Deep Learning

Digging into Data

April 28, 2014

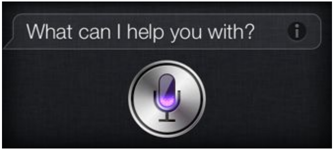
COLLEGE OF INFORMATION STUDIES

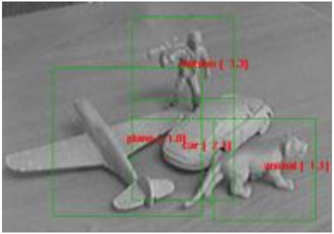
Based on material from Andrew Ng

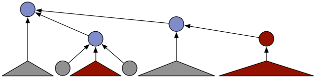
Outline

Deep Learning was once known as "Neural Networks"

But it came back . . .







- More data
- Better tricks (regularization)
- Faster computers

And companies are investing . . .

Google Hires Brains that Helped Supercharge Machine Learning

BY ROBERT MCMILLAN 03.13.13 | 6:30 AM | PERMALINK

Share 0 Tweet 1 +1 145 Share Pin it

And companies are investing . . .

'Chinese Google' Opens Artificial-Intelligence Lab in Silicon Valley

BY DANIELA HERNANDEZ 04.12.13 | 6:30 AM | PERMALINK

Share 0 Tweet 0 +1 228 Share Pin it

And companies are investing . . .

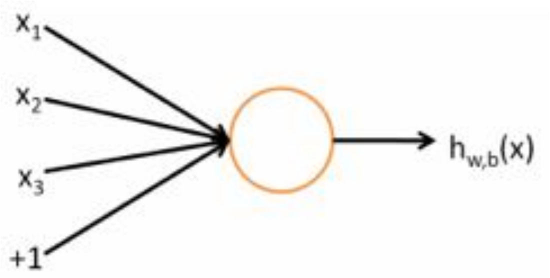
Facebook's 'Deep Learning' Guru Reveals the Future of AI

