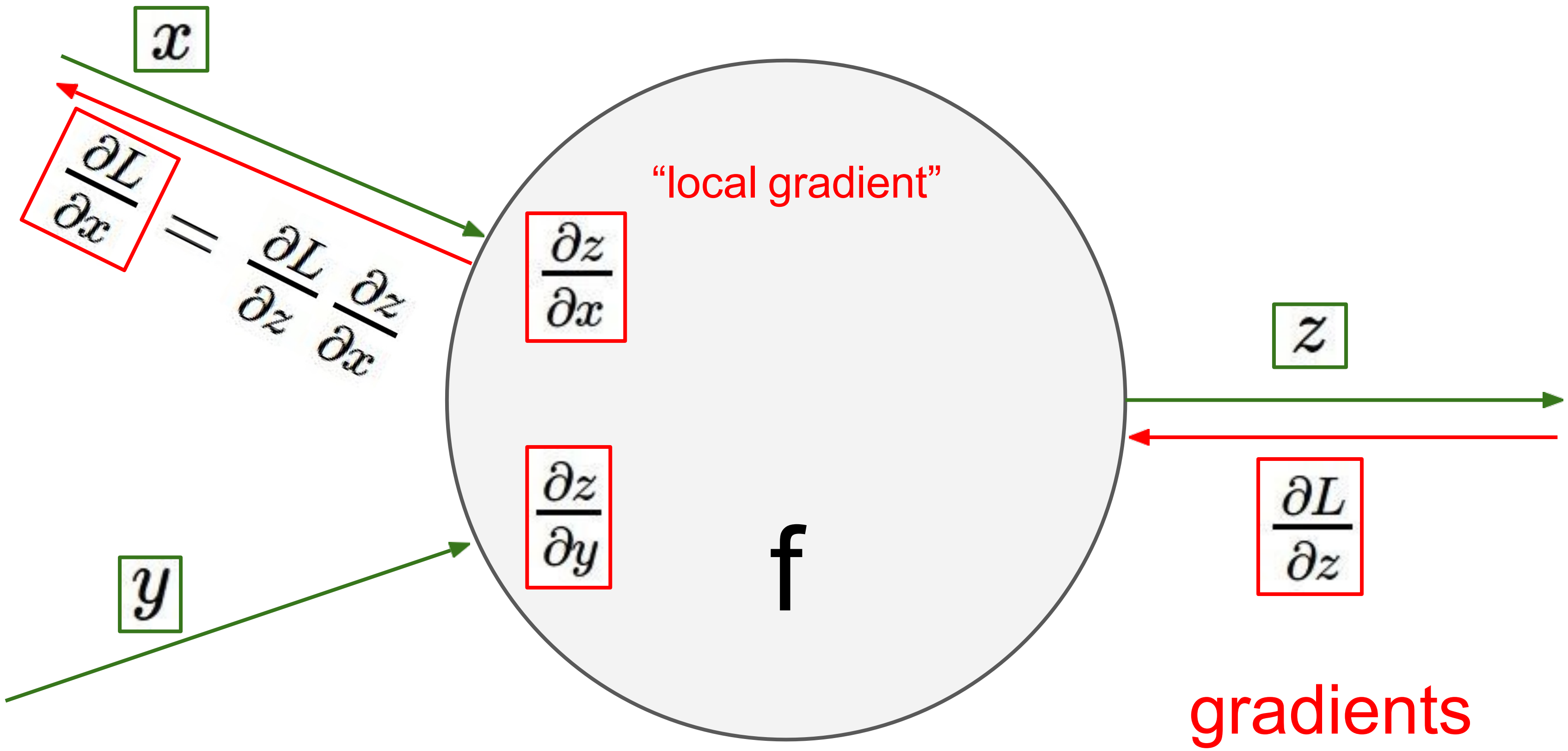
Forward propagation and Backpropagation



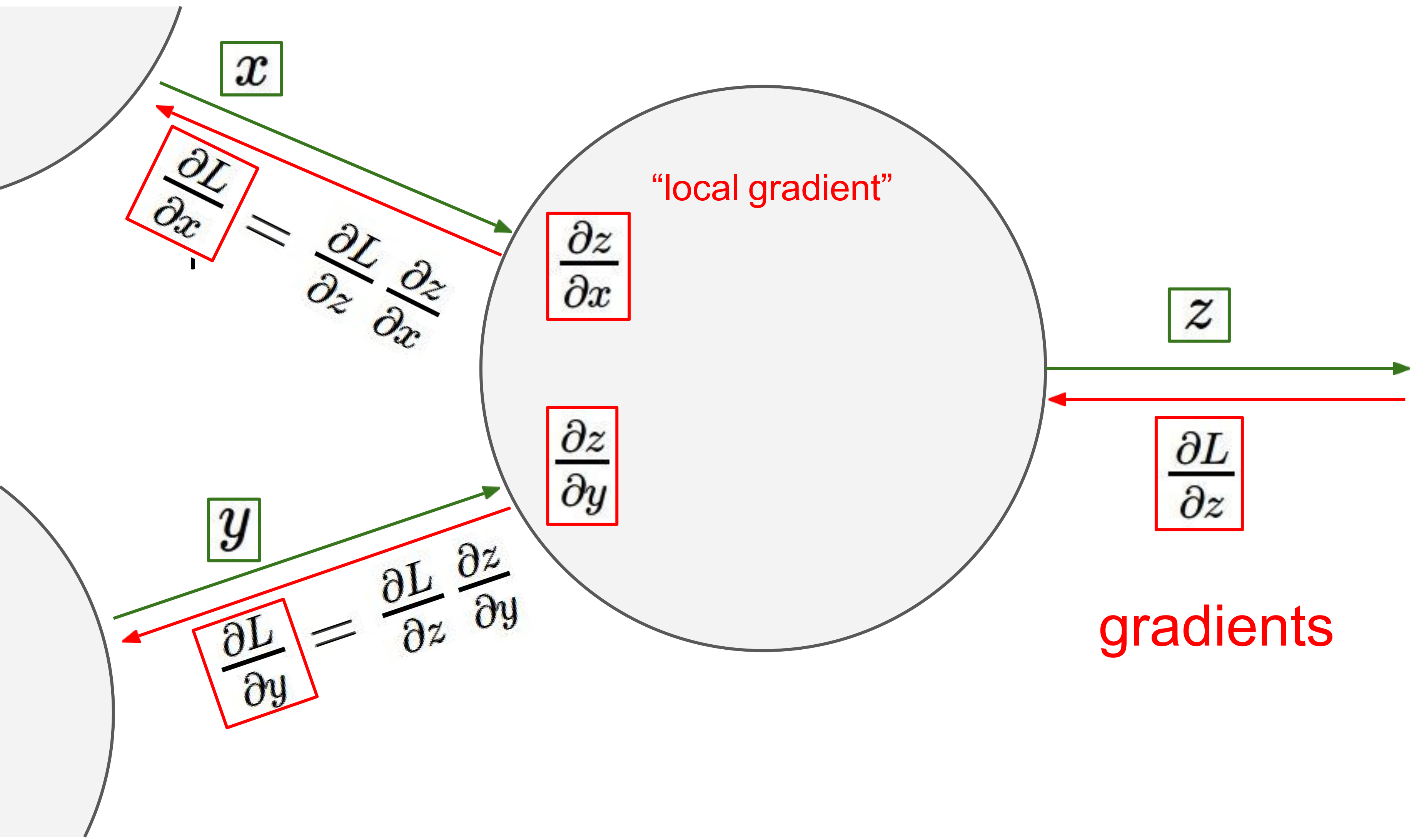


x
f
"local gradient"
$\frac{\partial z}{\partial x}$
z
$\frac{\partial z}{\partial y}$
y


x
"local gradient"
$\frac{\partial z}{\partial x}$
z
$\frac{\partial z}{\partial y}$
$\frac{\partial L}{\partial z}$
y
f
gradients

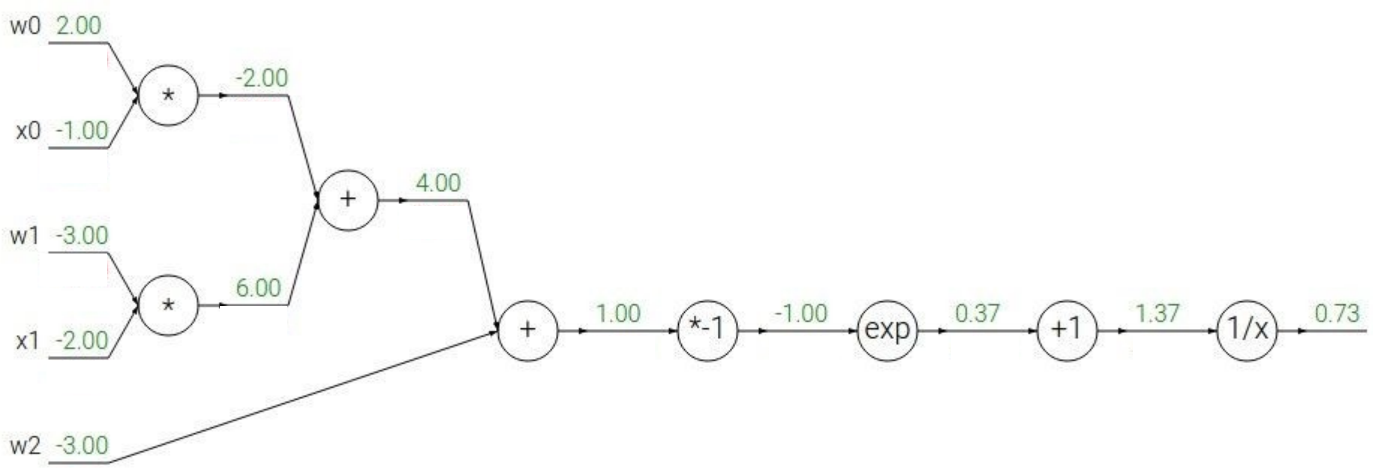


x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
f
gradients


x
y
z
"local gradient"
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
$\frac{\partial L}{\partial z}$
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

