Statistics and Artificial Intelligence

Lecture 4: Logistic Regression as a Neural Network (II)

Yixin Wang

Goals for Today

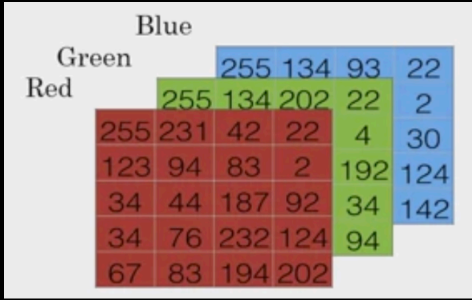
- Gradient descent
- Derivatives with a computational graph
- Logistic regression with gradient descent
- Deep Learning Frameworks: TensorFlow, Keras, Google Colab

next week: focus on TensorFlow

Binary Classification

Convert raw image into #'s computers can read

Logistic regression is for binary classification



- Input: Image X (Unroll pixel values into feature vector)
- Output: Label Y takes value 1 (cat) or 0 (non-cat)

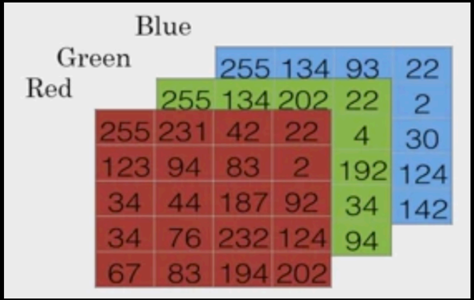
•Each image can be converted to a table
•Output label for each item in training data

Binary Classification

Convert images into vector

- $x \in \mathbb{R}^{n_x}, y \in \{0,1\}$

organize #s into column vector



$$x = \left.\begin{bmatrix}255\\231\\\vdots\\\vdots\\255\\134\\\vdots\\ \\ \end{bmatrix}\right\} n_x$$

1 (cat) vs 0 (non cat)

binary labels

Notation

- $(x, y), x \in \mathbb{R}^{n_x}, y \in \{0,1\}$ Paired Observation
  image + label
- m training examples: $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ → m training examples
- # training examples: $m = m_{\text{train}}$; similarly, # test examples: m_{test}
- $x \in \mathbb{R}^{n_x \times m}$, x.shape = (n_x, m) convert each image into a col.
- $y \in \mathbb{R}^{1 \times m}$, y.shape = $(1, m)$

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \quad \updownarrow n_x \text{ rows}$$

$\longleftarrow m \longrightarrow$ columns

$$Y = [y^{(1)} \; y^{(2)} \; \cdots, y^{(m)}]$$

one column for each example
- every image has nx dimensions
- have nx rows

Notation

- $(x, y), x \in \mathbb{R}^{n_x}, y \in \{0,1\}$
- m training examples: $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$
- # training examples: $m = m_{\text{train}}$; similarly, # test examples: m_{test}
- $x \in \mathbb{R}^{n_x \times m}$, x.shape = (n_x, m)

  Python

- $y \in \mathbb{R}^{1 \times m}$, y.shape = $(1, m)$

y vector collects label
- 1 row, m columns
- each label is just a scalar

$$\begin{array}{cc} X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} & \updownarrow n_x \\ \xleftrightarrow{\quad m \quad} & \end{array}$$

$$Y = [y^{(1)} \; y^{(2)} \; \cdots, y^{(m)}]$$

Logistic Regression

Model designed for binary classification

- Given x, want $\hat{y} = P(y = 1 \mid x)$,

$\hat{y}$ is the probability its labelled 1, given the image

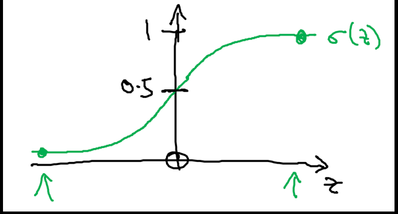
- $x \in \mathbb{R}^{n_x}$, $0 \le \hat{y} \le 1$
- Parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

Find values of w + b that make $\hat{y} \to y$

sigmoid

- Output: $\hat{y} = \sigma(\sum_{j=1}^{n_x} w_j x_j + b) = \sigma(w^\top x + b)$

sum weight for every #, add bias



sigmoid takes ↑ # to 1, small # to 0

- $\sigma(z) = \dfrac{1}{1 + e^{-z}}$
- If z large, $\sigma(z) \approx \dfrac{1}{1 + 0} = 1$; if z large negative number, $\sigma(z) \approx \dfrac{1}{1 + \text{big-number}} \approx 0$

sigmoid

How do we measure if $\hat{y}$ is close to 1?

