Recommender Systems II

(Algorithm from Netflix Prize)

Approach

- The approach described in the following was an important part of the solution of the winning team in the Netflix $1,000,000 competition.

- Concluded 21 September 2009

- Prize given to BellKor's Pragmatic Chaos team

Evaluating Rec. Systems

- Consider root-mean-square error (RMSE).

$$\sqrt{\frac{\sum_{u,i} e_{u,i}^2}{N}} = \sqrt{\frac{\sum_{u,i} \left(r_{u,i} - \hat{r}_{u,i}\right)^2}{N}}$$

- How to minimize it?

We will use gradient descent to minimize each term:

$$e_{u,i}^2 = \left(r_{u,i} - \hat{r}_{u,i}\right)^2$$

Mean and biases: Baseline

First what will the prediction be?

$$\hat{r}_{u,i} = \mu + b_u + b_i$$

b_u and b_i capture the deviations of user u and item i from the average μ.

E.g., suppose that we want a baseline predictor for the rating of movie **Titanic** by user **Joe**.

Now, say that the average rating over all movies, μ, is 3.7 stars.

Furthermore, **Titanic** is better than an average movie, so it tends to be rated 0.5 stars above the average, i.e. b_{Titanic}=0.5

On the other hand, **Joe** is a critical user, who tends to rate 0.3 stars lower than the average, i.e. b_{Joe}= -0.3.

Thus, the baseline predictor for **Titanic's** rating by **Joe** would be 3.9 stars by calculating 3.7−0.3+0.5.

Then use gradient descent to minimize $e_{u,i}^2 = \left(r_{u,i} - \hat{r}_{u,i}\right)^2$ thus finding b_u and b_i.

Mean and biases:Gradient

$$e_{u,i}^2 = \left(r_{u,i} - \hat{r}_{u,i}\right)^2 = \left(r_{u,i} - \mu - b_u - b_i\right)^2$$

We apply regularization to fight overfitting. So, **minimize**:

$$f\left(b_u, b_i\right) = \left(r_{u,i} - \mu - b_u - b_i\right)^2 + \lambda_1 b_u^2 + \lambda_2 b_i^2$$

Regularization: The last two terms are added to penalize big bu and bi values, so that bu and bi don't become too important for the particular dataset and thus have overfitting. It's penalty for being too good for one dataset and suffering for another.

$$\frac{\partial f}{\partial b_u} =$$

$$-2\left(r_{u,i} - \mu - b_u - b_i\right) + 2\lambda_1 b_u =$$

$$-2e_{u,i} + 2\lambda_1 b_u =$$

$$-2\left(e_{u,i} - \lambda_1 b_u\right)$$

$$\frac{\partial f}{\partial b_i} =$$

$$-2\left(r_{u,i} - \mu - b_u - b_i\right) + 2\lambda_2 b_i =$$

$$-2e_{u,i} + 2\lambda_2 b_i =$$

$$-2\left(e_{u,i} - \lambda_2 b_i\right)$$

Mean and biases: Gradient Descent

Iterate over each $r_{u,i}$, and apply gradient descent update rules:

$$b_u \leftarrow b_u + \gamma\left(e_{u,i} - \lambda_1 b_u\right)$$
$$b_i \leftarrow b_i + \gamma\left(e_{u,i} - \lambda_2 b_i\right)$$

The hyperparameters could be for example:

$$\lambda_1 = 0.02$$
$$\lambda_2 = 0.02$$
$$\gamma = 0.005$$

…grid-search can be used to find better values for hyperparameters.

30 iterations are often good enough.

