

Notes

Machine Learning by Andrew Ng on Coursera

Sparsh Jain

November 23, 2020

Contents

1	Introduction	6
1.1	Supervised Learning	6
1.1.1	Regression Problem	6
1.1.2	Classification Problems	6
1.2	Unsupervised Learning	6
I	Supervised Learning	7
2	Linear Regression with One Variable	8
2.1	Notations	8
2.2	Supervised Learning	8
2.3	Gradient Descent	9
2.4	Gradient Descent for Linear Regression	9
3	Linear Algebra	10
3.1	Matrix	10
3.2	Vector	10
3.3	Addition and Scalar Multiplication	11
3.4	Matrix Matrix Multiplication	11
3.5	Inverse and Transpose	11
4	Linear Regression with Multiple Variables	13
4.1	Notations	13
4.2	Hypothesis	13
4.3	Gradient Descent	14
4.3.1	Feature Scaling	14
4.3.2	Mean Normalization	15
4.3.3	Learning Rate	15
4.4	Features and Polynomial Regression	15
4.4.1	Features	15
4.4.2	Polynomial Regression	15

4.5	Normal Equation	16
4.5.1	Non Invertibility of $X^T X$	16
5	Octave Tutorial	17
6	Classification	18
6.1	Logistic Regression	18
6.2	Decision Boundary	19
6.3	Cost Function	19
6.4	Advanced Optimization:	21
6.5	Multi-Class Classification	22
6.5.1	One vs All	22
7	Regularization	23
7.1	Problem of Overfitting	23
7.2	Addressing Overfitting	23
7.3	Cost Function	24
7.4	Regularized Linear Regression	24
7.5	Regularized Logistic Regression	26
8	Neural Networks	27
8.1	Non-Linear Hypothesis	27
8.2	Neural Networks	27
8.3	Model Representation	27
8.4	Notations	28
8.5	Forward Propagation	29
8.6	Multi-Class Classification	30
9	Back Propagation	31
9.1	Notations	31
9.2	Cost Function	32
9.3	Backpropagation Algorithm	32
9.3.1	Gradient Computation	32
9.4	Implementation Note	34
9.4.1	Unrolling Parameters	34
9.4.2	Learning Algorithm	35
9.5	Gradient Checking	35
9.5.1	Numerical Estimation of gradients	35
9.5.2	Implementation Note:	37
9.6	Random Initialization	37
9.7	Putting it all together	38

10 Applying Machine Learning	40
10.1 Deciding what to try next	40
10.1.1 Debugging a Learning Algorithm	40
10.1.2 Machine Learning Diagnostics	40
10.2 Evaluating your Hypothesis	41
10.2.1 The Standard Way	41
10.2.2 Train/Test Procedure	41
10.3 Model Selection	41
10.3.1 Evaluating your hypothesis	42
10.4 Bias vs Variance	43
10.4.1 Effect of regularization	43
10.5 Learning Curves	44
10.5.1 High Bias	44
10.5.2 High Variance	45
10.6 Deciding what to try next (Revisited)	45
10.6.1 Neural Networks	45
11 Machine Learning System Design	46
11.1 Prioritizing what to work on: Spam Classification Example	46
11.1.1 Time Management?	46
11.2 Error Analysis	47
11.2.1 Recommended approach	47
11.2.2 Numerical Evaluation	47
11.3 Skewed Classes	48
11.3.1 Precision/Recall	48
11.3.2 Trade-Off	49
11.3.3 Working with Large Data	49
12 Support Vector Machines	50
12.1 Optimization Objective	50
12.1.1 Alternative view of logistic regression	50
12.1.2 SVM Hypothesis	51
12.2 Large Margin	52
12.2.1 Intuition	52
12.2.2 Math	52
12.3 Kernels	53
12.3.1 Non-linear Decision Boundary	53
12.3.2 Choosing the Landmarks	54
12.3.3 SVM Parameters:	55
12.4 Using SVM	55
12.4.1 Multi-Class Classification	56

