CS145 Discussion
Week 4

Junheng, Shengming, Yunsheng

10/26/2018

Roadmap

- Announcement
- Homework 1
  - Note on Logistic Regression → Exercise 1
- Neural Networks
  - Overview
  - Forward propagation → Exercise 2, 3
  - Backpropagation → Exercise 4, 5
  - Pros and Cons
- Homework 2
  - Note on numerical computation
  - Note on python plotting
  - Notes on Numpy vs Panda

Announcement

- Homework 2 due next Tuesday (10/30/2018 11:59 pm)
  - Please double check your submission
    - Make sure you can unzip
  - Report is important
  - Unwise to leave any problem blank
  - PDF file is much better than TXT files
  - ZIP file makes life much easier (no 7z file or rar file please)
  - Report all necessary values on your report rather than a pointer to your code
- Project midterm report due 11/12
  - Your team is required to submit to Kaggle at least once
  - Detailed requirement will be posted soon

Homework 1: Note

HW 1, Logistic Regression: Why the difference?

python logisticRegression.py 0 0
Learning Algorithm Type: 0
Is normalization used: 0
Beta Starts: [0.21786753 0.55430027 0.93930025 0.40048303 0.56706405]
average logL for iteration 0: 3.0665982168306196
average logL for iteration 1000: 0.1346015208194089

average logL for iteration 2000: 0.09611998275763771

average logL for iteration 3000: 0.08019343404773255

...
average logL for iteration 23000: 0.034675192368321416

average logL for iteration 24000: 0.034196784110673

Beta: [2.38942064 -2.26606374 -1.32837263 -1.55395439 -0.16195076]
Training avgLogL: 0.033751622818600426
Test accuracy: 0.9890510948905109

python logisticRegression.py 1 0
Learning Algorithm Type: 1
Is normalization used: 0
Beta Starts: [0. 0. 0. 0. 0.]
average logL for iteration 0: 0.1950693606893002
average logL for iteration 1000: 0.018429477101347867

average logL for iteration 2000: 0.018429477101347843

average logL for iteration 3000: 0.018429477101347843

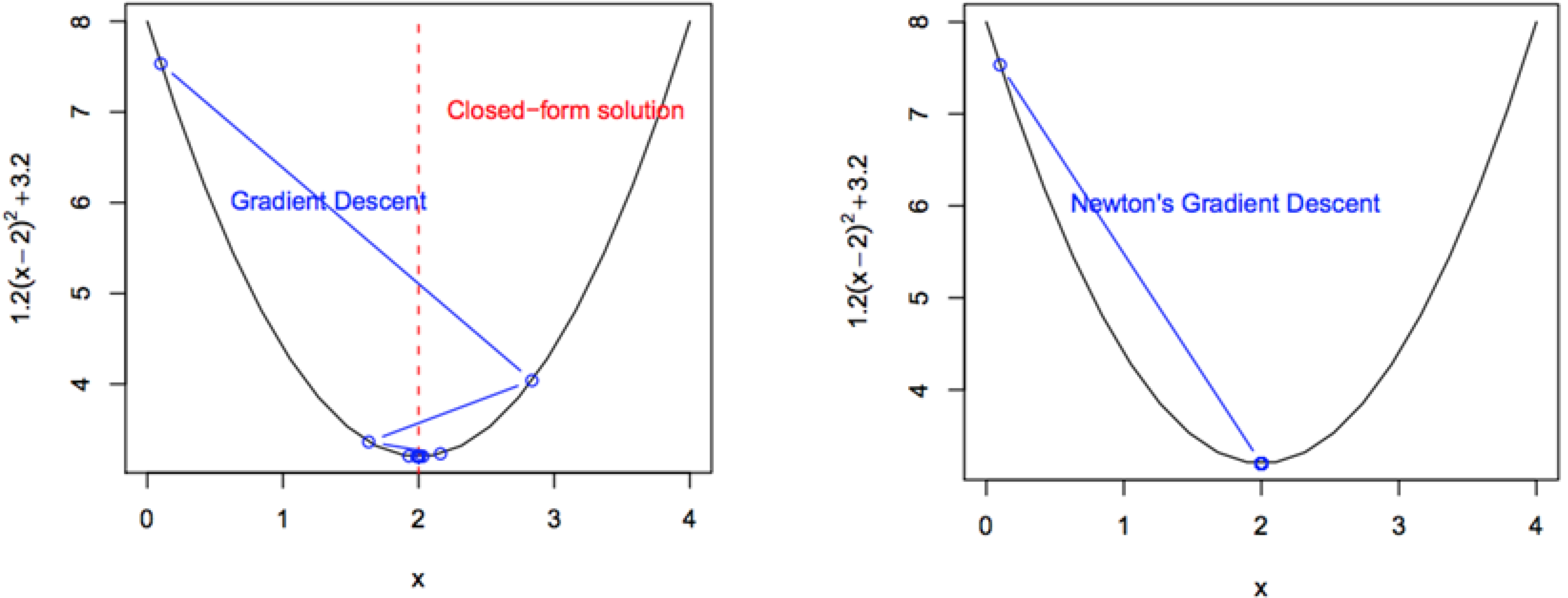
...
average logL for iteration 23000: 0.018429477101347867

average logL for iteration 24000: 0.018429477101347843

Beta: [7.31317701 -7.70705368 -4.15787617 -5.21346398 -0.58583733]
Training avgLogL: 0.018429477101347843
Test accuracy: 0.9890510948905109

HW 1, Logistic Regression: Why the difference?

- Fact: The loss function of logistic regression is ***convex***.
- (Check http://mathgotchas.blogspot.com/2011/10/why-is-error-function-minimized-in.html)



Exercise 1: Is there a closed-form solution to Logistic Regression?

First Derivative

$$\frac{\partial L(\beta)}{\beta_{1j}} = \sum_{i=1}^{N} y_i x_{ij} - \sum_{i=1}^{N} x_{ij} \frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}}$$

$\frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}} \leftarrow p(x_i; \beta)$

$$= \sum_{i=1}^{N} y_i x_{ij} - \sum_{i=1}^{N} p(x; \beta) x_{ij}$$

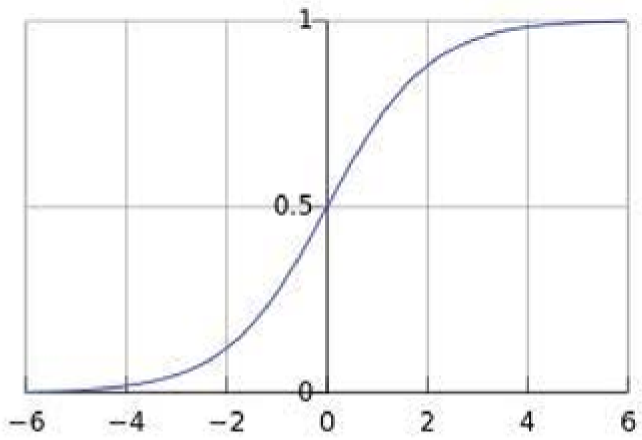
$$= \sum_{i=1}^{N} x_{ij} (y_i - p(x_i; \beta))$$

Now, back to HW 1, Logistic Regression: Why the difference?

Logistic Function

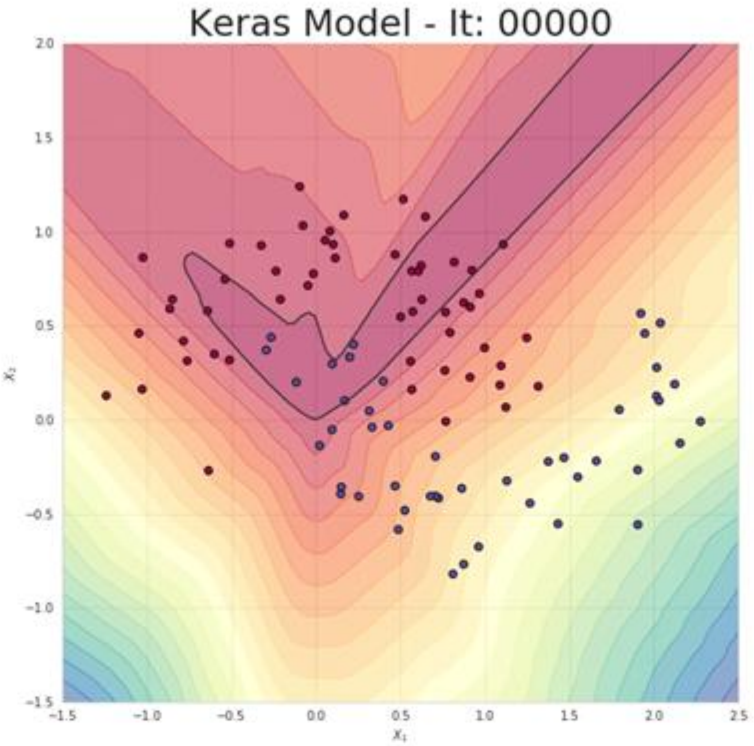
- Logistic Function / sigmoid function:

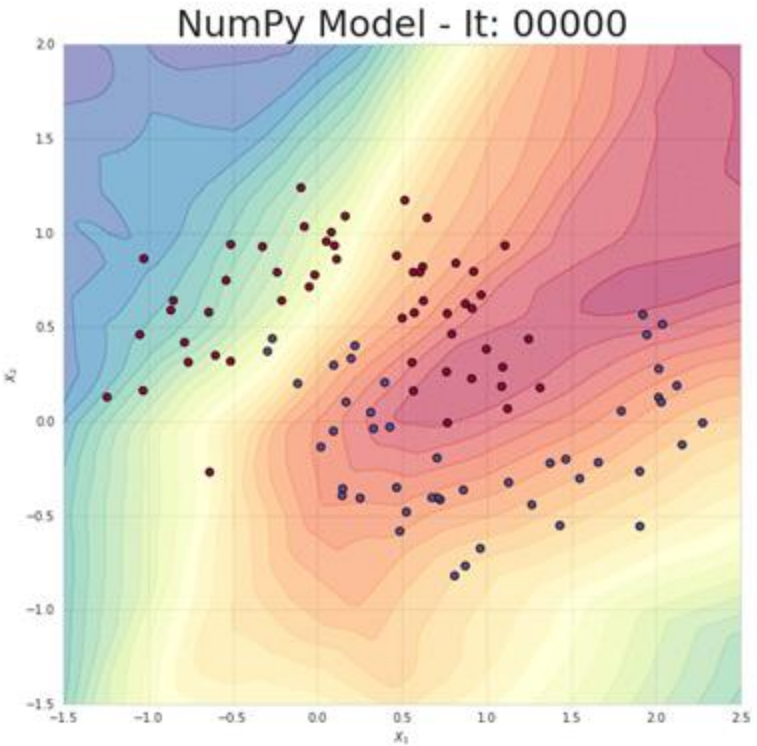
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Neural Networks

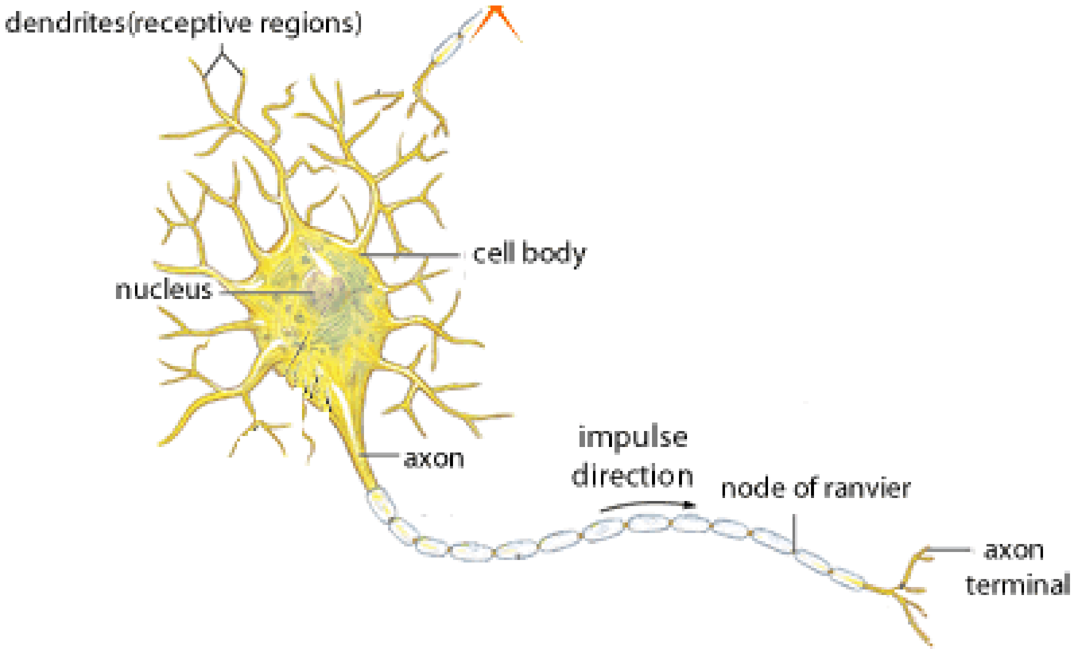
Neural Networks: Nonlinear Decision Boundary

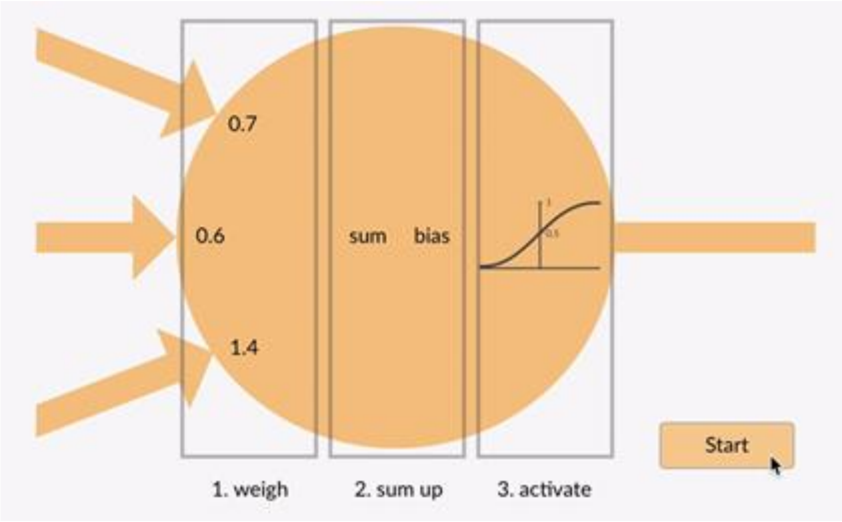




https://towardsdatascience.com/lets-code-a-neural-network-in-plain-numpy-ae7e74410795

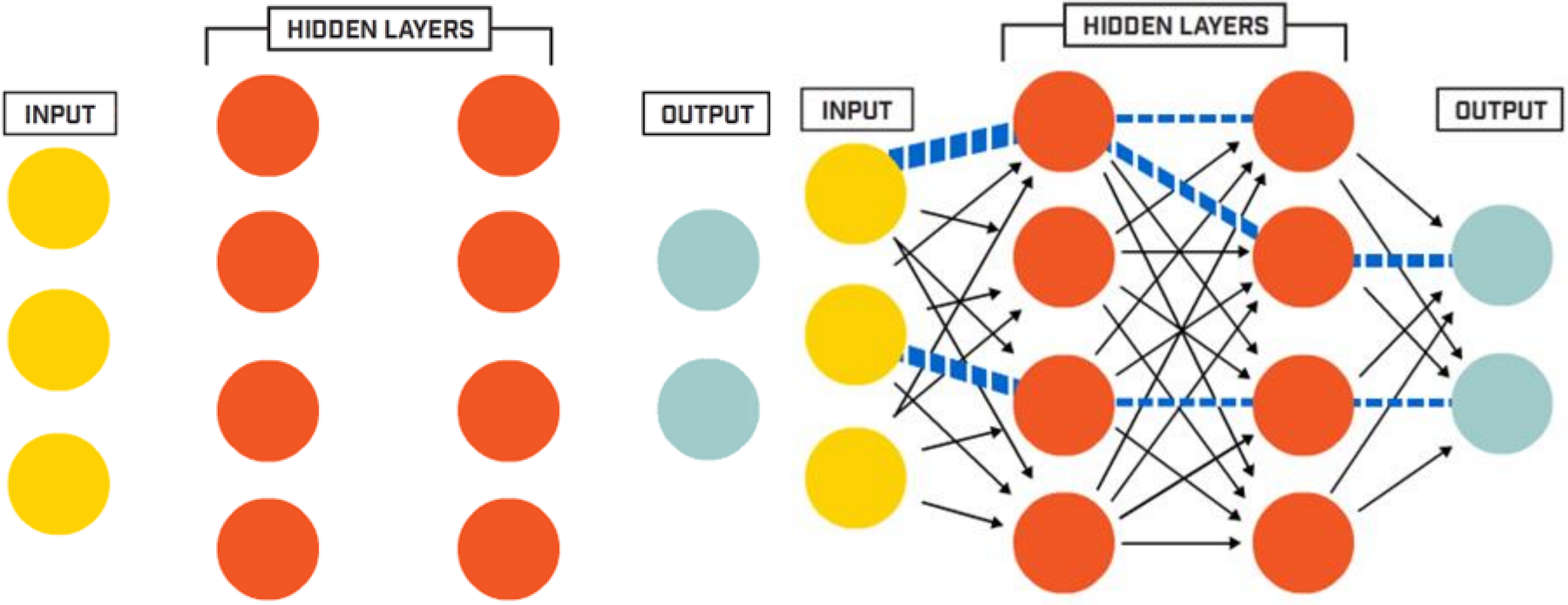
Neural Networks: Neuron/Perceptron





https://medium.com/typeme/lets-code-a-neural-network-from-scratch-part-1-24f0a30d7d62
https://becominghuman.ai/what-is-an-artificial-neuron-8b2e421ce42e

