share.coursera.org

ML:Recommender Systems - Coursera

Recommendation is currently a very popular application of machine learning.

Say we are trying to recommend movies to customers. We can use the following definitions

$nu =$ number of users

$nm =$ number of 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 $r(i, j) = 1$)

We can introduce two features, x_1 and x_2 which represents how much romance or how much action a movie may have (on a scale of $0 - 1$).

One approach is that we could do linear regression for every single user. For each user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

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

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

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

$m^{(j)} =$ number of movies rated by user j

To learn $\theta^{(j)}$, we do the following

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

This is our familiar linear regression. The base of the first summation is choosing all i such that $r(i, j) = 1$.

To get the parameters for all our users, we do the following:

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

We can apply our linear regression gradient descent update using the above cost function.

The only real difference is that we **eliminate the constant $\frac{1}{m}$**.

It can be very difficult to find features such as "amount of romance" or "amount of action" in a movie. To figure this out, we can use *feature finders*.

We can let the users tell us how much they like the different genres, providing their parameter vector immediately for us.

To infer the features from given parameters, we use the squared error function with regularization over all the users:

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

You can also **randomly guess** the values for theta to guess the features repeatedly. You will actually converge to a good set of features.

To speed things up, we can simultaneously minimize our features and our parameters:

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

It looks very complicated, but we've only combined the cost function for theta and the cost function for x.

Because the algorithm can learn them itself, the bias units where $x_0 = 1$ have been removed, therefore $x \in \mathbb{R}^n$ and $\theta \in \mathbb{R}^n$.

These are the steps in the algorithm:

1. Initialize $x^{(i)}, \ldots, x^{(nm)}, \theta^{(1)}, \ldots, \theta^{(nu)}$ to small random values. This serves to break symmetry and ensures that the algorithm learns features $x^{(i)}, \ldots, x^{(nm)}$ that are different from each other.

2. Minimize $J(x^{(i)}, \ldots, x^{(nm)}, \theta^{(1)}, \ldots, \theta^{(nu)})$ using gradient descent (or an advanced optimization algorithm).
   E.g. for every $j = 1, \ldots, nu, i = 1, \ldots nm$:
   $$x_k^{(i)} := x_k^{(i)} - \alpha \left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)} \right)$$
   $$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

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

Given matrices X (each row containing features of a particular movie) and Θ (each row containing the weights for those features for a given user), then the full matrix Y of all predicted ratings of all movies by all users is given simply by: $Y = X\Theta^T$.

Predicting how similar two movies i and j are can be done using the distance between their respective feature vectors x. Specifically, we are looking for a small value of $||x^{(i)} - x^{(j)}||$.

If the ranking system for movies is used from the previous lectures, then new users (who have watched no movies), will be assigned new movies incorrectly. Specifically, they will be assigned θ with all components equal to zero due to

the minimization of the regularization term. That is, we assume that the new user will rank all movies 0, which does not seem intuitively correct.

We rectify this problem by normalizing the data relative to the mean. First, we use a matrix Y to store the data from previous ratings, where the ith row of Y is the ratings for the ith movie and the jth column corresponds to the ratings for the jth user.

We can now define a vector

$$\mu = [\mu_1, \mu_2, ..., \mu_{nm}]$$

such that

$$\mu_i = \frac{\sum_{j:r(i,j)=1} Y_{i,j}}{\sum_j r(i,j)}$$

Which is effectively the mean of the previous ratings for the ith movie (where only movies that have been watched by users are counted). We now can normalize the data by subtracting u, the mean rating, from the actual ratings for each user (column in matrix Y):

As an example, consider the following matrix Y and mean ratings μ:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 4 & ? & ? & 0 \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}, \quad \mu = \begin{bmatrix} 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$

The resulting Y' vector is:

$$Y' = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 \\ 2 & ? & ? & -2 \\ -.2.25 & -2.25 & 3.75 & 1.25 \\ -1.25 & -1.25 & 3.75 & -1.25 \end{bmatrix}$$

Now we must slightly modify the linear regression prediction to include the mean normalization term:

$$(\theta^{(j)})^T x^{(i)} + \mu_i$$

Now, for a new user, the initial predicted values will be equal to the μ term instead of simply being initialized to zero, which is more accurate.

Next: Large Scale Machine Learning Back to Index: Main