Lecture 23: Gradient Descent, Forward and Backward Propagation

BIOS635

04/14/2020

Class notes for online teaching due to COVID-19.

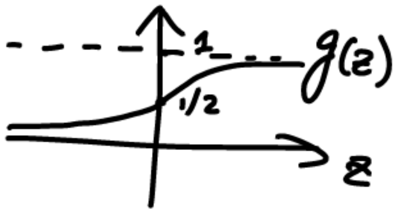
Review of logistic regression

Given $x \in \mathbb{R}^p$, want $\hat{y} = P(y=1|x)$ $\quad 0 \le \hat{y} \le 1$

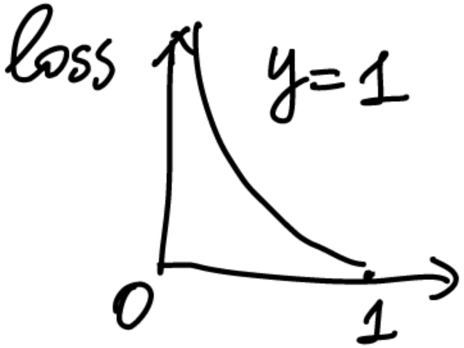
parameters $w \in \mathbb{R}^p$, $b \in \mathbb{R}$

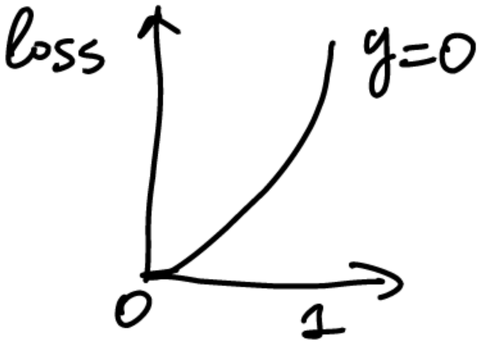
Output $\hat{y} = g(\underbrace{w^T x + b}_{z})$

$$g(z) = \frac{1}{1+e^{-z}}$$



Loss function : $\mathcal{L}(\hat{y}, y) = \begin{cases} -\log(\hat{y}) & \text{if } y=1 \\ -\log(1-\hat{y}) & \text{if } y=0 \end{cases}$





L_2 norm is non-convex and is hard to optimize

$$\mathcal{L}(\hat{y}, y) = -\underbrace{y}_{\text{Actual}} \log \underbrace{\hat{y}}_{\text{predicted}} - \underbrace{(1-y)}_{\text{Actual}} \log \underbrace{(1-\hat{y})}_{\text{Predicted}}$$

"binary Loss"

Cost function: $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log \hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$$

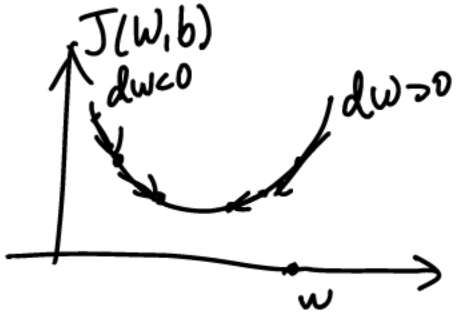
Want to find w and b that minimizes $J(w,b)$.

Use gradient descent!

$$w := w - \alpha\, dw = w - \alpha \boxed{\frac{\partial J(w,b)}{\partial w}} \rightarrow dw$$

$$b := b - \alpha\, db = b - \alpha \boxed{\frac{\partial J(w,b)}{\partial b}} \rightarrow db$$

"learning rate"



Logistic regression gradient descent

$$\left.\begin{matrix} x \\ w \\ b \end{matrix}\right\} \rightarrow \boxed{z = w^T x + b} \rightarrow \boxed{a = g(z)} \rightarrow \boxed{\mathcal{L}(a, y)}$$

$$a = g(z) = \frac{1}{1+e^{-z}} \qquad \mathcal{L}(a,y) = -y\log a - (1-y)\log(1-a)$$

non-linear transformation Loss function

$$\text{"}da\text{"} = \frac{d\mathcal{L}}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$$

$$\text{"}dz\text{"} = \frac{dL}{dz} = \frac{dL}{da} \cdot \frac{da}{dz} = \left(-\frac{y}{a} + \frac{1-y}{1-a}\right)\left(\frac{1}{(1+e^{-z})^2} e^{-z}\right)$$

$$= \left(-\frac{y}{a} + \frac{1-y}{1-a}\right) a(1-a)$$

$$= a - y$$

$$\text{"}dw\text{"} = \frac{dL}{dw} = \frac{dL}{dz} \cdot \frac{dz}{dw} = dz\, x$$

$$\text{"}db\text{"} = \frac{dL}{db} = \frac{dL}{dz} \cdot \frac{dz}{db} = dz.$$

On m samples,

$$\frac{\partial J(w,b)}{\partial w} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial w}\mathcal{L}(a^{(i)}, y^{(i)})$$

$$dz = a - y$$
$$dw = \frac{1}{m} X dz^T$$
$$db = \frac{1}{m}\,\text{np.sum}(dz)$$

