Ⅰ. Machine Learning의 개념

1) 3요소

① experience E: 표본

② task T: 목표

③ performance P: 성능

2) 분류

① Supervised Learning(지도학습)

- 미리 입력해둔 Training Data를 통해 정답을 학습한 후 일반화

② Unsupervised Learning(비지도학습, 자율학습)

- Training Data 없이 데이터가 어떻게 구성되었는지 알아내는 것

Ⅱ. Supervised Learning (지도학습)

1. Linear Regression (선형 회귀 분석)

1) 기본 개념

① $\boldsymbol{x_j^{(i)}}$

- i번째 Data의 $\boldsymbol{x_j}$값

- ex) x_3^2: 2번째 Data의 x_3값

② θ(Theta) (=Parameters)

- Parameters를 결정해 식을 완성할 수 있음

- 적합한 Parameters 값을 찾아내는 것이 목표

③ 식

➔ $\mathrm{h}_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

➔ $\mathrm{h}_\theta(x) = \theta^T x$ (θ와 x는 각각 n+1차원 vector)

④ Cost

- 실제 결과값과 Regression을 통해 나온 값 사이의 차이값

- Cost를 최소화하는 θ값(Parameters)을 찾아 $\mathrm{h}_\theta(x)$를 완성하는 것이 목표

2) Cost Function

① 목적: 실제 Data값과 예측값 사이의 차이를 측정하기 위한 함수

② Cost Function은 [Dataset의 Data] $y^{(i)}$와 [각각의 Data에 상응하는 예측값] $\mathrm{h}_\theta(x^{(i)})$사이의 **제곱오차의 합**

➔ $\mathrm{J}(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\mathrm{h}_\theta(x^{(i)}) - y^{(i)}\right)^2$

③ Cost Function이 최솟값이 되는 θ들을 찾아야 하기 때문에 Cost Function을 미분해야 함

④ Cost Function을 미분하기 쉽게 만들기 위해 식 변경 ($\frac{1}{2m}$ 곱함)

➔ $\mathrm{J}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(\mathrm{h}_\theta(x^{(i)}) - y^{(i)}\right)^2$

3) Gradient Descent (경사하강법)

① 개념: 함수를 기울어진 경사면이라 가정했을 때, 경사면에 공을 놓으면 공은 해당 지점에서 가장 가파른 방향으로 움직이는데, 이를 끊임없이 반복하면 결국 공은 최저점에 도달한다는 것에 착안한 방법

② Cost Function에 적용해 Cost Function이 최솟값이 되는 θ들을 찾는데 사용

③ 기본 개념:

⇨ $x := \mathrm{x} + \Delta x,\ \ y := y + \Delta y$

➔ $(\Delta x, \Delta y) = -\eta(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y})$ (η은 매우 작은 양의 정수)

➔ 단순화) $\boldsymbol{\Delta x = -\eta \nabla f}$

증명)

$$\mathrm{f}'(x) = \frac{\mathrm{f}(x + \Delta x) - f(x)}{\Delta x}$$

⇨ $\mathrm{f}(x + \Delta x) = f(x) + \mathrm{f}'(x)\Delta x$

⇨ $\mathrm{f}(x + \Delta x,\ y + \Delta y) = f(x, y) + \mathrm{f}'(x)\Delta x + \mathrm{f}'(y)\Delta y$

⇨ $\mathrm{f}(x + \Delta x,\ y + \Delta y) = f(x, y) + \frac{\partial f(x,y)}{\partial x}\Delta x + \frac{\partial f(x,y)}{\partial y}\Delta y$

⇨ $\mathrm{f}(x + \Delta x,\ y + \Delta y) - f(x, y) = \frac{\partial f(x,y)}{\partial x}\Delta x + \frac{\partial f(x,y)}{\partial y}\Delta y$

⇨ $\Delta z = \frac{\partial f(x,y)}{\partial x}\Delta x + \frac{\partial f(x,y)}{\partial y}\Delta y$

⇨ $\mathrm{f}(x + \Delta z)$가 최소인 값을 찾아야 하기 때문에 Δz는 최소여야 함

⇨ Δz는 $\left(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y}\right)$와 $(\Delta x, \Delta y)$의 내적

※ 내적의 최솟값은 두 벡터 값의 방향이 반대가 되어야 함.

