Networked Life

- **Q4:** How does Netflix recommend movies?

Key concepts

- Recommendation systems
- Machine learning, data analytics
  - **Collaborative filtering**: a method of making automatic predictions about the interests of a user by collecting preferences or taste information from many users
  - Use of **linear regressions**
  - **Sparse matric decomposition**
  - **Class project!**

Topics

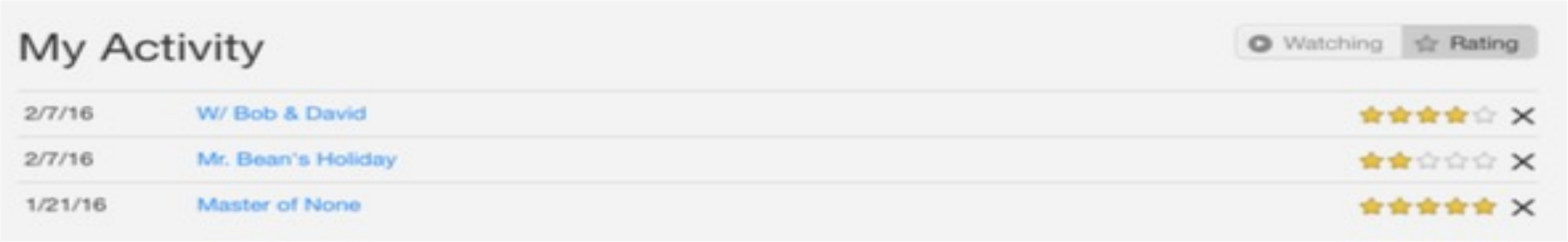
- Netflix case study and introduction to the problem
- The baseline predictor
- Linear regressions
- The Latent Factor model

The Netflix recommendation system

A word on Netflix

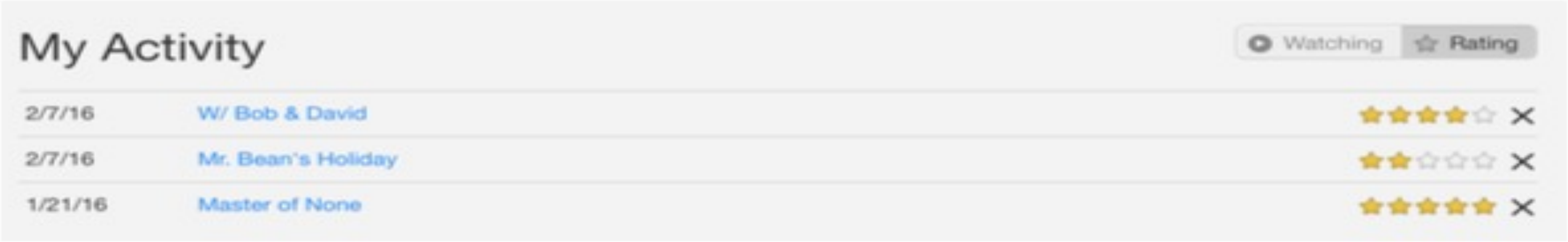
- Started as a DVD-by-mail company, huge growth due to their smart inventory management and large customer base
- Went into the streaming business in 2007. While DVD sales were declining globally, company continued to grow
- At some point, Netflix generated a fourth of the total number of transmitted bits over the Web
- Key to keeping the user engaged: a robust recommendation system

Recommend content



Based on these ratings, what should I watch next?

Recommend content



Based on these ratings, what should I watch next?

How does it work?

- Netflix has a large database of users and user ratings given to movies
- The company can leverage this data to predict a user's rating
- Similar to Amazon's "Customers who bought *x* also bought *y*"
- Find some structure in the existing rating/purchasing data

The Netflix prize

- Netflix had built its own in-house recommendation system CineMatch
- They launched in October 2006 an international competition with a $1 million prize
- The research team that would increase the accuracy of the recommendation system by >10% gets the prize
- **Question**: How to measure the precision of our recommendation system? What does it mean to predict 10% better?

The user-movie ratings matrix

- We have a database of ratings given by users on movies

		Movies							
		1	2	3	4	5	6	7	8
	1		5		2	4			
	2	4		3	1			3	
Users	3		5	4		5		4	
	4						1	1	2
	5	3		?		?	3		
	6		?	2		4		?	

The user-movie ratings matrix

- We have a database of ratings given by users on movies

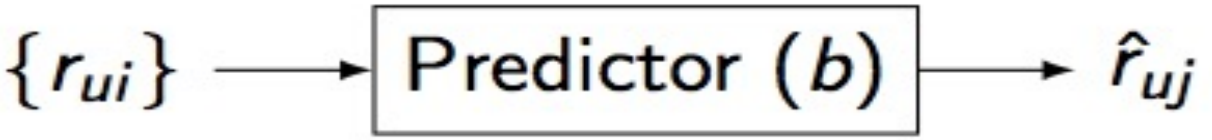
Movies

Users	1	2	3	4	5	6	7	8
1	?	5	?	2	4	?	?	?
2	4		3	1			3	
3		5	4		5		4	
4						1	1	2
5	3		?		?	3		
6		?	2		4		?	

- How to predict the “?”? How to measure the performance of our prediction?

The predictor

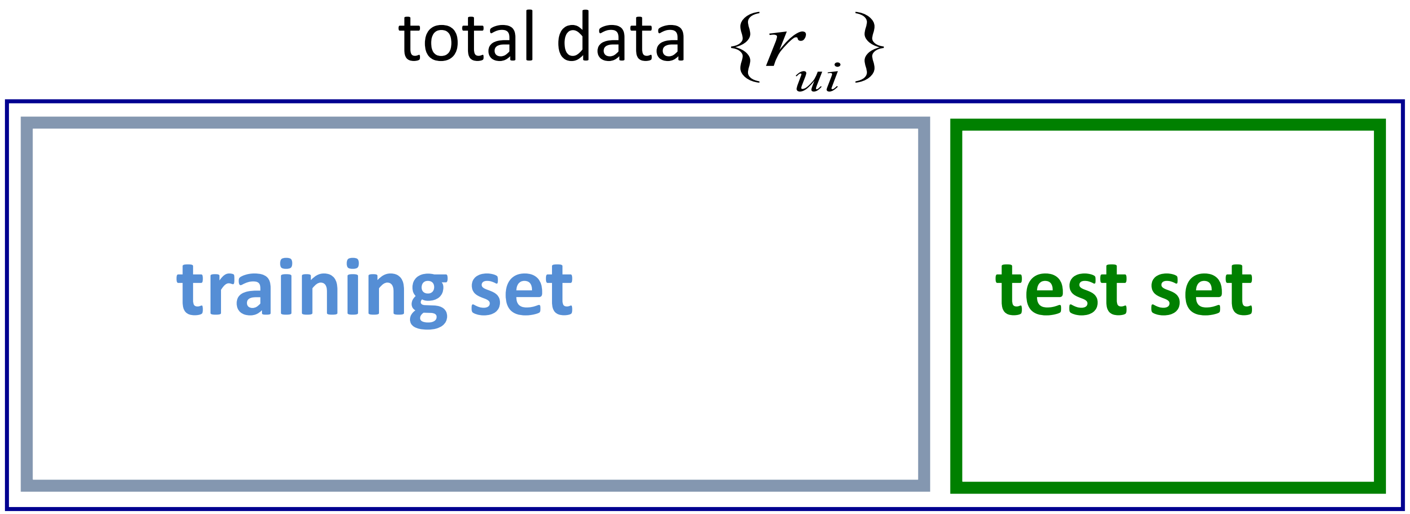
- We have a database of ratings $\{r_{ui}\}$:rating of movie i by user u
- We want to predict what rating $\{\hat{r}_{ui}\}$ user u will give to movie j
- We build a predictor: simple model based on a few parameters b, tuned with $\{r_{ui}\}$, able to return $\{\hat{r}_{ui}\}$



$\forall u, j$

Training and testing

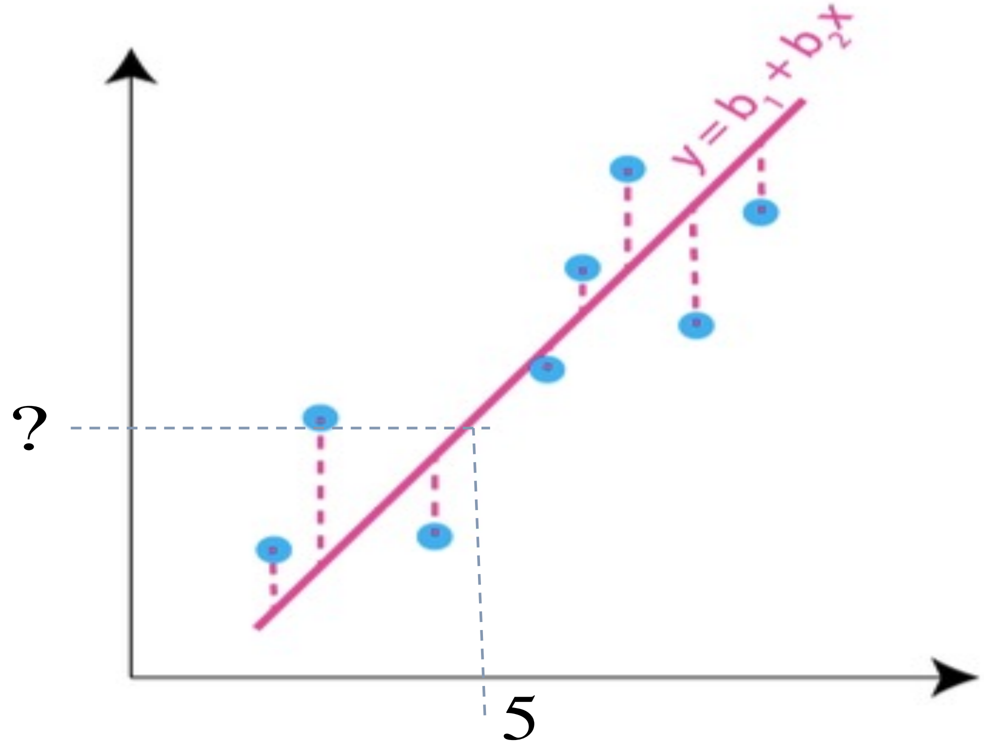
- We divide our database $\{r_{ui}\}$ in two: a **training set** and a **test set**



