Innovations in deep learning

Deep learning (neural networks) is the core idea driving the current revolution in AI.

Notes:

• Checkers is the last solved game (from game theory, where perfect player outcomes can be fully predicted from any gameboard).
  https://en.wikipedia.org/wiki/Solved_game

• The first machine learning algorithm defeated a world champion in Chess in 1996.
Computers making art

The Next Rembrandt

A Neural Algorithm of Artistic Style
https://arxiv.org/abs/1508.06576
https://github.com/jcjohnson/neural-style

Google Deep Dream
https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
Detecting skin cancer

Lets Start Training

https://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.c
Review
## Classification Task

<table>
<thead>
<tr>
<th>ROI 1</th>
<th>ROI 2</th>
<th>ROI $m$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Heart 1 ROI 1" /></td>
<td><img src="image2.png" alt="Heart 1 ROI 2" /></td>
<td><img src="image3.png" alt="Heart 1 ROI m" /></td>
<td>0</td>
</tr>
<tr>
<td><img src="image1.png" alt="Heart 2 ROI 1" /></td>
<td><img src="image2.png" alt="Heart 2 ROI 2" /></td>
<td><img src="image3.png" alt="Heart 2 ROI m" /></td>
<td>0</td>
</tr>
<tr>
<td><img src="image1.png" alt="Heart n ROI 1" /></td>
<td><img src="image2.png" alt="Heart n ROI 2" /></td>
<td><img src="image3.png" alt="Heart n ROI m" /></td>
<td>1</td>
</tr>
</tbody>
</table>

...
Machine Learning

$\mathbf{X}$
(inputs)

$\theta$
(model)

$\hat{y}$
(prediction)
The Training / Testing Paradigm

Dataset

Deployment
The Training / Testing Paradigm

If your model passes testing...

Training

Learn your parameters

Testing

Make sure that they work

Deployment
Logistic Regression

\[ P(Y = 1 | x) = \sigma\left( \sum \theta_i x_i \right) \]
A Journey From Pure Math into the beyond
Logistic Regression is like the Harry Potter Sorting Hat

\[
P(Y = 1) = 0.91
\]
Logistic Regression is like the Harry Potter Sorting Hat

\[ \sigma(\theta^T x) \]
Logistic Regression

\[ P(Y = 1|X = x) = \sigma(\theta^T x) \]
Logistic Regression

\[ P(Y = 1 | X = x) = \sigma(\theta^T x) \]
Logistic Regression

\[ P(Y = 1 | X = x) = \sigma(\theta^T x) \]
Logistic Regression

\[ P(Y = 1 | X = x) = \sigma(\theta^T x) \]
Logistic Regression

\[ P(Y = 1 | X = x) = \sigma(\theta^T x) \]

\[ P(Y = 1 | X) = 0.817 \]

\[ \sigma(z) = \frac{1}{1 + e^{-z}} \]
Logistic Regression

\[ P(Y = 1 | x) = \sigma(\theta^T x) \]
1. Make logistic regression assumption

\[ P(Y = 1 | X = x) = \sigma(\theta^T x) \]
\[ P(Y = 0 | X = x) = 1 - \sigma(\theta^T x) \]

2. Calculate the log likelihood for all data

\[ LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \sigma(\theta^T x^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T x^{(i)})] \]

3. Get derivative of log likelihood with respect to thetas

\[ \frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[ y^{(i)} - \sigma(\theta^T x^{(i)}) \right] x_j^{(i)} \]
Logistic Regression Training

Initialize: $\theta_j = 0$ for all $0 \leq j \leq m$

Repeat many times:

$\text{gradient}[j] = 0$ for all $0 \leq j \leq m$

For each parameter $j$

For each training example $(x, y)$:

$\text{gradient}[j] \leftarrow x_j \left( y - \frac{1}{1 + e^{-\theta^Tx}} \right)$

