Logistic Regression in Neural Networks and Python

Sidharth Baskaran

July 2021

Binary Classification

- Output is either 0 or 1
- Notation
  - Single training example $\rightarrow (x, y)$
  - Feature vector $x \in \mathbb{R}^{n_x}$ and output $y \in \{0, 1\}$ where n_x is number of features
  - Have m training examples $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$
  - The matrix X is defined as $X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$, and is $n_x \times m$ in dimension (`X.shape=(nx,m)`)
  - Define $Y = \begin{bmatrix} y^{(1)} & \cdots & y^{(m)} \end{bmatrix}$ where `Y.shape=(1,m)`

Logistic Regression

- Used in binary classification problems
- Given x, want $\hat{y} = P(y = 1|x)$ where $x \in \mathbb{R}^{n_x}$ with $0 \leq \hat{y} \leq 1$
- Parameters are $w \in\in \mathbb{R}^{n_x}, b \in \mathbb{R}$

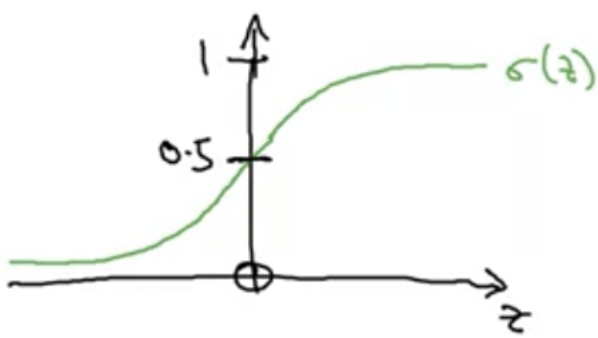

Figure 1: Sigmoid function

- Output is then $\hat{y} = \sigma(w^T + b)$
  - $\sigma(z) = \frac{1}{1+e^{-z}}$
- w is $theta_1 \rightarrow \theta_{n_x}$ and $b = \theta_0$ for notational correspondence

Logistic Regression Cost Function

- Ultimately want $\hat{y}^{(i)} \approx y^{(i)}$
- Define loss $\mathcal{L}(\hat{y}, y) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$ that is convex, for **single training example**

- If $y=1$, $\mathcal{L} = -\log(\hat{y} \rightarrow$ want $\hat{y}$ large (≈ 1)
  - If $y=0$, $\mathcal{L} = -\log(1-\hat{y})$, so want $\hat{y}$ small (≈ 0)
- Cost function tells how model does on **entire training set**
  - $J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Gradient Descent

- Method to find w, b for $\min(J(w,b))$, which is convex
  - No local minima

Repeat {

$$w := w - \alpha\frac{dJ(w)}{dw} b := b - \alpha\frac{dJ(w,b)}{db}$$

}

In code, call deriv. `dw`.

- Signs work out, so subtraction is used (want to move **opposite** to direction of function in order to converge to minima)

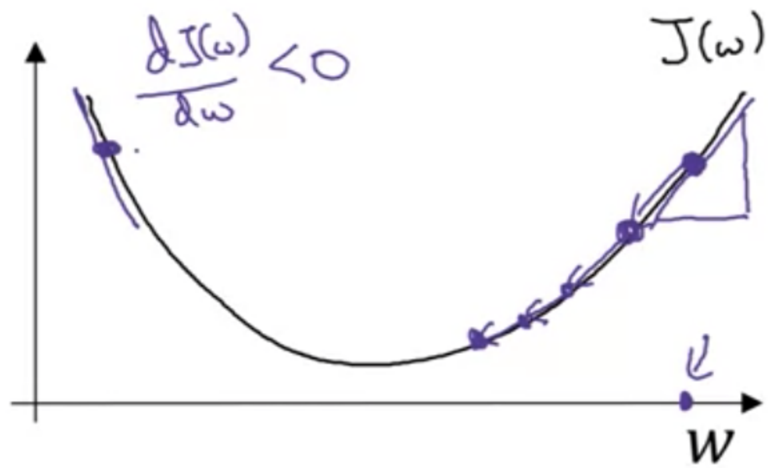

Figure 2: Gradient descent visual

- Would use partial derivative if $\text{arglen}(J) > 1$

Computation Graphs

- Example: Let $J(a,b,c) = 3(a+bc)$
  - Then, $u = bc$, and $v = a + u$, so $J = 3v$
- Drawing the computation graph
- Derivatives are a right $\rightarrow$ left computation, and in this case it is left $\rightarrow$ right

Derivatives with Computation Graphs

- Derivative of last step of graph is one step backwards in **backpropagation**
- Can take derivative of J with respect to any variable $\rightarrow$ chain rule
  - E.g. $\frac{dJ}{dv}\frac{dv}{da} = \frac{dJ}{da}$
- Many computations involve derivative of final variable (i.e. J) wrt arbitrary intermediate variable

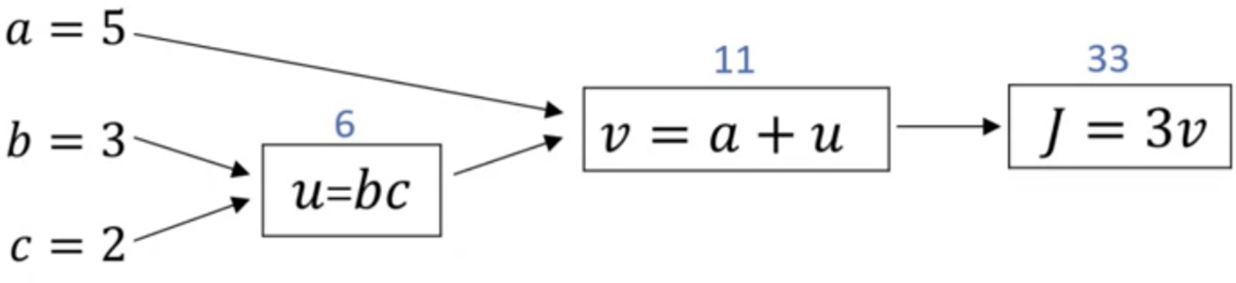

Figure 3: Computation graph

- Convention `dvar` means derivative of final output variable wrt some intermediate quantity

Logistic Regression Gradient Descent

Recall

$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z)$$
$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

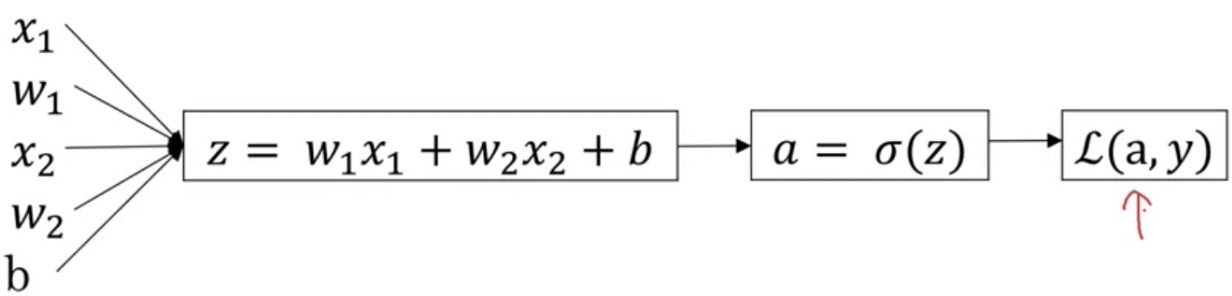

Figure 4: Gradient descent computation graph

- First step $da = \frac{d\mathcal{L}(a,y)}{da}$ in backpropagation
- Can show $dz = \frac{d\mathcal{L}}{dz}$