Elements of Scientific Computing with Julia

March 5, 2015

Introduction

*I*n this lecture we are going to explore a further application of linear regression: Recommender Systems.

Introduction

In this lecture we are going to explore a further application of linear regression: Recommender Systems.

These are systems that allows businesses (or otherwise content providers) to more *intelligently* provide their content based on their customers preferences.

Introduction

In this lecture we are going to explore a further application of linear regression: Recommender Systems.

These are systems that allows businesses (or otherwise content providers) to more *intelligently* provide their content based on their customers preferences.

For example: movie recommendations on Netflix or product recommendations on amazon.com.

Main Types

There are two main types of recommender systems:

Main Types

There are two main types of recommender systems:

1. *Content based systems:* these are systems that base recommendations on the content of the media or information. For example, if you watch actions movies on Netflix and rate those positively, chances are that you will enjoy other action movies.

Main Types

There are two main types of recommender systems:

1. *Content based systems:* these are systems that base recommendations on the content of the media or information. For example, if you watch actions movies on Netflix and rate those positively, chances are that you will enjoy other action movies.
2. *Collaborative filtering systems:* these are systems that base recommendations on similarities amongst users or customers. For example, if you bought a book on amazon that another user also bought along with other books, then maybe you would like those other books too.

Most recommender systems in use are based on these two main types or on some sort of hybrid of these types.

Most recommender systems in use are based on these two main types or on some sort of hybrid of these types.

There are many techniques from scientific computing and machine learning that can be applied to implement these different types of recommender systems - such as singular value decomposition and clustering analysis (which are not covered in this course).

Here, we will discuss simplified implementations of both content based and collaborative filtering systems that are basically applications of linear regression.

Here, we will discuss simplified implementations of both content based and collaborative filtering systems that are basically applications of linear regression.

We will focus on algorithms that use gradient-based optimization (such as gradient descent or some more advanced optimization algorithm like L-BFGS) to find the least squares solution.

Here, we will discuss simplified implementations of both content based and collaborative filtering systems that are basically applications of linear regression.

We will focus on algorithms that use gradient-based optimization (such as gradient descent or some more advanced optimization algorithm like L-BFGS) to find the least squares solution.

However, before we "jump in", let us take a detour through the topic of *regularization*.

Regularization

Regularization refers to the technique of adding extra information to our linear (or logistic) regression in other to give preferential treatment to features that seem more relevantly correlated and penalize features whose correlation may not be that relevant to solving the regression.

Regularized Cost Function and its Derivative

$$C(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

Regularized Cost Function and its Derivative

$$C(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

for $j = 0$:

$$\frac{\partial}{\partial\theta_0}C(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

Regularized Cost Function and its Derivative

$$C(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

for $j = 0$:

$$\frac{\partial}{\partial \theta_0} C(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

for $j > 0$:

$$\frac{\partial}{\partial \theta_j} C(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j$$

λ

λ is called the *regularization parameter* - it is a parameter we should tweak in order to reward (or penalize) certain features.

λ

λ is called the *regularization parameter* - it is a parameter we should tweak in order to reward (or penalize) certain features.

Note that we do not regularize θ_0 since it pertains to the "ones" feature and it is not affected by the weight of the features in consideration.

Content Based Systems

In content based systems, the *features* which describe the content of something being recommended to users are known.

Content Based Systems

In content based systems, the *features* which describe the content of something being recommended to users are known.

However, the user weights for these features are not known.

Content Based Systems

*I*n content based systems, the *features* which describe the content of something being recommended to users are known.

However, the user weights for these features are not known.

These features are information like amount of action or romance in a movie, or the different categories in which a particular product may fall.

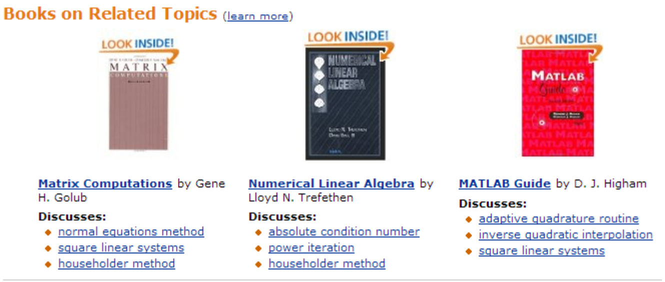
Example

For example, if you search for “scientific computing” on amazon.com, you will get a list of books that fall in that category.