$\theta_j \leftarrow \eta \times \text{gradient}[j]$ for all $0 \leq j \leq m$
Training

Dataset likelihood:

Likelihood vs. Training iterations

\( x_0 \), \( \theta_0 \)
\( x_1 \), \( \theta_1 \)
\( x_2 \), \( \theta_2 \)
\( x_3 \), \( \theta_3 \)
Artificial Neurons
Biological Basis for Neural Networks

A neuron

Your brain

Actually, it’s probably someone else’s brain
Core idea behind the revolution in AI
Alpha GO
Computers Making Art
Deep learning is (at its core) many logistic regression pieces stacked on top of each other. (aka Neural Networks)
Digit Recognition Example

Let’s make feature vectors from pictures of numbers

\[ x^{(i)} = [0, 0, 0, 0, \ldots, 1, 0, 0, 1, \ldots 0, 0, 1, 0] \]

\[ y^{(i)} = 0 \]

\[ x^{(i)} = [0, 0, 1, 1, \ldots, 0, 1, 1, 0, \ldots 0, 1, 0, 0] \]

\[ y^{(i)} = 1 \]
Hundreds of millions of neurons [1]
Visual neurons make up up 30% of your cortex [1]

Logistic Regression

This means it predicts a 0
Logistic Regression

This means it predicts a 0

Indicates logistic regression connection

This means it predicts a 0
Logistic Regression

This means it predicts a 1
Not So Good

This means it predicts a 1
We Can Put Neurons Together

This means it predicts a 0
We Can Put Neurons Together

Look at a single “hidden” neuron

There is a parameter for every connection

This means it predicts a 0
We Can Put Neurons Together

Look at another “hidden” neuron

There is a parameter for every connection

This means it predicts a 0
We Can Put Neurons Together

This means it predicts a 0
We Can Put Neurons Together

There is a parameter for every connection

This means it predicts a 0

Look at another neuron
We Can Put Neurons Together

This means it predicts a 0
We Can Put Neurons Together
We Can Put Neurons Together
We Can Put Neurons Together

*lots*
We Can Put Neurons Together

*lots* logistic regression
Deep learning

def **Deep learning** is maximum likelihood estimation with neural networks.

def **A neural network** is (at its core) many logistic regression pieces stacked on top of each other.

\[ [1, 0, \ldots, 1] \]
\[ x, \text{ input} \]
\[ \text{LOL} \]
\[ \text{Lots of Logistic (regressions)} \]
\[ \hat{y}, \text{ output} \]
\[ P(Y|X = x) \]
\[ > 0.5? \]
Yes. Predict 1
Demonstration

Deep learning gets its *intelligence* from its thetas (aka its parameters)
How do we train?
MLE of Thetas!
First: Learning Goals…
1. Understand Chain Rule as ♡ of Deep Learning
2. Demystify:
Deep Learning is MLE
3. Become experts of logistic regression
Math worth knowing:
New Notation

\[ \hat{y} = m h \sum_{i=0}^{X} b_i \bigtriangleup (\hat{y}) \]

\[ \hat{y} = m x \sum_{i=0}^{x} m h \bigtriangleup (h)_{i,j} \bigtriangleup (\hat{y}) \]
New Notation

Layer \( x \) \quad Layer \( h \) \quad Layer \( \hat{y} \)

\[ \hat{y} = m_h x \sum_{i=0}^{m_x} x_i \theta_{i,j}^{(h)} \]
\[ \hat{y} = \sigma \left( \sum_{j=0}^{m_n} h_j \theta_{j}^{(\hat{y})} \right) \]
Forward Pass

