**Neural Networks and Deep Learning**

**Week 1. Introduction to Deep Learning**

Input (X) 🡪 Neuron 🡪 Output (Y): Neuron Stacked = Layer (densely connected)

With enough data (x, y): neurons figure out functions

Supervised Learning: Input (X) – Output (Y)

* Standard NN (Structured Data): Real Estate (Home features – Price), Online Advertising (Ad, user – Ad Click)
* CNN (Image): Photo tagging (Image – Object (1, … , 1000)
* RNN (Sequential Data: Text, Audio): Speech recognition (Audio – Text transcript), Machine translation (Eng – Chn)
* Custom / Hybrid NN: Autonomous Driving (Image, Radar Info – Position of other cars)

1. Structured Data: Database (Row X Column)
2. Unstructured Data: Text, Audio, Image

🡪 Thanks to NN, better at understanding unstructured data

**Scale drives deep learning progress**

|  |  |
| --- | --- |
|  | Larger Model & (labeled) Data: High performance  If small training sets, relative ordering X defined  🡪 Depend on skills (Maybe SVM > NN)  Drivers   1. Data: Digitalization 2. Computation: Faster Iteration 3. Algorithms: Sigmoid 🡪 ReLU   Iteration Process: Idea > Code > Experiment > Idea |

**Week 2. Logistic Regression as a Neural Network**

**Binary Classification**: Result with a discrete value output (account hacked or not hacked)

Ex) Image 🡪 1 (cat) vs 0 (non cat)

Image = 3 matrices (R,G, B color channel): 64px X 64px X 3 🡪 Unrolled for Feature Vector with dimension n = 12288

**Notation**

* Single training example (with features): (x, y)
* Total m training examples:
* Input matrix: input examples stacked in row 🡪
* Output matrix: output examples stacked in row 🡪

**Logistic Regression**

Given , want

Parameters: / Output

Sigmoid function: (z is large: 1 / z = 0: 0.5 / z is large negative: 0)

. Given , want

**Loss(error) function**:

🡪 🡪

🡪 Loss Function과 p(y|x)는 역의 관계여야 하므로 음의 부호 추가!!

**Cost function**: 🡪 Find w, b that minimize J(w, b)

🡪 Cost Function은 Minimize해야하므로 maximize해야하는 확률과 달리 음의 부호 제거

Gradient Descent: Initialize with random (w, b) 🡪 Take step to global optimum

Repeat /

Derivative = Slope = Height / Width: 변수 x를 살짝 늘리거나 줄이면 y가 얼마나 바뀌냐?

**Computation Graph**

|  |  |
| --- | --- |
| **J(a, b, c) = 3(a + bc)**    / Right-to-Left Computation |  |

**Logistic Regression Gradient Descent**

|  |  |
| --- | --- |
| **Repeat** |  |

에서

**For Loop Implementation**

|  |  |
| --- | --- |
|  | Initialize Cost Function, Derivatives of weights, biases to 0  For i = 1 to m (Individual Example): Add Cost Function, Derivatives of weights, biases of  Average Cost Function, Derivatives of weights, biases  Gradient Descent  In Deep Learning with Large Data, explicit for loop X good 🡪 Vectorization |

**Vectorization**

For Loop을 통한 구현보다 numpy 기반의 vectorized 구현 (z = np.dot(w.T, x) + b)이 약 200~300배 빠름

Why? GPU/CPU: SIMD (Single Instruction Multiple Data) 🡪 Parallelism

Whenever possible, avoid explicit for-loops 🡪 Use numpy functions: np.dot(A, B), np.exp(v), np.log(v), np.abs(v), v \*\* 2, 1/v

,

,

,

🡪 Gradient Descent 반복을 위한 for loop은 필요함

**Broadcasting:** Auto-reshaping for element-wise calculation (+, -, \*, /) cf) np.dot(): matrix multiplication

* (m, n) & (1, n) 🡪 (m, n) & (m, n) / (m, n) & (m, 1) 🡪 (m, n) & (m, n)
* (m, 1) & (1, 1) 🡪 (m, 1) & (m, 1) / (1, n) & (1, 1) 🡪 (1, n) & (1, n)

Numpy에서 rank 1 array (5, )는 row vector, column vector가 아님 🡪 쓰지마 🡪 reshape((5,1))하거나 선언 시 (5,1)

**Week 3. Shallow Neural Networks**

Neuron (Node): Compute Linear Function + Non-linear Activation Function

**Neural Network**: Hidden Layer (Input, Output Layer가 아닌 neuron layer)가 있는 network

* N-Layer NN: hidden layer + output layer가 총 N개의 layer로 구성된 NN
* Why Hidden Layer? Training sets에서 observed되지 않기 때문
* 총 L개의 layer에 대해서 l번째 layer의 요소는 superscript [l]로 표현 cf) superscript (i): i번째 training example
  + Input Layer:
  + Hidden Layer: / subscript i는 l번째 layer에서 i번째 neuron (node)를 의미한다.
  + Output Layer:

Neural Network Representation (Single Example)

|  |  |  |
| --- | --- | --- |
|  |  |  |

: weight 열벡터를 transpose한 것을 vertically stacked한 matrix / : bias 열벡터

Vectorizing across multiple examples

|  |  |  |
| --- | --- | --- |
|  |  |  |

* Matrix에서 가로: horizontally stacked train example & results / 세로: features (X) or hidden units (Z, A)
* 열벡터를 horizontally stack하면 결과도 열벡터로 horizontally stack됨

**Gradient Descent**

Parameters:

Cost Function:

Repeat – Compute Prediction, Gradient Descent

텍스트, 라인, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

W는 열벡터 w들이 transpose되어 vertically stacked된 matrix였기 때문에 dw는 column vector로 X transpose 필요

Output Layer의 activation function은 sigmoid 가정 / Hidden Layer의 activation function은 g(x)로 표현

텍스트, 폰트, 영수증, 문서이(가) 표시된 사진

자동 생성된 설명

**Activation Functions**

Layer마다Neuron에서 사용되는 Non-linear activation function은 각자 달리할 수 있음!

NN에서 Linear Function이면 선형 함수의 합성은 선형 함수이므로 hidden layer가 의미가 없음 🡪 Non-Linear!

1. Sigmoid: Binary Classification의 output layer에서만 이고, 이어야 하므로 쓸만함
2. tanh : 일반적으로 sigmoid보다 좋음 (0 주위에 center되어 있기 때문)
3. ReLU: sigmoid와 tanh 모두 z가 너무 크거나 작으면 GD가 0에 가까움 🡪 learn much faster! 🡪 Best!
4. Leaky ReLU: z < 0일 때 GD = 0인 ReLU의 단점을 보완. Hyperparameter로 z < 0일 때 기울기 설정 가능

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Random Initialization**

Logistic Regression에서는 param (weight, biases)를 0으로 init해도 되지만, NN에서 weight = 0 이면 안 됨

🡪 Hidden Layer become symmetric이기 때문: dw가 동일한 row vector들이 stack된 형태가 됨

* 🡪 small value: weight too large면 z가 커져서 flat part 🡪 GD small