Regression

EE4708: Data Analytics Laboratory - Week 3

Course Instructors: Dr. Kaushik Mitra & Dr. Ramkrishna Pasumarthy

Indian Institute of Technology Madras

Table of contents

Supervised Learning

- **Supervised learning** is a machine learning task which learns a function or model that best relates the input features (independent variable) to observed outputs (dependent variable).
- The learnt function or model can be used for predicting the unknown outputs for given inputs.
- Here, learning implies updating the parameters of the function until the desired function is obtained.

Based on the type of dependent variable, supervised learning can be categorised as follows:

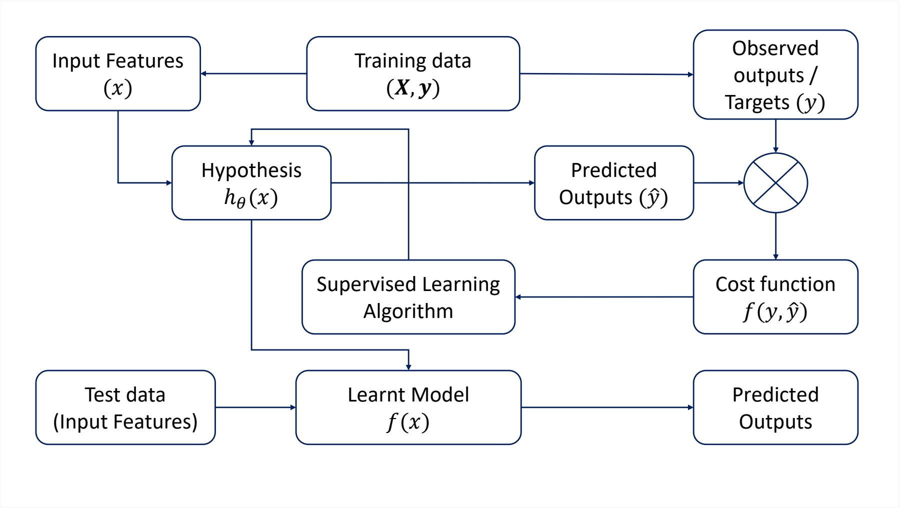
1. **Regression**: In this learning task, dependent variable takes continuous values. e.g. Predicting amount of rainfall based on past weather data.
2. **Classification**: In this task, dependent variable takes binary or discrete values i.e. categories or classes. e.g. Classifying emails as spam or not based on body and text of the email

DA Lab Week 3 Supervised Learning: Components

Following are the major components in a supervised learning framework:

1. **Data:** Comprising of multiple samples each with values of input features and targets (outputs)
2. **Model:** Refers to a function or approach which predicts outputs given inputs and is to be learnt by finding the right parameters
3. **Hypothesis set:** Refers to the set of chosen functions (model structure) among which the function which best fits the data is to be determined by optimizing the parameters
4. **Cost Function:** Function which compares predicted outputs with observed outputs and which is to be optimized to obtain the desired model
5. **Optimization algorithms:** Gradient descent and its variants are used to optimize the cost function

DA Lab Week 3 Supervised Learning: Flowchart


Input Features (x)
Training data $(\boldsymbol{X}, \boldsymbol{y})$
Observed outputs / Targets (y)
Hypothesis $h_\theta(x)$
Predicted Outputs $(\hat{y})$
Supervised Learning Algorithm
Cost function $f(y, \hat{y})$
Test data (Input Features)
Learnt Model $f(x)$
Predicted Outputs

DA Lab Week 3 Types of Regression

Following are the different types of regression tasks based on the features and model:

1. **Linear regression** is a regression task in which the model being learnt is assumed to be a linear function of parameters.
   - 1.1 **Simple linear regression** Linear regression task in which there is one input feature (independent variable) and one output (dependent variable).
   - 1.2 **Multiple linear regression** Linear regression task in which there are multiple input features and one output.
   - 1.3 **Polynomial regression** This is a generalisation case of Multiple linear regression in which the input features include higher powers
2. **Non-Linear regression** is a regression task in which the model being learnt is assumed to be a non-linear function of parameters.

The type of the regression model to be built is a user choice to be made based on the following factors:

1. Aprior knowledge of the model being learnt
2. Observation and visualisation of the dataset
3. Fit of the model and errors in prediction

DA Lab Week 3 Training, Validation and Testing of the Model

- In supervised learning, choosing appropriate model type and hyper-parameters is important in learning a good model which predicts well on unseen samples.
- Therefore, model validation is performed to evaluate the trained model in an unbiased manner
- After evaluation, it is trained again with changed type or hyperparameters
- Finally, the accepted model can be used to predict outputs.

