Dive Into Deep Learning

C++ Meetup 02/07/2018

Mesosphere 88 Stevenson Street SF

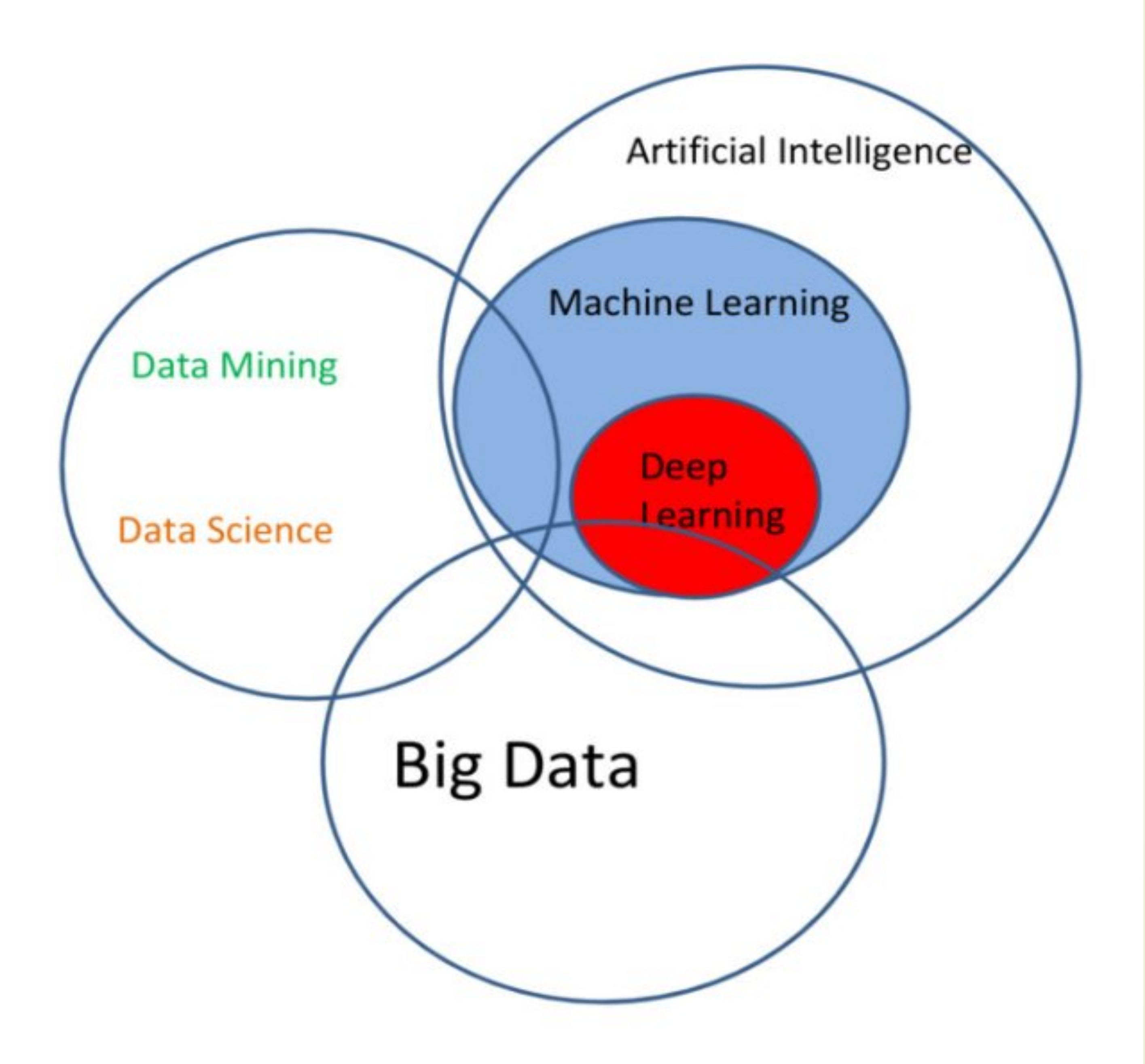
Oswald Campesato

ocampesato@yahoo.com

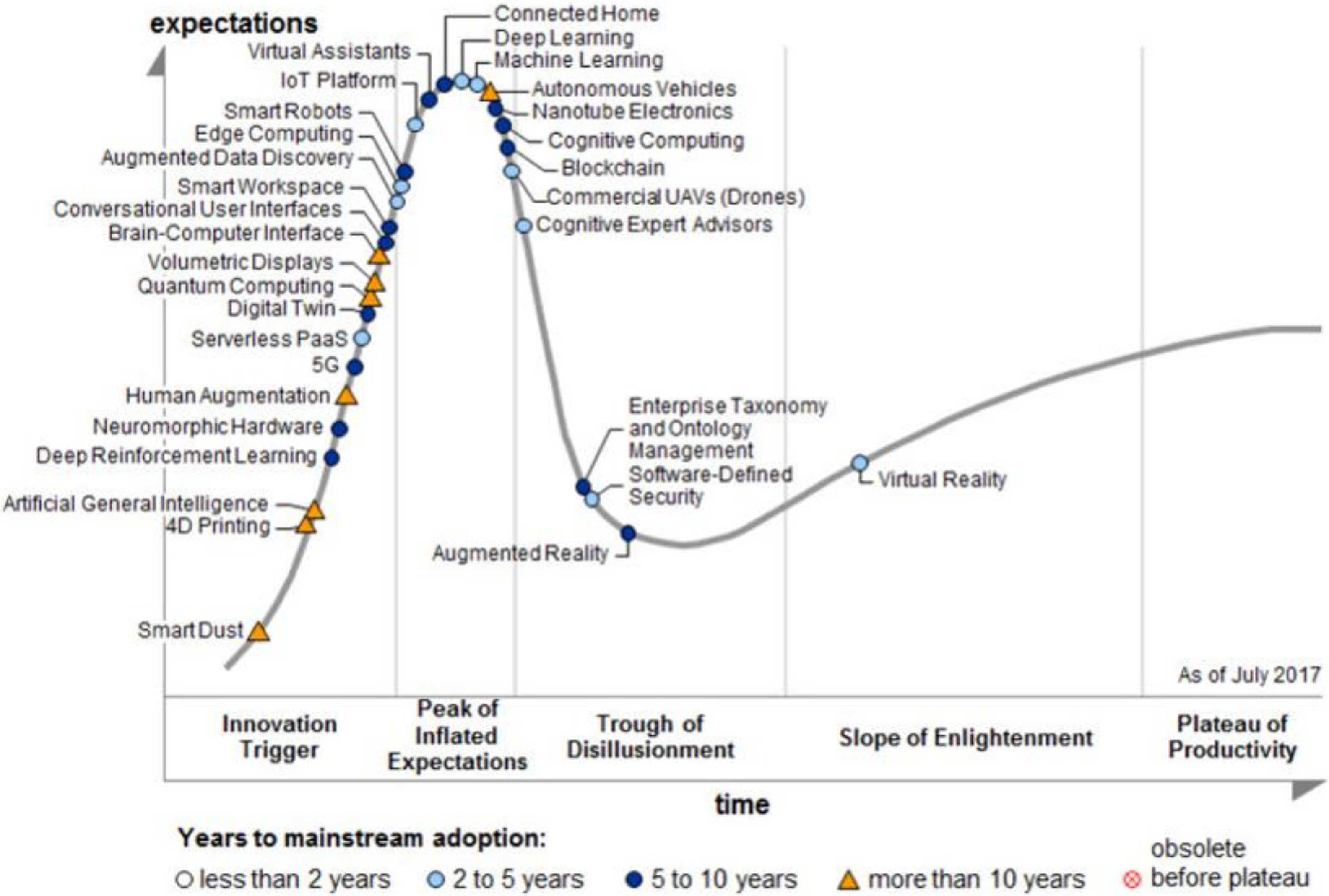
Highlights/Overview

- intro to AI/ML/DL
- linear regression
- activation/cost functions
- gradient descent
- back propagation
- hyper-parameters
- what are CNNs/RNNs?
- what is TensorFlow?
- C++ and TensorFlow

The Data/AI Landscape



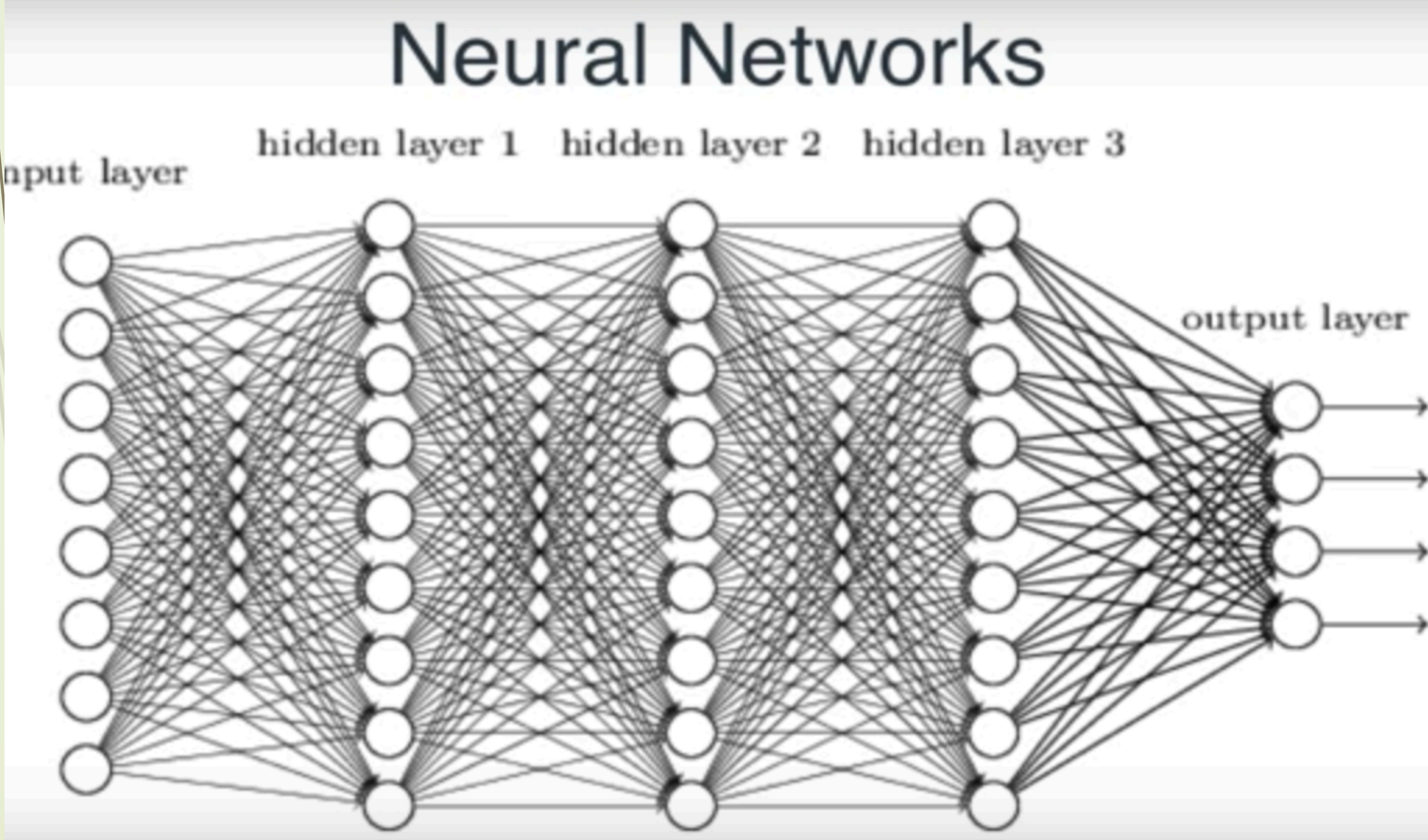
Gartner 2017: Deep Learning (YES!)