Logistic Regression

Scalar and matrix version

$\hat{y}$ very close to $y \Rightarrow$ loss function is small

$\hat{y}$ far from $y \Rightarrow$ loss function very large

- continue to minimize

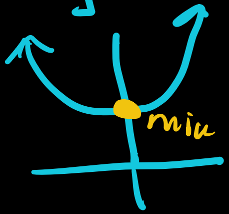
- d-dimensional input
- $\hat{y}^{(i)} = \sigma(\sum_{j=1}^{n_x} w_j x_j^{(i)} + b) = \sigma(w^\top \mathbf{x}^{(i)} + b)$ where $\sigma(z^{(i)}) = \dfrac{1}{1 + e^{-z^{(i)}}}$

convex = easier to minimize

- Given $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$
- Loss function / error function: $L(\hat{y}, y) = -(y \log \hat{y} + (1 - y)\log(1 - \hat{y}))$ (convex)

  cross entropy loss
- Cost function:

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$$



convex

look at predictions vs labels across whole dataset

Poll

- What are the parameters of logistic regression?
  - W, an n_x dimensional vector, and b, a real number.
  - W, an identity vector, and b, a real number.
  - W and b, both real numbers.
  - W and b, both n_x dimensional vectors.

W $\Rightarrow$ vector of weights

Logistic Regression Cost Function

Cost function / Loss function

- Loss function measures how good the prediction is, relative to the true label.

- Loss function / error function (defined for a **single** training example):
$L(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$

During training we minimize the cost function

- Training: Make the cost function as small as possible

- Cost function (defined for the **whole** training set): cost for parameters:

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m} (y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)}))$$

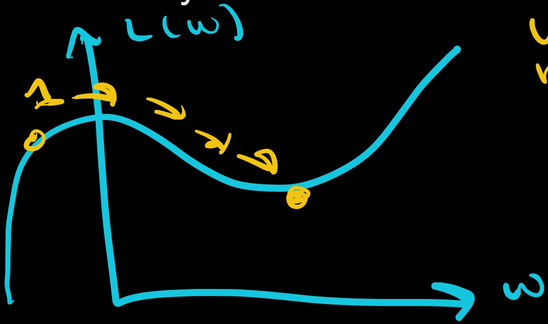
Generally can't move up in gradient descent

Logistic Regression Cost Function

Cost function / Loss function

- Loss function measures how good the prediction is, relative to the true label.
- Loss function / error function (defined for a **single** training example):
  $L(\hat{y}, y) = -\left(y \log \hat{y} + (1 - y)\log(1 - \hat{y})\right)$

- Why this loss function? Convexity!



we start somewhere + try to move towards minimum
- Have to decide when we get to min
- easier w/ convex shape
- don't want to end in local minima

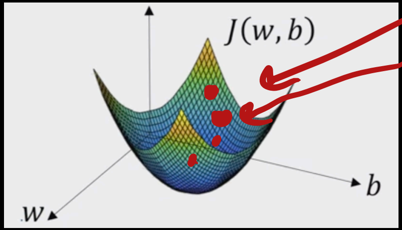
Gradient Descent for Training Neural Network

Logistic regression

Find w + b such that cost function is minimized

- Recap: $\hat{y}^{(i)} = \sigma(\sum_{j=1}^{n_x} w_j x_j^{(i)} + b) = \sigma(w^\top \mathbf{x}^{(i)} + b)$ where $\sigma(z^{(i)}) = \dfrac{1}{1 + e^{-z^{(i)}}}$

- $\underbrace{J(w, b)}_{\text{cost}} = \dfrac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\dfrac{1}{m} \sum_{i=1}^{m} (y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$

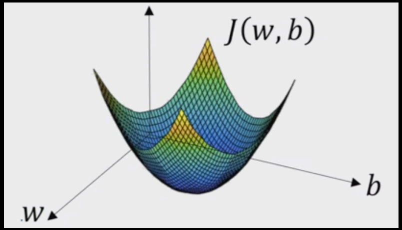


initialize anywhere in space
·calculate gradient + go down
Follow as we move towards a flat gradient
end at global minimum

https://math.stackexchange.com/questions/1582452/logistic-regression-prove-that-the-cost-function-is-convex