Layer $x$  Layer $h$  Layer $\hat{y}$
Forward Pass

Layer $x$  Layer $h$  Layer $\hat{y}$

$\hat{y}^i = m_h \sum_{i=0}^x b_i \cdot (\hat{y}^i)$
Forward Pass

Layer $\mathbf{x}$  Layer $\mathbf{h}$  Layer $\hat{\mathbf{y}}$

1

$\begin{align*}
\mathbf{h}_j &= \sigma \left( \sum_{i=0}^{m_x} \mathbf{x}_i \theta^{(h)}_{i,j} \right)
\end{align*}$
Forward Pass

Layer $\mathbf{x}$ \quad Layer $\mathbf{h}$ \quad Layer $\hat{\mathbf{y}}$

1

\[
LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]
\]

\[
\hat{y} = \sigma \left( \sum_{j=0}^{m_h} h_j \theta_j^{(\hat{y})} \right)
\]

\[
h_j = \sigma \left( \sum_{i=0}^{m_x} x_i \theta_i^{(h)} \right)
\]
All Together

\[ \hat{y} = m_h \sum_{i=0}^{m} h_{i,j} \theta^{(h)} \theta^{(\hat{y})} x_i \]

Neural Network

\[ x \rightarrow h \rightarrow \hat{y} \]
How many parameters in the last layer ?

a) 2  
b) 20  
c) 40  
d) 800
How many parameters in the second layer?

a) 2
b) 20
c) 40
d) 800
How many parameters in total?

a) 800  

b) 20  
c) 820  
d) 16000
Today: Do Something Brave
$$\hat{y} = m h x_i = 0 b_i \cdot (\hat{y})_i = m h x_i = 0 m x_i \cdot (h)_{i,j} \cdot (\hat{y})_i$$

20 parameters need setting

800 parameters need setting
Only Have to Do Three Things

1. Make deep learning assumption

2. Calculate the log probability for all data

3. Get partial derivative of log likelihood with respect to each theta
3. Get partial derivative of log likelihood with respect to each theta

Why?
A deep learning model gets its **intelligence** by having **useful thetas**.

We can find **useful thetas**, by searching for ones that **maximize likelihood** of our training data.

We can **maximize likelihood** using **optimization techniques** (such as gradient ascent).

In order to use **optimization techniques**, we need to calculate the **partial derivative** of likelihood with respect to thetas.
Okay gang, let's see what deep learning really is.

Thanks to Keith Eicher
Only Have to Do Three Things

1. Make deep learning assumption

\[ P(Y = 1|X = x) = \hat{y} \]
\[ P(Y = 0|X = x) = 1 - \hat{y} \]

2. Calculate the log probability for all data
Same Assumption, Same LL

\[
P(Y = 1|X = x) = \hat{y} \quad \hat{y} = \sigma \left( \sum_{j=0}^{m_h} h_j \theta_j^{(y)} \right) \quad h_j = \sigma \left( \sum_{i=0}^{m_x} x_i \theta_{i,j}^{(h)} \right)
\]

For one datum

\[
P(Y = y|X = x) = (\hat{y})^y(1 - \hat{y})^{1-y}
\]

Feel the Bern!

\[Y \sim \text{Bern}(\hat{y})\]

For IID data

\[
L(\theta) = \prod_{i=1}^{n} P(Y = y^{(i)}|X = x^{(i)})
\]

\[
= \prod_{i=1}^{n} (\hat{y}^{(i)})^{y^{(i)}} \cdot \left[1 - (\hat{y}^{(i)})\right]^{(1-y^{(i)})}
\]

Take the log

\[
LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log[1 - \hat{y}^{(i)}]
\]
Only Have to Do Three Things

1. Make deep learning assumption

\[ P(Y = 1|X = x) = \hat{y} \]
\[ P(Y = 0|X = x) = 1 - \hat{y} \]

2. Calculate the log probability for all data

\[ LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log[1 - \hat{y}^{(i)}] \]

3. Get partial derivative of log likelihood with respect to each theta

\[ \hat{y} = \sigma \left( \sum_{j=0}^{m_h} h_j \theta_j^{(\hat{y})} \right) \]
Derivative Goals