- We then "**train**" our predictor on the **training set**: find the optimal parameters (more on this later).
- Finally, "**predict**" the ratings of the **test set**. We can then compare the predicted ratings $\{r_{ui}\}$ with the actual ratings $\{\hat{r}_{ui}\}$

Linear regression approach

- Given a set of $\{(x_i, y_i)\}$ points, use a line to predict (5, ?)
- Fitting a line in high dimensions is equivalent to the least squares problem



Fitting the line $y = b_1 + b_2 x$:

- We want the sum of the **lengths squared** of **the dashed lines** to be **minimized**
- i.e., **minimize the RMSE**

The Root Mean Squared Error (RMSE)

- We need a metric to tell us how far our predictions are from the actual ratings
- The Root Mean Squared Error (RMSE) does the job well:

$$RMSE(\{r_{ui}\}\{\hat{r}_{ui}\}) = \sqrt{\frac{1}{C}\sum_{(u,i):r_{ui}\neq 0}(r_{ui}-\hat{r}_{ui})^2}$$

- C is the size of our test set. We sum over the C pairs of user-ratings that we have ratings

Example

- We have the following dataset:

$$
\begin{array}{cc}
 & \begin{array}{cccc} \text{The Godfather} & \text{Ip Man} & \text{Mrs. Doubtfire} & \text{Aliens} \end{array} \\
\begin{array}{c} \textit{Alice} \\ \textit{Bob} \\ \textit{Caroline} \\ \textit{David} \end{array} &
\begin{pmatrix}
5 & 2 & - & 3 \\
3 & 5 & 1 & - \\
5 & - & 4 & 2 \\
- & 3 & 2 & 5
\end{pmatrix}
\end{array}
$$

$$\{r_{ui}\} \longrightarrow \boxed{\text{Predictor } (b)} \longrightarrow \hat{r}_{uj}$$

Divide it between training and test:

$$\begin{array}{c|cccc} & \text{The Godfather} & \text{Ip Man} & \text{Mrs. Doubtfire} & \text{Aliens} \\ \hline \begin{array}{c} \textit{Alice} \\ \textit{Bob} \\ \textit{Caroline} \\ \textit{David} \end{array} & \multicolumn{4}{c}{\begin{pmatrix} 5 & 2\ ? & - & 3 \\ 3\ ? & 5 & 1 & - \\ 5 & - & 4 & 2\ ? \\ - & 3 & 2\ ? & 5 \end{pmatrix}} \end{array}$$

$$\{r_{ui}\} \longrightarrow \boxed{\text{Predictor } (b)} \longrightarrow \hat{r}_{uj}$$

Divide it between training and test:

$$
\begin{array}{r c}
 & \begin{array}{cccc} \text{The Godfather} & \text{Ip Man} & \text{Mrs. Doubtfire} & \text{Aliens} \end{array} \\
\begin{array}{r} \textit{Alice} \\ \textit{Bob} \\ \textit{Caroline} \\ \textit{David} \end{array} &
\left(\begin{array}{cccc}
5 & 2 \quad 1.3 & - & 3 \\
3 \quad 3.8 & 5 & 1 & - \\
5 & - & 4 & 2 \quad 2.1 \\
- & 3 & 2 \quad 2.5 & 5
\end{array} \right)
\end{array}
$$

We train our predictor, which gives us

$$\hat{r}_{\text{Alice, Ip Man}} = 1.3, \; \hat{r}_{\text{Bob, The Godfather}} = 3.8$$

$$\hat{r}_{\text{Caroline, Aliens}} = 2.1, \; \hat{r}_{\text{David, Mrs. Doubtfire}} = 2.5$$

$$
\begin{array}{cc}
 & \begin{array}{cccc} \text{The Godfather} & \text{Ip Man} & \text{Mrs. Doubtfire} & \text{Aliens} \end{array} \\
\begin{array}{c} \textit{Alice} \\ \textit{Bob} \\ \textit{Caroline} \\ \textit{David} \end{array} & \begin{pmatrix} 5 & 2 \;\; 1.3 & - & 3 \\ 3 \;\; 3.8 & 5 & 1 & - \\ 5 & - & 4 & 2.1 \;\; 2 \\ - & 3 & 2 \;\; 2.5 & 5 \end{pmatrix}
\end{array}
$$

$$\hat{r}_{\text{Alice, Ip Man}} = 1.3, \ \hat{r}_{\text{Bob, The Godfather}} = 3.8$$

$$\hat{r}_{\text{Caroline, Aliens}} = 2.1, \ \hat{r}_{\text{David, Mrs. Doubtfire}} = 2.5$$

So our RMSE is

$$RMSE = \sqrt{\frac{(1.3-2)^2 + (3.8-3)^2 + (2.1-2)^2 + (2.5-2)^2}{4}} = 0.5895$$

10% better?

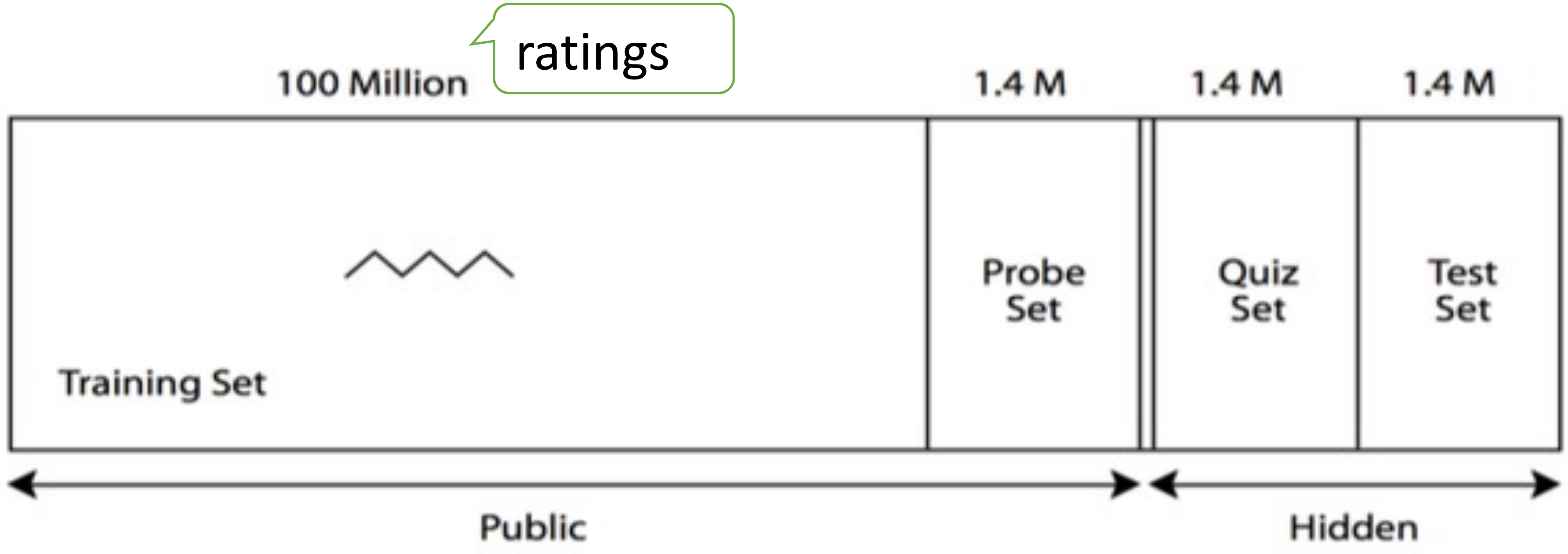
$$RMSE(\{r_{ui}\}\{\hat{r}_{ui}\}) = \sqrt{\frac{1}{C}\sum_{(u,i):r_{ui}\neq 0}(r_{ui}-\hat{r}_{ui})^2}$$

- Suppose our RMSE is 0.9514.
- One that is 10% better is at 0.8563

How was the Netflix prize organized?

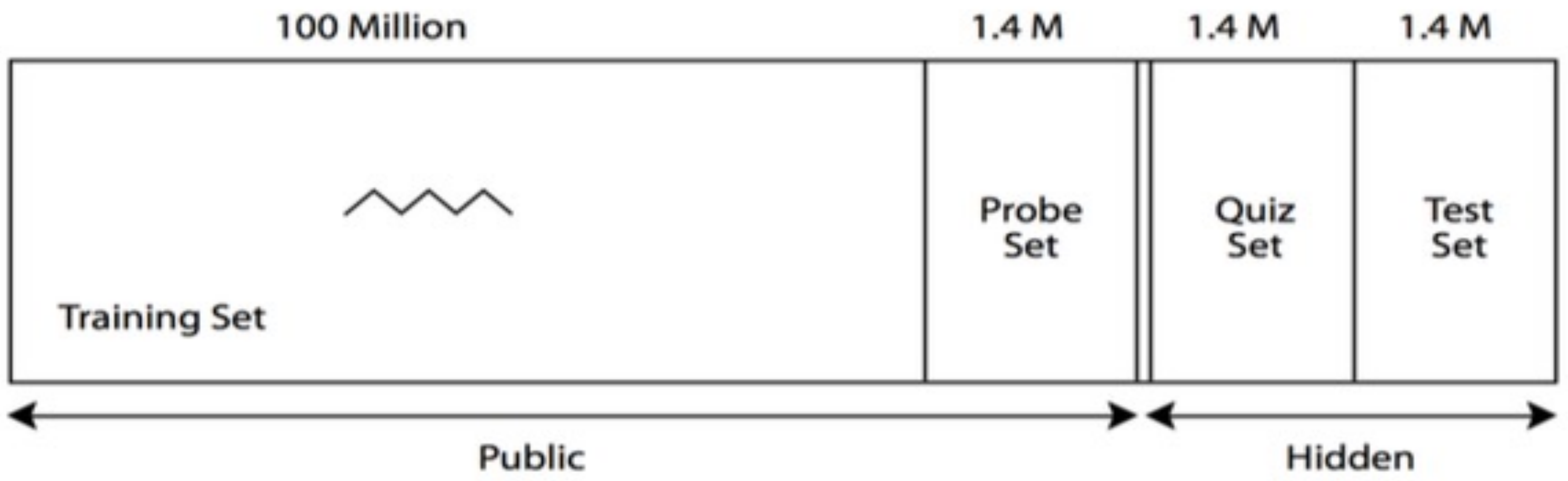
The Netflix datasets

- 480,189 users gave to 17,770 movies
- Netflix divided a large dataset of ratings in four:



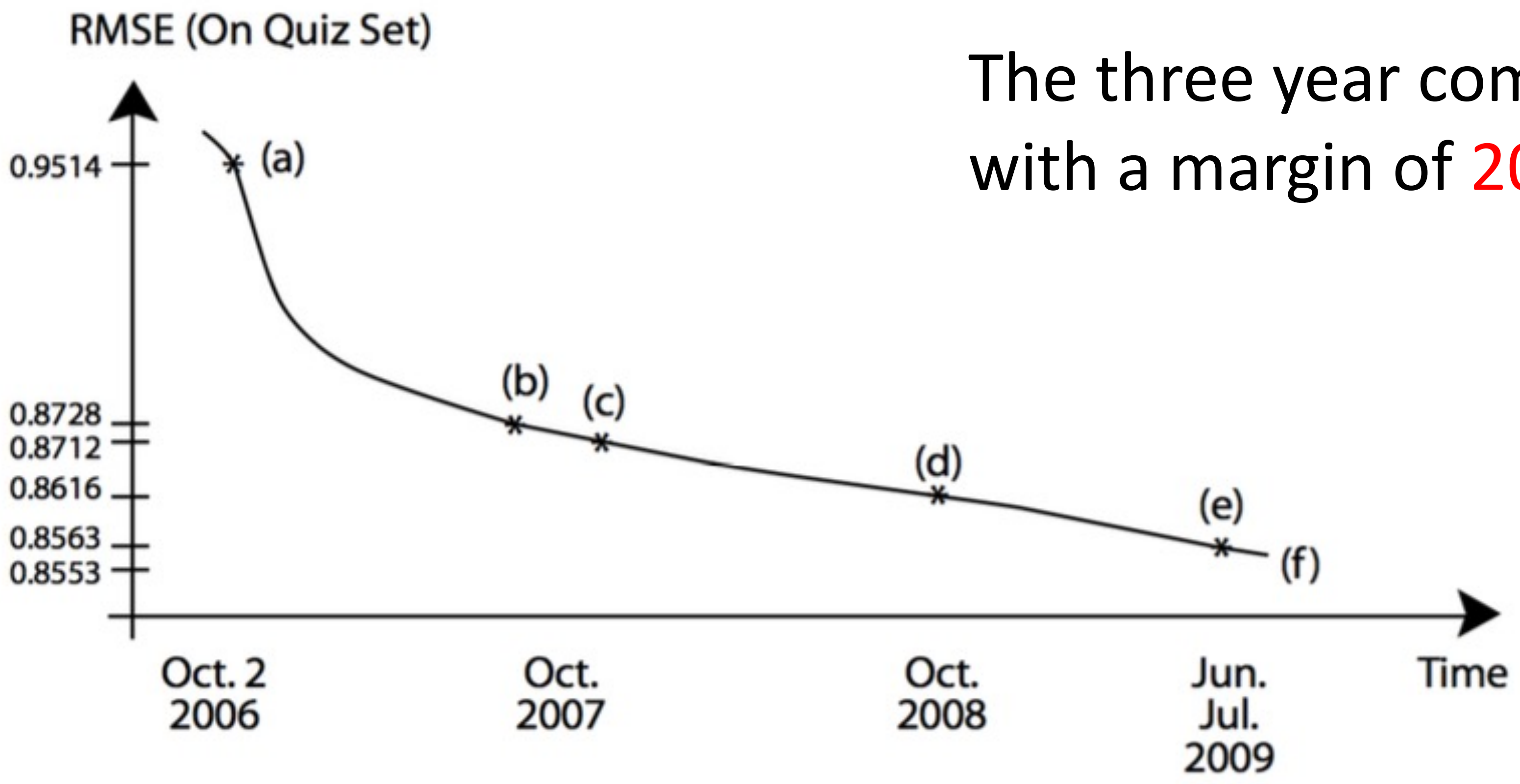
The Netflix datasets

- Netflix divided a large dataset of ratings in four:



- Predictors are trained on the **Training set**
- The teams can use the **Probe set** to test their predictors
- Once a day they can test on the **Quiz set**: current leaderboard is built from these results.
- The final results are obtained on the **Test set**

Timeline of the competition



The three year competition was won with a margin of 20 minutes!

Making predictions

How to predict?

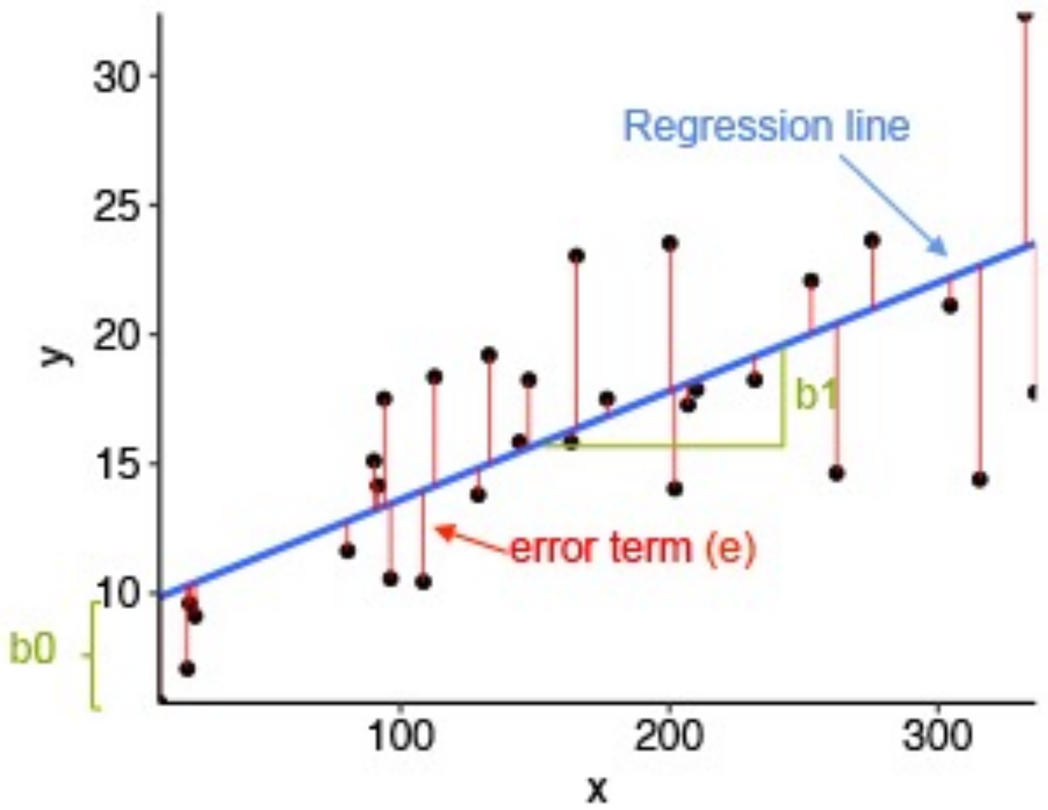
Two ways to build a "collaborative filtering" predictor:

- Construct a "baseline + neighborhood model": Alice is Bob's "neighbor" because both like same movies
  - Bob really liked Die Hard: we can expect that Alice will like it too
- Construct a "latent-factor model": assumes that there are few key factors for users and movies that make users like movies
  - Alice likes sci-fi movies, while The Great Gatsby usually receives low grades from sci-fi lovers: Alice will not like The Great Gatsby

Baseline predictor model

Baseline predictor model

- The baseline predictor model is a **regression model**: we build a simple linear model of the data



- We have a set of data points
- We are trying to fit the best line to have a **reduced model** of these data points
- This line is chosen by setting two parameters b_0 and b_1

Data

Training data

$$\mathcal{S}_n = \{\, (x^{(i)}, y^{(i)}) \mid i = 1, \dots, n \,\}$$

- Features/Inputs $x^{(i)} = \left(x_1^{(i)}, \dots, x_d^{(i)}\right)^\top \in \mathbb{R}^d$
- Target/Output/Response $y^{(i)} \in \mathbb{R}$

Matrix notation $X = \left[x^{(1)}, \dots, x^{(n)}\right]^\top,\ Y = \left[y^{(1)}, \dots, y^{(n)}\right]^\top$

Linear regression model

Each f is a regression function, predictor, or estimator

Model $\mathcal{H}$: set of linear functions with offset parametrized by b

$$f(x;b) = b_1x_1 + \cdots + b_dx_d + b_0 = x^Tb$$

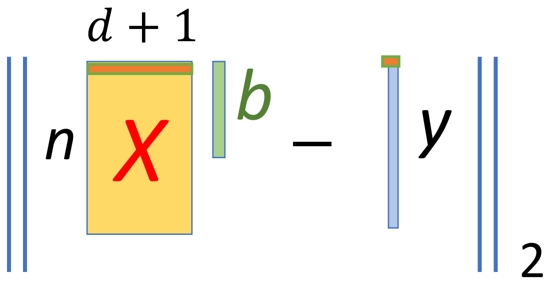
where we redefine $x = (x_1, \dots, x_d, 1)^T, b = (b_1, \dots, b_d, b_0)$

Matrix form: the vector of predictions = Xb

Training

Also known as fitting or learning

Minimize some loss function on our training data $\mathcal{S}_n$



$$\min_{b \in \mathbb{R}^{d+1}} \quad \mathcal{L}(b; \mathcal{S}_n)$$

Example. $\mathcal{L}(b; \mathcal{S}_n) = \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_n} \frac{1}{2} (f(x; b) - y)^2 = ||Xb - Y||_2^2$