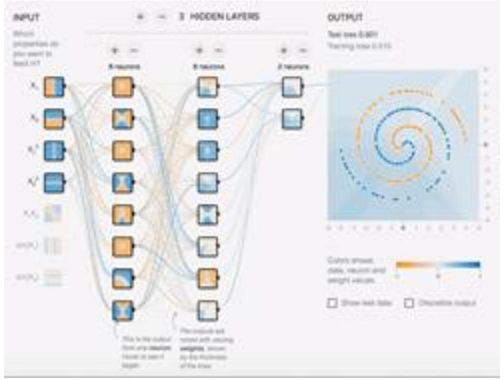
Neural Networks: A Simple Architecture

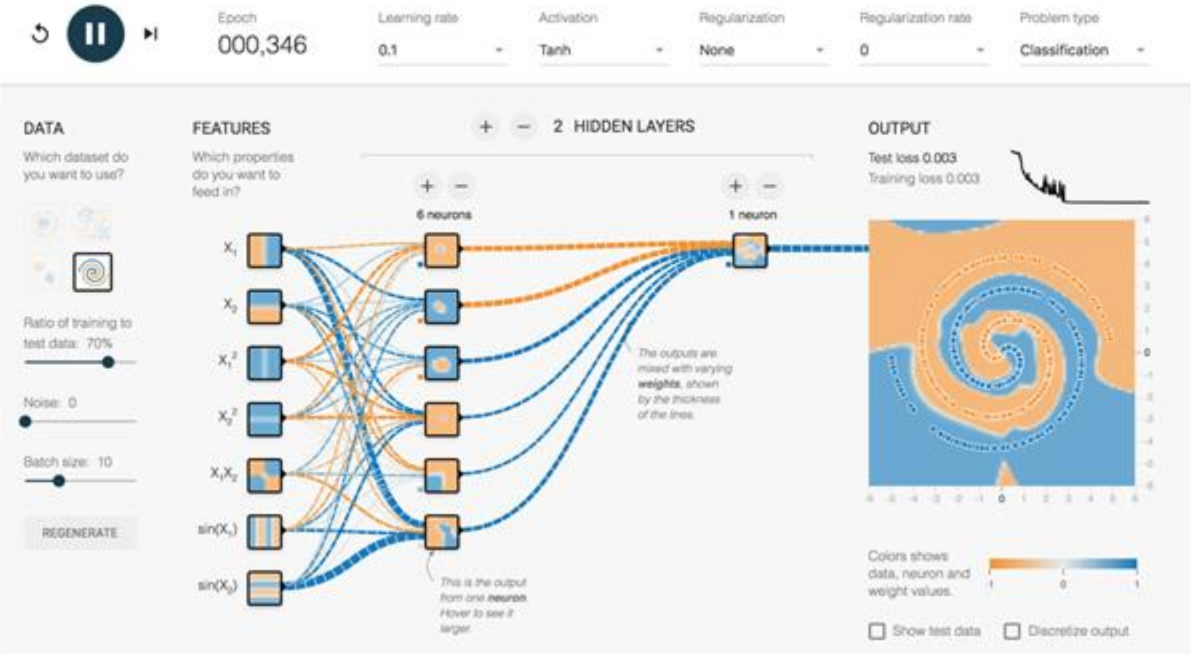


https://www.ptgrey.com/deep-learning

Neural Networks: Flexibility!

- Let's play with it: https://playground.tensorflow.org/





Exercise 2: XOR

- Which NN architecture corresponds to which function?

Y	1	0	1
	0	0	0
		0	1
		X	

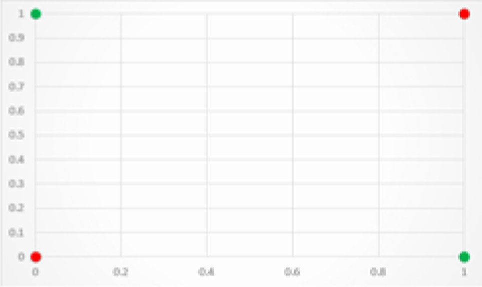
Table 1: Truth table for AND

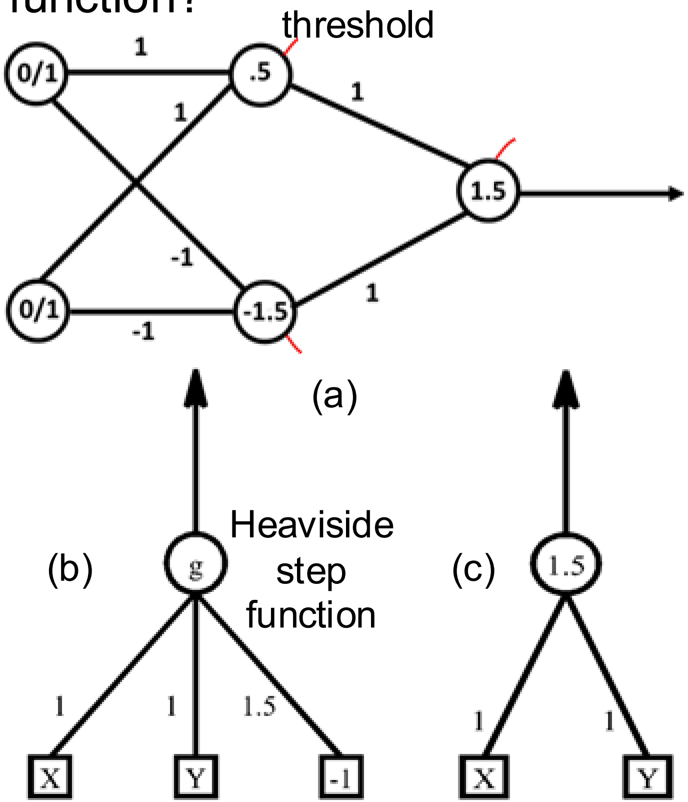
Y	1	1	1
	0	0	1
		0	1
		X	

Table 2: Truth table for OR

Y	1	1	0
	0	0	1
		0	1
		X	

Table 3: Truth Table for XOR





https://datascience.stackexchange.com/questions/11589/creating-neural-net-for-xor-function
http://yen.cs.stir.ac.uk/~kjt/techreps/pdf/TR148.pdf
https://medium.com/@jayeshbahire/the-xor-problem-in-neural-networks-50006411840b