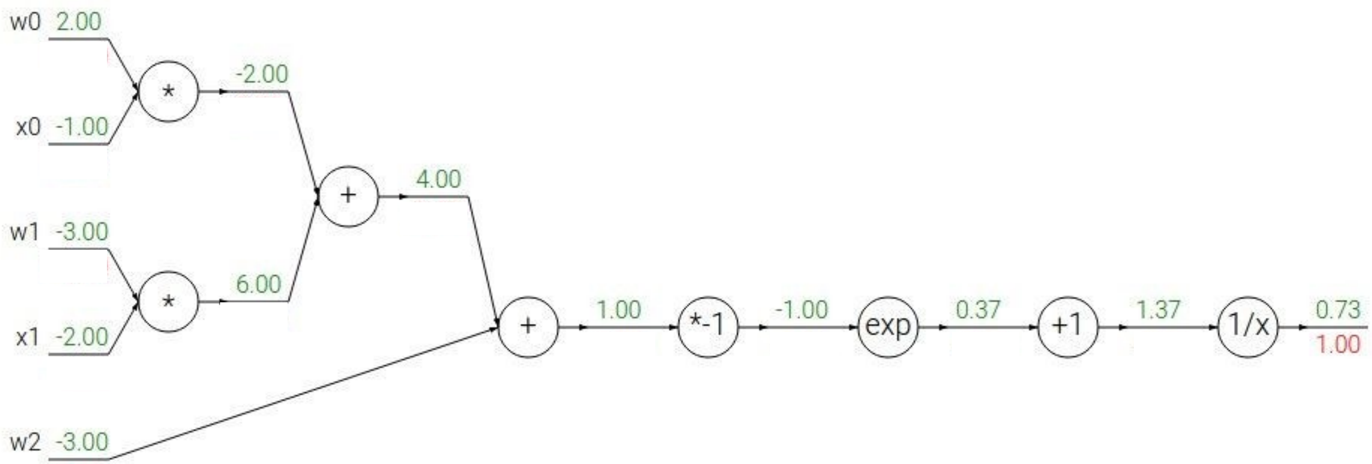
Another example:

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



Calculate the gradient of the LOSS with respect to w0, x0, w1, x1 and w2.

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



$$f(x)=e^x \quad \rightarrow \quad \frac{df}{dx}=e^x$$

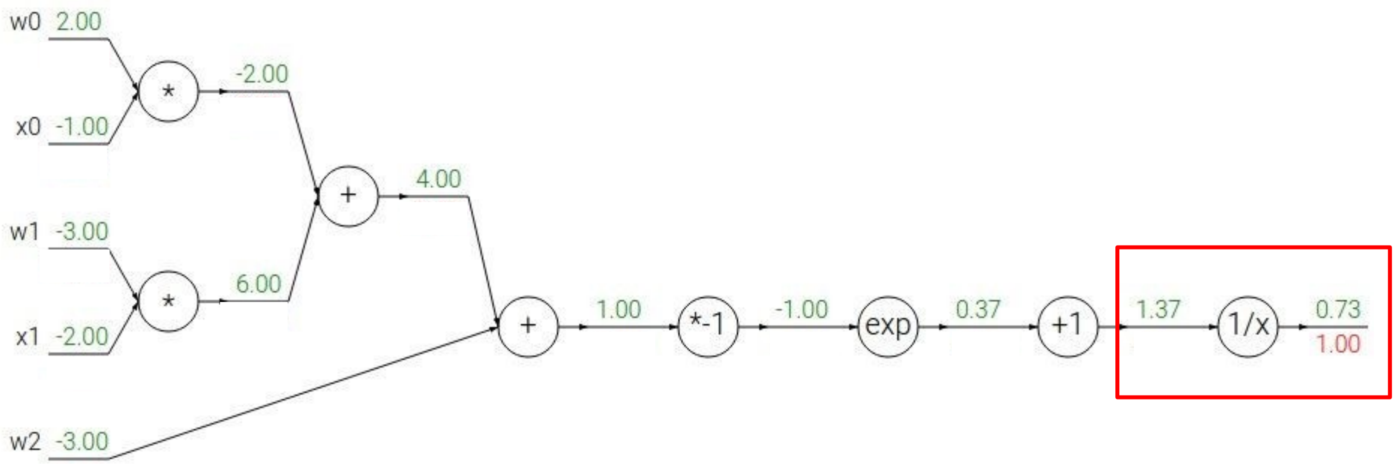
$$f_a(x)=ax \quad \rightarrow \quad \frac{df}{dx}=a$$

$$f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$$

$$f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$$

Another example:

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



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

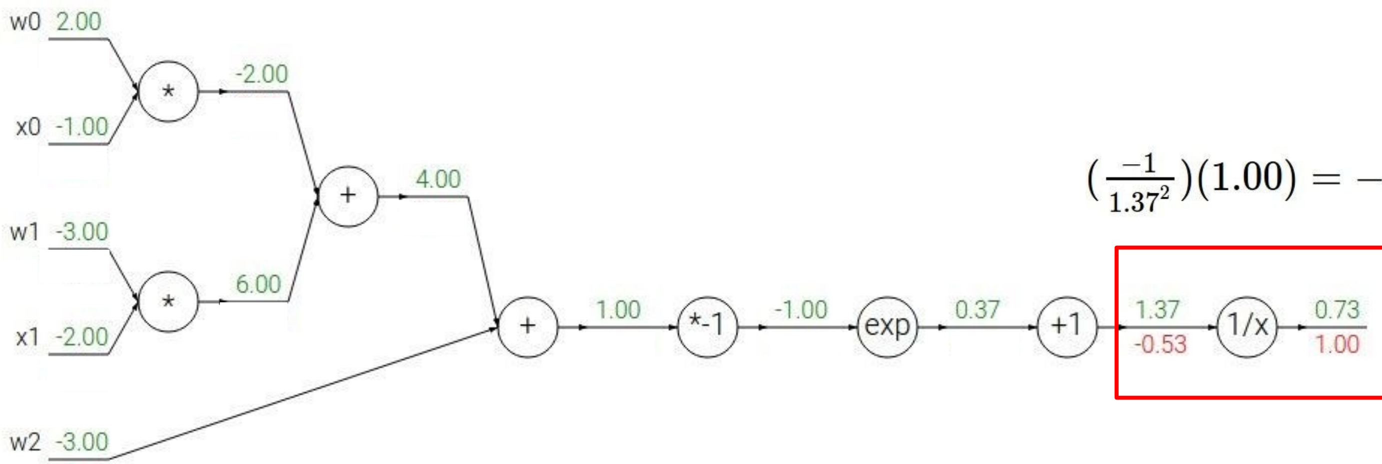
$$\boxed{f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2}$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another example:

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

$$\left(\frac{-1}{1.37^2}\right)(1.00) = -0.53$$



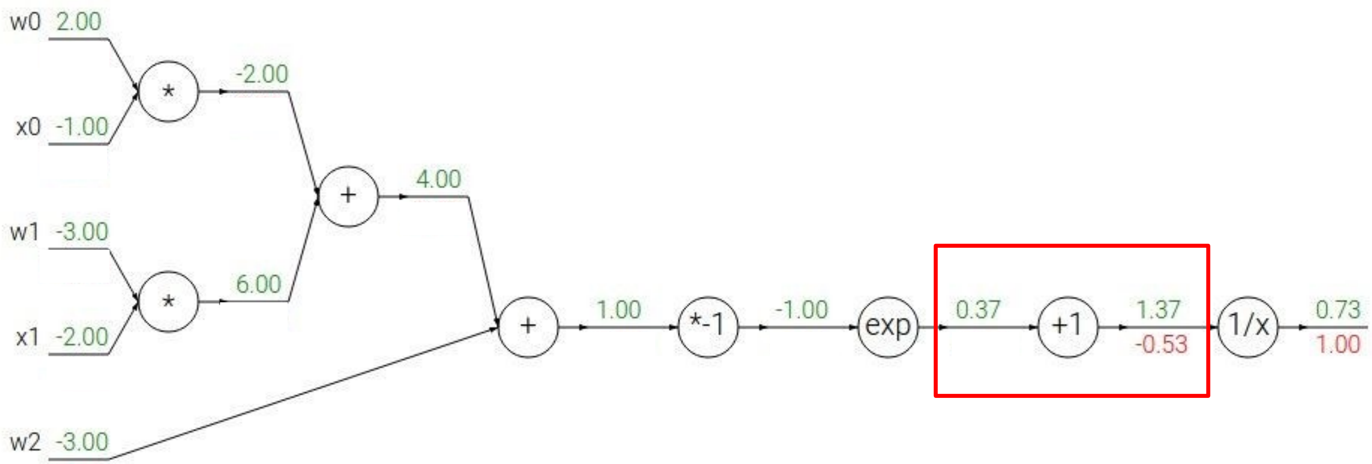
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$\boxed{f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2}$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