Note: PaaS = platform as a service; UAVs = unmanned aerial vehicles

The Official Start of AI (1956)

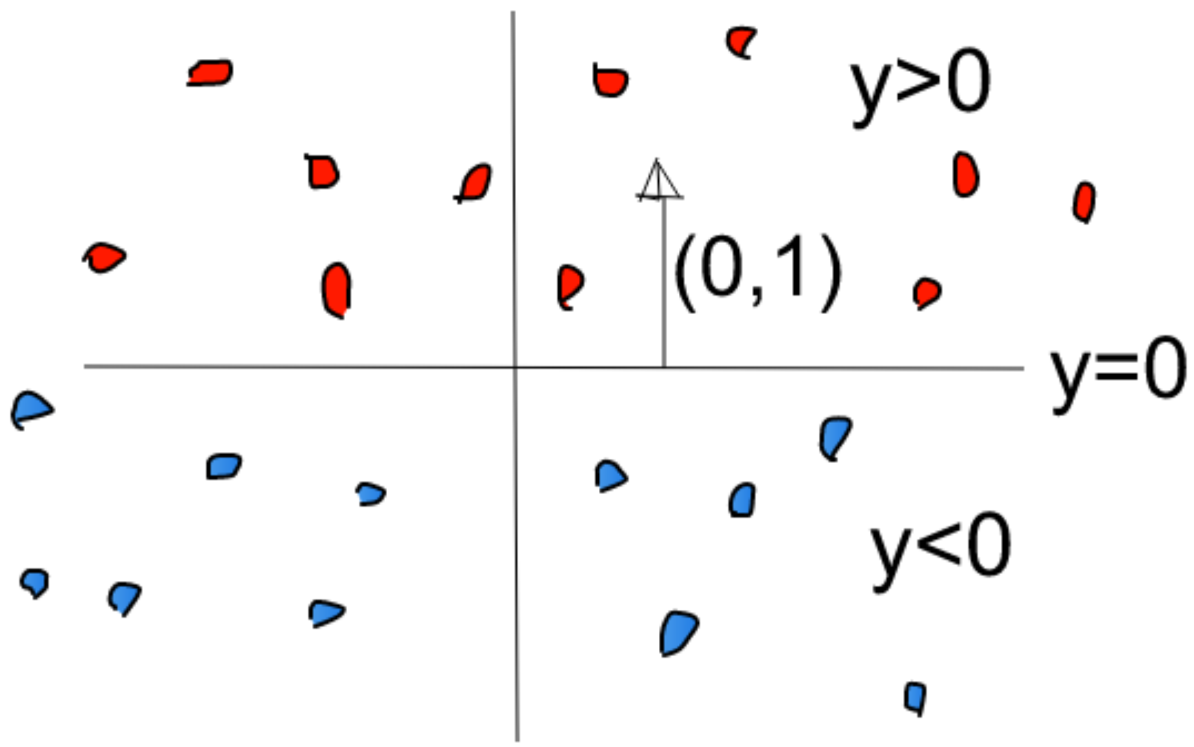
Neural Network with 3 Hidden Layers



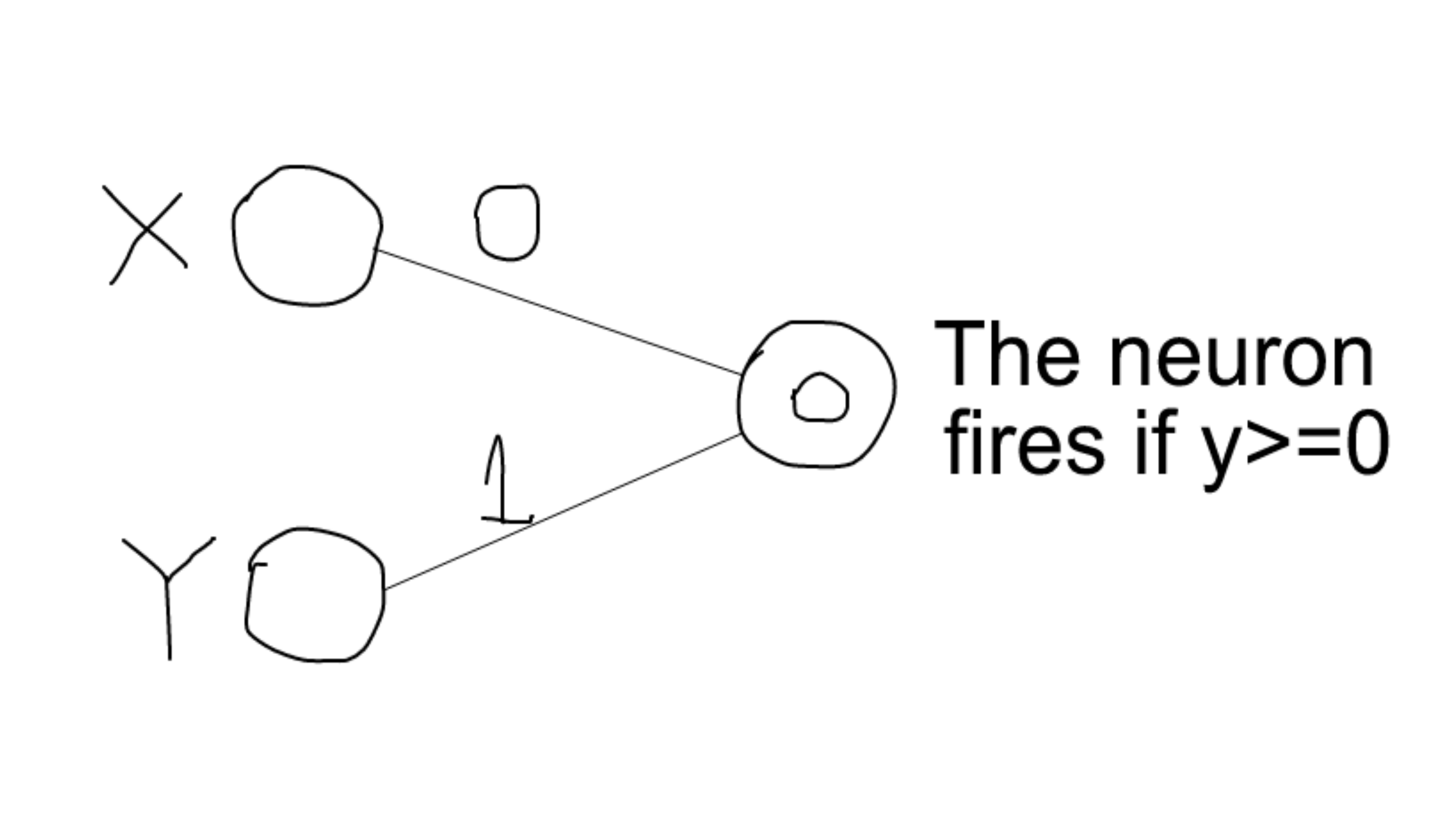
Clustering Example #1

- Given some red dots and blue dots
- Red dots are in the upper half plane
- Blue dots in the lower half plane
- How to detect if a point is red or blue?

Clustering Example #1



Clustering Example #1



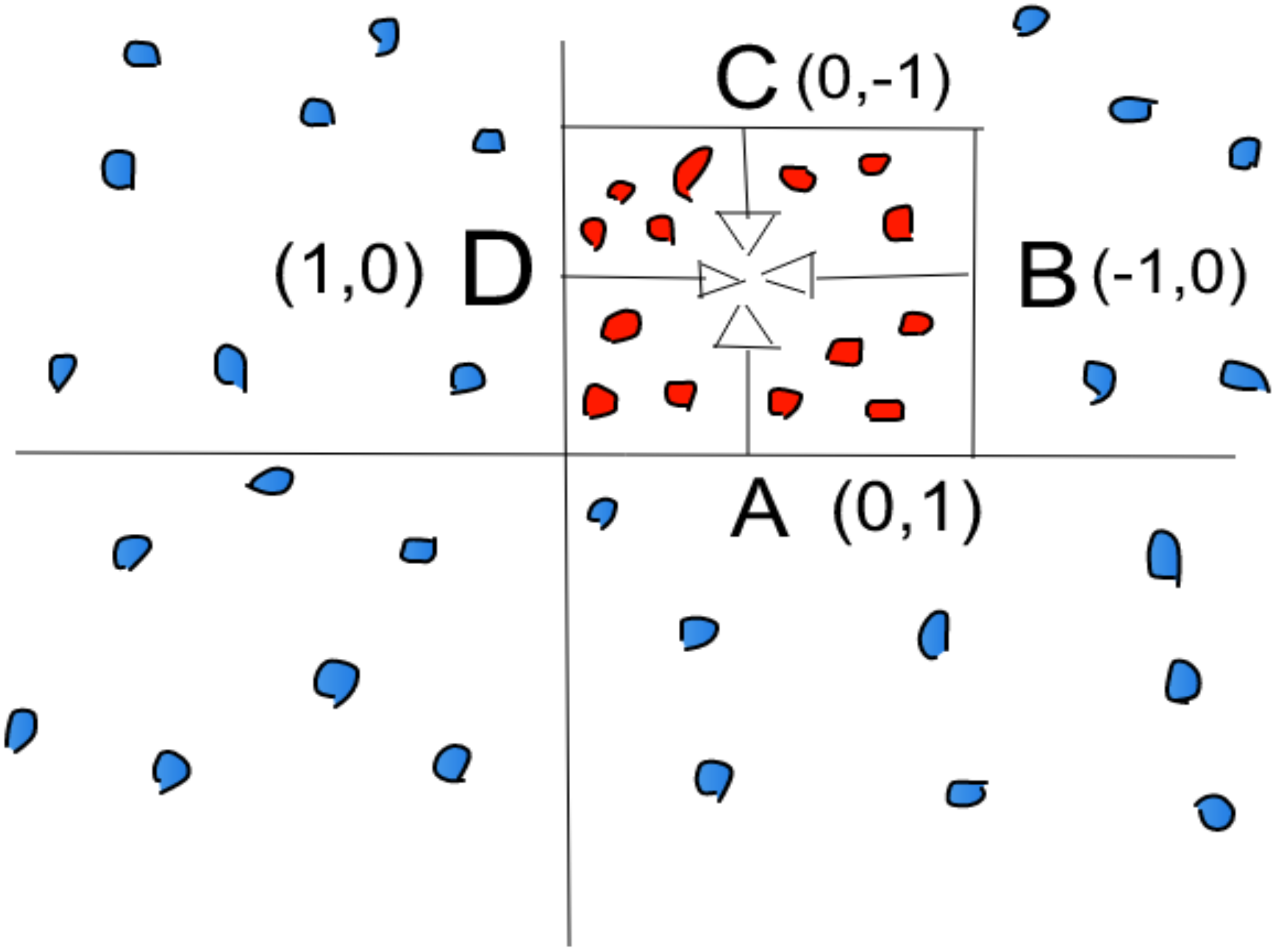
Clustering Example #2

- Given some red dots and blue dots
- Red dots are inside a unit square
- Blue dots are outside the unit square
- How to detect if a point is red or blue?