Exercise 2: XOR

		X = 0	X = 1
Y	1	0	1
	0	0	0

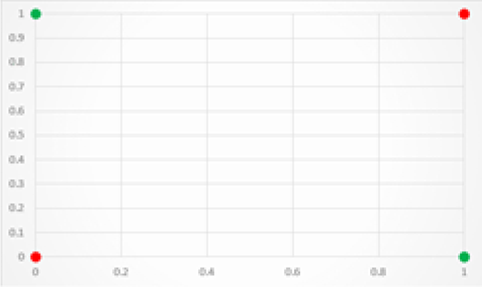
Table 1: Truth table for AND

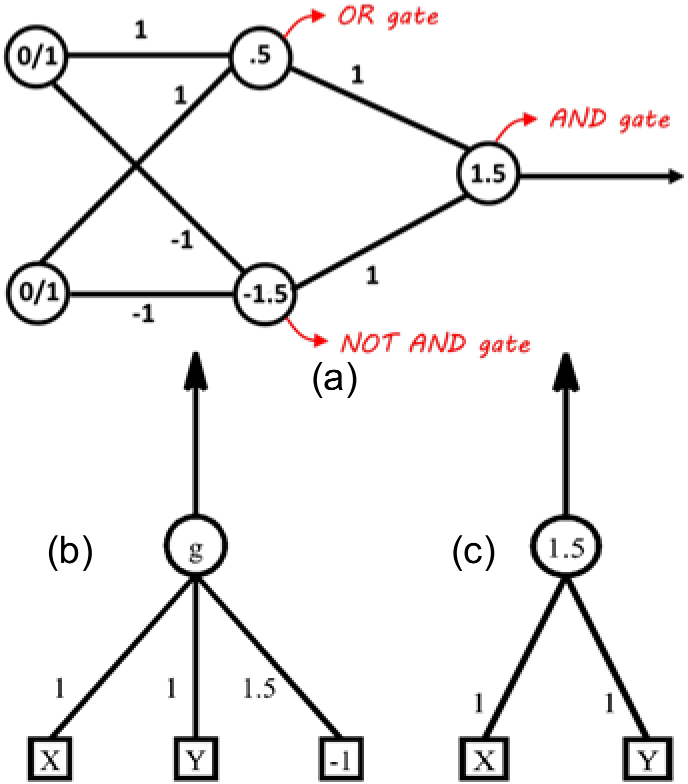
		X = 0	X = 1
Y	1	1	1
	0	0	1

Table 2: Truth table for OR

		X = 0	X = 1
Y	1	1	0
	0	0	1

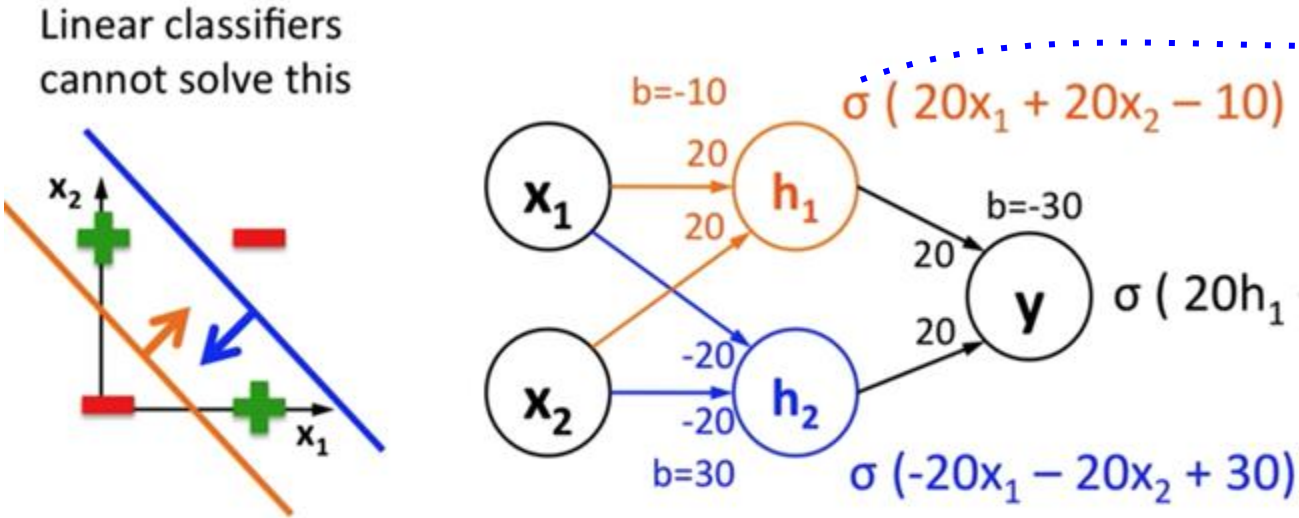
Table 3: Truth Table for XOR





https://datascience.stackexchange.com/questions/11589/creating-neural-net-for-xor-function
http://yen.cs.stir.ac.uk/~kjt/techreps/pdf/TR148.pdf
https://medium.com/@jayeshbahire/the-xor-problem-in-neural-networks-50006411840b

Exercise 2: XOR
Detailed Explanation



We can also use the Heaviside step function instead of sigmoid

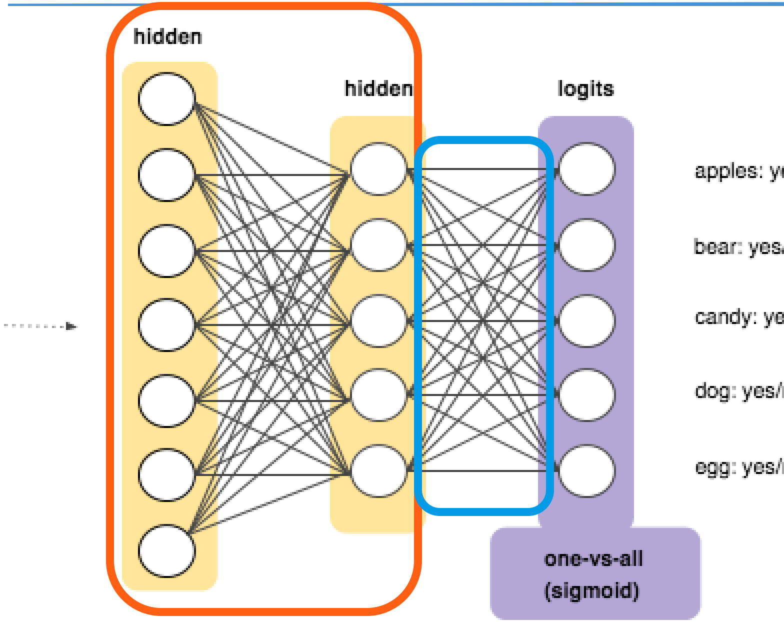
x_1 x_2	x_1 x_2	
$\sigma(20*0 + 20*0 - 10) \approx 0$	$\sigma(-20*0 - 20*0 + 30) \approx 1$	$\sigma(20*0 + 20*1 - 30) \approx 0$
$\sigma(20*1 + 20*1 - 10) \approx 1$	$\sigma(-20*1 - 20*1 + 30) \approx 0$	$\sigma(20*1 + 20*0 - 30) \approx 0$
$\sigma(20*0 + 20*1 - 10) \approx 1$	$\sigma(-20*0 - 20*1 + 30) \approx 1$	$\sigma(20*1 + 20*1 - 30) \approx 1$
$\sigma(20*1 + 20*0 - 10) \approx 1$	$\sigma(-20*1 - 20*0 + 30) \approx 1$	$\sigma(20*1 + 20*1 - 30) \approx 1$

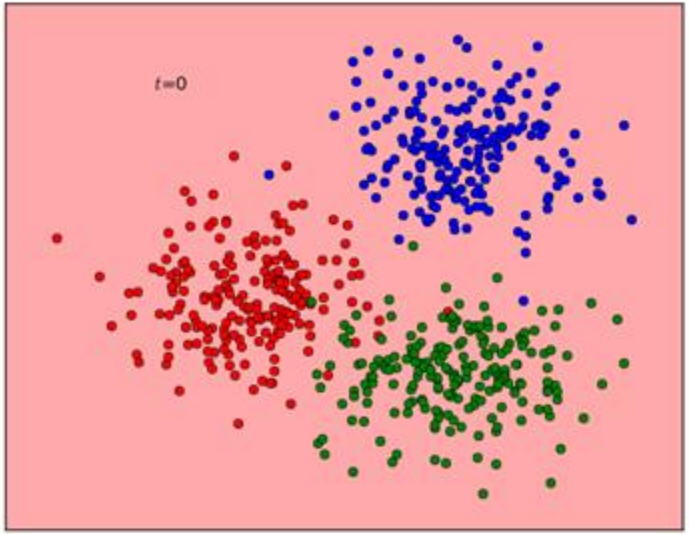
https://www.youtube.com/watch?v=kNPGXgzxoHw

Exercise 3

- Let’s revisit the quiz we did in Monday’s lecture!
- Can linear SVMs be considered as a special case of neural networks?
- How about nonlinear SVMs?
- How about decision trees?

Multiclass Classification





5 separate **binary classifiers**

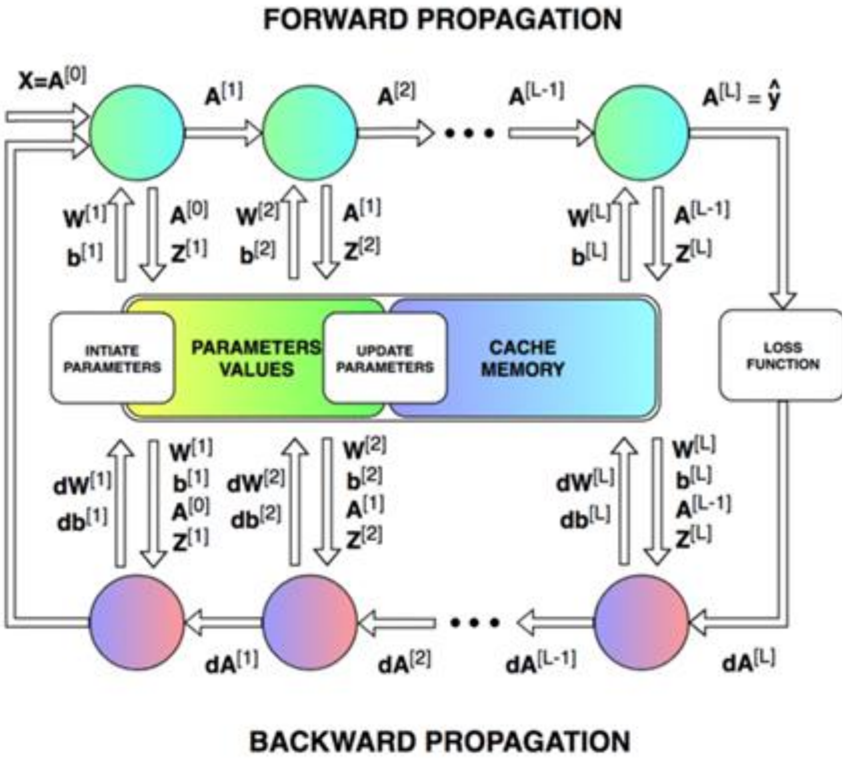
Key: **sharing the same hidden layers** with **different weights at the end**

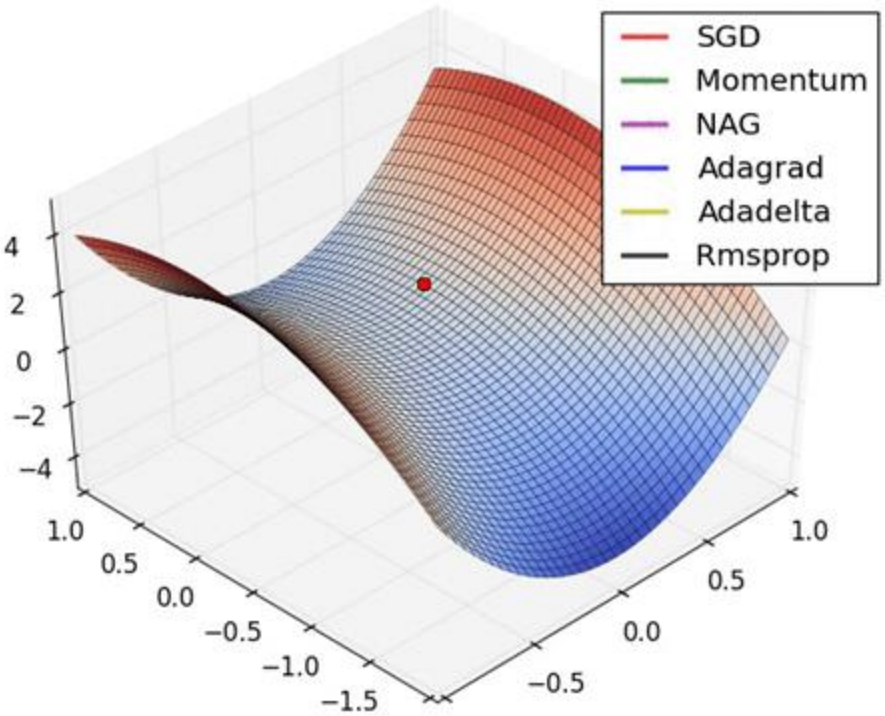
Question: Pros and cons?

https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/one-vs-all

http://www.briandolhansky.com/blog/2013/9/23/artificial-neural-nets-linear-multiclass-part-3

Neural Networks: Backpropagation





https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

How many iterations are needed to converge?

