Deep Learning

Supervised Learning with non-linear models

before: Kernel Methods $h_\theta(x) = \theta^T \phi(x)$ non-linear in x
linear in θ

Other non-linear models
eg. $h_\theta(x) = \sqrt{\theta_1^3 x + \theta_3 x_4 + \sqrt{\theta_5 x_8}}$

dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^n$ $x^{(i)} \in \mathbb{R}^d$, $y^{(i)} \in \mathbb{R}$
$h_\theta(x) : \mathbb{R}^d \rightarrow \mathbb{R}$

Cost / Loss f^n
$J^{(i)}(\theta) = (y^{(i)} - h_\theta(x^{(i)}))^2$ mean-squared loss

Cost f^n for entire dataset
$J(\theta) = \frac{1}{n} \sum_{i=1}^n J^{(i)}(\theta)$

Optimization Objective
$\min_\theta J(\theta)$

Gradient Descent
$\theta := \theta - \alpha \nabla J(\theta)$

Stochastic Gradient Descent (SGD)
for $i = 1$ to n_{iter}
Sample j from $\{1 - - n\}$ uniformly
$\theta := \theta - \alpha \nabla J^{(j)}(\theta)$

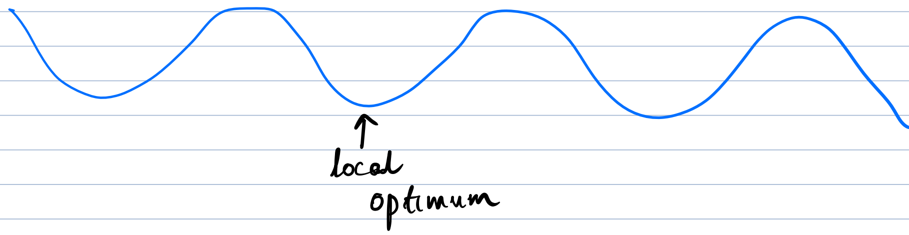
Mini-batch SGD
- Computing B gradients $\nabla^{(j_1)}(\theta) \ldots \nabla J^{(j_B)}(\theta)$ together is faster than individual compution

Mini-batch SGD

for $i := 1$ to n_{iter}

Sample B examples $\{j_1 \dots j_B\}$ from $\{1 \dots n\}$ without replacement

$$\theta := \theta - \frac{\alpha}{B} \sum_{k=1}^{B} \nabla J^{(j_k)}(\theta)$$



remaining questions

① how to define $h_\theta(x)$

② how to compute gradient

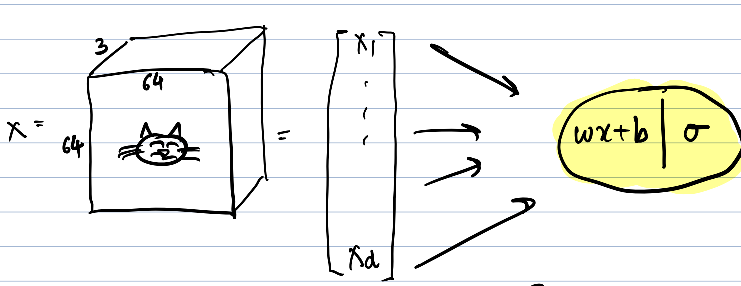
- Logistic Regression
- Neural Networks
  - computational power
  - data available
  - algorithms

