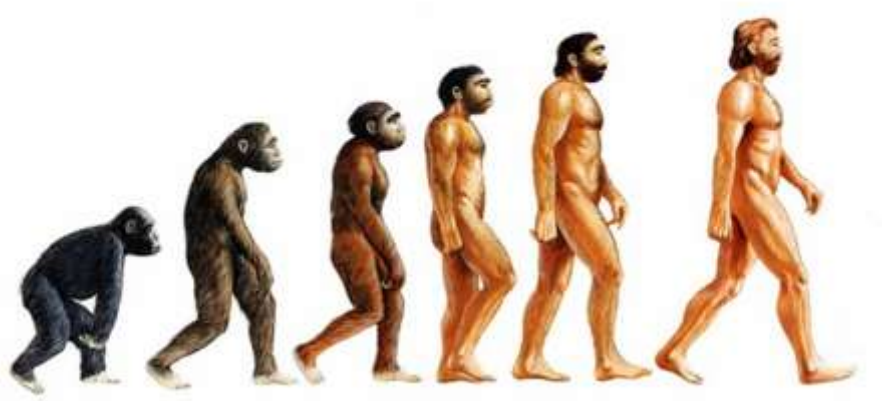
Learning Machines use Neural Networks to play Super Mario Bros.

SEBASTIAN WAZ ▪ MAY 20, 2015 ▪ CS 35L ▪ LAB 3: TAI-LIN CHU

So… what is machine learning?

- Deep philosophical question of what is knowledge?
  - Searle's *Chinese room* thought experiment
  - Is knowledge discrete? Is knowledge function?
- We typically see intelligent behavior as a robust relationship between output and diverse input
- **Machine learning**: machines do not have strategies explicitly programmed; rather, they refine their strategies ("learn") through interaction with new data.
- The important technological question: what models can we use do this most effectively?

Genetic model for machine learning



- **Genetic algorithms – model: natural selection**
- Requires a genetic representation of the decision algorithm
  - This means that the traits which define an algorithm's structure are coded by "genes"
  - For example: an algorithm may have a gene called "traversal" with code "pre-order"
  - Mutiple algorithms in a generation, each with the **same set of genes**, but **with different codes** on each gene
- Based on definition of **fitness**, some algorithms will be thrown away.
  - New ones generated by crossover and mutation of genes of surviving algorithms
  - Algortithms evolve over multiple generations

Q model for machine learning

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

- **Q-learning – model: reinforcement**
- Q = the usefulness of a particular action given a situation, based on immediate and long term reward.
- **Q is a recursive function (con't until terminal state reached):**

$Q(s_t, a_t) = R(s_t, a_t) + \gamma * (best(Q(s_{t+1}, a_{t+1})))$

- **R** = immediate reward of action
- **best($Q(s_{t+1}, a_{t+1})$)** = Q of best action of all possible next possible actions
- γ = time discount, 0:1.
  - If γ = 0, learner only cares about immediate reward