Loss with respect to output layer params

\[
\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}
\]

Loss with respect to hidden layer params

\[
\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}
\]

Neural Network

\[
\begin{align*}
x & \mapsto h \\
\theta^{(h)} & \\
\hat{y} & \mapsto \hat{y} \\
\theta^{(\hat{y})} & 
\end{align*}
\]
Bad Approach

\[ LL(\theta) = y \log \hat{y} + (1 - y) \log [1 - \hat{y}] \]

\[ \hat{y} = \sigma \left( \sum_{i=0}^{m_h} h_i \theta_i^{(\hat{y})} \right) \]

\[ = \sigma \left( \sum_{i=0}^{m_h} \left[ \sigma \left( \sum_{i=0}^{m_x} x_i \theta_{i,j}^{(h)} \right) \right] \theta_i^{(\hat{y})} \right) \]
Derivatives Without Tears
Woah Mrs Nicholson, you were right. Chain rule is useful!

$$\frac{\partial f(z)}{\partial x} = \frac{\partial f(z)}{\partial z} \cdot \frac{\partial z}{\partial x}$$

First use:

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$


True fact about sigmoid functions

\[
\frac{\partial}{\partial z} \sigma(z) = \sigma(z)[1 - \sigma(z)]
\]
Big Idea #3: Derivative of Sum

\[ LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log[1 - \hat{y}^{(i)}] \]

We only need to calculate the gradient for one training example!

\[ \frac{\partial}{\partial x} \sum f(x) = \sum \frac{\partial}{\partial x} f(x) \]

We will pretend we only have one example

\[ LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}] \]

We can sum up the gradients of each example to get the correct answer
Recall
Sigmoid has a Beautiful Slope

\[
\frac{\partial}{\partial \theta_j} \sigma(\theta^T x) = \sigma(z)[1 - \sigma(z)]
\]

where \( z = \theta^T x \)

\[
\frac{\partial}{\partial \theta_j} \sigma(\theta^T x) = \frac{\partial}{\partial z} \sigma(z) \cdot \frac{\partial z}{\partial \theta_j}
\]

Chain rule!

\[
\frac{\partial}{\partial \theta_j} \sigma(\theta^T x) = \sigma(\theta^T x)[1 - \sigma(\theta^T x)] x_j
\]

Plug and chug

Sigmoid, you should be a ski hill
This is Sparta!!!!!

Stanford
Derivative Goals

Loss with respect to output layer params

\[
\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}
\]

Loss with respect to hidden layer params

\[
\frac{\partial LL(\theta)}{\partial \theta_i^{(h)}}
\]

Neural Network

\[
x \xrightarrow{\theta^{(h)}} h \xrightarrow{\theta^{(\hat{y})}} \hat{y}
\]
\[ \frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} \]

**Neural Network**

\[ \hat{y} = \theta^{(\hat{y})} \]
\[ \theta^{(h)} \]
\[ x \]

**Goal**

\[ LL \]

**Decomposition**

\[ \frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}} \]
\[
\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}
\]

Chain Rule Example 2

Goal

Network

Decomposition
Decomposition
\[
\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}
\]
Gradient of output layer params

\[
\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \left[ \frac{\partial LL}{\partial \hat{y}} \right] \left[ \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}} \right]
\]

\[LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]\]

\[
\frac{\partial LL(\theta)}{\partial \hat{y}} = \frac{y}{\hat{y}} + \frac{(1 - y)}{(1 - \hat{y})} \cdot \frac{\partial(1 - \hat{y})}{\partial \hat{y}}
\]

\[
\frac{\partial LL(\theta)}{\partial \hat{y}} = \frac{y}{\hat{y}} - \frac{(1 - y)}{(1 - \hat{y})}
\]
Gradient of output layer params

\[
\frac{\partial LL(\theta)}{\partial \theta_i} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i}
\]

