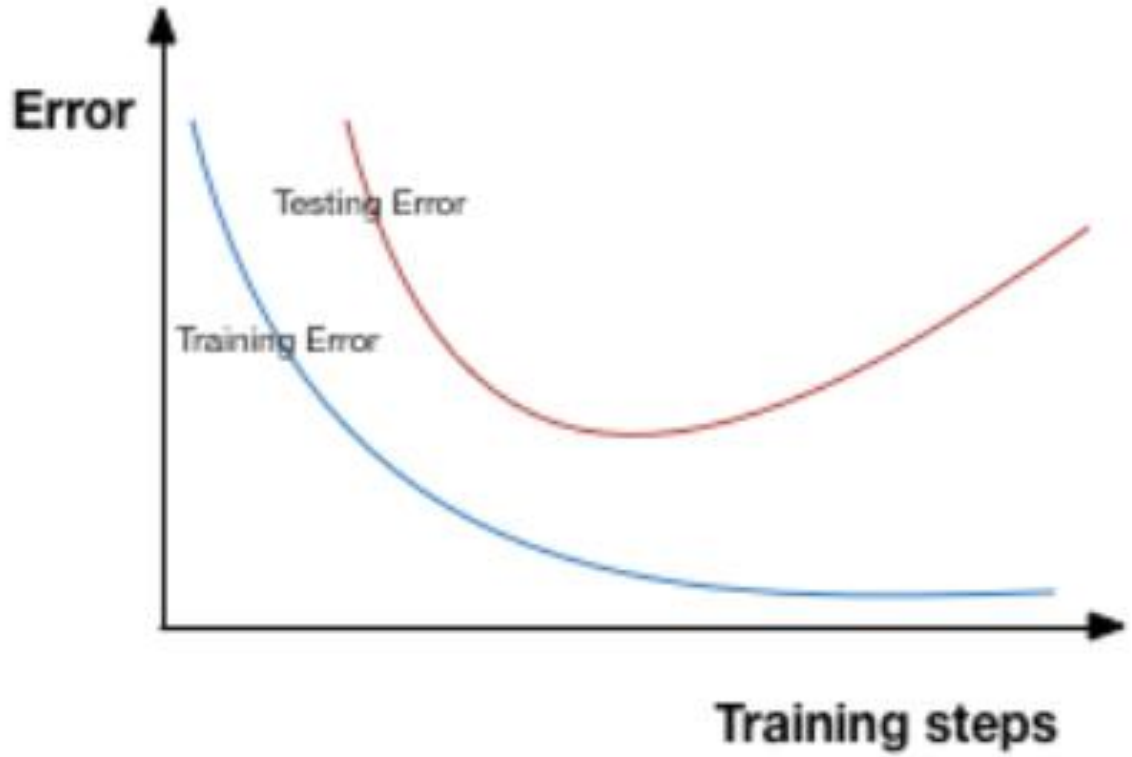
Session 4

- Recap of Session 3
- Build your own Alexa skills
- Pandas
- Intro to Deep Learning

