CSE 574: Introduction to Machine Learning (Fall 2018)
Project 2

Objective

Task is to find similarity between the handwritten samples of the known and the questioned writer by using linear regression.
Each instance in the CEDAR \AND" training data consists of set of input features for each hand-written \AND" sample. The features are obtained from two different sources:
1. Human Observed features: Features entered by human document examiners manually
2. GSC features: Features extracted using Gradient Structural Concavity (GSC) algorithm.
The target values are scalars that can take two values f1:same writer, 0:different writers.

Tasks performed for project implementation:

1. **Data Processing**
   Human Observed features: Features entered by human document examiners manually

   It has 18 features identifying each image id. We are provided with same and different pairs of images that display the handwriting samples of the same author and of different authors respectively.

   We take in data from same pairs and different pairs in equal proportion for both datasets (Human Observed and GSC) and store it in two different file sets for training, testing & validation of both datasets separately. We shuffle this data in order to train the model over a non-biased dataset.

2. **Algorithms implemented:**

   Linear Regression
   In linear regression, a linear hypothesis function hθ(x) which approximates target variable y.

$$y \approx h_\theta(x) = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n$$

   Here θ's are the parameters of the model. When the context of parameters is clear, we can drop θ in hθ(x). To simplify the notation, we define x0 = 1, so

$$h(x) = \sum_{i=1}^{n} \theta_i x_i = \theta^T x$$

   Job of our learning problem is to learn these parameters. Obvious method is to choose parameters which make h(x) as close to y as possible for the training examples provided to us.

Arooshi Avasthy
arooshia@buffalo.edu

Cost function:

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

Gradient Descent

We want to choose θ which minimizes J(θ). A general strategy would be to start with some random θ and keep changing θ to reduce our objective function J(θ).

More specifically, Gradient descent starts with some initialization of θ and repeatedly performs the following update until convergence:

$$\theta_i := \theta_i - \alpha\frac{\partial J(\theta)}{\partial \theta_i} \qquad \forall i \in [0, 1, 2, 3, .., n]$$

Here, α is the learning rate which is a hyperparameter and we will have to tune it. In order to implement this algorithm, we will first have to calculate the partial derivative.

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j}\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\end{aligned}$$

This method looks at all training examples for every iteration of the loop.

Closed Form Solution

Another method to find parameters is to set derivative of objective function to zero and obtain required parameters without resorting to iterative algorithm. Before doing this, we quickly introduce notations for matrix.

Given a training set, we define the design matrix X to be m × n matrix as

$$X = \begin{bmatrix}(x^{(1)})^T \\ (x^{(2)})^T \\ \vdots \\ (x^{(m)})^T\end{bmatrix}$$

Similarly, let Y be the m dimensional vector containing the target labels as

$$Y = \begin{bmatrix}y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)}\end{bmatrix}$$

Writing J(θ) in matrix representation, we get

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

Arooshi Avasthy
arooshia@buffalo.edu

Finally, we take derivative of J(θ) with respect to θ and set it to zero.

$$\nabla_\theta J(\theta) = X^T(X\theta - Y) = 0$$

Thus, the value of θ which minimizes J(θ) is given in closed form by

$$\theta = (X^T X)^{-1} X^T Y$$

Experiments through code implementation of Linear Regression:

Closed Form Solution: Human Observed Concatenation

No. of Data points	Accuracy	ERMS Training	ERMS Validation	ERMS Testing
1	52.2292	0.49927	0.49717	0.50021
11	54.1401	0.49761	0.49792	0.49975
21	52.2292	0.49698	0.49754	0.49698
31	52.2292	0.49761	0.49717	0.49618

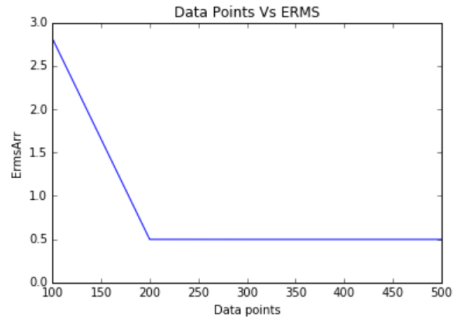
Closed Form Solution: Human Observed Subtracted

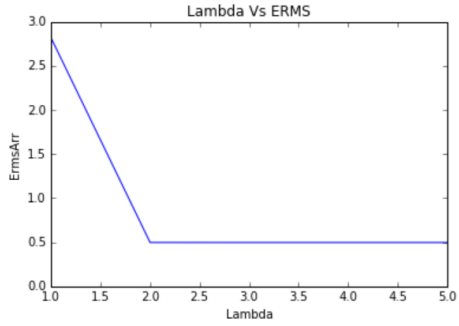
No. of Data points	Accuracy	ERMS Training	ERMS Validation	ERMS Testing
1	50.3184	0.49956	0.49954	0.50021
11	52.8662	0.49761	0.49792	0.49666
21	53.5031	0.49761	0.49754	0.49618
31	55.4140	0.49761	0.49754	0.49618