BY CADE METTZ 12.12.13 | 6:30 AM | PERMALINK

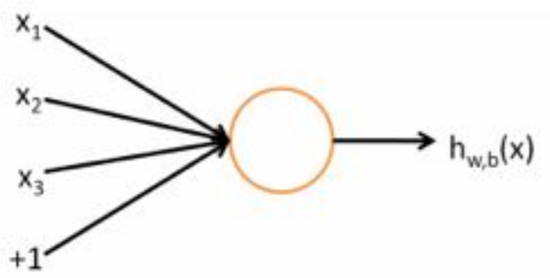
Share 2 Tweet 1 8+1 143 Share Pin it

Outline

Map inputs to output



Map inputs to output

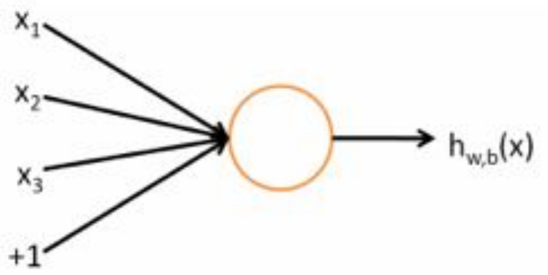


Input

Vector $x_1 \ldots x_d$

inputs encoded as real numbers

Map inputs to output



Input

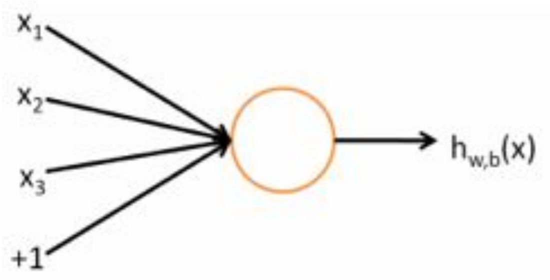
Vector $x_1 \dots x_d$

Output

$$f\left(\sum_i W_i x_i + b\right)$$

multiply inputs by weights

Map inputs to output



Input

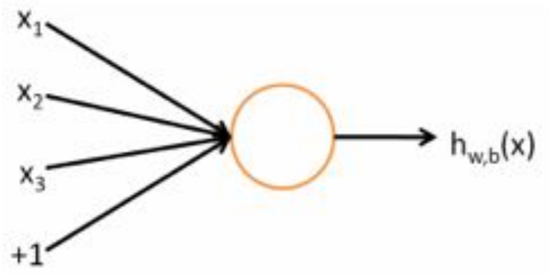
Vector $x_1 \dots x_d$

Output

$$f\left(\sum_i W_i x_i + b\right)$$

add bias

Map inputs to output



Input

Vector $x_1 \dots x_d$

Output

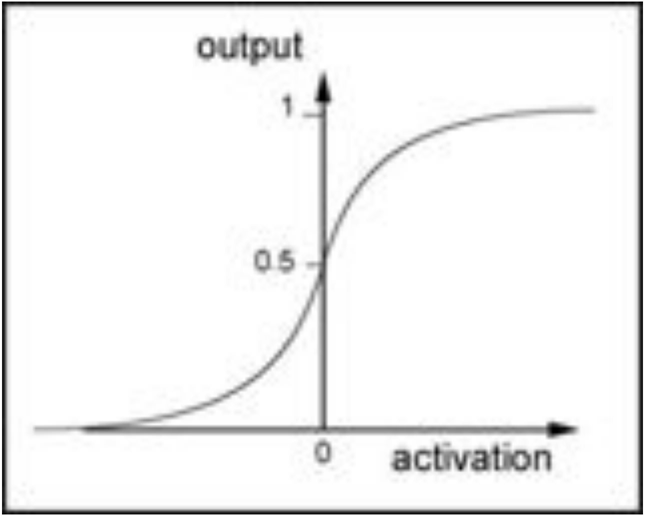
$$f\left(\sum_i W_i x_i + b\right)$$

Activation

$$f(z) \equiv \frac{1}{1 + \exp(-z)}$$

pass through nonlinear sigmoid

What's a sigmoid?

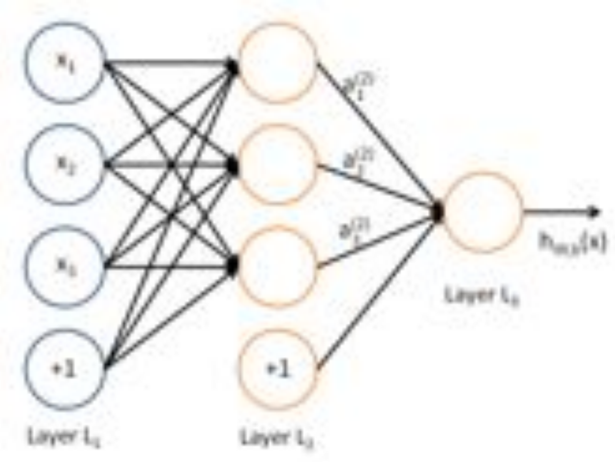


In the shallow end

- This is still logistic regression
- Engineering features x is difficult (and requires expertise)
- Can we learn how to represent inputs into final decision?