Session 4

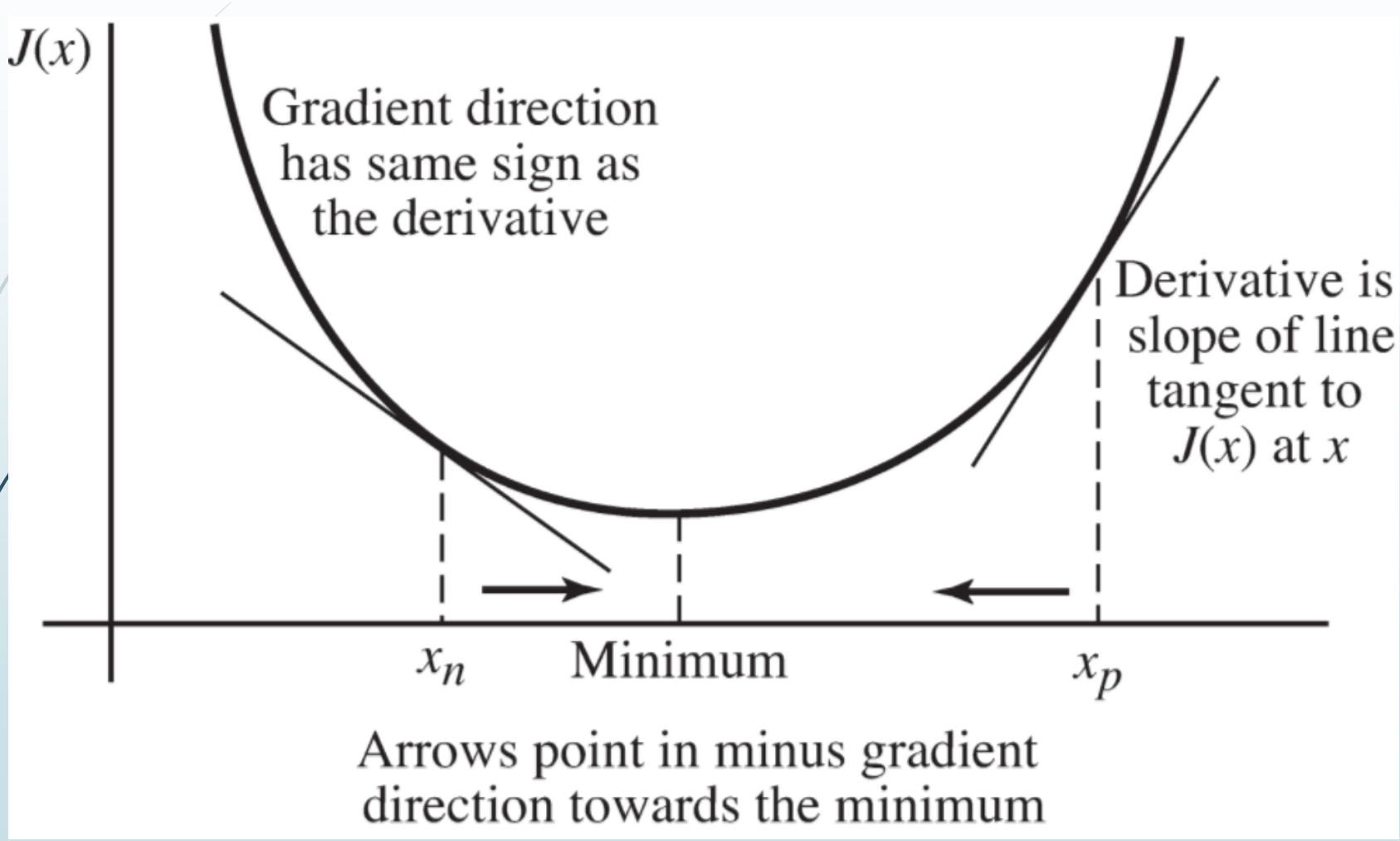
- Recap of Session 3
- Build your own Alexa skills
- Pandas
- Intro to Deep Learning

Training and Testing Error



Overfitting symptom: Testing error >> Training error

Gradient Descent - concept



Gradient Descent - ALgorithm

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \text{(for } j = 0 \text{ and } j = 1\text{)}$$

