Machine Learning

Period of learn: 15/11/2019 to 28/11/2019

Elaborated by: Mabrouki Semah

2ed year student at EPT

Year: 2019/2020

SUMMARY

TABLE OF FIGURES

1 INTRODUCTION

1.1 WHAT IS MACHINE LEARNING?

There are many definitions of machine learning like:

- The older definition of Arthur Samuel: **"M.L field of study that gives computers the ability to learn ability to learn without being explicitly programmed"**
- The best definition of Tom Mitchell: **"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E"**
  - ➔ M.L is the science to make the computer able to learn like a human

1.2 MACHINE LEARNING VS ARTIFICIAL INTELLIGENCE

Artificial intelligence AI is the intelligence of machine

➔ M.L gives the AI

1.3 ADVANTAGES OF M.L

- M.L applications are making the life of human more easy
- M.L application cannot program by hand.
- It is a new capability for the machine
- Give an estimation of the real result

1.4 EXAMPLES OF M.L APPLICATIONS

- ✓ Computer vision
- ✓ Autonomous helicopter
- ✓ Recommender systems
- ✓ House price prediction

1.5 DIFFERENT TYPES OF M.L

1.5.1 Supervised learning

1.5.1.1 Definition

Teach the computer by give it training examples with the actual result and found the great model who can give the best estimation of the result of any test example.

➔ In supervised learning, we are given a data set and already know what our correct output should look like

1.5.1.2 Types of supervised learning problems

There are two types of supervised learning problems

- **Regression problems:**

The machine learning application predict continuous values output, means the set of output isn't discrete

EXAMPLE of regression problem**:** House price prediction

- ➔ Predict prices of houses is the regression problem because the price of house is continuous and not limited.
- ➔ Also it is an supervised learning application because we give the machine an number of training example with the actual result of every example to determinate the best model (found the model and determine the parameters of this model) so with this model we can enter any test example that we don't know the result (like an input) and the output of application can be a good estimation of the result.

Example of model to solve a regressions problem: linear regression model…

- **Classification problem**
  - ➔ In classification problem, the values predicted are discrete (For Example: when the result can be only 0 or 1)
  - ➔ Supervised learning can also use when we are a classification problem. In fact, we can use some training example with them actual results (0 or 1 for example) to determine the parameters of the model who we choose

EXAMPLE of classification problem: Breast cancer (malignant, benign)

y=1 if result=malignant

y=0 if result=benign

- ➔ For create an application able to determine if a person is malignant or not, we can use a supervised learning. In fact, first we choose the convenable model then we use training example with the result of any one to determine the parameters of this model.

Example of model to solve a classifications problem: logistic regression model, neural networks model….

1.5.2 Unsupervised learning

1.5.2.1 Definition

- ➔ It means that we found the model with give the machine some training examples without results

- ➔ It used specifically when we have a classification problem
- ➔ Unsupervised learning allows us to approach problems with little or no idea what our results should look like.
- ➔ We just told it that this is our data set can you find some structure in this data.
- ➔ It based in the features of our data

2 LINEAR REGRESSION

It's a model to create some supervised learning machines. It used when we have a regression problem

2.1 MODEL REPRESENTATION

2.1.1 Simple case: number of features n=1

Hypothesis: $H_\theta(x) = \theta_0 + \theta_1 x$

Features: x

Parameters: θ_0 and θ_1

Dataset (training examples): {(x(i), y(i)); i=1, ..., m}

m: number of training examples

REMARQUE:

- ✓ It's a supervised learning problem every training example i should be accompanied with the result y(i)
- ✓ We use the training examples to determine the convenable parameters for this model

2.1.2 General case: Number of features n≥1

Hypothesis: $H_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \; ; \; x_0 = 1$

In this case: $\theta = \begin{pmatrix} \theta_0 \\ . \\ . \\ . \\ \theta_n \end{pmatrix}$ and $x = \begin{pmatrix} x_0 \\ . \\ . \\ . \\ x_n \end{pmatrix}$; $x_0 = 1$

Parameters: θ

How we can determinate this parameter?

To determinate the parameter θ we need training set {(x(i), y(i)); i=1, ..., m}

The best parameter θ is that who verified the minimum value of the cost function (Error function)

2.2 COST FUNCTION

For determine the module who give the estimation of the result of any test example. The predict of this model for any training example should be an estimation for the actual results.

So, we should determine the parameter verifier: $H_\theta\left(x^{(i)}\right) \approx y^{(i)}$

For that we define the cost function:

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

So, the parameter θ determinate by $minimize_\theta\, J(\theta) = minimize_{\theta_0,\ldots,\theta_n}\, J(\theta)$

How we will calculate this minimum?

➔ There many methods to minimize a function like gradient descend, conjugate gradient, BFGS…. But in all this course we will use gradient descend algorithm

2.3 GRADIENT DESCEND ALGORITHM

Initialization of θ: initialize $\theta_0, \ldots., \theta_n$

For i=1…. number of iterations:

For j=0…. n:

$$\theta_j := \theta_j - \alpha \frac{dJ(\theta)}{d\theta_j}$$

$$\alpha: learning\ rate$$

We have $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(H_\theta\left(x^{(i)}\right) - y^{(i)})^2$ → $\frac{dJ(\theta)}{d\theta_j} = \frac{1}{m}\sum_{i=1}^{m}(H_\theta\left(x^{(i)}\right) - y^{(i)}) * {x_j}^{(i)}$

So, our algorithm is:

Initialization of θ: initialize $\theta_0, \ldots., \theta_n$

For i=1…. number of iterations:

For j=0…. n:

$$\theta_j := \theta_j - \alpha * \frac{1}{m}\sum_{i=1}^{m}(H_\theta\left(x^{(i)}\right) - y^{(i)}) * {x_j}^{(i)}$$

- If α is too small → the algorithm converges slowly
- If α is too large → the algorithm can diverge

➔ We should choose the convenable learning rate

Remarque:

To program this algorithm without iterative and loops we can use linear algebra and metrics multiplication

2.3.1 How to make sur that gradient descend working correctly?

If after any iteration, the cost function $J(\theta)$ decrease so gradient descent converge correctly

It means, if θ^i the result of gradient descend algorithm before the iteration i, $J(\theta^{i+1}) < J(\theta^i)$

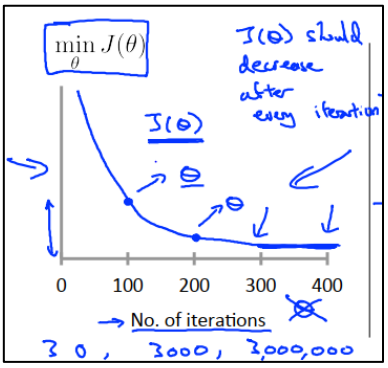

Figure 1:The cost is a function of the number of iterations

2.3.2 How to choose the learning rate α ?

If $J(\theta)$ increase after an iteration so gradient descend algorithm working incorrectly. For correct that, we need to use smaller learning rate α

But if α is too small, gradient descend converge very slowly

➔ To choose the best learning rate α we should try 0.001,0.002,0.03…,0.01, ….

