Week 7

Agenda

1. Project 4 check in
2. Breakout Project 2
3. Neural Network discussion

Project 4 - Do you have a group yet?

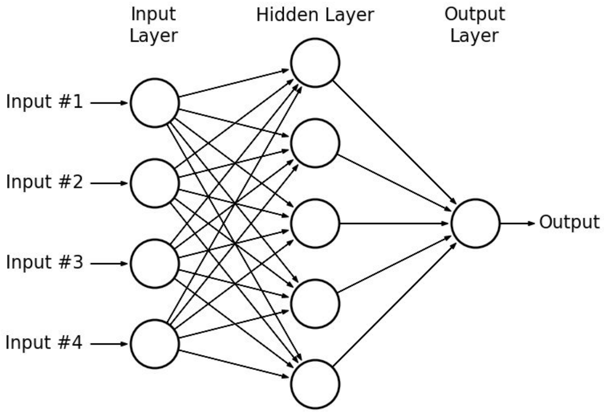
Some guidelines

Give a 10 minute presentation on **March 09** in class. You don't need to submit anything at this time.

- Not everyone has to speak
- Feel free to use slides or python notebooks
- Introduce your group
- Introduce your data set
- Introduce your problem statement
- Tell us how you think you are going to solve the problem

- Purpose: make sure that you have met w/ your group and started seriously thinking about the project.
- Typically the start of a project involves a good amount of EDA (exploratory data analysis).
- For your baseline presentation, if you have some results from a simple or baseline model, feel free to share results

History



Timeline

- 40s-50s Idea emerges.
- 1962 Perceptron learning
- 1969 Minisky: XOR problem
- 1982 Multi-layer neural networks
- 1986 Backpropagation
- 1989 Universal Approximation Theorem
- 90s-00s SVMs gain favor
- 2009-present Deep learning: return of neural nets

People to know/Papers to read

- Geoffrey Hinton. http://www.cs.toronto.edu/~hinton/ (https://www.coursera.org/learn/neural-networks)
- Yann LeCun. http://yann.lecun.com/
- Yoshua (and Samy) Bengio. http://www.iro.umontreal.ca/~bengioy/yoshua_en/
- Leon Bottou (SGD). http://leon.bottou.org/

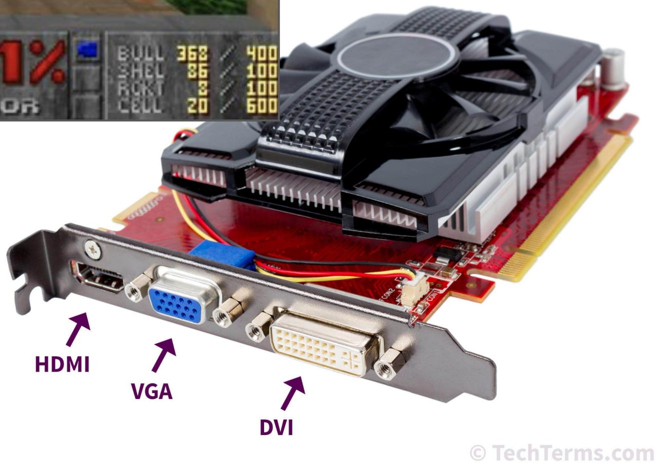
Conferences

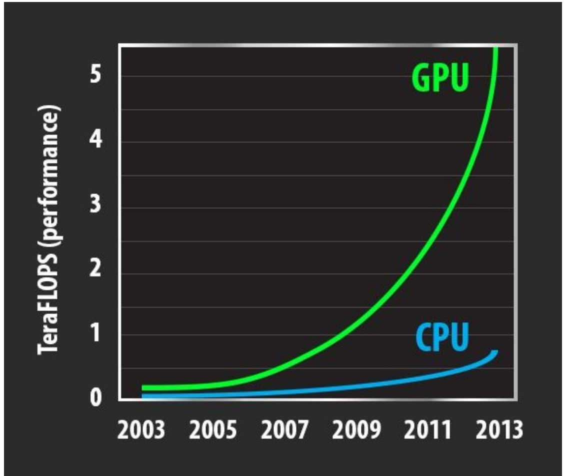
- Academic: NIPS (very popular, hard to get a ticket), ICML
- Appplied: KDD, SIGIR, AAAI