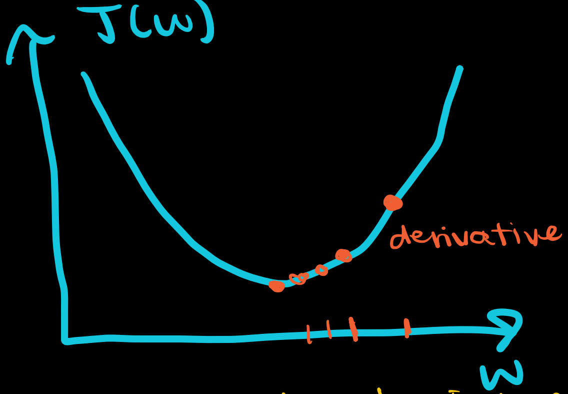
Gradient Descent for Training Neural Network

- Gradient Descent:
  - Initialize; figure this out based on the derivative
  - Take a step in the steepest downhill direction at each iteration
  - Repeat until the algorithm converges
    - get to global min



https://math.stackexchange.com/questions/1582452/logistic-regression-prove-that-the-cost-function-is-convex

Gradient Descent



set w to some # (0 or random)

Repeat

$$w \leftarrow w - \alpha \frac{dJ(w)}{dw}$$

learning rate

converge

$$\frac{dJ(w)}{dw}$$

calculate derivative at each point

→ move new point

→ calculate new point

we just trace the curve downward

learning rate decides how much we move at each step

$J(w,B)$

$$w \leftarrow w - \alpha \frac{dJ(w,b)}{dw}$$

$$b \leftarrow b - \alpha \frac{dJ(w,b)}{db}$$

$$\frac{\partial J(w,b)}{\partial w}$$ → dw

Partial Notation, means same thing

$$\frac{\partial J(w,b)}{\partial b}$$ → db

Gradient Descent

Almost all training algorithms are variants of gradient descent

- Learning rate: Control how big a step we take at each iteration

  How we trace curve
- Derivative: Slope of the function; the direction to go downhill
- Code: Often use 'dw' to represent the derivative dJ(w)/dw
  - The amount you want to update for w

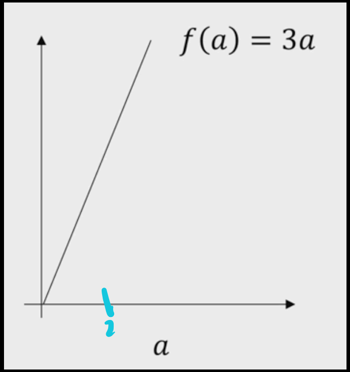
Question

- True or false. A convex function always has multiple local optima.

Intuition about Derivatives

$f(a) = 3a$

- Slope / Derivative: If I nudge a by a tiny little bit, I expect $f(a)$ to move three times as large as the nudge I gave a.

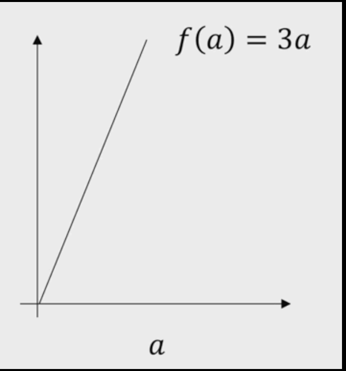


$$\frac{df(a)}{da} = 3$$

Intuition about Derivatives

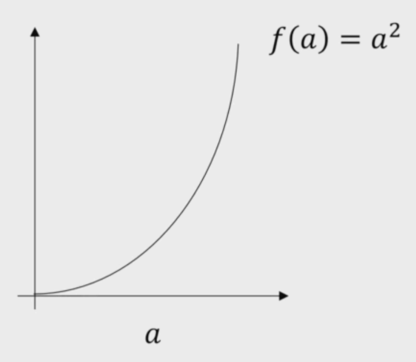
$f(a) = 3a$

- Formal definition through infinitesimal nudge.
- This function $f(a) = 3a$ has constant slope.



Intuition about Derivatives

$f(a) = a^2$



More Derivative Examples

How does the value change if we nudge the function

Computation Graph

- The computations of a neural network are organized in terms of
  - a forward pass or a forward propagation step, in which we compute the output of the neural network,
  - followed by a backward pass or back propagation step, which we use to compute gradients or compute derivatives.
- The computation graph explains why it is organized this way.