➔ $(\Delta x, \Delta y) = -\eta(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y})$ (η은 매우 작은 양의 정수)

④ Cost Function에 Gradient Descent적용

⇨ $f(x) := J(\theta)$

⇨ $\eta := \alpha$

⇨ $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

※ $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

➔ $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$

4) Feature Scaling

① 여러 Features가 있을 때, 각각의 Features마다 크기가 다를 수 있음

② Features 사이에 크기 차이가 많이 나게 될 경우 Gradient Descent 수행 시 시간 증가

③ 식: $x_n^i = \frac{x_n^i - x_n^{mean}}{x_n^{max} - x_n^{min}}$

5) α값 (Learning Rate 학습률) 설정

① 적절한 α값일 경우: $J(\theta)$가 점차 감소하는 그래프

② α값이 너무 클 경우: $J(\theta)$가 점차 증가 or 증가 및 감소하는 그래프

- 보폭이 너무 커져 최솟값을 찾지 못할 수 있음

③ α값이 너무 작을 경우

- 보폭이 너무 작아져 최솟값 찾는 데 오랜 시간이 걸림

④ 적절한 α값 찾는 방법

- 0.001 〈α 〈 1 범위 내에서 선택 (주로 0.001)

- 3배 단위로 증가시키면서 찾아가기

6) Normal Equation (정규방정식)

① θ를 찾는 (Gradient Descent를 대체하는) 방법

② 식

➔ $\theta = (X^T X)^{-1} X^T y$

(X: [features개수+1] Vector, y: [n+1] Vector)

③ Gradient Descent와의 비교

- 학습률 α값 찾을 필요 없음
- 반복 작업 없이 한 번에 계산 가능 -> 빠름
- $(X^TX)^{-1}$계산으로 인해 features 개수가 많아질수록 계산 속도 느려짐

  (features 개수가 10000 이하일 때 적합)

2. Logistic Regression – Classification

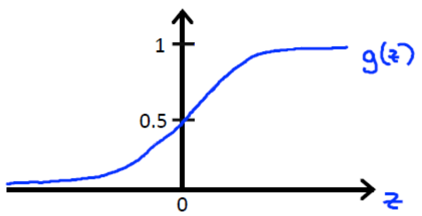
1) Sigmoid Function (=Logistic Function)

① 목적: 넓은 범위의 값을 0~1사이의 값으로 변환하는 것

② 식

➔ $g(z) = \frac{1}{1+e^{-z}}$

③ 그래프



2) Logistic Regression

① 식

➔ $h_\theta(x) = g(\theta^T x)$

➔ $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

② 확률의 의미

- $h_\theta(x) = 0$이면 $y = 0$일 확률이 1, $y = 1$일 확률이 0
- $h_\theta(x) = 1$이면 $y = 0$일 확률이 0, $y = 1$일 확률이 1

③ Decision Boundary: y가 0인지 1인지 판단하는 기준선

3) Cost Function

① $y = 0$ 또는 $y = 1$ 두 가지 경우의 수밖에 존재하지 않음

② $y = 0$일 때

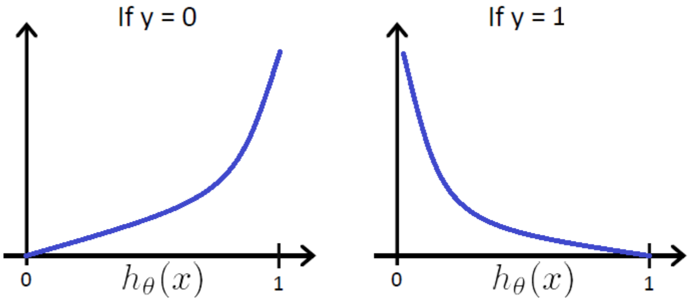
- $h_\theta(x) = 0$이면 $Cost(h_\theta(x), y) = 0$