GPUs

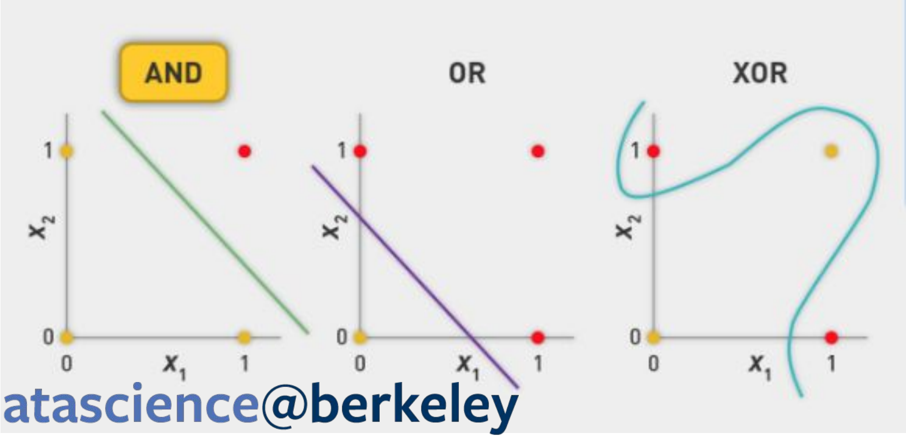
1. GPUs grew out of gaming in the 90s
2. GPUs do vector/matrix/tensor operations faster than CPUs.

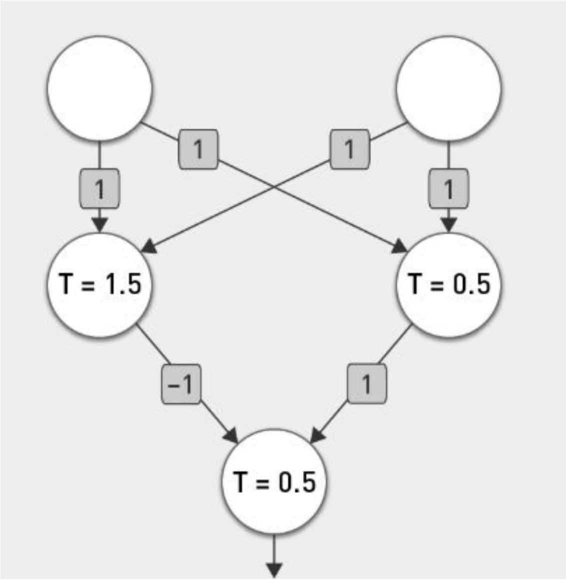
Why? GPU vs CPU? When should you use which?





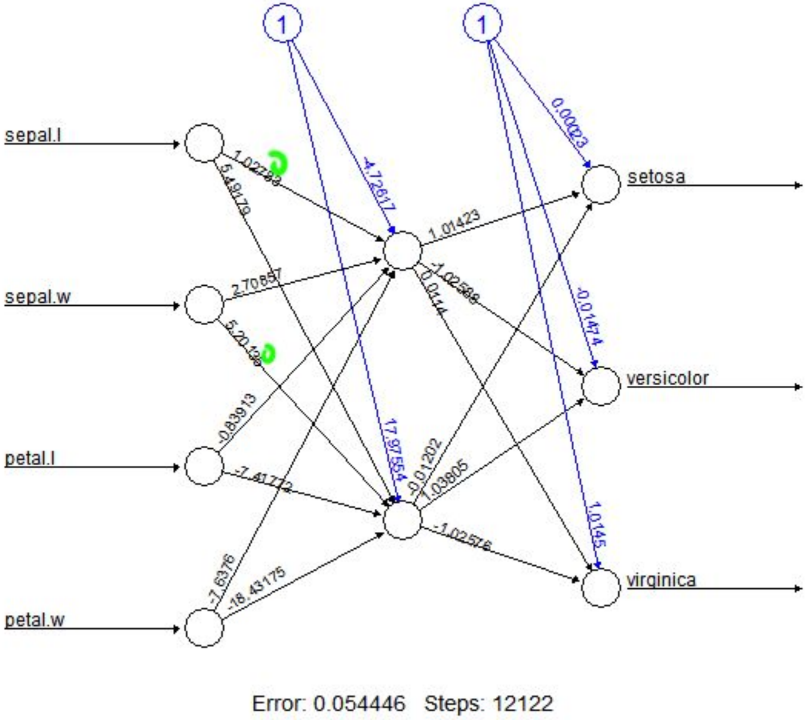
1. What's the limitation of a perceptron?
2. What differs about Neural Nets that allow them to learn non-linear function?