12.4.2 Logistic Regression vs SVM	56
II Unsupervised Learning	57
13 Clustering	58
13.1 Unsupervised Learning	58
13.2 Clustering	58
13.2.1 Application	58
13.3 K -means	58
13.3.1 Algorithm	58
13.3.2 K -means for non-separated clusters	59
13.3.3 Optimization Objective	59
13.3.4 Random Initialization	60
13.3.5 Choosing K	60
14 Dimensionality Reduction	61
14.1 Motivation	61
14.1.1 Data Compression	61
14.1.2 Data Visualization	61
14.2 Principle Component Analysis	61
14.2.1 Problem Formulation	61
14.3 Algorithm	62
14.3.1 Data Preprocessing	62
14.3.2 Procedure	62
14.3.3 Reconstruction	63
14.4 Choosing the number of principle components	63
14.4.1 Algorithm	64
14.5 Applying PCA	64
14.5.1 Supervised Learning Speedup	64
14.5.2 Visualization	64
14.5.3 Bad use of PCA: To prevent overfitting	65
14.5.4 PCA is sometimes used where it shouldn't be	65
15 Anomaly Detection	66
15.1 Example	66
15.1.1 Aircraft Engine!	66
15.1.2 Fraud Detection	66
15.1.3 Manufacturing	66
15.1.4 Monitoring computers in a data center	66
15.2 Gaussian (Normal) Distribution	67
15.2.1 Definition	67

15.2.2	Parameter Estimation	67
15.3	Algorithm	67
15.3.1	Density Estimation	67
15.3.2	Anomaly Detection Algorithm	68
15.4	Developing and Evaluating an Anomaly Detection System	69
15.4.1	The importance of real-number evaluation	69
15.4.2	Algorithm Evaluation	69
15.5	Anomaly Detection vs Supervised Learning	70
15.6	Choosing what features to use	70
15.6.1	Transformations	70
15.6.2	Error analysis for anomaly detection	71
15.6.3	Other techniques:	71
15.7	Multivariate Gaussian (Normal) Distribution	71
15.8	Anomaly Detection using Multivariate Gaussian Distribution	72
15.8.1	Model	72
15.8.2	Anomaly Detection	72
15.8.3	Relationship to original model	73
16	Recommender Systems	74
16.1	Problem Formulation	74
16.2	Content-Based Recommendations	74
16.2.1	Optimization Objective	75
16.2.2	Optimization Algorithm	75
16.3	Collaborative Filtering	76
16.3.1	Algorithm	76
16.4	Vectorization: Low Rank Matrix Vectorization	77
16.4.1	Vectorization	77
16.4.2	Finding related movies	78
16.5	Implementational Detail: Mean Normalization	78

Part I

Supervised Learning

Part II

Unsupervised Learning

Chapter 16

Recommender Systems

16.1 Problem Formulation

Example: Predicting movie ratings! Collect movie ratings from users for the movies they have watched, and predict ratings for the movies they haven't watched.

Notations:

n_u = no. users

n_m = no. movies

$r(i, j) = 1$ if user j has rated movie i

$y^{(i,j)}$ = rating given by user j to movie i
(defined only if $r(i, j) = 1$)

16.2 Content-Based Recommendations

Features could be the degree of genre of the movie, say romance or action.

One way could be, for each user j , learn a parameter $\theta^{(j)} \in R^{n+1}$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

Notations:

n = no. features

$\theta^{(j)}$ = parameter vector for user j

$x^{(i)}$ = feature vector for movie i

$(\theta^{(j)})^T x^{(i)}$ = predicted rating for user j , movie i

$m^{(j)}$ = no. movies rated by user j

To learn $\theta^{(j)}$:

$$\min_{\theta^{(j)}} \left(\frac{1}{2m^{(j)}} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2m^{(j)}} \sum_{k=1}^n (\theta_k^{(j)})^2 \right)$$

For recommender systems, we make some simplifications:

- Get rid of the constant $m^{(j)}$

16.2.1 Optimization Objective

To learn $\theta^{(j)}$ (parameter for user j):

$$\min_{\theta^{(j)}} J(\theta^{(j)})$$
$$\min_{\theta^{(j)}} \left(\frac{1}{2} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^n (\theta_k^{(j)})^2 \right)$$

To learn $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} J(\theta^{(1)}, \dots, \theta^{(n_u)})$$
$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \left(\frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2 \right)$$

16.2.2 Optimization Algorithm

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda \theta_k^{(j)} \right) \quad (\text{for } k = 0)$$

16.3 Collaborative Filtering

Feature Learning!

Each user j just tells us how much they like different types of movies $\theta^{(j)}$. If we get these parameters from the users, then it is possible to infer what are the features for each movie from the ratings.

