Personal Notes on

Machine Intelligence

Jeffrey Ming Han Li

jeffreyli2288@Outlook.com

Abstract

This note is based on ECE421 introduction to machine learning.

Contents

1 Introduction

Machine learning is essentially

> *Develop computational systems that adaptively improve performance with experience from accumulated data.*

this is a classical and formal definition of machine learning. For the scenerio of movie recommendation, the machine learning approach are the define But how good is our model?

- **Model Parameter**
  - User preference vector:
$$x_i = (x_{i1}, x_{i2}...x_{in})$$
  - Movie attribute vectors
$$y_i = (y_{i1}, y_{i2}...y_{in})$$

  from data, ML learn about these two vectors which are so called the **Model parameters**. let γ_{ij} be the ratting given by user i for movie j partial list ratings form a two dimensional matrix describing each users' rating for each movie.

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
User 1	γ_{11}	γ_{12}	γ_{13}	γ_{14}	γ_{15}
User 2	γ_{21}	γ_{22}	γ_{23}	γ_{24}	γ_{25}
User 3	γ_{31}	γ_{32}	γ_{33}	γ_{34}	γ_{35}
User 4	γ_{41}	γ_{42}	γ_{43}	γ_{44}	γ_{45}

Table 1: User–Movie rating matrix represented by γ_{ij}.

  given this matrix, we want to learn the two vectors defined previously. However, some ratings might not be available, so we define a set S such that
$$S = \{(i, j) : \gamma_{ij} \quad \text{is available in our dataset}\}$$

- **Model Proposal** Now it is time to propose a model for it: For user i & movie j, compute an estimated rating. which can be obtained by
$$\hat{\gamma}_{ij} = x_i^T y_j = \sum_{l=1}^{n} x_{il} y_{jl}$$
  note that that both vectors are initially column vectors.
- **Loss Function:** it is an evaluation of how good the prediction is, which is going to be determined, in the simplest form as:
$$E_{in} = \sum_{(i,j)\in S} (\gamma_{ij} - x_i^T y_j)^2$$
  this gives the error between the estimated rating and the true rating. The training is to find the combination of x and y such that this loss is minimized using *Gradient Descent*

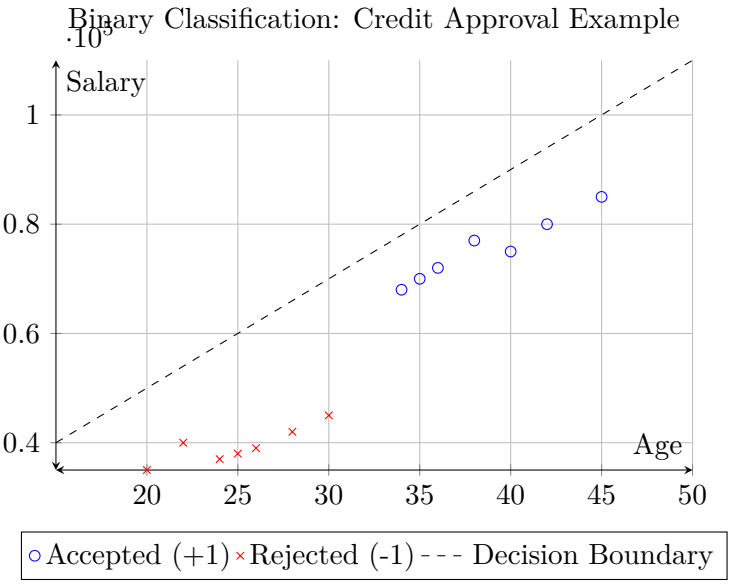
1.1 Binary Linear Classification

- **Attribute:** the attribute is a user vectors that
$$x = (\text{age, gender, salary...})$$
  and the output is
$$y = \begin{cases} +1 & \text{user is accepted} \\ -1 & \text{user is rejected} \end{cases}$$

so the data set is going to be like

$$\{(x_1, y_1)...(x_n, y_n)\}$$

so lets say we have only two parameters age and salary, so we can plot a 2D plot of acceptance like



- **Loss Function:**so now lets define a weight vector w and a constant bias b. Given a certain x we can determine y, which can be obtained by

$$If \quad w_1x_1 + w_2x_2...w_nx_n = w^Tx \geq b \rightarrow y = 1$$

and vice versa, this w and b can be trained such that we need to minimized the loss function.

$$E_{in} = avg(\text{number of error on data set}) \ = \frac{1}{N}\sum_{i=1}^{N} 1 * \{y_i \neq \hat{y}_i\}$$

In general minimizing the loss is NP hard. If data set is separable, we can achieve $E_{in} = 0$, then the perception learning algorithm can be used to find such a classifier. However, the model is easier with a modified weight vectors

$$w = (-b, w_1, w_2...w_n) \quad x = (x_0 = 1, x_1, x_2...x_n) \in \mathbb{R}^{n+1}$$

such that the inequality can be written as

$$w^Tx - b \leq 0$$

which just makes b another weight like training process.

1.2 Perception Algorithm

THe input is a dataset that is linearly separable and the output is a set of weight $w \in \mathbb{R}^{n+1}$ such that the error is 0.

- **Initialization:**

$$w = (0, 0..., 0)$$

- Step 1: check if error = 0
- Step 2: let (x,y) be any misclassified point, then we update w as follows:
  1. if y = 1, we turn $w \rightarrow w + x$ and vice versa for y = -1. so essentially

$$w + y_nx_n \rightarrow wv$$

then go back to step 1 and update iteratively to find the perfect classifier.

A very simple understanding this type of regression is the linear classifier. For instance, lets talk about the Mnist data set, the way how it works is.

- Given an image of 256 pixels, we can decompose it into arrays of

$$x = (x_1.....x_{256})$$
$$w = (w_1.....w_{256)}$$

We can take out the only effective and useful information so called the **Features**

$$x = (x_1, x_5, x_8)$$

which can be plotted onto a 3D map and generate a perception boundary to tell whether it is a 5 or a 1.

However, for a non-separable data set, the misclassification might occur frequently. Therefore, a 'pocket' algorithm is designed such that the only difference with the classical perception algorithm is that it iterates many times until it finds a set of weight that achieves the minimum loss.