TUAN TRAN
A20357888
CS584 - Summer 2020

INTRODUCTION:

1. Supervised learning is training a model given target values to compare against
   Unsupervised learning is training without being provided target values

2. Generative learning is learning to estimate distribution parameters => can do classification after learning such parameters
   Discriminative learning is learning to find the parameter of a discriminant function

3. Regression is supervised learning where the target value is continuous
   Classification is supervised learning where the target value is discrete

4. Parametric techniques: algorithms that involve learning parameters of the model
   Non-parametric: algorithms that stores training examples in order to perform inference, for example: perform classification using nearest neighbor

5. Offline algorithm: 2 separate pipelines where we train offline and performs inference online
   Online algorithm: model is updated continuously to account for changes in data distribution

6. Training error: error observed when we apply the trained model on the training data
   Testing error: error observed when we apply the trained model on testing data
   => Training error is to see how well the model fits the training data and testing error is to see how well the model generalizes to unseen test data

7. Process of performing K-fold cross validation:
   + Split data to K parts
   + Leave one part out and train on remaining K-1 parts
   + Repeat K times and compute average error
   + Tune model and repeat
   + Train the final tuned model on all examples

8. Q8:
   + Categorical features are features that maybe string and may have categories to them. For example: small, medium and large. These features need to be converted to model-readable format, for example, using one-hot encoding
   + Numerical features are features that are numbers and can be discrete or continuous

+ Ordinal features are categorical features that have order to them, ex: small, medium large since small < medium < large
+ Nominal features are categorical features with no orders => ex: color like red, green, orange => no natural orders to them

<!!!!!> NOTE: Regression and Kernel Methods part starts from next page

Tuan Tran
A20357888
CS584 - Summer 2020

~~Intro~~ Regression:

1, model.

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

$$= \theta^T z$$

$$\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix} \quad z = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

2, Objective Function $= J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$

To find the model parameters, we find θ^* where $J(\theta)$ is minimum

$$\rightarrow \theta^* = \underset{\theta}{\operatorname{argmin}}\, J(\theta)$$

3, we have $z^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$

~~We rewrite obj function in matrix form:~~ $\rightarrow$ ~~$J(\theta) = \frac{1}{2}$~~

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} - y^{(i)} \right)^2$$

$$\rightarrow \begin{cases} \frac{\partial J}{\partial \theta_0} = 0 \\ \frac{\partial J}{\partial \theta_1} = 0 \\ \frac{\partial J}{\partial \theta_2} = 0 \end{cases} \rightarrow \boxed{\begin{cases} \sum_{i=1}^{m} \theta_0 + \sum_{i=1}^{m} (\theta_1 x_1^{(i)}) + \sum \theta_2 x_2^{(i)} = \sum_{i=1}^{m} y^{(i)} \\ \sum \theta_0 x_1^{(i)} + \sum \theta_1 (x_1^{(i)})^2 + \sum \theta_2 x_2^{(i)} x_1^{(i)} = \sum_{i=1}^{m} x_1^{(i)} y^{(i)} \\ \sum \theta_0 x_2^{(i)} + \sum \theta_1 x_1^{(i)} x_2^{(i)} + \sum \theta_2 (x_2^{(i)})^2 = \sum x_2^{(i)} y^{(i)} \end{cases}}$$

CS Scanned with CamScanner

4. $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = 1 + 8 + 15 = 24.$

5. let $Z = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1^1 & x_1^2 & \cdots & x_1^m \\ \vdots & & & \\ x_n^1 & & \cdots & x_n^m \end{bmatrix}$

$= \begin{bmatrix} | & | & & | \\ x^1 & x^2 & \cdots & x^m \\ | & | & & | \end{bmatrix}$; let $\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix}$

$\rightarrow Z$ has dimension $(n+1) \times m$

$- J(\theta) = \frac{1}{2} (\theta^T Z - Y^T)^T (\theta^T Z - Y^T)$

(Y has dimension $m+1$)
$\hookrightarrow Y^T$ has dimension $1 \times m$ since $\theta^T Z$ has dimension $1 \times m$