- Depends on:
- Architecture/Meta-parameters of the network, e.g. # layers, activation
- Quality of training data (input-output correlation, normalization, noise cleansing, class distribution/imbalance)
- Random initialization of the parameters/weights
- Optimization algorithm, e.g. SGD, Adam, etc.
- Learning rate
- Batch size
- (In practice) Implementation quality (Bug-free? Optimized?)
- ...

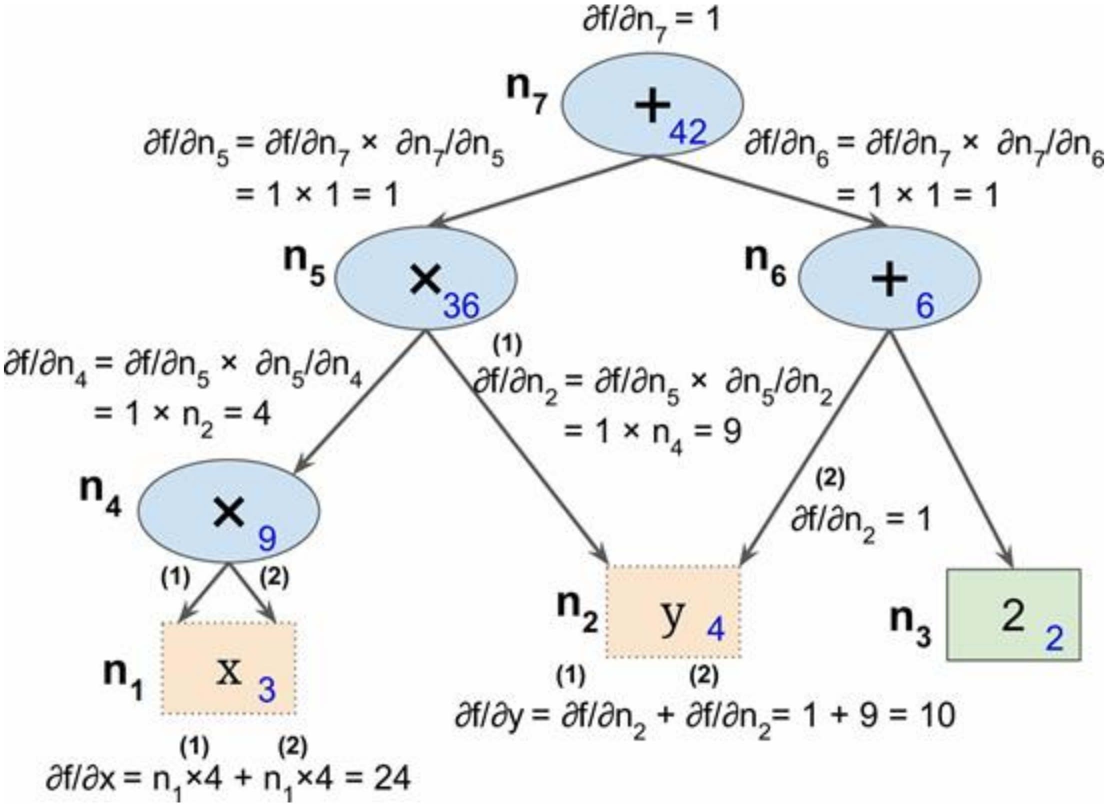
https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e
https://www.quora.com/Machine-Learning-What-are-some-tips-and-tricks-for-training-deep-neural-networks

Neural Networks: Backpropagation

- A simple example to understand the intuition
- **`f(x,y) = x`^{2}`y + y + 2`**
- Forward pass:
  - x = 3, y = 4 → f(3, 4)=42
- Backward pass:
  - Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial n_i} \times \frac{\partial n_i}{\partial x}$$

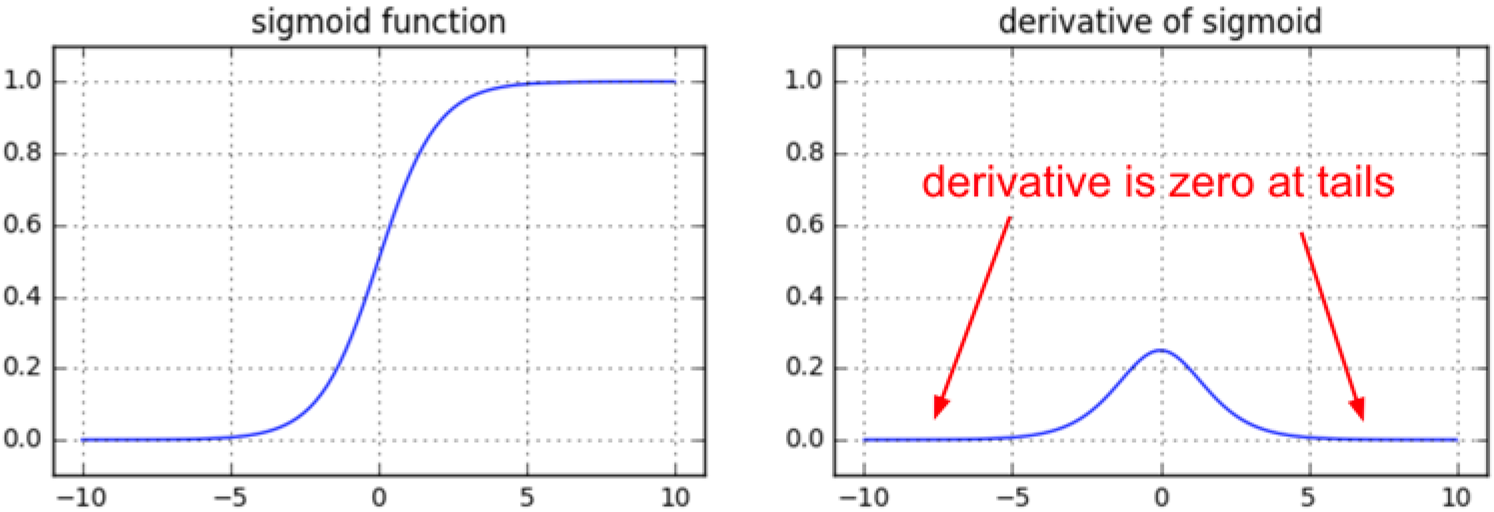
  - See the diagram
- Fun fact: This is called reverse-mode autodiff and how `Tensorflow` works



Géron, Aurélien. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2017.

Why understanding backpropagation?

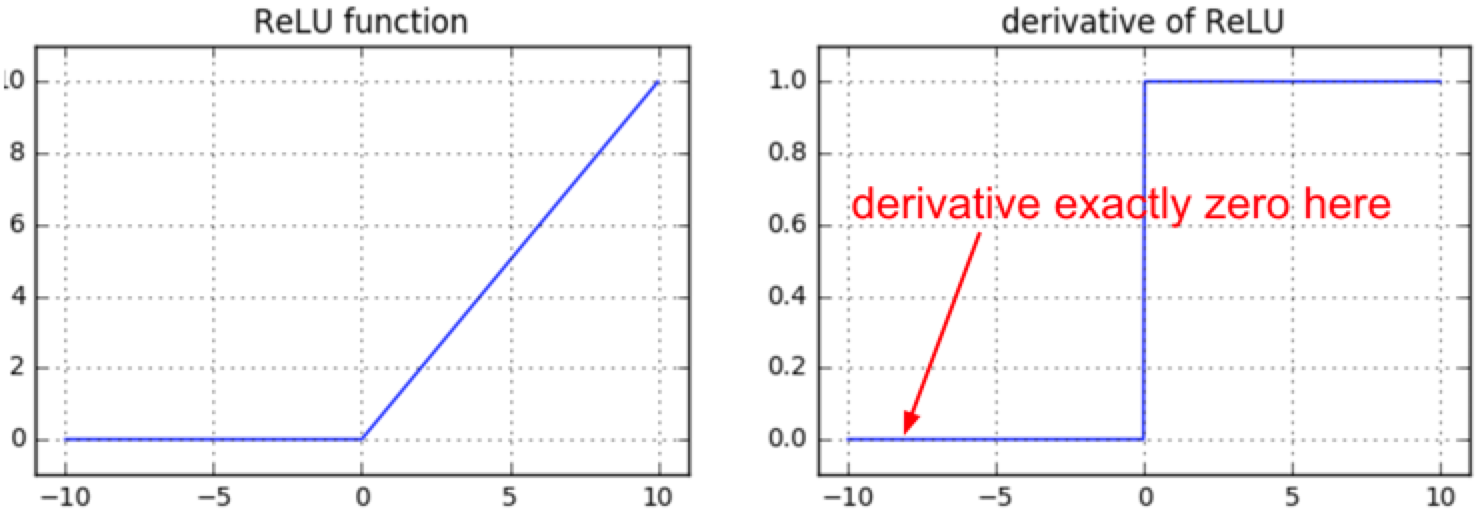
- *"Why do we have to write the backward pass when frameworks in the real world, such as* `TensorFlow`*, compute them for you automatically?"*
- Vanishing gradients on sigmoids



https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b

Why understanding backpropagation?