Example

For example, if you search for "scientific computing" on amazon.com, you will get a list of books that fall in that category.

Clicking one of the results and scrolling down a bit, we find yet another example of content based recommendations:



A more simple example...

In order to introduce the topic and formulate the problem, let us work on a simplified example:

Book	Alice	Bob	Charlie	Dave
Advanced Math in Greek	5	0	5	0
The Book of Proofs	?	2	5	0
Down-to-Earth Math	0	5	1	4
Everyday Math	0	?	?	?
Intermediate Crazy Math	5	0	?	?

Say we have a matrix as above, where rows correspond to book tiles and columns correspond to readers (i.e. users of this system).

Say we have a matrix as above, where rows correspond to book tiles and columns correspond to readers (i.e. users of this system).

An entry $a_{i,j}$ in this matrix is a number ranging from 0 through 5, representing the rating user j gave to book i.

Say we have a matrix as above, where rows correspond to book tiles and columns correspond to readers (i.e. users of this system).

An entry $a_{i,j}$ in this matrix is a number ranging from 0 through 5, representing the rating user j gave to book i.

If we do not have a rating from a particular user on a particular book, then the entry is represented by a "?".

Definitions

- n_b is the number of books;

Definitions

- n_b is the number of books;
- n_u is the number of users;

Definitions

- n_b is the number of books;
- n_u is the number of users;
- $r(i,j) = 1$ if user j has rated book i (otherwise 0);

Definitions

- n_b is the number of books;
- n_u is the number of users;
- $r(i,j) = 1$ if user j has rated book i (otherwise 0);
- $y^{(i,j)}$ is the rating by user j of book i if $r(i,j) = 1$;

Definitions

- n_b is the number of books;
- n_u is the number of users;
- $r(i,j) = 1$ if user j has rated book i (otherwise 0);
- $y^{(i,j)}$ is the rating by user j of book i if $r(i,j) = 1$;
- $\theta^{(j)}$ is the parameter vector for user j;

Definitions

- n_b is the number of books;
- n_u is the number of users;
- $r(i,j) = 1$ if user j has rated book i (otherwise 0);
- $y^{(i,j)}$ is the rating by user j of book i if $r(i,j) = 1$;
- $\theta^{(j)}$ is the parameter vector for user j;
- $x^{(i)}$ is the feature vector for book i; and

Definitions

- n_b is the number of books;
- n_u is the number of users;
- $r(i, j) = 1$ if user j has rated book i (otherwise 0);
- $y^{(i,j)}$ is the rating by user j of book i if $r(i, j) = 1$;
- $\theta^{(j)}$ is the parameter vector for user j;
- $x^{(i)}$ is the feature vector for book i; and
- $(\theta^{(j)})^T(x^{(i)})$ is thus, the predicted rating for user j, for book i.

Back to our Example

Furthermore, in this case we know the feature vectors being considered: x_1 = amount of abstract mathematics and x_2 = amount of practical mathematics.

Back to our Example

Furthermore, in this case we know the feature vectors being considered: $x_1 =$ amount of abstract mathematics and $x_2 =$ amount of practical mathematics.

That is, someone read all of these books and "graded" them on these features, say on a scale from 0.0 to 1.0.

Problem Formulation

In order for us to learn $\theta^{(j)}$ (the parameter vector for user j - or the weights user j gives to the features considered) we must solve the following optimization problem:

Problem Formulation

In order for us to learn $\theta^{(j)}$ (the parameter vector for user j - or the weights user j gives to the features considered) we must solve the following optimization problem:

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

Note ...

The expression inside the parenthesis looks just like the regularized cost function we mentioned earlier, with the exception of the division by m.

Note ...

The expression inside the parenthesis looks just like the regularized cost function we mentioned earlier, with the exception of the division by m.

Recall from our first discussion of linear regression that the $\frac{1}{m}$ constant multiplying our cost function was there for mathematical convenience and that it did not affect the minimum.

