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

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

$$E = Y - XW$$

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

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

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

Differentiate with 'w' & equate to zero.

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

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

closed Form:

Answer 2 :

2.

a) Iterative Mean and covariance

After receiving 'k' samples online,

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

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 $\boxed{x_1 \mid x_2 \mid \ \ - - - \mid x_M}$

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

~~Enqueue~~ Dequeue()

Enqueue (x_{M+1})

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

For online covariance between a pair of variables.

$$\frac{1}{k-1}\sum_{i=1}^{k}(x_i-\mu_x)(y_i-\mu_k) = \frac{1}{k-1}\left[\sum^{k-1}\left(x_i-\mu_{k-1}-\frac{x_k-\mu_{x,k-1}}{k}\right)\left(y_i-\mu_{k-1}-\frac{y_k-\mu_{y,k-1}}{k}\right) + (x_k-\mu_k)(y_k-\mu_k)\right] \quad -(1)$$

Using the property,

$$cov(X+A, Y) = cov(X, Y) + cov(A, Y)$$

$$cov(X, Y) = cov(Y, X).$$

Expanding & applying the properties to the first part of eq. (1).

$$\cancel{cov}\; cov_{k-1} + \cancel{cov}\; \frac{y_k-\mu_{y,k-1}}{-k}\sum^{k-1}(x_i-\mu_{k-1}) - \frac{x_k-\mu_{x,k-1}}{k}\sum^{k-1}(y_i-\mu_{y,k-1}) + \sum^{k-1}\left(\frac{x_k-\mu_{x,k-1}}{k}\right)\left(\frac{y_k-\mu_{y,k-1}}{k}\right)$$

$$\frac{k-2}{k-1}cov_{k-1} - \frac{y_k-\mu_{y,k-1}}{k}\times 0 - \frac{x_k-\mu_{x,k-1}}{k}\times 0 + \frac{k-1}{k^2}(x_k-\mu_{x,k-1})(y_k-\mu_{y,k-1})$$

Final equation

$$cov = \frac{k-2}{k-1}cov_{k-1} + \left[\frac{k-1}{k^2}(x_k-\mu_{x,k-1})(y_k-\mu_{y,k-1}) + (x_k-\mu_{xk})(y_k-\mu_{yk})\right] \times \frac{1}{k-1}$$

<u>Answer 3</u> :

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

Reasons -

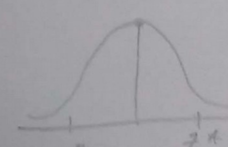
If σ is small, the training set would not be good enough to generalize over noisy ~~exam~~ examples.

If σ 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 ~~list~~ set of k-NN to detect outliers.

Then the average variance of outliers (only higher variances) will give the square of σ.

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.