}

Correct: Simultaneous update

$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

$\theta_0 := \text{temp0}$

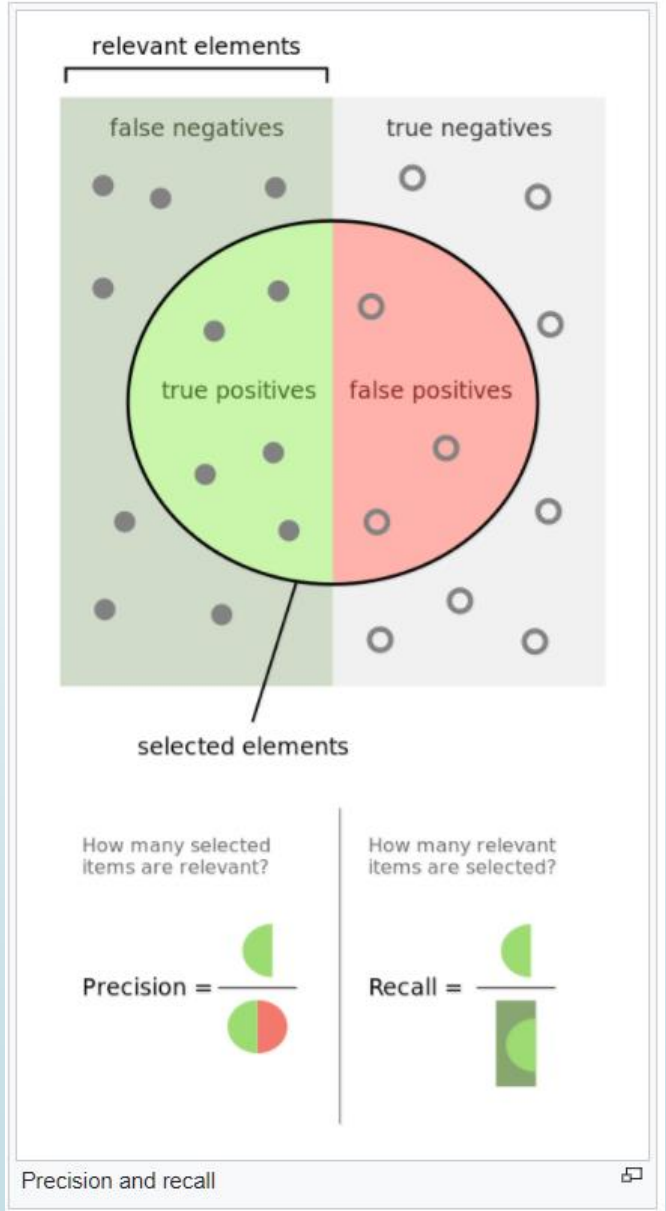
$\theta_1 := \text{temp1}$

Classification - F1 score

Confusion Matrix

		Actual class	
		Cat	**Non-cat**
Predicted class	**Cat**	5 True Positives	2 False Positives
	Non-cat	3 False Negatives	17 True Negatives

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$



Implementing in Scikit

- We saw (Regression, logistic regression, Decision tree, Random Forest, SVM)
- All have same interface in scikit
  - Step1: instantiate a model
  - Step2: "fit" X^{train} and y^{train}
  - Step3: Check model (accuracy, F1 score) using "predict" on X^{test} and comparing predictions with y^{test}
  - Step4: Model is ready to be used on new inputs now

Session 4

- Recap of Session 3
- Build your own Alexa skills
- Pandas
- Intro to Deep Learning