Optimization Algorithm

Given $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$, to learn $x^{(i)}$:

$$\min_{x^{(i)}} \left(\frac{1}{2} \sum_{j:r(i,j)=1} \left(\left(\theta^{(j)} \right)^T \left(x^{(i)} \right) - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^n \left(x_k^{(i)} \right)^2 \right)$$

Given $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$, to learn $x^{(1)}, \dots, x^{(n_m)}$:

$$\min_{x^{(i)}} \left(\frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left(\left(\theta^{(j)} \right)^T \left(x^{(i)} \right) - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n \left(x_k^{(i)} \right)^2 \right)$$

Collaboration

Guess $\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \dots$

16.3.1 Algorithm

Minimizing $x^{(1)}, \dots, x^{(n_m)}$ and $\theta^{(1)}, \dots, \theta^{(n_u)}$ simultaneously:

$$\begin{aligned} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) &= \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left(\left(\theta^{(j)} \right)^T x^{(i)} - y^{(i,j)} \right)^2 \\ &\quad + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n \left(x_k^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n \left(\theta_k^{(j)} \right)^2 \end{aligned}$$

$$\min_{\substack{x^{(1)}, \dots, x^{(n_m)} \\ \theta^{(1)}, \dots, \theta^{(n_u)}}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$$

When learning features this way, we're going to do away with $x_0 = 1$ and hence, now our $x \in \mathbb{R}^n$ and $\theta \in \mathbb{R}^n$, and the reason is that now that we're learning the features, if there is a need for such a feature, the algorithm can learn it by itself.

Collaborative Filtering Algorithm

1. Initialize $x^{(1)}, \dots, x^{(n_m)}$ and $\theta^{(1)}, \dots, \theta^{(n_u)}$ to small random values (just like neural network).
2. Minimize $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$ using Gradient Descent (or an advanced optimization algorithm).
E.g. for every $j = 1, \dots, n_u, i = 1, \dots, n_m$:

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

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

3. For a user with parameters θ and a movie with (learned) features x , predict a star rating of $\theta^T x$.

16.4 Vectorization: Low Rank Matrix Vectorization

16.4.1 Vectorization

$$Y = [y^{(i,j)}]$$

$$= \left[(\theta^{(j)})^T x^{(i)} \right]$$

$$X = \begin{bmatrix} \text{---} & (x^{(1)})^T & \text{---} \\ \text{---} & (x^{(2)})^T & \text{---} \\ & \vdots & \\ \text{---} & (x^{(n_m)})^T & \text{---} \end{bmatrix}$$

$$\Theta = \begin{bmatrix} \text{---} & (\theta^{(1)})^T & \text{---} \\ \text{---} & (\theta^{(2)})^T & \text{---} \\ & \vdots & \\ \text{---} & (\theta^{(n_u)})^T & \text{---} \end{bmatrix}$$

$$Y = X\Theta^T$$

Name *Low Rank Matrix Vectorization* comes from the fact that $X\Theta^T$ is a *Low Rank Matrix*.

16.4.2 Finding related movies

Q: For each product i , we learn a feature vector $x^{(i)} \in \mathbb{R}^n$. How to find movies j related to movie i ?

A: Small $\|x^{(i)} - x^{(j)}\| \rightarrow$ movies j and i are similar.

5 most similar movies to i :

Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$

16.5 Implementational Detail: Mean Normalization

Let's consider an example of a user who hasn't rated any movie. So in our optimization objective

$$\min_{\substack{x^{(1)}, \dots, x^{(n_m)} \\ \theta^{(1)}, \dots, \theta^{(n_u)}}} \frac{1}{2} \sum_{(i,j): r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n \left(x_k^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n \left(\theta_k^{(j)} \right)^2$$

the first term plays no role, and hence we have to minimize $\frac{\lambda}{2}(\theta_1^2 + \theta_2^2)$ which

will give $\theta = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and so we will predict $\theta^T x = 0$ for all x , which will give us no good way to recommend movies to the new user. So, calculate mean rating for each movie and subtract the mean from all ratings (so that each movie has a mean rating of zero).

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

$$\mu = \begin{bmatrix} \text{mean}(y^{(1)}) \\ \text{mean}(y^{(2)}) \\ \vdots \\ \text{mean}(y^{(n_m)}) \end{bmatrix}$$

$$Y_{new} := Y - \mu$$

Now, learn $\theta^{(j)}, x^{(i)}$ on this Y_{new} .

For user j , on the movie i predict: $(\theta^{(j)})^T (x^{(i)} + \mu_i)$

Check Lecture16.pdf for more details.