2강: Supervised Learning

🤍목차🤍

Supervised Learning

지도학습(Supervised Learning)이란, 데이터에 대한 학습데이터가 주어진 상태에서 컴퓨터를 학습시키는 방법이다. 쉽게 말하면, Y값이 주어진 경우라고 생각하면 된다.

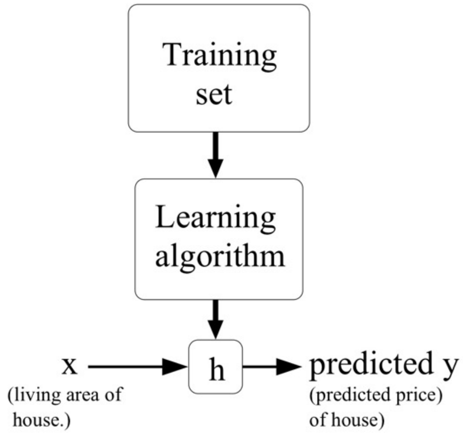
🍀 간단하게 notation 정리

1. $x^{(i)}$: input variables, input features
2. $y^{(i)}$: output or target variables
3. A pair $(x^{(i)}, y^{(i)})$: ith training example
4. a list of n training examples $(x^{(i)}, y^{(i)}); i = 1, \ldots, n$: a training set

y	x_1	x_2	$\cdots$	x_d
$y^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	$\cdots$	$x_d^{(1)}$
$y^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	$\cdots$	$x_d^{(2)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$y^{(n)}$	$x_1^{(n)}$	$x_2^{(n)}$	$\cdots$	$x_d^{(n)}$

- **지도학습의 목표**

지도학습의 목표는 주어진 training set에 대응되는 y값을 잘 예측하는 함수 $h(x)$를 추정하는 것이고 이 함수를 $hypothesis$라고 한다.



[통데마 수업] 지도학습이란?

- Format of supervised learning: $Y = f(x) + \epsilon$
- Y변수가 연속형이면 regression problem, y변수가 이산형이면 classification problem
-
ϵ : random error term (기본가정: $E(\epsilon) = 0/\ Var(\epsilon) = \sigma^2/\ \epsilon_1, \dots, \epsilon_i\ indep)$
* 머신러닝에서는 가정을 확인하지 않음!
-
지도학습의 목표는 f를 추정하여 Y를 예측하는 것이다.

[통데마 수업] f를 추정하는 방법

* training data를 사용!

1. Parametric methods (Model-based approach)
:
f의 형태 가정하고, training data를 통해 모델의 parameter 추정 / Linear or Nonlinear model
🚩
Estimation of parameters $\equiv$ Estimation of f

2. Nonparametric methods

: 오직 데이터에 의존해서 f를 추정/ Nonlinear model

- **모델 평가와 선택의 기준**

> [통데마 수업] 지도학습의 목표가 $prediction$인 만큼, 모델의 평가와 선택은 얼마나 Y를 잘 예측했는지에 달려있다.
> 💢주의💢 모델을 세울 때는 training data를 모델을 평가할 때는 test data를 사용한다.
>
> 1. Regression problem:
> - Criterion:
> $Test\ MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{f}(x_i))^2$
> - Example : KNN regression
>
> 2. Classification problem:
> - Criterion:
> $Test\ Misclassification\ rate = \frac{1}{m}\sum_{i=1}^{m} I(y_i \neq \hat{y}_i)$
> - Example : Bayes classifier / KNN classifier
>
> test data가 충분치 않은 경우에는 Sample re-use methods(e.g. bootstrap, cross-validation)를 사용한다.

▼ 여기까지 통데마 lecture2 내용

Ch1: Introduction

Ch2: Supervised learning

Part1 Linear Regression

- 지도학습(supervised learning)을 수행하기 위해서는 function/hypothesis h를 어떻게 나타낼 것인지 결정해야 한다. 초기선택으로, x, y가 대략적으로 linear관계가 있다고 결정하면, 다음과 같이 수식으로 나타낼 수 있다.

