2*

Linear Regression

(Lecture Notes)

$x^{(i)}$ ⇒ i^{th} input feature in training set

$y^{(i)}$ ⇒ i^{th} output in traing set

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

$\{(x^{(i)}, y^{(i)}) \mid 1 \leq i \leq m\}$ ⇒ training set

X ⇒ denotes the Space of input features.

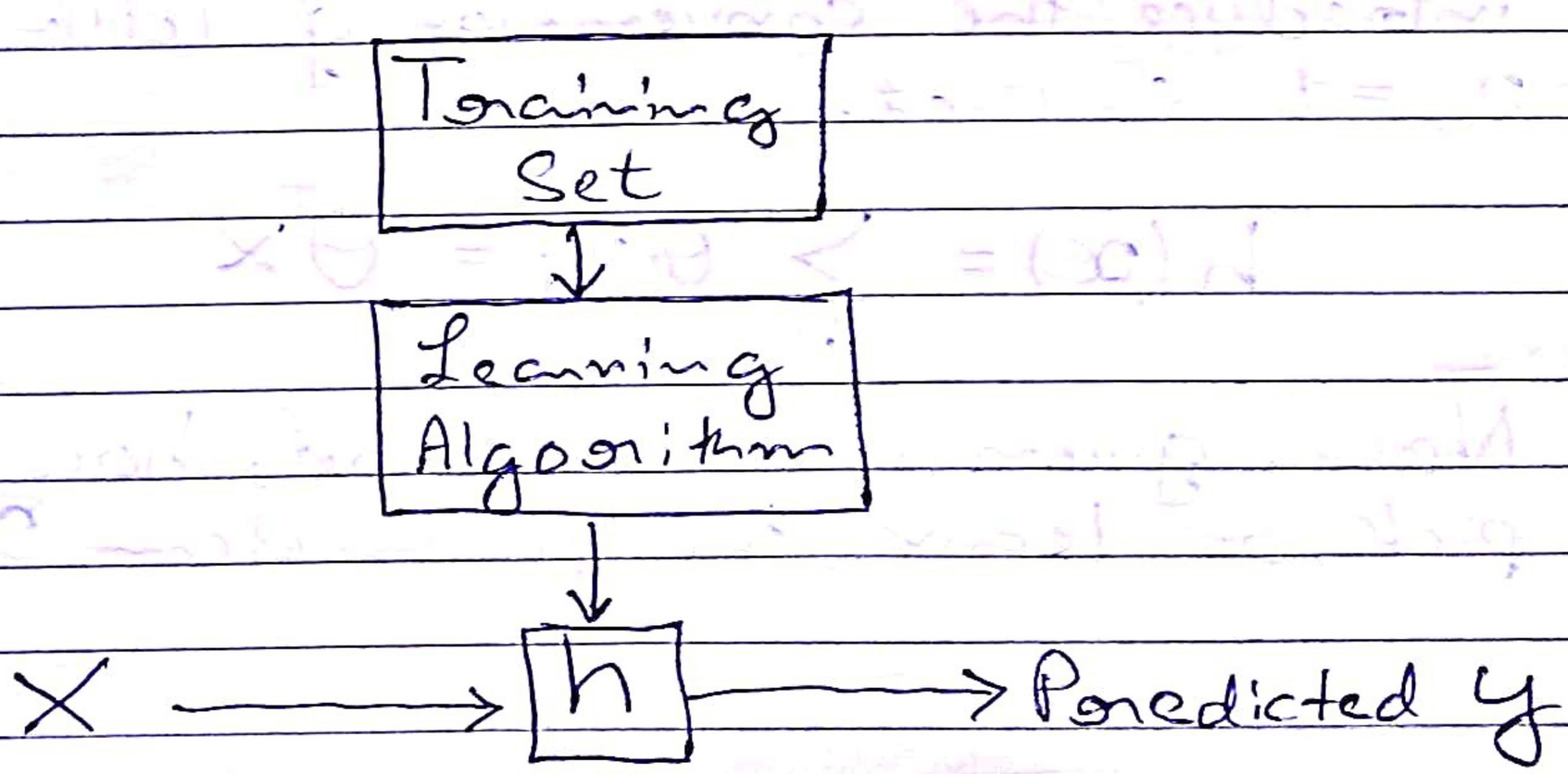
Y ⇒ denotes the Space of output

m ⇒ # training example

★ **Supervised learning**

"Given a training set, learn a function $h: X \mapsto Y$, so that $h(x)$ is a "good" predictor for the corresponding value of y"

↳ For historical reasons, this function h is called a **hypothesis**.



Date
Page

Linear Regression

⇒ To perform supervised learning, we must decide how we're going to represent functions/hypotheses h in a Computer.

⇒ As an initial choice, lets say we decide to approximate y as a linear function of x:

$$h_\theta(x) = \theta_0 + \theta_1 X_1 + \theta_2 X_2$$

$$\forall\, x = R^2$$

⇒ Here, the θ_i's are the parameters parameterizing the space of linear functions mapping X to Y.

⇒ To simplify our notation, we also introduce the convention of letting $x_0 = 1$ so that,

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

⇒ Now, given a training set, how do we pick, or learn, the parameters θ?

⇒ One reasonable method seems to be to make $h(x)$ close to y, at least for the training examples we have.

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

★ LMS algorithm

⇒ We want to choose θ so as to minimize $J(\theta)$.

⇒ To do so, lets use a search algorithm that starts with some initial "guess" for θ and that repeatedly changes θ to make $J(\theta)$ smaller, until hopefully we converge to a value of θ that minimizes $J(\theta)$

⇒ Let us consider the gradient descent algorithm.

⇒ Start with some initial θ, and repeatedly perform the update

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

for $j = 0, \cdots n$

$\alpha \Rightarrow$ Learning rate

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

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

$$\frac{\delta}{\delta \theta_j} \sum_{k=0}^{n} \theta_k x_k^{(i)}$$

$$= x_j^{(i)}$$

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

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

⇒ This rule is called the LMS update rule and is also known as the Widrow-Hoff learning rule.

⇒ This method looks at every example in the entire training set on every step, and is called batch gradient descent.

Data
Page

OM Student Notebooks

⇒ The cost function $J(\theta)$ is a convex quadratic function, So it only has one global optimum.

↳ Hence gradient descent will always converge to global optimum.

⇒ There is an alternative to batch gradient descent that also works very well.

for i = 1 to m {

$$\theta_j := \theta_j - \alpha \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)} \quad \{\text{FOR EVERY } j\}$$

}

⇒ This algorithm is called stochastic gradient descent.

⇒ Batch gradient descent has to scan through the entire training set before taking a single step.

{A costly operation if m is large}

⇒ Stochastic gradient descent can start making progress right away, and continue to make progress with each example it looks at.

Date
Page

⇒ On the down side, Stochastic gradient descent may never converge to the minimum, and the parameters θ will keep oscillating around the minimum of $J(\theta)$.

↳ But in practice most of the values near the minimum will be reasonable good approximations to the true minimum.

★ Normal equation (Least Square Solution)

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

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

$(m \times n+1)$, $(n+1 \times 1)$

$\Rightarrow$ Then, $\begin{bmatrix} h(x^{(1)}) \\ h(x^{(2)}) \\ \vdots \\ h(x^{(m)}) \end{bmatrix} = X\theta$

$\Rightarrow$ Let $y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$

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

Date
Page

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

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

$$= X^T X\theta - X^T y = 0$$

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