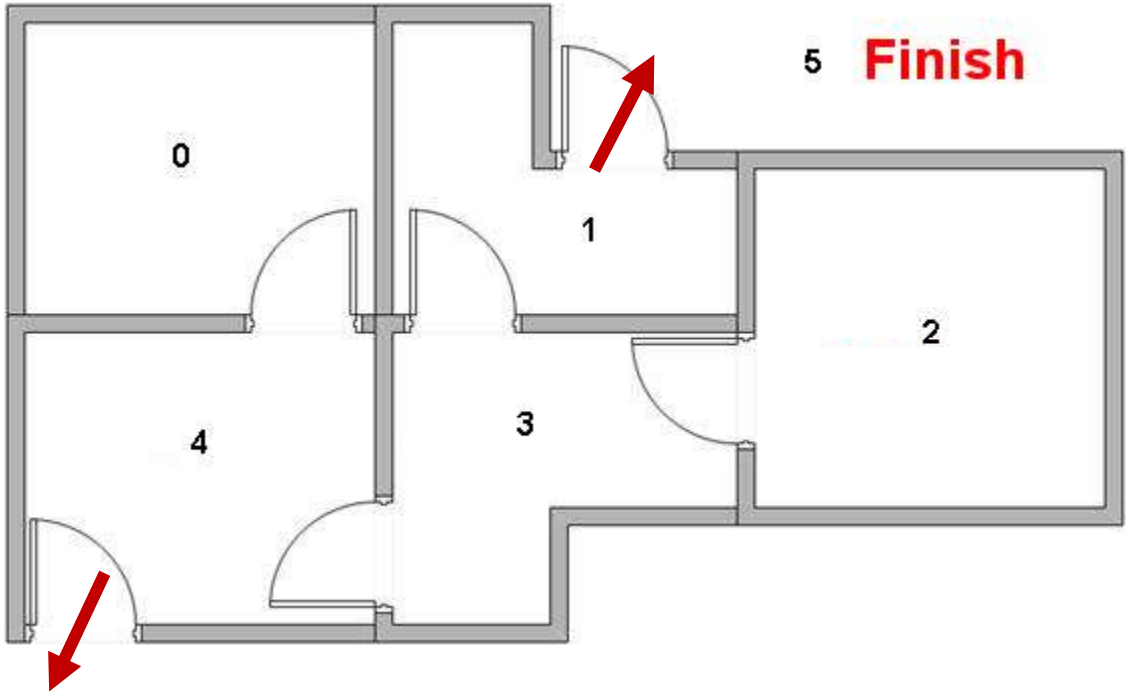
Q-learning example

Go to (action):

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Position (situation):

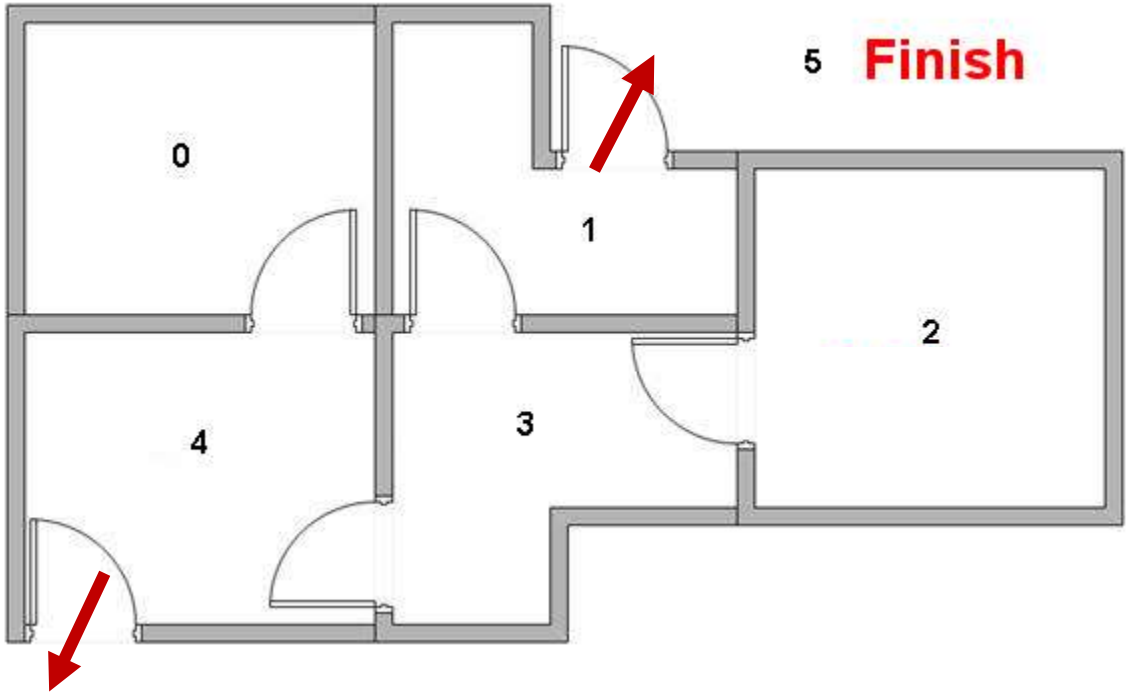
- **Reward:** 100 points for getting outside, else reward = 0
- $\gamma = 0.8$ (discount)
- Randomly start in position 3, then randomly (or guided) goes to position 1, then position 5.



