Notes

Machine Learning by Andrew Ng on Coursera

Sparsh Jain

November 23, 2020

Contents

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:

$$\begin{aligned} n_u &= \text{no. users} \\ n_m &= \text{no. movies} \\ r(i,j) &= 1 \text{ if user } j \text{ has rated movie } i \\ y^{(i,j)} &= \text{rating given by user } j \text{ to movie } i \\ &\quad \text{(defined only if } r(i,j) = 1) \end{aligned}$$

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:

$$\begin{aligned}
n &= \text{no. features}\\
\theta^{(j)} &= \text{parameter vector for user } j\\
x^{(i)} &= \text{feature vector for movie } i\\
(\theta^{(j)})^T x^{(i)} &= \text{predicted rating for user } j\text{, movie } i\\
m^{(j)} &= \text{no. movies rated by user } j
\end{aligned}$$

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

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

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):

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

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

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

16.2.2 Optimization Algorithm

Gradient descent update:

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

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

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

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

Collaboration

Guess $\theta \to x \to \theta \to x \to \ldots$

16.3.1 Algorithm

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

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

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

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 $j = 1,\dots,n_u, i = 1,\dots,n_m$:

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

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

$$\begin{aligned}
Y &= \left[y^{(i,j)}\right] \\
&= \left[\left(\theta^{(j)}\right)^T x^{(i)}\right] \\
X &= \begin{bmatrix} — & \left(x^{(1)}\right)^T & — \\ — & \left(x^{(2)}\right)^T & — \\ & \vdots & \\ — & \left(x^{(n_m)}\right)^T & — \end{bmatrix} \\
\Theta &= \begin{bmatrix} — & \left(\theta^{(1)}\right)^T & — \\ — & \left(\theta^{(2)}\right)^T & — \\ & \vdots & \\ — & \left(\theta^{(n_u)}\right)^T & — \end{bmatrix} \\
Y &= X\Theta^T
\end{aligned}$$

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 $\left\| x^{(i)} - x^{(j)} \right\| \rightarrow$ movies j and i are similar.
5 most similar movies to i:
Find the 5 movies j with the smallest $\left\| x^{(i)} - x^{(j)} \right\|$

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

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

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.