Optimal parameter $\quad b^* = \arg\min_b ||Xb - Y||_2^2$

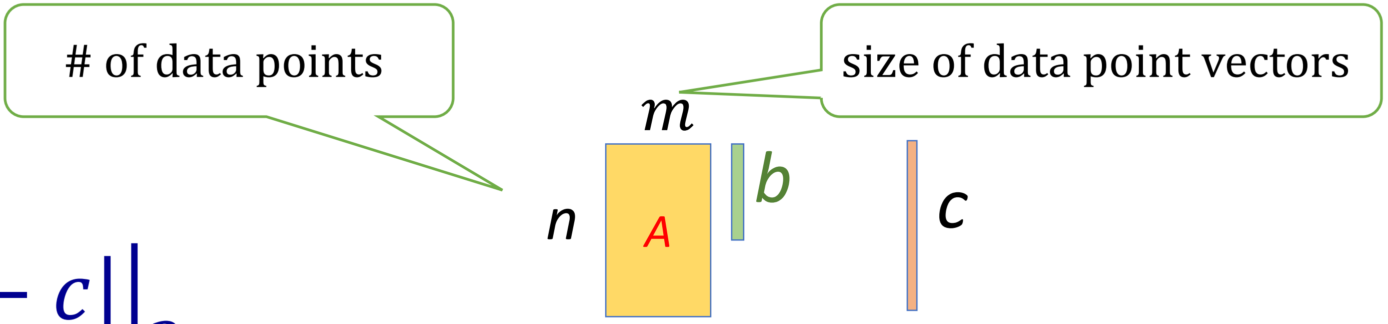
Exact solution $\quad b^* = (X^T X)^{-1} X^T Y$

Optimal estimator $\quad \hat{f}(x) = f(x; b^*)$

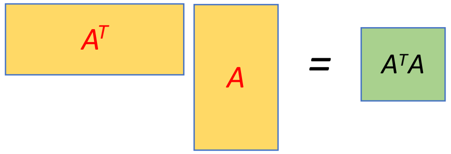
$L2$ norm:

$$||a||_2 = \sqrt{\sum_i a_i^2}$$

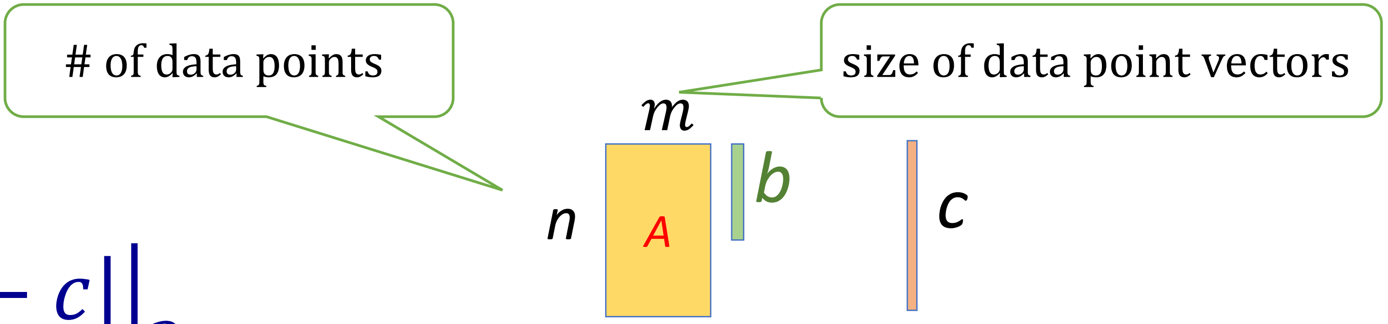
Optimization



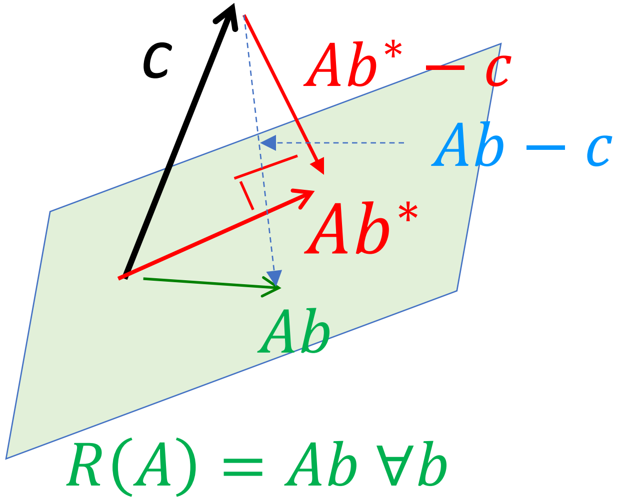
- We want to solve $\min_b ||Ab - c||_2$
- This is a convex problem (quadratic in b)
- Solve for b the set of equations
  $(A^T A)b = A^T c$
- If A has independent columns, then $A^T A$ is invertible and there is a unique solution $b^* = (A^T A)^{-1} A^T c$



Optimization



- We want to solve $\min_b ||Ab - c||_2$
- This is a convex problem (quadratic in b)
- Solve for b the set of equations $(A^T A)b = A^T c$
- If A has independent columns, then $A^T A$ is invertible and there is a unique solution $b^* = (A^T A)^{-1} A^T c$



$$\text{Find } b^* \text{ s.t. } Ab^* - c \perp R(A)$$
$$\Leftrightarrow Ab^* - c \in N(A^T)$$
$$\Leftrightarrow A^T(Ab^* - c) = 0$$

Example: Learning set: 4 points x in R^2

$$((x_{11}, x_{12}), y_1), ((x_{21}, x_{22}), y_2), ((x_{31}, x_{32}), y_3), ((x_{41}, x_{42}), y_4)$$

Need to construct predictor $y = b_1 x_1 + b_2 x_2 + b_0$

Find b that minimizes square error $e^T e = ||Ab - y||_2 = (Ab - y)^T (Ab - y)$

$$\text{Error } e(b) = Ab - y = \begin{pmatrix} x_{11} & x_{12} & 1 \\ x_{21} & x_{22} & 1 \\ x_{31} & x_{32} & 1 \\ x_{41} & x_{42} & 1 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_0 \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

Summary so far:

- Linear regression problem: how to find the vector b that minimizes the square error between the vectors Ab and c, i.e., solve the least square problem $\min_{b} ||Ab - c||_2$
- Theory: solve an LP: $(A^T A)b = A^T c$

- Can we write our recommendation problem as a linear regression?

Back to the baseline predictor model

- Our rating prediction problem is another type of linear regression
- We first build the model
- Then we reduce it to a linear regression!

Baseline predictor model

- If we have no idea which rating to predict for user u and movie i, we might as well return the overall mean rating
- We first compute the mean rating over our whole training set where C^{train} is the number of ratings in our training set

$$\bar{r} = \frac{\sum_{u,i} r_{ui}}{C^{train}}$$

Baseline predictor model

$$\bar{r} = \frac{\sum_{(u,i)} r_{ui}}{C^{train}}$$

$$\begin{array}{c} \\ Alice \\ Bob \\ Caroline \\ David \end{array} \begin{array}{cccc} \text{The Godfather} & \text{Ip Man} & \text{Mrs. Doubtfire} & \text{Aliens} \\ \end{array}$$

$$\begin{array}{l|c} & \begin{array}{cccc} \text{The Godfather} & \text{Ip Man} & \text{Mrs. Doubtfire} & \text{Aliens} \end{array} \\ \begin{array}{c} Alice \\ Bob \\ Caroline \\ David \end{array} & \begin{pmatrix} 5 & 2 & 3.3 & 3 \\ 3 & 5 & 1 & 3.3 \\ 5 & 3.3 & 4 & 2 \\ 3.3 & 3 & 2 & 5 \end{pmatrix} \end{array}$$

$$\bar{r} = (5+2+3+3+5+1+5+4+2+3+2+5)/12 = 3.3$$

Picking the right parameters: biases

- We then make the assumption that the rating given by a user u for a movie i is the **mean rating** plus **adjustments**, or **biases**, so we predict

$$\hat{r}_{ui} = \bar{r} + b_u + b_i$$

where b_u is the user's bias, b_i is the movie's bias

Picking the right parameters: biases

- We use $\hat{r}_{ui} = \bar{r} + b_u + b_i$
- In the example below, we take the biases as given:

$$
\begin{array}{c|cccc}
 & b_A = 0.7 & b_B = 0.4 & b_C = -0.6 & b_D = -0.2 \\
\hline
b_1 = 0.2 & 5 & 2 & 2.9 & 3 \\
b_2 = -0.4 & 3 & 5 & 1 & 2.7 \\
b_3 = 0.3 & 5 & 4 & 4 & 2 \\
b_4 = 0.1 & 4.1 & 3 & 2 & 5
\end{array}
$$

$$\hat{r}_{1C} = 3.3 + 0.2 - 0.6 = 2.9, \;\; \hat{r}_{2D} = 3.3 - 0.2 - 0.4 = 2.7$$

$$\hat{r}_{3B} = 3.3 + 0.3 + 0.4 = 4, \hat{r}_{4A} = 3.3 + 0.1 + 0.7 = 4.1$$

Tuning the model: the right biases

- We use $\hat{r}_{ui} = \bar{r} + b_u + b_i$
- We now want to find the optimal value for our $\{b_u\}$ and $\{b_i\}$, i.e., the one that fits best our **training set**
- Recall:

$$RMSE(\{r_{ui}\}\{\hat{r}_{ui}\}) = \sqrt{\frac{1}{C}\sum_{(u,i)\in\Omega}(\underbrace{r_{ui} - \hat{r}_{ui}}_{})^2}$$

$$\Omega = \{(u,i): r_{ui} \neq 0\}$$

$$error_{ui} = b_u + b_i - (r_{ui} - \bar{r})$$

Use b_u, b_i to predict $r_{ui} - \bar{r}$

The minimization problem

- To find the optimal $\{b_u\}$ and $\{bi\}$, we solve

$$\min_{\{b_u, b_i\}} \sqrt{\frac{1}{C} \sum_{(u,i)\in\Omega} (b_u + b_i - (r_{ui} - \bar{r})^2}$$

$$b_U = \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{pmatrix}$$

$$b_M = \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_m \end{pmatrix}$$

- Same as solving the problem:

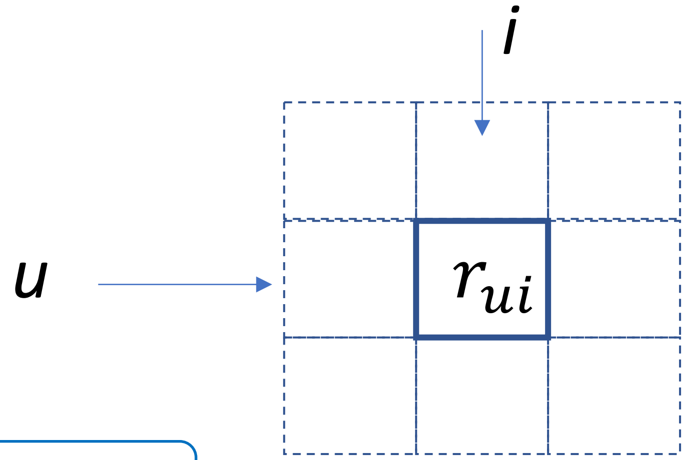
$$\min_{\{b_u, b_i\}} \sum_{(u,i)\in\Omega} (b_u + b_i - (r_{ui} - \bar{r})^2$$

- Want to write it as a least squares problem given some matrices A, c

$$\min_{b=(b_u, b_i)} ||Ab - c||^2$$

$$\sum_{(u,i)\in\Omega} \left(b_u + b_i - (r_{ui} - \bar{r})^2\right) = \|Ab - c\|_2^2$$

$$\begin{array}{c|cccc} & b_A & b_B & b_C & b_D \\ \hline b_1 & 5 & - & - & 3 \\ b_2 & - & 5 & 1 & - \\ b_3 & 5 & - & 4 & - \\ b_4 & - & 3 & - & 5 \end{array}$$



Cell with rating

# of ratings

$$\begin{array}{cccccccc} b_1 & b_2 & b_3 & b_4 & b_A & b_B & b_C & b_D \end{array}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_A \\ b_B \\ b_C \\ b_D \end{pmatrix} - \begin{pmatrix} 5-3.3 \\ 3-3.3 \\ 5-3.3 \\ 1-3.3 \\ 5-3.3 \\ 4-3.3 \\ 3-3.3 \\ 5-3.3 \end{pmatrix} = Ab - c$$

Recap

How does the baseline predictor model work?

- Identify the training set: database of ratings we have already collected
- Compute the mean rating $\hat{r}$ and construct our matrices *A*, *c*
- Compute biases $b^* = (A^T A)^{-1} A^T c$ which minimize the training set's RMSE.
- Compute the ratings for the test set using. $\hat{r}_{ui} = \bar{r} + b_u^* + b_i^*$

Further improvements

Refinements

- This approach was used to better the RMSE in the Netflix prize, with several other tricks:
- Include time-dependent biases: movies may be trendy at some point (high ratings) and later not so
- At the same time, users tend to rate similarly over short periods, affected by their moods, and their changing tastes
- Add a **regularization** term to the objective function to avoid **overfitting**: *our model fits "too well" the training set*, so gives bad predictions to new data

Regularization

- Sometimes, our model fits "too well" our training set: given new data to test, its predictions are all wrong (does not generalize).
- In particular, it happens when we are free to choose any b.
- Idea: penalize the large values of our parameters. New optimization program:

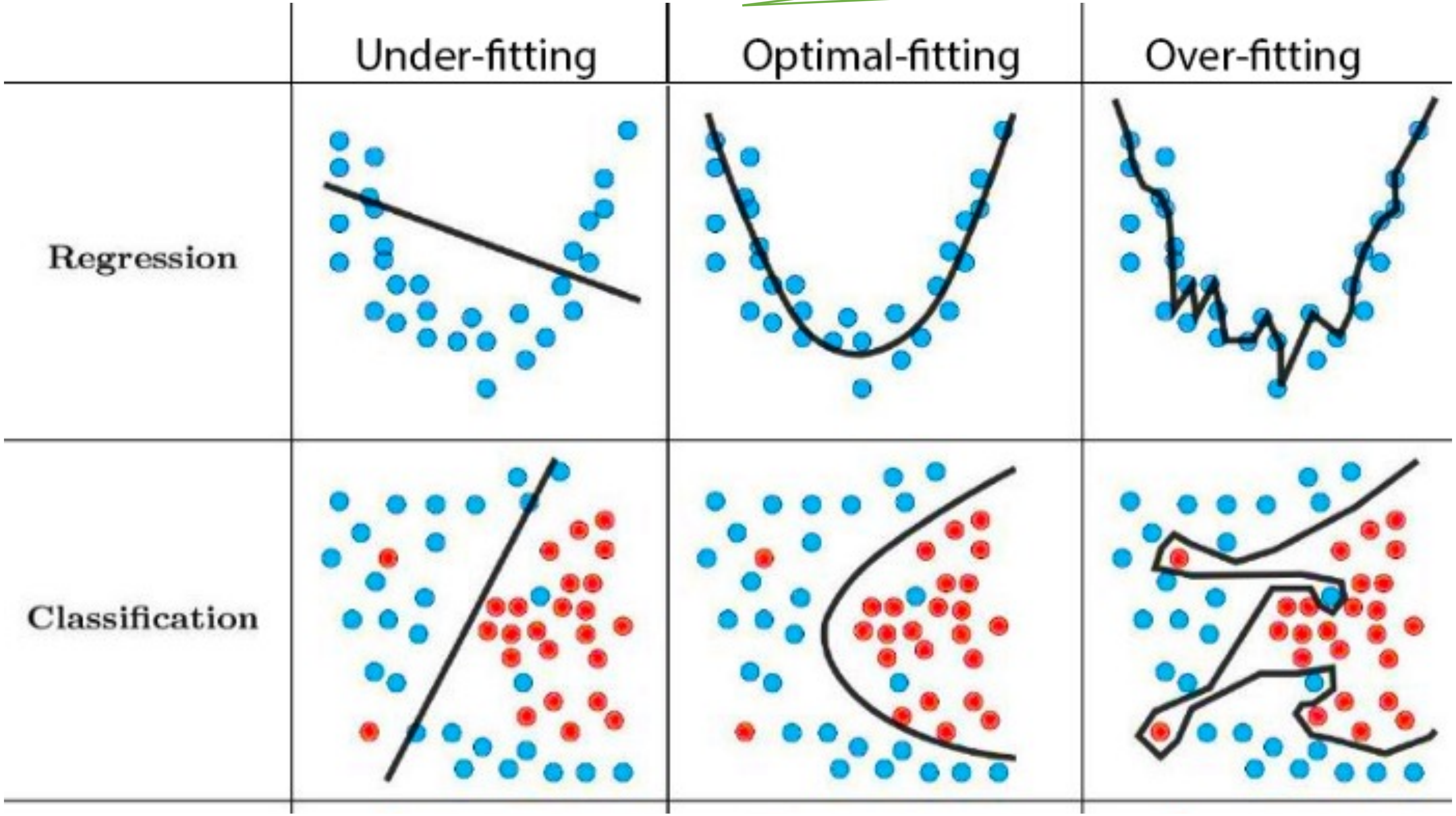
$$\min_{\{b_u, b_i\}} ||Ab - c||_2^2 + \lambda ||b||_2^2$$

where $||b||_2^2 = \sum_u b_u^2 + \sum_i b_i^2$.