Note ...

The expression inside the parenthesis looks just like the regularized cost function we mentioned earlier, with the exception of the division by m.

Recall from our first discussion of linear regression that the $\frac{1}{m}$ constant multiplying our cost function was there for mathematical convenience and that it did not affect the minimum.

Here, the analogous constant factor would be $\frac{1}{m^{(j)}}$ where $m^{(j)}$ is the number of books rated by user j.

Note ...

The expression inside the parenthesis looks just like the regularized cost function we mentioned earlier, with the exception of the division by m.

Recall from our first discussion of linear regression that the $\frac{1}{m}$ constant multiplying our cost function was there for mathematical convenience and that it did not affect the minimum.

Here, the analogous constant factor would be $\frac{1}{m^{(j)}}$ where $m^{(j)}$ is the number of books rated by user j.

Since we will have to optimize for these $\theta^{(j)}$'s across all users j, then the constant factor $\frac{1}{m^{(j)}}$ is no longer convenient.

Problem Formulation

Thus to learn the parameters for all users j, that is $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)}$, we need to solve:

Problem Formulation

Thus to learn the parameters for all users j, that is $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)}$, we need to solve:

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

Cost Function's Derivative

If we are using a gradient based optimization algorithm to solve for the above problem, then we provide the derivative of the cost function:

Cost Function's Derivative

If we are using a gradient based optimization algorithm to solve for the above problem, then we provide the derivative of the cost function:

For $k = 0$ (since j has a different meaning in our current formulation):

$$\frac{\partial}{\partial \theta_0^{(j)}} C(\theta^{(j)}) = \sum_{i:r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)}) x_0^{(i)}$$

Cost Function's Derivative

If we are using a gradient based optimization algorithm to solve for the above problem, then we provide the derivative of the cost function:

For $k = 0$ (since j has a different meaning in our current formulation):

$$\frac{\partial}{\partial \theta_0^{(j)}} C(\theta^{(j)}) = \sum_{i:r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)}) x_0^{(i)}$$

and for $k > 0$:

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

Collaborative Filtering Systems

In collaborative filtering systems, the features which describe the content of something being recommended to users are not known. However, we do know the *weights* users give to particular features.

Collaborative Filtering Systems

In collaborative filtering systems, the features which describe the content of something being recommended to users are not known. However, we do know the *weights* users give to particular features.

For example, we know how Sally weighs her genre preferences when it comes to movies - we know that she really likes action movies, then "contains action" is a likely feature of the movies she watches.

Example

On the same amazon.com page I was before, we find an example of collaborative filtering:

Example

On the same amazon.com page I was before, we find an example of collaborative filtering:

A more simple example...

Once again, let us work on a simplified example:

Book	Alice	Bob	Charlie	Dave
Advanced Math in Greek	5	0	5	0
The Book of Proofs	?	2	5	0
Down-to-Earth Math	0	5	1	4
Everyday Math	0	?	?	?
Intermediate Crazy Math	5	0	?	?

This is the same matrix as before, however now we do not have any tags on these books, i.e. there is no "grade" for the amount of abstract mathematics versus practical mathematics.

This is the same matrix as before, however now we do not have any tags on these books, i.e. there is no “grade” for the amount of abstract mathematics versus practical mathematics.

However, we do have the following weight vectors for each user, say:

This is the same matrix as before, however now we do not have any tags on these books, i.e. there is no “grade” for the amount of abstract mathematics versus practical mathematics.

However, we do have the following weight vectors for each user, say:

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \ \theta^{(2)} = \begin{bmatrix} 0 \\ 1 \\ 5 \end{bmatrix}, \ \theta^{(3)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \ \theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$$

Problem Formulation

The definitions for the notation we will use here are the same as before.

Problem Formulation

The definitions for the notation we will use here are the same as before.

So given vectors $\theta^{(j)}$'s for all n_u users, we wish to learn the vector $x^{(i)}$ (the feature vector for book i).

Problem Formulation

The definitions for the notation we will use here are the same as before.

So given vectors $\theta^{(j)}$'s for all n_u users, we wish to learn the vector $x^{(i)}$ (the feature vector for book i).

