Backpropagation and Neural Networks

Slides adapted from: http://cs231n.stanford.edu/syllabus.html

Gerard Pons-Moll
Where we are...

\[ s = f(x; W) = Wx \]

\[ L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \]

\[ L = \frac{1}{N} \sum_{i=1}^{N} L_i + \sum_k W_k^2 \]

want \( \nabla_W L \)
Optimization

Landscape image is CC0 1.0 public domain
Walking man image is CC0 1.0 public domain

```
# Vanilla Gradient Descent

while True:
    weights_grad = evaluate_gradient(loss_fun, data, weights)
    weights += - step_size * weights_grad # perform parameter update
```
Gradient descent

\[
\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}
\]

**Numerical gradient:** slow :(, approximate :(, easy to write :)  
**Analytic gradient:** fast :) , exact :) , error-prone :(  

In practice: Derive analytic gradient, check your implementation with numerical gradient
Computational graphs

\[ f = Wx \]

\[ L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \]
Convolutional network (AlexNet)

input image

weights

loss

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Neural Turing Machine

input image

loss

Figure reproduced with permission from a Twitter post by Andrej Karpathy.
Neural Turing Machine

Figure reproduced with permission from a Twitter post by Andrej Karpathy.
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

e.g. \( x = -2, y = 5, z = -4 \)
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

e.g. \( x = -2, \ y = 5, \ z = -4 \)

\[ q = x + y \quad \frac{\partial q}{\partial x} = 1, \quad \frac{\partial q}{\partial y} = 1 \]

\[ f = qz \quad \frac{\partial f}{\partial q} = z, \quad \frac{\partial f}{\partial z} = q \]

Want: \( \frac{\partial f}{\partial x}, \ \frac{\partial f}{\partial y}, \ \frac{\partial f}{\partial z} \)
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

\[ e.g. \ x = -2, \ y = 5, \ z = -4 \]

\[ q = x + y \quad \frac{\partial q}{\partial x} = 1, \quad \frac{\partial q}{\partial y} = 1 \]

\[ f = qz \quad \frac{\partial f}{\partial q} = z, \quad \frac{\partial f}{\partial z} = q \]

Want: \[ \frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} \]
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

e.g. \( x = -2, \ y = 5, \ z = -4 \)

\[ q = x + y \quad \frac{\partial q}{\partial x} = 1, \quad \frac{\partial q}{\partial y} = 1 \]

\[ f = qz \quad \frac{\partial f}{\partial q} = z, \quad \frac{\partial f}{\partial z} = q \]

Want: \( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \)
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

e.g. \( x = -2, y = 5, z = -4 \)

\[ q = x + y \quad \frac{\partial q}{\partial x} = 1, \quad \frac{\partial q}{\partial y} = 1 \]

\[ f = qz \quad \frac{\partial f}{\partial q} = z, \quad \frac{\partial f}{\partial z} = q \]

Want: \( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \)
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

e.g. \( x = -2, y = 5, z = -4 \)

\[ q = x + y \quad \frac{\partial q}{\partial x} = 1, \quad \frac{\partial q}{\partial y} = 1 \]

\[ f = qz \quad \frac{\partial f}{\partial q} = z, \quad \frac{\partial f}{\partial z} = q \]

Want: \( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \)
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

e.g. \( x = -2, \ y = 5, \ z = -4 \)

\[ q = x + y \quad \frac{\partial q}{\partial x} = 1, \ \frac{\partial q}{\partial y} = 1 \]

\[ f = qz \quad \frac{\partial f}{\partial q} = z, \ \frac{\partial f}{\partial z} = q \]

Want: \( \frac{\partial f}{\partial x}, \ \frac{\partial f}{\partial y}, \ \frac{\partial f}{\partial z} \)
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

e.g. \( x = -2, \ y = 5, \ z = -4 \)

\[ q = x + y \quad \frac{\partial q}{\partial x} = 1, \quad \frac{\partial q}{\partial y} = 1 \]

\[ f = qz \quad \frac{\partial f}{\partial q} = z, \quad \frac{\partial f}{\partial z} = q \]

Want: \( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \)
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

e.g. \( x = -2, \ y = 5, \ z = -4 \)

\[
q = x + y \quad \frac{\partial q}{\partial x} = 1, \quad \frac{\partial q}{\partial y} = 1
\]

\[ f = qz \quad \frac{\partial f}{\partial q} = z, \quad \frac{\partial f}{\partial z} = q \]

