Lec3 Supervised Learning (continued)

Outline:

- Linear Regression (Recap)
- Locally weighted regression
- Probablistic interpretation
- Logistic Regression
- Newton's method

Recap:

$(x^{(i)}, y^{(i)})$ – i^{th} example

$x^{(i)} \in \mathbb{R}^{n+1}$ $y^{(i)} \in \mathbb{R}$

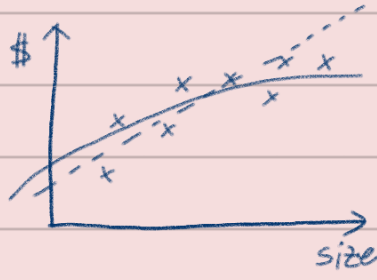
m = # of example n = # of features

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j = \theta^T x$$

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

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$ 파라미터 벡터 → $\theta^T = [\theta_0, \theta_1 \cdots \theta_n]$

x와 행렬곱할 때 전치.



$\theta_0 + \theta_1 x_1$ do we want linear fn?

$\theta_0 + \theta_1 x_1 + \theta_2 x^2$ or quadratic?

or sth else?

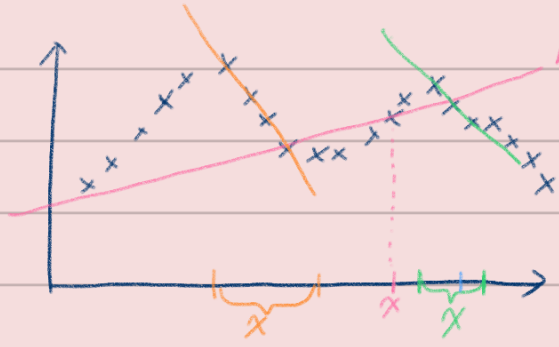
Locally weighted regression

"parametric" learning algorithm:

Fit fixed set of parameters (θ_i) to data

"Non-parametric" learning algorithm: → not good when data set is huge

Amount of data/parameters you need to keep grows (linearly) with the size of data



= make a prediction

To evaluate h at certain x:

LR: Fit θ to minimize $J(\theta)$, return $\theta^T x$

Locally weight regression:

Fit θ to minimize $\sum_{i=1}^{m} w^{(i)} (y^{(i)} - \theta^T x^{(i)})^2$

where $w^{(i)}$ is a weight function

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$$

$x^{(i)}$: ith training example

x: where I want to make the prediction

If $|x^{(i)} - x|$ is small, $w^{(i)} \approx 1$

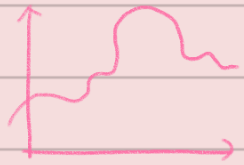
If $|x^{(i)} - x|$ is large, $w^{(i)} \approx 0$

⇒ net effect: training example이 예측하려고 하는 input에 따라, $w^{(i)}$가 0이 가까워지게 essentially sums over for the squared error for the examples that are close to x

τ (타우): bandwidth

LWR에서 τ(타우)가 클수록 모델이 underfitting 해짐
작을수록 모델이 overfitting 해짐

underfitting
training data와 예측하고자 하는 test data 모두에서 예측 성능 낮음



overfitting
training data에는 성능 Good
test data는 잘 안맞음
(∵ noise까지 학습해서)

Probablistic interpretation

Why Least Squares? $J(\theta)$

Assume ① $y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$ ← error term (unmodeled effect, random noise)

house price — mean — i번째 샘플의 오차 — e.g. the mood of the seller..!

다음에 오는 분포를 따름

$$\varepsilon^{(i)} \sim N(0, \sigma^2)$$ ← ∵ central limit theorum

실제 모델에서는 very far from Gaussian but default noise distribution 은 Gaussian!

probability function = "density"

$$P(\varepsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\varepsilon^{(i)})^2}{2\sigma^2}\right)$$

Assumption ② on ε : IID (independent & identically distributed)

this implies:

$$P(y^{(i)} | x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

$(y^{(i)} - \theta^T x^{(i)}) = \varepsilon^{(i)}$

"parameterized by θ" 라고 읽음
θ가 파라미터에 포함된다는 의미
random var 이 아니라 set of params.

i.e. $y^{(i)} | x^{(i)}; \theta \sim N(\theta^T x^{(i)}, \sigma^2)$

= Given $x^{(i)}$ & θ, the house price $y^{(i)}$의 probability (density)는 $\theta^T x^{(i)}$를 mean으로, σ^2을 variance로 가지는 정규분포 (Gaussian)를 따른다

이때 확률밀도함수 $p(y^{(i)} | x^{(i)}; \theta)$는, 모델이 θ를 파라미터로 가질 때, '$y^{(i)}$가 관측될 확률'을 나타냄

Likelihood of θ

$$\mathcal{L}(\theta) = P(\vec{y} | x; \theta)$$
$$= \prod_{i=1}^{m} p(y^{(i)} | x^{(i)}; \theta)$$
$$= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

'likelihood of parameter' (O)
likelihood of data (X)

