Answer 1 :

2-9-19

HW-7

1. y_i is scalar

$x_i \in \mathbb{R}^d$

N samples

closed form expression:

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}_{n\times 1} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n\times 1} - \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}_{n\times d} W_{d\times 1}$$

$$E = Y - XW$$

For $y_i \in \mathbb{R}^p$, equation dimensions change

$$\begin{bmatrix} \epsilon_1^T \\ \epsilon_2^T \\ \vdots \\ \vdots \end{bmatrix}_{n\times p} = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{bmatrix}_{n\times p} - \begin{bmatrix} x_1^T \\ \vdots \\ \vdots \\ x_n^T \end{bmatrix}_{n\times d} W_{d\times p}$$

$$E = Y - XW$$

$$J = \frac{1}{N} E^T E = \frac{1}{N} E^T_{n\times p} E_{p\times n}$$

$$J(W) = \frac{1}{N} [Y_{n\times p} - XW]^T [Y_{n\times p} - XW]$$

$$J(W) = \frac{1}{N} (Y^T Y + W^T X^T X W \cancel{- 2Y^T XW} - 2W^T(X^T Y))$$

Differentiate with 'w' & equate to zero.

$$2X^T X W - 2X^T Y = 0$$

$$W_{d\times p} = (X^T X)^{-1}_{d\times d} X^T_{d\times n} Y_{n\times p}$$

closed Form:

Answer 2 :

2.e

a) Iterative Mean and covariance

After receiving 'k' samples online,

Mean: $\cancel{M}$ $\mu_k = \mu_{k-1} + (x_k - \mu_{k-1})/k$

Derivation

Mean for $k-1$ samples is μ_{k-1}, so for k.

$$\mu_k = \frac{(k-1)\mu_{k-1} + x_k}{k}$$

$$= \mu_{k-1} + (x_k - \mu_{k-1})/k$$

covariance: $S_k = S_{k-1} + (x_k - M_{k-1}) * (x_k - M_k)$

Algorithm:

steps -

1) Initialize $M_1 = x_1$ & $S_1 = 0$, data (D) = [--]

2) For $i = 2$ to n

$$M_i = M_{i-1} + (D_i - M_{i-1})/i$$

$$S_i = S_{i-1} + (D_i - M_{i-1}) * (D_i - M_i)$$

Where M is mean & S is standard deviation

$$\text{variance}(S^2) = \frac{S_k}{k-1}$$

b) For 'M' most recent samples ($M < i < N$)

We could use a Queue for storing the 'M' most recent samples. (First-in-first-out).

1) Initialize Queue $| x_1 | x_2 | \ldots | x_M |$

2) For $i = M+1$ to N:

~~Enqueue~~ Dequeue()

Enqueue (x_{M+1})

Mean = Mean (Queue) ; variance = variance (Queue)

Answer 3 :

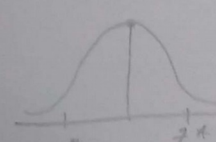
3. > Explain why 'θ' being small or large leads to ineffective algorithms.

Reasons -

If θ is small, the training set would not be good enough to generalize over noisy ~~[illegible]~~ examples.

If θ is large, it would lead to overfitting. ~~This would not include outliers as well.~~
This condition would lead to acceptance of outliers as inliers.

> We generally assume that distributions are usually 'Normal distributions' in nature.



$$z^* = \frac{\sigma}{\sqrt{n}}$$

The confidence intervals in the range of $[\mu - kz^*, \mu + kz^*]$ can help remove a certain portion data as outliers.

Here, the human defined parameter is 'What percentage of data is to be considered as outlier data?'

<u>Another method</u>

To get a ~~[illegible]~~ set of k-NN to detect outliers.
Then the average variance of outliers (only higher variances) will give the square of θ.

Hence, the average standard deviation of these can be used.

Here the ~~param~~ human-defined parameter would be 'k'.

Another method

Adding a regularization term to the loss function helps prevent overfitting to outliers.

We can do dimensionality reduction.
This would help distinguish outliers better & follow it up with the earlier methods

All are methods regularization.