Want: \( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \)

Chain rule:

\[
\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}
\]

Diagram:
- Upstream gradient
- Local gradient
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

e.g. \( x = -2, y = 5, z = -4 \)

\[ q = x + y \quad \frac{\partial q}{\partial x} = 1, \quad \frac{\partial q}{\partial y} = 1 \]

\[ f = qz \quad \frac{\partial f}{\partial q} = z, \quad \frac{\partial f}{\partial z} = q \]

Want: \( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \)

Chain rule:

\[
\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}
\]
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

e.g. \( x = -2, y = 5, z = -4 \)

\[ q = x + y \quad \frac{\partial q}{\partial x} = 1, \quad \frac{\partial q}{\partial y} = 1 \]

\[ f = qz \quad \frac{\partial f}{\partial q} = z, \quad \frac{\partial f}{\partial z} = q \]

Want: \( \frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} \)

Chain rule:

\[ \frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x} \]

Upstream gradient

Local gradient
Backpropagation: a simple example

\[ f(x, y, z) = (x + y)z \]

E.g. \( x = -2, \ y = 5, \ z = -4 \)

\[ q = x + y \quad \frac{\partial q}{\partial x} = 1, \ \frac{\partial q}{\partial y} = 1 \]

\[ f = qz \quad \frac{\partial f}{\partial q} = z, \ \frac{\partial f}{\partial z} = q \]

Want:

\[ \frac{\partial f}{\partial x}, \ \frac{\partial f}{\partial y}, \ \frac{\partial f}{\partial z} \]

Chain rule:

\[ \frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x} \]
"local gradient\"
"local gradient"

\[ \frac{\partial z}{\partial x} \]

\[ \frac{\partial z}{\partial y} \]

\[ \frac{\partial L}{\partial z} \]

gradients
The figure illustrates the concept of gradients in the context of a function $f(x, y, z)$. The partial derivatives are shown as:

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial x}$$

$$\frac{\partial L}{\partial y}$$

These derivatives are related to the function $f$ and the variables $x$, $y$, and $z$, representing the local gradient at a point in the function's domain.
\[
\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial x} \\
\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial y}
\]

“local gradient”
\[
\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial x}
\]

\[
\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial y}
\]

“local gradient”

\[
\frac{\partial z}{\partial x}
\]

\[
\frac{\partial z}{\partial y}
\]

\[
\frac{\partial L}{\partial z}
\]

gradients
Another example: 

\[ f(w, x) = \frac{1}{1 + e^{- (w_0 x_0 + w_1 x_1 + w_2)}} \]
Another example:

\[ f(w, x) = \frac{1}{1 + e^{- (w_0 x_0 + w_1 x_1 + w_2)}} \]
Another example:

\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[
\begin{align*}
 f(x) &= e^x \\
 f_a(x) &= ax \\
 f_c(x) &= c + x
\end{align*}
\]

\[
\begin{align*}
 \frac{df}{dx} &= e^x \\
 \frac{df}{dx} &= a \\
 \frac{df}{dx} &= 1
\end{align*}
\]

\[
\begin{align*}
 f(x) &= \frac{1}{x} \\
 \frac{df}{dx} &= -\frac{1}{x^2}
\end{align*}
\]
Another example:

\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[ f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \]

\[ f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \]

\[ f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1 \]
Another example:

\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[ f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \]
\[ f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \]
\[ f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1 \]

Upstream gradient

Upstream gradient

Local gradient

\[ (1.00) \left( \frac{-1}{1.37^2} \right) = -0.53 \]
Another example:

\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[
\begin{align*}
f(x) &= e^x \\
f_a(x) &= ax
\end{align*}
\]

\[
\begin{align*}
\frac{df}{dx} &= e^x \\
\frac{df}{dx} &= a
\end{align*}
\]

\[
\begin{align*}
f(x) &= \frac{1}{x} \\
f_c(x) &= c + x
\end{align*}
\]

\[
\begin{align*}
\frac{df}{dx} &= -\frac{1}{x^2} \\
\frac{df}{dx} &= 1
\end{align*}
\]
Another example: \[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[
\begin{align*}
\text{Upstream gradient} & \quad \text{Local gradient} \\
(-0.53)(1) & = -0.53
\end{align*}
\]

\[
f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x
\]

\[
f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a
\]

\[
f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2
\]

\[
f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1
\]
Another example:

\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[
\begin{align*}
\text{if} (x) &= e^x & \frac{df}{dx} &= e^x \\
\text{if} (x) &= ax & \frac{df}{dx} &= a \\
\text{if} (x) &= \frac{1}{x} & \frac{df}{dx} &= -\frac{1}{x^2} \\
\text{if} (x) &= c + x & \frac{df}{dx} &= 1
\end{align*}
\]
Another example: 

\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[
\begin{align*}
\text{Upstream gradient} & \quad \text{Local gradient} \\
(-0.53)(e^{-1}) &= -0.20
\end{align*}
\]

\[
\begin{align*}
f(x) &= e^x \\ f_a(x) &= ax \\ f_c(x) &= c + x
\end{align*}
\]

\[
\begin{align*}
\frac{df}{dx} &= e^x \\ \frac{df}{dx} &= a \\ \frac{df}{dx} &= -1/x^2 \\
\frac{df}{dx} &= 1
\end{align*}
\]
Another example:

\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[ f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \]

\[ f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \]

\[ f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -\frac{1}{x^2} \]

\[ f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1 \]
Another example: \[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[ f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \]

\[ f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \]

\[ f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -\frac{1}{x^2} \]

\[ f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1 \]
Another example: \[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[ f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \]

\[ f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \]

\[ f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1 \]
Another example:

\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

[upstream gradient] \times [local gradient]

\[ 0.2 \times 1 = 0.2 \]

\[ 0.2 \times 1 = 0.2 \text{ (both inputs!)} \]

\[
\begin{align*}
  f(x) &= e^x \\
  f_a(x) &= ax \\
  f_c(x) &= c + x
\end{align*}
\]

\[
\begin{align*}
  \frac{df}{dx} &= e^x \\
  \frac{df}{dx} &= a \\
  \frac{df}{dx} &= 1
\end{align*}
\]

\[
\begin{align*}
  f(x) &= \frac{1}{x} \\
  f'(x) &= -\frac{1}{x^2}
\end{align*}
\]
Another example:

\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[
\begin{align*}
    f(x) &= e^x \
    f_a(x) &= ax \\
    f_c(x) &= c + x
\end{align*}
\]

\[
\begin{align*}
    \frac{df}{dx} &= e^x \
    \frac{df}{dx} &= a \
    \frac{df}{dx} &= 1
\end{align*}
\]

\[
\begin{align*}
    f(x) &= \frac{1}{x} \\
    f_c(x) &= c + x
\end{align*}
\]

\[
\begin{align*}
    \frac{df}{dx} &= -\frac{1}{x^2} \\
    \frac{df}{dx} &= 1
\end{align*}
\]
Another example:

\[ f(w, x) = \frac{1}{1 + e^{- (w_0 x_0 + w_1 x_1 + w_2)}} \]

[upstream gradient] \times [local gradient]

- \( x_0: [0.2] \times [2] = 0.4 \)
- \( w_0: [0.2] \times [-1] = -0.2 \)

\[
\begin{align*}
f(x) &= e^x \\
f_a(x) &= ax \\
\frac{df}{dx} &= e^x \\
\frac{df}{dx} &= a \\
f_c(x) &= c + x \\
\frac{df}{dx} &= 1 \\
f(x) &= \frac{1}{x} \\
\frac{df}{dx} &= -\frac{1}{x^2}
\end{align*}
\]
\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

\[ \frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left( \frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left( \frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x)) \sigma(x) \]

Computational graph representation may not be unique. Choose one where local gradients at each node can be easily expressed!
sigmoid function

\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

\[ \frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left( \frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left( \frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x)) \sigma(x) \]

Computational graph representation may not be unique. Choose one where local gradients at each node can be easily expressed!

\[
\text{[upstream gradient]} \times \text{[local gradient]}
\]

\[
[1.00] \times [(1 - 0.73) (0.73)] = 0.2
\]
Patterns in backward flow

**add** gate: gradient distributor

![Diagram](image-url)
Patterns in backward flow

**add** gate: gradient distributor

Q: What is a **max** gate?
Patterns in backward flow

**add** gate: gradient distributor

**max** gate: gradient router
Patterns in backward flow

**add** gate: gradient distributor

**max** gate: gradient router

Q: What is a **mul** gate?
Patterns in backward flow

**add** gate: gradient distributor

**max** gate: gradient router

**mul** gate: gradient switcher
Gradients add at branches
Gradients for vectorized code

This is now the Jacobian matrix (derivative of each element of $z$ w.r.t. each element of $x$)