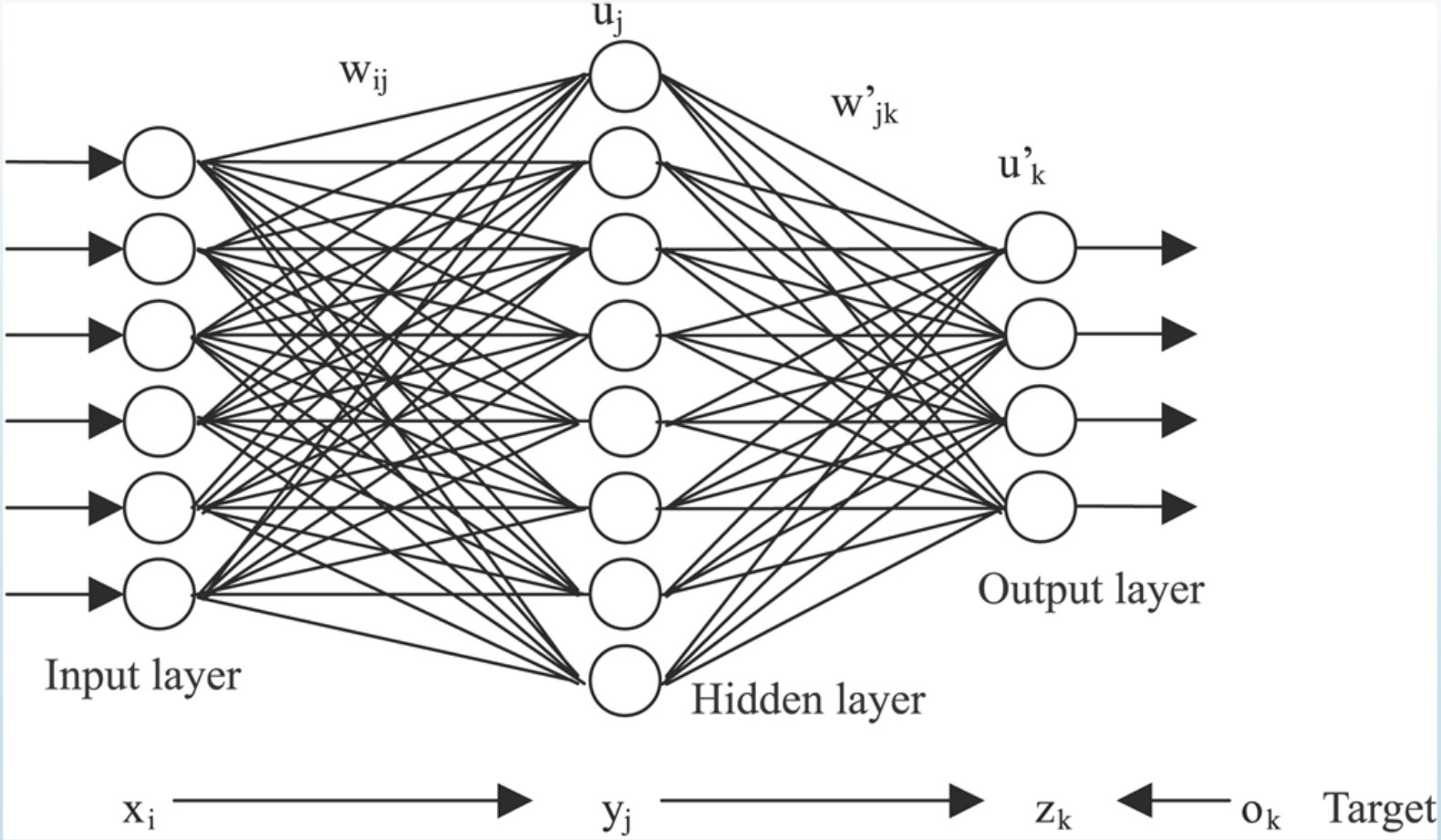
Session 4

- Recap of Session 3
- Build your own Alexa skills
- Pandas
- Intro to Deep Learning

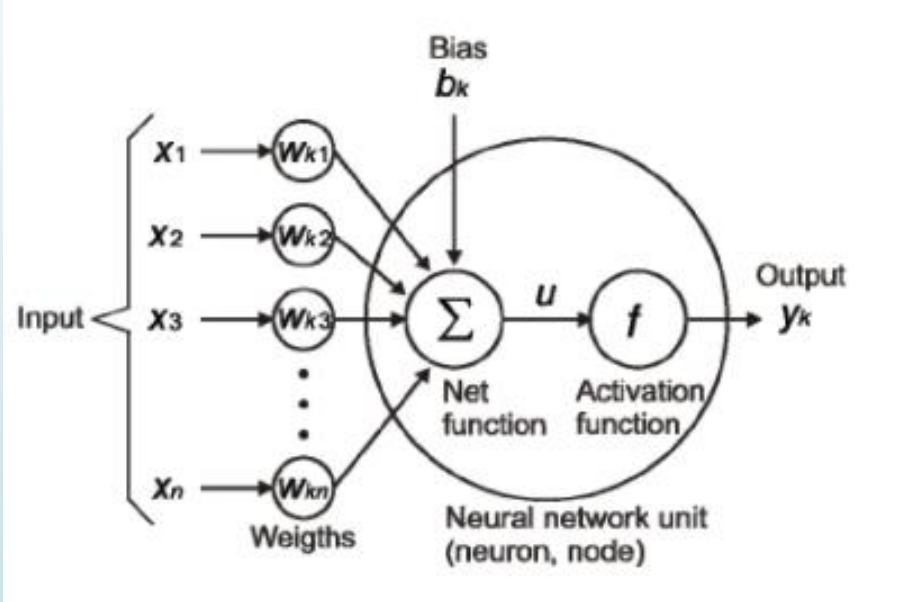
Session 4

- Recap of Session 3
- Build your own Alexa skills
- Pandas
- Intro to Neural Networks / Deep Learning

