Lecture 12:
- Computational Graph
- Backpropagation
Last time...

Multilayer Perceptron

• Layer Representation

\[ y_i = W_i x_i \]
\[ x_{i+1} = \sigma(y_i) \]

• (typically) iterate between linear mapping \( Wx \) and nonlinear function

• Loss function \( l(y, y_i) \) to measure quality of estimate so far
Output of the network can be written as:

\[ h_j(x) = f(v_{j0} + \sum_{i=1}^{D} x_i v_{ji}) \]

\[ o_k(x) = g(w_{k0} + \sum_{j=1}^{J} h_j(x) w_{kj}) \]

(j indexing hidden units, k indexing the output units, D number of inputs)

Activation functions \( f, g \): sigmoid/logistic, tanh, or rectified linear (ReLU)

\[ \sigma(z) = \frac{1}{1 + \exp(-z)}, \quad \tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}, \quad \text{ReLU}(z) = \max(0, z) \]
Last time... **Forward Pass in Python**

- Example code for a forward pass for a 3-layer network in Python:

  ```python
  # 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)
  ```

- Can be implemented efficiently using matrix operations
- Example above: $W_1$ is matrix of size $4 \times 3$, $W_2$ is $4 \times 4$. What about biases and $W_3$?

Last time… Mini-batch Gradient Descent

- only use a small portion of the training set to compute the gradient

```python
# Vanilla Minibatch Gradient Descent

while True:
    data_batch = sample_training_data(data, 256)  # sample 256 examples
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
    weights += - step_size * weights_grad  # perform parameter update
```

there are also more fancy update formulas (momentum, Adagrad, RMSProp, Adam, …)
Back-propagation
Computational Graph

\[ f = Wx \]

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

input tape

loss
\[ f(x, y, z) = (x + y)z \]
e.g. \( x = -2, y = 5, z = -4 \)
\[ f(x, y, z) = (x + y)z \]
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: \[ \frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} \]
\[ f(x, y, z) = (x + y)z \]

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: \( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \)
\[ f(x, y, z) = (x + y)z \]
e.g. \( x = -2, y = 5, z = -4 \)

\[
\begin{align*}
q &= x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1 \\
\end{align*}
\]

\[
\begin{align*}
f &= qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q \\
\end{align*}
\]

Want:

\[
\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}
\]
\[
f(x, y, z) = (x + y)z
\]
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: \(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\)
$$f(x, y, z) = (x + y)z$$

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: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$
\[ f(x, y, z) = (x + y)z \]
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: \( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \)
\[ f(x, y, z) = (x + y)z \]

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: \[ \frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} \]
\[ f(x, y, z) = (x + y)z \]
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: \( \frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z} \)
$f(x, y, z) = (x + y)z$

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: \quad \frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z}$
\[ f(x, y, z) = (x + y)z \]
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:

\[
\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}
\]
\[ f(x, y, z) = (x + y)z \]
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: \( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \)
\[ f(x, y, z) = (x + y)z \]
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:**

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

\[ f \]

\[ x \]

\[ y \]

\[ z \]
activations

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

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

"local gradient"

\[ f \]

\[ z \]
activations

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

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

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

"local gradient"

gradients
\[
\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial x}
\]

activations

“local gradient”

gradients
activations

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

“local gradient”

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

gradients

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

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

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

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

"local gradient"

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

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

gradients

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

28
Another example: \[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]
Another example: \[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

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

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

\[
\begin{align*}
  f(x) &= e^x & \Rightarrow & & \frac{df}{dx} &= e^x \\
  f_a(x) &= ax & \Rightarrow & & \frac{df}{dx} &= a
\end{align*}
\]

\[
\begin{align*}
  f(x) &= \frac{1}{x} & \Rightarrow & & \frac{df}{dx} &= -\frac{1}{x^2} \\
  f_c(x) &= c + x & \Rightarrow & & \frac{df}{dx} &= 1
\end{align*}
\]
Another example: \[ f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}} \]

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

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

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

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

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

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

\[
\begin{align*}
\text{f(x)} &= e^x \\
\text{f}_a(x) &= ax
\end{align*}
\]  
\[
\begin{align*}
\frac{df}{dx} &= e^x \\
\frac{df}{dx} &= a
\end{align*}
\]  
\[
\begin{align*}
\text{f}_c(x) &= c + x
\end{align*}
\]  
\[
\frac{df}{dx} = 1
\]
Another example: \[ f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}} \]

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

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

\[(e^{-1})(-0.53) = -0.20\]

- \[ f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \]
- \[ f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \]
- \[ f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1 \]
- \[ f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2 \]
Another example: \( f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \)

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

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

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

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

[local gradient] \times [its gradient]

\[ [1] \times [0.2] = 0.2 \]

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

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

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

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

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

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

\[ w_0 \cdot 2.00 \rightarrow 2.00 \] \[ x_0 \cdot -1.00 \rightarrow -2.00 \]
\[ 0.40 \]

\[ x_0 \cdot [2] \times [0.2] = 0.4 \]
\[ w_0 \cdot [-1] \times [0.2] = -0.2 \]

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

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

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

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

sigmoid function

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

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

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

(0.73) * (1 - 0.73) = 0.2

sigmoid function

sigmoid gate
Patterns in backward flow

- **add** gate: gradient distributor
- **max** gate: gradient router
- **mul** gate: gradient… “switcher”?
Gradients add at branches
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
```
Implementation: forward/backward API

\[
\begin{align*}
\text{class MultiplyGate(object):} \\
\quad \text{def forward(x,y):} \\
\qquad z &= x \times y \\
\qquad \text{return } z \\
\quad \text{def backward(dz):} \\
\qquad \# dx = ... \#todo \\
\qquad \# dy = ... \#todo \\
\qquad \text{return } [dx, dy]
\end{align*}
\]

\(\frac{\partial L}{\partial z}\) \(\frac{\partial L}{\partial x}\)

(x, y, z are scalars)
Implementation: forward/backward API

```
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)
Summary

- neural nets will be very large: no hope of writing 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.
Where are we now...

Mini-batch SGD

Loop:
1. **Sample** a batch of data
2. **Forward** prop it through the graph, get loss
3. **Backprop** to calculate the gradients
4. **Update** the parameters using the gradient
Next Lecture:

Convolutional Neural Networks