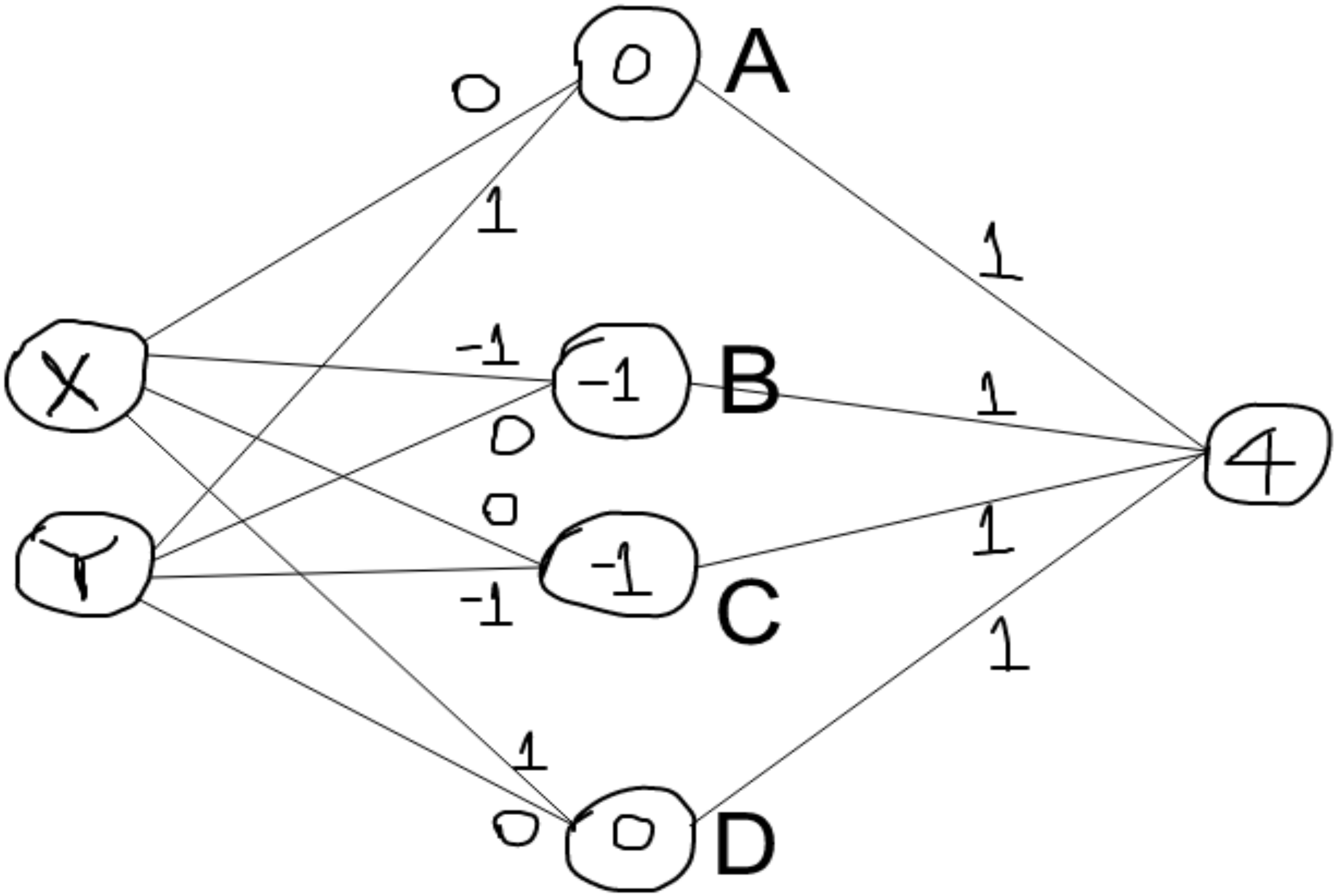
Clustering Example #2

- Two input nodes X and Y
- One hidden layer with 4 nodes (one per line)
- X & Y weights are the (x,y) values of the inward pointing perpendicular vector of each side
- The threshold values are the negative of the y-intercept (or the x-intercept)
- The outbound weights are all equal to 1
- The threshold for the output node node is 4

Clustering Example #2



Clustering Example #2



Clustering Exercises #1

- Describe an NN for a triangle
- Describe an NN for a pentagon
- Describe an NN for an n-gon (convex)
- Describe an NN for an n-gon (non-convex)

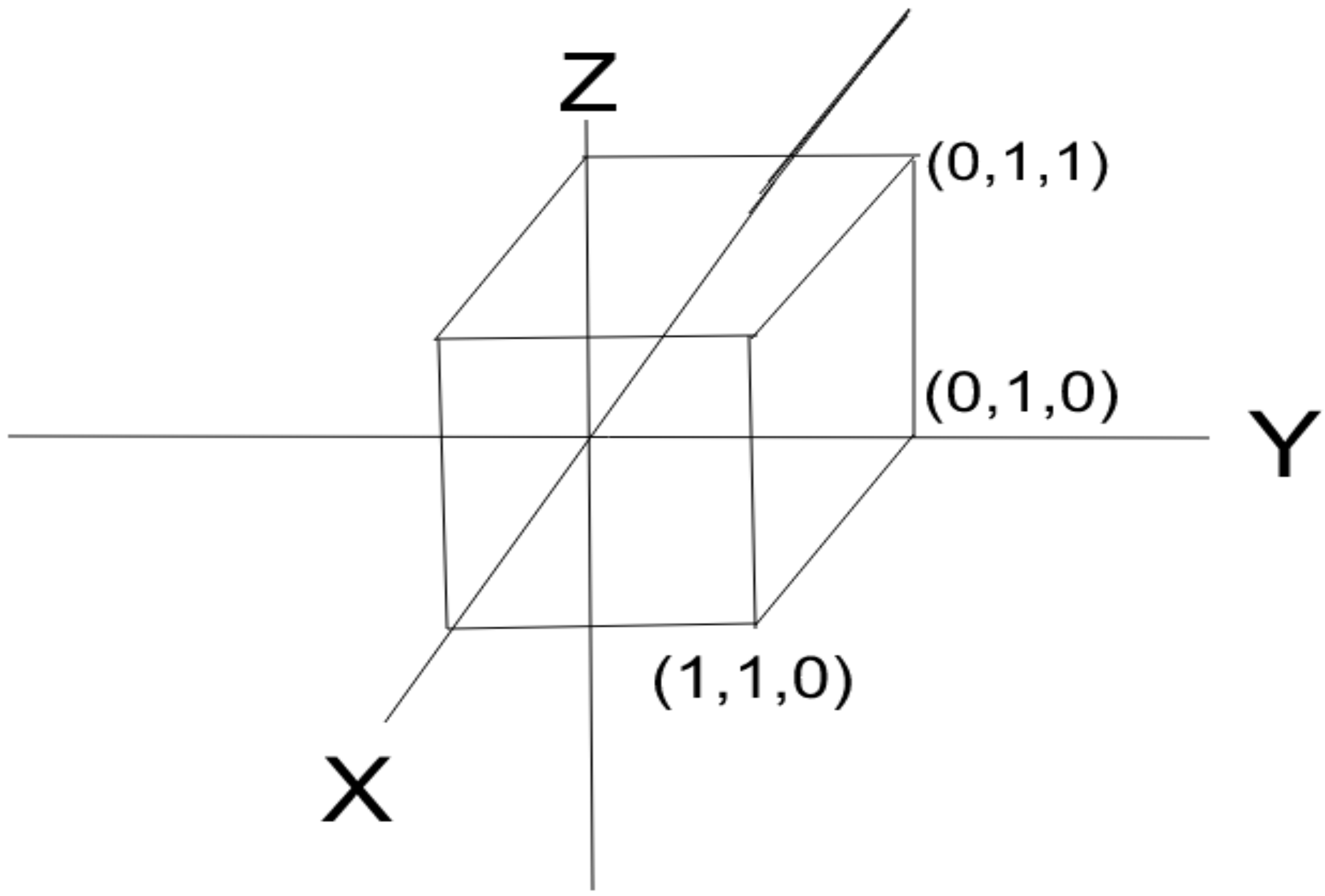
Clustering Exercises #2

- Create an NN for an OR gate
- Create an NN for a NOR gate
- Create an NN for an AND gate
- Create an NN for a NAND gate
- Create an NN for an XOR gate

=> requires TWO hidden layers

Clustering Exercises #3

- Convert example #2 to a 3D cube



Clustering Example #2

- A few points to keep in mind:
- A “step” activation function (0 or 1)
- No back propagation
- No cost function
- => no learning involved

A Basic Model in Machine Learning

- Let's perform the following steps:
- 1) Start with a simple model (2 variables)
- 2) Generalize that model (n variables)
- 3) See how it might apply to a NN

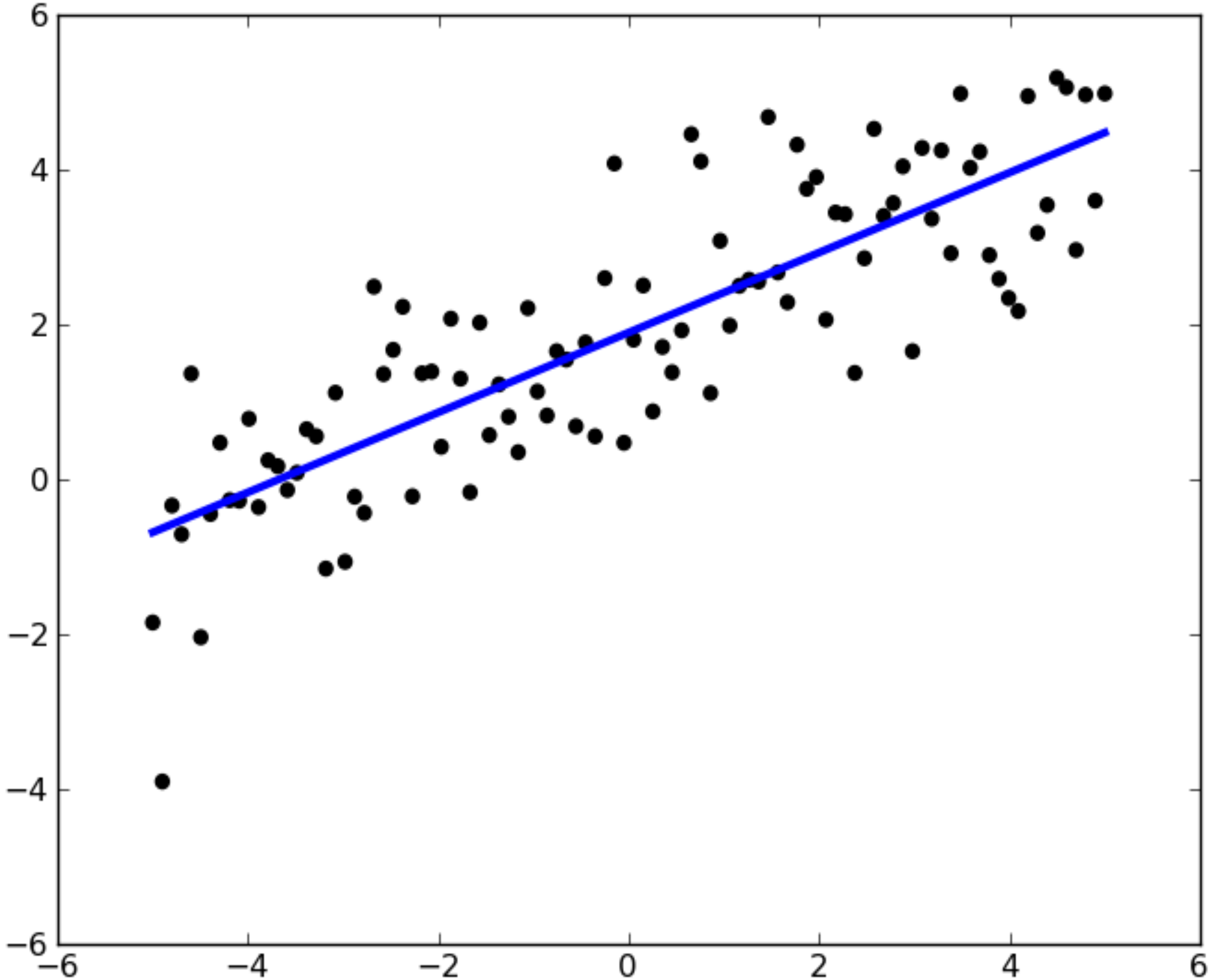
Linear Regression

- One of the simplest models in ML
- Fits a line ($y = m*x + b$) to data in 2D
- Finds best line by minimizing MSE:

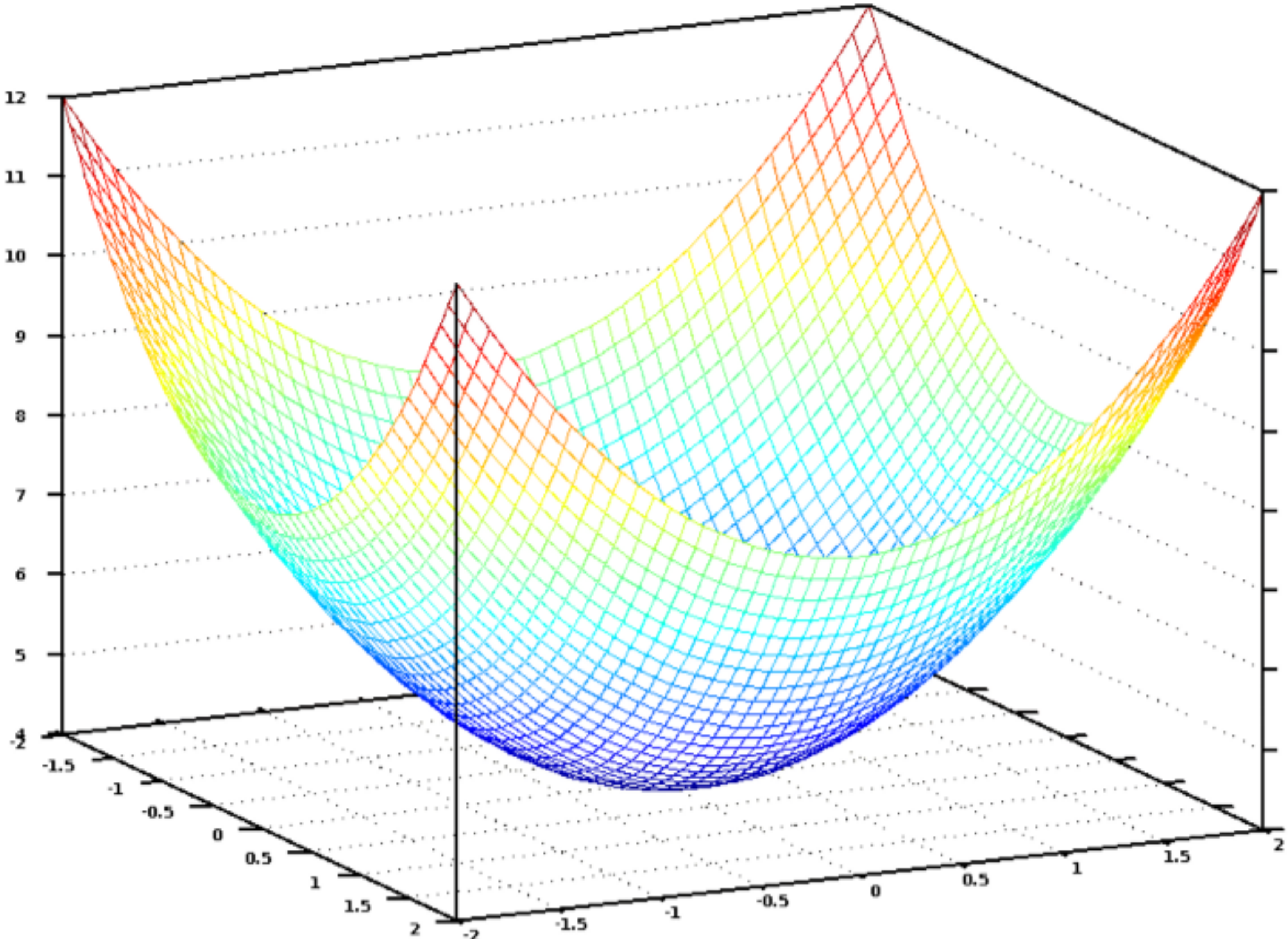
m = average of x values ("mean")

b also has a closed form solution

Linear Regression in 2D: example



Sample Cost Function #1 (MSE)



Linear Regression: example #1

- One feature (independent variable):

X = number of square feet

- Predicted value (dependent variable):

Y = cost of a house

- A very "coarse grained" model
- We can devise a much better model

Linear Regression: example #2

- Multiple features:
- X1 = # of square feet
- X2 = # of bedrooms
- X3 = # of bathrooms (dependency?)
- X4 = age of house
- X5 = cost of nearby houses
- X6 = corner lot (or not): Boolean

- a much better model (6 features)

Linear Multivariate Analysis

- General form of multivariate equation:

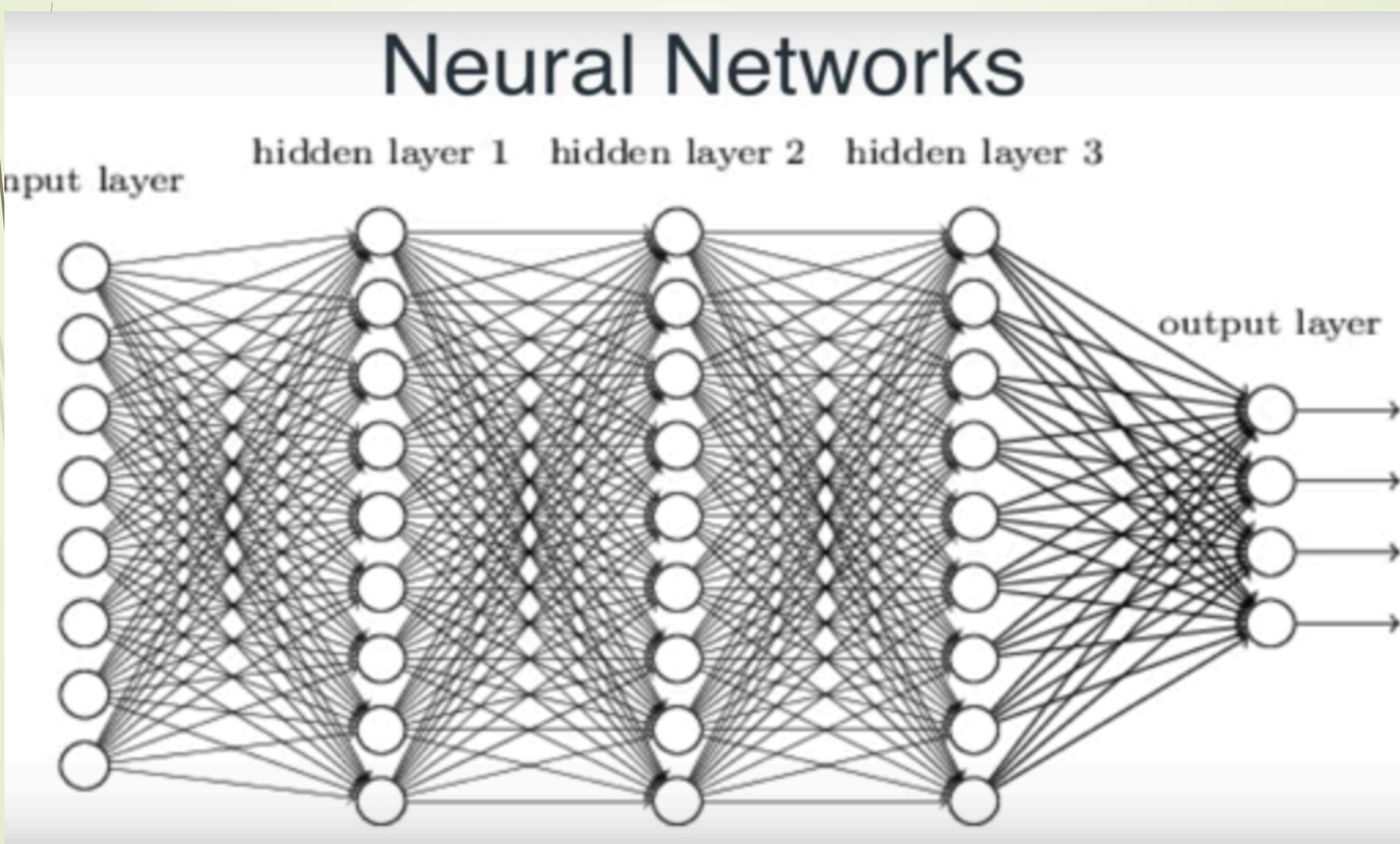
- Y = w1*x1 + w2*x2 + . . . + wn*xn + b
- w1, w2, . . . , wn are numeric values
- x1, x2, . . . , xn are variables (features)

- Properties of variables:

Can be independent (Naïve Bayes)

weak/strong dependencies can exist

Neural Network with 3 Hidden Layers



Neural Networks: equations

- Node "values" in first hidden layer:
- N1 = w11*x1+w21*x2+…+wn1*xn
- N2 = w12*x1+w22*x2+…+wn2*xn
- N3 = w13*x1+w23*x2+…+wn3*xn
- . . .
- Nn = w1n*x1+w2n*x2+…+wnn*xn
- Similar equations for other pairs of layers

Neural Networks: Matrices

- From inputs to first hidden layer:

Y1 = W1*X + B1 (X/Y1/B1: vectors; W1: matrix)

From first to second hidden layers:

Y2 = W2*X + B2 (X/Y2/B2: vectors; W2: matrix)

From second to third hidden layers:

Y3 = W3*X + B3 (X/Y3/B3: vectors; W3: matrix)

➔ Apply an “activation function” to y values

Neural Networks (general)

- Multiple hidden layers:

Layer composition is your decision

- Activation functions: sigmoid, tanh, RELU

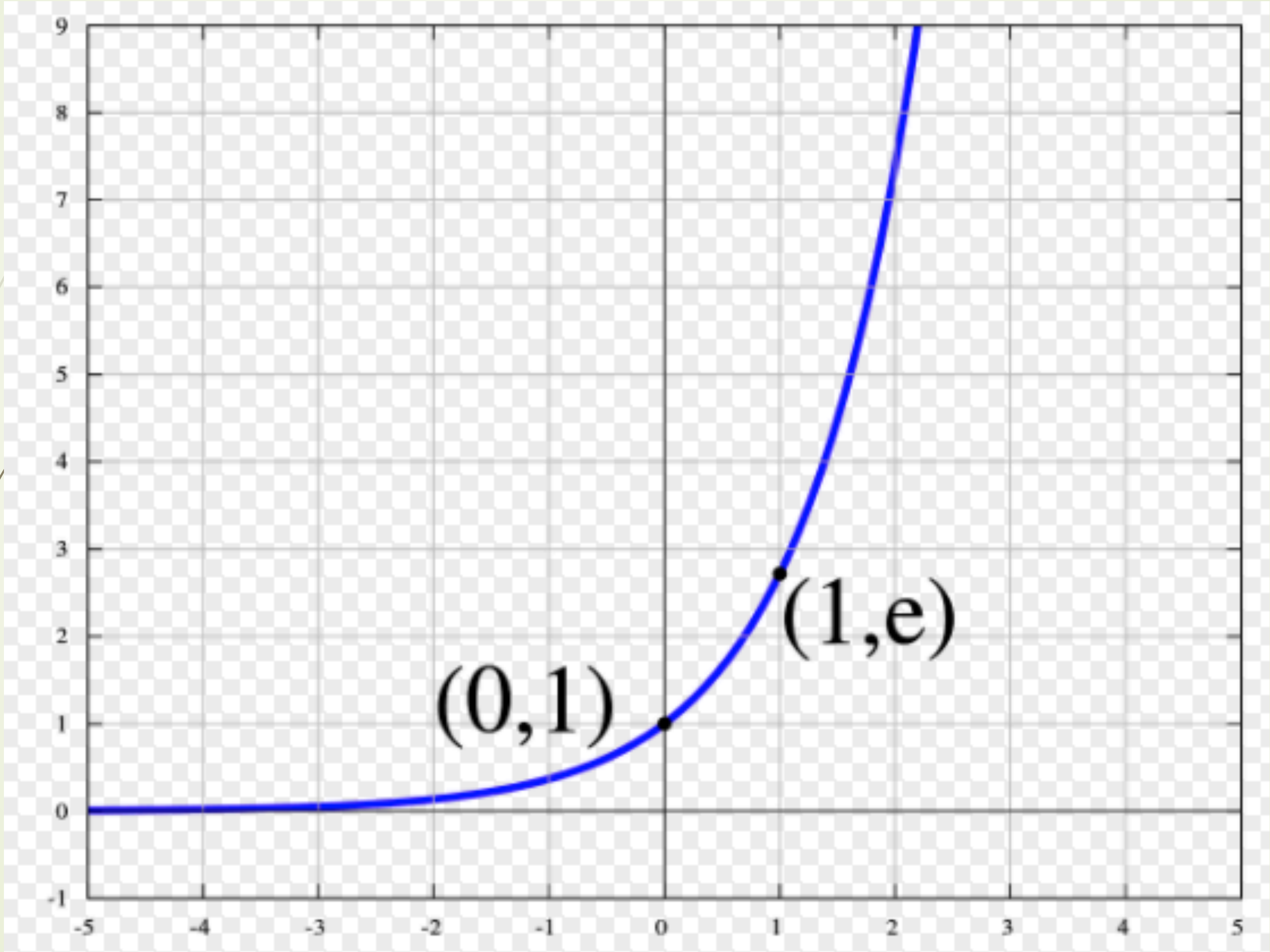
https://en.wikipedia.org/wiki/Activation_function