\[
\hat{y} = \sigma \left( \sum_{j=0}^{m_h} h_j \theta_j^{(\hat{y})} \right) = \sigma(z) \quad \text{where} \quad z = \sum_{j=0}^{m_h} h_j \theta_j^{(\hat{y})}
\]

\[
\frac{\partial \hat{y}}{\partial \theta_i} = \hat{y}[1 - \hat{y}] \cdot \frac{\partial}{\partial \theta_i} \sum_{j=0}^{m_h} h_j \theta_j^{(\hat{y})}
\]

\[
= \hat{y}[1 - \hat{y}] \cdot h_i
\]

What! That’s not scary!
Make it Simple

\[
\frac{\partial LL(\theta)}{\partial \theta_i^{(y)}} = \frac{y}{\hat{y}} - \frac{(1 - y)}{(1 - \hat{y})} = \hat{y}[1 - \hat{y}] \cdot h_i
\]
Boom!
\[
\frac{\partial L L(\theta)}{\partial \theta_{i,j}^{(h)}}
\]
Gradient of hidden layer params

\[
\frac{\partial LL(\theta)}{\partial \theta^{(h)}_{i,j}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_j} \cdot \frac{\partial h_j}{\partial \theta^{(h)}_{i,j}}
\]
Gradient of hidden layer params

\[
\frac{\partial LL(\theta)}{\partial \theta^{(h)}_{i,j}} = \left( \frac{\partial LL}{\partial \hat{y}} \right) \cdot \left( \frac{\partial \hat{y}}{\partial h_j} \right) \cdot \left( \frac{\partial h_j}{\partial \theta^{(h)}_{i,j}} \right)
\]

\[
\hat{y} = \sigma \left( \sum_{i=0}^{m_h} h_i \theta^{(\hat{y})}_i \right)
\]

\[
\frac{\partial \hat{y}}{\partial h_j} = \hat{y}[1 - \hat{y}]\theta^{(\hat{y})}_j
\]

Wait is it over?
Gradient of hidden layer params

\[ \frac{\partial LL(\theta)}{\partial \theta^{(h)}_{i,j}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_j} \cdot \frac{\partial h_j}{\partial \theta^{(h)}_{i,j}} \]

\[ h_j = \sigma \left( \sum_{k=0}^{m_x} x_k \theta_{k,j} \right) \]

\[ \frac{\partial h_j}{\partial \theta^{(h)}_{i,j}} = h_j [1 - h_j] x_i \]

That one too?
\[
\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \begin{pmatrix}
\hat{y} & (1 - \hat{y}) \\
\end{pmatrix}
\]

\[
= \frac{y}{\hat{y}} - \frac{(1 - y)}{(1 - \hat{y})}
\]

\[
= \hat{y}[1 - \hat{y}] \theta_{j}^{(\hat{y})}
\]

\[
= h_{j}[1 - h_{j}] x_{j}
\]
Congrats. You now know Backpropagation
Moment of silence
Loss with respect to output layer params

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Loss with respect to hidden layer params

$$\frac{\partial LL(\theta)}{\partial \theta^{(h)}_{i,j}}$$
What Would You Do Here?

Bigger Neural Network

\[ \hat{y} = m \sum_{i=0}^{h} x_i \theta_i \]

\[ \hat{y} = m \sum_{i=0}^{h} x_i \theta_i \]

\[ \hat{y} = m \sum_{i=0}^{h} x_i \theta_i \]

What Would You Do Here?
Chain rule: Game changer for artificial intelligence
Some data sets/functions are not separable

These are classifiers learned by neural networks

http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html
Some Extra Ideas!
Softmax is a generalization of the sigmoid function that squashes a K-dimensional vector \( z \) of arbitrary real values to a K-dimensional vector \( \text{softmax}(z) \) of real values in the range \([0, 1]\) that add up to 1.