(x, y, z are now vectors)

"local gradient"
Vectorized operations

\[ f(x) = \max(0, x) \quad (\text{elementwise}) \]
Vectorized operations

\[ \frac{\partial L}{\partial x} = \frac{\partial f}{\partial x} \frac{\partial L}{\partial f} \]

Jacobian matrix

4096-d input vector

f(x) = max(0,x) (elementwise)

4096-d output vector

Q: what is the size of the Jacobian matrix?
Vectorized operations

$$f(x) = \max(0,x)$$ (elementwise)

Q: what is the size of the Jacobian matrix? [4096 x 4096!]

$$\frac{\partial L}{\partial x} = \frac{\partial f}{\partial x} \frac{\partial L}{\partial f}$$

Jacobian matrix
Vectorized operations

Q: what is the size of the Jacobian matrix? [4096 x 4096!]

in practice we process an entire minibatch (e.g. 100) of examples at one time:

i.e. Jacobian would technically be a [409,600 x 409,600] matrix :\
Vectorized operations

\[ \frac{\partial L}{\partial x} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial L}{\partial f} \end{bmatrix} \]

Jacobian matrix

Q: what is the size of the Jacobian matrix?
[4096 x 4096!]

Q2: what does it look like?

\[ f(x) = \max(0,x) \text{ (elementwise)} \]

4096-d input vector

4096-d output vector
A vectorized example: \( f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2 \)
A vectorized example: \[ f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_{i}^{2} \]
\[
\in \mathbb{R}^{n} \in \mathbb{R}^{n \times n}
\]
A vectorized example: \( f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2 \)
A vectorized example: \( f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)^2_i \)

\[
W = \begin{bmatrix}
0.1 & 0.5 \\
-0.3 & 0.8 \\
\end{bmatrix}
\]

\[
x = \begin{bmatrix}
0.2 \\
0.4 \\
\end{bmatrix}
\]

\[
q = W \cdot x = \begin{pmatrix}
W_{1,1}x_1 + \cdots + W_{1,n}x_n \\
\vdots \\
W_{n,1}x_1 + \cdots + W_{n,n}x_n
\end{pmatrix}
\]

\[
f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2
\]
A vectorized example: \[ f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2 \]

\[
W = \begin{bmatrix}
0.1 & 0.5 \\
-0.3 & 0.8
end{bmatrix}
\]

\[
x = \begin{bmatrix}
0.2 \\
0.4
end{bmatrix}
\]

\[
q = W \cdot x = \begin{pmatrix}
W_{1,1}x_1 + \cdots + W_{1,n}x_n \\
\vdots \\
W_{n,1}x_1 + \cdots + W_{n,n}x_n
end{pmatrix}
\]

\[f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2\]
A vectorized example: \( f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2 \)

\[
\begin{bmatrix}
0.1 & 0.5 \\
-0.3 & 0.8
\end{bmatrix}
\begin{bmatrix}
0.2 \\
0.4
\end{bmatrix}
\]

\[
q = W \cdot x = 
\begin{pmatrix}
W_{1,1}x_1 + \cdots + W_{1,n}x_n \\
\vdots \\
W_{n,1}x_1 + \cdots + W_{n,n}x_n
\end{pmatrix}
\]

\[
f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2
\]

L2

0.116

1.00
A vectorized example: \( f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)^2_i \)

\[
W = \begin{bmatrix}
0.1 & 0.5 \\
-0.3 & 0.8
\end{bmatrix}
\]

\[
x = \begin{bmatrix}
0.2 \\
0.4
\end{bmatrix}
\]

\[
q = W \cdot x = \begin{pmatrix}
W_{1,1}x_1 + \cdots + W_{1,n}x_n \\
\vdots \\
W_{n,1}x_1 + \cdots + W_{n,n}x_n
\end{pmatrix}
\]

\[
f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2
\]

\[
\frac{\partial f}{\partial q_i} = 2q_i
\]

\[
\nabla_q f = 2q
\]
A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$$
\begin{bmatrix}
0.1 & 0.5 \\
-0.3 & 0.8
\end{bmatrix}
\begin{bmatrix}
0.2 \\
0.4
\end{bmatrix}
\rightarrow
\begin{bmatrix}
0.22 \\
0.26
\end{bmatrix}
\rightarrow
\begin{bmatrix}
0.44 \\
0.52
\end{bmatrix}
$$