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



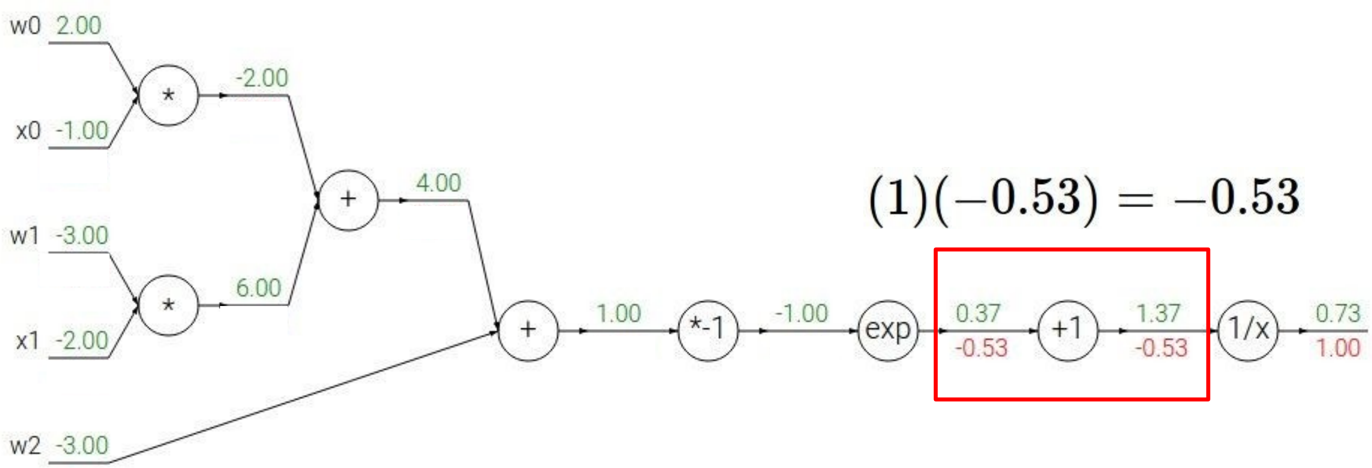
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$\boxed{f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1}$$

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



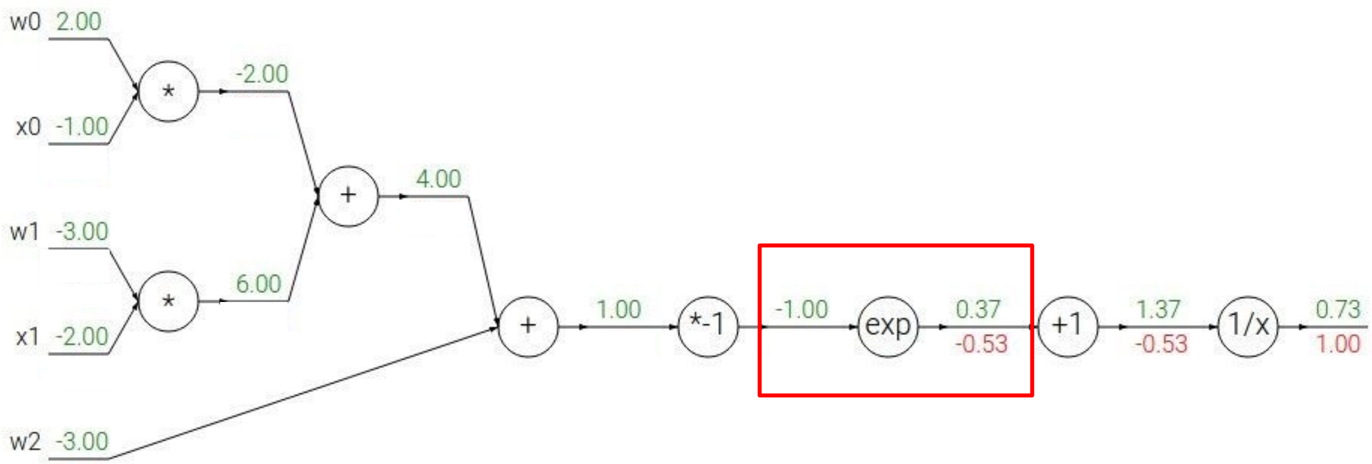
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$\boxed{f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1}$$

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



$$\boxed{f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x}$$

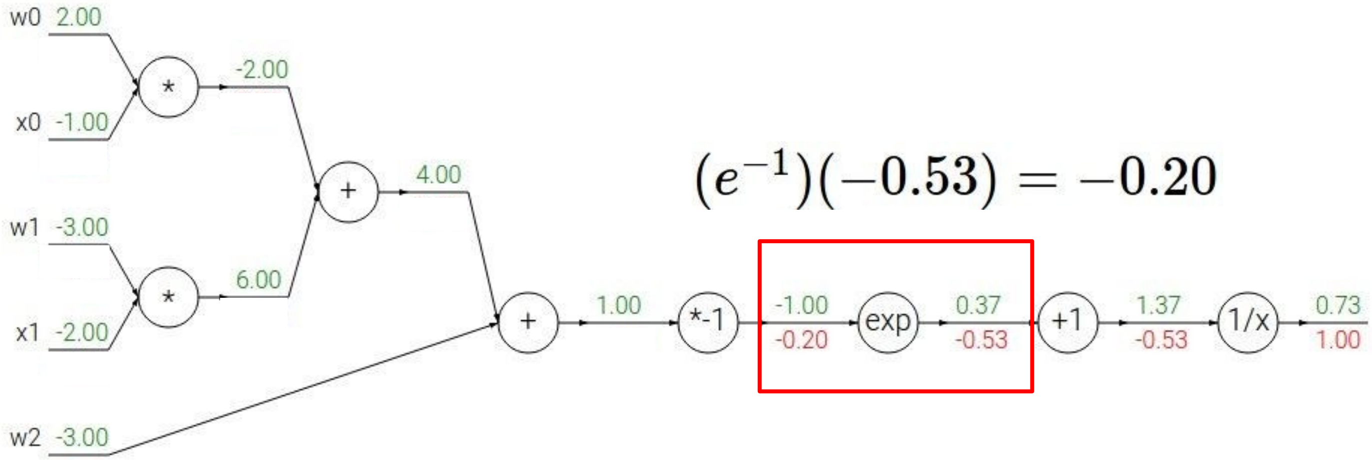
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another example:

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



$$(e^{-1})(-0.53) = -0.20$$

$$\boxed{f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x}$$

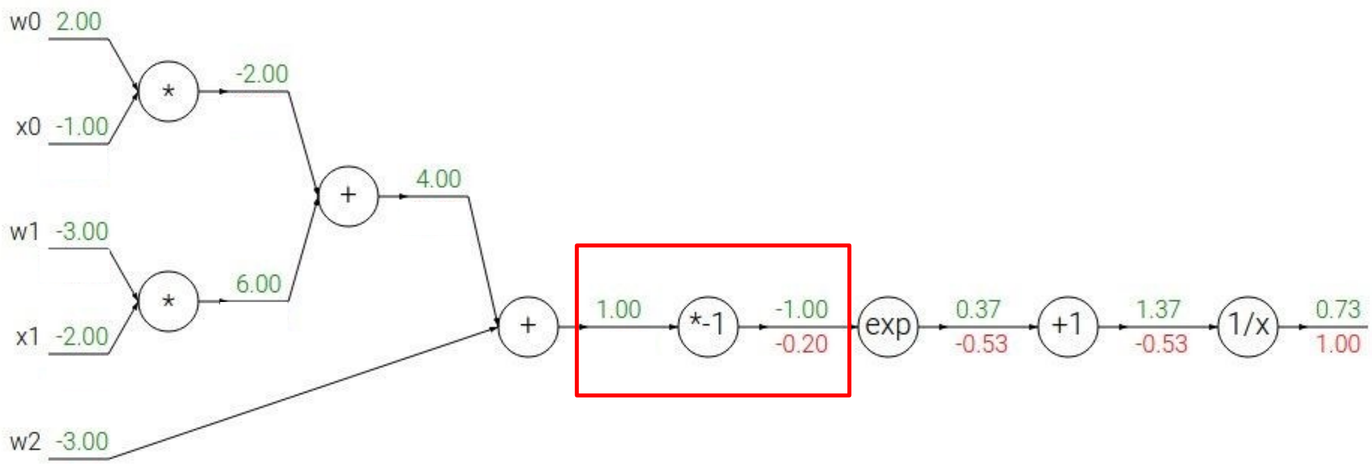
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another example:

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



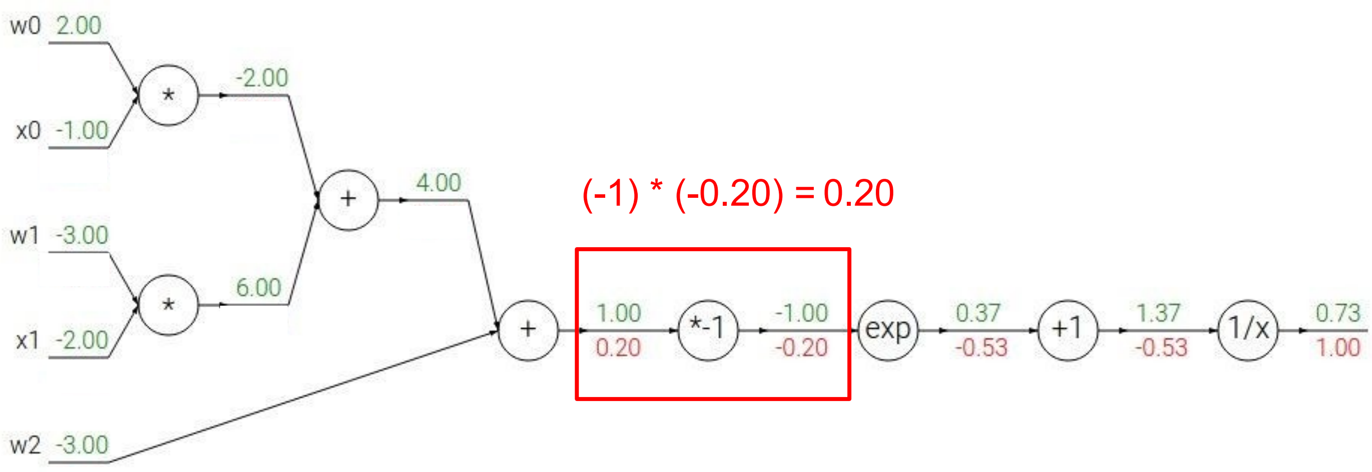
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$\boxed{f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a}$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

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



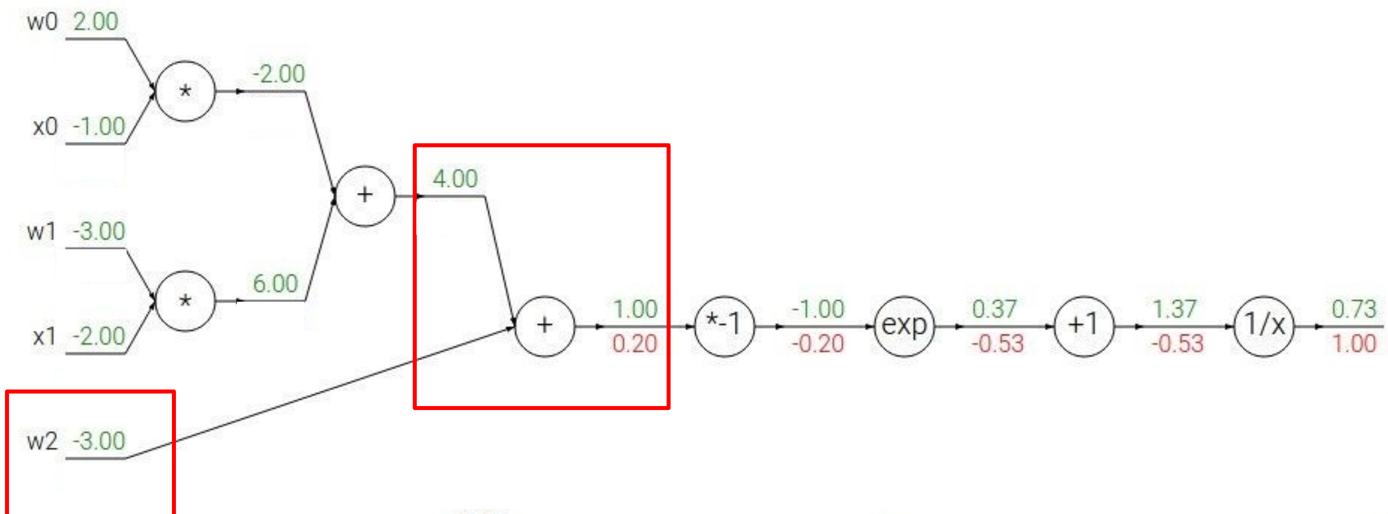
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$\boxed{f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a}$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another example: $f(w,x) = \dfrac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



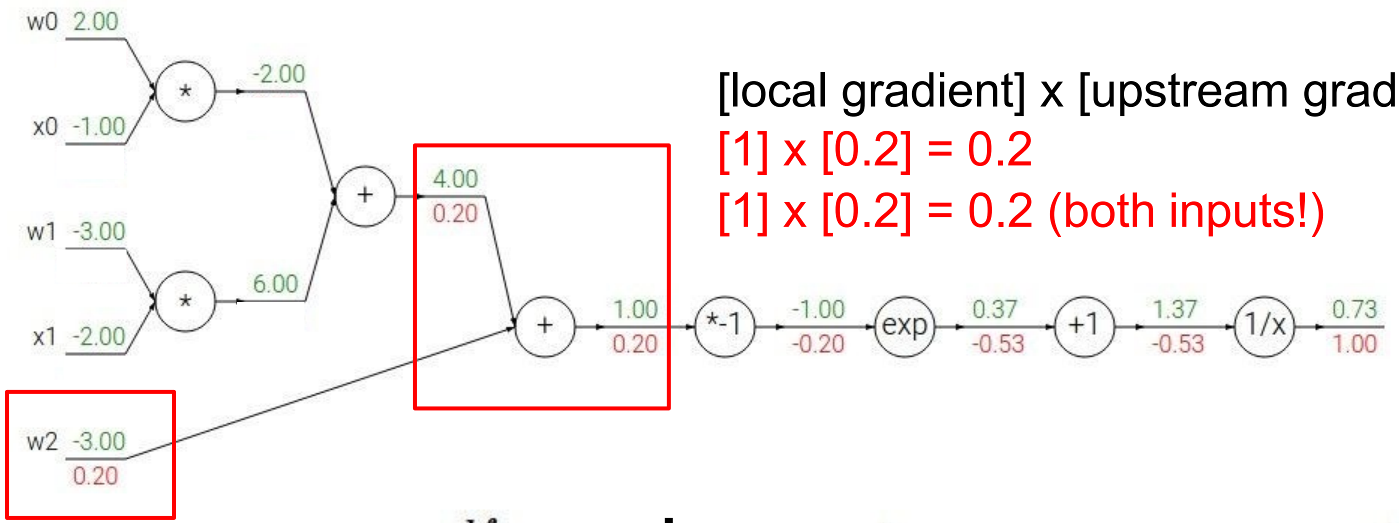
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

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



[local gradient] x [upstream gradient]

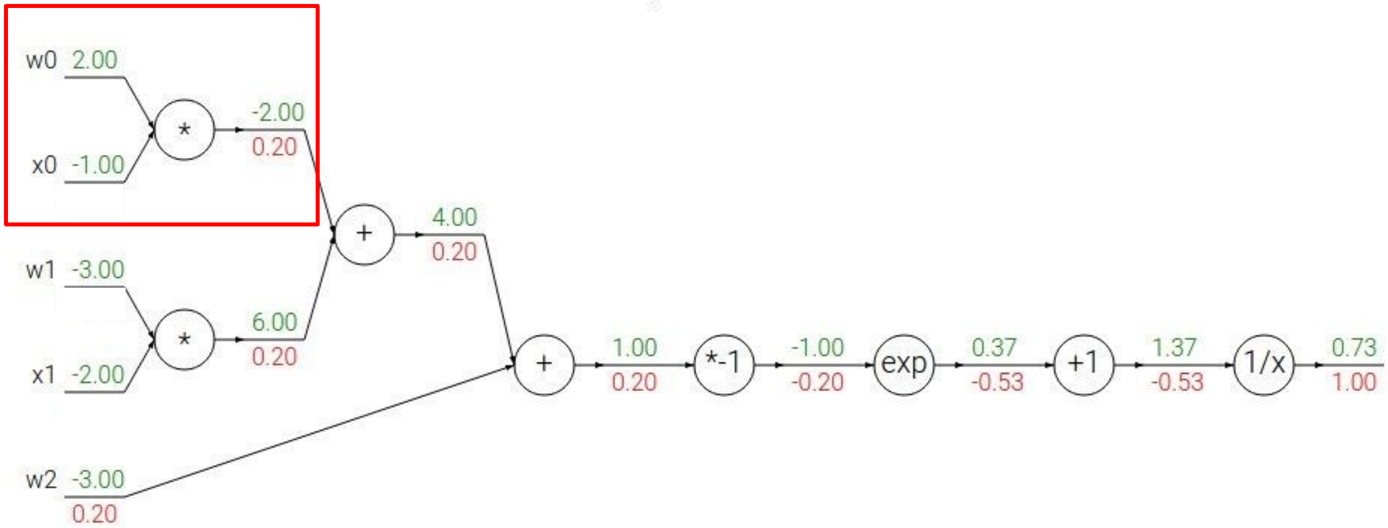
[1] x [0.2] = 0.2

[1] x [0.2] = 0.2 (both inputs!)