datascience@berkeley

Example Trained Neural Network



1. What do you remember about Iris dataset?
2. How many parameters in this model?
3. How is multi-class handled?
4. Sparse vs dense representation. Which one will we get? Why?
5. How many layers?
6. How can we think about this network as an ensemble/stacked model?
7. How can we think about this network as a series of matrix operations?

datascience@berkeley

Accuracy on MNIST

Type	Classifier	Distortion	Preprocessing	Error rate (%)
Linear classifier	Pairwise linear classifier	None	Deskewing	7.6[9]
Non-Linear Classifier	40 PCA + quadratic classifier	None	None	3.3[9]
Neural network	2-layer 784-800-10	None	None	1.6[17]
Boosted Stumps	Product of stumps on Haar features	None	Haar features	0.87[15]
Neural network	2-layer 784-800-10	elastic distortions	None	0.7[17]
Support vector machine	Virtual SVM, deg-9 poly, 2-pixel jittered	None	Deskewing	0.56[16]
K-Nearest Neighbors	K-NN with non-linear deformation (P2DHMDM)	None	Shiftable edges	0.52[14]
Deep neural network	6-layer 784-2500-2000-1500-1000-500-10	elastic distortions	None	0.35[18]
Convolutional neural network	Committee of 35 conv. net, 1-20-P-40-P-150-10	elastic distortions	Width normalizations	0.23[8]

Universal Approximation Theorem

- Two-layer networks are universal function approximators
  - Let F be a continuous function on a bounded subset of D-dimensional space. Then there exists a two-layer neural network F' with a finite number of hidden units that approximate F arbitrarily well. Namely, for all x in the domain of F,

$$|F(x) - F'(x)| < \varepsilon$$

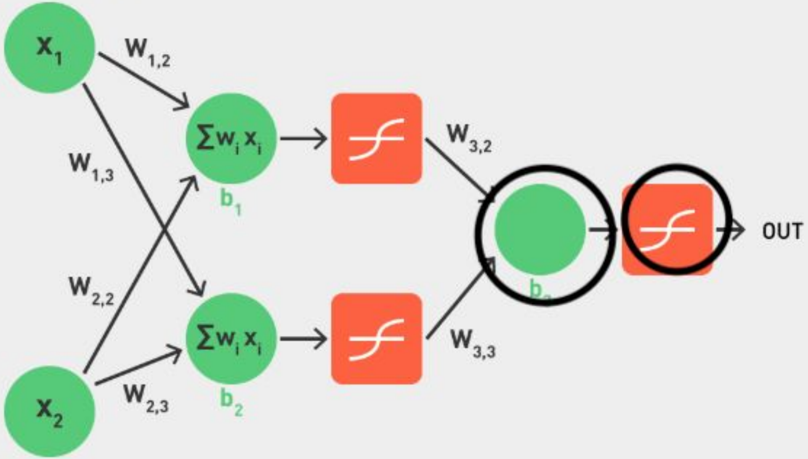
- Two-layer networks can approximate any function.
- Still may want more than two layers (fewer neurons, time to learn, time to compute, etc).