2.4 POLYNOMIAL REGRESSION

2.4.1 Hypothesis for this model

If every feature is the polynomial function of the other so our model is writing:

$H_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots$ or $H_\theta(x) = \theta_0 + \theta_1 \sqrt{x} + \theta_2 \sqrt[2]{x} + \cdots$

2.4.2 Polynomial regression vs linear regression

Sometime linear regression doesn't be a good model to solve a regression problem because it can give a bad estimation for the result for the training example or for the test example

Polynomial regression can be better for solve this problem

2.4.3 Example: House price prediction

If we consider that we have only the feature size of house x so we can define the polynomial regression model $H_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots \theta_p x^p$

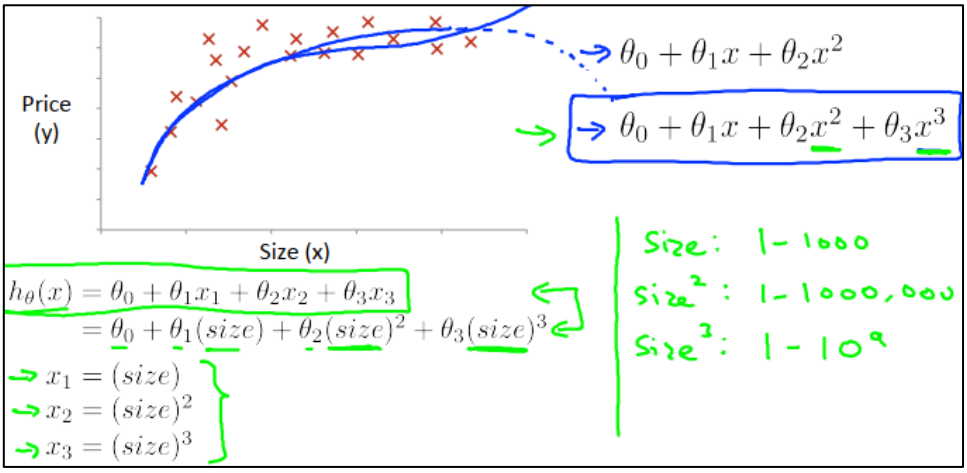

Figure 2:Linear regression model with different degree d

2.5 VECTORIZATION

To program the model faster we can use linear algebra and vectorization

Prediction: $H_\theta(x) = \theta^T x$; $\theta = \begin{pmatrix} \theta_0 \\ . \\ . \\ . \\ \theta_n \end{pmatrix}$ and $x = \begin{pmatrix} x_0 \\ . \\ . \\ . \\ x_n \end{pmatrix}$; $x_0 = 1$

Gradient descend:

For i=1…. number of iterations:

$$\theta = \theta - \alpha\delta$$

$\delta = \begin{pmatrix} \delta_0 \\ . \\ . \\ . \\ \delta_n \end{pmatrix}$ with $\delta_j = \frac{1}{m}\sum_{i=1}^{m}(H_\theta(x^{(i)}) - y^{(i)}) * x_j{}^{(i)}$

For $X = \begin{bmatrix} -- (x^{(1)})^T -- \\ -- (x^{(2)})^T -- \\ . \\ . \\ -- (x^{(m)})^T -- \end{bmatrix} = \begin{bmatrix} x_0^{(1)} \dots x_n^{(1)} \\ \dots . \\ \dots . \\ x_0^{(m)} \dots x_n^{(m)} \end{bmatrix}$ with $x_0^{(j)} = 1 \ \forall j = 1 \dots m$, $\theta = \begin{pmatrix} \theta_0 \\ . \\ . \\ . \\ \theta_n \end{pmatrix}$ and

$y = \begin{pmatrix} y_1 \\ . \\ . \\ . \\ y_m \end{pmatrix}$

➔ **On octave**:

$$\begin{pmatrix} H_\theta(x^{(1)}) \\ . \\ . \\ . \\ H_\theta(x^{(m)}) \end{pmatrix} = X * \theta \ ; \ X(:,j) = \begin{bmatrix} x_j^{(1)} \\ \dots . \\ \dots . \\ x_j^{(m)} \end{bmatrix}$$

$$\delta_j = \frac{1}{m} sum((X\theta - Y) .* X(:,j), 1)$$

$$\delta = \frac{1}{m} X^T * (X\theta - Y)$$

3 LOGISTIC REGRESSION

It used to solve a classifications problem

3.1 REPRESENTATION OF LINEAR REGRESSION MODEL

Hypothesis: $H_\theta(x) = g(\theta^T x)$; with the sigmoid function is $g(z) = \frac{1}{1+e^{-z}}$ $\quad 0 \le g(z) \le 1$

Parameters: θ

Training set: {(x(i), y(i)); i=1, ..., m}

We will find the parameters θ who verified:

$$\begin{cases} H_\theta(x) \ge 0.5 \ \ if \ y = 1 \\ H_\theta(x) < 0.5 \ \ if \ y = 0 \end{cases}$$

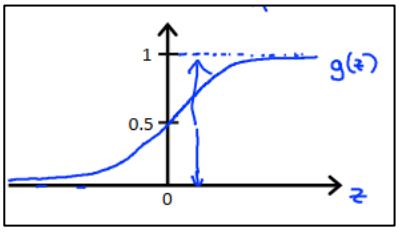

Figure 3:Sigmoid Function

➔ $\begin{cases} \theta^T x \ge 0 \ \ if \ y = 1 \\ \theta^T x < 0 \ \ if \ y = 0 \end{cases}$

→0.5 it's an example we can choose an ever value

→ $H_\theta(x) = p(y = 1 \backslash x; \theta)$

3.2 DECISION BOUNDARY

It's the plot who separated the examples x to two sets. The examples who verified $H_\theta(x) \geq 0.5$ and the examples who verified $H_\theta(x) < 0.5$

So, this plot is determinate by the resolution of the equation $\theta^T x = 0$

EXAMPLE1:

For example, we find that our model is $H_\theta(x) = g(-3 + x_1 + x_2)$

$$\begin{cases} \theta^T x \geq 0 \rightarrow y = 1 \\ \theta^T x < 0 \rightarrow y = 0 \end{cases} \rightarrow \begin{cases} x_1 + x_2 \geq 3 \rightarrow prediction\ y = 1 \\ x_1 + x_2 < 3 \rightarrow prediction\ y = 0 \end{cases}$$

➔ In this example, the decision boundary is linear

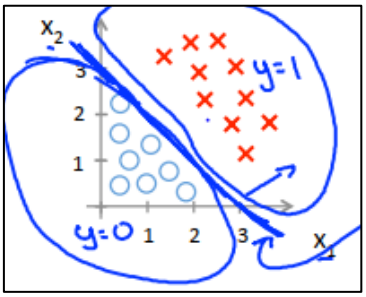

Figure 4:Example of linear decision boundary

EXAMPLE2:

For example, we find that our model is $H_\theta(x) = g(-1 + {x_1}^2 + {x_2}^2)$

$$\begin{cases} \theta^T x \geq 0 \rightarrow y = 1 \\ \theta^T x < 0 \rightarrow y = 0 \end{cases} \rightarrow \begin{cases} {x_1}^2 + {x_2}^2 \geq 1 \rightarrow prediction\ y = 1 \\ {x_1}^2 + {x_2}^2 < 1 \rightarrow prediction\ y = 0 \end{cases}$$