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \qquad\qquad f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \qquad\qquad f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

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



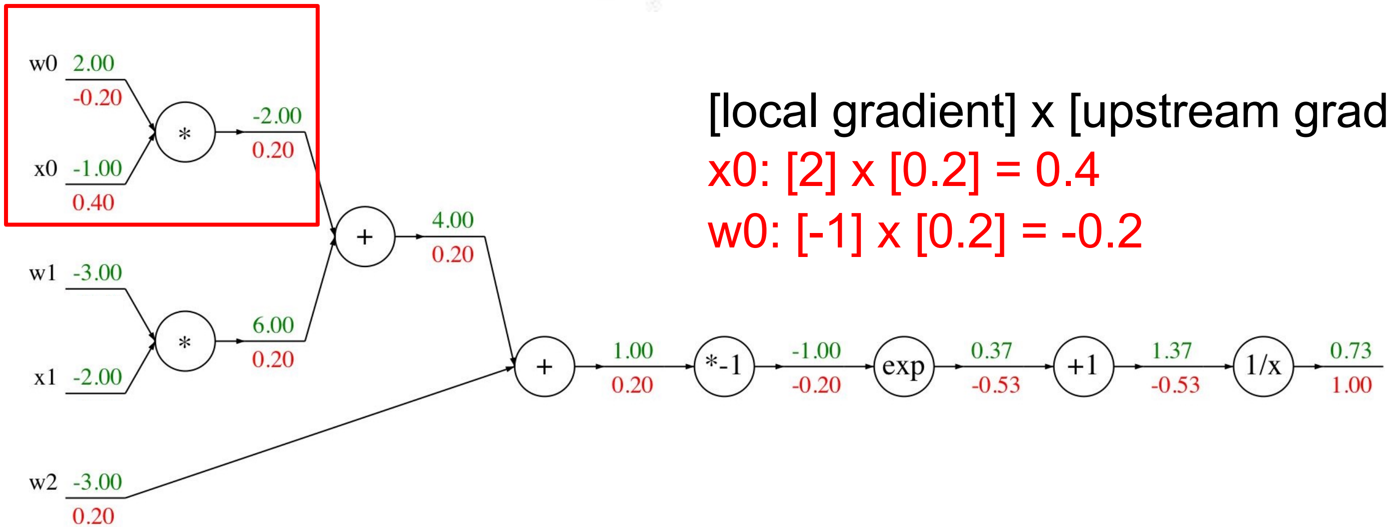
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

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



[local gradient] x [upstream gradient]

x0: [2] x [0.2] = 0.4

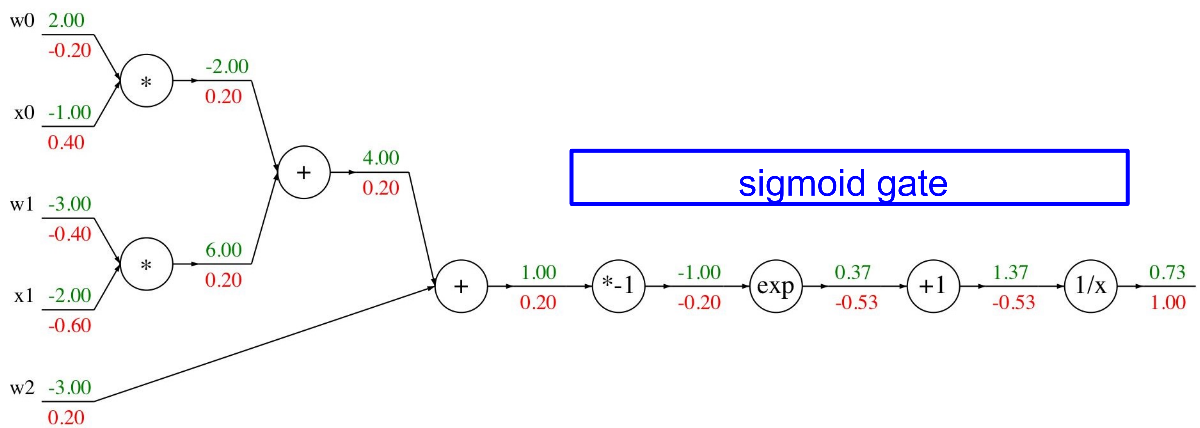
w0: [-1] x [0.2] = -0.2

$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$
$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$	$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

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

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

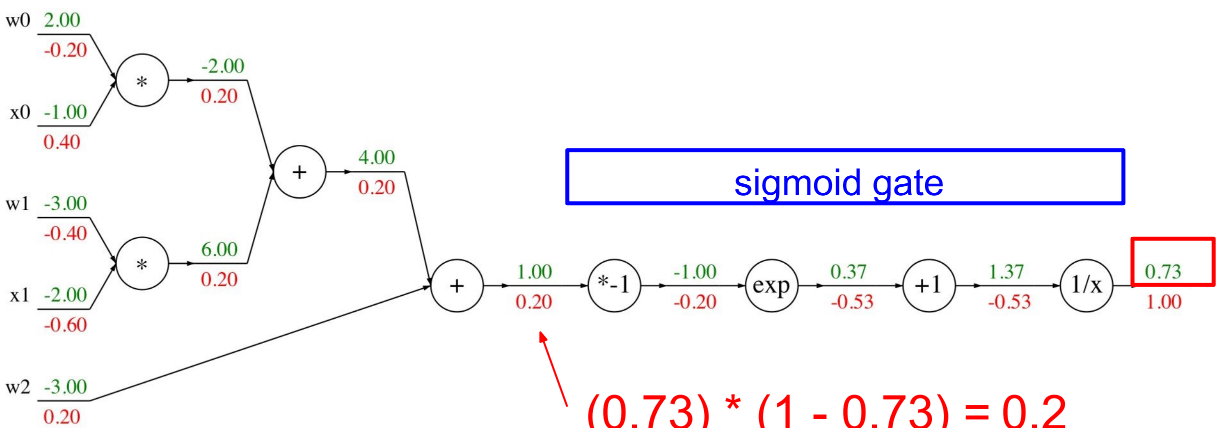
$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\,\sigma(x)$$



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

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

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



(0.73) * (1 - 0.73) = 0.2

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, … , 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

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

where we define the net input z as

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \sum_{l=0}^{m} w_l x_l = \mathbf{w}^T \mathbf{x}.$$

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:

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

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

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:

$$\Pr(Y_i = 0) = \frac{e^{\boldsymbol{\beta}_0 \cdot \mathbf{X}_i}}{\sum_{0 \le c \le K} e^{\boldsymbol{\beta}_c \cdot \mathbf{X}_i}} = \frac{e^{\boldsymbol{\beta}_0 \cdot \mathbf{X}_i}}{e^{\boldsymbol{\beta}_0 \cdot \mathbf{X}_i} + e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i}} = \frac{e^{(\boldsymbol{\beta}_0 - \boldsymbol{\beta}_1) \cdot \mathbf{X}_i}}{e^{(\boldsymbol{\beta}_0 - \boldsymbol{\beta}_1) \cdot \mathbf{X}_i} + 1} = \frac{e^{-\boldsymbol{\beta} . \mathbf{X}_i}}{1 + e^{-\boldsymbol{\beta} \cdot \mathbf{X}_i}}$$

$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i}}{\sum_{0 \le c \le K} e^{\boldsymbol{\beta}_c \cdot \mathbf{X}_i}} = \frac{e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i}}{e^{\boldsymbol{\beta}_0 \cdot \mathbf{X}_i} + e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i}} = \frac{1}{e^{(\boldsymbol{\beta}_0 - \boldsymbol{\beta}_1) \cdot \mathbf{X}_i} + 1} = \frac{1}{1 + e^{-\boldsymbol{\beta} . \mathbf{X}_i}}$$

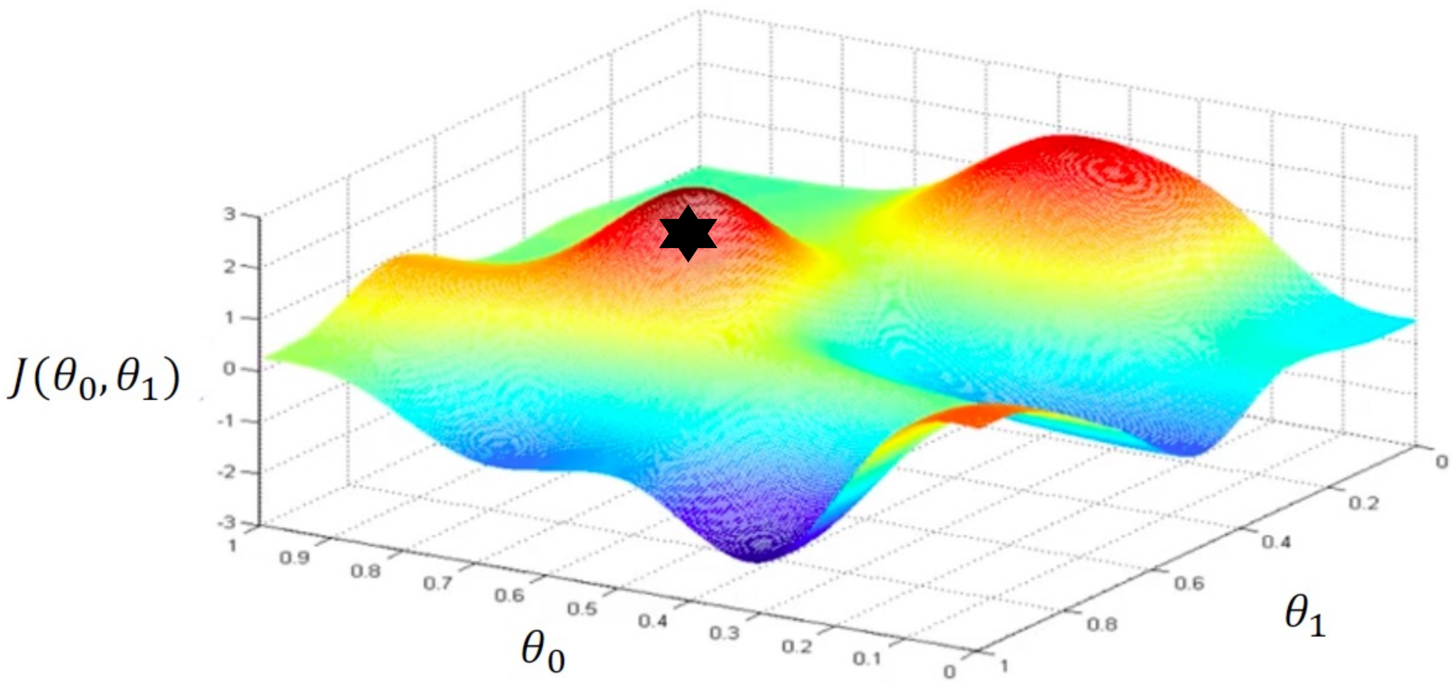
where $\boldsymbol{\beta} = -(\boldsymbol{\beta}_0 - \boldsymbol{\beta}_1)$