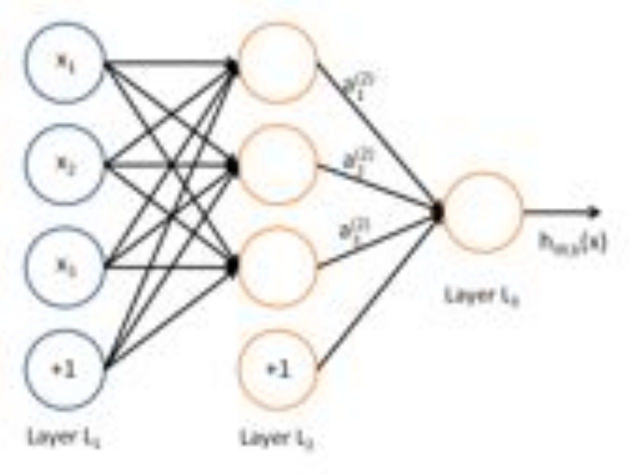
Outline

Learn the features and the function



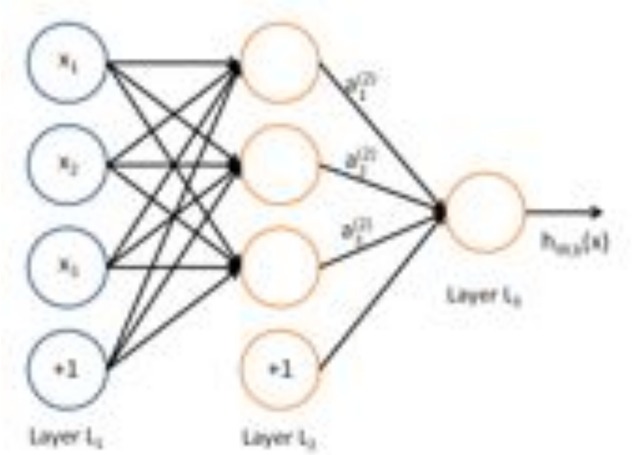
$$a_1^{(2)} = f\left(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}\right)$$

Learn the features and the function



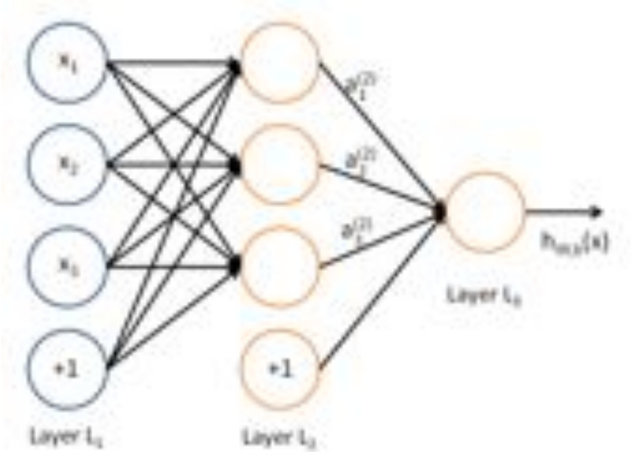
$$a_2^{(2)} = f\left(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)}\right)$$

Learn the features and the function



$$a_3^{(2)} = f\left(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)}\right)$$

Learn the features and the function



$$h_{W,b}(x) = a_1^{(3)} = f\left(W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)}\right)$$

Objective Function

- For every example x, y of our supervised training set, we want the label y to match the prediction $h_{W,b}(x)$.

$$J(W, b; x, y) \equiv \frac{1}{2}||h_{W,b}(x) - y||^2 \tag{1}$$

Objective Function

- For every example x, y of our supervised training set, we want the label y to match the prediction $h_{W,b}(x)$.

$$J(W, b; x, y) \equiv \frac{1}{2}||h_{W,b}(x) - y||^2 \tag{1}$$

- We want this value, summed over all of the examples to be as small as possible

Objective Function

- For every example x, y of our supervised training set, we want the label y to match the prediction $h_{W,b}(x)$.

$$J(W,b;x,y) \equiv \frac{1}{2}||h_{W,b}(x) - y||^2 \tag{1}$$

- We want this value, summed over all of the examples to be as small as possible
- We also want the weights not to be too large

$$\frac{\lambda}{2}\sum_{l}^{n_l - 1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{l}\right)^2 \tag{2}$$

Objective Function

- For every example x, y of our supervised training set, we want the label y to match the prediction $h_{W,b}(x)$.

$$J(W, b; x, y) \equiv \frac{1}{2} ||h_{W,b}(x) - y||^2 \tag{1}$$

- We want this value, summed over all of the examples to be as small as possible
- We also want the weights not to be too large

$$\frac{\lambda}{2} \sum_{l}^{n_l - 1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{l} \right)^2 \tag{2}$$

Objective Function

- For every example x,y of our supervised training set, we want the label y to match the prediction $h_{W,b}(x)$.

$$J(W,b;x,y) \equiv \frac{1}{2}||h_{W,b}(x) - y||^2 \quad (1)$$

- We want this value, summed over all of the examples to be as small as possible
- We also want the weights not to be too large

$$\frac{\lambda}{2} \sum_{l}^{n_l - 1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{l} \right)^2 \quad (2)$$

