Neural Networks: Representation
Non-linear hypotheses

Machine Learning
Non-linear Classification
Computer Vision: Car detection

Cars

Not a car

Testing:

What is this?
50 x 50 pixel images $\rightarrow$ 2500 pixels

$x = \begin{bmatrix}
\text{pixel 1 intensity} \\
\text{pixel 2 intensity} \\
\vdots \\
\text{pixel 2500 intensity}
\end{bmatrix}$

Quadratic features $(x_i \times x_j)$: $\approx 3$ million features
Neural Networks: Representation
Model representation
Machine Learning
Neural Networks

Origins: Algorithms that try to mimic the brain. Was very widely used in 80s and early 90s; popularity diminished in late 90s. Recent resurgence: State-of-the-art technique for many applications
Neuron in the brain

- Dendrite
- Cell body
- Node of Ranvier
- Axon
- Schwann cell
- Axon terminal
- Nucleus
- Myelin sheath
Neurons in the brain

[Credit: US National Institutes of Health, National Institute on Aging]
Neuron model: Logistic unit

\[ x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \]

\[ h_\theta(x) \]

Sigmoid (logistic) activation function.
Neural Network

\[ a^{(i)}_j = \text{“activation” of unit } i \text{ in layer } j \]

\[ \Theta^{(j)} = \text{matrix of weights controlling function mapping from layer } j \text{ to layer } j + 1 \]

\[ a^{(2)}_1 = g(\Theta^{(1)}_{10} x_0 + \Theta^{(1)}_{11} x_1 + \Theta^{(1)}_{12} x_2 + \Theta^{(1)}_{13} x_3) \]
\[ a^{(2)}_2 = g(\Theta^{(1)}_{20} x_0 + \Theta^{(1)}_{21} x_1 + \Theta^{(1)}_{22} x_2 + \Theta^{(1)}_{23} x_3) \]
\[ a^{(2)}_3 = g(\Theta^{(1)}_{30} x_0 + \Theta^{(1)}_{31} x_1 + \Theta^{(1)}_{32} x_2 + \Theta^{(1)}_{33} x_3) \]

\[ h_{\Theta}(x) = a^{(3)}_1 = g(\Theta^{(2)}_{10} a^{(2)}_0 + \Theta^{(2)}_{11} a^{(2)}_1 + \Theta^{(2)}_{12} a^{(2)}_2 + \Theta^{(2)}_{13} a^{(2)}_3) \]

If network has \( s_j \) units in layer \( j \), \( s_{j+1} \) units in layer \( j + 1 \), then \( \Theta^{(j)} \) will be of dimension \( s_{j+1} \times (s_j + 1) \).
Neural Networks: Representation Model
representation II

Machine Learning
Add $a_0^{(2)} = 1$.

$z^{(3)} = \Theta^{(2)} a^{(2)}$

$h_\Theta(x) = a^{(3)} = g(z^{(3)})$
Neural Network learning its own features
Other network architectures

\[ x_1 \rightarrow \text{Layer 1} \rightarrow \text{Layer 2} \rightarrow \text{Layer 3} \rightarrow \text{Layer 4} \rightarrow h_\Theta(x) \]
Neural Networks: Representation Examples and intuitions I
Non-linear classification example: XOR/XNOR

$x_1, x_2$ are binary (0 or 1).

\[ y = x_1 \text{ XOR } x_2 \]
\[ x_1 \text{ XNOR } x_2 \]
\[ \text{NOT } (x_1 \text{ XOR } x_2) \]
Simple example: AND

\[ x_1, x_2 \in \{0, 1\} \]
\[ y = x_1 \text{ AND } x_2 \]

\[
\begin{array}{c|c|c}
 x_1 & x_2 & h_\Theta(x) \\
 0 & 0 & 0 \\
 0 & 1 & 1 \\
 1 & 0 & 1 \\
 1 & 1 & 1 \\
\end{array}
\]
Example: OR function

\[
\begin{align*}
\text{Table: } & x_1 & x_2 & h_\Theta(x) \\
& 0 & 0 & 0 \\
& 0 & 1 & 1 \\
& 1 & 0 & 0 \\
& 1 & 1 & 1 \\
\end{align*}
\]
Neural Networks: Representation
Examples and intuitions II