There exist 3 types of datasets which are involved in the above process and they are as follows:

- **Training set:** The set of data samples used to train or learn the model parameters
- **Validation set:** The set of data samples used to provide an unbiased evaluation of a model fit on the training dataset while tuning any hyper-parameters
- **Testing set:** The set of samples for which the outputs (may be unknown) are to be predicted given input features

In general, the given dataset with known outputs is split into training and validation sets to train and validate the model. There are standard model validation techniques which will be discussed in later weeks.

DA Lab Week 3 Linear Regression: Model structure

The most general form of the linear regression model is:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \hat{y} = \theta_0 + \Sigma_{j=1}^{m-1} \theta_j \phi_j(\mathbf{x}) \tag{1}$$

where $\hat{y}$ is the predicted output for a given input, $\mathbf{x} = (x_1, \ldots, x_d)^T$, is a d-dimensional input feature vector, $\phi_j(.)$ represents a basis function, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)^T$ are the model parameters and m is the order of the linear model. The basis function for different types of linear regression models are as follows:

1. **Simple linear regression:** $\phi_1 = x$, $m = 2$.
2. **Multiple linear regression:** $\phi_j = x_j \;\; \forall j = 1, \ldots, m$.
3. **Polynomial regression:** $\phi_i = \Pi_{j=1}^{d} x_j^k, \;\; 0 \leq k \leq p$ where p is the chosen degree of the polynomial
   Eg. Basis functions for a model with 2 features (x_1, x_2) and degree 2 are:
   $x_1, x_2, x_1^2, x_2^2, x_1 x_2$

Certain assumptions are made while building a linear regression model. Refer to this link for details: http://r-statistics.co/Assumptions-of-Linear-Regression.html

DA Lab Week 3 Cost Function

- **Cost Function** is a measure of how wrong the model is in terms of its ability to estimate the relation between inputs and outputs. There exist different types of cost functions and the most popular among them is the Mean Squared Error (MSE) between the predicted and observed outputs.
- The formula mean squared error for a dataset with N samples is given by:

$$J(\theta) = \frac{1}{2N}\Sigma_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{2}$$

  where $\hat{y}_i$ is the predicted output and y_i is the observed or given output.
- The objective is to find θ that minimizes the cost function J
- Check this link for more details about the cost function for linear regression: https://machinelearningmedium.com/2017/08/11/cost-function-of-linear-regression/

DA Lab Week 3 Solution Approaches

Following are the different approaches for finding the parameters of the desired regression model:

1. **Analytical approach:** An approach to find the parameters directly using an equation called the normal equation
2. **Maximum Likelihood approach:** An probabilistic approach to estimating the model parameters
3. **Gradient Descent approach:** An optimization approach to minimizing the cost function and finding the parameters
4. **Grid Search:** A brute force search approach to finding the optimal parameters

Note: The maximum likelihood solution is identical the solution given by the normal equation when minimising least squares with normally distributed errors. This is shown in the following slides.

Why mean square error?

Let us re-write Eq.(1) for the case of multiple regression as,

$$\hat{y}(\mathbf{x}, \boldsymbol{\theta}) = \Sigma_{j=0}^{m-1} \theta_j x_j$$

where $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_{m-1})$ and $x_0 = 1$. Then $\hat{y}$ is conveniently written in dot-product form,

$$\hat{y} = \boldsymbol{\theta}^T \mathbf{x} \tag{3}$$

Hence observed values $\hat{y}$ are related to the true values as:

$$y = \hat{y} + \epsilon \tag{4}$$

where ϵ is a zero-mean error variable with distribution $\epsilon \sim \mathcal{N}(0, \beta^{-1})$. Then

$$p(y|\mathbf{x}, \boldsymbol{\theta}, \beta) = \mathcal{N}(y|\hat{y}, \beta^{-1}) \tag{5}$$

For a set of independent random variables $\mathrm{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, it is required to maximize the likelihood of observed values for given input and parameter values. In the following steps, it is shown that maximizing the likelihood is equivalent to minimizing the MSE.