➔ In this example, the decision boundary is not linear

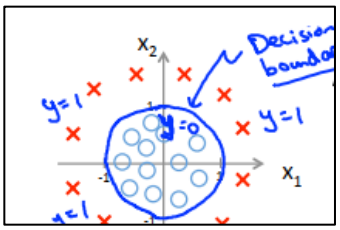

Figure 5:Example of not linear decision boundary

3.3 COST FUNCTION OF LOGISTIC REGRESSION MODEL

For logistic regression, the cost function is different than for linear regression

- ➔ In this model we will use supervised learning
- ➔ We will determine θ convenable. Means, θ who verified $H_\theta(x) \approx y$ (the prediction value is the approximation of the real value)

Our training set: {(x(i), y(i)); i=1, ..., m}

The cost function is: $J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log\left(H_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - H_\theta\left(x^{(i)}\right)\right)$

Remarque:

To determine the parameters of the logistic regression model we don't use some cost function that linear regression. Because if $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(H_\theta\left(x^{(i)}\right) - y^{(i)})^2$ for this model, this function have many local minimums. So, our parameter θ can be one of these minimums and in this case $J(\theta)$ can be too large. So, we have a bad model

How we can determinate our parameter θ of this model?

3.4 GRADIENT DESCENT

The problem is: $minimize_\theta\ J(\theta) = minimize_{\theta_0,\ldots,\theta_n}\ J(\theta)$ and

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log\left(H_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - H_\theta\left(x^{(i)}\right)\right)$$

➔ The gradient descend algorithm:

Initialization of θ

Repeat (number of iterations):

For j=0…. n:

$$\theta_j := \theta_j - \alpha \frac{dJ(\theta)}{d\theta_j}; \qquad \frac{dJ(\theta)}{d\theta_j} = \frac{1}{m}\sum_{i=1}^{m}(H_\theta\left(x^{(i)}\right) - y^{(i)}) * {x_j}^{(i)}$$

➔ $\theta_j := \theta_j - \alpha * \frac{1}{m}\sum_{i=1}^{m}(H_\theta\left(x^{(i)}\right) - y^{(i)}) * {x_j}^{(i)}$

3.5 MULTICLASS CLASSIFICATION

In some classifications problem we can found many classes: class_1, class_2…, class_p

And in this case if the result of our example is the class_i (1≤i≤p) the result is

$$y = \begin{pmatrix} 0 \\ . \\ . \\ 0 \\ y_i = 1 \\ 0 \\ . \\ . \\ 0 \end{pmatrix}$$

So, our real result is $y \in \left\{\begin{pmatrix}1\\0\\.\\.\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\.\\0\end{pmatrix}, \ldots \begin{pmatrix}0\\0\\.\\0\\1\end{pmatrix}\right\}$

logistic regression model:

$$H_\theta(x) = \begin{pmatrix} (H_\theta(x))_1 \\ (H_\theta(x))_2 \\ . \\ . \\ (H_\theta(x))_p \end{pmatrix} = \begin{pmatrix} g((\theta^{(1)})^T x) \\ g((\theta^{(2)})^T x) \\ . \\ . \\ g((\theta^{(p)})^T x) \end{pmatrix}$$

4 OVERFITTING AND UNDERFITTING PROBLEMS

4.1 DEFINITION

- Overfitting problem (or high variance): The application makes accurate predictions for the examples in the training $J(\theta) \approx 0$ but it does not generalize well to make accurate

predictions on new. Means it can give the result perfectly for the training example but when we test another example the result is too far then the reality.

- Underfitting problem (or high bias) is the opposite of overfitting. That means it is the problem when the application gives a far result for the training example. $J(\theta)$ is too large.
  - ➔ We can find these problems when we have a classification or regression problems

4.2 RAISONS OF THIS PROBLEMS

The most reasons are the number of features and the number of training examples

- If there are many features and the small number of training examples, we can find an overfitting problem.
- If there are a small number of features and the large number of training examples, we can find an underfitting problem.

EXEMPLE1:in linear regression model

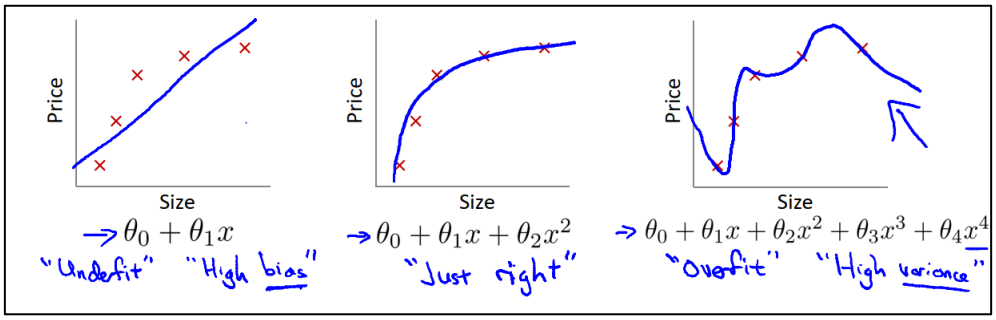

Figure 6:Overfitting and underfitting problems in linear regression model

EXAMPLE2:in Logistic regression model

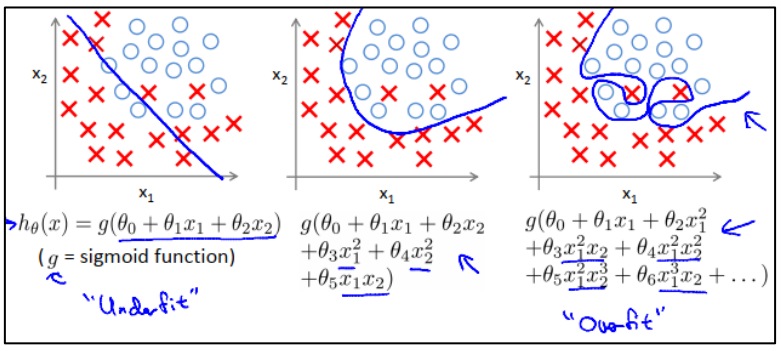

Figure 7:Overfitting and underfitting problems in logistic regression

4.3 REGULARIZATION

To not find an overfitting when we build our model, our parameters who find by minimize the cost function should be smaller. For that we will use another parameter $\boldsymbol{\lambda}$ called **regularization parameter**

4.3.1 Linear regression with regularization

Hypothesis: $H_\theta(x) = \theta^T x$

Cost function: $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(H_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$

$\lambda\sum_{j=1}^{n}\theta_j^2$: regularization term

Gradient descend:

Initialization of θ: initialize $\theta_0, \ldots., \theta_n$

For i=1…. number of iterations:

For j=0…. n:

$$\theta_j := \theta_j - \alpha * \frac{1}{m}\left(\sum_{i=1}^{m}(H_\theta(x^{(i)}) - y^{(i)}) * x_j{}^{(i)} + \lambda\theta_j\right)$$