$y \in \mathbb{R}^{1\times m}$, $a \in \mathbb{R}^{1\times m}$, $z \in \mathbb{R}^{1\times m}$
$X \in \mathbb{R}^{p\times m}$, $W \in \mathbb{R}^{p}$, $b \in \mathbb{R}$

Putting together,

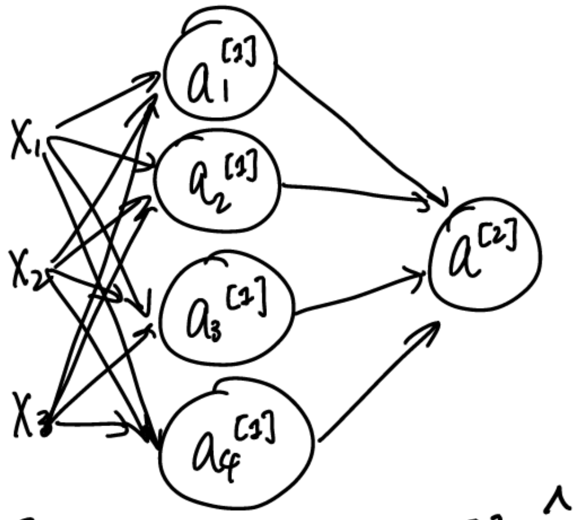
FORWARD propogation
$$Z = W^T X + b$$
$$A = g(Z)$$

BACKWARD propagation
$$dz = A - Y$$
$$dw = \frac{1}{m} X dz^T$$
$$db = \frac{1}{m}\,\text{np.sum}(dz)$$

$$w := w - \alpha dw$$
$$b := b - \alpha db$$

iterate till convergence

Perceptron: one hidden-layer neural network



$a^{[0]} = X$ $\quad a^{[1]} \quad$ $a^{[2]} = \hat{y}$

Input layer Hidden layer Output layer

$\boxed{\text{FORWARD PROPAGATION}}$

$$\overset{n[1]\times m}{Z^{[1]}} = \overset{n[1]\times n[0]}{W^{[1]}} \, \overset{n[0]\times m}{A^{[0]}} + \overset{n[1]\times 1}{b^{[1]}}$$

$$\overset{n[1]\times m}{A^{[1]}} = g^{[1]}\left(\overset{n[1]\times m}{Z^{[1]}}\right)$$

$$\overset{n[2]\times m}{Z^{[2]}} = \overset{n[2]\times n[1]}{W^{[2]}} \, \overset{n[1]\times m}{A^{[1]}} + \overset{n[2]\times 1}{b^{[2]}}$$

$$\overset{n[2]\times m}{A^{[2]}} = g^{[2]}\left(\overset{n[2]\times m}{Z^{[2]}}\right)$$

First layer

First node — node 1

$$z^{[1]}_1 = w^{[1]}_{11} \cdot x_1 + w^{[1]}_{12} x_2 + w^{[1]}_{13} x_3 + b^{[1]}_1$$

second node — node 1

$$z^{[1]}_2 = w^{[1]}_{21} x_1 + w^{[1]}_{22} x_2 + w^{[1]}_{23} x_3 + b^{[1]}_2$$

$\vdots$

Final node

$$z^{[1]}_4 = w^{[1]}_{41} x_1 + w^{[1]}_{42} x_2 + w^{[1]}_{43} x_3 + b^{[1]}_4$$

- parameters:

$W^{[1]} \in \mathbb{R}^{n[1]\times n[0]}$, $b^{[1]} \in \mathbb{R}^{n[1]}$

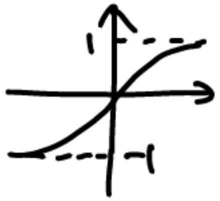
$W^{[2]} \in \mathbb{R}^{n[2]\times n[1]}$, $b^{[2]} \in \mathbb{R}^{n[2]}$

- Activation functions (all nonlinear)

(i) sigmoid $g(z) = \frac{1}{1+e^{-z}}$

$g'(z) = a(1-a)$

(ii) hyperbolic tangent $g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$



$g'(z) = 1-a^{2}$

(iii) Rectified linear unit (ReLU)