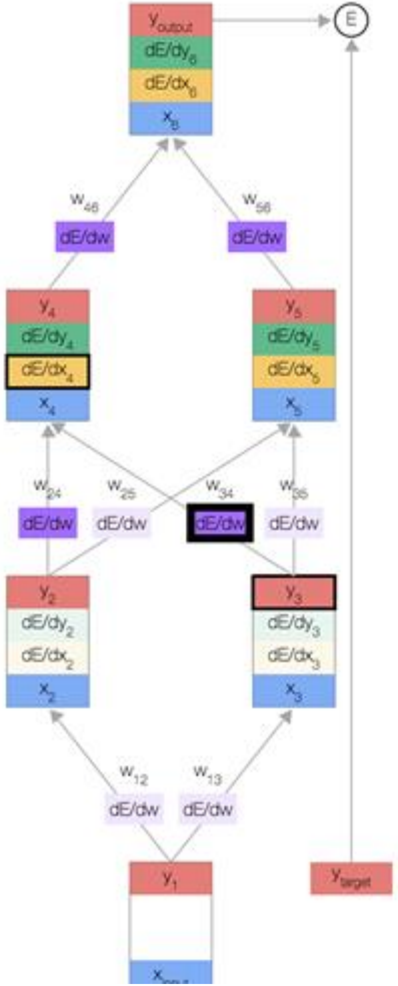
- *"Why do we have to write the backward pass when frameworks in the real world, such as* `TensorFlow`*, compute them for you automatically?"*
- Dying ReLUs



https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b

Neural Networks: Backpropagation

- Backpropagation (Interactive): https://google-developers.appspot.com/machine-learning/crash-course/backprop-scroll/
- Backpropagation (CS 231n at Stanford): https://cs231n.github.io/optimization-2/ and https://www.youtube.com/watch?v=i94OvYb6noo
- (Optional) Matrix-Level Operation: https://medium.com/@14prakash/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c



Exercise 4

- Suppose you have a fully-connected multilayer neural network with 1 input, 2 hidden and 1 output layers. If your dataset has *p* features, the two hidden layers have 3 and 4 neurons respectively, and the output layer has *k* outputs, calculate the number of parameters in the neural network in terms of *p* and *k*. Assume that the bias terms have not been considered in the specified neurons and need to be added to the parameter count.

Q.1 'p' neurons in input layer ; 3 neurons in 1^{st} hidden layer ; 4 neurons in 2^{nd} hidden layer ; 'k' neurons in output layer.

⇒ number of weights w_{ij}

$$= \underbrace{(p \times 3)}_{\text{input to } H_1} + \underbrace{(3 \times 4)}_{H_1 \text{ to } H_2} + \underbrace{(4 \times k)}_{H_2 \text{ to output}}$$

$$= 3p + 4k + 12$$

number of bias θ_j = number of neurons in hidden & output layers

$$= 3 + 4 + k = k + 7$$

⇒ Total number of parameters = $\# w_{ij} + \# \theta_j$

$$= 3p + 4k + 12 + k + 7$$

$$= \boxed{3p + 5k + 19}$$

Exercise 5

- Write down the major steps involved in backpropagation algorithm.

Q.2. Propagate input forward:

$$I_j = \Sigma_i w_{ij} O_i + \theta_j$$

~~$O_j = \sigma(\Sigma_i w_{ij}$~~ $O_j = \sigma(I_j) = \frac{1}{1+e^{-I_j}}$

Backpropagate error:

$Err_j = O_j(1-O_j).(T_j - O_j)$ → output laye

$Err_j = O_j(1-O_j) \cdot \Sigma_k Err_k \cdot w_{jk}$ → hidden lay

$$w_{ij} = w_{ij} + \eta \, Err_j \, O_i$$

$$\theta_j = \theta_j + \eta \, Err_j$$

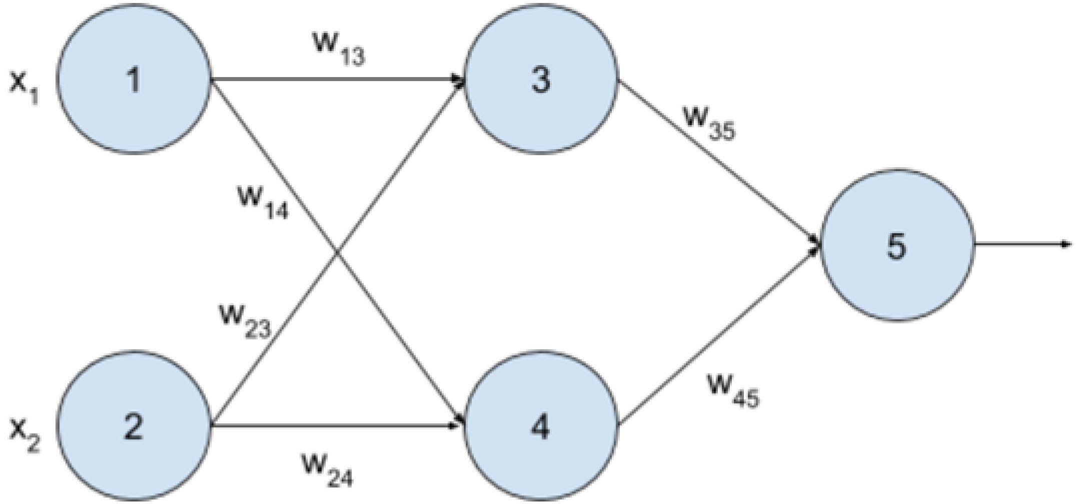
Exercise 5

- Given the following multilayer neural network, a training data point $x=(x_1=0,x_2=1)$, and the target value T=1, please calculate weights and bias after 1 iteration of backpropagation algorithm (show your calculations and fill out the empty tables given below). The learning rate =0.8. The initial weights and bias are in the following table.

w_{13}	w_{14}	w_{23}	w_{24}	w_{35}	w_{45}	3	4	5
-0.3	0.2	0.4	-0.1	-0.2	-0.3	0.2	-0.4	0.1

UCLA

Exercise 5



Net Input and Output Calculations

Unit, j	Net Input, I_j	Output, O_j
3	- 0.3 (0) + 0.4 (1) + 0.2 = 0.6	0.6457
4	0.2 (0) - 0.1 (1) – 0.4 = - 0.5	0.3775
5	- 0.2 (0.6457) – 0.3 (0.3775) + 0.1 = - 0.14239	0.4645

Calculation of the error at each node

Pay attention to whether it is **err** or **derivative**.

Unit, j	Err_j
5	0.4645 (1 - 0.4645) (1 – 0.4645) = 0.1332
4	0.3775 (1 – 0.3775) (0.1332) (- 0.3) = - 0.0094
3	0.6457 (1 – 0.6457) (0.1332) (- 0.2) = - 0.0061

Pay attention to the sign here! If **err**, +; If **derivative**, - (~SGD).