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + ... + \theta_d x_d \\ &= \sum_{i=0}^{d} \theta_i x_i \\ &= \theta^T X, \end{aligned}$$

이때,

θ_i는 parameter (weight)

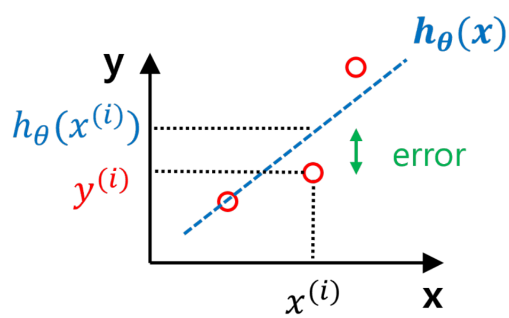
$x_0 = 1$로 intercept term

d은 input variable의 개수

> **[통데마 수업] linear regression**
> : supervised learning/
> parametric method
> - model:
> $f(x) = \beta_0 + \sum_{j=1}^{p} \beta_j \mathbf{X}_j$ (f모델을 세우고 Y값을 예측)
> - least square estimation(LSE): RSS를 최소화하는
> β를 찾기

- 지도학습의 목표는 y값을 잘 예측하는 것이라고 했다. 이는 주어진 training data에 대해 y값에 유사하도록 $h(x)$의 parameter를 정하는 것과 같다. 이때 우리는 각 θ값에 대해 $h(x^{(i)})$가 $y^{(i)}$에 얼마나 가까운지/유사한지 측정하는 함수를 정의할 수 있다. 이를 **cost function** 이라고 하며 다음과 같이 나타낼 수 있다.

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2$$



이전에 linear regression을 본적이 있다면 least squares cost function과 유사함을 알 수 있을 것이다.

1.1 LMS algorithm

우리는 계속해서 지도학습에 대해서 배우고 있다. 지도학습의 목표는 y값을 잘 예측하는 함수를 찾는 것이고 이 함수를 찾기 위해서는 함수의 파라미터를 찾아야 한다. 이를 구하기 위해 linear regression에서 함수값과 y값이 얼마나 가까운지를 나타내는 cost function $J(\theta)$를 정의했고, 최종적으로 $J(\theta)$값을 최소화 하는 방향으로 θ를 찾아야 y를 가장 잘 예측하는 함수를 찾고자 하는 지도학습의 목표에 달성할 수 있을 것이다.

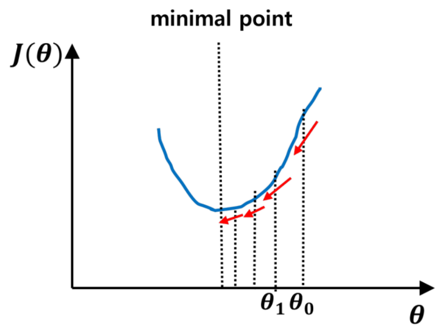
💡 **gradient descent algorithm**

1) 개념

이를 위해 첫번째로 배울 search algorithm은 gradient descent algorithm으로, 어떤 초기 θ값에서 시작하여 $J(\theta)$를 줄이기 위해 다음과 같이 반복적으로 θ를 업데이트하는 방법이다.

🍀 추가설명 🍀

먼저 parameter theta를 어떤 값으로 초기화한다. 이제 초기값으로 설정한 theta에서 J(theta)의 gradient를 구하고, 초기값에서 gradient 만큼 뺀 값을 그 다음 theta 값으로 삼는다. 이러한 작업을 theta가 수렴할 때까지 반복한다



gradient가 0인 지점이 곧 minimal point이므로 계속 iteration을 반복하다 보면 minimizing point로 수렴하게 되는 것이다.

2) 수식

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$
$$j = 0, \ldots, d$$

α는 learning rate, **이것은 각 iteration마다 parameter theta를 얼마나 변화시킬지를 결정**
한다. 너무 작으면 iteration이 많아져 연산량이 많아지고, 너무 크면 minimal point를 건너뛰어 버리는 tradeoff 가 있다.

J에서 가장 가파른 방향으로 감속하도록 업데이트를 반복하는 방법이다.