Gradient descent: Human Observed Concatenated

No. of Data points	Lambda	ERMS Training	ERMS Validation	ERMS Testing
100	**1**	2.60362	2.74615	2.81509
200	**2**	0.49761	0.49792	0.49666
300	**3**	0.49761	0.49754	0.49618
400	**4**	0.49761	0.49754	0.49618
500	**5**	0.49761	0.49754	0.49618

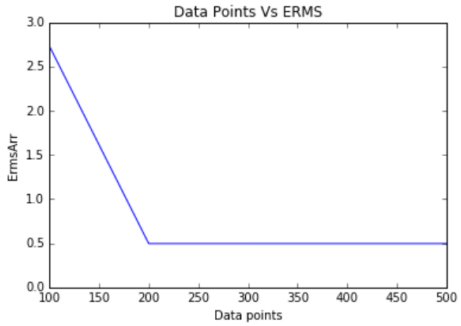
Arooshi Avasthy
arooshia@buffalo.edu

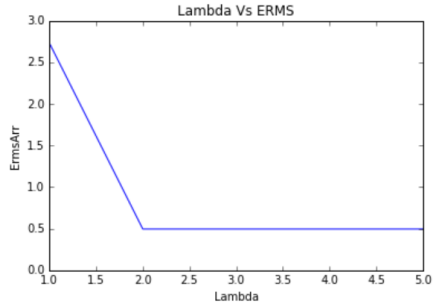




Human Observed Subtracted

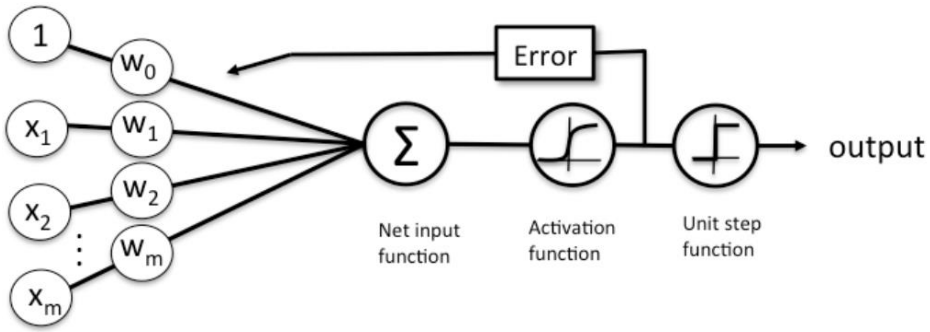
No. of Data points	Lambda	ERMS Training	ERMS Validation	ERMS Testing
100	**1**	2.55735	2.50738	2.7304
200	**2**	0.49813	0.49868	0.49597
300	**3**	0.49813	0.4984	0.49595
400	**4**	0.49813	0.4984	0.49595
500	**5**	0.49813	0.4984	0.49595





Arooshi Avasthy
arooshia@buffalo.edu

Logistic Regression:



Schematic of a logistic regression classifier.

Logistic regression is implemented as a one layer neural network which does not include any hidden layers in between. Simple implementation of logistic regression involves multiplying the inputs with weights and passing the result through Activation function, preferably sigmoid to obtain the output. Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. Logistic regression predictions are discrete values unlike Linear regression which gives out continuous values. To squash the predicted value between 0 and 1, we use the sigmoid function.

$$z = \theta_0 + \theta_1 \cdot x_1 + \theta \cdot x_2 + \cdots \qquad g(x) = \frac{1}{1 + e^{-x}}$$

Linear Equation and Sigmoid Function

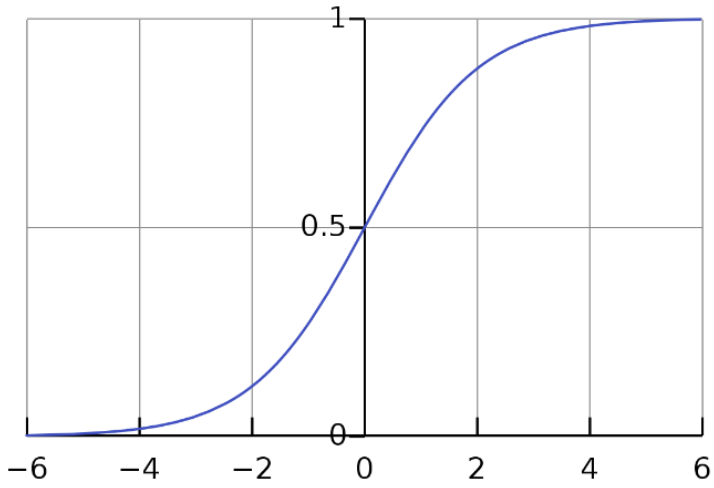
We take the output(z) of the linear equation and give to the function g(x) which returns a squashed value h, the value h will lie in the range of 0 to 1.

$$h = g(z) = \frac{1}{1 + e^{-z}}$$

Squashed output-h

Graph of the Sigmoid Function:

Arooshi Avasthy
arooshia@buffalo.edu



Sigmoid Function graph

The graph shows that the sigmoid function becomes asymptote to y=1 for positive values of x and becomes asymptote to y=0 for negative values of x.

