**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 2-1. 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)