So Far: Linear Models

\[ L(w) = \lambda \|w\|_2^2 + \sum_{i=1}^{n} (y_i - w^T x_i)^2 \]

- Example: find \( w \) minimizing squared error over data
- Each datapoint represented by some vector \( x \)
- Can find optimal \( w \) with \(~10\) line derivation
Last Class

\[ L(w) = \lambda \|w\|^2_2 + \sum_{i=1}^{n} L(y_i, f(x; x)) \]

- What about an arbitrary loss function \( L \)?
- What about an arbitrary parametric function \( f \)?
- Solution: take the gradient, do gradient descent

\[ w_{i+1} = w_i - \alpha \nabla_w L(f(w_i)) \]

What if \( L(f(w)) \) is complicated?

Today!
Taking the Gradient – Review

\[ f(x) = (-x + 3)^2 \]

\[ f = q^2 \quad q = r + 3 \quad r = -x \]

\[ \frac{\partial f}{\partial q} = 2q \quad \frac{\partial q}{\partial r} = 1 \quad \frac{\partial q}{\partial x} = -1 \]

\[ \frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial r} \frac{\partial r}{\partial x} = 2q \times 1 \times -1 \]

\[ = -2(-x + 3) \]

\[ = 2x - 6 \]
Supplemental Reading

- Lectures can only introduce you to a topic
- You will solidify your knowledge by **doing**
- I highly recommend working through everything in the Stanford CS213N resources
  - [http://cs231n.github.io/optimization-2/](http://cs231n.github.io/optimization-2/)
- These slides follow the general examples with a few modifications. The primary difference is that I define local variables n, m per-block.
Let’s Do This Another Way

Suppose we have a box representing a function $f$.

This box does two things:

**Forward:** Given forward input $n$, compute $f(n)$

**Backwards:** Given backwards input $g$, return $g \cdot \frac{df}{dn}$
Let’s Do This Another Way

\[ f(x) = (-x + 3)^2 \]

\[
\frac{\partial}{\partial n} n^2 = 2n = 2(-x + 3) = -2x + 6
\]

\[
\frac{\partial}{\partial n} \times 1 = (-2x + 6) \times 1
\]
Let’s Do This Another Way

\[ f(x) = (-x + 3)^2 \]
Let’s Do This Another Way

\[ f(x) = (-x + 3)^2 \]

\[
\begin{align*}
\frac{\partial}{\partial n} &= -1 \\
-1 \ast (-2x + 6) \\
2x - 6
\end{align*}
\]
Let’s Do This Another Way

\[ f(x) = (-x + 3)^2 \]
Two Inputs

Given two inputs, just have two input/output wires

**Forward:** the same

**Backward:** the same – send gradients with respect to each variable
\[ f(x, y, z) = (x+y)z \]
\[ f(x, y, z) = (x+y)z \]

Multiplication swaps inputs, multiplies gradient

\[ \frac{\partial}{\partial n} nm = m \quad \rightarrow z \ast 1 \]
\[ \frac{\partial}{\partial m} nm = n \quad \rightarrow (x+y) \ast 1 \]

Example Credit: Karpathy and Fei-Fei
\[ f(x, y, z) = (x + y)z \]

**Example Credit:** Karpathy and Fei-Fei
\[ f(x,y,z) = (x+y)z \]

\[
\frac{\partial (x + y)z}{\partial x} = z \quad \frac{\partial (x + y)z}{\partial y} = z \quad \frac{\partial (x + y)z}{\partial z} = (x + y)
\]

Example Credit: Karpathy and Fei-Fei
Once More, With Numbers!
\[ f(x, y, z) = (x + y)z \]
\[ f(x,y,z) = (x+y)z \]

\[ \frac{\partial}{\partial n} nm = m \quad \rightarrow \quad 10 \times 1 \]

\[ \frac{\partial}{\partial m} nm = n \quad \rightarrow \quad 5 \times 1 \]

Example Credit: Karpathy and Fei-Fei
\[ f(x,y,z) = (x+y)z \]

\[ \frac{\partial}{\partial n} (n + m) = 1 \]
\[ \rightarrow 1 \times 10 \times 1 \]

\[ \frac{\partial}{\partial m} (n + m) = 1 \]
\[ \rightarrow 1 \times 5 \times 1 \]
Think You’ve Got It?

\[ L(x) = (w - 6)^2 \]

- We want to fit a model \( w \) that just will equal 6.
- World’s most basic linear model / neural net: no inputs, just constant output.
I’ll Need a Few Volunteers

\[ L(x) = (w - 6)^2 \]