4.3.2 Logistic regression with regularization

Hypothesis: $H_\theta(x) = g(\theta^T x)$

Cost function: $J(\theta) = -\frac{1}{m}\left(\sum_{i=1}^{m} y^{(i)} \log\left(H_\theta(x^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - H_\theta(x^{(i)})\right)\right)$
$+ \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$

$\lambda\sum_{j=1}^{n}\theta_j^2$: regularization term

Gradient descend:

Initialization of θ: initialize $\theta_0, \ldots., \theta_n$

For i=1…. number of iterations:

For j=0…. n: $\theta_j := \theta_j - \alpha * \frac{1}{m}\left(\sum_{i=1}^{m}(H_\theta(x^{(i)}) - y^{(i)}) * x_j{}^{(i)} + \lambda\theta_j\right)$

5 NEURAL NETWORKS

5.1 DEFINITION

- This is a model to solve a classifications problem
- This model is characterized by nonlinear hypothesis:

$$H_\theta(x) = g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1{}^2 x_2 + \theta_5 x_2{}^2 x_1 \dots)$$

It is a composition of many logistic regression. It composed by $L \geq 3$ layers: an input layer (layer1) and an output layer (layer L) and a few hidden layers. Every hidden layer is composed by a few units

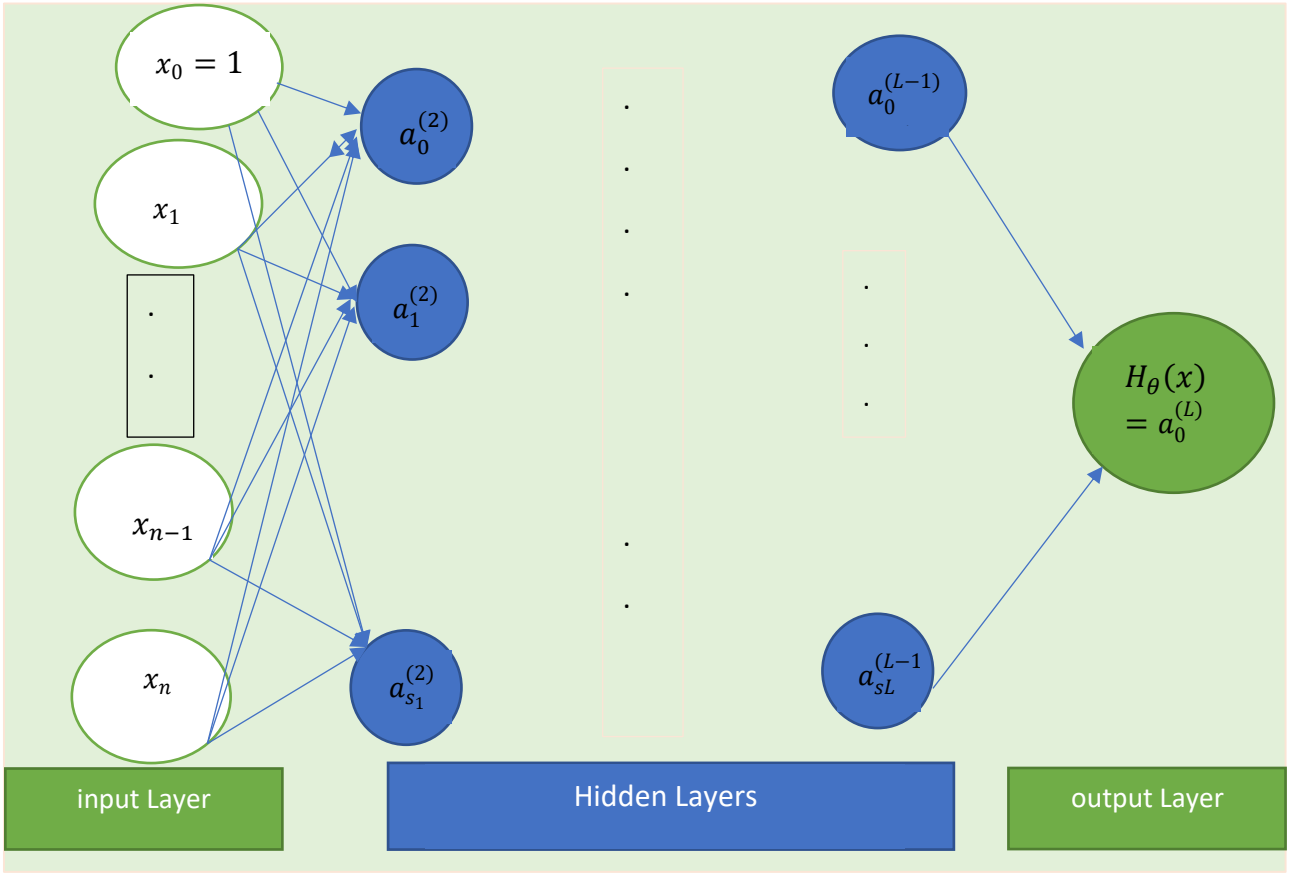

Figure 8:General representation of neural networks model

➔ In this neural network we have L hidden layers and every hidden layer l composed by s_l units

5.2 FORWARD PROPAGATION

Parameters: $\theta^{(l)}{}_p$ is the parameter of unit p in the layer $l+1$; $l \in \{1, \dots L\}$

$a^{(1)} = x$

$a^{(l)} = g(z^{(l)})$ and $z^{(l)} = (\theta^{(l-1)})^T a^{(l-1)}$ $for\ l \in \{1, \dots L\}$

With:

$$z^{(l)} = \begin{pmatrix} z_0^{(l)} \\ . \\ . \\ . \\ z_{sl}^{(l)} \end{pmatrix} \quad ; \quad a^{(l)} = \begin{pmatrix} a_0^{(l)} \\ . \\ . \\ . \\ a_{sl}^{(l)} \end{pmatrix} \quad \text{and}$$

$$\theta^{(l-1)} = \begin{pmatrix} {\theta^{(l-1)}}_{00} \dots \dots \dots {\theta^{(l-1)}}_{0sl-1} \\ . \\ . \\ . \\ {\theta^{(l-1)}}_{sl0} \dots \dots \dots {\theta^{(l-1)}}_{slsl-1} \end{pmatrix} = \begin{pmatrix} ({\theta^{(l-1)}}_1)^t \\ . \\ . \\ . \\ ({\theta^{(l-1)}}_{sl})^t \end{pmatrix}$$

➔ We have sL=1 because in this case there are only two class
But if we have K class, in the output layer we should have K units

So,

$$H_\theta(x) = \begin{pmatrix} (H_\theta(x))_1 \\ (H_\theta(x))_2 \\ . \\ . \\ (H_\theta(x))_K \end{pmatrix} = \begin{pmatrix} g(({\theta^{(L-1)}}_0)^T a^{(L-1)}) \\ g(({\theta^{(L-1)}}_1)^T a^{(L-1)}) \\ . \\ . \\ g(({\theta^{(L-1)}}_K)^T a^{(L-1)}) \end{pmatrix}$$

5.3 GENERAL COST FUNCTION

In the general case ($s_L = K \geq 1$ and $H_\theta(x^{(i)}) = \begin{pmatrix} (H_\theta(x^{(i)}))_1 \\ (H_\theta(x^{(i)}))_2 \\ . \\ . \\ (H_\theta(x^{(i)}))_K \end{pmatrix}$) and $y^{(i)} = \begin{pmatrix} {y^{(i)}}_1 \\ {y^{(i)}}_2 \\ . \\ . \\ {y^{(i)}}_K \end{pmatrix}$

$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} {y^{(i)}}_k \log\left((H_\theta(x^{(i)}))_k\right) + \left(1 - {y^{(i)}}_k\right)\log\left(1 - (H_\theta(x^{(i)}))_k\right) +$
$\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{sL}\sum_{j=1}^{sL+1}({\theta_{ji}}^{(l)})\text{^}2$

How to minimize $J(\theta)$, that means how to calculate : $minimize_\theta\, J(\theta) = minimize_{\theta^{(1)},\dots\theta^{(L-1)}}\, J(\theta)$?