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

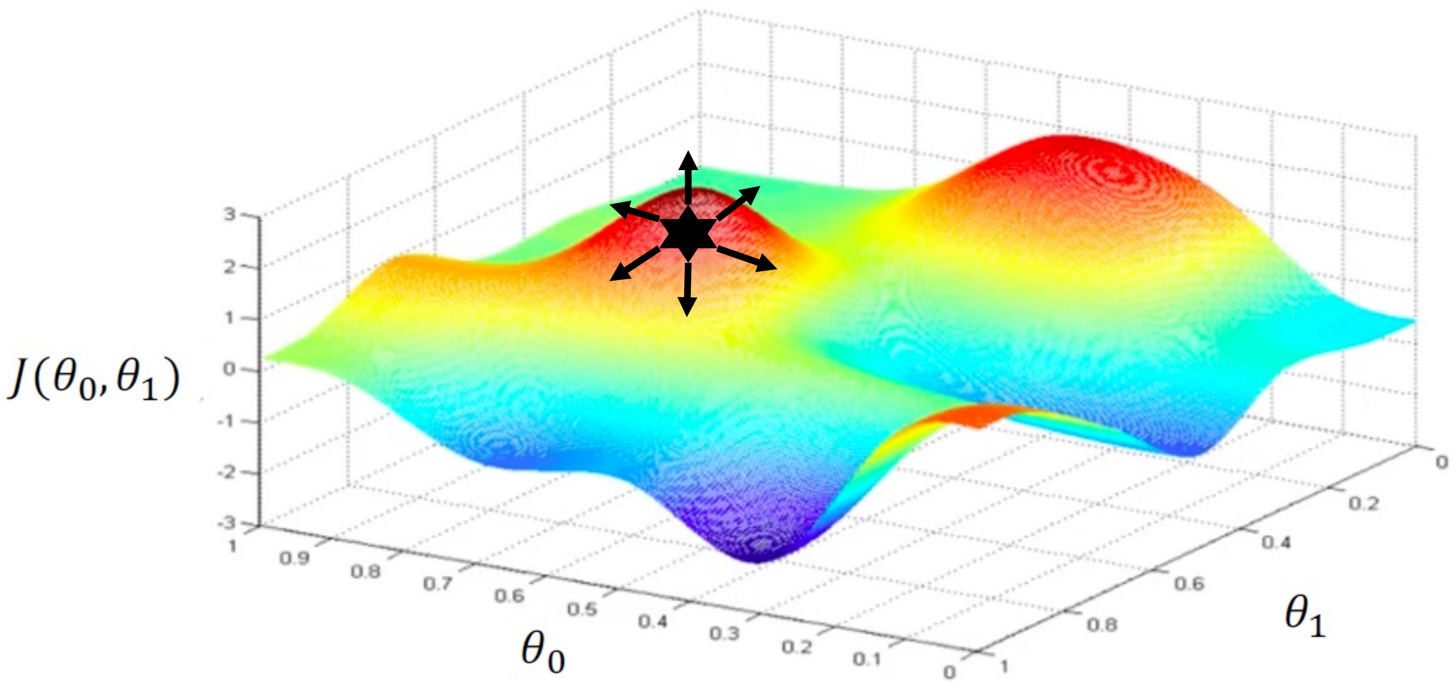
Gradient Descent

- Standard approach for learning weights



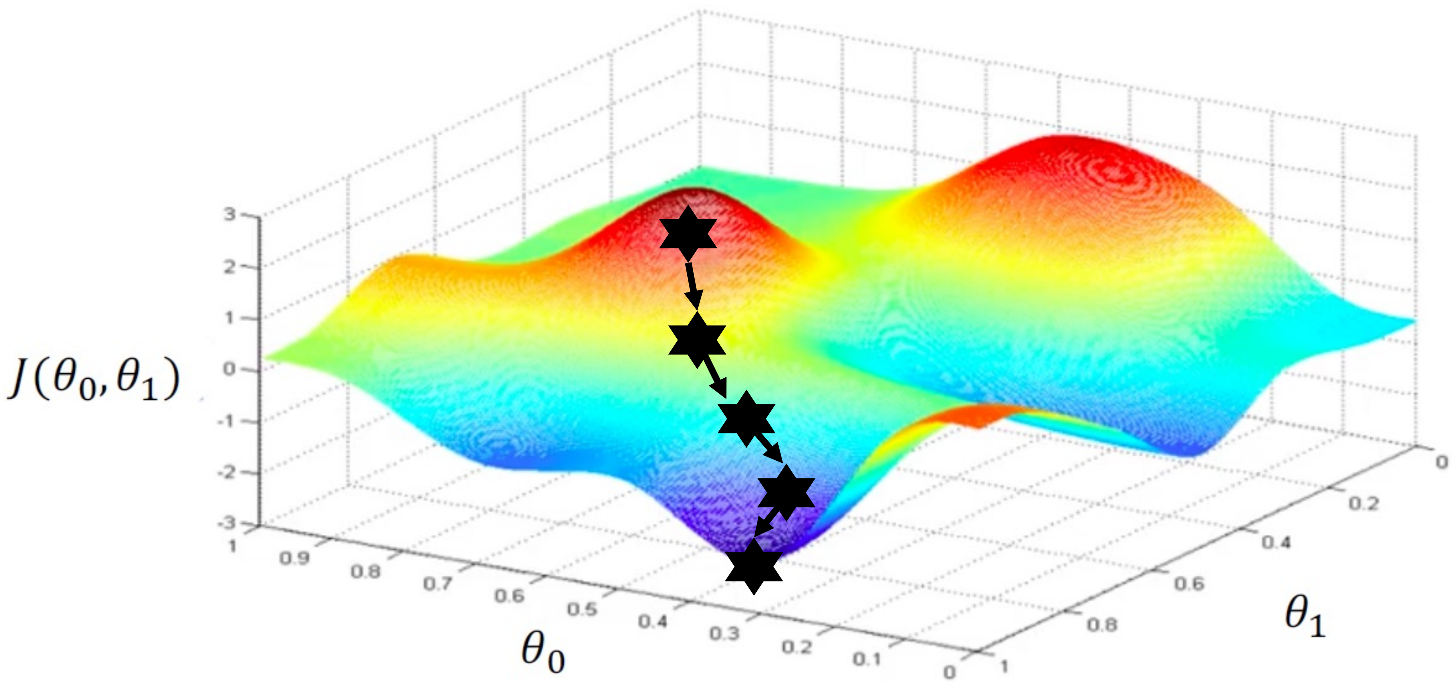
Gradient Descent

- Standard approach for learning weights



Gradient Descent

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



Exercise 1: Gradient Descent

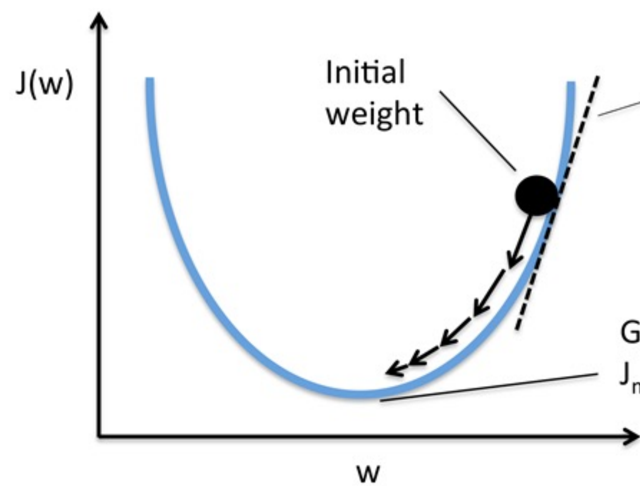
- What are the effects of learning rates on gradient descent?