Sum over all layers

Objective Function

- For every example x, y of our supervised training set, we want the label y to match the prediction $h_{W,b}(x)$.

$$J(W, b; x, y) \equiv \frac{1}{2} ||h_{W,b}(x) - y||^2 \tag{1}$$

- We want this value, summed over all of the examples to be as small as possible
- We also want the weights not to be too large

$$\frac{\lambda}{2} \sum_{l}^{n_l - 1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{l} \right)^2 \tag{2}$$

Sum over all sources

Objective Function

- For every example x, y of our supervised training set, we want the label y to match the prediction $h_{W,b}(x)$.

$$J(W,b;x,y) \equiv \frac{1}{2}||h_{W,b}(x) - y||^2 \tag{1}$$

- We want this value, summed over all of the examples to be as small as possible
- We also want the weights not to be too large

$$\frac{\lambda}{2} \sum_{l}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{l} \right)^2 \tag{2}$$

Sum over all destinations

Objective Function

Putting it all together:

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}||h_{W,b}(x^{(i)}) - y^{(i)}||^2\right] + \frac{\lambda}{2}\sum_{l}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{l}\right)^2 \tag{3}$$

Objective Function

Putting it all together:

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}||h_{W,b}(x^{(i)}) - y^{(i)}||^2\right] + \frac{\lambda}{2}\sum_{l}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{l}\right)^2 \tag{3}$$

- Our goal is to minimize $J(W,b)$ as a function of W and b

Objective Function

Putting it all together:

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}||h_{W,b}(x^{(i)}) - y^{(i)}||^2\right] + \frac{\lambda}{2}\sum_{l}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{l}\right)^2 \quad (3)$$

- Our goal is to minimize $J(W,b)$ as a function of W and b
- Initialize W and b to small random value near zero

Objective Function

Putting it all together:

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}||h_{W,b}(x^{(i)}) - y^{(i)}||^2\right] + \frac{\lambda}{2}\sum_{l}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{l}\right)^2 \tag{3}$$

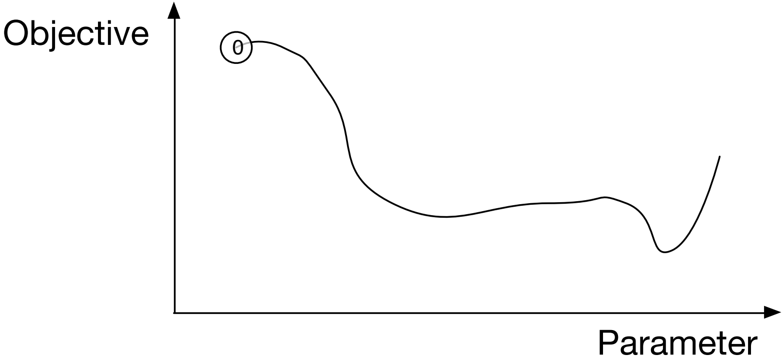
- Our goal is to minimize $J(W,b)$ as a function of W and b
- Initialize W and b to small random value near zero
- Adjust parameters to optimize J

Outline

Gradient Descent

Goal

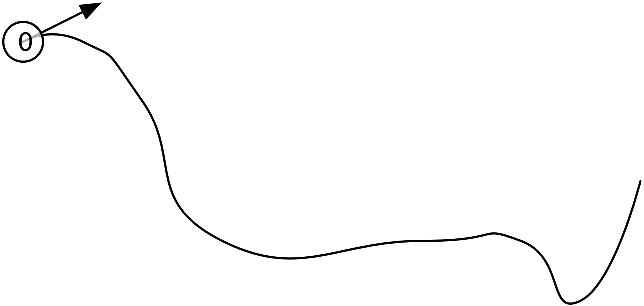
Optimize J with respect to variables W and b