① Logistic Regression

goal: Find cats in images

$1 \rightarrow$ presence of cat

$0 \rightarrow$ absence of cat



$d = 64 \times 64 \times 3 = 12,288$

$$\hat{y} = \sigma(\theta^T x)$$

$$= \sigma(wx + b)$$

$1 \times 12,288$ $\quad$ $12,288 \times 1$

i) Initialize w, b

$w \downarrow$ weights, $b \downarrow$ bias

ii) Find optimal w, b

iii) Use $\hat{y} = \sigma(wx + b)$ to predict

$$\mathcal{L} = -\left[y \log \hat{y} + (1-y)\log(1-\hat{y})\right]$$

$$w := w - \alpha \frac{\partial \mathcal{L}}{\partial w}$$

$$b := b - \alpha \frac{\partial \mathcal{L}}{\partial b}$$

$$\sigma' = \sigma(1-\sigma)$$

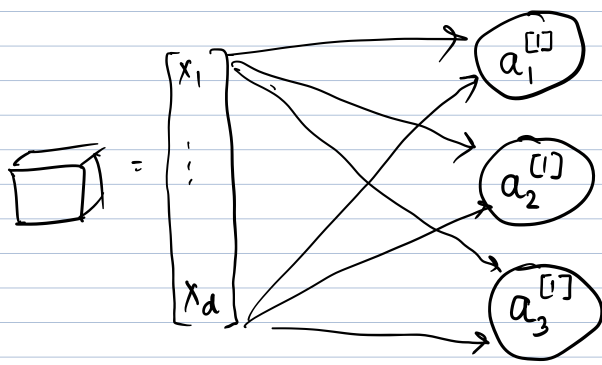
\# parameters = 12,288 + L

neuron = linear + activation

model = architecture + parameters

1 neuron — w, b

Goal 2.0 : Find cat / lion / iguana in images



$$\hat{y}_1 = a_1^{[1]} = \sigma(w_1^{[1]} x + b_1^{[1]})$$

$$\hat{y}_2 = a_2^{[1]} = \sigma(w_2^{[1]} x + b_2^{[1]})$$

$$\hat{y}_3 = a_3^{[1]} = \sigma(w_3^{[1]} x + b_3^{[1]})$$

square brackets $[1] \equiv$ layer

Subscripts $\equiv$ identify neuron within layer

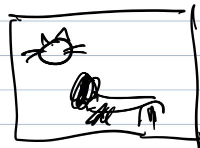
$a_2^{[1]}$ — $[1] \leftarrow$ 1st layer; $2 \longleftrightarrow$ 2nd neuron in 1st layer

\# param = $3(d+1)$

Dataset: images + labels

$\begin{bmatrix}1\\0\\0\end{bmatrix}$ cat, lion, iguana $\begin{bmatrix}0\\1\\0\end{bmatrix}$ lion $\begin{bmatrix}0\\0\\1\end{bmatrix}$ iguana

one hot encoding



$$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \text{cat} \\ \text{lion} \\ \end{matrix}$$

Goal 3.0: add constraint : unique animal in image

$$\square = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

$\cancel{a_1^{[1]}}$ $\quad z_1^{[1]} \quad e^{z_1^{[1]}} / \sum_{k=1}^{3} e^{z_k^{[1]}}$

$\cancel{a_2^{[1]}}$ $\quad z_2^{[1]} \quad e^{z_2^{[1]}} / \sum_{k=1}^{3} e^{z_k^{[1]}}$

$\cancel{a_3^{[1]}}$ $\quad z_3^{[1]} \quad e^{z_3^{[1]}} / \sum_{k=1}^{3} e^{z_k^{[1]}}$

$$\hat{y}_1 = \sigma(\underbrace{w_1^{[1]} x + b_1^{[1]}}_{z_1^{[1]}})$$

Softmax

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$\#$ parameters $= 3(d+1)$

How to train?

Loss function

$$\mathcal{L}_{3N} = -\sum_{k=1}^{3} \left[y_k \log \hat{y}_k + (1-y_k) \log (1-\hat{y}_k) \right]$$

$$\mathcal{L}_{CE} = -\sum_{k=1}^{3} y_k \log \hat{y}_k$$

$\uparrow$ Cross entropy

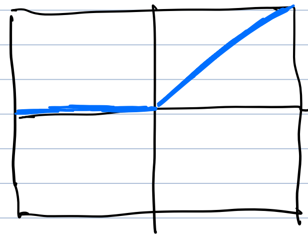
Q^n : Instead of predicting 1/0, predict age of cat?

