Supervised learning

DS 4400
Instructor: Ehsan Elhamifar

(A) Regression

Given input / feature / attribute vectors $\{x_i \in \mathcal{X}\}_{i=1}^N$ and associated numeric values $\{y_i \in \mathbb{R}\}_{i=1}^N$, find a function / hypothesis $h: \mathcal{X} \longrightarrow \mathcal{Y}$ so that $h(x_i) \simeq y_i, \forall i$

Ex) We want to predict the price of house

each house x_i = (# bedrooms, square footage, townhouse (0) or apartment (1)) has a price $y_i = \$$

* What is our data? $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ → training samples

* How to choose h? Equivalently, what is the hypothesis space?

In "linear regression" $h(x_i) = \theta^T x_i$ (unknown, must be learned from D.) linear combination of features / attributes / covariates

Eg, square footage might be more important in price than # bedrooms, etc

* How to measure closeness of $h(x_i)$ to y_i? I.e., what objective function to choose?

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \ell(h(x_i), y_i)$$

(ℓ: loss / distance function)

A popular objective / cost function is the $(\ell_2\text{-norm})^2$

$$\ell(h(x_i), y_i) = (h(x_i) - y_i)^2$$

Putting the two together, we want to minimize

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - h_\theta(x_i))^2 = \sum_{i=1}^{N} (y_i - \theta^T x_i)^2 = J(\theta)$$

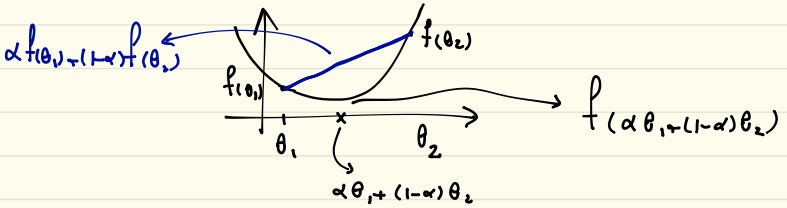
* How to solve for θ ?

Let's write this in matrix form $x_i \in \mathbb{R}^d$, $X = \begin{pmatrix} x_1^T \\ x_2^T \\ \\ x_N^T \end{pmatrix} \in \mathbb{R}^{N \times d}$, $Y = \begin{pmatrix} y_1 \\ y_2 \\ y_N \end{pmatrix} \in \mathbb{R}^N$

$$J(\theta) = \sum_{i=1}^{N} (y_i - \theta^T x_i)^2 = \| Y - X\theta \|_2^2$$

* Digression $J(\theta)$ is called "convex" function iff

$$\forall \theta_1, \theta_2 \text{ and } \forall \alpha \in [0,1] \quad J(\alpha\theta_1 + (1-\alpha)\theta_2) \leq \alpha J(\theta_1) + (1-\alpha) J(\theta_2)$$



In other words, the function always lies below the segment connecting any of its two points

+ If $J()$ is convex, then a minimizer of $J()$ can be found by setting its derivative to zero

$$\left.\frac{\partial J}{\partial \theta}\right|_{\theta_{min}} = 0 \quad \text{(gradient is zero)}$$

$$\begin{cases} \frac{\partial\, a^T\theta}{\partial\theta} = a \\ \frac{\partial\, \theta^T A\theta}{\partial\theta} = 2A\theta \\ \|a-b\|_2^2 = \|a\|_2^2 - 2a^Tb + \|b\|_2^2 \end{cases}$$

→ Back to regression is $J(\theta) = \|Y - X\theta\|_2^2$ convex ? YES

ⓐ check using first def ✓ ⓑ $\nabla^2 J(\theta) = 2X^TX \geq 0$ ✓ (X is $n \times d$)

How to find optimal θ (minimizer of $J(\theta)$) ?

① Analytical solution $\left.\frac{\partial J(\theta)}{\partial\theta}\right|_{\theta_{min}} = 0$

$$\nabla J(\theta) = 2X^T(X\theta - Y) = 0 \longrightarrow X^TX\theta = X^TY \longrightarrow \theta = (X^TX)^{-1}X^TY$$

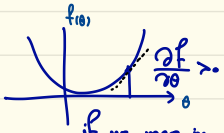
↳ thus should be invertible need $N \geq d$!

Advantage solution in one shot !

Possible drawbacks need $N \geq d$ # points $\geq$ # features

if d large need large training, also inverting $X^TX \in \mathbb{R}^{d\times d}$ costly ! $O(d^3 + d^2N)$

② Gradient descent algorithm



if we move in positive direction, $f\uparrow$

⇒ To minimize, move in direction of $-\frac{\partial f}{\partial\theta}$!