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

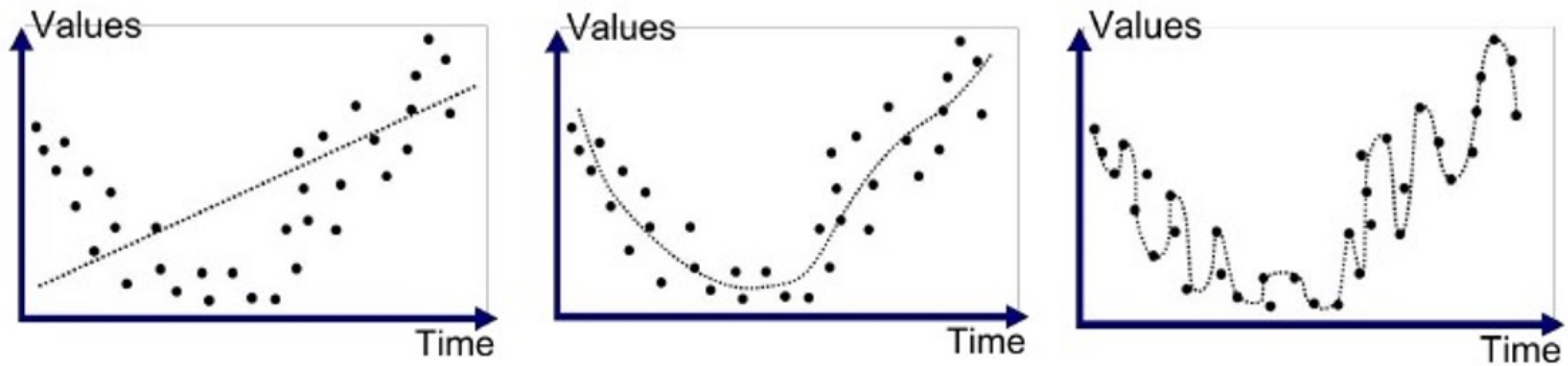


https://wiki.tum.de/display/lfdv/Adaptive+Learning+Rate+Method

Adaptive Learning Rates

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

Under/Over-fitting Problem



Underfitted Good Fit/Robust Overfitted

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

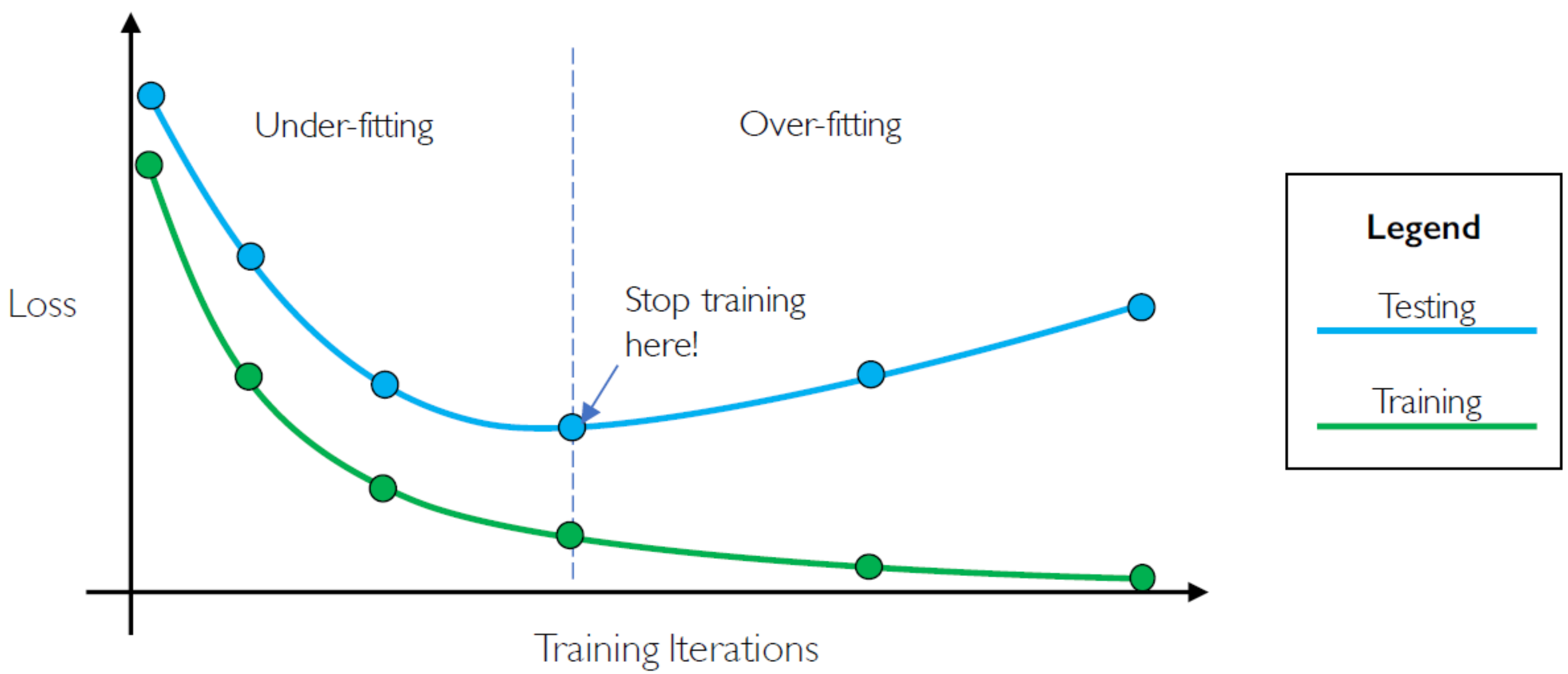
Regularization

- 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?

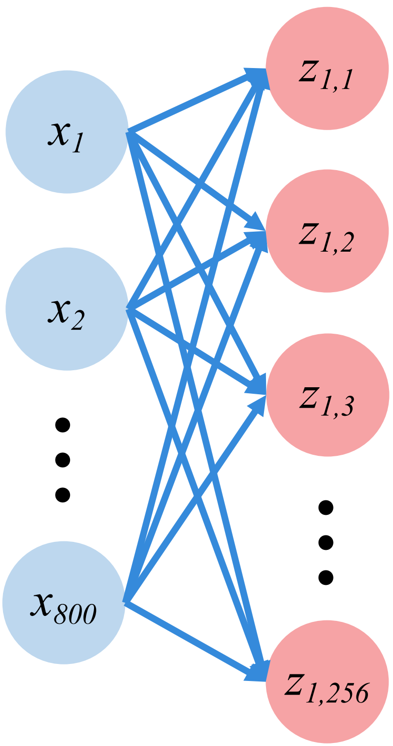
Another ***fantastic*** entry to the ***best*** ***action*** franchise, putting James Bond films to ***shame*** once again. While the beginning is still a little ***bumpy***, by the middle of the movie it is developing into a maelstrom of ***excitement*** and wonder about how ***action*** scenes are thought of, directed and performed. The Paris chase is already pretty ***fantastic*** but the last 30 minutes top it all off, pushing the envelope with every minute and twist. Cruise keeps risking his health for his audience, I know I am on board. ***Perfect*** ***action*** entertainment.

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

Vocabulary = 800 Words

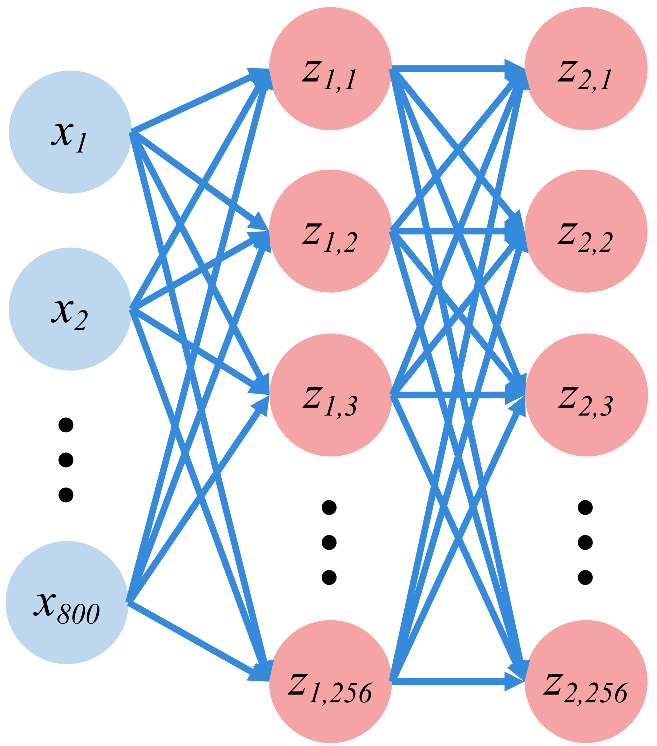
Classes = {V.Good, Good, Neutral, Bad, V.Bad}