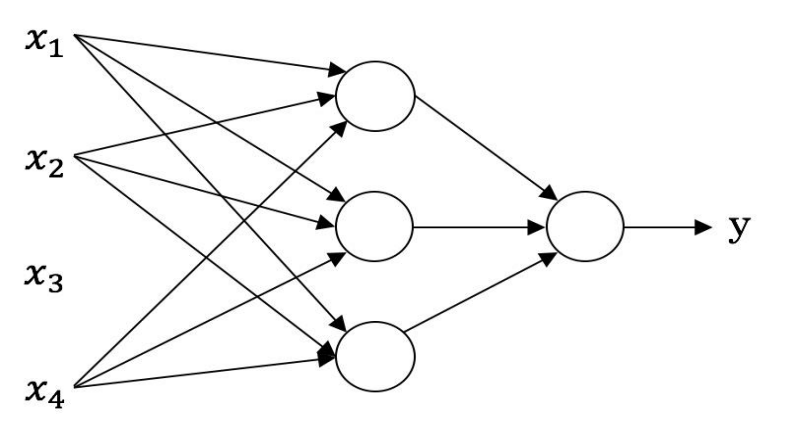
What is a neural network



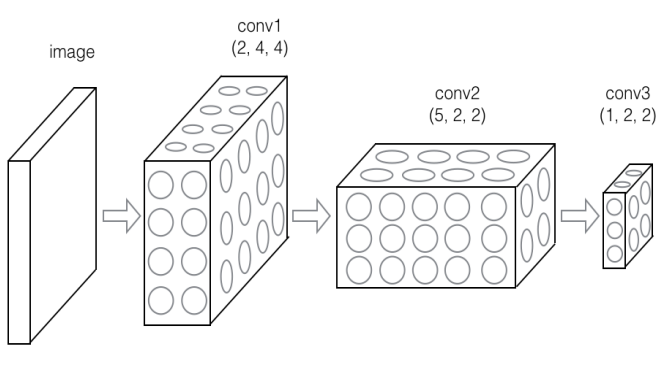
What is a neural network



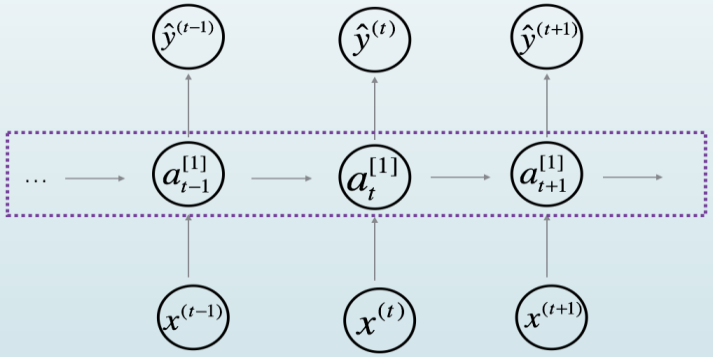
Types of Neural Networks



Standard NN



Convolutional NN



Recurrent NN

Neural Networks

Input(x)	Output (y)	Application
Home features	Price	Real Estate
Ad, user info	Click on ad? (0/1)	Online Advertising
Image	Object (1,…,1000)	Photo tagging
Audio	Text transcript	Speech recognition
English	Chinese	Machine translation
Image, Radar info	Position of other cars	Autonomous driving

Binary Classification

→ 1 (cat) vs 0 (non cat)

Blue
Green
Red

		255	134	93	22
	255	134	202	22	2
255	231	42	22	4	30
123	94	83	2	192	124
34	44	187	92	34	142
34	76	232	124	94	
67	83	194	202		

Our Model of Neural Network

$\hat{y} = \sigma(w^T x + b)$, where $\sigma(z) = \frac{1}{1+e^{-z}}$

Given $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

Loss (error) function: $L(w,b) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} \log(\hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})]$

