CS 66: Machine Learning

Prof. Sara Mathieson

Spring 2019
Outline for April 3

- Continue backpropagation
- Neural network architectures
- Choice of non-linearity (activation function)
- Choice of loss function
- Choice of weight initialization

- Lab 6 due Friday (check in today during lab)
- Lab 7 released Friday (fill out partner forms!)
- Alternative presentation time: May 13, 1-3pm
Turing award goes to neural network research

• LeCun, Hinton, Bengio win Turing award for their work on neural networks

Big picture for today

• Neural networks can approximate any function!
Big picture for today

• Neural networks can approximate any function!

• For our purposes in ML, we want to use them to approximate a function from our inputs to our outputs
Big picture for today

• Neural networks can approximate any function!

• For our purposes in ML, we want to use them to approximate a function from our inputs to our outputs

• We will train our network by asking it to minimize the loss between it’s output and the true output
Big picture for today

• Neural networks can approximate any function!

• For our purposes in ML, we want to use them to approximate a function from our inputs to our outputs

• We will train our network by asking it to minimize the loss between it’s output and the true output

• We will use backpropagation to minimize loss
Outline for April 3

- Continue backpropagation
- Neural network architectures
- Choice of non-linearity (activation function)
- Choice of loss function
- Choice of weight initialization

Following Stanford CNN course reading:
http://cs231n.github.io/optimization-2/
Backpropagation: Example 1

Forward pass: compute values

Example from: http://cs231n.github.io/optimization-2/
Backpropagation: Example 1

Forward pass: compute values

Example from: http://cs231n.github.io/optimization-2/
Backpropagation: Example 1

Forward pass: compute values

Example from: http://cs231n.github.io/optimization-2/
Backpropagation: Example 1

Backward pass: compute local gradients

Example from: http://cs231n.github.io/optimization-2/
Backpropagation: Example 1

Backward pass: compute local gradients

Example from: http://cs231n.github.io/optimization-2/
Backpropagation: Example 1

Now if we wanted to maximize $f$, move in direction of gradient

$x \quad -2 - 4 = -6$

$y \quad 5 - 4 = 1$

$z \quad -4 + 3 = -1$

Example from: http://cs231n.github.io/optimization-2/
Backpropagation: Example 1

Now if we wanted to maximize $f$, move in direction of gradient.

- $x = 2 - 4 = -6$
- $y = 5 - 4 = 1$
- $z = -4 + 3 = -1$

Example from: http://cs231n.github.io/optimization-2/
Backpropagation: Example 1

Now if we wanted to maximize $f$, move in direction of gradient

$$x = -2 - 4 = -6$$

$$y = 5 - 4 = 1$$

$$z = -4 + 3 = -1$$

$f$ has gotten bigger!

Example from: http://cs231n.github.io/optimization-2/

<rinse and repeat>
Handout 13, Example 2

\[ f(x, \theta) = \frac{1}{1 + e^{-\theta_0 - \theta_1 x - \theta_2 x^2}} \]

\[ e^{-1} \approx 0.37 \]

\[ \frac{df}{d\theta_0} = 0.20 \]
\[ \frac{df}{d\theta_1} = 0.20 \]
\[ \frac{df}{d\theta_2} = 0.20 \]

\[ g(x) = x + y \]
\[ g'(x) = 1 \]
\[ g(-1) \approx 0.37 \]
\[ (0.37)(-0.53) \]

\[ g(x) = x + 1 \]
\[ g'(x) = 1 \]

\[ g(x) = \frac{1}{x} \]
\[ g'(x) = -\frac{1}{x^2} \]
\[ g'(1.37) \approx -0.53 \]

\[ f = 0.73 \]
\[ g(\omega_0, x_0) = \omega_0 x_0 \]

\[ \frac{\partial g}{\partial \omega_0} = x_0 \]

\[ \frac{\partial g}{\partial x_0} = \omega_0 \]

Handout 13, Example 2
Notes

• In the previous example, we were allowing all variables to change \((w_0, x_0, w_1, x_1, w_2)\)

• In the next (more realistic) example, \(x\) & \(y\) are fixed and we are allowing the weights \((w & b)\) to change
Linear Regression

Example

\[ J(w, b) = \frac{1}{2} \sum_{i=1}^{n} (wx_i + b - y_i)^2 \]

\[ g'(w) = (2)(-1) \]

\[ \begin{align*}
\omega &= 0 \\
b &= 0 \\
y &= 1 \\
x &= 1
\end{align*} \]
\( g'(w) = x \)

\( (2)(-1) \)

\( \sqrt{g'(w)} \)

\( g(a, b) = a + b \)

\( g(a) = 1 \quad g'(b) = 1 \)

\( g(x) = x - y \)

\( g'(x) = 1 \)

\( g(x) = x^2 \)

\( g'(x) = 2x \)

\( J = \frac{1}{\hat{z}} \)

\( \frac{\partial J}{\partial J} = 1 \)
Update weights using gradient descent idea

\[ w \leftarrow w - \alpha \frac{\partial J}{\partial w} \]

\[ b \leftarrow b - \alpha \frac{\partial J}{\partial b} \]

\[ w = 0.2 \]

\[ b = 0.1 \]
Outline for April 3

• Continue backpropagation
• Neural network architectures
• Choice of non-linearity (activation function)
• Choice of loss function
• Choice of weight initialization

Following Stanford CNN course reading:
http://cs231n.github.io/optimization-2/