Cost Function

We do not use ERMS for calculating cost like incase of linear regression because we are trying to predict class values. Therefore, we use a logarithmic loss function to calculate the cost for misclassifying.

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

The above cost function can be rewritten as below since calculating gradients from the above equation is difficult.

$$-\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Calculating Gradients

Take partial derivatives of the cost function with respect to each parameter(theta_0, theta_1, …) to obtain the gradients. Using these gradients, update the values of theta_0, theta_1, …

Arooshi Avasthy
arooshia@buffalo.edu

$$J = \frac{-1}{m} \cdot [\sum_{i=1}^{m} y_i \cdot \log h_i + (1 - y_i) \cdot \log 1 - h_i]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{-1}{m} \cdot [\sum_{i=1}^{m} \frac{y_i}{h_i} \cdot h_i^2 \cdot x_n \cdot \frac{1 - h_i}{h_i} + \frac{1 - y_i}{1 - h_i} \cdot -h_i^2 \cdot x_n \cdot \frac{1 - h_i}{h_i}]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{-1}{m} \cdot [\sum_{i=1}^{m} x_n \cdot (1 - h_i) \cdot y_i - x_n \cdot h_i \cdot (1 - y_i)]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{1}{m} \cdot x_i \cdot [\sum_{i=1}^{m} h_i - y_i]$$

Gradients

Experiments through code implementation of Logistic Regression: Graphs depicting change in accuracy with change in epoch included with the code

	Concatenated_HOD	Subtracted_HOD	Concatenated_GSC	Subtracted_GSC
Number of Epochs	1000	1000	1000	1000
Batch Size	50	50	50	50
Learning Rate	0.5	0.5	0.5	0.5
Features	18	9	1024	512
Training Data Samples	1500	1500	2000	2000
Testing Accuracy	56.68789808	52.2292993630	97.48743718592	92.4623115577

Neural Network:

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information.

An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process.

Arooshi Avasthy
arooshia@buffalo.edu

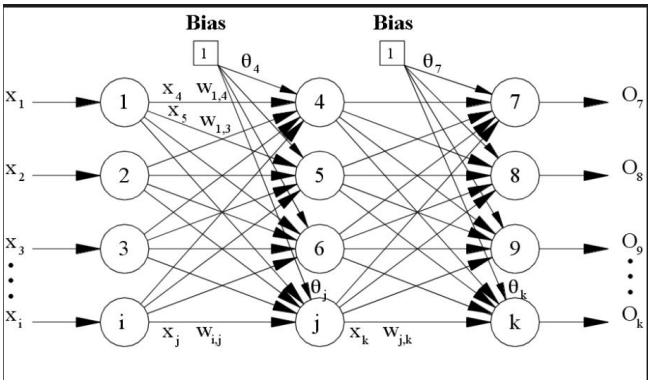
When a signal(value) arrives, it gets multiplied by a weight value. If a neuron has 4 inputs, it has 4 weight values which can be adjusted during training time.

Genesis Equation:

$$z = x_1 * w_1 + x_2 * w_2 + \ldots\ldots + x_n * w_n + b * 1$$

$$\hat{y} = a_{out} = sigmoid(z)$$

$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$



Forward propagation is a process of feeding input values to the neural network and getting an output which we call predicted value. When input values are fed to the neural network's first layer, it goes without any operations. Second layer takes values from first layer and applies multiplication, addition and activation operations and passes this value to the next layer. Same process repeats for subsequent layers and finally we get an output value from the last layer.

After forward propagation, the predicted value is obtained. To calculate error the predicted value is compared with the actual output value. Loss function is used to calculate the error value. Then the derivative of the error value is calculated with respect to each and every weight in the neural network. Back-Propagation uses chain rule of Differential Calculus. In chain rule first the derivatives of error value is calculated with respect to the weight values of the last layer. These derivatives, gradients and use these gradient values to calculate the gradients of the second last layer. This process is repeated until gradients for each and every weight is obtained. This gradient value is then subtracted from the weight value to reduce the error value. This way Local Minima(means minimum loss) is obtained.

Experiments through code implementation: Graphs depicting change in accuracy with change in epoch included with the code

	Concatenated_HOD	Subtracted_HOD	Concatenated_GSC	Subtracted_GSC
Number of Epochs	1000	1000	1000	1000
Batch Size	50	50	50	50
Learning Rate	0.5	0.5	0.5	0.5

Arooshi Avasthy
arooshia@buffalo.edu

Features	18	9	1024	512
Training Data Samples	1500	1500	2000	2000
Testing Accuracy	53.503184	47.13375796	98.4924623115	98.4924623115

Arooshi Avasthy
arooshia@buffalo.edu