defines a set of models;
chose the one that fits best
the validation (test) data

Bias vs variance

model complexity: degree of polynomial used for fitting the data



high bias, high variance

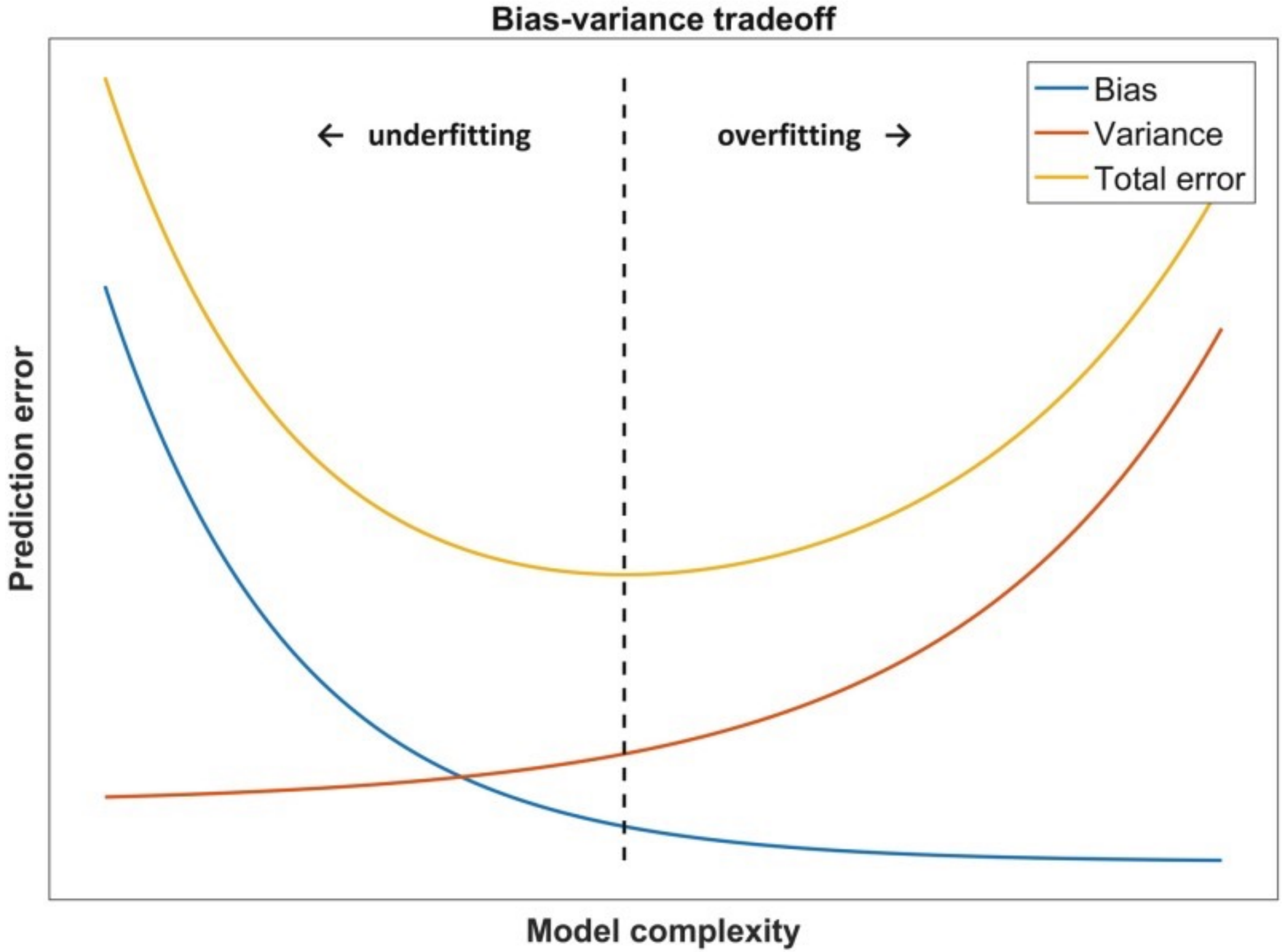
low bias, low variance

low bias, high variance

Bias: average model vs actual

variance: average model vs instance

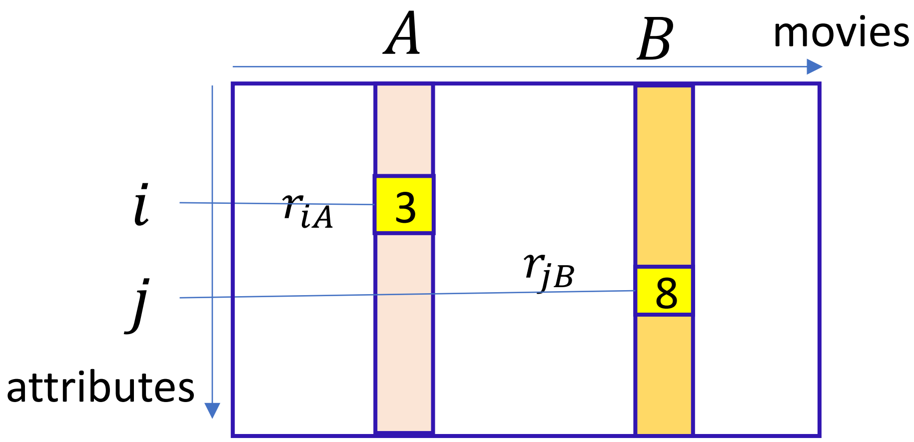
Underfitting - overfitting



In our case: model complexity is $\approx 1/\lambda$

A different approach: Neighborhood model

- **Neigborhood**: we want to be able to build a graph of "connected" movies based on some general $N \times M$ **attribute** matrix R'

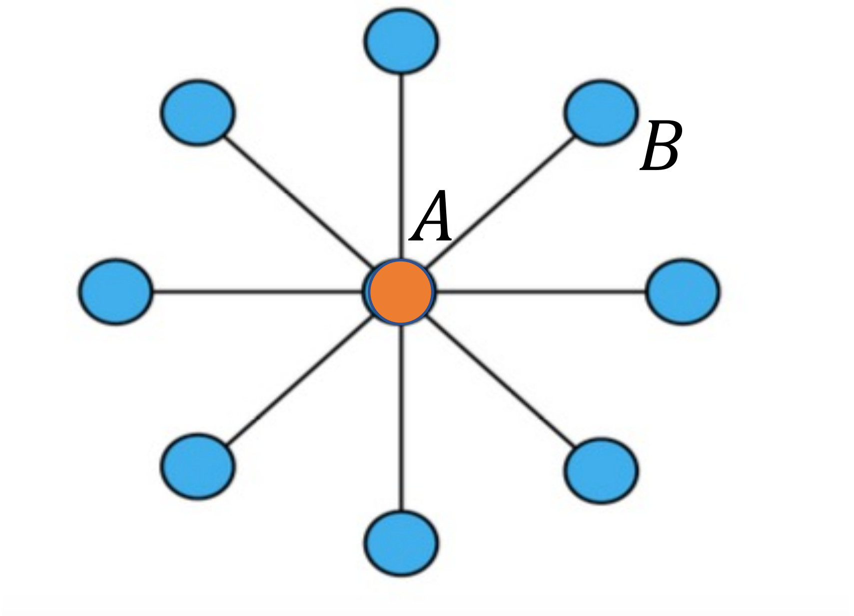


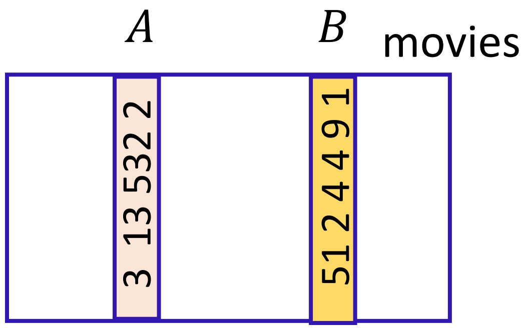
connected = "similar"

r_{iA} = value of ith attribute of movie A
e.g., could be the rating by user i
(or something else in general)
R may not be fully defined

A different approach: neighborhood model

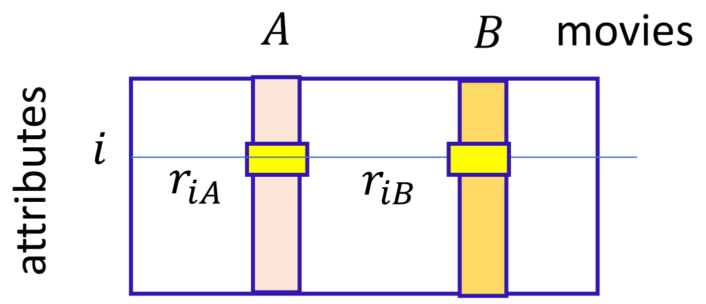
- **Neigborhood**: we want to be able to build a graph of "connected" movies based on some general $N \times M$ **attribute** matrix R'





Add a "**connection**" between movies A, B: there is significant **correlation** in the attributes of the two movies

Correlations



Given R', two movies A, B are:

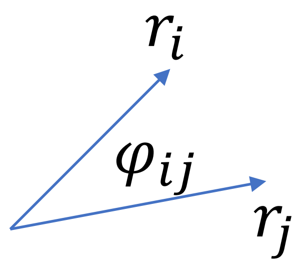
- **Positively** *correlated*: $\forall i$, a high (low) value for attribute r_{iA} mostly implies a high (low) value for r_{iB}
- **Negatively** *correlated*: $\forall i$, a high (low) value for attribute r_{iA} mostly implies a low (high) value for r_{iB}
- Uncorrelated: $\forall i, r_{iA}$ offers no information regarding r_{iB}
- *Correlation gives us information we can exploit*

Creating an 8-neighborhood based on R'

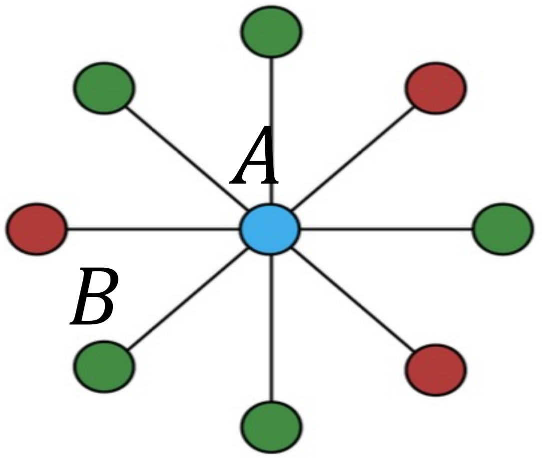
- Each column r_A of R' defines the values of attributes for movie A
- To find the similarity d_{AB} between two movies A and B, use

$$d_{AB} = \frac{r_A^T r_B}{||r_A||_2 ||r_B||_2} \in [-1, +1] = \cos \varphi_{AB}$$

Here r_A, r_B are projected over their components that are jointly defined



- Given movie A, we select the 8 movies B with the highest absolute similarity, i.e., largest $|d_{AB}|$



Guessing missing values in *R'*

- Value r_{iA} of the ith attribute for movie A is missing. How to make a reasonable guess?
- Take a weighted average of "suggestions" from movies in the neighborhood of A
- Let $S =$ set of movies C in the neighborhood of A s.t. $r_{iC} \neq nil$
- Then we use the estimator $\hat{r}_{iA} = \frac{1}{\sum_{B \in S} |d_{AB}|} \sum_{B \in S} d_{AB} r_{iB}$
- Take into account the values r_{iB} for the $same\ attribute$ of all reasonably similar movies B to A, multiplied by normalized weights that reflect the correlation

Applications

- We can use this method to infer missing ratings by using as R' the existing ratings matrix R we use for training
- More interesting: use it to improve the results $\hat{R}$ of the baseline predictor
- Define our attribute matrix R' to be equal to $\tilde{R} = R - \hat{R}$
- Think of $\tilde{R}$ as the value to add to baseline predictor to get the actual rating (i.e., $\tilde{R}$ is the error of the predictor)
- If I could predict $\tilde{R}$ for all missing u, i pairs, then I could improve my prediction by using $\hat{R} + \tilde{R}$ instead of $\hat{R}$

→ Use the neighborhood model to predict the missing values in $\tilde{R}$

Example

$$\begin{array}{c} \\ \tilde{R} = \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0.8 & -1.9 & - & -0.3 \\ -0.6 & 1.7 & 1 & - \\ 0.7 & - & 1 & -1.4 \\ \overset{?}{-} & \boxed{-0.8} & -0.8 & 1.8 \end{pmatrix} \end{array}$$

- Consider movies A, B
- Can we guess the error for predicting 4A by taking into account the error we made for 4B?
- Compute the similarity d_{AB} for movie A and B:

$$d_{AB} = \frac{0.8\times(-1.9) + (-0.6)\times 1.7}{\sqrt{((0.8)^2 + (-0.6)^2}\times\sqrt{((-1.9)^2 + (1.7)^2)}} \approx -1$$

- We only sum over users that gave ratings to both movies!
- We repeat this exercise for movies C, D: $d_{AC} \approx 0.1, d_{AD} \approx -0.8$