$\nabla J(\theta) = (\theta^T Z - Y^T) Z^T = 0$

$\rightarrow \theta^T Z Z^T - Y^T Z^T$

$\rightarrow \theta^{*T} = (Z Z^T)^{-1} Y^T Z^T$

$\rightarrow$ In my solution, $Z \in \mathbb{R}^{(n+1) \times m}$ and the values in first column of this matrix is $x^1 = \begin{bmatrix} 1 \\ x_1^1 \\ \vdots \\ x_n^1 \end{bmatrix}$

6. Non-linear regression can be performed by using polynomial regression where we extend the features by raising them to some power

CS Scanned with CamScanner

The form of the solution ~~is~~ looks the same, but the difference lies in matrix Z. Say ~~x, is~~ x is the only feature, &

In linear we have, $Z = \begin{bmatrix} 1 & x^{(1)} \\ \vdots & \vdots \\ 1 & x^{(m)} \end{bmatrix}$

In polynomial, we have $Z = \begin{bmatrix} 1 & x^{(1)} & \dots & (x^{(1)})^n \\ \vdots & & & \\ 1 & \dots & \dots & (x^{(m)})^n \end{bmatrix}$

→ dimension of Z in this case is $m \times (n+1)$

First Column contains ~~vect~~ all 1s.

7,

- fitting error is the error observed ~~when~~ on the training data when we train the model
- generalization error is the error observed ~~when on~~ when we apply our trained model on unseen data.
- We can reduce fitting error by using a more complex model to fit data

We can reduce generalization error by either adding more data or use regularization ~~techniques~~ techniques.

- Theoretically, our expected hypothesis error is the sum of Variance of hypothesis ~~and~~ (variance error) and how far the expected value of hypothesis is from actual value (bias error)

↳ If we decrease variance error, we may increase bias error and vice versa.

8,

The purpose of Ridge Regression is to add a regularization term → lower complexity → lower weights → better generalization

CS Scanned with CamScanner

- Ridge Objective Function $J(\theta) = \sum (\hat{y}^{(i)} - y^{(i)})^2 + \lambda \|\theta\|^2$

In the Solution obtained using Ridge, we have: $\theta^* = (Z^T Z + \lambda I)^{-1} Z^T Y$

→ we add λ to the diagonal elements of $Z^T Z$

9,

~~RSE~~ $RSE = \frac{1}{m} \sum \frac{(\hat{y}^{(i)} - y^{(i)})^2}{(y^{(i)})^2}$

→ With RSE, ~~we~~ it is easier to make sense of the loss value since we basically put L2 loss in the scale of the output ($(y^{(i)})^2$ division)

→ We ~~we~~ will know if say ~~S~~ RSE = 5 is considered good or bad since the scale of actual output is taken into account.

10,

- LWR is a non parametric regression method ~~whereas~~ where data is stored and used during inference. It also fit multiple local models at each local neighborhoods.

Whereas Ordinary Regression is parametric; ~~it~~ it just needs the learned weights for inference and not the data. It also fits a single global model to all the training data.

- Advantage.
  - +, more fine grained prediction and better fit since it looks at most relevant examples w.r.t the new instance.
  - +, Have an regularization-like effect since local models are much simpler than global model.

~~Disadvantage~~

- Solution: $\theta_x = (Z^T W_x Z)^{-1} Z^T W_x Y$

CS Scanned with CamScanner

where W_x is the weights of points w.r.t ~~x~~ x and θ_x is the vector parameters of the local model, whereas θ of ordinary Regression is parameters of the global model

~~Kernel~~

Kernel methods:

1,

We want to move to higher dimensional space because.

+, linear relationship ~~is pose~~ may be possible in higher dimensional space but not in the lower one.

+, We can increase the capacity of the model since no of parameters is increased.

2,

- Primal:

$$\begin{cases} J(\theta) = \sum \left(\theta^T x^{(i)} - y^{(i)}\right)^2 \\ \theta^* = \underset{\theta}{\arg\min}\, J(\theta) \end{cases}$$

-> $n+1$ unknowns: $\theta_0 \dots \theta_n$.

