Machine learning

Cse 574

Project 2

Virinchi urivinti | UBNO: 50292192 | 10/10/18

Introduction :

In this project, we predict whether writers of the given data are same or different using linear regression ,logistic regression and nueral networks. The 4 different datasets are applied for each model and accuracies are compared.

Data pre-processing:

- ➔ For human data: The corresponding features of the combination of writers are stored in different csv files. So, in order to find all the features for different combinations and to combine them, we use dictionary. storing the human.features.csv file in a dictionary(hash_map) and features of each example of same pairs and different pairs is found using the dictionary. Now ,the data that we have is imbalanced data i.e we have many examples in different pairs than in same pairs. In order to balance the data we randomly select 791 pairs from different.pairs.csv and attach them row wise with the same.pairs.csv. Features are formed using two methods . subtracting the features of two writers and the other method is concatenating the features of two writers. Thus we have two datasets for human-extracted features of 9 and 18 coloumns respectively .
- ➔ For Gsc data : procedure followed in combinig the datasets is same as that of human data. For Gsc data we have 512 , 1024 features for subtracted and concatenated data respectively

Data partition : For each dataset we consider 80 percent of the data for training and 20 percent for testing and validation. 3 kinds of testing data is generated . Seen, unseen and random data . In seen testing data we only consider the writers that we came across in training set. Similarly in unseen testing data we only consider the the writers that are not a part of training set . And for random testing data we consider both seen and unseen writers .

Linear Regression:

Calculation of Design Matrix. Phi(x):
To calculate the basis function we use the vector form of the Gaussian function which is represented in the box.

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^\top \Sigma_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right)$$

- **x** = input data matrix having input features with dimension(1*9) or (1*18).
- **μj** : Is the center of the basis function ie.,mean of the phij(x) having input features with dimension(1*9) or (1*18)
- **μ:** The mean for each feature is different so in total we form 9 or 18 means in each cluster and a matrix is formed to store the mean of all the features and the matrix has a dimension of 10*9 or 10*18

- **Big-Sigma** = Covariance matrix of the input features with dimensions (9*9) or (18*18). The covariance matrix is the diagonal matrix because all other elements are zero as the features are independent of one another. In the covariance matrix, we also scale the values of variances with a factor so that we can have a better usage of the values. Here we have chosen 200 as the scaling factor. We also add some bias with value 0.2 to the big-sigma matrix in order to avoid the singular matrix.
- Now using the above equation we can get the phij() matrix. That is the design matrix:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{bmatrix}$$

- Here N=Length of the Training Data and M= Number of clusters. Thus the design matrix is formed.

Regularization:

The weights which have been initialized may get unbalanced and may not fetch the optimum weights. To eliminate this problem we add lambda term and penalize sum of squares of weights to find out the optimum weights. This will make the model little biased but reduces lot of variance(sum of squares).Biased term is lambda.

Now we calculate the optimum weights using Batch Gradient descent solution:

Batch Gradient descent :

In this method we define the cost function to be the sum of squares of differences between predicted and actual output.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

Where θ is set of weights. This cost function can visualized as n dimensional curve where n is the number of weights. Each set of weights correspond to a point on this cost function.Our goal is to find the minimum of the of this cost function. Minimum can be obtained by decreasing each weights iteratively until we reach the minimum value of J(θ). This is done by calculating the gradient of the function and decreasing the weights in the opposite direction of the gradient. Amount of decrement is determined by α (learning rate) .

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta).$$

Above update is performed for all weights (j=1,2……n) until minimum is obtained

Repeat until convergence

$$\theta_j := \theta_j + \alpha\sum_{i=1}^{m}\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)} \qquad (\text{for every } j).$$

This method of updating the weights has a disadvantage. for every iteration all the rows in the training data set are read. This will take a lot of time to run if the data set is huge. In such cases stochastic gradient descent is used which we repeatedly run through the training set, We used Mini batch SGD method to update the weights. The computation of gradient error can be done by matrix multiplication and dividing the data in batches largely out performs the speed of computing the gradient error., we update the parameters according to the gradient of the error with respect to that single training example only .

.