Learn(adjust w and b) : Gradient Descent using Back Propagation

Gradient Descent

We want to find w and b such that L(w,b) is minimum

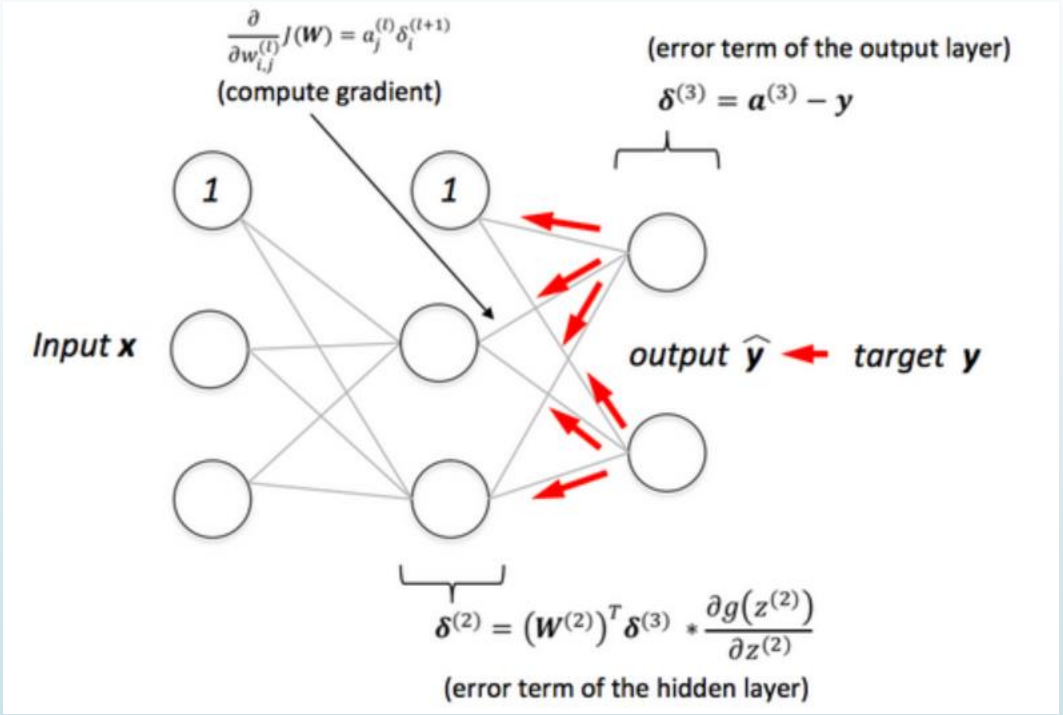
Use gradient Descent:

$$w = w - \alpha . \frac{\partial L(w,b)}{\partial w}$$

$$b = b - \alpha . \frac{\partial L(w,b)}{\partial b}$$

Back Propagation

- Apply chain rule of calculus and propagate error from output layer to input layer.
- It is like a flow graph you flow the error back and try to find the impact of each variable (weight, bias) on the total error
- The most popular algorithm to train neural networks
- "Tensorflow" from Google allows you to build the graph of neural network and apply chain rule with partial derivatives in a systematic and efficient manner.
- This field is also known as "Automatic differentiation"



Tensorflow and Keras

- Tensorflow – open source framework from Google to build and train Neural network models.
- Caffe from Facebook is another very popular framework
- MSFT: Microsoft Cognitive Toolkit—previously known as CNTK
- And many more
  - They are try to do the same – build, train deep learning models but vary in the approach they take.
- Keras: is a higher level framework that can sit on top of Caffe, Tensorflow. It abstarcts the nuts and bolts of underlying framework. Same Keras code will work even if you swap the lower level framework.

Sample of building a neural network using Keras

- We will initially uses Keras which is a higher level library that sits on top of Tensorflow and makes the code very concise and simple. The steps are similar to what we did with Keras
- You can switch to Tensorflow (or mix Keras with tensorlfow code) to have more granular control on the model and various other aspects