- Back propagation (1980s)

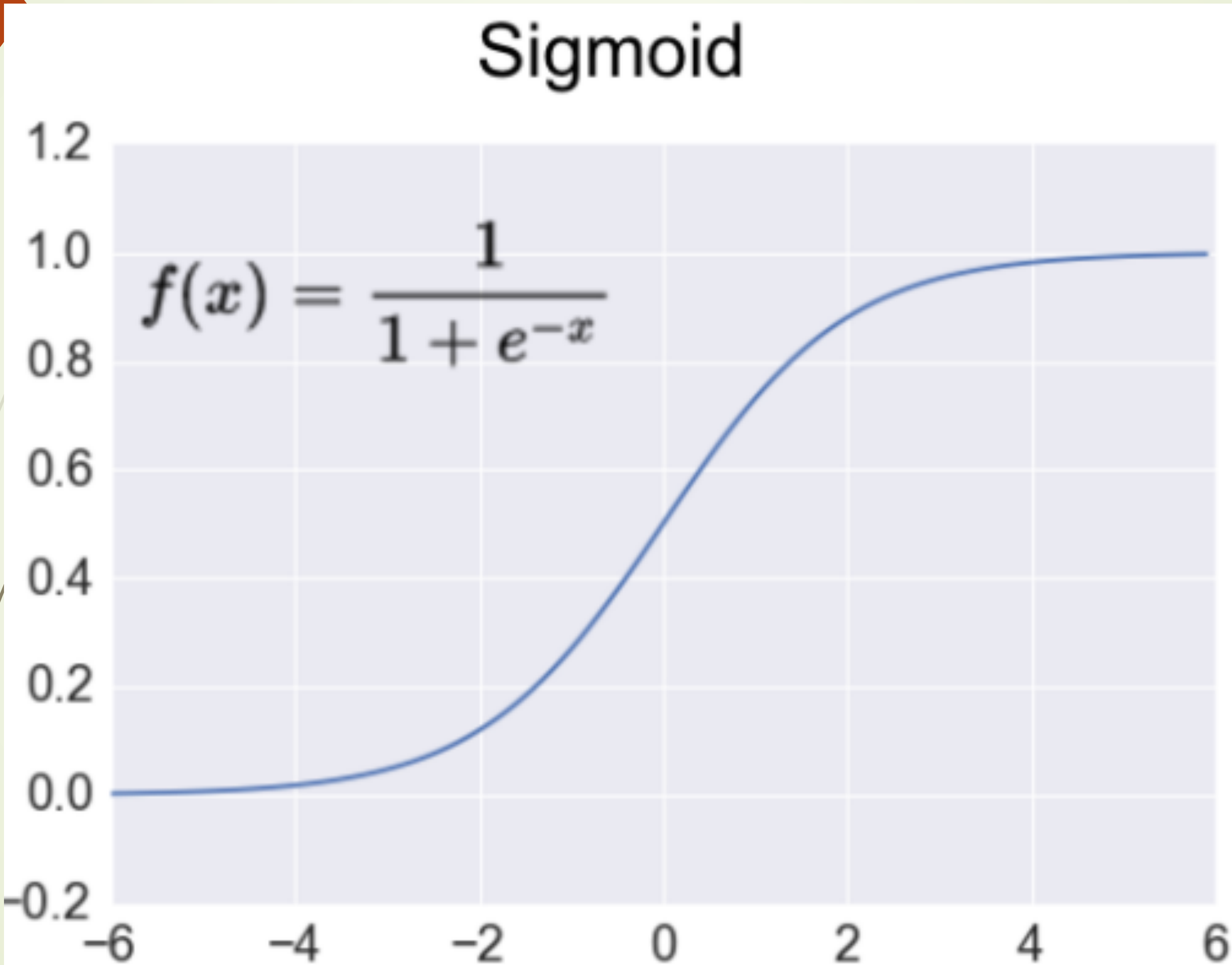
https://en.wikipedia.org/wiki/Backpropagation

- => Initial weights: small random numbers

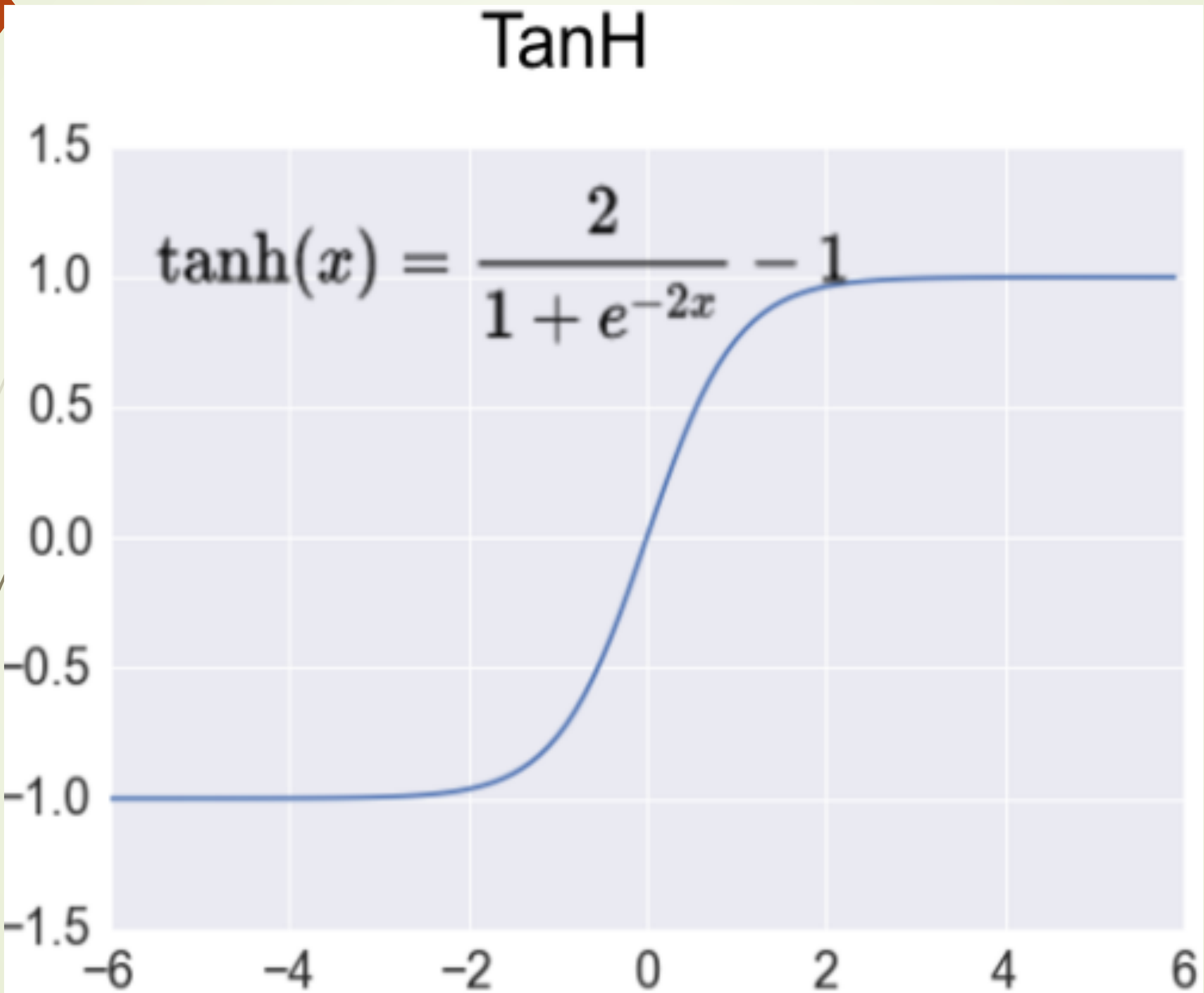
Euler's Function



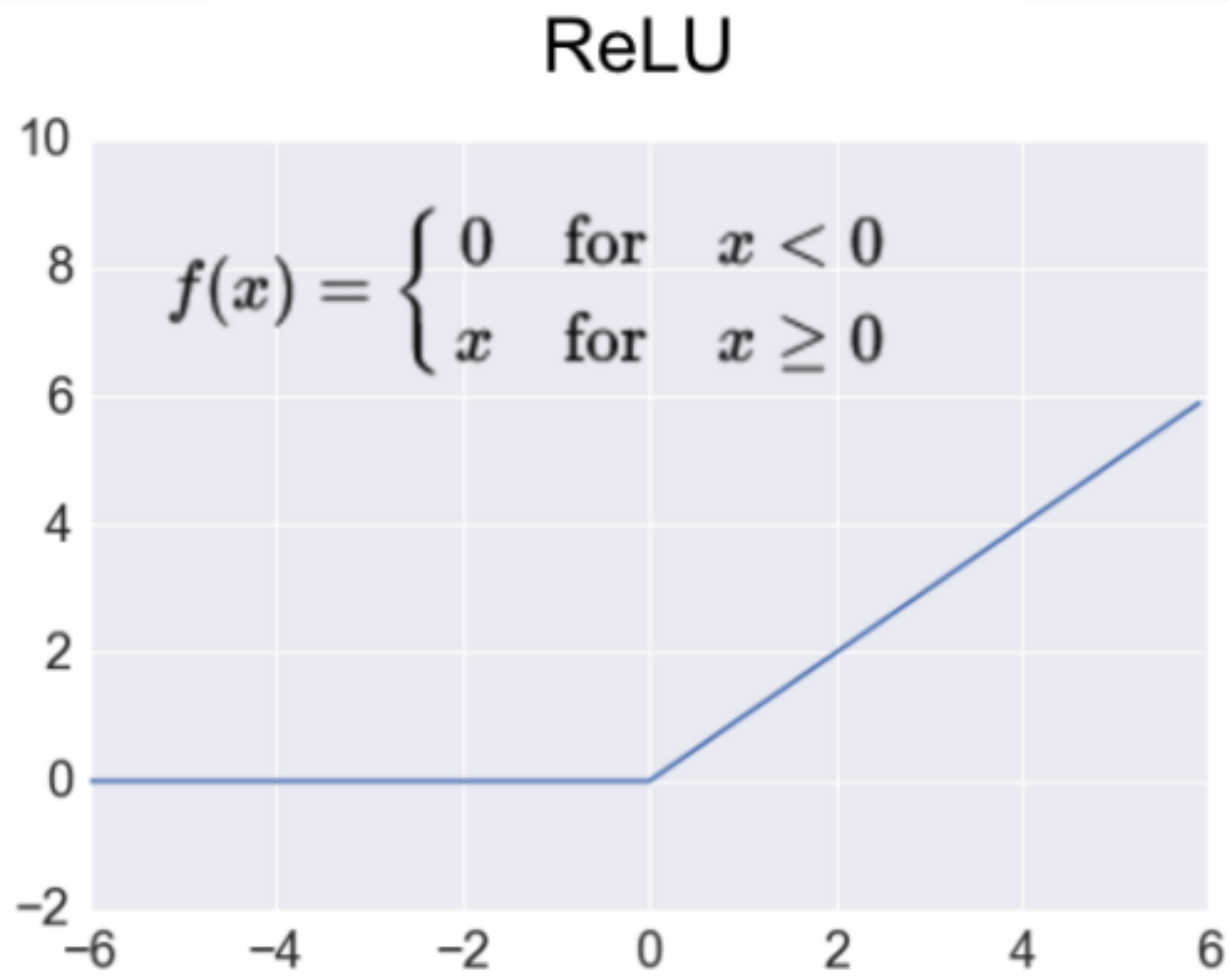
The sigmoid Activation Function



The tanh Activation Function



The ReLU Activation Function



The softmax Activation Function

$$o(x)_j = \frac{e^{x_i}}{\sum_{n=1}^{N} e^{x_n}} \quad for\, j = 1 \ldots .N$$

Activation Functions in Python

- import numpy as np
- ...
- # Python sigmoid example:
- z = 1 / (1 + np.exp(-np.dot(W, x)))
- . . .
- # Python tanh example:
- z = np.tanh(np.dot(W,x));