– start with $\theta^{(0)} \in \mathbb{R}^d$ (eg $\theta^{(0)} = 0$ or random vector)

– At iteration k, update θ by (until $\|\theta^{(k)} - \theta^{(k-1)}\| \leq \varepsilon$)

$$\theta^{(k)} = \theta^{(k-1)} - \rho \left.\frac{\partial J(\theta)}{\partial\theta}\right|_{\theta^{(k-1)}}$$

↳ learning rate ↳ gradient of J wrt θ

Here $\theta^{(k)} = \theta^{(k-1)} - \rho\, X^T(X\theta - Y)$ ⟶ Batch gradient descent

Advantage less computations (still costly if $N\uparrow \to O(Nd)$)

disadvantage can be slow (convergence rate depends on ρ and X^TX)

Alternatively

- start with $\theta^{(0)} \in \mathbb{R}^d$ (eg $\theta^{(0)} = 0$ or random vector)
- At iteration k, select one point at random, update θ by (until $\|\theta^{(k)} - \theta^{(k-1)}\| \leq \varepsilon$)
  - $\hookrightarrow x_i$

$$\theta^{(k)} = \theta^{(k-1)} - \gamma\, x_i (x_i^T \theta - y_i) \longrightarrow \text{Stochastic gradient descent}$$

Advantage computationally very efficient $\rightarrow O(d)$

disadvantage can be slow

many empirical studies show it works well

DS 4400
Instructor: Ehsan Elhamifar

Regression:

Given training samples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $x_i \in \mathbb{R}^d$ is a feature vector ($\in \mathcal{X}$), $y_i \in \mathbb{R}$ is the desired output (called response) ($\in \mathcal{Y}$)

Goal find an approximate function (hypothesis) $h: \mathcal{X} \to \mathcal{Y}$ so that $h(x_i) \approx y_i$, $\forall i = 1, \ldots, N$

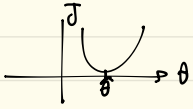
Hypothesis? $\Rightarrow$ Linear Regression $h_\theta(x) = \theta^T x$ for $\theta \in \mathbb{R}^d$, i.e., response is a linear function of unknown parameters

Cost function $\min_\theta J(\theta) \triangleq \sum_{i=1}^N (y_i - h_\theta(x_i))^2 = \sum_{i=1}^N (y_i - \theta^T x_i)^2$

we showed $J(\theta) = \sum_{i=1}^N (y_i - \theta^T x_i) = \| Y - X\theta \|_2^2$

where $X = \begin{pmatrix} x_1^T \\ \vdots \\ x_N^T \end{pmatrix} \in \mathbb{R}^{N \times d}$ and $Y = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \in \mathbb{R}^N$

We saw $J(\theta)$ is convex $\longrightarrow$ allows to be minimized wrt



① $\nabla J(\theta) = \frac{\partial J(\theta)}{\partial \theta} = 2X^T(X\theta - Y) = 0 \longrightarrow \hat{\theta} = (X^T X)^{-1} X^T Y$

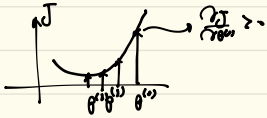
The strategy of $J(\cdot)$ is diff wrt θ!

$X \in \mathbb{R}^{N \times d} \Rightarrow X^T X \in \mathbb{R}^{d \times d}$ to be invertible needs $N \gg d$

Adv $\hat{\theta}$ in one line (closed form)

Dis $\hat{\theta}$ computation when d is large (# pixels, # words) is costly $O(d^3 + dN)$

② Gradient descent algorithm



- start with $\theta^{(0)} \in \mathbb{R}^d$ (eg $\theta^{(0)} = 0$ or random vector)
- At iteration k, update θ by (until $\|\theta^{(k)} - \theta^{(k-1)}\| \leq \varepsilon$ or $\|\nabla J(\theta^{(k)})\| \leq \varepsilon$)

$$\theta^{(k)} = \theta^{(k-1)} - \rho \left.\frac{\partial J(\theta)}{\partial \theta}\right|_{\theta^{(k-1)}}$$

ρ: learning rate; $\frac{\partial J(\theta)}{\partial \theta}$: gradient of J wrt θ

Here $\theta^{(k)} = \theta^{(k-1)} - \rho X^T (X\theta^{(k-1)} - Y)$ ⟶ Batch gradient descent