$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial f}{\partial q_i} = 2q_i$$

$\nabla_q f = 2q$
A vectorized example: \( f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)^2_i \)

\[
W = \begin{bmatrix}
0.1 & 0.5 \\
-0.3 & 0.8 \\
\end{bmatrix} \\
x = \begin{bmatrix}
0.2 \\
0.4 \\
\end{bmatrix}
\]

\[
q = W \cdot x = \begin{pmatrix}
W_{1,1}x_1 + \cdots + W_{1,n}x_n \\
\vdots \\
W_{n,1}x_1 + \cdots + W_{n,n}x_n \\
\end{pmatrix}
\]

\[
f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2
\]

\[
\frac{\partial q_k}{\partial W_{i,j}} = 1_{k=i}x_j
\]
A vectorized example:

\[ f(x, W) = \| W \cdot x \|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2 \]

\[
W = \begin{bmatrix}
0.1 & 0.5 \\
-0.3 & 0.8 
\end{bmatrix}
\]

\[
x = \begin{bmatrix}
0.2 \\
0.4 
\end{bmatrix}
\]

\[
q = W \cdot x = \begin{pmatrix}
W_{1,1}x_1 + \cdots + W_{1,n}x_n \\
\vdots \\
W_{n,1}x_1 + \cdots + W_{n,n}x_n
\end{pmatrix}
\]

\[ f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2 \]

\[
\frac{\partial q_k}{\partial W_{i,j}} = 1_{k=i}x_j
\]

\[
\frac{\partial f}{\partial W_{i,j}} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}}
= \sum_k (2q_k)(1_{k=i}x_j)
= 2q_i x_j
\]
A vectorized example: $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)^2_i$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$
A vectorized example:

\[ f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)^2_i \]

\[ \nabla_W f = 2q \cdot x^T \]

\[ q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix} \]

\[ f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2 \]
A vectorized example: \( f(x, W) = \| W \cdot x \|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2 \)

\[
W = \begin{bmatrix}
0.1 & 0.5 \\
-0.3 & 0.8 \\
0.088 & 0.176 \\
0.104 & 0.208 \\
0.2 & 0.4
\end{bmatrix}
\]

\[
x = \begin{bmatrix}
0.22 \\
0.26 \\
0.44 \\
0.52
\end{bmatrix}
\]

\[
L2 = \begin{bmatrix}
0.116 \\
1.00
\end{bmatrix}
\]

\[
\nabla_W f = 2q \cdot x^T
\]

Always check: The gradient with respect to a variable should have the same shape as the variable.

\[
q = W \cdot x = \begin{pmatrix}
W_{1,1} x_1 + \cdots + W_{1,n} x_n \\
\vdots \\
W_{n,1} x_1 + \cdots + W_{n,n} x_n
\end{pmatrix}
\]

\[
f(q) = \| q \|^2 = q_1^2 + \cdots + q_n^2
\]

\[
\frac{\partial q_k}{\partial W_{i,j}} = \begin{cases}
1_{k=i} x_j & \text{if } k = i \\
0 & \text{otherwise}
\end{cases}
\]

\[
\frac{\partial f}{\partial W_{i,j}} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}}
\]

\[
= \sum_k (2q_k) (1_{k=i} x_j)
\]

\[
= 2q_i x_j
\]
A vectorized example: \[ f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2 \]

\[
W = \begin{bmatrix}
0.1 & 0.5 \\
-0.3 & 0.8 \\
0.088 & 0.176 \\
0.104 & 0.208 \\
0.2 & 0.4
\end{bmatrix}
\]

\[
x = \begin{bmatrix}
0.22 \\
0.26 \\
0.44 \\
0.52
\end{bmatrix}
\]

\[
q = W \cdot x = \begin{pmatrix}
W_{1,1}x_1 + \cdots + W_{1,n}x_n \\
\vdots \\
W_{n,1}x_1 + \cdots + W_{n,n}x_n
\end{pmatrix}
\]

\[
f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2
\]