Gradient Descent

Goal

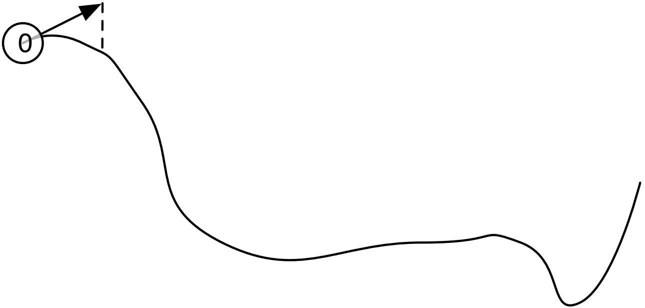
Optimize J with respect to variables W and b



Gradient Descent

Goal

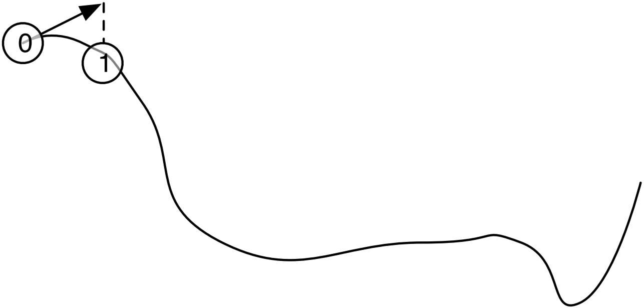
Optimize J with respect to variables W and b



Gradient Descent

Goal

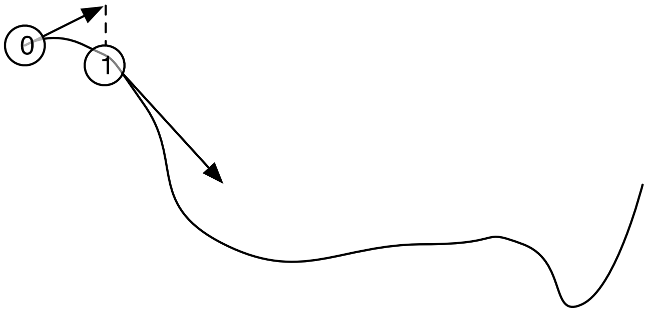
Optimize J with respect to variables W and b



Gradient Descent

Goal

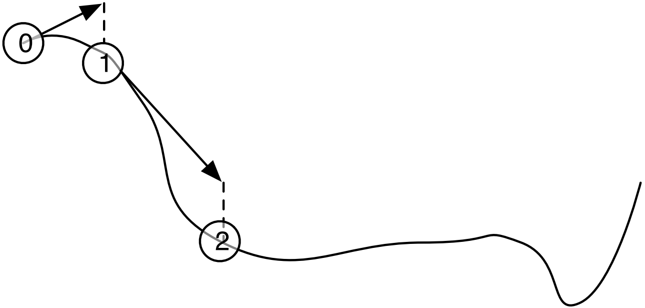
Optimize J with respect to variables W and b



Gradient Descent

Goal

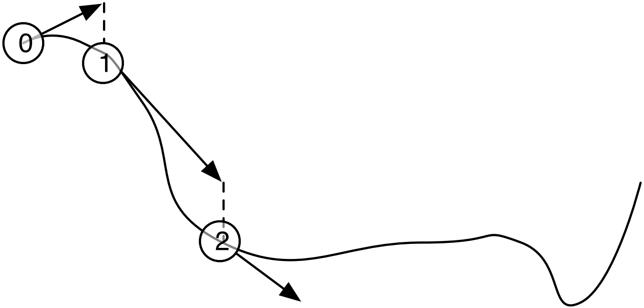
Optimize J with respect to variables W and b



Gradient Descent

Goal

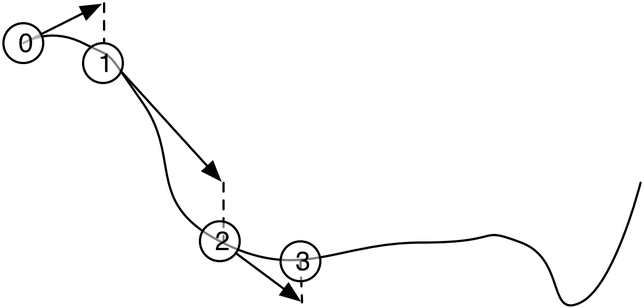
Optimize J with respect to variables W and b