Q-learning example

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

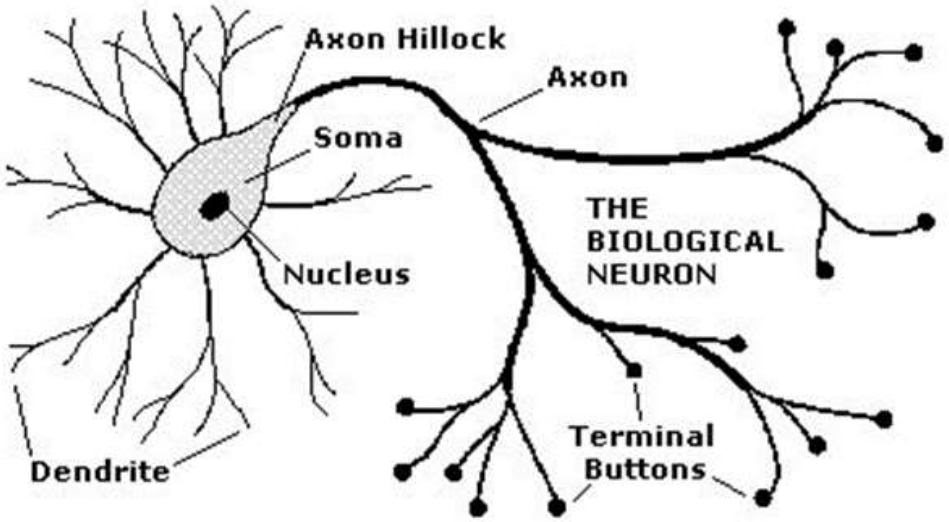
- Over multiple training trials, we add the Q's (and usually normalize them)
- Using the Q table (which has been learned through training), the machine can decide which action is best to take in a given situation.



The problem with the Q table

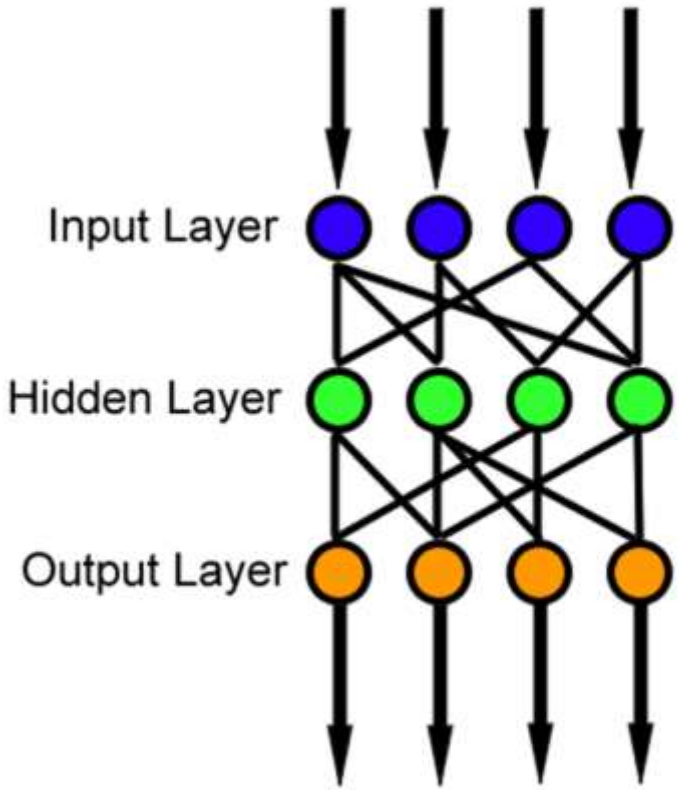
Problems for which machine learning would be useful are not at all this simple! Machine learning is most useful in scenarios which have a myriad of states and state dimensions (called the state space).