\[
\frac{\partial q_k}{\partial x_i} = W_{k,i}
\]
A vectorized example: \[ f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)^2_i \]

\[
W = \begin{bmatrix}
0.1 & 0.5 \\
-0.3 & 0.8 \\
0.088 & 0.176 \\
0.104 & 0.208 \\
0.2 & 0.4 \\
\end{bmatrix}
\]

\[
x = \begin{bmatrix}
0.22 \\
0.26 \\
0.44 \\
0.52 \\
\end{bmatrix}
\]

\[
q = W \cdot x = \begin{pmatrix}
W_{1,1}x_1 + \cdots + W_{1,n}x_n \\
\vdots \\
W_{n,1}x_1 + \cdots + W_{n,n}x_n \\
\end{pmatrix}
\]

\[
f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2
\]

\[
\frac{\partial q_k}{\partial x_i} = W_{k,i}
\]

\[
\frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial x_i} = \sum_k 2q_k W_{k,i}
\]
A vectorized example: $f(x, W) = \|W \cdot x\|^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

\[ q = W \cdot x = \begin{pmatrix} W_{1,1} x_1 + \cdots + W_{1,n} x_n \\ \vdots \\ W_{n,1} x_1 + \cdots + W_{n,n} x_n \end{pmatrix} \]

\[ f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2 \]

\[ \nabla_x f = 2W^T \cdot q \]
In discussion section: A matrix example...

\[ z_1 = X W_1 \]
\[ h_1 = \text{ReLU}(z_1) \]
\[ \hat{y} = h_1 W_2 \]
\[ L = ||\hat{y}||^2_2 \]
\[ \frac{\partial L}{\partial W_2} = \text{?} \]
\[ \frac{\partial L}{\partial W_1} = \text{?} \]
Modularized implementation: forward / backward API

Graph (or Net) object  *(rough pseudo code)*

```python
class ComputationalGraph(object):
    
    def forward(inputs):
        
        # 1. [pass inputs to input gates...]
        
        # 2. forward the computational graph:
        
        for gate in self.graph.nodes_topologically_sorted():
            gate.forward()

        return loss # the final gate in the graph outputs the loss

    def backward():
        
        for gate in reversed(self.graph.nodes_topologically_sorted()):
            gate.backward() # little piece of backprop (chain rule applied)

        return inputs_gradients
```
Modularized implementation: forward / backward API

```python
class MultiplyGate(object):
    def forward(x, y):
        z = x * y
        self.x = x # must keep these around!
        self.y = y
        return z
    def backward(dz):
        dx = self.y * dz # [dz/dx * dL/dz]
        dy = self.x * dz # [dz/dy * dL/dz]
        return [dx, dy]
```

(x, y, z are scalars)
Example: Caffe layers

Caffe is licensed under BSD 2-Clause
Caffe Sigmoid Layer

\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

* top_diff (chain rule)
In Assignment 1: Writing SVM / Softmax

Stage your forward/backward computation!

E.g. for the SVM:

```python
# receive W (weights), X (data)
# forward pass (we have 6 lines)
scores = #...
margins = #...
data_loss = #...
reg_loss = #...
loss = data_loss + reg_loss
# backward pass (we have 5 lines)
dmargins = # ... (optionally, we go direct to dscores)
dscores = #...
dW = #...
```

\[ f = Wx \]
\[ L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \]
Summary so far...

- neural nets will be very large: impractical to write down gradient formula by hand for all parameters
- **backpropagation** = recursive application of the chain rule along a computational graph to compute the gradients of all inputs/parameters/intermediates
- implementations maintain a graph structure, where the nodes implement the `forward()` / `backward()` API
- **forward**: compute result of an operation and save any intermediates needed for gradient computation in memory
- **backward**: apply the chain rule to compute the gradient of the loss function with respect to the inputs
Next: Neural Networks
Neural networks: without the brain stuff

(Before) Linear score function: \( f = Wx \)
Neural networks: without the brain stuff

(Before) Linear score function: \[ f = Wx \]

(Now) 2-layer Neural Network \[ f = W_2 \max(0, W_1 x) \]
Neural networks: without the brain stuff

(Before) Linear score function: \( f = Wx \)

(Now) 2-layer Neural Network

\[
f = W_2 \max(0, W_1 x)
\]
Neural networks: without the brain stuff

(Before) Linear score function: \( f = Wx \)