- $h_\theta(x) = 1$이면 $Cost(h_\theta(x), y) = 1$

③ $y = 1$일 때

- $h_\theta(x) = 0$이면 $Cost(h_\theta(x), y) = 1$
- $h_\theta(x) = 1$이면 $Cost(h_\theta(x), y) = 0$

④ 그래프: log함수 차용



⑤ 식

$$Cost(h_\theta(x), y) = \begin{cases} -\log h_\theta(x) & if\ y = 1 \\ -\log(1 - h_\theta(x)) & if\ y = 0 \end{cases}$$

⇨ $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x), y)$

➔ $J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m}(y^{(i)}\log h_\theta(x^{(i)}) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)})))]$

4) Multiclass classification

① 이분법이 아닌, 결과값이 여러 개로 분류되는 경우 ($y = 1, y = 2,\ y = 3$ ...)

② One vs Rest 방식

③ 각각의 $h_{\theta_x}^{i}$는 $y = i$일 확률을 뜻함

④ 새로운 입력 x_n이 추가되었을 때:

- $h_{\theta_{x_n}}^{i}$을 모두 파악해본 후, 가장 높은 $h_{\theta_{x_n}}^{i}$을 가진 i로 x_n을 분류함

3. Regularization (정규화)

1) 개요

① Overfitting(과적합): 최적화가 너무 과도하게 진행되어, 일반화하기 어려운 상태 (새로운 입력이 들어왔을 때 제대로 예측하리라고 보기 어려운 상태)

- 결과에 큰 영향을 미치지 않는 features를 삭제하면서 features의 수를 줄여서 해결 가능하나 위험성 존재하므로 Regularization을 통해 해결함이 바람직함

② Regularization: 일부 parameters를 작은 값(0에 수렴하게)으로 만들어 $h_\theta(x)$를 단순화

- 정규화 시 상수 parameter(θ_0)은 포함하지 않음

2) Linear Regression의 정규화

① Linear Regression – Cost Function의 정규화

⇨ $J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2\right]$

➔ $J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \boldsymbol{\lambda}\sum_{j=1}^{n}\boldsymbol{\theta_j^2}\right]$

※ 원리: Cost Function의 뒤에 parameters와 큰 수($\boldsymbol{\lambda}$ lamda)를 곱한 값을 더함

⇨ Cost Function을 최소화해야 하는 것이 목적이므로, 계산 결과 해당 parameters는 아주 작은 값을 가질 수밖에 없음

② Linear Regression – Gradient Descent의 정규화

⇨ $\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$ $(j > 0)$

⇨ $\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})- y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]$ $(j > 0)$

➔ $\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})- y^{(i)}\right)x_j^{(i)}$ $\mathbf{(j > 0)}$

※ 원리: θ_j에 1보다 작은 값을 곱해 θ_j를 줄여나감

⇨ $\alpha > 1,\ \lambda > 1$이므로 $\left(1 - \alpha\frac{\lambda}{m}\right) < 1$

③ Linear Regression – Normal Equation의 정규화

⇨ $\theta = (X^TX)^{-1}X^Ty$

➔ $$\theta = \left(X^TX\ \lambda\begin{bmatrix} \mathbf{0} & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 \end{bmatrix}\right)^{-1} X^Ty$$

3) Logistic Regression의 정규화

① Logistic Regression - Cost Function의 정규화

⇨ $J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\left(y^{(i)}\log h_\theta\left(x^{(i)}\right) + (1 - y^{(i)})\log(1 - h_\theta\left(x^{(i)}\right))\right)\right]$

➔ $= -\frac{1}{m}\left[\sum_{i=1}^{m}\left(y^{(i)}\log h_\theta\left(x^{(i)}\right) + (1 - y^{(i)})\log(1 - h_\theta\left(x^{(i)}\right))\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$

② Logistic Regression - Gradient Descent의 정규화

$$\Rightarrow \theta_j := \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} \quad (j > 0)$$

4. Neural Networks

1)

5. Support Vector Machines

Ⅲ. Unsupervised Learning

i. K-means

ii. PCA

iii. Anomaly Detection