Job #1 (n-6):
- **Forward:** Compute n-6
- **Backwards:** Multiply by 1

Job #2 (n^2):
- **Forward:** Compute n^2
- **Backwards:** Multiply by 2n

Job #3:
- **Backwards:** Write down 1
Preemptively

• The diagrams look complex but that’s since we’re covering the details together
Something More Complex

\[ f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}} \]
\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]
\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

a. \[ \frac{\partial}{\partial n} m + n = 1 \]
b. \[ \frac{\partial}{\partial n} mn = m \]
c. \[ \frac{\partial}{\partial n} e^n = e^n \]
d. \[ \frac{\partial}{\partial n} n^{-1} = -n^{-2} \]
e. \[ \frac{\partial}{\partial n} an = a \]
f. \[ \frac{\partial}{\partial n} c + n = 1 \]
\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

Example Credit: Karpathy and Fei-Fei
Where does 1.37 come from?

Example Credit: Karpathy and Fei-Fei
\[ f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}} \]

\[
\begin{align*}
\frac{\partial}{\partial n} m + n &= 1 \\
\frac{\partial}{\partial n} mn &= m \\
\frac{\partial}{\partial n} e^n &= e^n \\
\frac{\partial}{\partial n} n^{-1} &= -n^{-2} \\
\frac{\partial}{\partial n} an &= a \\
\frac{\partial}{\partial n} c + n &= 1
\end{align*}
\]

Example Credit: Karpathy and Fei-Fei
\[ f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}} \]

\[ a \quad \frac{\partial}{\partial n} m + n = 1 \]

\[ b \quad \frac{\partial}{\partial n} mn = m \]

\[ c \quad \frac{\partial}{\partial n} e^n = e^n \]

\[ d \quad \frac{\partial}{\partial n} n^{-1} = -n^{-2} \]

\[ e \quad \frac{\partial}{\partial n} an = a \]

\[ f \quad \frac{\partial}{\partial n} c + n = 1 \]

\[ e^{-1} \times -0.53 = -0.2 \]

Example Credit: Karpathy and Fei-Fei
\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[ \frac{\partial}{\partial n} m + n = 1 \]

\[ \frac{\partial}{\partial n} mn = m \]

\[ \frac{\partial}{\partial n} e^n = e^n \]

\[ \frac{\partial}{\partial n} n^{-1} = -n^{-2} \]

\[ \frac{\partial}{\partial n} an = a \]

\[ \frac{\partial}{\partial n} c + n = 1 \]
\[ f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}} \]

\[ a \quad \frac{\partial}{\partial n} m + n = 1 \]

\[ b \quad \frac{\partial}{\partial n} mn = m \]

\[ c \quad \frac{\partial}{\partial n} e^n = e^n \]

\[ d \quad \frac{\partial}{\partial n} n^{-1} = -n^{-2} \]

\[ e \quad \frac{\partial}{\partial n} an = a \]

\[ f \quad \frac{\partial}{\partial n} c + n = 1 \]

Example Credit: Karpathy and Fei-Fei
\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[ a \frac{\partial}{\partial n} m + n = 1 \]
\[ b \frac{\partial}{\partial n} mn = m \]
\[ c \frac{\partial}{\partial n} e^n = e^n \]
\[ d \frac{\partial}{\partial n} n^{-1} = -n^{-2} \]
\[ e \frac{\partial}{\partial n} an = a \]
\[ f \frac{\partial}{\partial n} c + n = 1 \]

Example Credit: Karpathy and Fei-Fei
\[ f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}} \]

\[ \begin{align*}
  a & : \frac{\partial}{\partial n} m + n = 1 \\
  b & : \frac{\partial}{\partial n} mn = m \\
  c & : \frac{\partial}{\partial n} e^n = e^n \\
  d & : \frac{\partial}{\partial n} n^{-1} = -n^{-2} \\
  e & : \frac{\partial}{\partial n} an = a \\
  f & : \frac{\partial}{\partial n} c + n = 1
\end{align*} \]

Example Credit: Karpathy and Fei-Fei
\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

**Example Credit:** Karpathy and Fei-Fei
\[ f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}} \]

\[ \frac{\partial}{\partial n} m + n = 1 \]
\[ \frac{\partial}{\partial n} mn = m \]
\[ \frac{\partial}{\partial n} e^n = e^n \]
\[ \frac{\partial}{\partial n} n^{-1} = -n^{-2} \]
\[ \frac{\partial}{\partial n} an = a \]
\[ \frac{\partial}{\partial n} c + n = 1 \]

Example Credit: Karpathy and Fei-Fei
\[ f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}} \]

\[ \frac{\partial}{\partial n} m + n = 1 \]
\[ \frac{\partial}{\partial n} mn = m \]
\[ \frac{\partial}{\partial n} e^n = e^n \]
\[ \frac{\partial}{\partial n} n^{-1} = -n^{-2} \]
\[ \frac{\partial}{\partial n} an = a \]
\[ \frac{\partial}{\partial n} c + n = 1 \]

PHEW!