(Now) 2-layer Neural Network: \( f = W_2 \max(0, W_1x) \)
Neural networks: without the brain stuff

(Before) Linear score function: \( f = Wx \)

(Now) 2-layer Neural Network

\( f = W_2 \max(0, W_1 x) \)

or 3-layer Neural Network

\( f = W_3 \max(0, W_2 \max(0, W_1 x)) \)
Full implementation of training a 2-layer Neural Network needs ~20 lines:

```python
import numpy as np
from numpy.random import randn

N, D_in, H, D_out = 64, 1000, 100, 10
x, y = randn(N, D_in), randn(N, D_out)
w1, w2 = randn(D_in, H), randn(H, D_out)

for t in range(2000):
    h = 1 / (1 + np.exp(-x.dot(w1)))
    y_pred = h.dot(w2)
    loss = np.square(y_pred - y).sum()
    print(t, loss)

    grad_y_pred = 2.0 * (y_pred - y)
    grad_w2 = h.T.dot(grad_y_pred)
    grad_h = grad_y_pred.dot(w2.T)
    grad_w1 = x.T.dot(grad_h * h * (1 - h))

    w1 -= 1e-4 * grad_w1
    w2 -= 1e-4 * grad_w2
```
In HW: Writing a 2-layer net

```python
# receive W1,W2,b1,b2 (weights/biases), X (data)
# forward pass:
h1 = #... function of X,W1,b1
scores = #... function of h1,W2,b2
loss = #... (several lines of code to evaluate Softmax loss)
# backward pass:
dscores = #...
dh1,dW2,db2 = #...
dW1,db1 = #...
```
Impulses carried toward cell body

Impulses carried away from cell body
Impulses carried toward cell body

This image by Felipe Perucho
is licensed under CC-BY 3.0

Impulses carried away from cell body

---

This image by Felipe Perucho
is licensed under CC-BY 3.0
Impulses carried toward cell body

Impulses carried away from cell body

dendrite

axon

presynaptic terminal

cell body

sigmoid activation function

\[
\frac{1}{1 + e^{-x}}
\]
Impulses carried toward cell body

Impulses carried away from cell body

This image by Felipe Perucho is licensed under CC-BY 3.0

```python
class Neuron:
    # ...
    def neuron_tick(inputs):
        """assume inputs and weights are 1-D numpy arrays and bias is a number """
        cell_body_sum = np.sum(inputs * self.weights) + self.bias
        firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum))  # sigmoid activation func
        return firing_rate
```
Be very careful with your brain analogies!

**Biological Neurons:**
- Many different types
- Dendrites can perform complex non-linear computations
- Synapses are not a single weight but a complex non-linear dynamical system
- Rate code may not be adequate

[Dendritic Computation. London and Hausser]
Activation functions

**Sigmoid**
\[ \sigma(x) = \frac{1}{1+e^{-x}} \]

\( \tanh(x) \)

**ReLU**
\[ \max(0, x) \]

**Leaky ReLU**
\[ \max(0.1x, x) \]

**Maxout**
\[ \max(w_1^T x + b_1, w_2^T x + b_2) \]

**ELU**
\[
\begin{cases} 
  x & \text{if } x \geq 0 \\
  \alpha(e^x - 1) & \text{if } x < 0 
\end{cases}
\]
Neural networks: Architectures

“2-layer Neural Net”, or “1-hidden-layer Neural Net”

“3-layer Neural Net”, or “2-hidden-layer Neural Net”

“Fully-connected” layers
Example feed-forward computation of a neural network

```python
class Neuron:
    # ...
    def neuron_tick(inputs):
        """ assume inputs and weights are 1-D numpy arrays and bias is a number """
        cell_body_sum = np.sum(inputs * self.weights) + self.bias
        firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum))  # sigmoid activation function
        return firing_rate
```

We can efficiently evaluate an entire layer of neurons.
Example feed-forward computation of a neural network

```
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x))  # activation function (use sigmoid)
x = np.random.randn(3, 1)  # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1)  # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2)  # calculate second hidden layer activations (4x1)
out = np.dot(W3, h2) + b3  # output neuron (1x1)
```
Summary

- We arrange neurons into fully-connected layers
- The abstraction of a layer has the nice property that it allows us to use efficient vectorized code (e.g. matrix multiplies)
- Neural networks are not really neural
- Next time: Convolutional Neural Networks