Contd...

$$p(y|\mathrm{X}, \boldsymbol{\theta}, \beta) = \Pi_{i=1}^{N} \mathcal{N}(y|\boldsymbol{\theta}^T \mathbf{x}_i, \beta^{-1})$$
$$\implies \ln p(y|\mathrm{X}, \boldsymbol{\theta}, \beta) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln 2\pi - N\beta J \tag{6}$$

Gradient of the log-likelihood function,

$$\nabla \ln p(y|\mathrm{X}, \boldsymbol{\theta}, \beta) = \beta \Sigma_{i=1}^{N} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i) \mathbf{x}_i^T \tag{7}$$

Setting the gradient to zero, and solving for $\boldsymbol{\theta}$,

$$0 = \Sigma_{i=1}^{N} y_i \mathbf{x}_i^T - \boldsymbol{\theta}^T \Sigma_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T \tag{8}$$

The ML estimate of $\boldsymbol{\theta}$ is then,

$$\boldsymbol{\theta}_{ML} = (\mathrm{X}^T \mathrm{X})^{-1} \mathrm{X}^T \mathrm{Y} \tag{9}$$

Properties of Normal equation

- It provides closed form solution to the Least squares problem.
- With the increase in dimensionality and the number of samples, computation of the inverse becomes expensive.

DA Lab Week 3 I Gradient Descent

Gradient descent is an optimization algorithm used to minimize the cost function by moving in the direction opposite to that of the gradient of the cost function in an iterative manner. to perform gradient descent, the cost function must be differentiable.

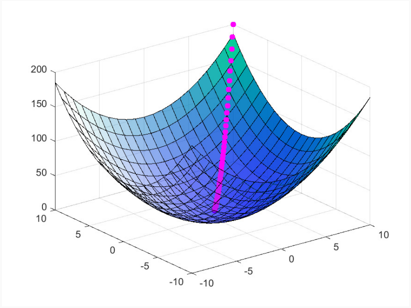


Figure: Simple LR is a 2D loss surface. The magenta line is path by gradient descent algorithm at every iteration.

The gradient descent algorithm to minimize a cost function $J(\boldsymbol{\theta})$, starting with a initial estimate of parameter $\boldsymbol{\theta}^{(0)}$ is as follows:

```
**for** $i = 0, 1, 2, \dots$ **do**
    $g^{(i)} \leftarrow \nabla J(\boldsymbol{\theta}^{(i)})$
    $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)} - \alpha g^{(i)}$
**end**
```

α is the learning rate.

Parameter initialization

- Often, parameters are initialized as zero.
- However, for some functions, the choice of initial point may be critical, especially when optimizing non-linear functions.
- When the cost function has several minima, one can run the algorithm several times for different initial points, and then select the best result.

Stopping criteria

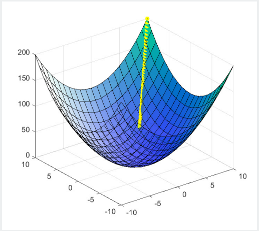
Stopping criteria is a condition which determines that the gradient descent algorithm has converged. There are multiple ways of defining a stopping criteria:

1. Setting the maximum number of iterations apriori
2. Stopping when the change in cost function is less than a certain threshold
3. Stopping when the change in parameters is less than a certain threshold
4. Stopping when the change in gradient is less than a certain threshold

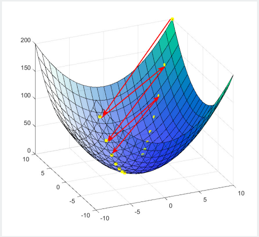
DA Lab Week 3 III Gradient Descent

Effect of Learning rate

- The choice of learning rate can decide the convergence of gradient descent:



(a) A small value of learning rate may take too many iterations to reach minima



(b) A large value of learning rate may oscillate and even lead to divergence.

Figure: Effect of learning rate.

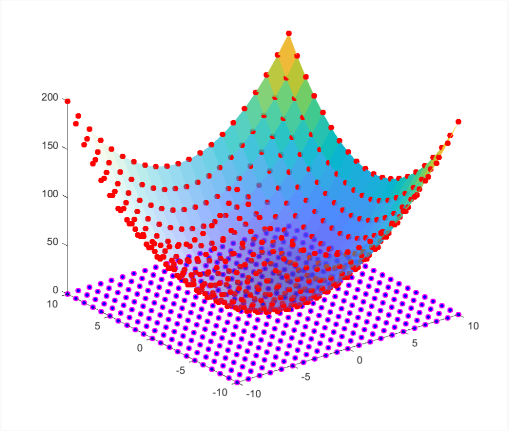
Refer this link for more details on gradient descent in linear regression:
https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

DA Lab Week 3 I Grid Search

Grid search method is a brute force method for optimization. Given the cost function $J(\mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta} \in \mathbb{R}^M$ construct a grid over domain of $J(\Theta)$.

$$G = \{\theta_0, \ldots, \theta_{M-1} | \theta_0 \in \{l_1, \ldots, h_1\}, \ldots, \theta_{M-1} \in \{l_{M-1}, \ldots, h_{M-1}\}\}$$

To form the grid, an interim solution must be available.