Options:

① Bucket ages, several neurons to predict

② Change activation f^n

linear f^n : $f(x) = x$ linear regression



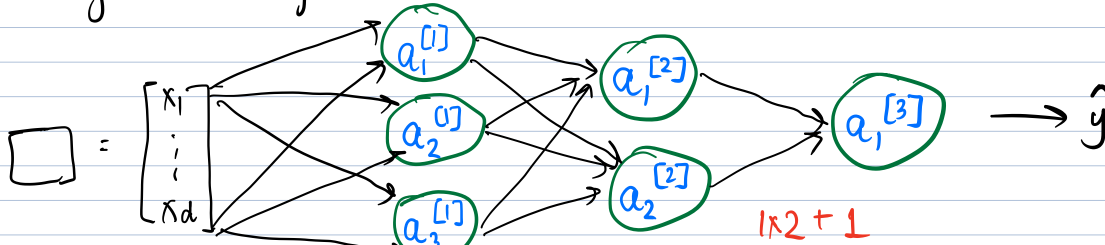
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \le 0 \end{cases}$$

ReLU
Rectified Linear Unit

Modified loss f^n $\quad \|\hat{y} - y\|_1 \quad \|\hat{y} - y\|_2^2$

② Neural Networks

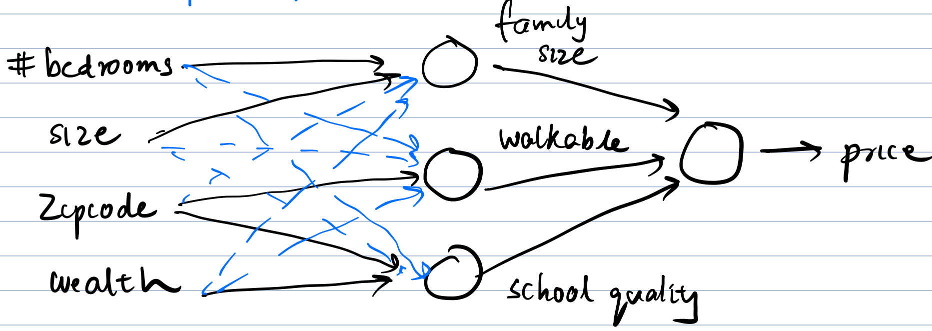
goal : Image ⇒ cat vs. no cat



3d+3 $\quad$ 2x3+2

input layer $\quad$ output layer
hidden layer

House price prediction



Propagation equation

$$\underset{3\times 1}{z^{[1]}} = \underset{3\times d}{w^{[1]}} \underset{d\times 1}{x} + \underset{3\times 1}{b^{[1]}}$$

$$\underset{3\times n}{Z^{[1]}} = \underset{3\times d}{w^{[1]}} \underset{d\times n}{X} + \underset{3\times 1}{b^{[1]}}$$

$$\underset{3\times 1}{a^{[1]}} = \sigma(z^{[1]})$$

Broadcasting

$$\tilde{b}^{[1]} = \Big[\underbrace{b^{[1]} \; \cdots \; b^{[1]}}_{n \text{ copies}}\Big]$$

$$\underset{2\times 1}{z^{[2]}} = \underset{2\times 3}{w^{[2]}} \underset{3\times 1}{a^{[1]}} + \underset{2\times 1}{b^{[2]}}$$

$$\underset{2\times 1}{a^{[2]}} = \sigma(z^{[2]})$$

added to fix omission in lecture

$$\underset{1\times 1}{z^{[3]}} = \underset{1\times 2}{w^{[3]}} \underset{2\times 1}{a^{[2]}} + \underset{1\times 1}{b^{[3]}}$$

$$a^{[3]} = \sigma(z^{[3]})$$

$$X = \begin{bmatrix} x^{(1)} & \cdots & x^{(n)} \end{bmatrix}$$

[ ] — layer

( ) — id of example

capital : batch of examples.

Optimize $w^{[1]}$ $w^{[2]}$ $w^{[3]}$ $b^{[1]}$ $b^{[2]}$ $b^{[3]}$

Define Loss / Cost fn

Backward Propagation