Latent Factors

$$\hat{r}_{u,i} = \mu + b_u + b_i + \mathbf{p}_u \cdot \mathbf{q}_i$$

$\mathbf{p}_{Gus} = (2, -2)$
$\mathbf{q}_{DumbAndDumber} = (2.2, -1.8)$

$\mathbf{p}_{Gus} \cdot \mathbf{q}_{DumbAndDumber} = 4.4+3.6 = 8$

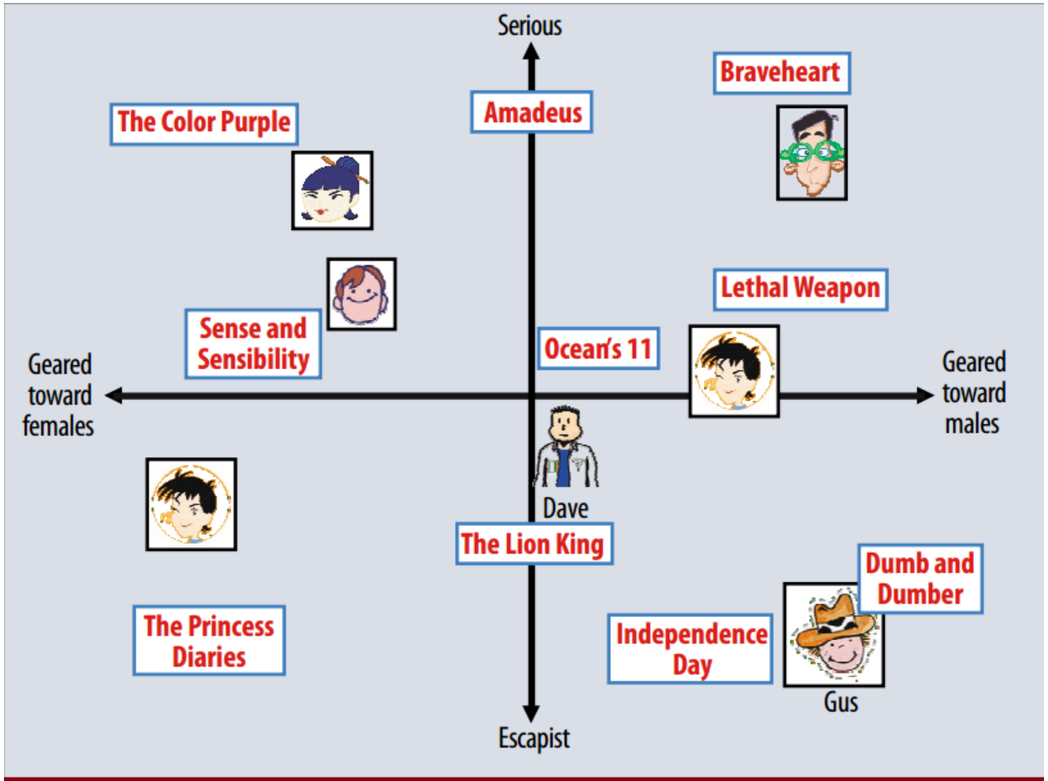

Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Factor is a "characteristic" of users and movies. Both users and movies are mapped to a common "factor" space. The closer a user and a movie are in factor space, the greater their dot-product.

While the figure shows some names for the factors, e.g. "serious/escapist", "geared toward males/geared toward females", in general these are latent (hidden) factors for which it is not possible to come up with a name.

We set the number of factors by setting the dimension of p and q vectors. Typical dimension is 100.

Latent Factors: Gradient

$$e_{u,i}^2 = \left(r_{u,i} - \hat{r}_{u,i}\right)^2 = \left(r_{u,i} - \mu - b_u - b_i - \mathbf{p}_u \cdot \mathbf{q}_i\right)^2$$

We apply regularization to fight overfitting. So, **minimize**:

$$f\left(b_u, b_i, \mathbf{p}_u, \mathbf{q}_i\right) = \left(r_{u,i} - \mu - b_u - b_i - \mathbf{p}_u \cdot \mathbf{q}_i\right)^2 + \lambda_1 b_u^2 + \lambda_2 b_i^2 + \lambda_3 \left\|\mathbf{p}_u\right\|^2 + \lambda_4 \left\|\mathbf{q}_i\right\|^2$$

$$\frac{\partial f}{\partial b_u} = -2\left(e_{u,i} - \lambda_1 b_u\right)$$

$$\frac{\partial f}{\partial b_i} = -2\left(e_{u,i} - \lambda_2 b_i\right)$$

$$\frac{\partial f}{\partial \mathbf{p}_u} = -2\left(e_{u,i}\mathbf{q}_i - \lambda_3 \mathbf{p}_u\right)$$

$$\frac{\partial f}{\partial \mathbf{q}_i} = -2\left(e_{u,i}\mathbf{p}_u - \lambda_4 \mathbf{q}_i\right)$$

Latent Factors: Gradient Descent

Iterate over each $r_{u,i}$, and apply gradient descent update rules:

$$b_u \leftarrow b_u + \gamma\left(e_{u,i} - \lambda_1 b_u\right)$$
$$b_i \leftarrow b_i + \gamma\left(e_{u,i} - \lambda_2 b_i\right)$$
$$\mathbf{p}_u \leftarrow \mathbf{p}_u + \gamma\left(e_{u,i}\mathbf{q}_i - \lambda_3 \mathbf{p}_u\right)$$
$$\mathbf{q}_i \leftarrow \mathbf{q}_i + \gamma\left(e_{u,i}\mathbf{p}_u - \lambda_4 \mathbf{q}_i\right)$$

The hyperparameters could be for example:

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0.02$$
$$\gamma = 0.005$$

…grid-search can be used to find better values for hyperparameters.

30 iterations are often good enough.

Based on

- Yehuda Koren, Robert Bell. *Advances in Collaborative Filtering*. In Recommender Systems Handbook, Springer 2011, pp 145-186.