Final predictor

$$\tilde{R} = \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0.8 & -1.9 & - & -0.3 \\ -0.6 & 1.7 & 1 & - \\ 0.7 & - & 1 & -1.4 \\ -\,? & \boxed{-0.8} & -0.8 & \boxed{1.8} \end{pmatrix} \end{array}$$

$$d_{AB} \approx -1, d_{AD} \approx -0.8$$

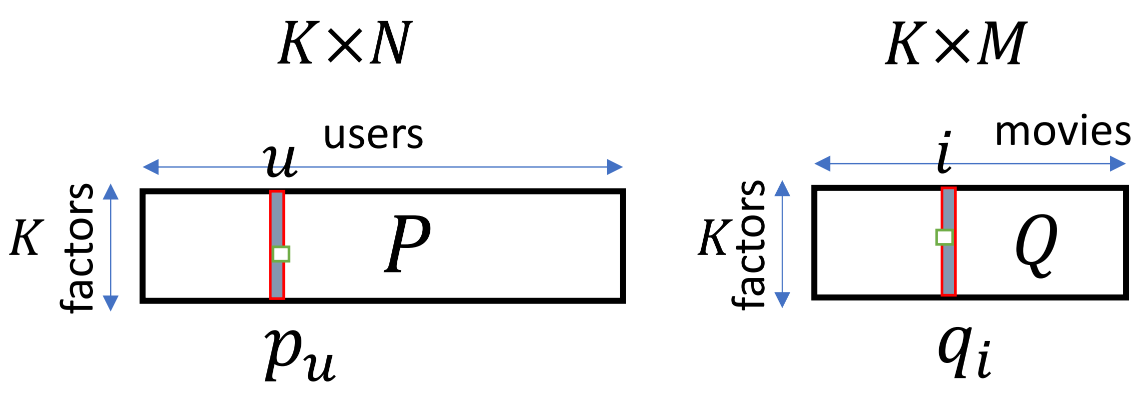
- Our prediction for $\tilde{r}_{4A} = \frac{-1}{|-1|+|-0.8|} \times (-0.8) + \frac{-0.8}{|-1|+|-0.8|} \times 1.8 \approx -0.4$
- Final prediction $r^*_{4A} = \hat{r}_{4A} + \tilde{r}_{4A} \approx 4.1 + (-0.4) = 3.7$

Baseline predictor value $\hat{r}_{4A} = \bar{r} + b_4 + b_A = 3.3 + 0.1 + 0.7 = 4.1$

A latent factor model

Latent-factor model

- The user-movie matrix is very **sparse**: only 1% of all cells are occupied with a rating!
- We want to find a way to “reduce” the matrix by representing it as a product of matrices of lower dimension
- We can categorize the movies and users: is it more action or romance? Does Alice like arty movies or mainstream blockbusters?



P_{ku} = how much factor k affects user u

Q_{ki} = how much factor k occurs in movie i

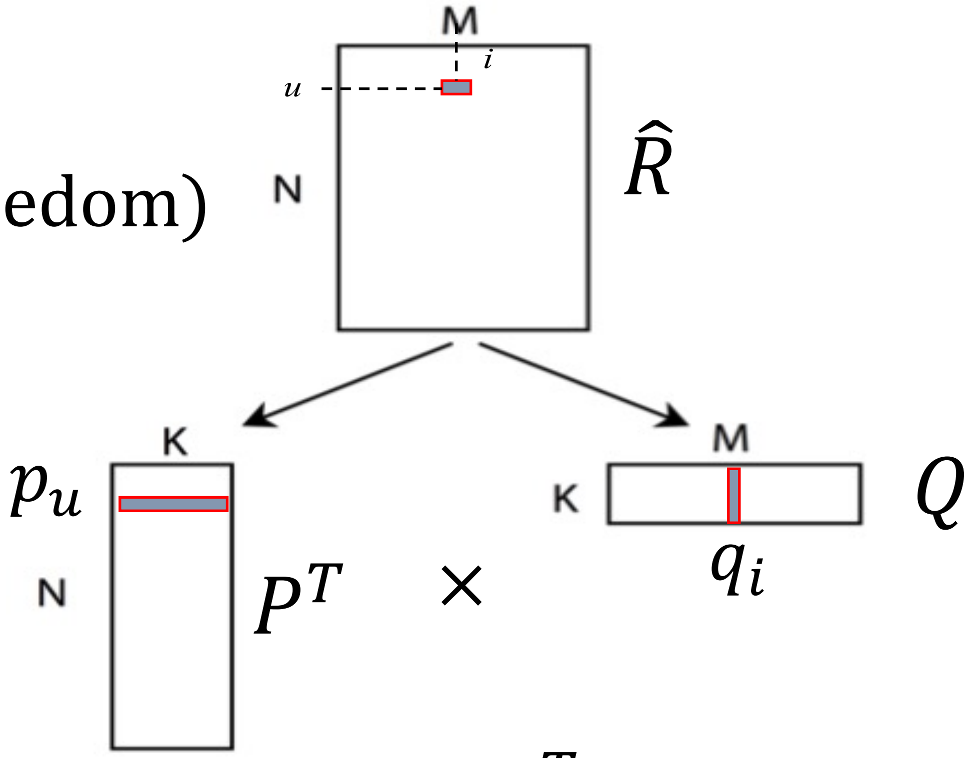
$$\hat{r}_{ui} = \langle p_u, q_i \rangle = p_u^T q_i$$

Find $P, Q \ s.t. \ R \approx_\Omega P^T Q$

for the set of elements Ω of R where it is defined, get for free a value for missing entries: **matrix completion problem**

Properties of the model

We have $K(N + M)$ parameters (degrees of freedom) to match the set of $\{r_{ui}\}$ of α available ratings



Say $N = 5 \cdot 10^5, M = 2 \cdot 10^4$

$\alpha = 10^8 \ll NM = 10^{10}$

$KN + KM$
$\approx 10^7 \ll NM$

if $K = 10$:

$KN = 5 \cdot 10^6, KM = 2 \cdot 10^5$

$KN + KM = 5.2 \cdot 10^6 \ll \alpha$

if $K = 100$:

$KN = 5 \cdot 10^7, KM = 2 \cdot 10^6$

$KN + KM = 5.2 \cdot 10^7 < \alpha$

if $K = 200$: ...

$P^T Q$

$$\hat{r}_{ui} = \langle p_u, q_i \rangle = p_u^T q_i$$

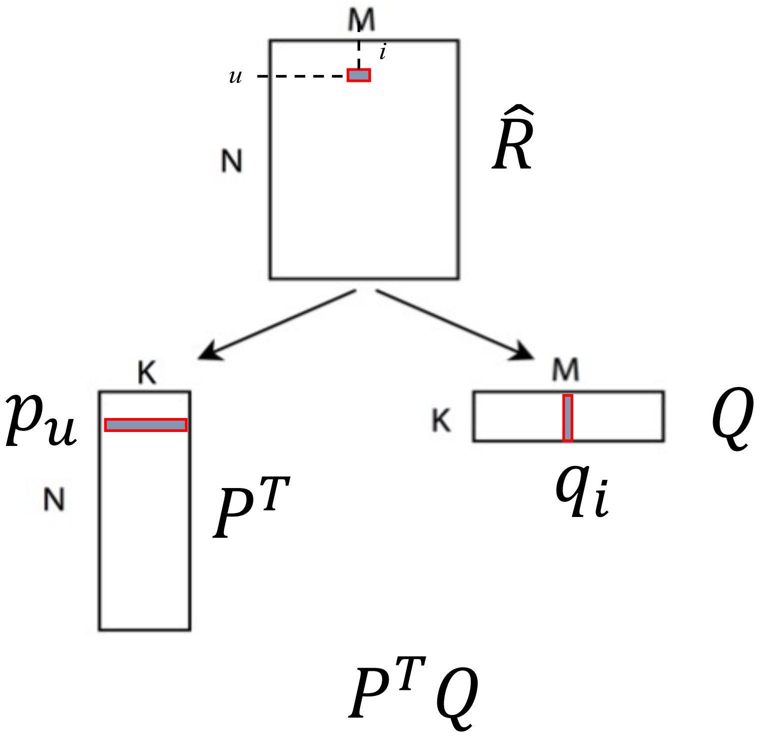
Error minimization

Fix number of factors = K

- P^T: matrix of size $N \times K$ with rows $\{p_u\}$
- Q: matrix of size $K \times M$ with columns $\{q_i\}$
- To find P, Q we solve the following optimization problem:

$$\min_{P,Q} \sum_{(u,i)\in\Omega} (r_{ui} - p_u^T q_i)^2$$

- We can pick K such that $K(M + N)$ is of the same order as the number of ratings in our training set
- Solve by alternating minimization
- If interested more on this topic check https://ee227c.github.io/code/lecture19.html



Alternating minimization

- Problem: $\min_{P,Q} \sum_{(u,i)\in\Omega} (r_{ui} - p_u^T q_i)^2 = \min_{P,Q} \sum_{(u,i)\in\Omega} (r_{ui} - (P^T Q)_{ui})^2$

Algorithm

- Start with some initial P_0
- Step 1: $Q_1 = \arg\min_Q \sum_{(u,i)\in\Omega} (r_{ui} - (P_0^T Q)_{ui})^2,$

$$P_1 = \arg\min_P \sum_{(u,i)\in\Omega} (r_{ui} - (P^T Q_1)_{ui})^2.$$

- Step l: $Q_l = \arg\min_Q \sum_{(u,i)\in\Omega} \left(r_{ui} - \left(P_{l-1}^T Q\right)_{ui}\right)^2,$

$$P_l = \arg\min_P \sum_{(u,i)\in\Omega} (r_{ui} - (P^T Q_{l-1})_{ui})^2.$$

… until convergence

Conclusions

- We examined a basic case of data analysis
- Making predictions is a very important and useful task
- Need to extract information from available data and avoid over fitting
- Many different approaches
- In your project you learn more on the latent factor approach using RBMs (Restricted Boltzmann Machines)