- # Python ReLU example:
- z = np.maximum(0, np.dot(W, x))

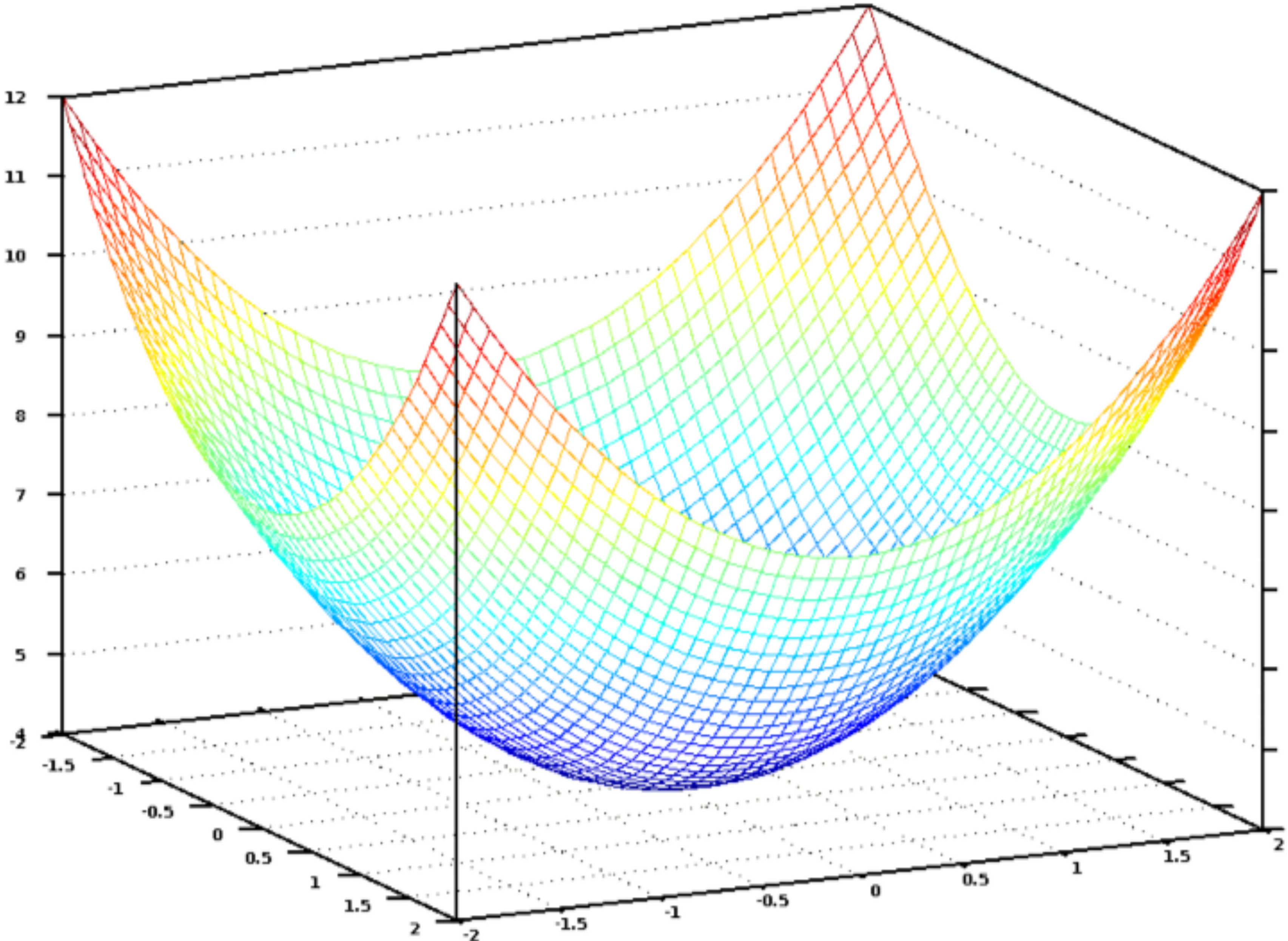
What's the "Best" Activation Function?

- Initially: sigmoid was popular
- Then: tanh became popular
- Now: RELU is preferred (better results)
- Softmax: for FC (fully connected) layers
- NB: sigmoid and tanh are used in LSTMs

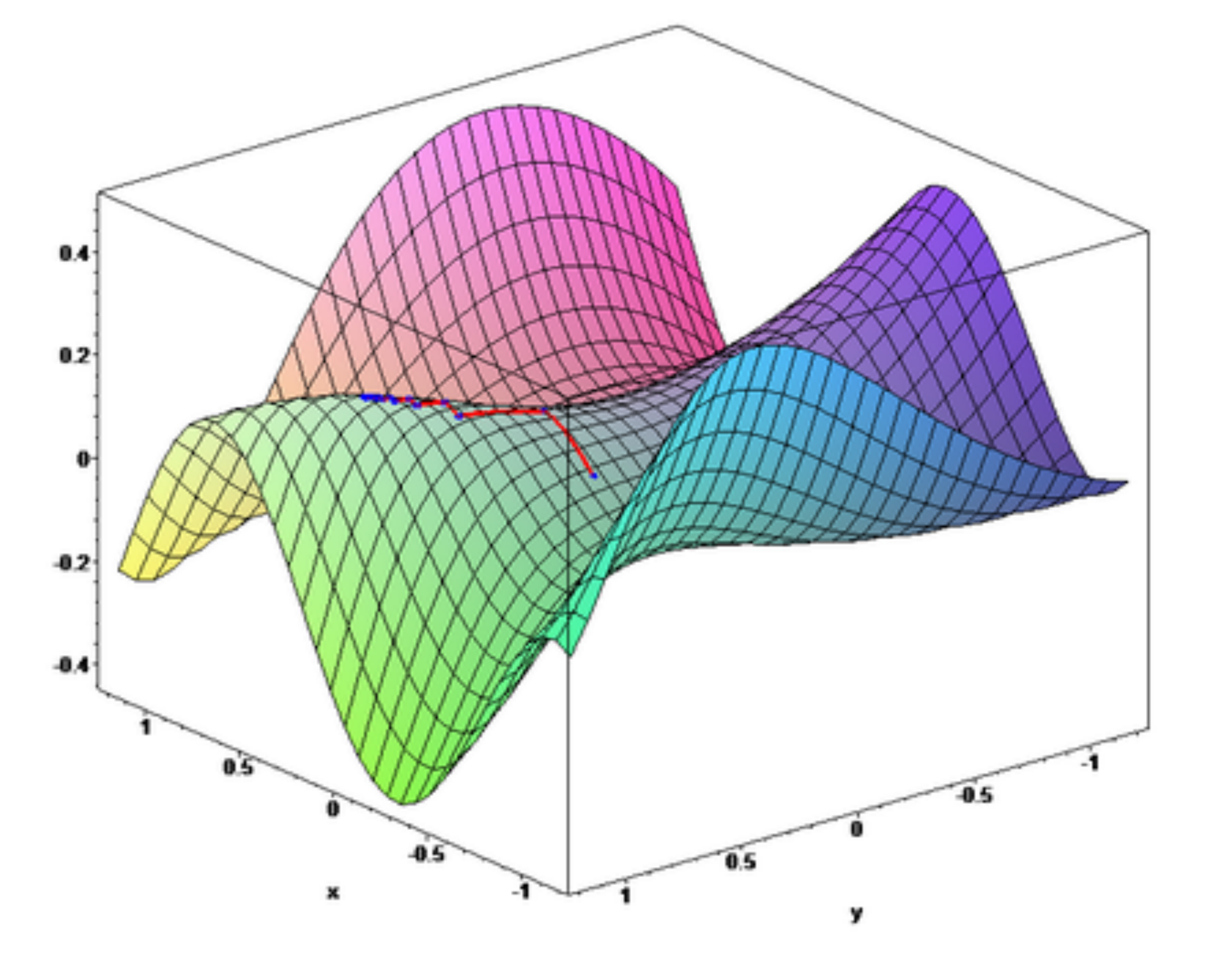
Even More Activation Functions!

- https://stats.stackexchange.com/questions/115258/comprehensive-list-of-activation-functions-in-neural-networks-with-pros-cons
- https://medium.com/towards-data-science/activation-functions-and-its-types-which-is-better-a9a5310cc8f
- https://medium.com/towards-data-science/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

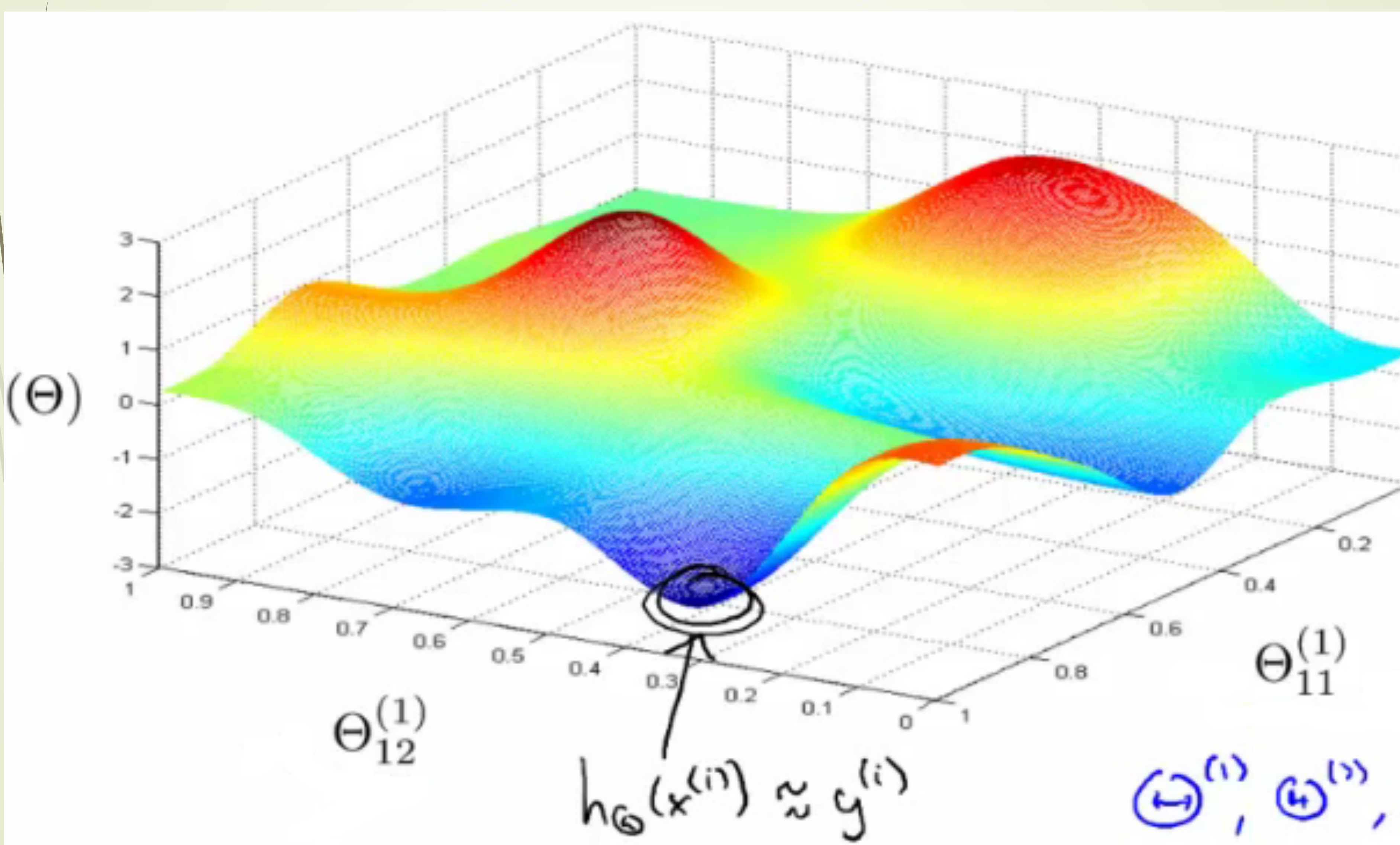
Sample Cost Function #1 (MSE)