Advantage: less computations ($X\theta^{(k-1)}$, $X^T(X\theta^{(k-1)})$ ⟶ $O(Nd)$ still depends on N) ($N \uparrow \Rightarrow$ costly)

disadvantage: need all data at each iteration (convergence rate depends on ρ and X^TX)

③ Stochastic Grad Descent

- start with $\theta^{(0)} \in \mathbb{R}^d$ (eg $\theta^{(0)} = 0$ or random vector)
- At iteration k, select one point at random, update θ by (until $\|\theta^{(k)} - \theta^{(k-1)}\| \leq \varepsilon$)

↳ x_i

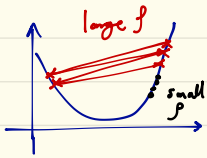
$$\theta^{(k)} = \theta^{(k-1)} - \rho\, x_i (x_i^T\theta - y_i) \longrightarrow$$

Replace $X = \begin{pmatrix} x_1^T \\ x_N^T \end{pmatrix}$ by $X = x_i$ for a randomly selected point !

Advantage: computationally very efficient ⟶ $O(d)$

disadvantage: can be slow (however, generally as slow as Batch GD)

many empirical studies show it works well

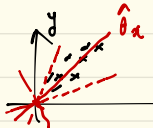


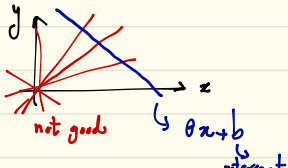
$\sum_k \rho_k = \infty$, $\sum \rho_k^2 < \infty$ Robbins-Monro Condi

e.g. $\rho_k = \frac{1}{k+0.1}$

One limitation of what we showed $h_\theta(x) = \theta^T x$ is that it is a linear function of x, which can be restrictive

Ex) $\{(x_i \in \mathbb{R}, y_i \in \mathbb{R})\}_{i=1}^N$





To account for such affine transformations let feature vector be $\begin{pmatrix} x_i \\ 1 \end{pmatrix} \to \tilde{x}_i \in \mathbb{R}^{d+1}$ $\quad \tilde{X} = \begin{pmatrix} \tilde{x}_1^T \\ \tilde{x}_N^T \end{pmatrix}$, $y = \begin{pmatrix} y_1 \\ y_N \end{pmatrix}$

and learn $\tilde{\theta} \in \mathbb{R}^{d+1} = \begin{pmatrix} \hat{\theta} \\ \hat{b} \end{pmatrix}$

Another limitation of this is its linear dependence on features / covariates $x_i = \begin{pmatrix} x_{i1} \\ x_{id} \end{pmatrix} \in \mathbb{R}^d$

$$h_\theta(x_i) = \sum_{j=1}^{d} \theta_j x_{ji}, \quad \text{if } x_{ji} \text{ increases by } \delta_j \text{ then } h_\theta(x_i) \text{ changes by } \theta_j \delta_j !$$

This can be restrictive !

Ex) Price Regression $x_i = \begin{pmatrix} \#\text{bedrooms} \\ \text{sq ft} \\ \text{area} \end{pmatrix}$, y_i = price of house i

if #bdrooms $1 \to 2$ I expect higher increase in price than $7 \to 8$, i.e "dimmishing return" property $\longrightarrow$ makes sense to consider quadratic features

$$x_i = \begin{pmatrix} \#\text{bedrooms} \\ (\#\text{bedrooms})^2 \\ \text{sq ft} \end{pmatrix}$$

$2\alpha(2-1) - 1(2^2 - 1^2) = 1\beta$ 1bd $\to$ 2bd : price increase: 1β

$2\alpha(8-7) - 1(8^2 - 7^2) = 5$ 7bd $\to$ 8bd : price increase: 5

More generally, we can consider $\varphi(x_i)$ instead of x_i, where $\varphi: \mathcal{X} \to \mathcal{H}$ is a complex nonlinear function $\Longrightarrow$ Basis function expansion

Ex) $\varphi(x) = (x_1 \; x_1^2 \; x_1^n \quad x_d \; x_d^2 \; x_d^n \; 1)^T \rightsquigarrow$ n-degree polynomials

Q) Is this still a linear regression ?

Yes, since still the function is linear in θ $\quad h_\theta(x) = \theta^T \varphi(x)$

Nonlinear in x but linear in θ $\longrightarrow$ Every thing we did before, we can still do by using $\Phi = \begin{pmatrix} \varphi(x_1)^T \\ \varphi(x_N)^T \end{pmatrix} \in \mathbb{R}^{N \times d'}$