Computational Graph

Left-to-right pass

$$J(a,b,c) = \underbrace{3(\underbrace{a+\underbrace{bc}_{u})}_{v}}_{J}$$

$a=5$
$b=3$
$c=2$

$J(a,b,c) = 33$

computers approach this calculation more "step by step"

$$\left.\begin{aligned} u &= bc \\ v &= a+u \\ J &= 3 \cdot v \end{aligned}\right\} \text{computational graph}$$

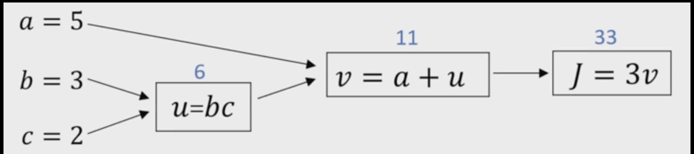
every neural net has a computational graph

$$\begin{tikzcd}
a=5 \arrow[rrd] & & & \\
b=3 \arrow[r] & \overset{6}{\boxed{u=bc}} \arrow[r] & \overset{11}{\boxed{v=a+u}} \arrow[r] & \overset{33}{\boxed{J=3v}} \\
c=2 \arrow[ru] & & &
\end{tikzcd}$$

Computing Derivatives with a Computation Graph

Right-to-left pass (Chain rule) $a \to v \to J$

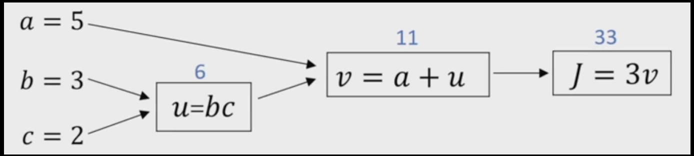
- Backpropagation $\dfrac{\mathrm{d}J}{\mathrm{d}a}$



Computing Derivatives with a Computation Graph

Right-to-left pass (Chain rule) $b \to J, c \to J$

- Backpropagation: $\dfrac{\mathrm{d}J}{\mathrm{d}b}$, and $\dfrac{\mathrm{d}J}{\mathrm{d}c}$



Questions?

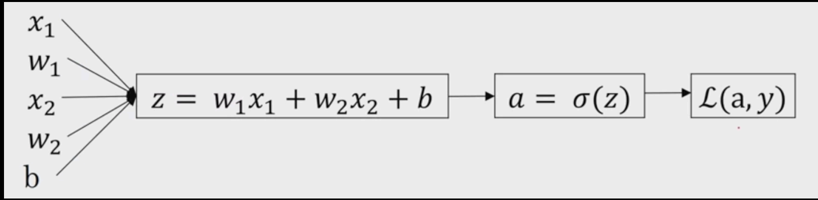
Logistic Regression Gradient Descent

Logistic regression recap

- $z = \sum_{j}^{n_x} w_j x_j + b, w^\top x + b$
- $\hat{y} = a = \sigma(z)$
- $L(a, y) = -(y \log(a) + (1 - y)\log(1 - a))$

Logistic Regression Gradient Descent

Logistic regression derivatives



Logistic Regression Gradient Descent

Logistic regression on m examples

- $J(w, b) = \dfrac{1}{m}\sum_{i=1}^{m} L(a^{(i)}, y^{(i)})$
- $a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)} = \sigma(w^\top x^{(i)} + b)$

Logistic Regression Gradient Descent

Logistic regression on m examples

$J=0;\ dw_1=0;\ dw_2=0;\ db=0$

For $i=1$ to m

$$z^{(i)} = w^T x^{(i)} + b$$

$$a^{(i)} = \sigma(z^{(i)})$$

$$J \mathrel{+}= -\left[y^{(i)} \log a^{(i)} + (1-y^{(i)}) \log(1-a^{(i)})\right]$$

$$dz^{(i)} = a^{(i)} - y^{(i)}$$

$$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$$

$$dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}$$

$$db \mathrel{+}= dz^{(i)}$$

$n=2$

dw_3
$\vdots$
dw_n

Questions?

Deep Learning Frameworks

- Caffe/Caffe2
- CNTK
- DL4J
- Keras
- Lasagne
- mxnet
- PaddlePaddle
- TensorFlow
- Theano
- Torch

Deep Learning Frameworks

- Choosing deep learning frameworks
- - Easeofprogramming(development and deployment)
- - Runningspeed
- - Truly open (open source with good governance)

The materials in this course are adapted from materials created by Alexander Amini, Alfredo Canziani, Justin Johnson, Andrew Ng, Bhiksha Raj, Grant Sanderson and the 3blue1brown channel, Rita Singh, Ava Soleimany, and Ambuj Tewari.