1. Why is this a theorem about representation rather than learning?

Training and Predicting

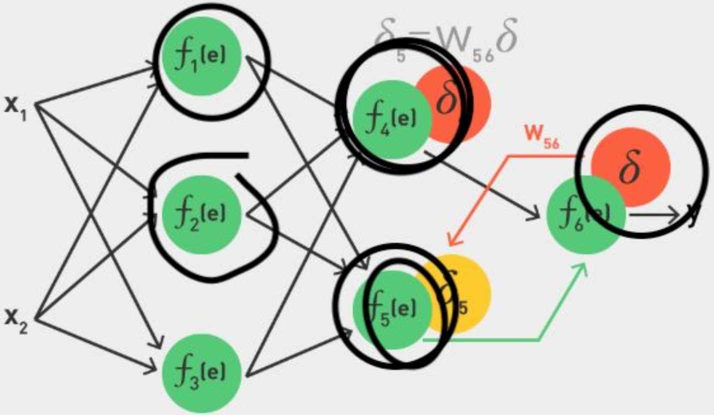
Intuition: Forward Propagation

- Given a training example (X_1, X_2) and output Y_i
- Propagate inputs/activations forward, applying sigmoid function on dot products



Intuition: Backward Propagation (cont.)

Propagate costs backward to earlier nodes:

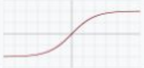
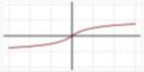


- For each hidden unit h in k^{th} layer:

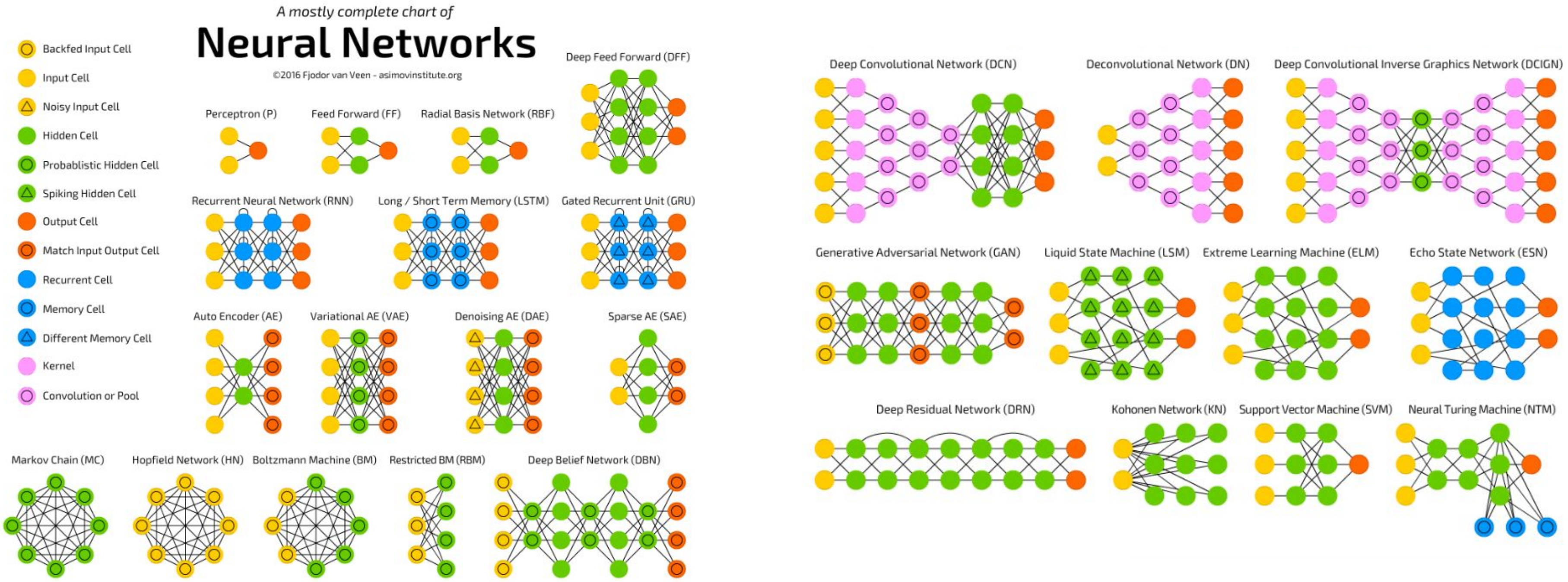
$$\delta_{hk} = Y_{hk}(1 - Y_{hk}) \sum_{j \in K} w_{hj}\delta_j$$

- Update each weight as $+\eta\delta_{hk}x_i$.
  - Daume ch. 8 for full algorithm