Calculations for weight and bias updating

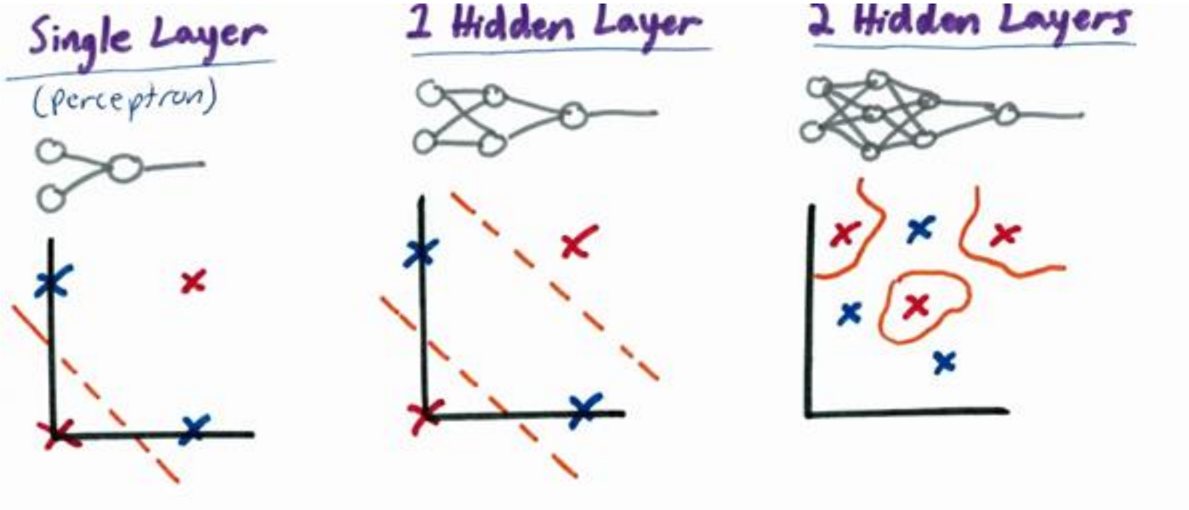
Weight or Bias	New Value
w_{35}	- 0.2 + 0.8 (0.1332) (0.6457) = - 0.1312
w_{45}	- 0.3 + 0.8 (0.1332) (0.3775) = - 0.2598
w_{13}	- 0.3 + 0.8 (- 0.0061) (0) = - 0.3
w_{14}	0.2 + 0.8 (- 0.0094) (0) = 0.2
w_{23}	0.4 + 0.8 (- 0.0061) (1) = 0.3951
w_{24}	- 0.1 + 0.8 (- 0.0094) (1) = - 0.1075
θ_5	0.1 + 0.8 (0.1332) = 0.2066
θ_4	- 0.4 + 0.8 (- 0.0094) = - 0.4075
θ_3	0.2 + 0.8 (- 0.0061) = 0.1951

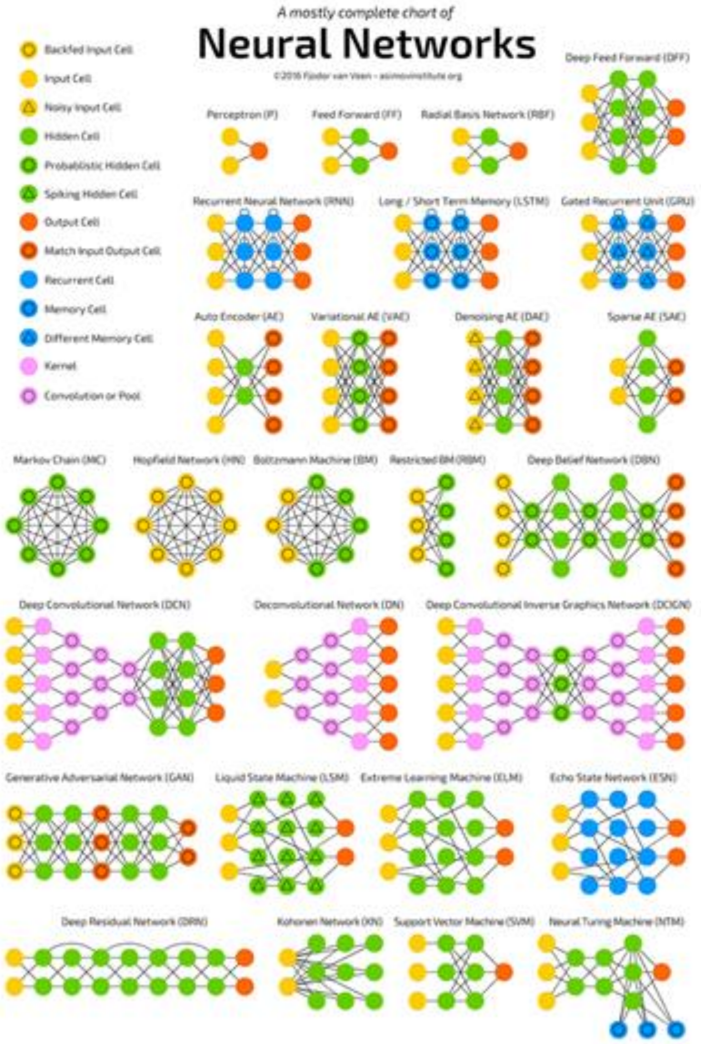
Neural Networks: Pros and Cons

- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or "structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of "hidden units" in the network
- Strength
  - High tolerance to noisy data
  - Successful on an array of real-world data, e.g., hand-written letters
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks
  - Deep neural network is powerful

UCLA

Neural Networks: Pros and Cons

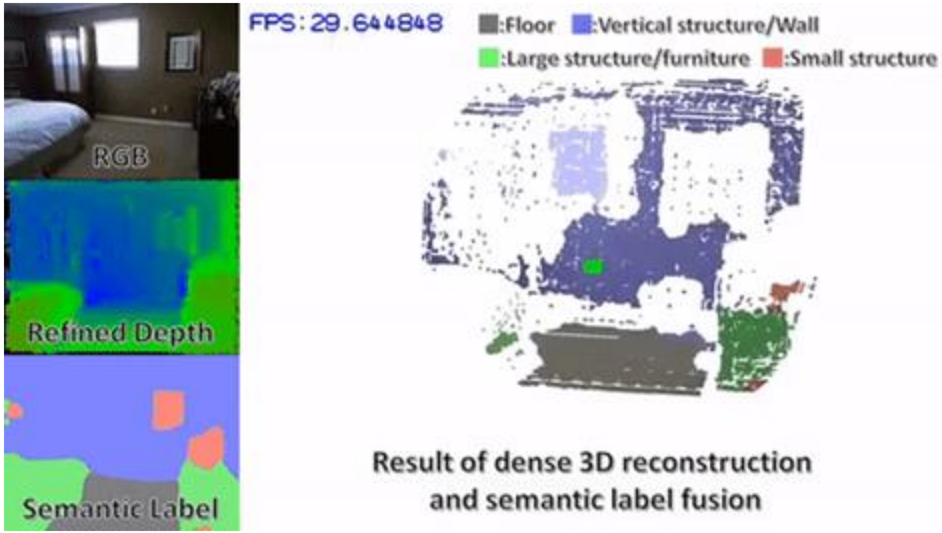


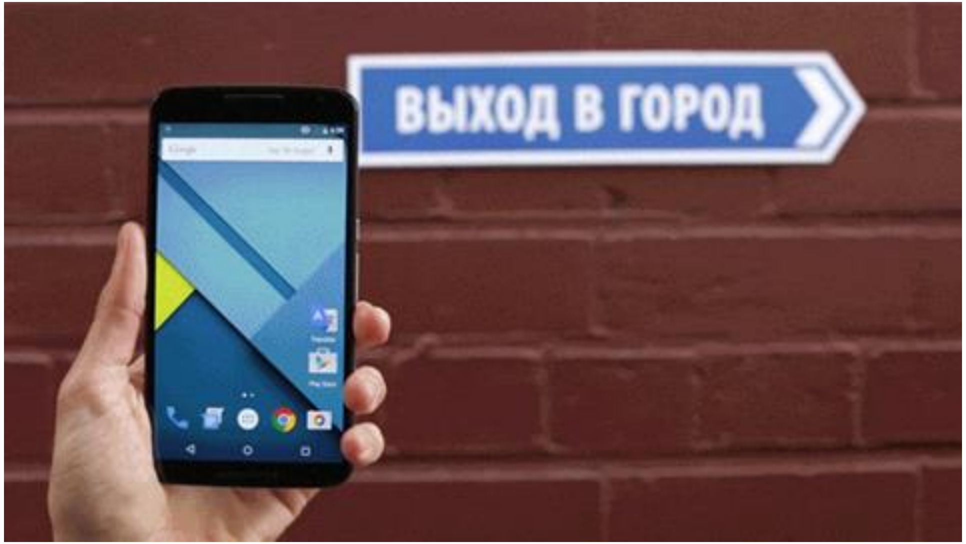


Flexibility

https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec27/pros-and-cons-of-neural-networks
http://kseow.com/nn/
https://towardsdatascience.com/hype-disadvantages-of-neural-networks-6af04904ba5b

Neural Networks: Pros and Cons

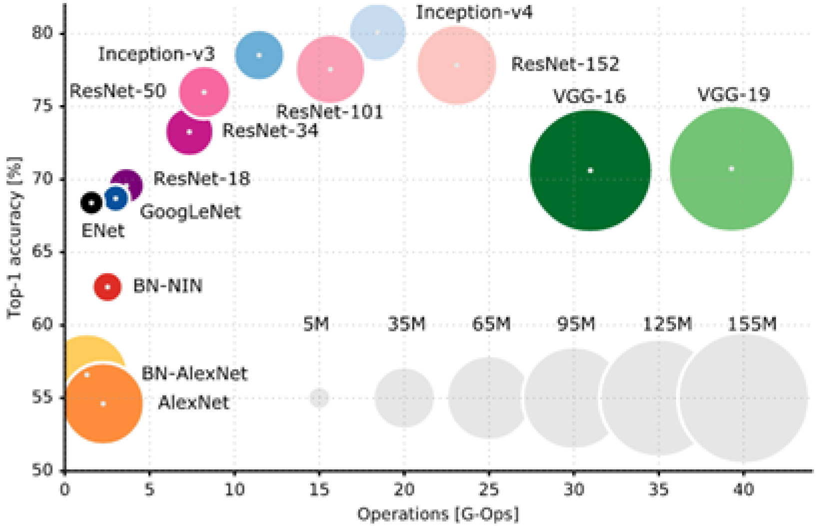




Efficiency (In many cases, prediction/inference/testing is fast)