▼ *공식 유도를 위한 수학개념 정리는 아주 긴 여정이 될 것 같아서 추후에 다시 정리하겠습니당,,,,* 💛 지지지난 선대팀 교안 참고할 예정!

~~이 공식이 나오게 된 원리에 편미분, 그라디언트 등의 추가적인 수학적 설명이 필요한데 강의에서는 수학 내용을 다 생략하기 때문에 내일부터 열심히 정리해서 올려놓겠습니다,,,!!~~

- 이 알고리즘을 수행하기 위해서는 우변에 편미분을 해야한다. 이를 쉽게 설명하기 위해 하나의 training example (x, y)만 있다고 생각해보자

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2}(h_\theta(x) - y)^2 \\
&= 2 \, \frac{1}{2}(h_\theta(x) - y)^2 \, \frac{\partial}{\partial \theta_j}(h_\theta(x) - y) \\
&= (h_\theta(x) - y) \, \frac{\partial}{\partial \theta_j}(\sum_{i=0}^{d} \theta_i x_i - y) \\
&= (h_\theta(x) - y) x_j
\end{aligned}$$

따라서 하나의 training example에 대한 update rule은 다음과 같다.

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$$

바로 이 rule을 LMS update rule이라고 하는 것이다~~

- 위에서는 하나의 training example에 대해서 LMS rule을 이끌어낸 것이다. 이제 하나 이상의 training set에 대해서 나타낼 수 있는 방법은 다음과 같이 2가지 방식이 있다.

1. **Batch gradient descent**

   첫번째 방법은 다음과 같이 한번의 step마다 전체 training set을 모두 보면서 합을 구하는 방식이다.

   Repeat until convergence {

   $\theta_j := \theta_j + \alpha \sum_{i=1}^{n}(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}\ (for\ every\ j)$

   }

2. **Stochastic gradient descent (incremental gradient descent)**

   두번째 방법은 한번에 하나의 training example에 대해 파라미터를 업데이트하면서 training set을 도는 방법이다.

   Loop {

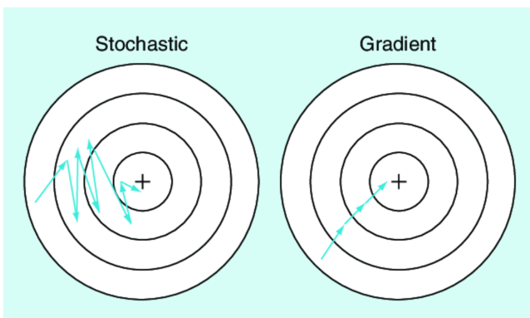
   for i=1 to n, {

   $\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}\ (for\ every\ j)$

   }

   }

   두번째 방법은 데이터가 많을 경우 첫번째 방법보다 매우 빠르지만 $J(\theta)$를 최소화 하기 위해 업데이트하는 과정에서 example 하나마다 업데이트되기 때문에 진동하여 miminum지점에 수렴하지 않을 수도 있다. 참고로 이러한 경우에는 α값을 줄여서 천천히 감소하게 한다면 해결될 수 있으며 상당한 경우에 true mininum에 유사한 값을 주기 때문에 선호되는 방식이긴 하다.



1.2 The normal equations

이번에는 J를 최소화하는 방법들 중에서 gradient descent와 같이 반복하는 알고리즘 대신 θ_j에 대한 미분을 통해 명확하게 최소화를 수행하는 방법을 알아보자.

- 파라미터 적을 때 유용!

이를 쉽게 설명하게 위해서는 matrix 계산이 필요하기 때문에 이에 대한 notation을 먼저 정리하고자 한다.

1.2.1 Matrix derivatives

- **Gradient** $\nabla_A f(A)$

  $f : \mathbb{R}^{n\times d} \longmapsto \mathbb{R}$, n by d matrices를 실수로 대응시키는 함수가 있다고 할 때, f를 A로 미분한 것은 그라디언트 $\nabla_A f(A)$를 의미하고 다음과 같이 나타낸다.

