1- Neural Networks and Deep Learning

Sunday, December 11, 2022 5:02 PM

Application:

- Real State
- Online Advertise
- Photo Tagging
- Speech Recognition
- Machine Translation
- Autonous Driving

Real State, Online Advertise → standard NN

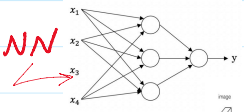
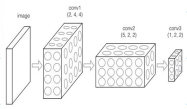
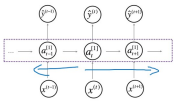


Photo Tagging → CNN



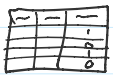
Speech Recognition, Machine Translation → RNN



Autonous Driving → Complex hybrid

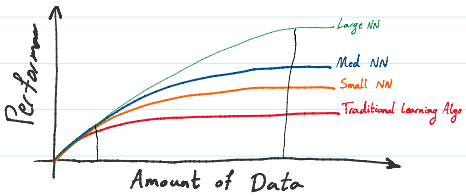
Supervised Learning:

- Structed Data →



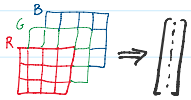
- Unstructed Data →
  - * Audio
  - * Image
  - * Text

Deep Learning Take_off:



Binary Classification:

- 1 (cat) vs 0 (non cat)



* Images : $64 \times 64 \times 3 \longrightarrow n_x = 12288$

(x, y) : $x \in \mathbb{R}^{n_x}$, $y \in \{0, 1\}$

m training example $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \quad (n_x \times m) \longrightarrow X \in \mathbb{R}^{n_x \times m} \qquad Y = [y^{(1)} y^{(2)} \cdots y^{(m)}] \longrightarrow Y \in \mathbb{R}^{1 \times m}$$

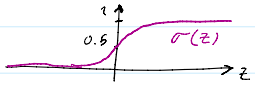
Logistic Regression:

given x, $\hat{y} = P(y=1 \mid x)$

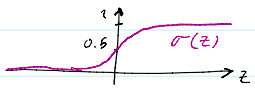
Parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

Output: $\hat{y} = \sigma(w^T x + b)$, $z = w^T x + b$

sigmoid $\longrightarrow \sigma(z) = \frac{1}{1+e^{-z}}$



sigmoid ⟶ $\sigma(z) = \frac{1}{1+e^{-z}}$



Cost Function:

Loss (error) function: $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y}-y)^2$

$\rightarrow \mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$

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

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

Gradient Descent:

- find w, b that minimize $J(w,b)$

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

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

learning rate ↙

LR GD:

$$\begin{cases} z = w^T x + b \\ \hat{y} = a = \sigma(z) \\ \mathcal{L}(a,y) = -(y \log a + (1-y)\log(1-a)) \end{cases} \Longrightarrow \begin{cases} dz = \frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial a} \times \frac{\partial a}{\partial z} = a - y \quad (a(1-a)) \\ da = \frac{\partial \mathcal{L}}{\partial a} = \frac{-y}{a} + \frac{1-y}{1-a} \\ dw = \frac{\partial \mathcal{L}}{\partial w} = x\, dz \end{cases}$$

$db = dz$

On m examples ⟶ $J(w,b) = \frac{1}{m}\sum_{1}^{m} \mathcal{L}(a^{(i)}, y^{(i)})$ $\quad * a^{(i)} = \sigma^{(i)}(w^T x^{(i)} + b)$

$dZ = [dz^{(1)} \dots dz^{(m)}]$

$A = [a^{(1)} \dots a^{(m)}] \quad Y = [y^{(1)} \dots y^{(m)}]$ } $dZ = A - Y$

$db = \frac{1}{m}\sum_{1}^{m} dz^{(i)}$

$dw = \frac{1}{m} X dZ^T = \frac{1}{m}\begin{bmatrix} | & & | \\ x^{(1)} & \dots & x^{(m)} \\ | & & | \end{bmatrix}\begin{bmatrix} dz^{(1)} \\ dz^{(m)} \end{bmatrix} = \frac{1}{m}\left[x^{(1)} dz^{(1)} \dots x^{(m)} dz^{(m)}\right]$

$w -= \alpha\, dw \quad b -= \alpha\, db$

Python Broadcasting :

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} + \begin{bmatrix}10 & 10 & 10\\ 10 & 10 & 10\end{bmatrix} = \begin{bmatrix}11 & 12 & 13\\ 14 & 15 & 16\end{bmatrix}$$

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} + \begin{bmatrix}10\\ 20\end{bmatrix} \begin{matrix}10 & 10\\ 10 & 10\end{matrix} = \begin{bmatrix}11 & 12 & 13\\ 24 & 25 & 26\end{bmatrix}$$

Notations:

$(i) \rightarrow i^{th}$ example
$[l] \rightarrow l^{th}$ layer

m: Num of examples
n_x: Input size
n_y: Num of classes

$n_h^{[l]}$: Num of hidden layers of l^{th}

$n_x = n_h^{[0]}$ $\quad n_y = n_h^{[L+1]}$

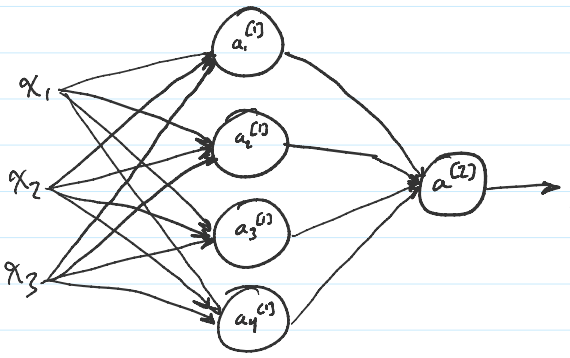
$X \in \mathbb{R}^{n_x \times m}$
$x^{(i)} \in \mathbb{R}^{n_x}$ column

$Y \in \mathbb{R}^{n_y \times m}$ lable matrix
$y^{(i)} \in \mathbb{R}^{n_y}$ output lable of i^{th}

$W^{[l]} \in \mathbb{R}^{next \times prev}$ weights
$b^{[l]} \in \mathbb{R}^{next}$ bias

$$a = g^{[l]}(W_x x^{(i)} + b_1) = g^{[l]}(z_1) \qquad \hat{y}^{(i)} = softmax(W_h h + b_2)$$

Neural Network :



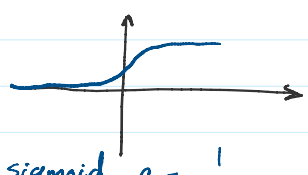
$$z^{[1]} = W^{[1]} x + b^{[1]} \quad (4,1)$$
$$a^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]} \quad (1,1)$$
$$a^{[2]} = \sigma(z^{[2]})$$

$$Z^{[1]} = W^{[1]} X + b^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$
$$A^{[2]} = \sigma(Z^{[2]})$$

m Examples

$X = a^{[0]}$ $\quad W^{[1]}$ $\quad b^{[1]}$ $\quad W^{[2]}$ $\quad b^{[2]}$
$(3,1)$ $\quad (4,3)$ $\quad (4,1)$ $\quad (1,4)$ $\quad (1,1)$

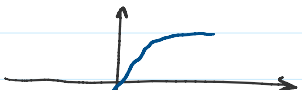
Activation Functions:



Sigmoid $a = \frac{1}{1+e^{-z}}$

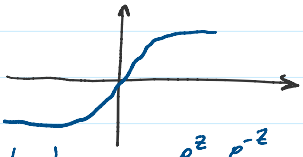
$g'(z) = a(1-a)$

* output layer when binary classification.



$g'(z) = 1 - \tanh^2(z)$

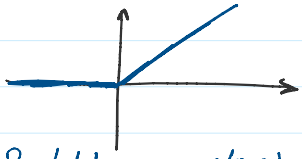
* almost always works better than sigmoid (mean = 0)



tanh $a = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

$g'(z) = 1 - \tanh^2(z)$

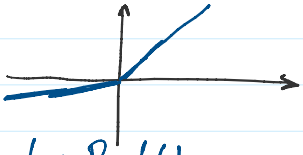
* almost always works better than sigmoid (mean = 0)



ReLU $a = \max(0, z)$

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

* defult option



Leaky ReLU $a = \max(0.01z, z)$

$g'(z) = \begin{cases} 0.01 & z < 0 \\ 1 & z \geq 1 \end{cases}$

Gradient Descent for NN:

forward:

$Z^{[1]} \rightarrow A^{[1]}$

$Z^{[2]} \rightarrow A^{[2]}$

backpropagation:

$dZ^{[2]} = A^{[2]} - Y^{[2]}$

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

$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis=1, keepdims=T)$ $(n^{[2]},) \rightarrow (n^{[2]}, 1)$

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

$(n^{[1]}, m) \quad (n^{[1]}, m)$

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

$db^{[1]} = \frac{1}{m} np.sum(\_\_)$

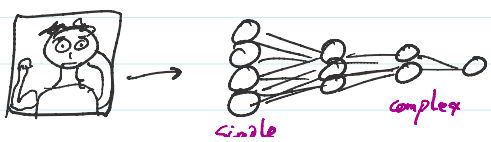
1 layer (Shallow) $\rightarrow$ 5 layers (Deep)

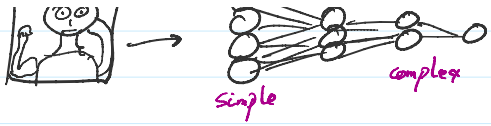
$W^{[l]} \quad (n^{[l]}, n^{[l-1]})$

$b^{[l]} \quad (n^{[l]}, 1)$

$dW^{[l]} \quad (n^{[l]}, n^{[l-1]})$

$db^{[l]} \quad (n^{[l]}, 1)$





* shallower networks require exponentionally more hidden units.