- Dual:

$$\theta = \sum_{i=1}^{m} \alpha_i x^{(i)}$$

$$\rightarrow \begin{cases} J(\alpha) = \sum_{i=1}^{m} \left[\left(\sum \alpha_j x^{(j)} \right)^T x^{(i)} - y^{(i)} \right]^2 \\ \alpha^* = \underset{\alpha}{\arg\min}\, J(\alpha) \end{cases}$$

-> m unknowns

CS Scanned with CamScanner

3,

the advantage is that we don't have to specify or find the basis function, which may be extremely inefficient to compute

4,

In matrix form:

$$\theta = X^T \alpha .$$

$$J(\theta) = (X\theta - y)^T (X\theta - y)$$

$$\to J(\alpha) = (XX^T\alpha - y)^T (XX^T\alpha - y)$$

let $G = XX^T$

$$\to J(\alpha) = (G\alpha - y)^T (G\alpha - y)$$

$$\to \nabla J(\alpha) = 2G^T(G\alpha - y) = 0$$

$$\to G^T(G\alpha - y) = 0$$

$$\to \alpha^* = (G^T G)^{-1} G^T y$$

5,

$$G = XX^T = \begin{bmatrix} x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & & \vdots \\ x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} x_1^{(1)} & \cdots & x_1^{(m)} \\ \vdots & & \vdots \\ x_n^{(1)} & \cdots & x_n^{(m)} \end{bmatrix} = \begin{bmatrix} — & x^{(1)T} & — \\ & \vdots & \\ — & x^{(m)T} & — \end{bmatrix} \begin{bmatrix} | & & | \\ x^{(1)} & \cdots & x^{(m)} \\ | & & | \end{bmatrix}$$

$$= \begin{bmatrix} x^{(1)T} x^{(1)} & \cdots & x^{(1)T} x^{(m)} \\ \vdots & & \\ x^{(m)T} x^{(1)} & \cdots & x^{(m)T} x^{(m)} \end{bmatrix} = \begin{bmatrix} K(x^{(1)}, x^{(1)}) & \cdots & K(x^1, x^m) \\ \vdots & & \vdots \\ K(x^m, x^1) & \cdots & K(x^m, x^m) \end{bmatrix}$$

Gram matrix is basically a similarity measurement matrix, it is a square matrix

CS Scanned with CamScanner

6,

Given α, we can compute ~~θ~~ θ by: $\theta = X^T\alpha$

No it is not ~~necessa~~ necessary since we can simply use α.

7,

- let $k(x,y)$ be a kernel function and ϕ be basis function

→ we can rewrite k in the form of.

$$k(x,y) = \phi(x)^T \phi(y)$$

→ Following this expression, ~~we can~~ an explicit representation for ϕ is not necessary

and thus we can compute k without having to define ϕ.

- Conditions for a ~~kernel~~ kernel func $k(x,y)$ to be valid:

1, $k(x,y) = c k_1(x,y)$

2, $k(x,y) = k_1(x,y) + k_2(x,y)$

3, $k(x,y) = k_1(x,y) \cdot k_2(x,y)$

4, $k(x,y) = e^{k_1(x,y)}$

where $k_1(x,y)$ and $k_2(x,y)$ are valid kernel func.

8,

$$k(x,y) = e^{-\frac{\|x-y\|_2}{2\sigma^2}} = e^{-\frac{1}{2\sigma^2}(x-y)^T(x-y)}$$

$$= e^{-\frac{1}{2\sigma^2}(x^Tx - 2x^Ty + y^Ty)}$$

$$= e^{-\frac{1}{2\sigma^2}x^Tx} \, e^{\frac{1}{\sigma^2}x^Ty} \, e^{-\frac{1}{2\sigma^2}y^Ty}$$

→ Using condition 1, 4 and 3 in order → Gaussian kernel is valid

CS Scanned with CamScanner