Machine Learning
$x_1 \text{ AND } x_2$ \hspace{1cm} $x_1 \text{ OR } x_2$

**Negation:**

$h_\Theta(x) = g(10 - 20x_1) \hspace{1cm} (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$h_\Theta(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Putting it together: $x_1 \text{ XNOR } x_2$

\[ x_1 \text{ AND } x_2 \]

\[ (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2) \]

\[ x_1 \text{ OR } x_2 \]

\[
\begin{array}{c|cc|c}
\hline
x_1 & x_2 & a_1^{(2)} & a_2^{(2)} & h_\Theta(x) \\
\hline
0 & 0 & & & \\
0 & 1 & & & \\
1 & 0 & & & \\
1 & 1 & & & \\
\hline
\end{array}
\]
Neural Network intuition

Layer 1

Layer 2

Layer 3

Layer 4

$h_\Theta(x)$
Handwritten digit classification

[Courtesy of Yann LeCun]
Handwritten digit classification

[Courtesy of Yann LeCun]
Neural Networks: Representation
Multi-class classification
Machine Learning
Multiple output units: One-vs-all.

Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian  
when car  
when motorcycle
Multiple output units: One-vs-all.

Want \( h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \text{etc.} \)

when pedestrian  
when car  
when motorcycle

Training set: \((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\)

\( y^{(i)} \) one of \[
\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \]

pedestrian  car  motorcycle  truck
Neural Networks: Learning
Cost function

Machine Learning
Neural Network (Classification)

\[ \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\} \]

\[ L = \text{total no. of layers in network} \]

\[ s_l = \text{no. of units (not counting bias unit) in layer } l \]

Binary classification

\[ y = 0 \text{ or } 1 \]

1 output unit

Multi-class classification (K classes)

\[ y \in \mathbb{R}^K \quad \text{E.g.} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \]

pedestrian car motorcycle truck

K output units
Cost function

Logistic regression:

\[
J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2
\]

Neural network:

\[
h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}
\]

\[
J(\Theta) = -\frac{1}{m} \left[ \sum_{i=1}^{m} \sum_{k=1}^{K} y_{k}^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_{k}^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k) \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2
\]
Neural Networks: Learning

Gradient descent

Machine Learning
Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

• Start with some $\theta_0, \theta_1$

• Keep changing $\theta_0, \theta_1$ to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum
$J(\theta_0, \theta_1)$
$J(\theta_0, \theta_1)$
Gradient descent algorithm

repeat until convergence {
\[ \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1) \]
}
Neural Networks: Learning

Backpropagation algorithm

Machine Learning
Gradient computation

\[
J(\Theta) = -\frac{1}{m} \left[ \sum_{i=1}^{m} \sum_{k=1}^{K} y^{(i)}_k \log h_\theta(x^{(i)})_k + (1 - y^{(i)}_k) \log(1 - h_\theta(x^{(i)})_k) \right]
\]

\[
+ \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{s_l} \sum_{s_{l+1}} (\Theta^{(l)}_j)^2
\]

\[
\min_{\Theta} J(\Theta)
\]

Need code to compute:
- \( J(\Theta) \)
- \( \frac{\partial}{\partial \Theta^{(l)}_{ij}} J(\Theta) \)
Backpropagation algorithm

Training set \( \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\} \)

Set \( \Delta_{ij}^{(l)} = 0 \) (for all \( l, i, j \)).