Appendix

The key matrix calculus properties

scalars = italics

$$a = \mathrm{y}^T \mathrm{x} = \mathrm{x}^T \mathrm{y} \Leftrightarrow \frac{da}{d\mathrm{x}} = \left(\frac{da}{dx_1}, \cdots, \frac{da}{dx_n}\right) = \mathrm{y}^T \qquad (*)$$

$$\mathrm{y} = \mathrm{B}\mathrm{x} \Leftrightarrow \frac{d\mathrm{y}}{d\mathrm{x}} = \frac{d}{d\mathrm{x}}(y_1(\mathrm{x}), \ldots, y_n(\mathrm{x}))^T = \mathrm{B} \qquad (**)$$

$$a = \mathrm{y}(\mathrm{x})^T \mathrm{z}(\mathrm{x}) \Leftrightarrow \frac{da}{d\mathrm{x}} = \mathrm{y}(\mathrm{x})^T \frac{d\mathrm{z}(\mathrm{x})}{d\mathrm{x}} + \mathrm{z}(\mathrm{x})^T \frac{d\mathrm{y}(\mathrm{x})}{d\mathrm{x}} \qquad (***)$$

$$a = \mathrm{x}^T \mathrm{B}\mathrm{x} \Leftrightarrow \frac{da}{d\mathrm{x}} \overset{(***)}{=} \mathrm{x}^T \frac{d(\mathrm{B}\mathrm{x})}{d\mathrm{x}} + (\mathrm{B}\mathrm{x})^T \frac{d\mathrm{x}}{d\mathrm{x}} \overset{(**)}{=} \mathrm{x}^T(\mathrm{B} + \mathrm{B}^T) \qquad (1)$$

For matrix calculus please see section 5 in
http://www.atmos.washington.edu/~dennis/MatrixCalculus.pdf

Minimizing SE

$$\min_b ||Ab - y||^2 = \min_b (Ab - y)^T (Ab - y)$$

$$\frac{d}{db}(Ab - y)^T (Ab - y) = 2(Ab - y)^T A$$

$$\frac{d}{db}||Ab - y||^2 = 0 \Leftrightarrow 2(Ab - y)^T A = 0$$

$$\Leftrightarrow A^T (Ab - y) = 0 \Leftrightarrow A^T Ab = A^T y$$

Minimizing SE (alternative way)

$$\min_{\mathrm{b}} SE(\mathrm{b}) = \| \mathrm{Ab} - \mathrm{y} \|_2^2 = (\mathrm{Ab} - \mathrm{y})^T (\mathrm{Ab} - \mathrm{y}) = \text{quadratic function of b}$$

Find b for which $\frac{d}{d\mathrm{b}}\mathrm{SE(b)} = 0$

$$\frac{d[(\mathrm{Ab}-\mathrm{y})^T(\mathrm{Ab}-\mathrm{y})]}{d\mathrm{b}} = \frac{d(\mathrm{b}^T\mathrm{A}^T\mathrm{Ab} - \mathrm{b}^T\mathrm{A}^T\mathrm{y} - \mathrm{y}^T\mathrm{Ab} + \mathrm{y}^T\mathrm{y})}{d\mathrm{b}}$$

$$= \frac{d(\mathrm{b}^T\mathrm{A}^T\mathrm{A}\mathrm{b})}{d\mathrm{b}} + \frac{d(-\mathrm{b}^T\mathrm{A}^T\mathrm{y} - \mathrm{y}^T\mathrm{Ab})}{d\mathrm{b}} \overset{(1),(**)}{=} 2\mathrm{b}^T(\mathrm{A}^T\mathrm{A}) - 2\mathrm{y}^T\mathrm{A}$$

Hence need to solve $\mathrm{b}^T(\mathrm{A}^T\mathrm{A}) = \mathrm{y}^T\mathrm{A} \Leftrightarrow (\mathrm{A}^T\mathrm{A})\mathrm{b} = \mathrm{A}^T\mathrm{y}$

Solve an LP!

If $\mathrm{A}^T\mathrm{A}$ invertible $\Rightarrow \mathrm{b} = (\mathrm{A}^T\mathrm{A})^{-1}\mathrm{A}^T\mathrm{y}$

Note that

i) $(A^T A)^T = A^T A$

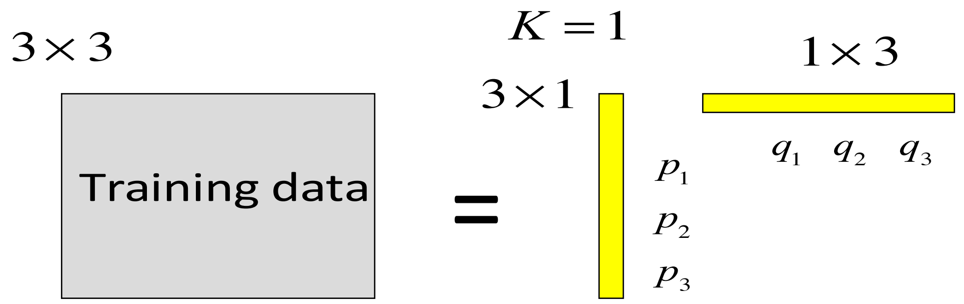
ii) $\mathrm{b}^T\mathrm{A}^T\mathrm{y} = \mathrm{y}^T\mathrm{Ab}$

Latent factor example (next 2 slides)

1. Small data set case: over fitting
   - Number of variables (6) > number of ratings (5)
   - We can match the training set exactly!
2. Larger data set case
   - Number of variables (8) = number of ratings (8)
   - Cannot match exactly the ratings since it would require some of the variables to be <0. Hence we do not over fit (we see better results for our forecasted ratings for the test set)

```
ClearAll["Global`*"];
T = MatrixForm[{{5, _, _}, {_, 5, 1}, {5, _, 4}}];
s = NMinimize[{(5 - p1 q1) ^2 + (5 - p2 q2) ^2 + (1 - p2 q3) ^2 + (5 - p3 q1) ^2 + (4 - p3 q3) ^2,
    p1 ≥ 0 && p2 ≥ 0 && p3 ≥ 0 && q1 ≥ 0 && q2 ≥ 0 && q3 ≥ 0},
   {p1, p2, p3, q1, q2, q3}]
p1 = p1 /. s[[2, 1]];
p2 = p2 /. s[[2, 2]];
p3 = p3 /. s[[2, 3]];
q1 = q1 /. s[[2, 4]];
q2 = q2 /. s[[2, 5]];
q3 = q3 /. s[[2, 6]];
v = {{q1, q2, q3}};
u = {{p1}, {p2}, {p3}};
T
R = MatrixForm[u.v]
```

The 3x3 sub-case



$\{6.75576 \times 10^{-17},$
$\{p1 \to 6.28402,\ p2 \to 1.571,\ p3 \to 6.28402,\ q1 \to 0.79567,\ q2 \to 3.18268,\ q3 \to 0.636536\}\}$

$$\begin{pmatrix} 5 & \_ & \_ \\ \_ & 5 & 1 \\ 5 & \_ & 4 \end{pmatrix}$$

training data

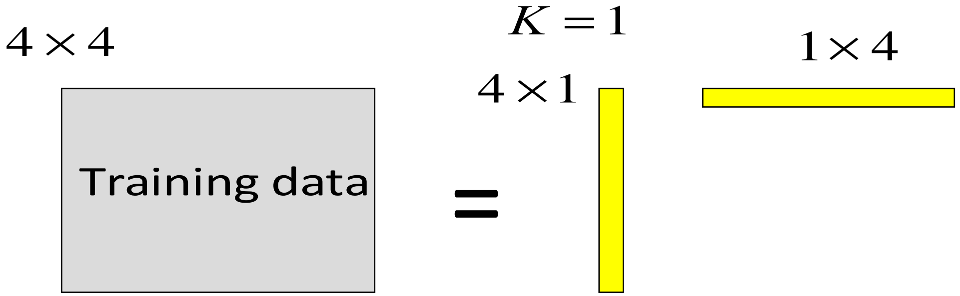
$$\begin{pmatrix} 5. & 20. & 4. \\ 1.25 & 5. & 1. \\ 5. & 20. & 4. \end{pmatrix}$$

prediction

	The Godfather	Ip Man	Mrs. Doubtfire	Aliens
Alice	5	2 ?	—	3
Bob	3 ?	5	1	—
Caroline	5	—	4	? 2
David	—	3	2 ?	5

```
ClearAll["Global`*"];
T = MatrixForm[{{5, _, _, 3}, {_, 5, 1, _}, {5, _, 4, _}, {_, 3, _, 5}}];
s =
 NMinimize[{(5 - p1 q1) ^2 + (3 - p1 q4) ^2 + (5 - p2 q2) ^2 + (1 - p2 q3) ^2 + (5 - p3 q1) ^2 + (4 - p3 q3) ^2 +
     (3 - p4 q2) ^2 + (5 - p4 q4) ^2, p1 ≥ 0 && p2 ≥ 0 && p3 ≥ 0 && p4 ≥ 0 && q1 ≥ 0 && q2 ≥ 0 && q3 ≥ 0 && q4 ≥ 0},
  {p1, p2, p3, p4, q1, q2, q3, q4}]
p1 = p1 /. s[[2, 1]];
p2 = p2 /. s[[2, 2]];
p3 = p3 /. s[[2, 3]];
p4 = p4 /. s[[2, 4]];
q1 = q1 /. s[[2, 5]];
q2 = q2 /. s[[2, 6]];
q3 = q3 /. s[[2, 7]];
q4 = q4 /. s[[2, 8]];
v = {{q1, q2, q3, q4}};
u = {{p1}, {p2}, {p3}, {p4}};
T
MatrixForm[u.v]
```

The full 4x4 case




```
{7.39765, {p1 → 1.78748, p2 → 2.08337, p3 → 2.39396,
  p4 → 1.92976, q1 → 2.34225, q2 → 2.00959, q3 → 1.15765, q4 → 2.16955}}
```

$$\begin{pmatrix} 5 & \_ & \_ & 3 \\ \_ & 5 & 1 & \_ \\ 5 & \_ & 4 & \_ \\ \_ & 3 & \_ & 5 \end{pmatrix}$$

training data

$$\begin{array}{c|cccc} & \text{The Godfather} & \text{Ip Man} & \text{Mrs. Doubtfire} & \text{Aliens} \\ \hline \textit{Alice} & 5 & 2\ ? & - & 3 \\ \textit{Bob} & 3\ ? & 5 & 1 & - \\ \textit{Caroline} & 5 & - & 4 & ?\ 2 \\ \textit{David} & - & 3 & 2\ ? & 5 \end{array}$$

$$\begin{pmatrix} 4.18671 & 3.59209 & 2.06927 & 3.87802 \\ 4.87976 & 4.18671 & 2.41181 & 4.51997 \\ 5.60725 & 4.81088 & 2.77136 & 5.19382 \\ 4.51997 & 3.87802 & 2.23398 & 4.18671 \end{pmatrix}$$

prediction