Gradient Descent

Goal

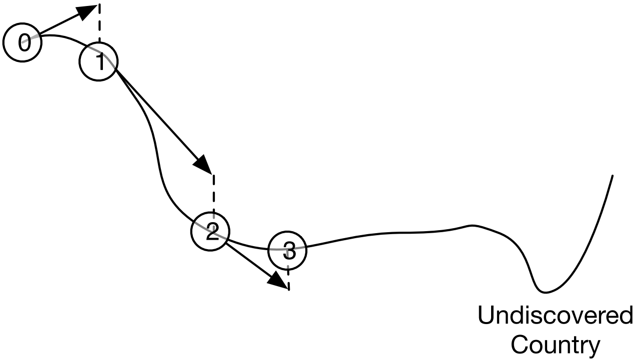
Optimize J with respect to variables W and b



Gradient Descent

Goal

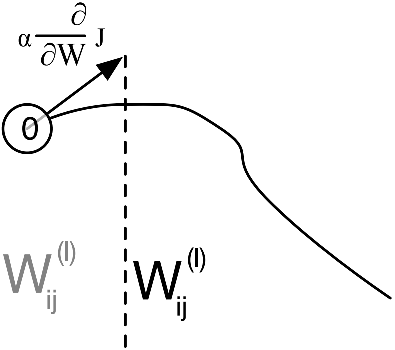
Optimize J with respect to variables W and b



Gradient Descent

Goal

Optimize J with respect to variables W and b



Backpropagation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l-1)} x_j + b_i^{(l-1)} \tag{4}$$

Backpropagation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l-1)} x_j + b_i^{(l-1)} \tag{4}$$

- The gradient is a function of a node's error $\delta_i^{(l)}$

Backpropagation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l-1)} x_j + b_i^{(l-1)} \tag{4}$$

- The gradient is a function of a node's error $\delta_i^{(l)}$
- For output nodes, the error is obvious:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} ||y - h_{w,b}(x)||^2 = -\left(y_i - a_i^{(n_l)}\right) \cdot f'\left(z_i^{(n_l)}\right) \frac{1}{2} \tag{5}$$

Backpropagation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l-1)} x_j + b_i^{(l-1)} \tag{4}$$

- The gradient is a function of a node's error $\delta_i^{(l)}$
- For output nodes, the error is obvious:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} ||y - h_{w,b}(x)||^2 = -\left(y_i - a_i^{(n_l)}\right) \cdot f'\left(z_i^{(n_l)}\right) \frac{1}{2} \tag{5}$$

- Other nodes must "backpropagate" downstream error based on connection strength

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{t+1}} W_{ji}^{(l+1)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \tag{6}$$

Backpropagation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l-1)} x_j + b_i^{(l-1)} \tag{4}$$

- The gradient is a function of a node's error $\delta_i^{(l)}$
- For output nodes, the error is obvious:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} ||y - h_{w,b}(x)||^2 = -\left(y_i - a_i^{(n_l)}\right) \cdot f'\left(z_i^{(n_l)}\right) \frac{1}{2} \tag{5}$$

- Other nodes must "backpropagate" downstream error based on connection strength

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{t+1}} W_{ji}^{(l+1)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \tag{6}$$

Backpropagation

- For convenience, write the input to sigmoid

$$z_i^{(l)} = \sum_{j=1}^{n} W_{ij}^{(l-1)} x_j + b_i^{(l-1)} \tag{4}$$

- The gradient is a function of a node's error $\delta_i^{(l)}$
- For output nodes, the error is obvious:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} ||y - h_{w,b}(x)||^2 = -\left(y_i - a_i^{(n_l)}\right) \cdot f'\left(z_i^{(n_l)}\right) \frac{1}{2} \tag{5}$$

- Other nodes must "backpropagate" downstream error based on connection strength

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{t+1}} W_{ji}^{(l+1)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \tag{6}$$

(chain rule)

Partial Derivatives

- For weights, the partial derivatives are

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)} \tag{7}$$

- For the bias terms, the partial derivatives are

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)} \tag{8}$$

- But this is just for a single example ...