A 2-d grid is created on the X-Y plane (in purple). The loss at every point on the grid is calculated (marked in red). The parameters corresponding to the minimum

DA Lab Week 3 II Grid Search

Clearly, the algorithm has a disadvantage as the number of iterations increases exponentially with the increase in model complexity. The application of this algorithm is in parameter tuning. Once, a local minima is found using Gradient descent, the neighbourhood may be searched for fine-tuning.

DA Lab Week 3 Algorithm: Analytical Approach (Normal Equation)

Algorithm 1 NORMAL EQUATION(X, Y)

```
1: Input: TRAINING FEATURES $X$, SHAPE($N$, $D$)
   TARGET VARIABLES $y$, SHAPE($N$, 1)
2: Output: $w$, SHAPE($D + 1$, 1)
3:     $X \leftarrow$ APPEND($X$, 1)
4:     $w \leftarrow$ (PSEUDO_INVERSE($X^T X$))($X^T y$)
5: RETURN $w$
6:
```

DA Lab Week 3 Algorithm: Gradient Descent

Algorithm 2 GRADIENT_DESCENT($X, y, \boldsymbol{\theta}^{(0)}, \epsilon, max\_iter$)

```
1:  Input: Training features $X$, shape $(N, D)$, target variable $y$, shape $(N, 1)$
    Initial parameter vector $\boldsymbol{\theta}^{(0)}$, shape $(D + 1, 1)$
    $pr$, scalar ( precision for stopping criteria)
2:  Output: $\boldsymbol{\theta}$, SHAPE$(D + 1, 1)$
3:  Local: $J_{hist}$ , LIST to store loss history $J$ at each iteration,$J_{prev}$ is loss in the previous iteration, $J_{diff}$ is
    the difference between losses of current and previous iteration, $i$, iteration count.
4:  $X \leftarrow$ APPEND$(X, 1)$
5:  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
6:  Initialize $J_{hist}$
7:  $i \leftarrow 0$
8:  $J_{prev} \leftarrow 0$
9:  $N \leftarrow$ LENGTH$(y)$
10:   while $J_{diff} \leq \epsilon$
11:       $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \frac{\alpha}{N}(X\boldsymbol{\theta}^{(i)} - y)^T X$
12:       $J \leftarrow$ COMPUTE_COST$(X, y, \theta_{i+1})$
13:       $J_{hist} \leftarrow$ APPEND$(J_{hist}, J)$
14:       $J_{diff} \leftarrow J - J_{prev}$
15:       $J_{prev} \leftarrow J$
16:       $i \leftarrow i + 1$
17:       if $i = max\_iter$ then
18:           BREAK
19: RETURN $\boldsymbol{\theta}, i, J_{hist}$
20:
```

DA Lab Week 3 Algorithm: Grid Search

Algorithm 3 GRID_SEARCH2D(X, y)

```
 1: Input:Training features $X$, shape $(N, 2)$ , target variable $y$, shape $(N, 1)$
 2: Output: $\boldsymbol{\theta} = [\theta_0, \theta_1]$, shape $(2, 1)$
 3: Local: $g0$ is the set of search points of size $m_0$ along $\theta_0$, $g1$ is the set of search points of size $m_1$
    along $\theta_1$. $J$ is the loss at current iteration. $J_{min}$ is the minimum loss value.
 4: Initialize
 5: $g_0 \leftarrow$ LINSPACE$(l_0, h_0, m_0)$, $g_1 \leftarrow$ LINSPACE$(l_1, h_1, m_1)$
 6: $J_{min} \leftarrow inf$
 7: for each $i \in g0$
 8:    for each $j \in g1$
 9:    $J \leftarrow$ COMPUTE_COST$(X, y, \boldsymbol{\theta})$
10:    if $J < J_{min}$ then
11:       $J_{min} \leftarrow J$
12:       $\theta_0 \leftarrow i$
13:       $\theta_1 \leftarrow j$
14: RETURN $\boldsymbol{\theta}$
15:
```