Lecture 4: Linear Regression (contd.)

19 January 2024 11:08

Linear Regression reduces to the following optimisation problem

$$\min_w E(w, D) = \|Xw - y\|^2$$

$\|Ax-y\|^2$ is a convex fn $\left(\frac{d^2f}{dx^2} \geq 0\right)$

$$w^* = (X^TX)^{-1}X^Ty \quad \text{if} \quad X^TX \text{ is invertible}$$

Otherwise we can solve for

$$\nabla_w E = 0 \Rightarrow X^TXw - X^Ty = 0$$

Recall: A linear system of equations can be written as $Ax = b$

Now if $A = [a_1, \cdots, a_d]$ where a_j is column vector

then $Ax = x_1a_1 + \cdots + x_da_d$ where $x = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}$

Note that $Ax = b$ has a soln iff $b \in C(A)$: Column space of A

Using this we can deduce that $X^TXw = X^Ty$ always has a soln as $X^Ty \in C(X^TX)$, hence can be solved to find w^*.

each column of $X^TX \in C(X^T)$?

$y \in C(X)$

$X^Ty \in C(X^TX)$?

$$\begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} y = \begin{bmatrix} x_1^Ty \\ \vdots \\ x_n^Ty \end{bmatrix} \quad \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} (x_1 \cdots x_n)$$

$$X^T(\underbrace{\alpha_1x_1, \cdots \alpha_n x_n}_{y}) \quad \begin{bmatrix} x_1^Tx_1 & \cdots \\ \vdots & \\ x_n^Tx_1 & \\ \downarrow & \\ X^Tx_1, X^Tx_2 & \cdots \end{bmatrix}$$

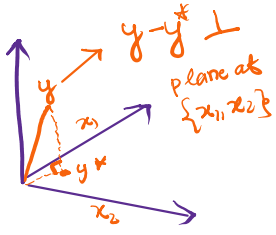
Case I: $Xw^* = y$

Case II: Can't find a w^*: $X^TXw - X^Ty = 0 \Rightarrow X^T(Xw - y) = 0$

$Ax = b \rightarrow$ No soln

$Ax = b'$

x^* (best approx soln as squared error is min).



b' is projection of b on $C(A)$

Regression model with basis functions

Take $d = 1$: $\hat{y}_i = w_0 + w_1 x_i + (w_2 x_i^2 + \dots w_m x_i^m)$ (x_i's are scalars)

we define basis fns:

$$\phi_0(x_i) = 1$$
$$\phi_1(x_i) = x_i$$
$$\vdots$$
$$\phi_m(x_m) = x_i^m$$

$$\Rightarrow \quad \hat{y}_i = \sum_{j=0}^{m} w_j \phi_j(x_i)$$

For d-dimensions, $\hat{y}_i = \sum_{j=0}^{m} \phi_j(x_i)\, w_j$, $m >> d$

where, $\Phi = \begin{bmatrix} \phi_0(x_1) & \dots & \phi_m(x_1) \\ \vdots & & \\ \phi_0(x_n) & \dots & \phi_m(x_n) \end{bmatrix}_{n \times (m+1)}$, $w = \begin{pmatrix} w_0 \\ \vdots \\ w_m \end{pmatrix}_{(m+1)\times 1}$, $y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$

$$\Leftrightarrow \quad \text{argmin } \|\Phi w - y\|^2 \quad \text{with} \quad w^* = (\Phi^T \Phi)^{-1} \Phi^T y$$

Probabilistic model of Linear Regression

In real life cases some error/randomness is also observed.

$$\therefore \; y_i = w^T x_i + \epsilon_i \text{ (noisy linear model)}$$

This leads to a propabilistic regression model.

In a noisy linear model, we have

i) parameters : w

ii) noise : $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, with $\text{cov}(\epsilon_i, \epsilon_j) = 0 \;\; \forall i \neq j$ (iid's)

iii) Data set: $D = \{(x_i, y_i)\}_{i=1}^{n}$

Our task is to estimate w from this probabilistic model using following methods: (i) Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE)

Maximum Likelihood Estimation (MLE)

A set of independent and identically distributed observations $\{y_1, \dots y_n\}$ are generated by a probabilistic model parametrised by θ

$$y_i \sim P(y|\theta)$$ → likelihood function

Log likelihood: $\log\left(P(y|\theta)\right)$ (useful as log is monotonic fn, mathematically easier, numerical advantage)

We've:

$$\theta_{MLE} = \arg\max L(\theta) = \arg\max \sum_{j=1}^{n} \log P(y_i|\theta)$$

Ex: Toss a coin 'n' times. Each one of the outcome is a binary Random Variable with Bernoulli distribution with

$$P(y_j|\theta) = \theta^{y_j}(1-\theta)^{1-y_j}$$

$$L(\theta) = \log P(y|\theta) = \log\left(\prod_{j=1}^{n} P(y_j|\theta)\right)$$

$$= \sum_{j=1}^{n} \log P_j(y_i|\theta) = \sum_{j=1}^{n}\left(y_j \log\theta + (1-y_i)\log(1-\theta)\right)$$

$$\Rightarrow \hat{\theta}_{max} = \frac{\sum y_i}{n}$$

MLE for regression

$$y_j = w^T x_j + \epsilon_j \sim N(0, \sigma^2)$$

$$y_j \sim N(w^T x_j, \sigma^2)$$

$$P(y_j|x_j, w) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(y_j - w^T x)^2}{2\sigma^2}\right\}$$

$$L(w) = (\text{const.}) - \sum_{j=1}^{n} \frac{(y_j - w^T x_j)^2}{2\sigma^2}$$

$$\Rightarrow \arg\max_w L(w) = \arg\min_w \sum_{j=1}^{n} (y_j - w^T x_j)^2 = w^*$$

(This is same as deterministic regression)

$$\equiv \arg\min \|Xw - y\|^2$$

$\equiv \text{argmin} \; \|Xw - y\|^2$ (This is same as deterministic regression)

We know $w^* = (X^T X)^{-1} X^T y$, but this needs inverse which is computationaly costly.

So we adopt an algorithmic way to optimize $\|Xw - y\|^2$, called Gradient Descent.

- $w \leftarrow w_0$
- Repeat until convergence: $\|\nabla E(w)\| < \varepsilon$ → $E(w) < \varepsilon$ (very small)

  $w \leftarrow w - \eta \nabla_w E$ (η: learning rate)