For \( i = 1 \) to \( m \)

Set \( a^{(1)} = x^{(i)} \)

Perform forward propagation to compute \( a^{(l)} \) for \( l = 2, 3, \ldots, L \)

Using \( y^{(i)} \), compute \( \delta^{(L)} = a^{(L)} - y^{(i)} \)

Compute \( \delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)} \)

\[
\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_{j}^{(l)} \delta_{i}^{(l+1)}
\]

\[
D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} \quad \text{if } j \neq 0
\]

\[
D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \quad \text{if } j = 0
\]

\[
\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}
\]
Neural Networks: Learning
Random initialization

Machine Learning
Initial value of $\Theta$

For gradient descent and advanced optimization method, need initial value for $\Theta$.

```matlab
optTheta = fminunc(@costFunction, initialTheta, options)
```

Consider gradient descent
Set `initialTheta = zeros(n,1)` ?
Zero initialization

After each update, parameters corresponding to inputs going into each of two hidden units are identical.

$\Theta_{ij}^{(l)} = 0$ for all $i, j, l$. 
Random initialization: Symmetry breaking

Initialize each $\Theta^{(l)}_{ij}$ to a random value in $[-\epsilon, \epsilon]$ (i.e. $-\epsilon \leq \Theta^{(l)}_{ij} \leq \epsilon$)

E.g.

$\Theta_1 = \operatorname{rand}(10,11) \times (2 \times \text{INIT\_EPSILON}) - \text{INIT\_EPSILON};$

$\Theta_2 = \operatorname{rand}(1,11) \times (2 \times \text{INIT\_EPSILON}) - \text{INIT\_EPSILON};$
Neural Networks: Learning
Putting it together
Machine Learning
Training a neural network

Pick a network architecture (connectivity pattern between neurons)

No. of input units: Dimension of features $x^{(i)}$
No. output units: Number of classes
Reasonable default: 1 hidden layer, or if >1 hidden layer, have same no. of hidden units in every layer (usually the more the better)
Training a neural network

1. Randomly initialize weights
2. Implement forward propagation to get $h_{\Theta}(x^{(i)})$ for any $x^{(i)}$
3. Implement code to compute cost function $J(\Theta)$
4. Implement backprop to compute partial derivatives $\frac{\partial}{\partial \Theta^{(l)}_{jk}} J(\Theta)$

for $i = 1:m$

Perform forward propagation and backpropagation using example $(x^{(i)}, y^{(i)})$

(Get activations $a^{(l)}$ and delta terms $\delta^{(l)}$ for $l = 2, \ldots, L$).
Training a neural network

5. Use gradient checking to compare \( \frac{\partial}{\partial \Theta_{jk}^{(i)}} J(\Theta) \) computed using backpropagation vs. using numerical estimate of gradient of \( J(\Theta) \). Then disable gradient checking code.

6. Use gradient descent or advanced optimization method with backpropagation to try to minimize \( J(\Theta) \) as a function of parameters \( \Theta \)
Neural Networks: Learning

Backpropagation example: Autonomous driving

Machine Learning
Neural Network-Based Autonomous Driving

23 November 1992
Neural Networks: Feature Learning

Autoencoder

Machine Learning
Autoencoders (and sparsity)
Sparse autoencoders
Unsupervised Feature Learning/Self-taught learning

\[ \begin{align*}
\text{Input: } &\quad x_1, x_2, x_3, x_4, x_5, x_6, +1 \\
\text{Features: } &\quad a_1, a_2, a_3 \\
\end{align*} \]

\[ P(y = 0 \mid x) \]

\[ \text{Input (features)} \quad \text{Logistic classifier} \]

Andrew Ng
Unsupervised pre-training + Fine-tuning

\[ P(y = 0 \mid x) \]

Input \begin{align*} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ +1 \end{align*}

Features \begin{align*} a_1 \\ a_2 \\ a_3 \\ +1 \end{align*}

Logistic classifier
Neural Networks: Feature Learning

Deep Learning

Machine Learning
Unsupervised pre-training + Fine-tuning

Input Features I Output

Input (Features I) Features II Output

Input (Features II) Softmax classifier

P(y = 0 | x) P(y = 1 | x) P(y = 2 | x)