Log Likelihood of θ

$$\ell(\theta) = \log \mathcal{L}(\theta)$$
$$= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp(\cdots)$$
$$= \sum_{i=1}^{m} \left[\log \frac{1}{\sqrt{2\pi}\sigma} + \log \exp(\cdots)\right]$$
$$= m \log \frac{1}{\sqrt{2\pi}\sigma} + \sum_{i=1}^{m} \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$$

MLE : Maximum Likelihood estimation : choose θ to maximize $\mathcal{L}(\theta)$

$\leftrightarrow$ choose θ to maximize $\ell(\theta)$

i.e. choose θ to minimize $\sum_{i=1}^{m} \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$

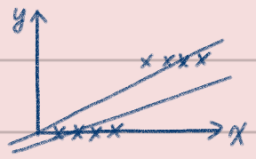
(∵ 첫번째 term엔 θ가 없으니까)

which is $J(\theta)$.

∴ Least Squares!

Classification

$y \in \{0, 1\}$ binary classification



for classification, linear regression is not a good algorithm.

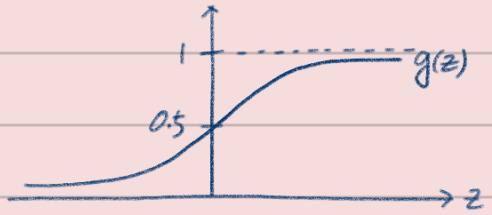
Logistic regression

이건 예측하는 아님. LR과 비슷해서 $\ell(\theta)$가 global max를 가지므로 제공된다 No local optimas

want $h_\theta(x) \in [0, 1]$

$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$ $h_\theta(x) = \theta^T x$를 sigmoid function 래퍼로 감

$g(z) = \frac{1}{1+e^{-z}}$ "Sigmoid" or "logistic" function



e.g. $P(y=1 \mid x; \theta) = h_\theta(x)$ (if tumor is malignant; size of the tumor)

$P(y=0 \mid x; \theta) = 1 - h_\theta(x)$

Linear Algebra Compress $\Rightarrow p(y \mid x; \theta) = h(x)^{y^{(1)}} (1-h(x))^{1-y^{(0)}}$

① Define Likelihood

$$L(\theta) = p(\vec{y} \mid x; \theta) = \prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)}; \theta) = \prod_{i=1}^{m} h_\theta(x^{(i)})^{y^{(i)}} (1-h_\theta(x^{(i)}))^{1-y^{(i)}}$$

② Define log Likelihood

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{m} y^{(i)} \cdot \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1-h_\theta(x^{(i)}))$$

③ Choose θ to maximize $\ell(\theta)$

④ plug θ to the model and use the new feature $x \rightarrow$ predict y

①~④를 반복하면서 : $\theta_j := \theta_j + \alpha \frac{\partial}{\partial \theta_j} \ell(\theta)$ (양수; 경사 상승의 계산식 사용)

<> square error $J(\theta)$를 쓰는데는 (least squares)

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

오차를 minimize 해야하므로

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

Newton's method

Linear Regression Normal Eq을 쓰지않고 더 빠르게 progress

Have f, want to find θ s.t. $f(\theta) = 0$

[Want to maximize $\ell(\theta)$ i.e. want $\ell'(\theta) = 0$]

$\theta^{(1)} := \theta^{(0)} - \Delta$

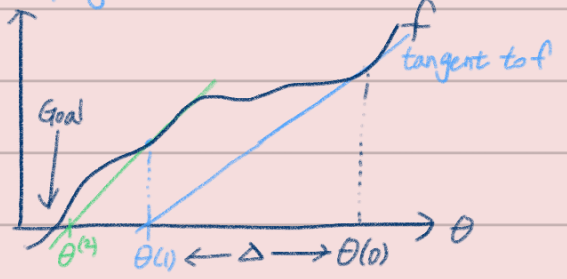
$f'(\theta^{(0)}) = \frac{f(\theta^{(0)})}{\Delta}$

$\Delta = \frac{f(\theta^{(0)})}{f'(\theta^{(0)})}$

$\theta^{(t+1)} = \theta^{(t)} - \frac{f(\theta^{(t)})}{f'(\theta^{(t)})}$

Let $f(\theta) = \ell'(\theta)$

$\theta^{(t+1)} := \theta^{(t)} - \frac{\ell'(\theta^{(t)})}{\ell''(\theta^{(t)})}$



"Quadratic convergence" Newton's method의 장점

0.01 error → 0.0001 error → 0.00000001 error

becomes much more accurate by 1 single iteration

장점이 있는데 # of parameters ≤ 50 ish otherwise go for Gradient Decent instead

when θ is a vector: $(\theta \in \mathbb{R}^{n+1})$ ← 정확한 θ^0

$$\theta^{(t+1)} := \theta^{(t)} + H^{-1}\nabla_\theta \ell$$

H^{-1}: $\mathbb{R}^{(n+1)\times(n+1)}$ → vector of derivatives $\mathbb{R}^{n+1}$

when H is the Hessian matrix

$$H_{ij} = \frac{\partial^2 \ell}{\partial\theta_i \partial\theta_j}$$

→ θ가 high dimentional vector이면 newton's method each step 매우 비싸짐

∵ having to invert a pretty big matrix (H^{-1})