Example Credit: Karpathy and Fei-Fei
Each block computes backwards \((g) \ast \text{local gradient } (\frac{df}{dx_i})\) at the evaluation point.
Multiple Outputs Flowing Back

Gradients from different backwards sum up

\[ \sum_{j=1}^{K} g_j \left( \frac{\partial f_j}{\partial x_i} \right) \]
Multiple Outputs Flowing Back

\[ f(x) = (-x + 3)^2 \]
Multiple Outputs Flowing Back

\[ f(x) = (-x + 3)^2 \]

\[ \frac{\partial f}{\partial x} = (x - 3) + (x - 3) \]

\[ = 2x - 6 \]
Does It Have To Be So Painful?

\[ f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}} \]

Example Credit: Karpathy and Fei-Fei
Does It Have To Be So Painful?

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

\[ \frac{\partial}{\partial n} \sigma(n) = \frac{e^{-n}}{(1 + e^{-n})^2} = \left( \frac{1 + e^{-n} - 1}{1 + e^{-n}} \right) \left( \frac{1}{1 + e^{-n}} \right) \]

\[ = \left( 1 - \sigma(n) \right) \sigma(n) \]

For the curious

Line 1 to 2: \[ \frac{\partial}{\partial n} \sigma(n) = \left( \frac{-1}{(1 + e^{-n})^2} \right) \times 1 \times e^{-n} \times -1 \]

Chain rule: \[ \frac{d}{dx} \left( \frac{1}{x} \right) \times \frac{d}{dx} (1+x) \]

\[ \frac{d}{dx} (e^x) \times \frac{d}{dx} (-x) \]

Example Credit: Karpathy and Fei-Fei
Does It Have To Be So Painful?

\[
f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}
\]

\[
\sigma(n) = \frac{1}{1 + e^{-n}} \quad \frac{\partial \sigma(n)}{\partial n} = (1 - \sigma(n))\sigma(n)
\]

Example Credit: Karpathy and Fei-Fei
Does It Have To Be So Painful?

- Can compute for any function
- Pick your functions carefully: existing code is usually structured into sensible blocks
Building Blocks

Takes signals from other cells, processes, and sends out.

Input from other cells

Output to other cells

Neuron diagram credit: Karpathy and Fei-Fei
Artificial Neuron

Weighted average of other neuron outputs passed through an activation function

\[
\sum_{i} w_i x_i + b \rightarrow \text{Activation} \rightarrow f\left(\sum_{i} w_i x_i + b\right)
\]
Artificial Neuron

Can differentiate whole thing e.g., dNeuron/dx_1.

What can we now do?
Artificial Neuron

Each artificial neuron is a linear model + an activation function $f$
Can find $w$, $b$ that minimizes a loss function with gradient descent
Artificial Neurons

Connect neurons to make a more complex function; use backprop to compute gradient.
What’s The Activation Function

Sigmoid

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

- Nice interpretation
- Squashes things to (0,1)
- Gradients are near zero if neuron is high/low
What’s The Activation Function

ReLU (Rectifying Linear Unit)
\[ \max(0, x) \]

- Constant gradient
- Converges ~6x faster
- If neuron negative, zero gradient. Be careful!
What’s The Activation Function

Leaky ReLU

(Rectifying Linear Unit)

\[ x : x \geq 0 \]
\[ 0.01x : x < 0 \]

- ReLU, but allows some small gradient for negative values
Setting Up A Neural Net

Input  Hidden  Output

\[ h_1 \]
\[ h_2 \]
\[ h_3 \]
\[ h_4 \]

\[ x_1 \]
\[ x_2 \]

\[ y_1 \]
\[ y_2 \]
\[ y_3 \]
Setting Up A Neural Net

Input  Hidden 1  Hidden 2  Output
Fully Connected Network

Each neuron connects to each neuron in the previous layer
How do we do all the neurons all at once?

\[ h_i = f(w_i^T a + b_i) \]
Fully Connected Network

\[ h = f(Wa + b) \]

- **a**: All layer a values
- **\(w_i, b_i\)**: Neuron i weights, bias
- **f**: Activation function
Fully Connected Network

Define New Block: “Linear Layer”
(Ok technically it’s ffine)

\[ L(n) = Wn + b \]

Can get gradient with respect to all the inputs
(do on your own; useful trick: have to be able
to do matrix multiply)
Fully Connected Network
What happens if we remove the activation functions?
Demo Time

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html