<2. Perceptrons>

● Inspiration(영감) from Neurons
- Input과 Output이 존재하며, 뉴런들은 계속 연결되서 신호를 보냄.
- 시냅스를 통한 화학적, 전기적 통신을 진행함. (propagate)
- 뉴런은 bias를 통한 가중치를 둬서 원한느 정보를 더 강조해서 전달함.

● Neuron-Inspired Classifier
- input: x[0], x[1], x[2], ... , x[d]
- weight parameters: w[0], w[1], ... , w[d]
- $\sum_{i=0}^{d} x_i W_i + bias$
- activation function을 통해 어디를 강조시킬 건지 확장함.
- output 전달

● A "Parametric" Hypothesis
- $f(x) = sigmoid(Wx + b)$ $\begin{matrix} Wx + b > 0 \rightarrow 1 \\ Wx + b < 0 \rightarrow 0 \end{matrix}$
- remembering that: $WX = \sum_{j=1}^{d} w[j]\,x[j]$
- 우리의 목표는 학습을 통해 적절한 W와 b를 찾는 것이다.

● Linear Decision Boundaries
- 1. Linear Decision Boundaries란 어떻게 정의되는가?
- binary classification에서 Linear Decision Boundary는 입력 벡터(특징)의 결정 기준으로서 1차 방정식을 가짐.
- $f_{LM}(x) = \begin{matrix} 1 \ \text{if} \ w^T x + b > 0 \\ -1 \ \text{if} \ w^T x + b < 0 \\ undefined \ otherwise (= decision boundary) \end{matrix}$
- $f_{LM}(x) = sigmoid(w^T x + b)$

- 2. feature coefficient(특성 계수) = weight
- 내적의 개념을 사용하여, linear decision boundary를 명시한다.
- feature vector인 input X와 parameter vector인 W의 내적을 통해 linear decision boundary를 나타냄.
- $W^T X + b = w[0]x[0] + w[1]x[1] + ... + w[m-1]x[m-1] + b$

- 3. Bias (intercept)
- Bias: features에 의존하지 않는 linear model 안의 parameter이다.
- 직관적으로 보면, Bias는 전체적인 예측값을 더 positive나 더 negative로 이동시킨다.
- 학습시킬 때, bias는 필수적이다.
- 3차원 이상의 데이터를 구분하는 평면: hyper plain하고 있다.

● Online Learning

- 데이터가 시간의 흐름에 따라 순차적으로 제공될 때, 이를 즉각적으로 학습하는 기계 학습 방법론이다.
- 데이터가 한꺼번에 주어지는 것이 아니라, 스트리밍 데이터처럼 한 번에 한 개씩 들어오는 상황에서 학습하고 모델을 업데이트하는 방식이다.
  - 스트리밍 데이터: 짧은 대기 시간 처리를 목표로 계속해서 증분하는 방식으로 내보내지는 대용량 데이터이다.
- 데이터가 매우 크거나 연속적으로 생성되는 상황에서 효과적이다.

- Incremental Learning (점진적 학습)
  - Online Learning은 데이터가 들어올 때마다 모델이 그 데이터를 바탕으로 점진적으로 업데이트되는 점진적 학습 방식을 취한다.
  - 새 데이터를 처리할 때마다 모델의 파라미터가 조정되어 즉각적으로 변화를 반영한다.

- Adaptive Learning (적응형 학습)
  - 데이터의 분포가 시간이 지나면서 바뀌는 경우, Online Learning은 새로운 데이터에 맞게 모델을 적응시킬 수 있다.
  - ex) 사용자의 취향이나 시장 트렌드가 변화할 때, 모델이 그 변화를 즉각 반영할 수 있다.
  - 이는 데이터의 변화에 대한 컨셉 드리프트(concept drift)를 다루는데 유리함.

- How should we change weight if we do not make an error on some point (x, y)?
  - 어떤 점 (x, y)에 대해 에러를 범하지 않은 경우, 가중치를 어떻게 변경해야 하는가?
  - No, 바꾸면 안된다.
- 그렇다면 에러가 발생했을 경우, 어떻게 weight를 바꿔야 하나?
  - 1. 에러값이 0이 되는 방향으로 W, b를 update한다.
  - 2. W, b를 어떻게 update update해야 하는가?
    - 학습을 통해 loss function에서 update해야함.
    - Perceptron전이라 Input을 통해 Update한다.
      - w ← w + x (input)
      - w ← w − x (input)
        - w ← **w y*x (y = -1 or 1, x = input)**