5.4 BACKPROPAGATION ALGORITHM

For use the gradient descend algorithm we need $\frac{dJ(\theta)}{d\theta^{(l-1)}}$ with $\theta^{(l-1)}$ is the parameter of layer l

$$\theta^{(l-1)} = \begin{pmatrix} (\theta^{(l-1)}{}_0)^T \\ (\theta^{(l-1)}{}_1)^T \\ \cdot \\ \cdot \\ \cdot \\ (\theta^{(l-1)}{}_{sl})^T \end{pmatrix}$$ with $\theta^{(l-1)}{}_i$ is the parameter of unit i in layer l

$$(\theta^{(l-1)}{}_i)^T = \left(\theta^{(l-1)}{}_{i0} \dots \dots \theta^{(l-1)}{}_{is_{l-1}}\right)$$

So, $\theta^{(l-1)}$ is $sl * s_{l-1}$ metrics

Notes:

$$\Delta^{(l)} = \frac{d}{d\theta^{(l)}} \sum_{i=1}^{m} \sum_{k=1}^{K} y^{(i)}{}_k \log\left((H_\theta(x^{(i)}))_k\right) + \left(1 - y^{(i)}{}_k\right) \log\left(1 - (H_\theta(x^{(i)}))_k\right)$$

$$\delta^{(l)} = \mathrm{a}^{(l)} - y$$

$$D_{ij}^{(l)} = \frac{dJ(\theta)}{d\theta^{(l)}{}_{ij}}$$

We have: $\Delta^{(l)} = \delta^{(l+1)} * (\mathrm{a}^{(l)})^T$

So, to calculate $\frac{dJ(\theta)}{d\theta^{(l)}}$ we need $a^{(l+1)}$

➔ For that, we define the forward propagation and backward propagation

Algorithm:

Initialize: $\theta^{(l-1)}$ $\forall\, l \in \{2 \dots L\}$

Repeat (number of iterations) {

$$a^{(1)} = x$$

For $l = 2 \dots L$:

$$z^{(l)} = \theta^{(l-1)} * a^{(l-1)}$$

$$a^{(l)} = sigmoid(z^{(l)})$$

For $l = L - 1 \dots 1$:

$$\delta^{(l+1)} = \mathrm{a}^{(l+1)} - y$$

$$\Delta^{(l)} = \delta^{(l+1)} * (\mathrm{a}^{(l)})^T$$

$$\begin{cases} D_{ij}^{(l)} = \frac{1}{m}\Delta^{(l)}{}_{ij} \; if \; j = 0 \\ D_{ij}^{(l)} = \frac{1}{m}\Delta^{(l)}{}_{ij} + \lambda\theta^{(l)}{}_{ij} \; if \; j \neq 0 \end{cases}$$

For $l = 1 \dots L - 1$:

$$\theta_{ij}^{(l)} = \theta_{ij}^{(l)} - \alpha * D_{ij}^{(l)}$$

5.5 RANDOM INITIALIZATION OF PARAMETERS OF NEURAL NETWORKS

The initialization of the parameters of logistic regression model is $\theta = zeros(number_{features}, 1)$

But for neural networks model, if we initialize $\theta^{(l)} = zeros(sl, sl-1)$ we find $a^{(l)}{}_i = a^{(l)}{}_j \; \forall \; l, i, j$

To initialize of the parameters to program the backpropagation and minimize the cost function for the neural networks model we need the random initialization

$\theta^{(l)}$ =rand($sl, sl-1$)*(2*INIT_EPSILON) - INIT_EPSILON;

$So, -\varepsilon \leq \theta_{ij}^{(l)} \leq \varepsilon \quad \forall \; l, i, j$

5.6 GRADIENT CHECKING

We can estimate the gradient by

$$\frac{dJ(\theta)}{d\theta^{(l)}} \approx \frac{J\left(\theta^{(0)}, \dots \theta^{(l-1)}, \theta^{(l)} + \varepsilon, \theta^{(l+1)}, \dots \theta^{(L-1)}\right) + J\left(\theta^{(0)}, \dots \theta^{(l-1)}, \theta^{(l)} - \varepsilon, \theta^{(l+1)}, \dots \theta^{(L-1)}\right)}{2\varepsilon}$$

➔ This is another method then backpropagation to calculate the gradient of the cost function

5.7 REMARQUE

Backpropagation algorithm can be the solution of determination parameters for many completed model

6 ADVICE FOR APPLYING MACHINE LEARNING

If when we test our model, we find inaccessible large error in its predictions, we decide what we should apply next?

Some solutions can be helpful:

- Increase or decrease the number of training examples

- Try smaller set of features (in the case of high variance problem)
- Try getting additional features (in the case of high bias problem)
- Try adding polynomial features (in the case of high bias problem)
- Decrease or increase the regularization parameter λ

6.1 M.L DIAGNOSTIC

It's the test who can get what is the best solution to improve the machine learning application performance

6.2 EVALUATED THE MODEL

To evaluate the model after the test we need a validation set and a test set and we defined:

$J_{test}(\theta)$: the test error

$J_{cv}(\theta)$: the validation error

➔ Sometimes we use only the test set and $J_{test}(\theta)$ the test and the validation

❖ **$J_{test}(\theta)$ and $J_{cv}(\theta)$ for linear regression:**

$$\mathrm{J_{test}}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(H_\theta\left({x_{test}}^{(i)}\right) - {y_{test}}^{(i)}\right)^2 \text{ and}$$

$$\mathrm{J_{cv}}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{test}} \left(H_\theta\left({x_{cv}}^{(i)}\right) - {y_{cv}}^{(i)}\right)^2$$

❖ **$J_{test}(\theta)$ and $J_{cv}(\theta)$ for logistic regression:**

$$\mathrm{J_{test}}(\theta) = -\frac{1}{m_{test}} \left(\sum_{i=1}^{m} {y_{test}}^{(i)} \log\left(H_\theta\left({x_{test}}^{(i)}\right)\right) + (1 - {y_{test}}^{(i)}) \log\left(1 - H_\theta\left({x_{test}}^{(i)}\right)\right)\right)$$

$$\mathrm{J_{cv}}(\theta) = -\frac{1}{m_{cv}} \left(\sum_{i=1}^{m} {y_{cv}}^{(i)} \log\left(H_\theta\left({x_{cv}}^{(i)}\right)\right) + (1 - {y_{cv}}^{(i)}) \log\left(1 - H_\theta\left({x_{cv}}^{(i)}\right)\right)\right)$$

Remarque:

In the test cost function and validation cost function we shouldn't write the regularization term

6.3 EVOLUTION OF $J_{cv}(\theta)$ AND $J_{train}(\theta)$ IN FUNCTION OF DEGREE OF THE MODEL

- If d is too small→ inaccessible high error in the validation and the train sets→ underfitting problem

- If d is too large→ $J_{trian}(\theta) \approx 0$ but $J_{cv}(\theta)$ Is too large→ overfitting problem.