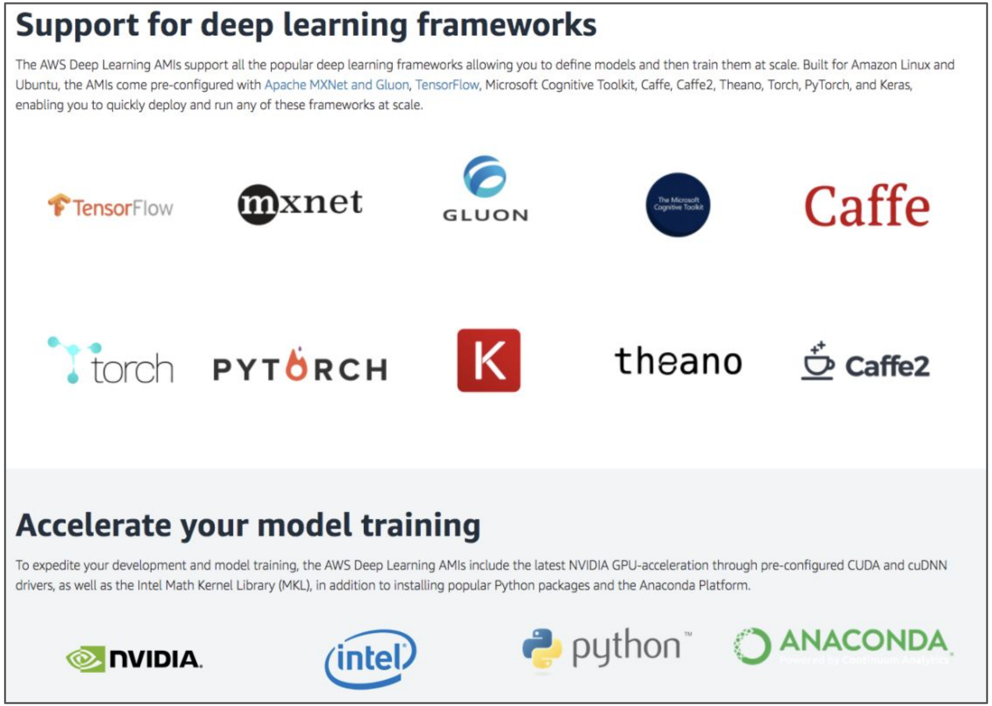
Activation Functions are Active Area of Research

Name	Plot	Equation	Derivative (with respect to x)	Range
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$
Softsign [7][8]		$f(x) = \frac{x}{1 + \|x\|}$	$f'(x) = \frac{1}{(1 + \|x\|)^2}$	$(-1, 1)$
Rectified linear unit (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$
Leaky rectified linear unit (Leaky ReLU)[10]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Parameteric rectified linear unit (PReLU)[11]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$

Beyond Basic FF Networks -- Many Architectures



Neural Network Frameworks



- Large number of NN libraries have emerged in recent years
- All support GPUs

NN Frameworks

TensorFlow Vs Theano Vs Torch Vs Keras Vs infer.net Vs CNTK Vs MXNet Vs Caffe: Key Differences

Library	Platform	Written in	Cuda support	Parallel Execution	Has trained models	RNN	CNN
Torch	Linux, MacOS, Windows	Lua	Yes	Yes	Yes	Yes	Yes
Infer.Net	Linux, MacOS, Windows	Visual Studio	No	No	No	No	No
Keras	Linux, MacOS, Windows	Python	Yes	Yes	Yes	Yes	Yes
Theano	Cross-platform	Python	Yes	Yes	Yes	Yes	Yes
TensorFlow	Linux, MacOS, Windows, Android	C++, Python, CUDA	Yes	Yes	Yes	Yes	Yes
MICROSOFT COGNITIVE TOOLKIT	Linux, Windows, Mac with Docker	C++	Yes	Yes	Yes	Yes	Yes
Caffe	Linux, MacOS, Windows	C++	Yes	Yes	Yes	Yes	Yes
MXNet	Linux, Windows, MacOs, Android, iOS, Javascript	C++	Yes	Yes	Yes	Yes	Yes