● Perceptron Learning Algorithm

- Learning from error
  - 반복적으로 예측값을 올바르게 되도록 학습하면서 decision boundary를 배우길 원한다.
- 조건: y = 시그모이드 함수로 인해 {-1, 1}밖에 나오지 않는다.
- 1. 먼저 W = 1, b = 1이라고 설정하고 학습을 시작하자.
- 2. 예측값을 만들기 위해 naive decision boundary를 사용한다면, 처음 epoch가 돌면 실제 정답과 prediction이 같지 않을 것이다.
  - $sigmoid(W^T x_i + b) \neq y_i$
- 3. 현재 예측값와 과거 예측값가 같지 않으면, 현재 예측값 * 과거 예측값 < 0이다.
  - $-y_i(W^T x_i + b) < 0$ (y = {-1, 1})

- incorrect prediction이 발생하면, 어떻게 model을 개선해야 할까?
  - Logic: perceptron algorithm
    - 1. "negative한 학습 데이터를 positive으로 잘못 분류한 경우, 해당 데이터에 대해 예측을 더 negative로 만들기 위해 가중치를 조정해야 한다."
    - 2. "positive한 학습 데이터를 negative으로 잘못 분류한 경우, 해당 데이터에 대해 예측을 더 positive로 만들기 위해 가중치를 조정해야 한다."
    - $w^{NEW} = w^{old} + y_i x_i$
      $b^{NEW} = b^{old} + y_i$
    - 이 수식이 우리가 원하는 방향으로 update가 될까?

● Perceptron Learning Algorithm의 W, b Update하기
- 만약 update를 한다면 아래의 식이 보장된다.
  - $y_i((w^{NEW})^T x_i + b^{NEW}) \geq y_i((w^{old})^T x_i + b^{old})$

● Perceptron Learning Algorithm에 대한 코드
- Data: D = $< x_n, y_n >_{n=1}^{N}$, $number\ of\ epochs\ E$
- Result: weight w and bias b
- initialize: w = 0 and b = 0;
- for e∈{1, ... , E} do
- for n∈{1, ... , N} in random order do
- # predict
- $\hat{y}$= sigmoid(w*x_n + b);
- if $\hat{y} \neq y_n$ then # 만약 error가 발생했다면
- # update
- w = w + y_n*x_n;
- b = b + y_n;
- return w, b

● Parameters and Hyperparameters
- 이것은 첫 supervised algorithm으로 반복적으로 w, b를 업데이트해서 모델이 각 샘플에 대해 원하는 해답을 도출할 때까지 반복함.
- parameters = w, b
- perceptron learning algorithm의 단독 hyperparameter = E (number of epochs)

● Linear Decision Boundary
Linear Decision Boundary에 따라 클래스가 나뉘어진다.

● What would we like to do?
- Optimization problem
- perceptron algorithm의 linear decision boundary는 Non Linear data 분포를 정확히 classification되도록 최적화 boundary를 그려낼 수 없다.
- 일반적으로 위의 설명은 NP-Hard problem을 겪는다.
- 적합한 parameters(w, b)를 찾기 위해 loss function이 minimization이 되도록 학습을 시키는 것이 일반적인 접근법이다.

● Hebbian Learning
- "함께 활동하는 뉴런은 연결이 강화된다."는 개념을 기반으로 한다.
- 뉴런 간의 시냅스 연결이 강화되는 메커니즘을 설명한 것임.
- A 뉴런과 B 뉴런이 동시에 활성화된다면, A 뉴런이 B 뉴런을 활성화시키는 연결이 강화된다.
- 이는 신경 연결의 가중치가 뉴런 간의 동시 활동에 따라 강화되거나 약화되는 과정을 설명함.
- $\Delta w_{ij} = \eta x_i y_j$
- Δw_{ij} : 뉴런 i와 뉴런 j 사이의 가중치 변화
  η : 학습률, 가중치가 변화하는 속도를 조절하는 상수
  x_i : 뉴런 i의 활성화 상태
  y_j : 뉴런 j의 활성화 상태
- 이 수식은 두 뉴런이 모두 활성화되었을 때, 그들 사이의 연결 강도가 증가함을 나타냄.
- 만약 뉴런 i와 j가 동시에 활성화되지 않는다면, 가중치 변화량이 작거나 0이 될 수 있다.

● Perceptron Learning Rule
- supervised training이라고 가정함.
- Hebbian Learning에서의 synaptic weights를 배운다.
- Perceptron은 반복적인 update를 사용한다.
- binary classification task라고 가정했을 때, Peceptron은 class들이 잘 나눠지는 decision boundary를 찾는다.

● Perceptron Learning Algorithm
- Data = {(x[0], y[0]), (x[1], y[1]), (x[2], y[2]), (x[3], y[3]), ... , (x[n], y[n])}
- y = {0, +1}
- 1. Initialize w = 0^m
- 2. for every training epoch:
- for every (x[i], y[i]) in Data:
- (a) $\hat{y}$[i] = $\sigma(x[i]^T$*w)
- (b) err = (y[i] - $\hat{y}$[i])
- (c) w = w + err * x[i]

- 만약 옳다면 (prediction output == target), Do nothing
- 1) 만약 옳지 않다면 (prediction output = 0, target = 1), weight vector + input vector
- 2) 만약 옳지 않다면 (prediction output = 1, target = 0), weight vector - input vector

● Geometric Intuition

- Decision boundary가 있다면, Weight vector와 Decision boundary는 수직이다.