Sample Cost Function #2



Sample Cost Function #3



How to Select a Cost Function

- mean-squared error:

for a regression problem

- binary cross-entropy (or mse):

for a two-class classification problem

- categorical cross-entropy:

for a many-class classification problem

GD versus SGD

- SGD (Stochastic Gradient Descent):

+ involves a SUBSET of the dataset

+ aka Minibatch Stochastic Gradient Descent

- GD (Gradient Descent):

+ involves the ENTIRE dataset

- More details:

http://cs229.stanford.edu/notes/cs229-notes1.pdf

Setting up Data & the Model

- Normalize the data:

Subtract the 'mean' and divide by stddev

[Central Limit Theorem]

- Initial weight values for NNs:

Random numbers between 0 and 1 (or N(0,1))

- More details:

http://cs231n.github.io/neural-networks-2/#losses

Hyper Parameters (examples)

- # of hidden layers in a neural network
- the learning rate (in many models)
- the dropout rate
- # of leaves or depth of a tree
- # of latent factors in a matrix factorization
- # of clusters in a k-means clustering

Hyper Parameter: dropout rate

- "dropout" refers to dropping out units (both hidden and visible) in a neural network
- a regularization technique for reducing overfitting in neural networks
- prevents complex co-adaptations on training data
- a very efficient way of performing model averaging with neural networks

How Many Layers in a DNN?

- Algorithm #1 (from Geoffrey Hinton):

1) add layers until you start overfitting your training set
2) now add dropout or some another regularization method

- Algorithm #2 (Yoshua Bengio):

"Add layers until the test error does not improve anymore."

How Many Hidden Nodes in a DNN?

- Based on a relationship between:
- # of input and # of output nodes
- Amount of training data available
- Complexity of the cost function
- The training algorithm

- TF playground home page:

http://playground.tensorflow.org

CNNs versus RNNs

- CNNs (Convolutional NNs):

Good for image processing

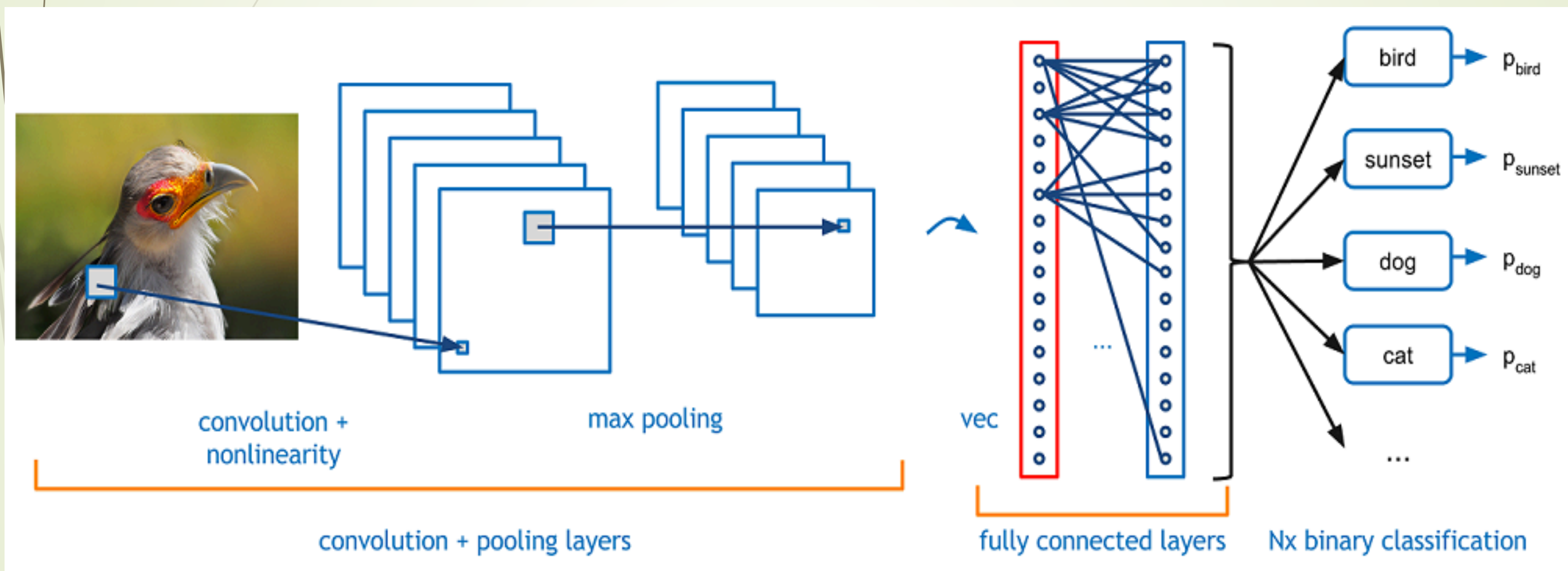
2000: CNNs processed 10-20% of all checks

=> Approximately 60% of all NNs

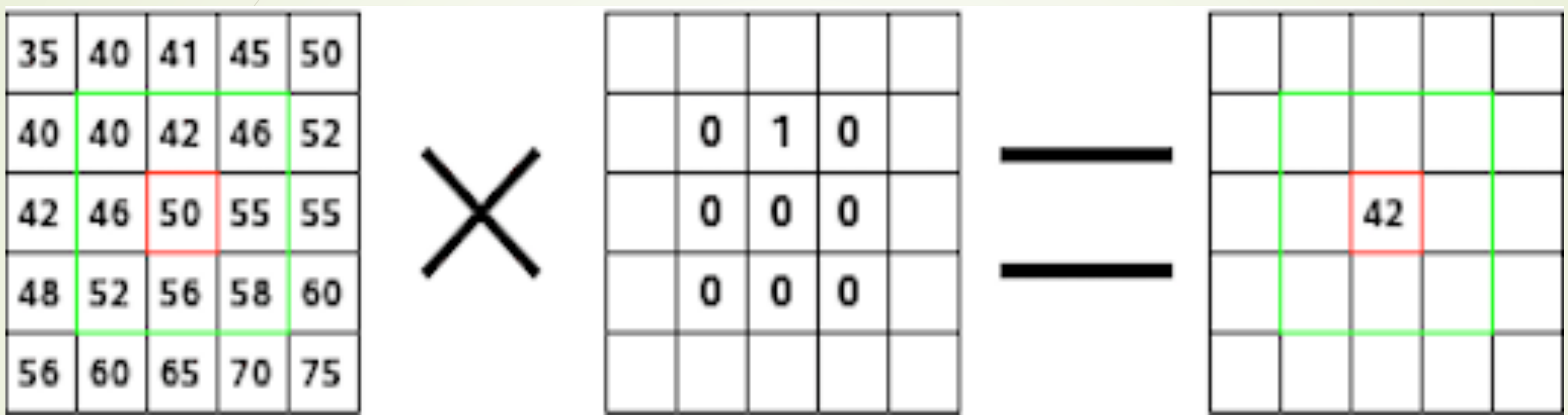
- RNNs (Recurrent NNs):

Good for NLP and audio

CNNs: convolution and pooling (2)



CNNs: Convolution Calculations



- https://docs.gimp.org/en/plug-in-convmatrix.html

CNNs: Convolution Matrices (examples)

- Sharpen:

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

- Blur:

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

CNNs: Convolution Matrices (examples)

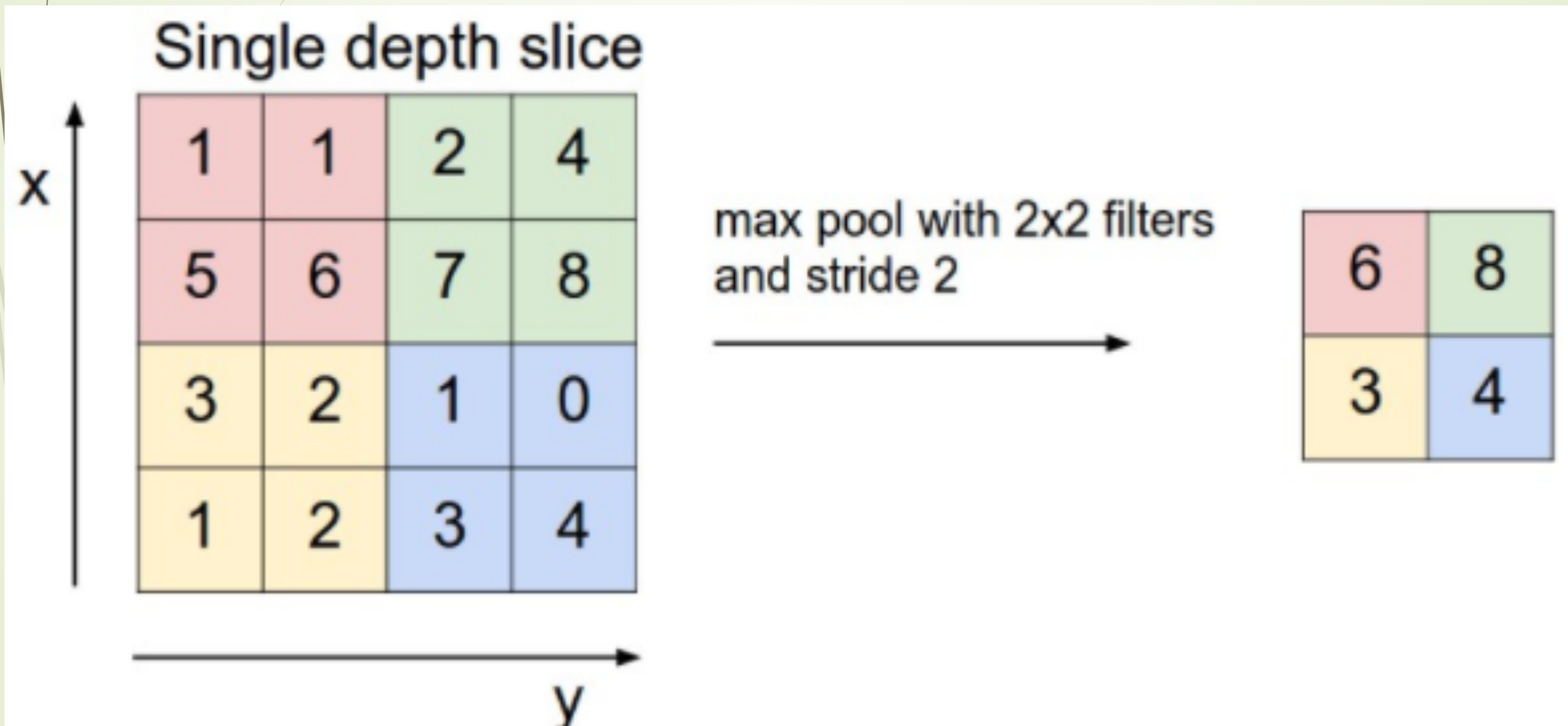
- Edge detect:

	0	1	0	
	1	-4	1	
	0	1	0	

- Emboss:

	-2	-1	0	
	-1	1	1	
	0	1	2	

CNNs: Max Pooling Example



CNN in Python/Keras (fragment)

```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Flatten, Activation
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.optimizers import Adadelta

input_shape = (3, 32, 32)
nb_classes = 10

model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape=input_shape))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

What is TensorFlow?

- An open source framework for ML and DL
- A "computation" graph
- Created by Google (released 11/2015)
- Evolved from Google Brain
- Linux and Mac OS X support (VM for Windows)

- TF home page: https://www.tensorflow.org/

What is TensorFlow?

- Support for Python, Java, C++
- Desktop, server, mobile device (TensorFlow Lite)
- CPU/GPU/TPU support

- Visualization via TensorBoard
- Can be embedded in Python scripts
- Installation: pip install tensorflow

TensorFlow cluster:

https://www.tensorflow.org/deploy/distributed

TensorFlow Use Cases (Generic)

- Image recognition
- Computer vision
- Voice/sound recognition
- Time series analysis
- Language detection
- Language translation
- Text-based processing
- Handwriting Recognition

What is TensorFlow?

- Graph: graph of operations (DAG)
- Sessions: contains Graph(s)
- lazy execution (default)
- operations in parallel (default)
- Nodes: operators/variables/constants
- Edges: tensors

- => graphs are split into subgraphs and executed in parallel (or multiple CPUs)

TensorFlow Graph Execution

- Execute statements in a tf.Session() object
- Invoke the “run” method of that object
- “eager” execution is now possible
- Not part of the mainline yet
- Installation: pip install tf-nightly

What is a Tensor?

- TF tensors are n-dimensional arrays
- TF tensors are very similar to numpy ndarrays

- scalar number: a zeroth-order tensor
- vector: a first-order tensor
- matrix: a second-order tensor
- 3-dimensional array: a 3rd order tensor

- https://dzone.com/articles/tensorflow-simplified-examples

TensorFlow "primitive types"

- tf.constant: initialized immediately

- tf.placeholder (a function):

+ initial value is not required

+ assigned value via feed_dict at run time

+ are not modified during training

- tf.Variable (a class):

+ initial value is required

+ updated during training

+ in-memory buffer (saved/restored from disk)

+ can be shared between works (distributed env)

TensorFlow: constants (immutable)

- import tensorflow as tf # tf-const.py

- aconst = tf.constant(3.0)
- print(aconst)

\# output: Tensor("Const:0", shape=(), dtype=float32)

- sess = tf.Session()
- print(sess.run(aconst))

\# output: 3.0

- sess.close()

- \# => there's a better way...

TensorFlow: constants

- import tensorflow as tf # tf-const2.py

- aconst = tf.constant(3.0)
- print(aconst)

- Automatically close “sess”
- with tf.Session() as sess:
-   print(sess.run(aconst))

TensorFlow Arithmetic

```
import tensorflow as tf # basic1.py

a = tf.add(4, 2)
b = tf.subtract(8, 6)
c = tf.multiply(a, 3)
d = tf.div(a, 6)

with tf.Session() as sess:
  print(sess.run(a)) # 6
  print(sess.run(b)) # 2
  print(sess.run(c)) # 18
  print(sess.run(d)) # 1
```

TensorFlow Arithmetic Methods

```
import tensorflow as tf #tf-math-ops.py

PI = 3.141592
sess = tf.Session()

print(sess.run(tf.div(12,8)))
print(sess.run(tf.floordiv(20.0,8.0)))
print(sess.run(tf.sin(PI)))
print(sess.run(tf.cos(PI)))
print(sess.run(tf.div(tf.sin(PI/4.), tf.cos(PI/4.))))
```

TensorFlow Arithmetic Methods

- Output from tf-math-ops.py:

- 1
- 2.0
- 6.27833e-07
- -1.0
- 1.0

TF placeholders and feed_dict

```
import tensorflow as tf # tf-var-multiply.py

a = tf.placeholder("float")
b = tf.placeholder("float")
c = tf.multiply(a,b)

# initialize a and b:
feed_dict = {a:2, b:3}

# multiply a and b:
with tf.Session() as sess:
  print(sess.run(c, feed_dict))
```

TensorFlow and Linear Regression

- import tensorflow as tf

- # W and x are 1d arrays
- W = tf.constant([10,20], name='W')
- x = tf.placeholder(tf.int32, name='x')
- b = tf.placeholder(tf.int32, name='b')

- Wx = tf.multiply(W, x, name='Wx')
- y = tf.add(Wx, b, name='y')

TensorFlow fetch/feed_dict

```
with tf.Session() as sess:
 print("Result 1: Wx = ",
             sess.run(Wx, feed_dict={x: [5,10]}))
 print("Result 2: y = ",
             sess.run(y, feed_dict={x: [5,10], b: [15,25]}))
```

- Result 1: Wx = [50 200]
- Result 2: y = [65 225]

Saving Graphs for TensorBoard

```
import tensorflow as tf # tf-save-data.py
x = tf.constant(5,name="x")
y = tf.constant(8,name="y")
z = tf.Variable(2*x+3*y, name="z”)
model = tf.global_variables_initializer()

with tf.Session() as session:
  writer = tf.summary.FileWriter(”./tf_logs",session.graph)
  session.run(model)
  print 'z = ',session.run(z) # =>  z = 34

# launch tensorboard: tensorboard –logdir=./tf_logs
```

TensorFlow Eager Execution

- An imperative interface to TF (experimental)
- Fast debugging & immediate run-time errors

- Eager execution is not included in v1.4 of TF
- build TF from source or install the nightly build

- pip install tf-nightly # CPU
- pip install tf-nightly-gpu #GPU

- => requires Python 3.x (not Python 2.x)

TensorFlow Eager Execution

- integration with Python tools
- Supports dynamic models + Python control flow
- support for custom and higher-order gradients
- Supports most TensorFlow operations

- https://research.googleblog.com/2017/10/eager-execution-imperative-define-by.html

TensorFlow Eager Execution

- import tensorflow as tf # tf-eager1.py
- import tensorflow.contrib.eager as tfe

- tfe.enable_eager_execution()

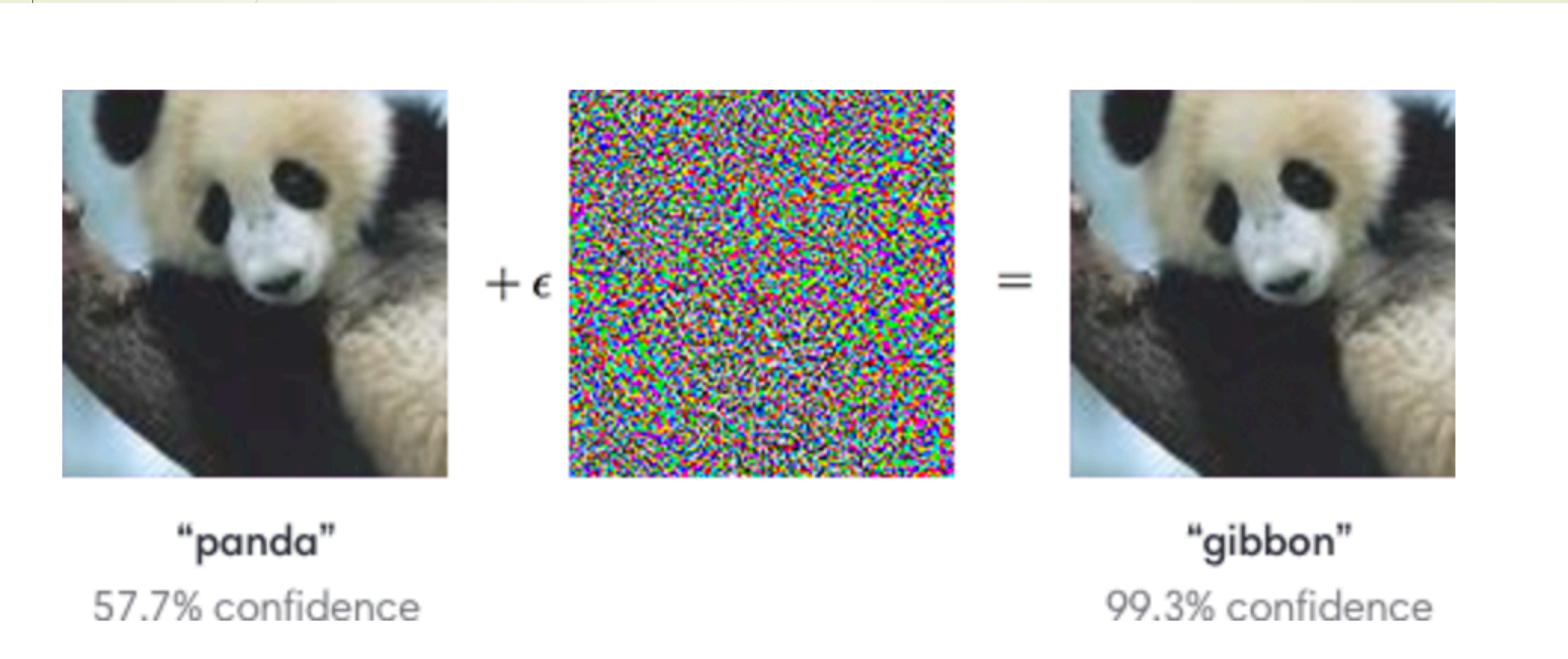
- x = [[2.]]
- m = tf.matmul(x, x)
- print(m)

tf.Tensor([[4.]], shape=(1, 1), dtype=float32)

C++ and TensorFlow

- https://www.tensorflow.org/api_guides/cc/guide
- https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/label_image
- With bazel:
- https://jacobgil.github.io/deeplearning/tensorflow-cpp
- Without bazel:
- https://github.com/tensorflow/tensorflow/issues/2412

GANs: Generative Adversarial Networks



GANs: Generative Adversarial Networks

- Make imperceptible changes to images
- Can consistently defeat all NNs
- Can have extremely high error rate
- Some images create optical illusions
- https://www.quora.com/What-are-the-pros-and-cons-of-using-generative-adversarial-networks-a-type-of-neural-network

GANs: Generative Adversarial Networks

- Create your own GANs:

https://www.oreilly.com/learning/generative-adversarial-networks-for-beginners

https://github.com/jonbruner/generative-adversarial-networks

- GANs from MNIST:

http://edwardlib.org/tutorials/gan

GANs: Generative Adversarial Networks

- GANs, Graffiti, and Art:

  https://thenewstack.io/camouflaged-graffiti-road-signs-can-fool-machine-learning-models/

  GANs and audio:

  https://www.technologyreview.com/s/608381/ai-shouldnt-believe-everything-it-hears

  Image recognition (single pixel change):

  http://www.bbc.com/news/technology-41845878

  Houdini algorithm: https://arxiv.org/abs/1707.05373

Deep Learning and Art

- “Convolutional Blending” (19-layer CNN):

www.deepart.io

- Bots created their own language:

https://www.recode.net/2017/3/23/14962182/ai-learning-language-open-ai-research

- https://www.fastcodesign.com/90124942/this-google-engineer-taught-an-algorithm-to-make-train-footage-and-its-hypnotic

What Do I Learn Next?

- PGMs (Probabilistic Graphical Models)
- MC (Markov Chains)
- MCMC (Markov Chains Monte Carlo)
- HMMs (Hidden Markov Models)
- RL (Reinforcement Learning)
- Hopfield Nets
- Neural Turing Machines
- Autoencoders
- Hypernetworks
- Pixel Recurrent Neural Networks
- Bayesian Neural Networks
- SVMs

About Me: Recent Books

1) HTML5 Canvas and CSS3 Graphics (2013)
2) jQuery, CSS3, and HTML5 for Mobile (2013)
3) HTML5 Pocket Primer (2013)
4) jQuery Pocket Primer (2013)
5) HTML5 Mobile Pocket Primer (2014)
6) D3 Pocket Primer (2015)
7) Python Pocket Primer (2015)
8) SVG Pocket Primer (2016)
9) CSS3 Pocket Primer (2016)
10) Android Pocket Primer (2017)
11) Angular Pocket Primer (2017)
12) Data Cleaning Pocket Primer (2018)
13) RegEx Pocket Primer (2018)

About Me: Training

=> Deep Learning. Keras, and TensorFlow:

http://codeavision.io/training/deep-learning-workshop

=> Mobile and TensorFlow Lite

=> R and Deep Learning (WIP)

=> Android for Beginners