Keras Example: NLP Application



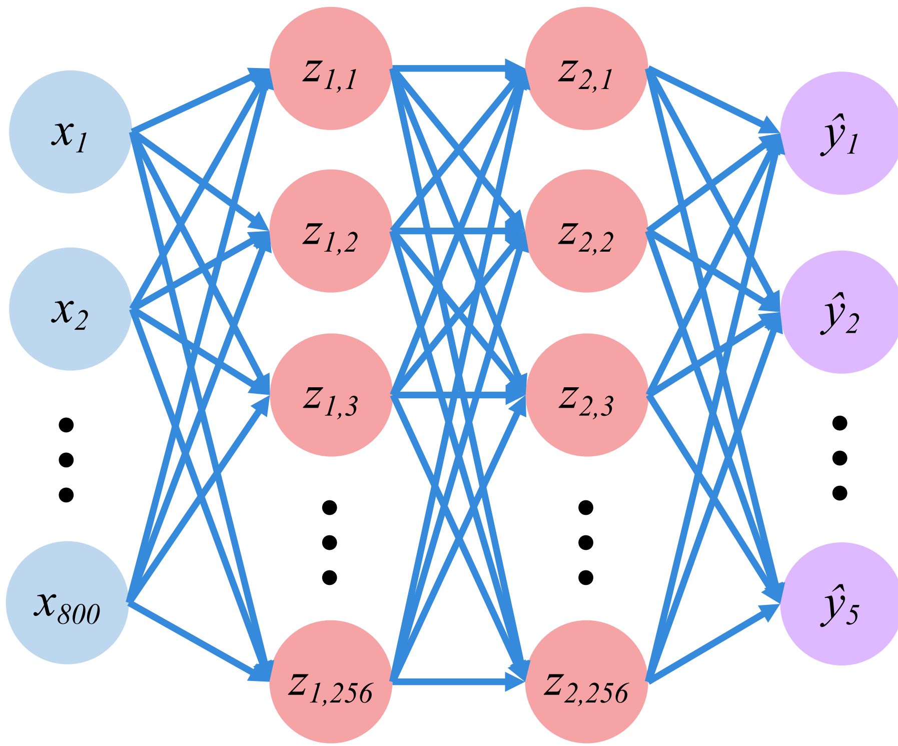

```python
model = Sequential()

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

Keras Example: NLP Application



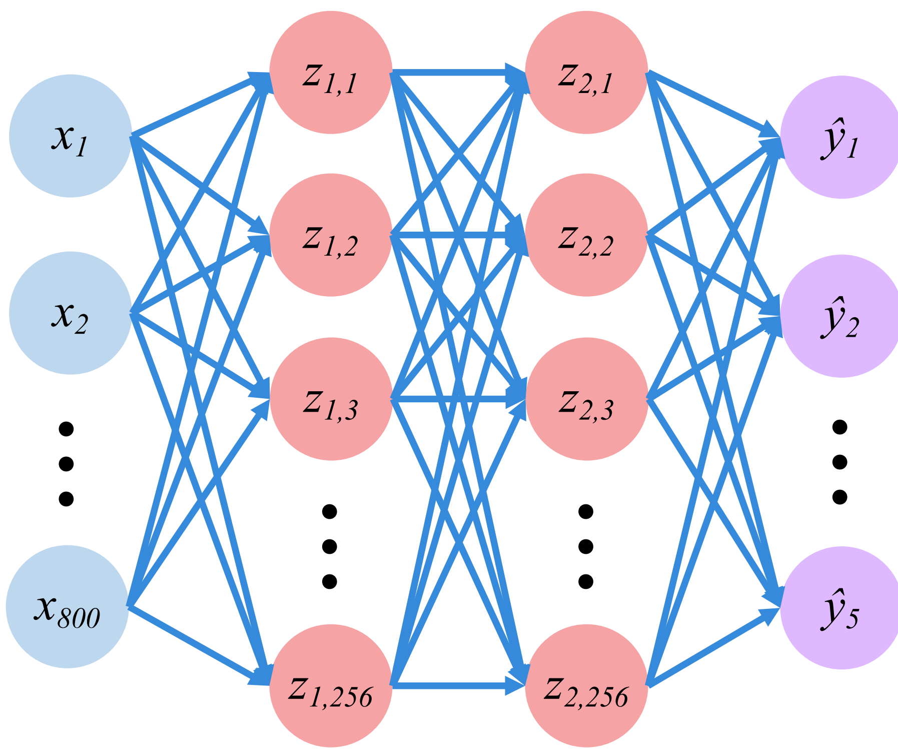

```
model = Sequential()

# input layer and hidden layer 1
model.add(Dense(256, activation='relu', input_dim=800))
model.add(Dropout(0.5))

# hidden layer 2
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
```

Keras Example: NLP Application



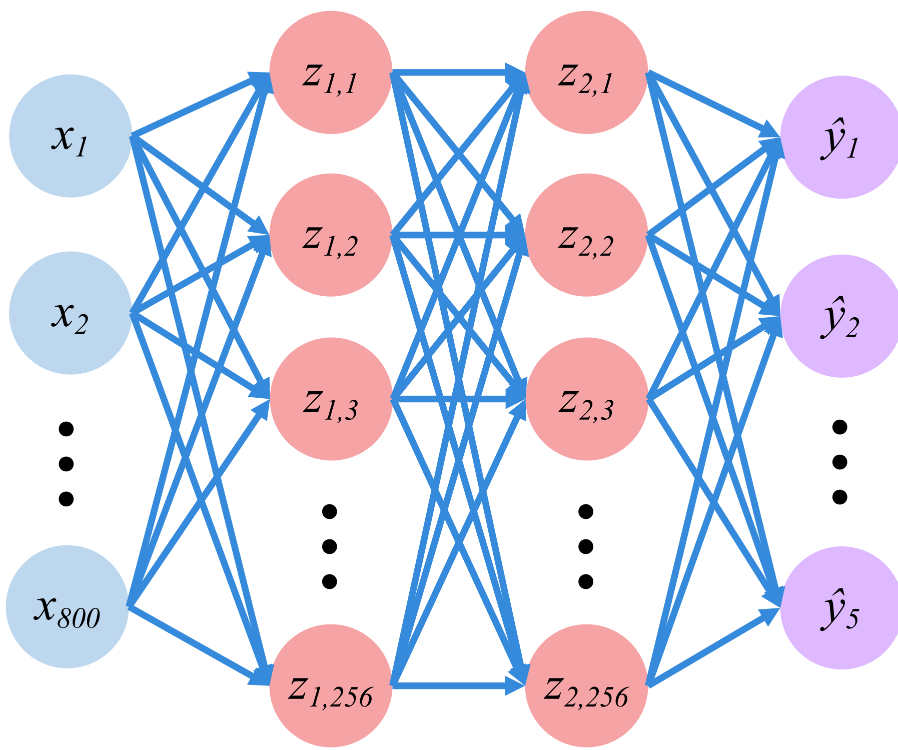

```python
model = Sequential()

# input layer and hidden layer 1
model.add(Dense(256, activation='relu', input_dim=800))
model.add(Dropout(0.5))

# hidden layer 2
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

# output layer
model.add(Dense(5, activation='softmax'))
```

Keras Example: NLP Application



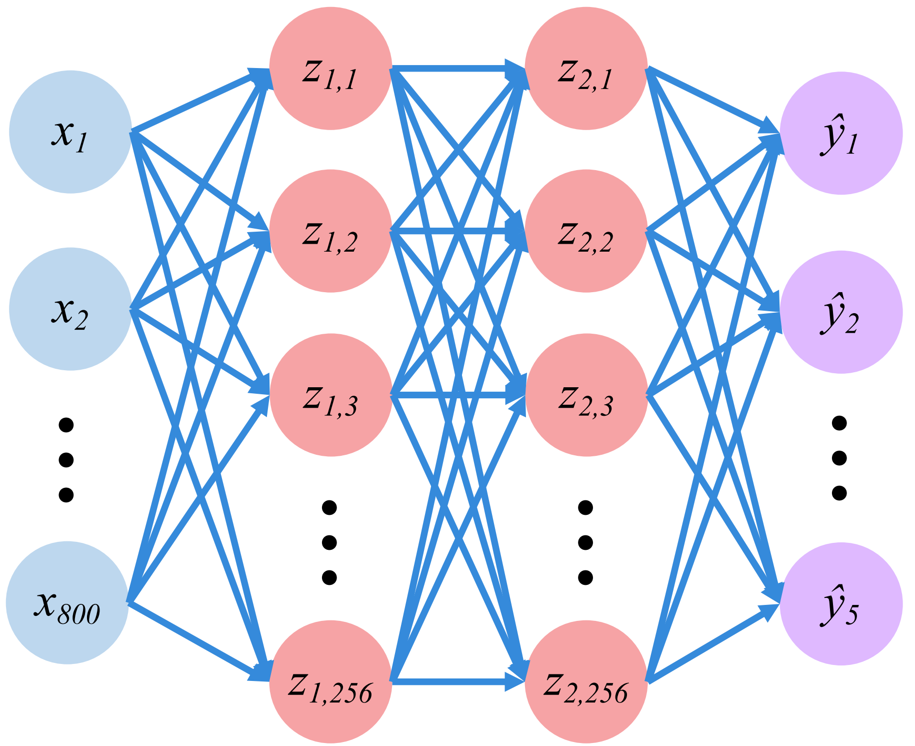

```
model = Sequential()

# input layer and hidden layer 1
model.add(Dense(256, activation='relu', input_dim=800))
model.add(Dropout(0.5))

# hidden layer 2
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

# output layer
model.add(Dense(5, activation='softmax'))

# configure learning process
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Keras Example: NLP Application




```
model = Sequential()

# input layer and hidden layer 1
model.add(Dense(256, activation='relu', input_dim=800))
model.add(Dropout(0.5))

# hidden layer 2
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

# output layer
model.add(Dense(5, activation='softmax'))

# configure learning process
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# train our model
model.fit(x_train, y_train,
          epochs=50,
          batch_size=128)
```

Keras Example: NLP Application




```python
model = Sequential()

# input layer and hidden layer 1
model.add(Dense(256, activation='relu', input_dim=800))
model.add(Dropout(0.5))

# hidden layer 2
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

# output layer
model.add(Dense(5, activation='softmax'))

# configure learning process
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# train our model
model.fit(x_train, y_train,
          epochs=50,
          batch_size=128)

# test our model
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?