Lecture - 1

Machine Learning
Gives computer the ability to learn without being explicitly programmed

Supervised Learning
for given dataset with input x and label Y,
then learn mapping from x to Y.

Regression problem - value Y is continuous
Classification problem - Y takes on discrete number of variable

Support vector machine ⇒ infinite dimension

Deep learning
a neural network with 3 or more layers

Unsupervised learing
dataset with no labels (i.e x with no Y given)
* Use x and find interesting things about it.

Reinforcment learning
Let it fn the way it wants and if it's correct output, then it's fine but if it show wrong output, we correct it.

Lecture-2

Linear Regression

Training set

↓

Learning algorithm

↓

h (hypothesis)

function takes input & tries to tell output.

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

θ parameter

x_1 feature 1

x_2 feature 2

$$h(x) = \sum_{j=0}^{n} \theta_j x_j$$

where $x_0 = 1$ (dummy feature)

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \qquad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

θ = parameter

m = training example / no of rows in table

x = "input" / feature

y = output / target variable

(x, y) = training example

(x^i, y^i) = i^{th} training example

$$h_\theta(x) = h(x)$$

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

minimize $J(\theta)$

Gradient Descent

Start with some θ (say $\theta = \vec{0}$)

keep changing θ to reduce $J(\theta)$.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \qquad (j = 0, 1, 2 \ldots)$$

($:=$ assignment; α = learning rate)

$$\frac{d}{d\theta_j} J(\theta) = \frac{d}{d\theta_j} \frac{1}{2} \left(h_\theta(x) - y \right)^2$$

$$= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y \right) \cdot \frac{d}{d\theta_j} \left(h_\theta(x) - y \right)$$

$$= \left(h_\theta(x) - y \right) \cdot \frac{d}{d\theta_j} \left(\theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n - y \right)$$

$$= \left(h_\theta(x) - y \right) \cdot x_j$$

Repeat until convergence

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

$\left(\sum_{i=1}^{m} \left(h_\theta(x^i) - y^i \right) \cdot x_j^i = \frac{\partial}{\partial \theta_j} J(\theta) \right)$

(for $j = 0, 1 \ldots n$)

Stahastic gradient descent

repeat {
    for i = 1 to m {
        $\theta_j := \theta_j - \alpha (h_\theta(x^i) - y^i)\, x_j^i$
    }
}

(for every j)

* Normal eqn

to set theta = to that and in one step with a few matrix multiplications you end up with the optimal value of theta that land you right at the global optimum (in 1 step)

$\theta = (\ldots)$

$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \end{bmatrix}$

$\theta \in \mathbb{R}^{n+1}$ (3-D)

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

$$X\theta = \begin{bmatrix} (x^1)^T \\ (x^2)^T \\ \vdots \\ (x^m)^T \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

$$= \begin{bmatrix} (x^1)^T \theta \\ (x^2)^T \theta \\ \vdots \\ (x^m)^T \theta \end{bmatrix}$$

$$= \begin{bmatrix} h_\theta(x^1) \\ \vdots \\ h_\theta(x^m) \end{bmatrix}$$

$$y = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix}$$

$$J(\theta) = \frac{1}{2} (X\theta - y)^T (X\theta - y)$$

$$X\theta - y = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

$$(X\theta - y)^T (X\theta - y) \qquad \left[\because z^T z = \sum z^2\right]$$

$$\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2} (X\theta - y)^T (X\theta - y)$$

$$= \frac{1}{2} \nabla_\theta (\theta^T X^T - y^T)(X\theta - y)$$

$$= \frac{1}{2} \nabla_\theta \left[\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y\right]$$

$$= \frac{1}{2} \left[X^T X\theta + X^T X\theta - X^T y - X^T y\right]$$

$$= X^T X\theta - X^T y \overset{\text{set}}{=} \vec{0}$$

$$X^T X\theta = X^T y \qquad \text{["Normal eqn"]}$$

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

06/08/08

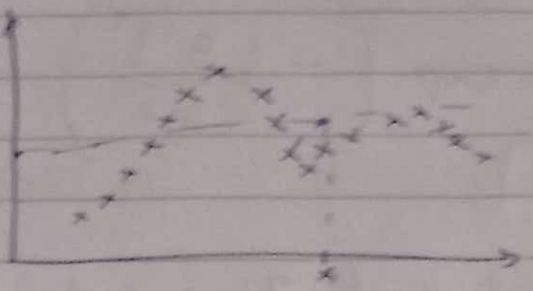
Lecture - 3

Locally weighted Regression

"Parametric" learning algorithm
Fit fixed set of parameters (θ_i) to data

"Non-parametric learning algorithm
Amount of data/parameter you need to keep grows (linearly) with size of data