Full Gradient Descent Algorithm

1. Initialize $U^{(l)}$ and $V^{(l)}$ as zero
2. For each example $i = 1 \dots m$
   1. Use backpropagation to compute $\nabla_W J$ and $\nabla_b J$
   2. Update weight shifts $U^{(l)} = U^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y)$
   3. Update bias shifts $V^{(l)} = V^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$
3. Update the parameters

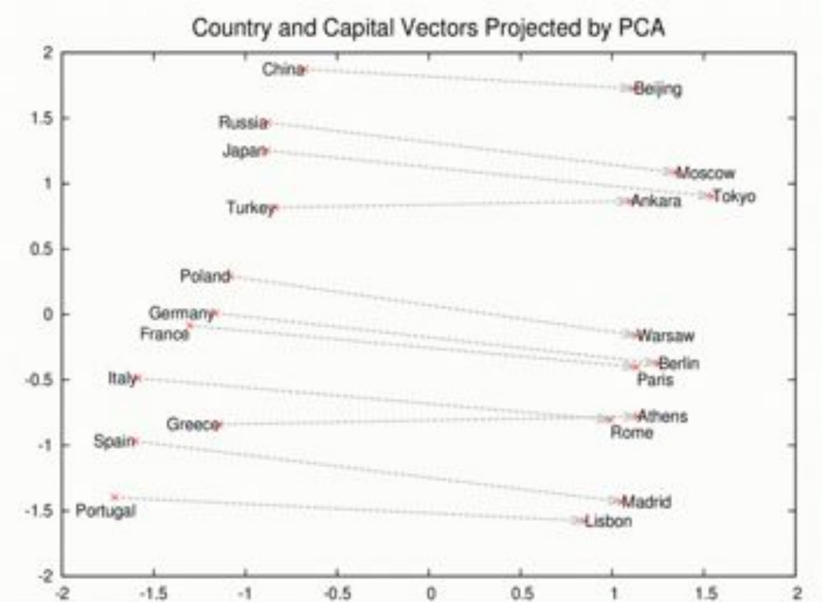
$$W^{(l)} = W^{(l)} - \alpha \left[\left(\frac{1}{m} U^{(l)} \right) \right] \tag{9}$$

$$b^{(l)} = b^{(l)} - \alpha \left[\frac{1}{m} V^{(l)} \right] \tag{10}$$

4. Repeat until weights stop changing

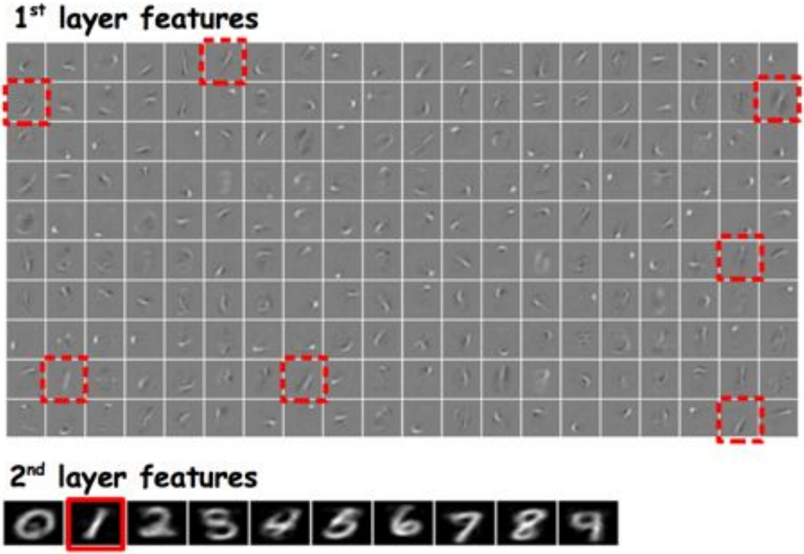
Outline

What do you learn?



What do you learn?