- Weight vector와 Decision boundary가 수직인 이유는 무엇인가?
  - 결정 경계(Decision Boundary)와 가중치 벡터(Weight Vector)가 수직(Perpendicular)한 이유는 결정 경계가 데이터를 두 클래스로 분리하는 초평면(Hyperplane)을 형성하는 반면, Weight Vector는 이 Hyperplane의 방향을 결정하기 때문이다.

- Decision Boundary
  - Perceptron에서의 결정 경계(Decision Boundary)는 두 클래스를 나누는 초평면이다.
  - Perceptron은 선형 분류기로, 수식은 y = w*x + b로 쓰인다. 이 수식에서 Decision Boundary는 y = 0인 지점에서 형성되며, 이때 w*x + b = 0인 방정식을 만족한다.
  - 이 방정식은 Hyperplane을 나타낸다.
  - Decision Boundary는 Perceptron이 0을 출력하는 곳이며, 이 초평면에서 입력 벡터 x에 대해 예측 클래스가 변경된다.

- Decision Boundary와 Weight Vector의 수직관계
  - 가중치 벡터 w는 결정 경계에서 수직 바양으로 작용한다.
  - 벡터가 평면에 수직한다는 것은 벡터가 해당 평면에 대해 법선 벡터(normal vector)라는 뜻이다.
  - Decision Boundary의 식인 w*x + b = 0에서 w*x = 0이란, 벡터 x가 가중치 벡터 w에 수직함을 나타낸다. 결정 경계에 있는 모든 점 x는 가중치 벡터 w와 내적이 0이 되어야 하므로 결정 경계의 모든 점들은 가중치 벡터에 대해 수직하게 배치된다.
  - 결론: 학습 중인 Weight vector와 input vector가 내적을 통해 각도를 구했을 때, 그 각도가 90도 미만이면 correct prediction이라는 결론이 도출됨.
  - 그래서 만약 그 각도 90도 이상이 나왔을 경우, 지금 weight vector를 input vector 방향으로 이동시킴.

● Artificial Neuron Model

- Model network = 전체 세포 그래프
- nodes, synaptic connection = 노드로부터 weighted edges
- weight: 연결강도를 의미함.

● Neural Computation

- 입력(노드) * weight(연결강도) -> weighted sum -> threshold(임계점을 통해 시그널을 binary signal로 변환) -> binary signal
- 퍼셉트론을 활용한 논리 게이트를 구현
- XOR의 경우 싱글 퍼셉트론으로 해결이 불가능하다. -> 다중 퍼셉트론은 해결이 가능함.

● General Notation for Single-Layer Neural Networks

- Single-Layer Perceptron에 대해서 알아보자.

- $\sigma(\sum_{i=1}^{m} x_i w_i + b) = \sigma(X^T W + b) = \hat{y}$

  $\sigma(z) = \begin{matrix} 0, z \leqq \Theta \\ 1, z > \Theta \end{matrix}$

  $b = -\Theta$

- b = bias

- Θ = threshold

- b = -Θ라는 수식이 왜 나왔는지에 대해서 살펴보자.
  - bias와 임계값이 서로 대칭적인 역할을 한다.
  - 1. bias가 나오기 전의 Perceptron에 대해서 살펴보자.
    - $ouput = \begin{matrix} 1, wx \geqq \theta \\ 0, wx < \theta \end{matrix}$
  - 2. bias의 도입
    - $output = \begin{matrix} 1, wx + b \geqq 0 \\ 0, wx + b < 0 \end{matrix}$
  - 3. b=-θ의 관계
    - 1과 2의 식 둘 다 성립하려면, bias는 임계값과 반대되는 역할을 해야 한다.
    - b=-θ 수식은 bias가 임계값을 대체하면서, 그 반대 방향으로 조정된다는 것을 나타낸다.
    - bias는 분류 경계의 기준점을 원점으로 옮기기 때문에, 임계값을 0으로 대체한 것처럼 동작함.

● Linear Separability(선형 분리 가능성)

- 만약 데이터가 완벽하게 선형 분리 가능하다면, w가 unit vector라는 가정 하에 아래 식이 성립한다.
- $y(wx) \geqq 1$

● Perceptron Problem

- 1. non-linear boundaries 불가능
- 2. XOR문제 해결 불가능
- 3. 클래스들이 선형 분리 가능하지 않으면, 수렴하지 않는다. -> optimal 불가
- 이 문제들을 Multilayer Perceptron으로 해결이 가능하다.