Solution: neural networks!



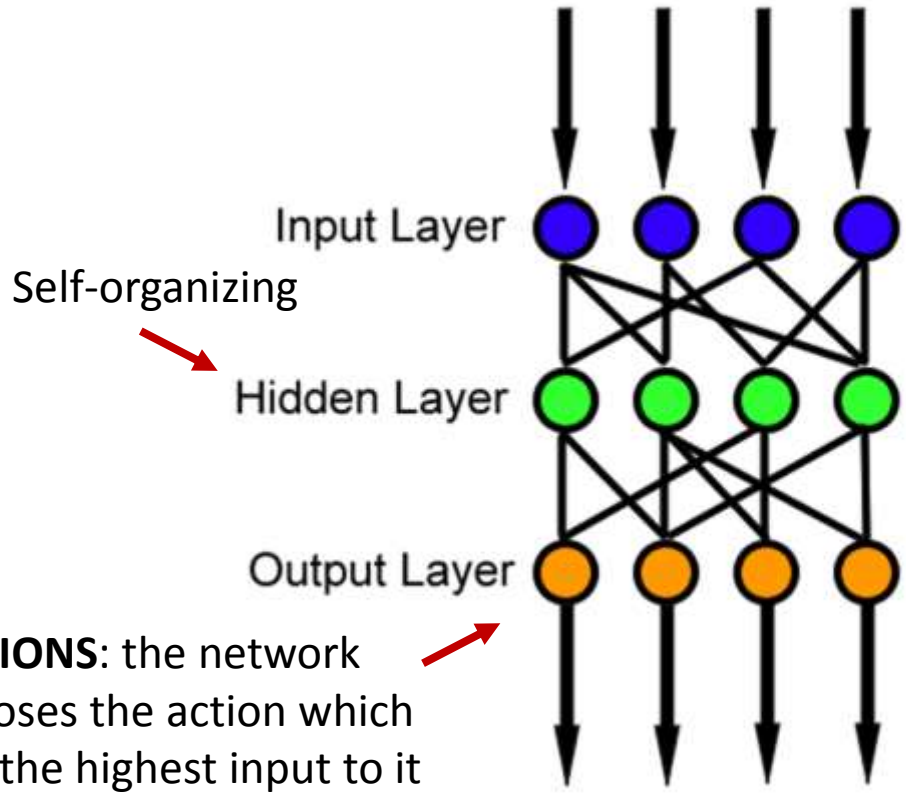
- Neurons are very complex microprocessors
- Have 3 important traits:
- **Synaptic weights**
  - When a neuron fires, it sends a propagates a signal (a potential) with a specific voltage
- **Cumulative stimulus**
  - A neuron may receive input from several other neurons. The net input is summed at the axon hillock
- **All or none**
  - If the net input exceeds a threshold, the neuron fires at it's specific weight. Otherwise, it gives no output.

Artificial neural networks!



