Notes for Machine Learning

Jack Wang

April 11, 2015

Contents

1 Introducition (1)

1.1 Definition of Machine Learning (ML)

TEP 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**.

Machine learning the field of study that gives computers the ability to learn without being explicitly programmed.

1.2 Types of learning

Supervised learning Given {Features, Label} to estimate the label for new data.

- regression (target variable is continuous)
- classfication (target variable is discrete)

Un-supservised learning Given {Features} to do somthing, e.g, clustering. **No lables are given.**

- clustering (image processing to 3D, seprate sounds from two person(SVD) [1])

[1] Is it necessary to have two speakers? What if only one speaker and two person?

Reinforcement learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward

- robot

1.3 Todo

How to seprate sound from two person?

2 Linear Regression (2,3,4,5)

2.1 Regression

Notation	Meaning
$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{(n+1)\times 1}$	input variables, features
y	output/target variable
$\hat{y} = h(\mathbf{x}) = h_{\boldsymbol{\theta}}(\mathbf{x})$	hypothesis function of $\mathbf{x}$ with parameter $\boldsymbol{\theta}$
$\mathbf{x}^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}$	i^{th} observation of $\mathbf{x}$ where $1 \leq i \leq m$
$y^{(i)}$	i^{th} observation of y where $1 \leq i \leq m$
$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^{m\times 1}$	m observations of y
$\mathbf{X} = \begin{bmatrix} \left(\mathbf{x}^{(1)}\right)^T \\ \left(\mathbf{x}^{(2)}\right)^T \\ \vdots \\ \left(\mathbf{x}^{(m)}\right)^T \end{bmatrix} \in \mathbb{R}^{m\times(n+1)}$	m observations of $\mathbf{x}$
m	# of traing examples
n	# of traing features (excluding addtional 1 vector)
$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$	parameter of hypothesis funtion(model)

Table 1: Notation of Math symbols

TEP defintion of regression:

- E: Given m observations: $(\mathbf{X}, \mathbf{y})$,

- T: Predict y using model $h(\mathbf{x}) = h_{\boldsymbol{\theta}}(\mathbf{x})$,
- P: which minimizes loss(error) fuction $J(\boldsymbol{\theta})$

After selecting the model (e.g, linear model), T becomes to find $\arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ where:

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\right)^2 \quad \text{(SSE)} \tag{1}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\right)^2 \quad \text{(MSE)} \tag{2}$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} = \begin{bmatrix} \left(\mathbf{x}^{(1)}\right)^T \\ \left(\mathbf{x}^{(2)}\right)^T \\ \vdots \\ \left(\mathbf{x}^{(m)}\right)^T \end{bmatrix} \in \mathbb{R}^{m \times (n+1)} \tag{3}$$

- **解析解**
- Iteration method Gradicent decent method
- Regression models:
  - linear model $h_{\boldsymbol{\theta}}(\mathbf{x}) = \langle \boldsymbol{\theta}, \mathbf{x} \rangle = \mathbf{x}^T \boldsymbol{\theta}$
  - poly model

2.2 Model and Cost function

2.3 Parameter learning

Traing Set -> Learning Algorithem (Chosen) -> $h_\theta(x) = h(x)$

2.4 Demo of Gradicent Decent

All figures 1, 2, 3, 4, 5 and 6 are generated by `../src/demo_gradient_descent.py`
We can find:

- No scaling, Gradicent Decent **在不同维度上的陡峭程度不一样（二维中轮廓图由圆变为椭圆，等高线越密集，越陡峭），越陡峭, 下降降幅度越厉害，梯度下降法给予该陡峭维度的权重比较大，所以会沿着陡峭维度的主方向下降，之后是第二个陡峭的维度成为主方向，以此类推。与圆形轮廓图相比，误入“歧图”，多乖了个“弯儿”**
- Gradicent Decent can only find the **local minimum** point
- The step size during update of θ will reduce automatically while approaching non-steep surface, even for the same **learning rate**