: Human concat

Lam,M	M=2	M=5	M=10	M=15
Lam =10^-5	0.547	0.556	0.978	0.5529
Lam = 0.05	0.563	0.524	0.987	0.546
Lam = 2	0.581	0.52	**0.451**	0.546
Lam = 100	0.5732	0.558	0.610	0.547
Lam = 10^2	0.5781	0.5592	0.810	0.5479

Human subtract

Lam,M	M=2	M=5	M=10	M=15
Lam =10^-5	0.522	0.56	0.98	0.5529
Lam = 0.05	0.5747	0.5628	0.94	0.546
Lam = 2	0.572	0.542	**0.495**	0.546
Lam = 100	0.5832	0.5681	0.75	0.557
Lam = 10^2	0.5881	0.5692	0.785	0.5579

GSC concat

Lam,M	M=2	M=5	M=10	M=15
Lam =10^-5	0.578	0.553	0.5128	0.5529
Lam = 0.05	0.5647	0.5528	0.509	0.546
Lam = 2	0.5627	0.5512	**0.421**	0.546
Lam = 100	0.5732	0.5581	0.5282	0.547
Lam = 10^2	0.5781	0.5592	0.5201	0.5479

GSC subtract

Lam,M	M=2	M=5	M=10	M=15
Lam =10^-5	0.5529	0.55	0.5528	0.5529
Lam = 0.05	0.5647	0.5528	0.499	0.501
Lam = 2	0.5627	0.5512	**0.441**	0.456
Lam = 100	0.5732	0.5581	0.5082	0.547
Lam = 10^2	0.5781	0.5592	0.5101	0.5479

Logistic regression :

In this project, we implement regularized logestic regression where we add the regularization term lambda times sum of squares of the coeffeicents to the cost function. We use cross entropy as our cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2.$$

Final cost function

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \quad \text{for } j \geq 1$$

Gradient with respect to coefficents

Similar to that linear regression we subtract the gradient of the cost function wrt to the coefficents repetedly until it converges.

Accuracy for testing : Human concat

Lam,lr	Lr=0.001	lr=0.01	lr=0.05	lr=0.1
Lam =10^-5	0.52	0.48	0.41	0.51
Lam = 0.01	0.47	0.54	0.52	0.52
Lam = 2	0.52	0.54	**0.58**	0.55
Lam = 10	0.51	0.51	0.53	0.54

Human subtract

Lam,lr	Lr=0.001	lr=0.01	lr=0.05	lr=0.1
Lam =10^-5	0.51	0.49	0.50	0.49
Lam = 0.01	0.46	0.53	0.51	0.47
Lam = 0.03	0.50	0.54	**0.57**	0.55
Lam = 0.09	0.50	0.53	0.53	0.54

GSC concat

Lam,lr	Lr=0.001	lr=0.01	lr=0.05	lr=0.1
Lam =10^-5	0.47	0.48	0.42	0.48
Lam = 0.01	0.52	0.51	0.53	0.46
Lam = 2	0.56	0.55	**0.62**	0.57
Lam = 10	0.51	0.54	0.55	0.55

GSC subtract

Lam,lr	Lr=0.001	lr=0.01	lr=0.05	lr=0.1
Lam=10^-5	0.56	0.48	0.49	0.51
Lam = 0.01	0.54	0.50	0.52	0.54
Lam = 2	0.54	0.55	**0.60**	0.58
Lam = 10	0.50	0.51	0.57	0.55

We test the data using randomdataset(ie dataset containing both seen and unseen data) and observe that gsc concat gives highest accuracy among other datasets for logistic regression

Nueral network :

4 different types of nueral networks are generated for 4 datasets. Each nueral network takes different inputs . For human concatenated data we have 18 inputs. similarly number of coloumns in the training data becomes number of inputs for the nueral network.

For gsc concatenated data :

We use 4 layers in this network. 1000,500,100,10 are the corresponding nodes in each layer. Among all the optimizes adams optimizer gives the highest accuracy .And among all the datasets gsc-concatinated dataset gives the highest accuracy of 79.924

Gsc subtracted Data :

We use 4 layers in this network. 1000,500,100,10 are the corresponding nodes in each layer. Among all the optimizes adams optimizer gives the highest accuracy. Accuracy obtained is 75.278 for this data

human concatenated Data :

Using many nodes in this layers overfits the model and gives very low accuracy. So we use less number of nodes compared to gsc data. We only use 3 hidden layers with 100,25 ,15 nodes in corresponding layers.This data gave an accuracy of 55.856

Human subtracted Data :

This data gives least accuracy among all the other data . With 9 inputs and 3 hidden layers this data gives an accuracy of 50.356

Observation :

We observe that gsc-concatinated data set performs better when compared to other data sets**.**