- Imitate a network of neurons
- Instead have interconnected **nodes**, arranged in layers. These use the same 3 traits:
- **Synaptic weights**
  - When a node receives input >= threshold, it passes a specific value (based on **weight of connection**) to all of the nodes it is connected to downstream
- **Cumulative stimulus**
- **All or none**

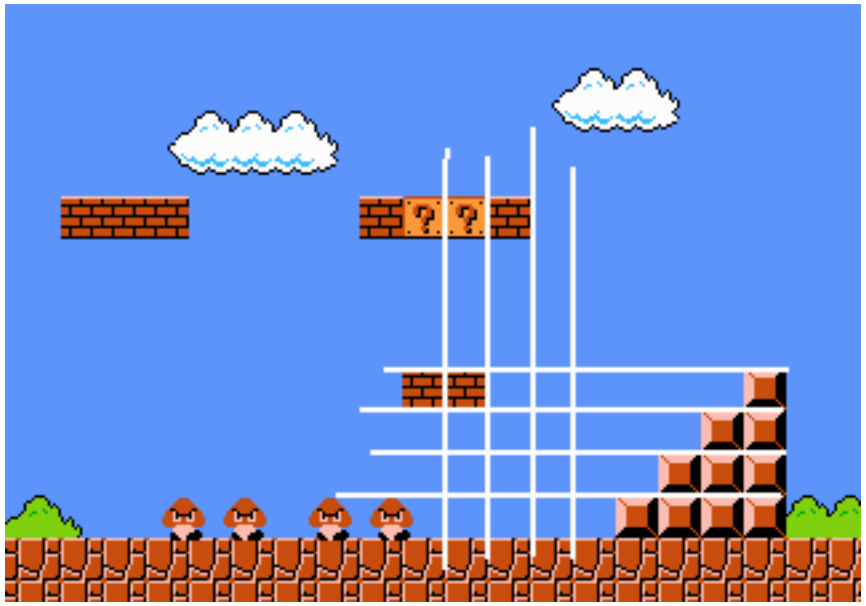
The beauty of neural networks



ACTIONS: the network chooses the action which has the highest input to it from a given state.

- Can be used to fit to a function (**such as our Q function**!)
- Instead of having a unique value for each possible state, it instead takes in values for each dimension of a state, representing the entire statespace in fewer values
  - In the room example, the neural network would need 1 input node, taking 1 of 5 possible values (for each room).
- Our Q values are not discretely represented in the network. Instead, our Q-learning procedure adjusts the weights between nodes which were fired to produce an action, based on the action's outcome.

Now how do we get it to play Mario?

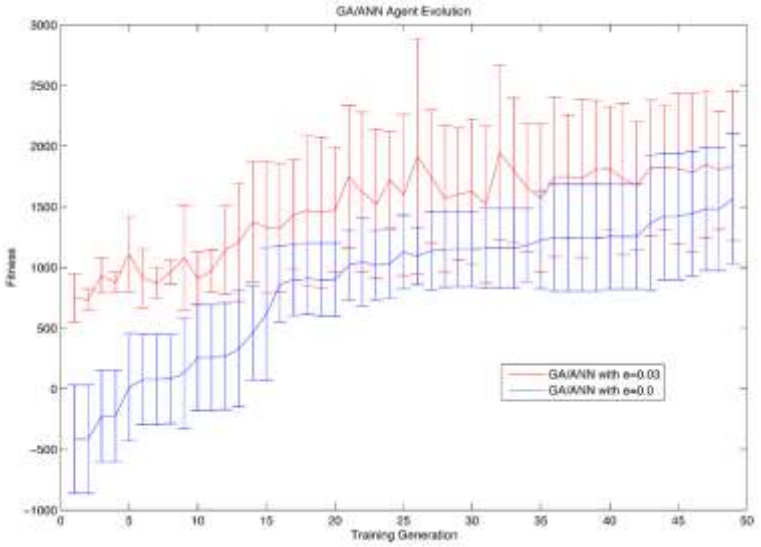


- Define **reward values** (i.e. win = 200, killEnemy = 20, collectCoin = 5, death = -1000). This will be used to adjust connection weights.
- **Divide the screen space** into a grid. Each grid space feeds to an input node with a value that represents what is in the space.
- Include **input nodes for other dimensions** of state: Mario's velocity, whether or not he has a power up, etc.
- Include **output nodes** for each of the possible **controls** Mario has.
- Decide how many processing layers and how many nodes in each layer will connect state (input) nodes to action (output) nodes.
- Initialize 100 or so networks with this structure, with randomized weights (Gaussian SD = 0.5)

Now how do we get it to play Mario?