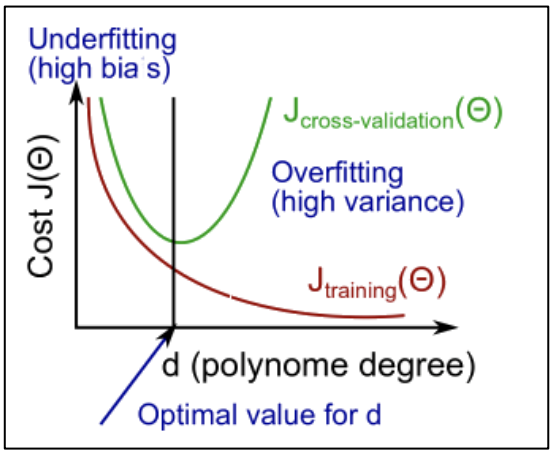

Figure 9:Jtrain and Jcv are a function of the degree of the model.

6.4 EVOLUTION OF $J_{cv}(\theta)$ AND $J_{train}(\theta)$ IN FUNCTION OF THE REGULARIZATION PARAMETER

- If λ *is too small* → $J_{trian}(\theta) \approx 0$ but $J_{cv}(\theta)$ Is too large→ overfitting problem
- If λ is too large→ inaccessible high error in the validation and the train sets→ underfitting problem

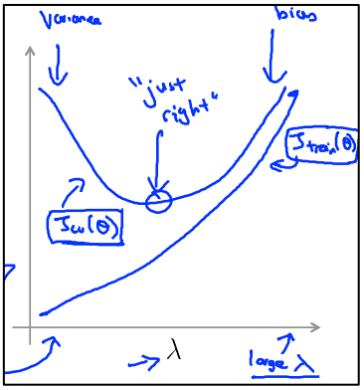

Figure 10:Jtrain and Jtest are a function of regularization parameter

6.5 LEARNING CURVES: $J_{cv}(\theta)$ AND $J_{train}(\theta)$ IN FUNCTION OF THE NUMBER OF TRAINING EXAMPLES

❖ **High bias**

- If $N\ is\ too\ small$ → $J_{trian}(\theta) \approx 0$ but $J_{test}(\theta)$Is too large
- If N is too large→ $J_{trian}(\theta)$ increase and $J_{test}(\theta)$ decrease

  ➔ The decrease of N only cannot solve the underfitting problem

  ➔ If N is too large, $J_{trian}(\theta) \approx J_{test}(\theta)$

Figure 11:Jtrain and Jtest are a function of number of training examples when we have a high bias problem

❖ **High variance**

- If $N\ is\ too\ small$ → $J_{trian}(\theta) \approx 0$ but $J_{test}(\theta)$Is too large
- If N is too large→ $J_{trian}(\theta)$ increase and $J_{test}(\theta)$ decrease

  ➔ Some if N too large, it rests $J_{trian}(\theta) \gg J_{test}(\theta)$

Figure 12:Jtrain and Jtest are a function of number of training examples when we have a high variance problem

6.6 SKEWED CLASSES AND THE THRESHOLD VALUE

In the classification problems we have

$$if\ H_\theta(x) \geq 0.5 \rightarrow\ predict = 1$$
$$if\ H_\theta(x) < 0.5 \rightarrow\ predict = 0$$

0.5 is called the threshold value. The threshold value is not fixed at 0.5. It can be another value

Remarque: We should choose the value convenable with our problem

EXAMPLE:

Considered an application who determine if the person have a cancer(y=1) or not (y=0)

NB: y=1 only if $H_\theta(x) \geq 0.9$ (That means it's very rare when the response of the application is true, and the person does not have cancer. The response is true only if it's Presque sur that the person has cancer). → In this example threshold value=0.9

	Actual True	Actual False
Predict Positive	Positive True	Positive False
Predict Negative	Negative True	Negative False

We define:

- Precision= $\frac{Positive\ True}{predict\ Positif} = \frac{Positive\ True}{Positive\ True+Positive\ False}$
- Recall = $\frac{Positive\ True}{actual\ true} = \frac{Positive\ True}{Positive\ True+Negative\ True}$

Remarque :

$0 \leq precision \leq 1$ and $0 \leq recall \leq 1$

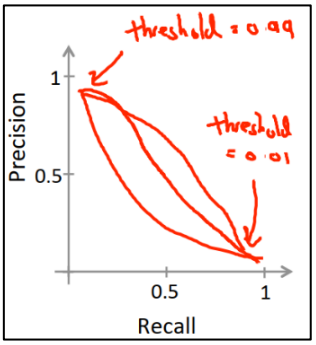

Figure 13:plot the couple (precision, recall) for different values of threshold.

➔ **precision and recall are opposite**

If the threshold is large (≈ 1) ➔ $precision \approx 1$ and $recall \approx 0$

If the threshold is large (≈ 0) ➔ $precision \approx 0$ and $recall \approx 1$

➔ We should choose the convenable value of the threshold (that depend on the problem)

For choose the most convenable threshold value we should try many values and calculate every once $\mathrm{F1_{Score}} = 2 * \frac{P*R}{P+R}$

➔ The most convenable value of the threshold is that who verified the most important value of $\mathrm{F1_{Score}}$

7 SUPPORT VECTOR MACHINE (SVM)

7.1 DEFINITION

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.

7.2 OPTIMIZATION OBJECTIVE

Suppose we have a classification problem

So, we should solve this problem:

$$minimize_\theta\ J(\theta) =$$

$$minimize_\theta(-\frac{1}{m}(\sum_{i=1}^{m} y^{(i)} \log\left(H_\theta\left(x^{(i)}\right)\right) + (1-y^{(i)}) \log\left(1-H_\theta\left(x^{(i)}\right)\right)) \ + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2)$$

$$\Leftrightarrow$$

$$minimize_\theta(\boldsymbol{C}\sum_{i=1}^{m} y^{(i)} \log\left(H_\theta\left(x^{(i)}\right)\right) + (1-y^{(i)}) \log\left(1-H_\theta\left(x^{(i)}\right)\right)) \ + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2)$$

With $C = \frac{1}{\lambda}$

➔ This is for the SVM with linear kernel ($\theta^t x$ is linear)

But the kernel ken be not linear and this case the optimization objective is:

$$minimize_\theta(\boldsymbol{C}\sum_{i=1}^{m} y^{(i)} \text{cost}_1(\theta^t x) + (1-y^{(i)})\text{cost}_1(\theta^t x)) \ + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2)$$

With

7.3 GAUSSIAN KERNEL

The kernel is: $H_\theta\left(x^{(i)}\right) = g(\theta^t f) = g(\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \cdots + \theta_m f_m)$

With $f_i = \exp(-\frac{||x-l^{(i)}||\text{^}2}{2*\sigma^2})$ and $l^{(1)}, \ldots l^{(m)}$ are called a landmarks

➔ more SVM choose $l^{(i)} = y^{(i)}\ \forall i = 1..m$

Remarque:

σ^2 is too large → higher bias and low variance

σ^2 is too small → low bias and high variance

8 UNSUPERVISED LEARNING

We know that in unsupervised learning machine we use an unlabeled to find the model.