Ex

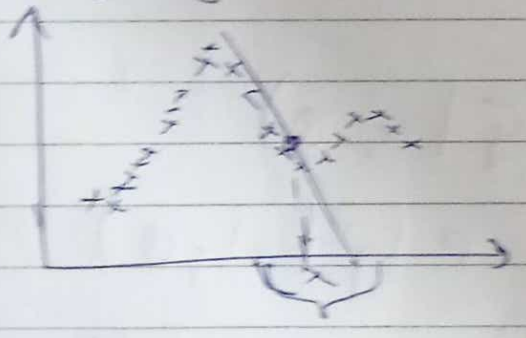


To evaluate h at certain x:
LR: fit θ to minimize

$$\frac{1}{2}\sum_i (y^i - \theta^T x^i)^2$$

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 "weighting" fn

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

If $|x^i - x|$ is small, $w^i \approx 1$
If $|x^i - x|$ is large, $w^i \approx 0$

(τ = bandwidth)

why τ?

* Why least squares?

Assume $y^i = \theta^T x^i + \varepsilon^i$

ε^i = error; includes unmodeled effects, random noise

$\varepsilon^i \sim \mathcal{N}(0, \sigma^2)$

$\sim$ = twiddle = distributed as

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

IID = independently & Identically Distributed

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

given; parameterized by; mean; variance

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

$L(\theta)$ = "likelihood of θ"

$$= p(\vec{y} \mid x; \theta)$$

$$= \prod_{i=1}^{m} p(y^i \mid x^i; \theta)$$

$$= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(\frac{-(y^i - \theta^T x^i)^2}{2\sigma^2}\right)$$

Likelihood Of Probability

- If you view this as a fn of the parameters holding the data fixed, then we call that the likelihood.
- So if we think as training set as a fixed thing and then varying parameters theta, then we used term "likelihood"
- Whereas if we view the parameters theta as fixed and maybe varying the data, we are gonna say probability
- Likelihood of parameter
  Probability of data

"log likelihood"

$$\ell(\theta) = \log L(\theta)$$

$$= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp(\dots)$$

$$= \sum_{i=1}^{m} \left[\log \frac{1}{\sqrt{2\pi}\sigma} + \log \exp(\dots) \right]$$

$$= m \log \frac{1}{\sqrt{2\pi}\sigma} + \sum_{i=1}^{m} - \frac{(y^i - \theta^T x^i)^2}{2\sigma^2}$$

MLE : maximum likelihood estimation

Choose θ to maximize $\ell(\theta)$

i.e Choose θ to minize $\frac{1}{2}\sum_{i=1}^{n}\left(y^i - \theta^T x^i\right)^2 = J(\theta)$

Classification

Logistic regression

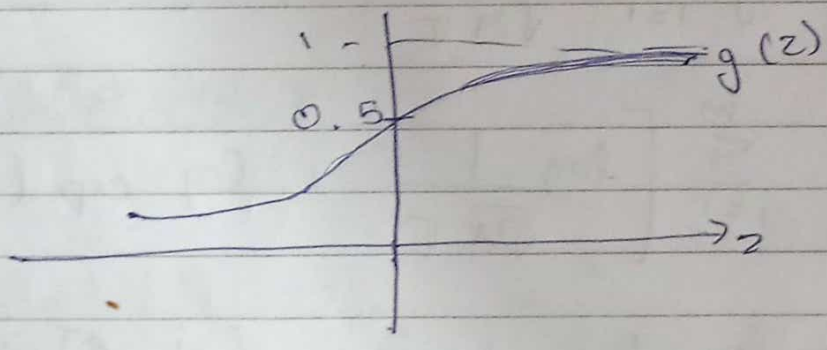
Want $h_\theta(x) \in [0,1]$

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

$$h_\theta(x) = \theta^T x$$

$$g(z) = \frac{1}{1+e^{-z}}$$

"Sigmoid" or "logistic" function



$$\begin{cases} P(y=1 \mid x ; \theta) = h_\theta(x) \\ P(y=0 \mid x ; \theta) = 1 - h_\theta(x) \end{cases}$$

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

$$\rightarrow P(y \mid x ; \theta) = h_\theta(x)^y \left(1 - h(x)\right)^{1-y}$$

$$L(\theta) = p(\vec{y} \mid x; \theta)$$

$$= \prod_{i=1}^{m} p(y^i \mid x^i, \theta)$$

$$= \prod_{i=1}^{m} h_\theta(x^i)^{y^i} (1 - h_\theta x^i))^{1-y^i}$$

$$\ell(\theta) = \log L(\theta)$$

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

Choos θ to maximize $\ell(\theta)$

Batch gradient ascent:

$$\theta := \theta_j + \alpha \frac{\partial}{\partial \theta_j} \ell(\theta)$$

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

$$\frac{\partial}{\partial \theta_j} \ell(\theta)$$