Deep Learning

Supervised learning w/ non-linear models

before kernel method $h_\theta(x) = \theta^T \phi(x)$ non-linear in x, linear θ

eg. $h_\theta(x) = \sqrt{\theta_1^3 + \theta_2 x_3 + \sqrt{\theta_4 x_7}}$

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

$h_\theta(x) : \mathbb{R}^d \to \mathbb{R}$

cost/loss fn

$J^{(i)}(\theta) = (y^{(i)} - h_\theta(x^{(i)}))^2$ squared loss

Cost fn 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 $\{1 \dots n\}$ randomly

$\theta := \theta - \alpha \nabla J^{(j)}(\theta)$

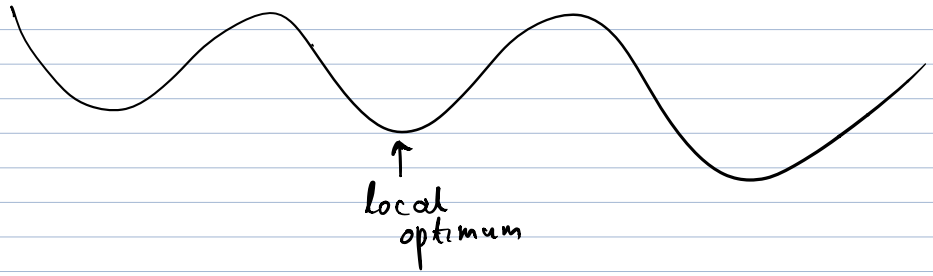
minibatch SGD

Idea: Computing B gradients $\nabla J^{(j_1)}(\theta) \dots \nabla J^{(j_B)}(\theta)$ together is faster than sequential computations

for $i = 1$ to n_{iter}

Sample B examples $\{j_1 \dots j_B\}$ from $\{1 \dots n\}$

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



remaining questions

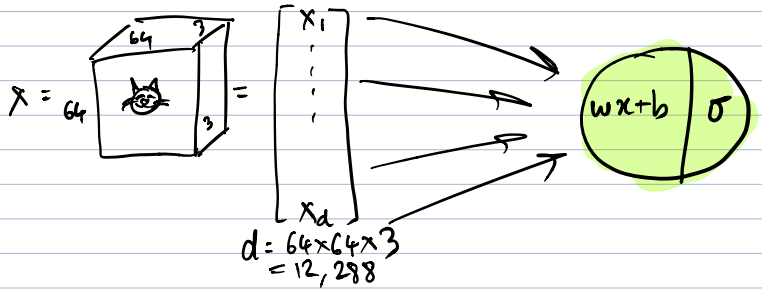
① How to define $h_\theta(x)$?

② how to compute gradient

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

① Logistic Regression

goal Find cats in images 1 → presence of cat, 0 → absence of cat



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

$= \sigma(wx + b)$

1×12288 ↙ ↘ 12288×1

weights ↑ bias ↓

(i) initialize w, b

(ii) Find optimal w, b

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

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

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

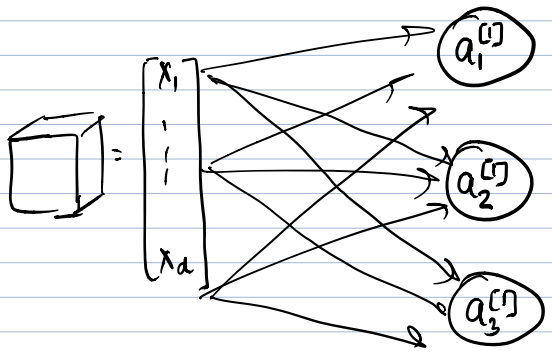
$$b = b - \alpha \frac{\partial \mathcal{L}}{\partial b}$$ ← Note: fixed typo in the live notes

$$\#\text{parameters} = d + 1$$
$$12288 + 1$$

neuron = linear + activation

model = architecture + parameters

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 $[1]$ ≡ layer

subscripts ≡ identify neuron within layer

$a_2^{[1]}$ ← 2nd neuron in 1st layer

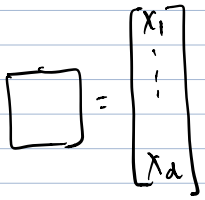
param = 3(d+1)