- Allow each network to experience many testing cases.
- At first each network will act randomly and ignorantly. However after many trials, Q-learning adjusts weights to produce more effective behaviors.
- However, we cannot expect initial randomness alone to tend towards optimal behaviors through Q-learning.
- Use **genetic model** as well!
- Also use ε – greedy behaviors
- Use randomly-generated levels only!

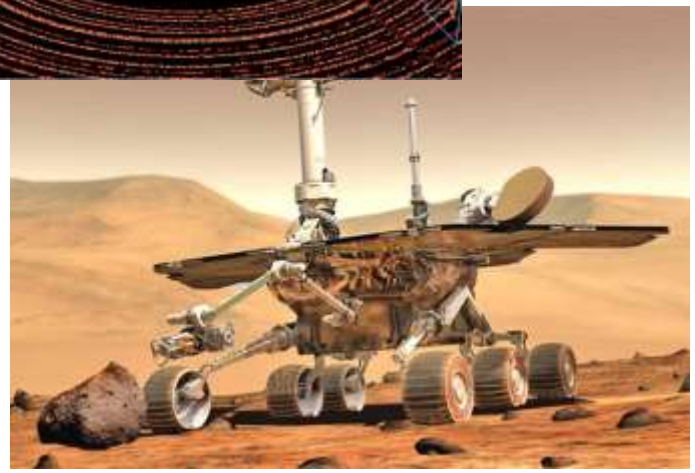
How well does it perform?



Work by Mullen, J. and Southerland, J. show a substantial increase in performance over generations

- This basic model has been implemented by many enthusiasts and researchers, using variations on these parameters.
- In general, these networks outperform basic algorithms, and can reliably complete randomly generated levels.
- In some cases, the network is able to find previously unknown exploits.
- However, performance vs. humans not a reliable comparison due to ceiling effects: both succeed at similar rates and quality of successful performance is subjective.

Why does it even matter?



- If we can develop reliable learners in complex scenarios, we can have them learn to perform menial tasks that would nonetheless demand human attention.
  - Autonomous vehicles
  - Unsupervised exploration
  - Language emulation and processing
  - Image processing, object recognition
- Applicable to almost any field or function!
- One problem: neural networks are a “black box.” They contain no transparent representation of their function, and so neural networks (as of yet) cannot be used to generate new knowledge about mathematical relations in the world.

References

McCullock, John. "Q-Learning .*;." *A Painless Q-Learning Tutorial*. N.p., 2012. Web. 20 May 2015. <http://mnemstudio.org/path-finding-q-learning-tutorial.htm>.

Mullen, Jonathan, and Joshua Southerland. "Machine Learning Agents for Playing Super Mario Bros."*SpringerReference* (2011): n. pag. University of Oklahoma. Web. <http://www.cs.ou.edu/~amy/courses/cs5033_fall2009/Mullen_Southerland.pdf>.

Riedmiller, Martin. "Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method." *CiteSeer*. University of Onsabruck, n.d. Web. <http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.72.1193%26rep%3Drep1%26type%3Dpdf>.

Smith, Steven W. "Neural Network Architecture." *The Scientist and Engineer's Guide ToDigital Signal ProcessingBy Steven W. Smith, Ph.D.* California Technical Publishing, 2011. Web. 20 May 2015. <http://www.dspguide.com/ch26/2.htm>.

"12. Learning: Neural Nets, Back Propagation." *YouTube*. MITOpenCourse, 10 Jan. 2014. Web. 20 May 2015. <https://www.youtube.com/watch?v=q0pm3BrIUFo>.