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{n1}} & \cdots & \frac{\partial f}{\partial A_{nd}} \end{bmatrix}$$

이때, $(i,j)\ element$ 는 $\frac{\partial f}{\partial A_{ij}}$으로 나타낸다.

예를 들어,

$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$이고

$f : \mathbb{R}^{2\times 2} \longmapsto \mathbb{R}$ 함수가 $f(A) = \frac{3}{2}A_{11} + 5A_{12}^2 + A_{21}A_{22}$로 주어질 때

$\nabla_A f(A) = \begin{bmatrix} \frac{3}{2} & 10A_{12} \\ A_{22} & A_{21} \end{bmatrix}$으로 나타낼 수 있다.

- **trace** : square matrix의 diagonal entries의 합

$$tr(A) = \sum_{i=1}^{d} A_{ii}$$

성질1

$$\begin{gathered} trAB = trBA \\ trABC = trCAB = trBCA, \\ trABCD = trDABC = trCDAB = trBCDA. \end{gathered}$$

성질2

$$\begin{gathered} trA = trA^T \\ tr(A+B) = trA + trB \\ tr\ aA = atrA \end{gathered}$$

- matrix derivatives

$$\begin{gathered} \nabla_A trAB = B^T \qquad (1) \\ \nabla_{A^T} f(A) = (\nabla_A f(A))^T \qquad (2) \\ \nabla_A trABA^TC = CAB + C^TAB^T \qquad (3) \\ \nabla_A |A| = |A| (A^{-1})^T \qquad (4) \end{gathered}$$

▼ 증명은,,,생략하려고 합니다,,,

~~하지만 원하는 분이 있다면 공장 가동합니다,,!~~

1.2.2 Least squares revisited

이제 위에서 정리한 matrix derivatives를 통해 $J(\theta)$를 최소화하는 θ를 찾을 수 있을 것이다. $J(\theta)$를 matrix-vectorial notation으로 다시 나타내보자.

- design matrix X (input values from the training set):

  $n \times d$ matrix (intercept term 포함 시 $n \times (d+1)$)

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(n)})^T — \end{bmatrix}$$

- $\vec{y}$ (target values from the training set): n차원 벡터

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

- $J(\theta)$

$$J(\theta) \quad = \quad \tfrac{1}{2} \textstyle\sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2$$

  - since $h_\theta(x^{(i)}) = (x^{(i)})^T\theta$,

$$\begin{aligned} X\theta - \vec{y} &= \begin{bmatrix} (x^{(1)})^T\theta \\ \vdots \\ (x^{(n)})^T\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \\ &= \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(n)}) - y^{(n)} \end{bmatrix}. \end{aligned}$$

  - using the fact for a vector z, $z^T z = \sum_i z_i^2$:

$$\begin{aligned} \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) &= \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= J(\theta) \end{aligned}$$

이제 J를 최소화하기 위해 θ에 대해 미분하는 방법을 알아야 한다. 1.2.1의 equation (2)와 (3)을 합하면, 다음과 같은 식을 유도할 수 있다.

$$\nabla_{A^T} trABA^TC = B^TA^TC^T + BA^TC \qquad (5)$$

따라서,

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) \\
&= \frac{1}{2}\nabla_\theta \left(\theta^T X^T X\theta - \theta^T X^T \vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}\right) \\
&= \frac{1}{2}\nabla_\theta \operatorname{tr}\left(\theta^T X^T X\theta - \theta^T X^T \vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}\right) \\
&= \frac{1}{2}\nabla_\theta \left(\operatorname{tr}\theta^T X^T X\theta - 2\operatorname{tr}\vec{y}^T X\theta\right) \\
&= \frac{1}{2}\left(X^T X\theta + X^T X\theta - 2X^T\vec{y}\right) \\
&= X^T X\theta - X^T\vec{y}
\end{aligned}
$$

이를 0으로 두면, 다음과 같이 **normal equations**을 구할 수 있다.

$$X^T X\theta = X^T\vec{y}$$

최종적으로 식을 풀면 $J(\theta)$를 최소화하는 θ값을 구할 수 있다.

$$\theta = (X^T X)^{-1}X^T\vec{y}$$