Thus, we must solve the following optimization problem:

Problem Formulation

The definitions for the notation we will use here are the same as before.

So given vectors $\theta^{(j)}$'s for all n_u users, we wish to learn the vector $x^{(i)}$ (the feature vector for book i).

Thus, we must solve the following optimization problem:

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

Problem Formulation

In order to learn the features for all books i, that is $x^{(1)}, x^{(2)}, \ldots, x^{(n_b)}$, we need to solve:

Problem Formulation

In order to learn the features for all books i, that is $x^{(1)}, x^{(2)}, ..., x^{(n_b)}$, we need to solve:

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

Cost Function's Derivative

If we are using a gradient based optimization algorithm to solve for the above problem, then the derivative of the cost function here should be with respect to $x^{(i)}$,

Cost Function's Derivative

If we are using a gradient based optimization algorithm to solve for the above problem, then the derivative of the cost function here should be with respect to $x^{(i)}$,

however it should not look too different from before and working the differentiation is left to you as an exercise.

Going back and forth...

So once we solve the above problem and get a good approximation of $x^{(1)}, x^{(2)}, ..., x^{(n_b)}$, there is no reason why we can't use this new information to estimate $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)}$

Going back and forth...

So once we solve the above problem and get a good approximation of $x^{(1)}, x^{(2)}, ..., x^{(n_b)}$, there is no reason why we can't use this new information to estimate $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)}$

- as we might not have all of them, or there may be new users over time.

Going back and forth...

So once we solve the above problem and get a good approximation of $x^{(1)}, x^{(2)}, ..., x^{(n_b)}$, there is no reason why we can't use this new information to estimate $\theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)}$

- as we might not have all of them, or there may be new users over time.

But there may also be new books over time, and we can go back and forth in this manner estimating features and parameters.

Look carefully...

If you study the two minimization problems below, you will see that it is possible to combine them into one:

Look carefully...

If you study the two minimization problems below, you will see that it is possible to combine them into one:

In order to learn the features for all books i we need to solve:

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

Look carefully...

If you study the two minimization problems below, you will see that it is possible to combine them into one:

In order to learn the features for all books i we need to solve:

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

And in order to learn the parameters for all users j we need to solve:

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

Combining the Problems

... combined, becomes:

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

Collaborative Filtering Algorithm

1. Initialize $x^{(1)}, x^{(2)}, ..., x^{(n_b)}, \theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)}$ to small random numbers.

Collaborative Filtering Algorithm

1. Initialize $x^{(1)}, x^{(2)}, ..., x^{(n_b)}, \theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)}$ to small random numbers.

2. Minimize the cost function

$$C(x^{(1)}, x^{(2)}, ..., x^{(n_b)}, \theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)})$$

using gradient descent or another optimization algorithm.

Collaborative Filtering Algorithm

1. Initialize $x^{(1)}, x^{(2)}, ..., x^{(n_b)}, \theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)}$ to small random numbers.

2. Minimize the cost function

$$C(x^{(1)}, x^{(2)}, ..., x^{(n_b)}, \theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)})$$

   using gradient descent or another optimization algorithm.

3. For a user j with parameters $\theta^{(j)}$ and for a book i with features $x^{(i)}$, predict a rating of $(\theta^{(j)})^T(x^{(i)})$.

Collaborative Filtering Algorithm

1. Initialize $x^{(1)}, x^{(2)}, ..., x^{(n_b)}, \theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)}$ to small random numbers.

2. Minimize the cost function

$$C(x^{(1)}, x^{(2)}, ..., x^{(n_b)}, \theta^{(1)}, \theta^{(2)}, ..., \theta^{(n_u)})$$

   using gradient descent or another optimization algorithm.

3. For a user j with parameters $\theta^{(j)}$ and for a book i with features $x^{(i)}$, predict a rating of $(\theta^{(j)})^T(x^{(i)})$.

The algorithm just described here is also known as *low-rank matrix factorization*.

Top 10

We can use the results to find the top ten similar books to some book i as follows:

Top 10

We can use the results to find the top ten similar books to some book i as follows:

find the ten books l with the smallest distance from book i, that is $||x^{(i)} - x^{(l)}||$.