2.4.1 $J(\ \#$ **of iterations** $)$

2.4.2 $J(\theta_0, \theta_1)$ **in 3D**

2.4.3 $J(\theta_0, \theta_1)$ **in contour**

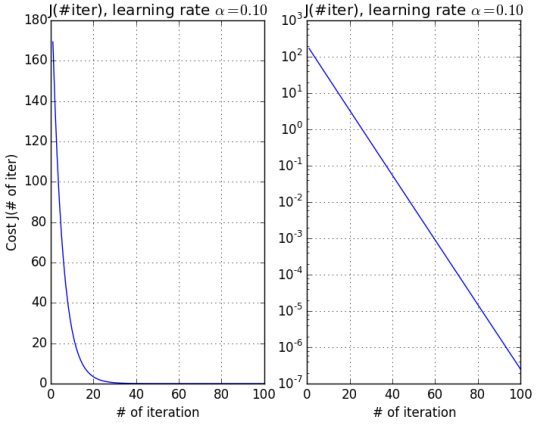

Figure 1: J(# of iterations)

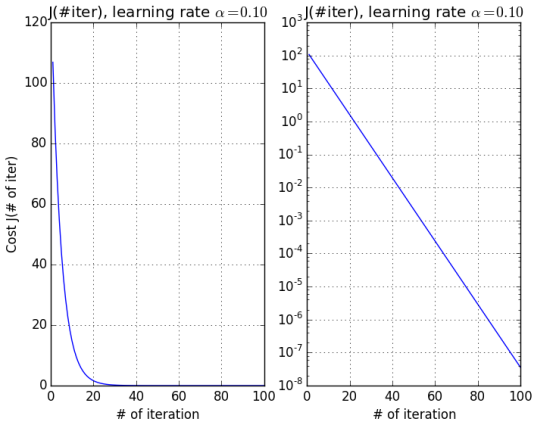

Figure 2: J(# of iterations) without scaling

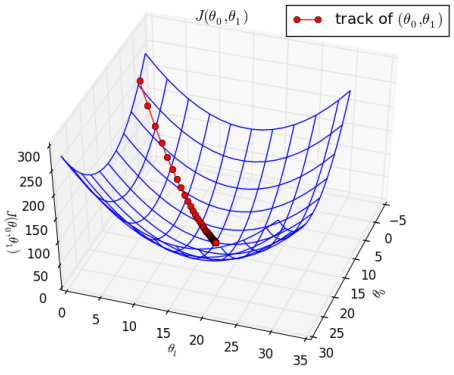

Figure 3: $J(\theta_0, \theta_1)$ in 3D

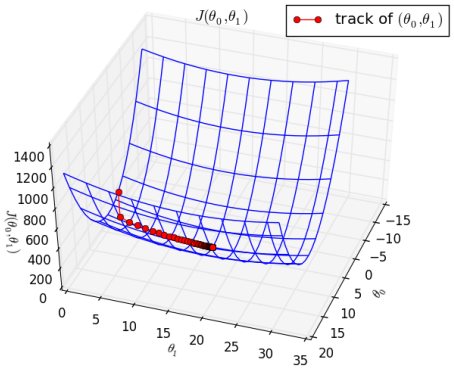

Figure 4: $J(\theta_0, \theta_1)$ in 3D without scaling

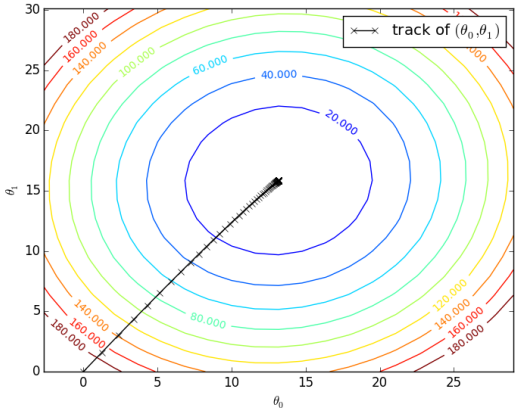

Figure 5: $J(\theta_0, \theta_1)$ in contour

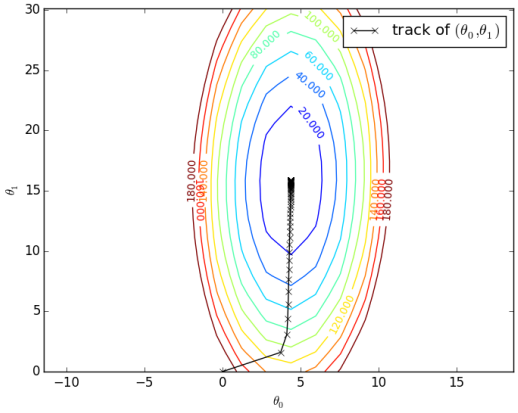

Figure 6: $J(\theta_0, \theta_1)$ in contour without scaling