images + labels

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

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

Goal 3.0 :
add constraint : unique animal in image

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

$z_1^{[1]}$ $z_2^{[1]}$ $z_3^{[1]}$

$$\hat{y}_1 = e^{z_1^{[1]}} \Big/ \sum_{k=1}^{3} e^{z_k^{[1]}}$$

$$\hat{y}_2 = e^{z_2^{[1]}} \Big/ \sum_{k=1}^{3} e^{z_k^{[1]}}$$

$$\hat{y}_3 = e^{z_3^{[1]}} \Big/ \sum_{k=1}^{3} e^{z_k^{[1]}}$$

Note: Fixed typo in live notes

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

Softmax
multiclass
network

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

parameters = 3(d+1)

Loss function

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

Binary Cross Entropy Loss Function

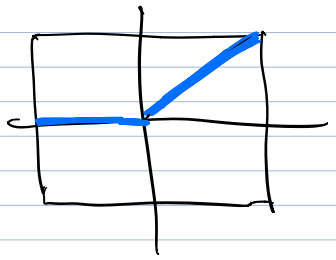
$$\mathcal{L}_{CE} = -\sum_{k=1}^{3} y_k \log \hat{y}_k$$

Q^n: Predict age of cat instead of presence of cat?

Options

① Several neurons to predict different ages

② Change activation fn



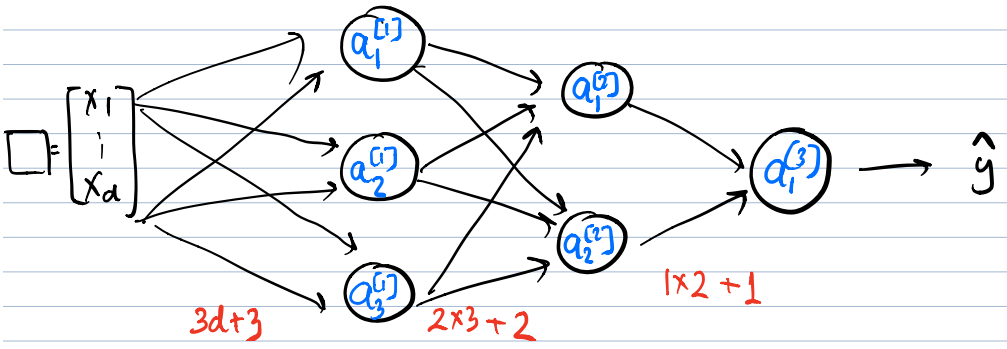
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

ReLU

Rectified Linear Unit

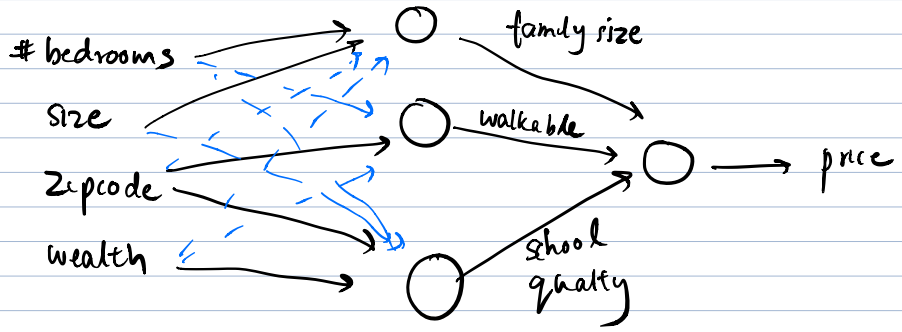
Modified Loss fn $\quad \|\hat{y} - y\|_1 \quad \|\hat{y} - y\|_2^2$

② Neural Networks



input layer, output layer
hidden layer

House price prediction



Propagation equations

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

3×1 — $a^{[1]} = \sigma(z^{[1]})$

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

2×1 — $a^{[2]} = \sigma(z^{[2]})$

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

1×1 — $a^{[3]} = \sigma(z^{[3]})$

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

Broadcasting

$\tilde{b}^{[1]} = \underbrace{\begin{bmatrix} b^{[1]} & \dots & b^{[1]} \end{bmatrix}}_{n \text{ copies}}$

$X = \begin{bmatrix} x^{(1)} & \dots & x^{(m)} \end{bmatrix}$

[] — layer

() — id of example

capital — batch of examples

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

Define loss fn