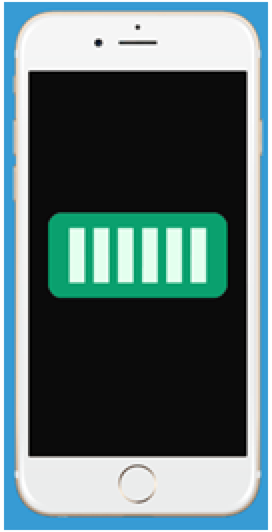
https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec27/pros-and-cons-of-neural-networks
http://www.luigifreda.com/2017/04/08/cnn-slam-real-time-dense-monocular-slam-learned-depth-prediction/
http://www.missqt.com/google-translate-app-now-supports-instant-voice-and-visual-translations/

Neural Networks: Pros and Cons



We trained both our baseline models for about 600,000 iterations (33 epochs) – this is similar to the 35 epochs required by Nallapati et al.'s (2016) best model. Training took 4 days and 14 hours for the 50k vocabulary model, and 8 days 21 hours for the 150k vocabulary model. We found the pointer-generator model quicker to train, requiring less than 230,000 training iterations (12.8 epochs); a total of 3 days and 4 hours. In particular, the pointer-generator model makes much quicker progress in the early phases of training.

ments. This work was begun while the first author was an intern at Google Brain and continued at Stanford. Stanford University gratefully acknowl-



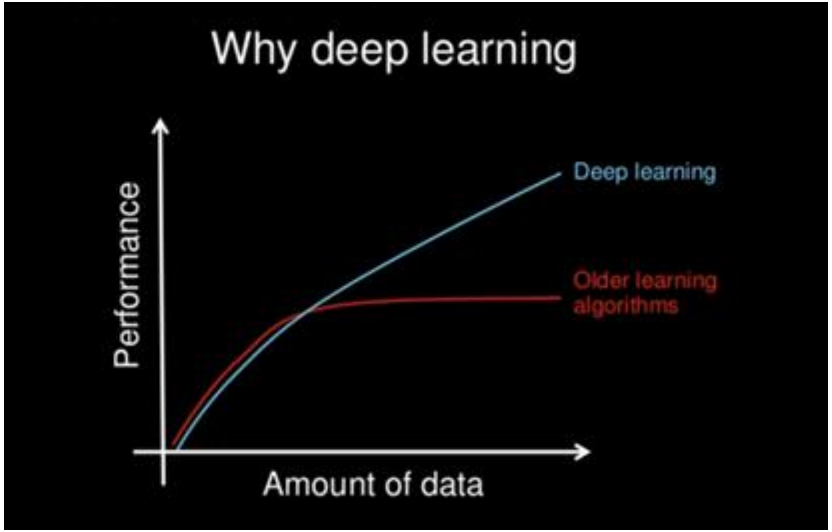
Efficiency (Big model → slow training, huge energy consumption (e.g. for cell phone))

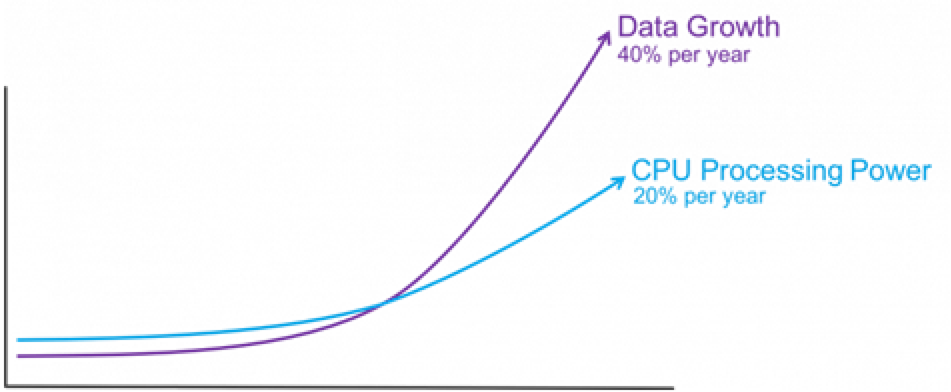
https://www.kdnuggets.com/2017/08/first-steps-learning-deep-learning-image-classification-keras.html/2

See, Abigail, Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." *arXiv preprint arXiv:1704.04368* (2017).

https://www.lifewire.com/my-iphone-wont-charge-what-do-i-do-2000147

Neural Networks: Pros and Cons





Data (Both a pro and a con)

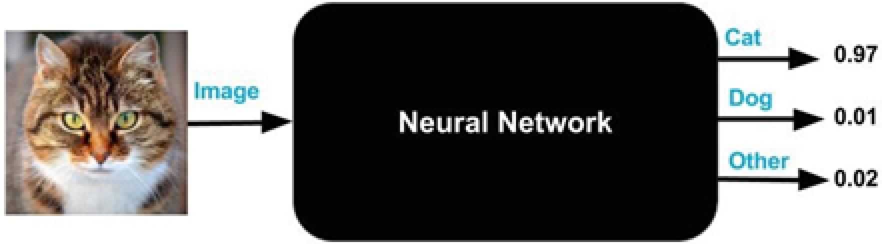
https://towardsdatascience.com/hype-disadvantages-of-neural-networks-6af04904ba5b

Neural Networks: Pros and Cons

Computational Power (Both a pro and a con)

https://www.anandtech.com/show/10864/discrete-desktop-gpu-market-trends-q3-2016
https://www.zdnet.com/article/gpu-killer-google-reveals-just-how-powerful-its-tpu2-chip-really-is/

Neural Networks: Pros and Cons



Black Box
Interpretability

https://towardsdatascience.com/hype-disadvantages-of-neural-networks-6af04904ba5b
https://xkcd.com/1838/

Homework 2: Notes

HW 2, Prob 1.2: Numerically Different

- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D (conditional entropy):

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

Step (1) Step (2)

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

HW 2, Prob 1.2: Numerically Different

- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D (conditional entropy):

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

Step (1)

Step (2)

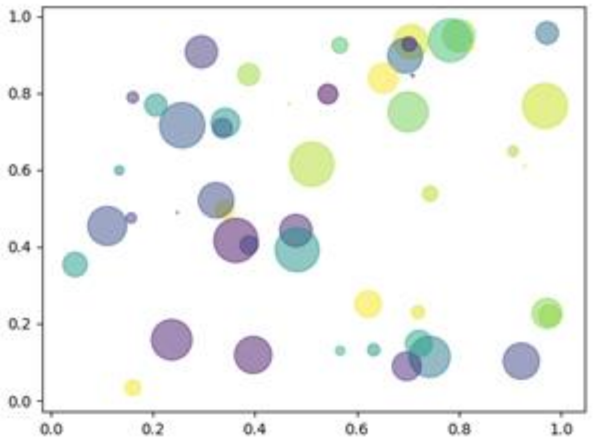
- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

HW 2, Problem 2.1 (f): How to Plot with Python?

- ***Use `Matplotlib`***
- A simple example:

```
import numpy as np
import matplotlib.pyplot as plt
# Fixing random state for reproducibility
np.random.seed(19680801)
N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2  # 0 to 15 point radii
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```



https://matplotlib.org/gallery/shapes_and_collections/scatter.html#sphx-glr-gallery-shapes-and-collections-scatter-py

HW 2: Numpy vs Pandas

```
import numpy as np
import pandas as pd
np.random.seed(123)
X = pd.DataFrame(np.random.randint(0,2,size=(2,
4)), columns=list('ABCD'))
print(type(X))
print(X)
way_one = np.dot(X, X.T)
way_two = X.dot(X.T)

print(type(way_one))
print(type(way_two))
print(way_one)
print(way_two)
```

```
<class 'pandas.core.frame.DataFrame'>
   A  B  C  D
0  0  1  0  0
1  0  0  0  1

<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
[[1 0]
 [0 1]]
   0  1
0  1  0
1  0  1
```

Which one to choose? Depends on how you use the result in the homework!

In general, it is a good practice to ***be always aware of the data type*** of the variables you use!

HW 2: Numpy vs Pandas

```
import numpy as np
import pandas as pd
np.random.seed(123)
X_np = np.random.randint(0,2,size=(2, 4))
X_df = pd.DataFrame(X_np, columns=list('ABCD'))
print(type(X_np))
print(X_np)
print()
print(type(X_df))
print(X_df)
print()
print(type(X_np[0]))
# print(type(X_df[0])) # won't work
print(type(X_df.iloc[0]))
print(type(X_df.iloc[[0]]))
print(type(X_df.values[0]))
```

```
<class 'numpy.ndarray'>
[[0 1 0 0]
 [0 0 0 1]]

<class 'pandas.core.frame.DataFrame'>
   A  B  C  D
0  0  1  0  0
1  0  0  0  1

<class 'numpy.ndarray'>

<class 'pandas.core.series.Series'>
<class 'pandas.core.frame.DataFrame'>
<class 'numpy.ndarray'>
```

https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/

Thank you!

Q & A