DEEP LEARNING

Coursera

Neural Network & Deep learning 4
Improving Deep Neural Network 3
Structuring Machine learning project 2
Convolution Neural Network
Sequence Models

INTRODUCTION

Artificial Intelligence (AI)

Neural Network

Nuron function

↓

Rectified Linear Unit (RELU)

Supervised learning (Output)

- Real Estate / Online Advertising / Standard
- Photo tagging / CNN
- Speech Recognition / Machine Translation / RNN
- Autonomous driving / Hybrid

← Data (Input)

- Structured Data – Computer
- Unstructured Data – Human

process: Idea → Code → Experiment → Idea

Scale drives deep learning process

- Data – Labelled
- Size of NN
- Computation
- Algorithms – Sigmoid → RELU
  - parameter changes slowly
  - gradient ↓ extreme

NEURAL NETWORK BASICS

Binary Classification

Logistic Regression

Image – RGB Channels

pixel intensity value → Feature Value

Setup

Given X

Output $\hat{y} = P(y=1/x)$

$= \sigma(w^T x + b)$ $\quad \sigma(z) = \frac{1}{1+e^{-z}}$ Sigmoid function

Parameter $w \in R^{n_x}$, $b \in R$

Easy $b \to x_0 = 1$ $\quad$ New $X \in R^{n_x+1}$

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

$X \in R^{n_x}$

$y \in \{0,1\}$

m training example

example – $(x^{(i)}, y^{(i)})$

$X = \begin{bmatrix} x^{(1)} & x^{(2)} & \cdots \end{bmatrix}$ (n_x ↓, m →)

$X \in R^{n_x \times m}$

shape

$Y = \begin{bmatrix} y^{(1)} & y^{(2)} \end{bmatrix} \in R^{1 \times m}$

Cost function i th – q th example

Loss (error) function (single training example) $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

Logistic – non convex

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

Cost function (training examples set)

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

$= -\frac{1}{m} \sum_{i=1}^{m}$ loss function

Gradient Descent

Cost function

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

Repeat { $w := w - \alpha \frac{d}{dw} J(w)$ }

learning rate, partial derivative

$-\alpha\, dw$

$y=1$ $\quad P(y/x) = \hat{y}$

$y=0$ $\quad P(y/x) = 1-\hat{y}$

$P(y/x) = \hat{y}^{y}(1-\hat{y})^{(1-y)}$

Max $\log P(y/x)$

Computation Graph

Forward Propagation / Pass ——→ compute cost function

Backward Propagation / Pass ——propagate——→ derivative

chain rule

Logistic Regression

$$z = w^T x + b$$

$$\hat{y} = \sigma(z)$$

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

i) Single Example

ii) ~~Training~~ set

$$w := w - \alpha \frac{dw}{=}$$

Vectorization

for loop ——→ vectorization
algebra

np.dot( )

np.exp( )

CPU parallelization

GPU SIMD - Single instruction multiple data

broadcasting - matrix manipulation

forward propagation

backward propagation

assert

.reshape

Jupyter / Python Notebook

numpy - rank 1 array

: vector

Neural Network

Layers

Input Hidden Output

[layer]

(individual training example)

activation

$a^{[0]}$ $a^{[layer]}_{i^{th} unit}$

w of layer = # input layer

dimension of $a^{[i]}$ = (# of nodes, # of input nodes)

Shallow neural networks

Computational graph

no. of layer - 1 input layer
hidden layer + output layer
↳ parameter associated

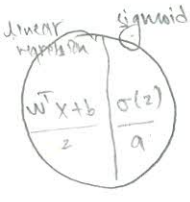
$[1]$ $[2]$ → layer
(1) (2) - training example

dimension
$w^{[1]}$ = no. of nodes × no. of input nodes / features
$b^{[1]}$ = no. of nodes × 1

Neural Network Representation - learning

$a^{[0]}$ $a^{[1]}$ $a^{[2]}$ - activation layer

$a^{[1]}_2$ → node



$$\sigma(w^T \cdot X + B) \rightarrow z \rightarrow A \text{ activation}$$

1) Single Training Example
2) ~~Entire data~~ across multiple example

n - features
m - training example

Activation functions

non-linear function

① Sigmoid
1) ~~Hidden~~ layer
2) Output activation function

② tanh - hyperbolic -1 to 1 $\frac{e^z - e^{-z}}{e^z + e^{+z}}$
mean = 0 centering of data

③ RELU - Rectified linear unit $\max(0, z)$

③.1 Leaky RELU $\max(\alpha z, z)$

Choice - Design
- Initialization
- Activation function

Non-linear activation function

linear activation function
~~activation~~ identity function

composition of two linear function is linear function

Derivative of activation function → Backward propogation

Sigmoid $g(z) = \frac{1}{1+e^{-z}}$ $g'(z) = g(z)(1 - g(z))$ - easy to compute

Tanh $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ $= 1 - (\tanh(z))^2$

ReLU

step function

Random initialization → small
Symmetric gradient problem
" breaking "

Deep-neural Network

Network structure
- Shallow
- Deep

} Neural Network

- no. of hidden layer
  - hyper parameter

Forward Propagation

Backward Propagation

$a^{[l-1]} \rightarrow a^{[l]} = w^{[l]} \cdot a^{[l-1]} + b^{[l]}$

($w^{[l]}$: current layer × previous layer; $b^{[l]}$: current layer × 1)

$da^{[l]} \rightarrow da^{[l-1]}$

$dz^{[l]}$

$dw^{[l]}$

$db^{[l]}$

($da^{[l-1]}$: previous layer × no. of example)

L = no. of layer

$n^{[l]}$ = # neuron in layer l node

$a^{[l]}$ - activation in layer l

$a^{[l]} = g^{[l]}(z^{[l]})$

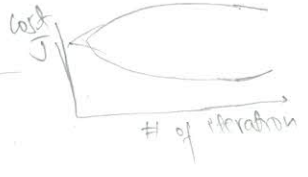
Why deep learning

low level features → basic unit → complex pattern

1) over simplified analogy — learning procedure
   neuro scientist → human brain
2) Circuit Theory — logic gate (& ∨ not, XOR) → function
   complexity $O(\log n)$ ↓ exponentially large

Parameter, Hyperparameters

- W
- b
- learning rate
- no. of hidden layer
- hidden unit
- activation function
- iteration
- Momentum
- mini batch
- regularization parameter



Emphirical prices

X → Y mapping } NN