\[
P(Y = j | X = x) = \text{softmax}(f(x))_j
\]
**Convolution** it turns out if you want to force some of your weights to be shared for different neurons, the math isn’t that much harder. This is used a lot for vision (CNN).
Works for any number of layers
1 Trillion Artificial Neurons
GoogLeNet Brain

22 layers deep

Multiple, Multi class output
The Cat Neuron

Top stimuli from the test set

Optimal stimulus by numerical optimization

Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012
Hire the smartest people in the world

Invent cat detector
Best Neuron Stimuli

Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012
Best Neuron Stimuli

Neuron 6

Neuron 7

Neuron 8

Neuron 9

Le, et al., Building high-level features using large-scale unsupervised learning. ICML 2012
Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012

**Best Neuron Stimuli**

- Neuron 10
- Neuron 11
- Neuron 12
- Neuron 13
ImageNet Classification

22,000 categories

14,000,000 images

Hand-engineered features (SIFT, HOG, LBP), Spatial pyramid, SparseCoding/Compression

Le, et al., Building high-level features using large-scale unsupervised learning. ICML 2012
smoothhound, smoothhound shark, Mustelus mustelus
American smooth dogfish, Mustelus canis
Florida smoothhound, Mustelus norrisi
whitetip shark, reef whitetip shark, Triaenodon obesus
Atlantic spiny dogfish, Squalus acanthias
Pacific spiny dogfish, Squalus suckleyi
hammerhead, hammerhead shark
smooth hammerhead, Sphyrna zygaena
smalleye hammerhead, Sphyrna tudes
shovelhead, bonnethead, bonnet shark, Sphyrna tiburo
angel shark, angelfish, Squatina squatina, monkfish
electric ray, crampfish, numbfish, torpedo
smalltooth sawfish, Pristis pectinatus
guitarfish
I\textbf{oughttail stingray}, Dasyatis centroura
butterfly ray
eagle ray
spotted eagle ray, spotted ray, Aetobatus narinari
cownose ray, cow-nosed ray, Rhinoptera bonasus
manta, manta ray, devilfish
\textbf{Atlantic manta}, Manta birostris
devil ray, Mobula hypostoma
grey skate, gray skate, Raja batis
little skate, Raja erinacea

\textbf{Stingray}

\textbf{Mantaray}
<table>
<thead>
<tr>
<th>Percentage</th>
<th>Method</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005%</td>
<td>Random guess</td>
<td></td>
</tr>
<tr>
<td>1.5%</td>
<td>Pre Neural Networks</td>
<td>GoogLeNet</td>
</tr>
</tbody>
</table>

Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012
0.005%  1.5%  43.9%
Random guess  Pre Neural Networks  GoogLeNet

Szegedy et al, Going Deeper With Convolutions, CVPR 2015
0.005% Random guess  1.5% Pre Neural Networks  95.1% SE-ResNet
How many parameters is too many?
Goal of machine learning: build models that **generalize** well to predicting new data

- “Overfitting”: fitting the training data too well, so we lose generality of model

- Polynomial on the right fits training data perfectly!
- Which would you rather use to predict a new data point?
Prevent Overfitting?

**Dropout** when your model is training, randomly turn off your neurons with probability 0.5. It will make your network more robust.
Not everything is classification
Deep Reinforcement Learning
Instead of having the output of a model be a probability you can make it an expectation.
Deep Reinforcement Learning

http://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html
Deep Reinforcement Learning

R is a reward and $A_i$ is a legal action.

Input is a representation of current state ($S$).

Interpret outputs as expected reward for a given action.


$\hat{y} = \theta^{(h)}(h)$

$\hat{y} = \theta^{(\hat{y})}$. 

$\theta^{(h)}$: hidden layer parameters

$\theta^{(\hat{y})}$: output layer parameters
Deep Mind Atari Games

Score compared to best human
Can A.I. Grade Your Next Test?

Neural networks could give online education a boost by providing automated feedback to students.