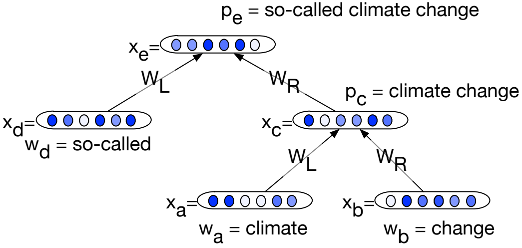
What do you learn?



Political Framing

- Death Tax
- Estate Tax
- Pro-Choice
- Pro-Life
- Entitlements
- Obamacare

Political Framing as Deep Learning



$$x_p = f(W_L \cdot x_a + W_R \cdot x_b + b_1),$$

$$\hat{y}_d = \text{softmax}(W_{cat} \cdot x_p + b_2)$$

Annotation

the Republican leadership

- (•) Neutral
- ( ) Conservative
- ( ) Liberal
- ( ) Not neutral, but I'm unsure of which direction

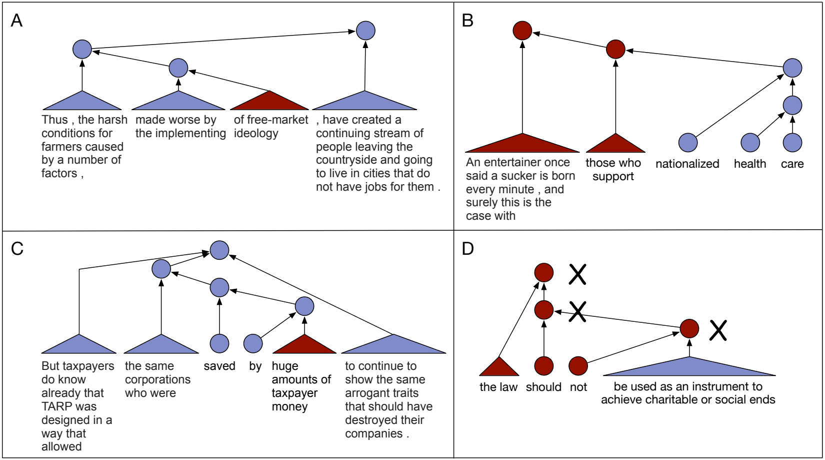
the Republican leadership making clear it wanted no piece of meaningful health care reform

- ( ) Neutral
- ( ) Conservative
- (•) Liberal
- ( ) Not neutral, but I'm unsure of which direction

But , with the Republican leadership making clear it wanted no piece of meaningful health care reform , few Republicans were interested in nego-tiating seriously .

- ( ) Neutral
- ( ) Conservative
- (•) Liberal
- ( ) Not neutral, but I'm unsure of which direction

Results


A
Thus , the harsh conditions for farmers caused by a number of factors ,
made worse by the implementing
of free-market ideology
, have created a continuing stream of people leaving the countryside and going to live in cities that do not have jobs for them .
B
An entertainer once said a sucker is born every minute , and surely this is the case with
those who support
nationalized
health
care
C
But taxpayers do know already that TARP was designed in a way that allowed
the same corporations who were
saved
by
huge amounts of taxpayer money
to continue to show the same arrogant traits that should have destroyed their companies .
D
the law
should
not
be used as an instrument to achieve charitable or social ends

Results

Model	Convote	IBC
Random	50%	50%
Bag of Words	64.7%	62.1%
Phrase Annotations	–	61.9%
Syntax	66.9%	62.6%
word2vec Regression	66.6%	63.7%
RNN	69.4%	66.2%
RNN w/ word2vec	**70.2**%	67.1%
RNN w/ word2vec and phrases	–	**69.3**%

Outline

Tricks

- **Stochastic gradient**: compute gradient from a few examples
- **Hardware**: Do matrix computations on gpus
- **Dropout**: Randomly set some inputs to zero
- **Initialization**: Using an autoencoder can help representation

Outline

- **Theano**: Python package (Yoshua Bengio)
- **Torch7**: Lua package (Yann LeCunn)
- **ConvNetJS**: Javascript package (Andrej Karpathy)
- Both automatically compute gradients and have numerical optimization
- Working group this summer at umd