$g(z) = \max(0, z)$

$g'(z) = \begin{cases} 0, & z<0 \\ 1, & z \geq 0 \end{cases}$

BACKWARD PROPAGATION

Cost function: $J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\overset{\substack{a^{[2](i)}\\ \uparrow}}{\hat{y}^{(i)}}, y^{(i)})$

Repeat $\{$ Compute prediction $(\hat{y}^{(i)}, i=1,\cdots,m)$

$dW^{[1]} = \frac{\partial J}{\partial W^{[1]}}, \quad db^{[1]} = \frac{\partial J}{\partial b^{[1]}}, \quad \cdots$

$W^{[1]} = W^{[1]} - \alpha dW^{[1]}, \quad \cdots$

$b^{[1]} = b^{[1]} - \alpha db^{[1]}, \quad \cdots \quad \}$

$$\overset{n^{[2]}\times m}{dZ^{[2]}} = \overset{n^{[2]}\times m}{A^{[2]}} - \overset{n^{[2]}\times m}{Y}$$

$$\overset{n^{[2]}\times n^{[1]}}{dW^{[2]}} = \frac{1}{m} \overset{n^{[2]}\times m}{dZ^{[2]}} \overset{n^{[1]}\times m}{A^{[1]T}}$$

$$\overset{n^{[2]}\times 1}{db^{[2]}} = \frac{1}{m} \text{np.sum}(\overset{n^{[2]}\times m}{dZ^{[2]}}, \text{axis}=1, \text{keepdims}=\text{True})$$

$$\overset{n^{[1]}\times m}{dZ^{[1]}} = \overset{n^{[1]}\times m}{dA^{[1]}} * \overset{n^{[1]}\times m}{g^{[1]\prime}(Z^{[1]})} = \overset{n^{[1]}\times n^{[2]}}{W^{[2]T}} \overset{n^{[2]}\times m}{dZ^{[2]}} * \overset{n^{[1]}\times m}{g^{[1]\prime}(Z^{[1]})}$$

$$\overset{n^{[1]}\times n^{[0]}}{dW^{[1]}} = \frac{1}{m} \overset{n^{[1]}\times m}{dZ^{[1]}} \overset{m\times n^{[0]}}{X^{T}}$$

$$\overset{n^{[1]}\times 1}{db^{[1]}} = \frac{1}{m} \text{np.sum}(\overset{n^{[1]}\times m}{dZ^{[1]}}, \text{axis}=1, \text{keepdims}=\text{True})$$

Deep neural network

FORWARD

$$Z^{[1]} = W^{[1]}A^{[0]} + b^{[1]}$$

$$A^{[1]} = g^{[1]}(Z^{[1]})$$

$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

$$A^{[2]} = g^{[2]}(Z^{[2]})$$

$$\vdots$$

$$A^{L} = g^{(L)}(Z^{[L]}) = \hat{Y}$$

BACKWARD

$$dZ^{[L]} = A^{[L]} - Y$$

$$dW^{[L]} = \frac{1}{m} dZ^{[L]} A^{[L-1]T}$$

$$db^{[L]} = \frac{1}{m} np.sum(dZ^{[L]}, axis=1, keepdims=True)$$

$$dZ^{[L-1]} = W^{[L]T} dZ^{[L]} * g^{[L-1]'}(Z^{[L-1]})$$

$$\vdots$$

$$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]'}(Z^{[1]})$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} A^{[0]T}$$

$$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis=1, keepdims=True)$$

parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}, \dots$

hyperparameters: $\begin{cases} \text{learning rate } \alpha \\ \text{\# iterations} \\ \text{\# hidden layers } L \\ \text{\# hidden units } n^{[1]}, n^{[2]}, \dots \\ \text{choice of activation function} \\ \dots \end{cases}$

Regularization

$$J(W^{[1]}, b^{[1]}, \cdots, W^{[L]}, b^{[L]}) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \|W^{[l]}\|_F^2$$

$$W^{[l]} \in \mathbb{R}^{n^{[l]} \times n^{[l-1]}}, \quad \|W^{[l]}\|_F^2 = \sum_{i=1}^{n^{[l]}} \sum_{j=1}^{n^{[l-1]}} (W_{ij}^{[l]})^2$$

"Frobenius norm" (Euclidean norm of a matrix)

$$dW^{[l]} = (\text{from backprop}) + \frac{\lambda}{m} W^{[l]}$$

$$W^{[l]} := W^{[l]} - \alpha\left[(\text{from backprop}) + \frac{\lambda}{m} W^{[l]}\right]$$

$$= \underbrace{\left(1 - \frac{\alpha\lambda}{m}\right)}_{<1} W^{[l]} - \alpha(\text{from backprop})$$

<1: "weight decay" for L2 norm.

Other regularization methods

- Dropout : cannot rely on any one feature
- Early stopping :