What is the algorithm able to find the model for an unsupervised learning?

8.1 K-MEANS ALGORITHM

$\{x^{(1)}, x^{(2)}, \ldots x^{(m)}\}$ our unlabeled data

Suppose K the number of clusters

Repeat: (to the convergence)

Initialize $\mu_1, \ldots., \mu_K$ randomly

For i=1…m

$c_i = min_{1 \leq k \leq K} ||x^{(i)} - \mu_k||$ (For every training example, find the index of the more approach cluster)

For k=1…K

$\mu_k = \frac{\sum_{ci=k} x^{(i)}}{\sum_{ci=k} 1}$ (change the position of μ_k to the means between $x^{(i)}$ who verified that $ci = k$)

Remarque:

Sometimes, the result of this algorithm is bad because the initialization of $\mu_1, \ldots., \mu_K$ is bad

8.2 HOW CHOOSE THE BEST INITIALIZATION OF $\boldsymbol{\mu_1, \ldots., \mu_K}$

For choose the best initialization, we should try many initialization

We define the cost function for the unsupervised learning:

$$J(c^{(1)}, \ldots c^{(m)}, \mu_1, \ldots., \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||$$

So, every once and after determining the position of $\mu_1, \ldots., \mu_K$ we check the value of $J(c^{(1)}, \ldots c^{(m)}, \mu_1, \ldots., \mu_K)$

➔ We choose the initialization for the smaller value of this error

9 DIMENSIONALITY REDUCTION

In this part we will ask about the method to reduce the number of futures with save approximatively all information about these features

9.1 PCA: PRINCIPAL COMPONENT ANALYSIS

Suppose we have n features , $x = \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{pmatrix}$ Our objective is to reduce these features to k features $z_1, \dots z_k$ $(with\ k \leq n)$ so every training example $x^{(i)}$ transformed to $z^{(i)} = \begin{pmatrix} z_1 \\ . \\ . \\ . \\ . \\ z_k \end{pmatrix}$

9.2 PCA ALGORITHM

{x(i); i=1, ..., m} set of input (for supervised or unsupervised learning)

$sigma = \left(\frac{1}{m}\right) \sum_{i=1}^{m} x^{(i)} \left(x^{(i)}\right)^t = \left(\frac{1}{m}\right) X^t * X$

[U, S, V]= $svd(sigma)$ (svd is a function know by octave)

U$reduce = U(:,1:k)$

So, $z =$ U$reduce^t * x$

Remarque: $U \in R_{n*n}$

9.3 HOW WE CHOOSE K?

We should choose the smallest k who verified $\frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{n} s_{ii}} \approx 1$ or $\frac{\frac{1}{m}\sum_{i=1}^{m} ||x^{(i)} - x_{app}{}^{(i)}||}{\frac{1}{m}\sum_{i=1}^{m} ||x^{(i)}||} = 1 - \frac{\sum_{i=1}^{k} s_{ii}}{\sum_{i=1}^{n} s_{ii}} \approx 0$

Remarque: We can use PCA for supervised or unsupervised learning and it can help to make our model better

10 ANOMALY DETECTION

10.1 DEFINITION

It is another supervised learning model witch we can use in some particular cases

10.2 GAUSSIAN DISTRIBUTION

A feature x have a gaussian distribution if the density of this feature is

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{\left||x-\mu|\right|^2}{2\sigma^2})$$

With μ is the means of x and σ^2 the variance

If $\{x^{(1)}, x^{(2)}, \dots x^{(m)}\}$ the set of labeled data

And x_j is the jth feature of the data x so,

$$p(x_j) = \frac{1}{\sqrt{2\pi}\sigma_j}\exp(-\frac{\left||x_j-\mu_j|\right|^2}{2{\sigma_j}^2})$$

With $\mu_j = \frac{1}{m}\sum_{i=1}^{m} {x_j}^{(i)}$ and ${\sigma_j}^2 = \frac{1}{m}\sum_{i=1}^{m}({x_j}^{(i)} - \mu_j)\text{^}2$

So, if a value of the feature of our test data x_j is very different for the values of the values of this feature for the training data ${x_j}^{(i)}$, $p(x_j) \approx 0$

10.3 GAUSSIAN DISTRIBUTION ALGORITHM

$\{x^{(1)}, x^{(2)}, \dots x^{(m)}\}$ the set of labeled data with $y^{(i)} = 1 \forall\, i = 1 \dots m$ (this is a set of normal data)

For $j = 1 \dots n$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} {x_j}^{(i)}$$

$${\sigma_j}^2 = \frac{1}{m}\sum_{i=1}^{m}({x_j}^{(i)} - \mu_j)\text{^}2$$

$$p(x_j, \mu_j, {\sigma_j}^2) = \frac{1}{\sqrt{2\pi}\sigma_j}\exp(-\frac{\left||x_j-\mu_j|\right|^2}{2{\sigma_j}^2})$$

For x a test data:

$$p(x) = \prod_{j=1}^{n} p(x_j, \mu_j, {\sigma_j}^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j}\exp(-\frac{\left||x_j-\mu_j|\right|^2}{2{\sigma_j}^2})$$

$$\varepsilon: small\ value$$

If $p(x) < \varepsilon$ → x is an anomaly data

➔ That means if x is different to the normal data $\{x^{(1)}, x^{(2)}, \dots x^{(m)}\}$, $p(x) < \varepsilon$ so x is an anomaly data.

11 RECOMMENDER SYSTEMS

The recommender systems are the most important machine learning applications

11.1 EXAMPLE: PREDICTION MOVIES RATINGS

Suppose we have n_m movies and n_u users. Every user gives a rat to everyone for these movies

We create a recommender system that able to predict the rat given by the user j to the movie i

➔ This system uses the learning regression model. So, it's a supervised machine

Note: $y^{(i,j)}$ the rat given by the user j to the movie i

r (i, j) =1 if the user j gives a rat to a movies i → this is a training example

r (i, j) =0 if the user j doesn't give the rat to a movie i ($y^{(i,j)} = ?$)→ the model should predict the rat

For example:

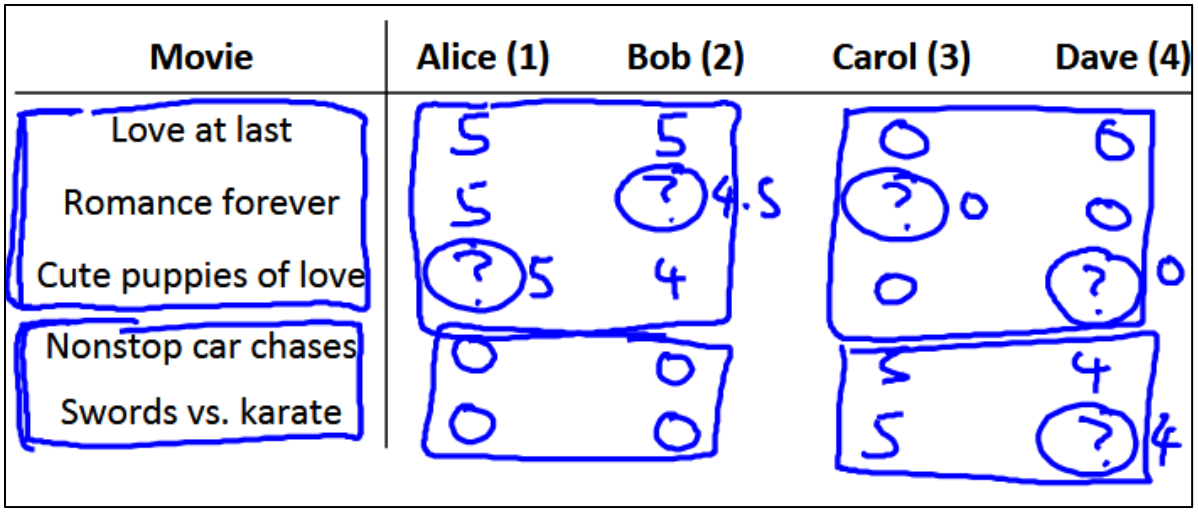

Figure 14:Table of the different rates to different movies given by different users

First, we need to find the features who give all the important information about the input (movie)

For example, x_1: percentage of action and x_2: percentage of comedy

So, for every user j we have $\{ (x^{(i)}, y^{(i)}) \ ; r(i,j) = 1)\}$ training set with $x^{(i)}$ represent the movie i and $y^{(i)}$ represent the rat given to this movie with $m = \sum_{i=1}^{n_m} r(i,j)$

And for every user j, the hypothesis: $H_\theta(x) = \theta^T x$, the parameter: $\theta^{(j)}$

and the optimization objective:

$$min_{\theta^{(j)}} J(\theta^{(j)}) = min_{\theta_j} (\frac{1}{2} \sum_{\substack{i=1 \\ r(i,j)=1}}^{n_m} (H_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \theta_k^2)$$

➔ We can optimize this by determine $\theta^{(1)}, \dots \theta^{(n_u)}$ In one algorithm. So, our optimization objective is:

$$min_{\theta^{(1)},\dots\theta^{(n_u)}} J(\theta^{(1)}, \dots \theta^{(n_u)})$$
$$= min_{\theta^{(1)},\dots\theta^{(n_u)}} (\frac{1}{2} \sum_{j=1}^{n_u} \sum_{\substack{i=1 \\ r(i,j)=1}}^{n_m} (\theta^{(j)^T} x^{(i)} - y^{(i,j)}) \text{^} 2$$
$$+ \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \theta^{(j)}{}_k^2)$$

11.2 How we can find $x^{(i)}, i = 1 \dots n_m$ if we have $\theta^{(j)}$, j=1 … n_u

For find $x^{(i)}, i = 1 \dots n_m$ we define this optimization objective:

$$min_{x^{(1)},\dots x^{(n_m)}} J(x^{(1)}, \dots x^{(n_m)})$$
$$= min_{x^{(1)},\dots x^{(n_m)}} (\frac{1}{2} \sum_{i=1}^{n_m} \sum_{\substack{j=1 \\ r(i,j)=1}}^{n_u} (\theta^{(j)^T} x^{(i)} - y^{(i,j)}) \text{^} 2$$
$$+ \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} x^{(i)}{}_k^2)$$

11.3 Collaborative filtering

If we need to determine in the sometime the values of the input $x^{(i)}$ and the rats $y^{(i,j)}$ when

r (i, j) =0

So, we need to find collaboratively $x^{(i)}$ and $\theta^{(j)}$

For that we use this algorithm who can solve this problem

Initialize $\theta^{(1)}, \dots \theta^{(n_u)}, x^{(1)}, \dots x^{(n_m)}$

Determine the solution of this optimization problem:

$$min_{x^{(1)},\dots x^{(n_m)},\theta^{(1)},\dots\theta^{(n_u)}} J(x^{(1)}, \dots x^{(n_m)}, \theta^{(1)}, \dots \theta^{(n_u)}) =$$
$$min_{x^{(1)},\dots x^{(n_m)},\theta^{(1)},\dots\theta^{(n_u)}} (\frac{1}{2} \sum_{\substack{i,j \\ r(i,j)=1}} (\theta^{(j)^T} x^{(i)} - y^{(i,j)}) \text{^} 2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \theta^{(j)}{}_k^2 +$$
$$\frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} x^{(i)}{}_k^2)$$

By the gradient descend algorithm:

Repeat {

$$x^{(i)}{}_k = x^{(i)}{}_k - \alpha((\sum_{j:r(i,j)=1}\left(\theta^{(j)^T}x^{(i)} - y^{(i,j)}\right) * \theta^{(j)}{}_k) \ + \lambda x^{(i)}{}_k)$$

$$\theta^{(j)}{}_k = \theta^{(j)}{}_k - \alpha((\sum_{i:r(i,j)=1}\left(\theta^{(j)^T}x^{(i)} - y^{(i,j)}\right) * x^{(i)}{}_k) \ + \lambda\theta^{(i)}{}_k)$$

11.4 VECTORIZATION

For program this algorithm easier, we need to vectorize

$$\theta = \begin{pmatrix} -- \theta^{(1)^t} -- \\ . \\ . \\ . \\ -- \theta^{(n_u)^t} -- \end{pmatrix}, X = \begin{pmatrix} -- x^{(1)^t} -- \\ . \\ . \\ . \\ -- x^{(n_m)^t} -- \end{pmatrix}$$

$$Predictions = \theta * X^t = \begin{pmatrix} y^{(1,1)} \dots \dots \dots y^{(1,n_u)} \\ . \\ . \\ . \\ y^{(n_m,1)} \dots \dots \dots y^{(n_m,n_u)} \end{pmatrix}$$

With $y^{(i,j)}$ is the predict of the rat given by the user j to the movie i.

Suppose $R = (r(i,j))_{i,j}$ with $\begin{cases} r(i,j) = 0 \ if \ y(i,j) = ? \\ r(i,j) = 1 \quad else \end{cases}$

So, the cost function with regularization is

$$J(\ x^{(1)}, \dots x^{(n_m)}, \theta^{(1)}, \dots \theta^{(n_u)}) = \frac{1}{2} * sum(C(:)) + \lambda * sum(D(:)) \ with$$

$$C = ((\theta * X^t - Y).^2).* R$$

$$D = \theta(:, 2: end).\hat{}2$$

And the metrics of gradients are:

$$X_{grad} = ((\theta * X^t - Y).^* R) * \theta + \lambda * X$$

$$\theta_{grad} = ((\theta * X^t - Y).* R)' * X + \lambda * \theta$$