Recommender Systems:

- **Given**
    - Users: $\mathbf{U_1}$, **…** , $\mathbf{U_n}$
    - Movies: $\mathbf{M_1}$, **…** , $\mathbf{M_m}$
    - Ratings: $\mathbf{R_{ij}}$
- **Goal**: Recommend movies to users
    - **Challenges**:
        - Scale (millions of users, millions of movies)
        - Cold Start (change in user base, change in content)
        - Sparse Data (Not many users rank movies)
- Example: movie recommendation:
    -

	$\mathbf{M_1}$	$\mathbf{M_2}$	$\mathbf{M_3}$	$\mathbf{M_4}$
$\mathbf{U_1}$	R_{11}	R_{12}	R_{13}	R_{14}
$\mathbf{U_2}$	R_{21}	R_{22}	R_{23}	R_{24}
$\mathbf{U_3}$	R_{31}	R_{32}	R_{33}	R_{34}

- Neighborhood Methods:
    - (user, user) similarity measure
        - i.e. recommend same movies to similar users (requires info about users)
    - (item, item) similarity measure
        - i.e. recommend movies that are similar (requires info about movies)

Pros:

- Intuitive / easy to explain
- No training
- Handles new users/items

Challenges:

- Users rate differently (bias)
- Ratings change over time (bias)

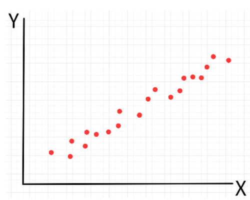
Feature Extraction - Content-Based:

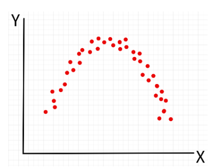
- Realistically:
- It's difficult to characterize movies and users with the right features
- Characterization of users and movies may not be accurate

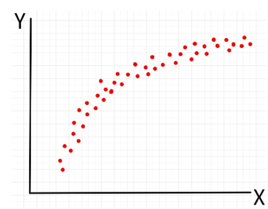
- If you are using genres for example, movies with varying degree of “comedy” will get the tag “comedy”.
- Goal:
    - Discover the best features in an automated way
    - **Content-Based**: assume you have features for movies - want to learn features for users
    - **Collaborative filtering**: want to learn features for both users and movies
- Suppose we have a set of features that characterizes each movie (ex: category, genre…), we could obtain the following **feature-to-movie** similarity matrix:
    -

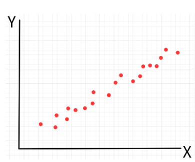
	M_1	**M_2**	**M_3**	**M_4**
F_1 (Romance)	.9	1	.1	0
F_2 (Action)	0	.01	1	.9

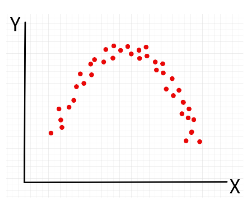
- Given this **feature-to-movie** similarity matrix, how can we predict rating for User 2 or Movie 1 (i.e. R_{12})?
- If we had a **user-to-feature** similarity matrix, we could multiply:
    - **user-to-feature x feature-to-movie = user-to-movie = R_{ij}**
- Collaborative Filtering:
    - Challenge with content-based:
        - How to get the right features $f_1, \ldots, f_k$ **and** $p^{(1)}, \ldots, p^{(n)}$?
    - Can we learn these features?
        - **R = PQ**
    - Can’t use SVD because R is sparse… BUT, we can formulate an optimization problem to solve:
        - $$\min \sum_{i,j \in R} (r_{ij} - p_i^T q_j)^2 + \lambda(\|p\|_F^2 \; \|p\|_F^2)$$
    - To solve, take derivatives wrt P & Q. Then, just like Expectation-Maximization Algorithm from GMM:
        - Start with random Q
        - Get P
        - Improve Q
        - Repeat 2 & 3
- Linear Regression:
    - Find the data.csv file in the regression folder of our course repo
    - Challenge:

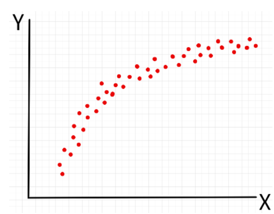
        - Every day my alarm goes off at seemingly random times…
        - I've recorded the times for the past year of so (1 - 355 days)
        - Today is day 356
        - Can you predict when my alarm will ring?
    - Motivation:
        - Given **n** samples / data points **(y_i , x_i)**
        - Understand/explain how **y** varies as a function of **x** (i.e. find a function **y = h(x)** that best fits our data)
    - Suppose we are given a curve **y = h(x)**, how can we evaluate whether it is a good fit to our data?
    - Compare **$h(x_i)$** to **y_i** for all **i**.
    - Goal: For a given distance function **d**, find **h** where **L** is smallest.
        - $$L(h) = \sum_i d(h(x_i), y_i)$$
    - Should **h** be the curve that goes through the most samples? I.e. do we want **$h(x_i)$ = y_i** for the maximum number of **i**?
        - 

;


;


        - **h** may be too complex
overfitting - may not perform well on unseen data
    - The following curves seem the most intuitive "best fit" to our samples. How can we define this best fit mathematically? Is it just about finding the right distance function?
        - 

;


;


Motivation:

- Another way to define this problem is in terms of probability.
- Define **P(Y | h)** as the probability of observing **Y** given that it was sampled from **h**.
- Goal: Find **h** that maximizes the probability of having observed our data.
    - To sum up we can either:
        - Minimize: $$L(h) = \sum_i d(h(x_i), y_i)$$
        - Maximize: **L(h) = P(Y | h)**

Assumptions:

- Let's start by assuming our data was generated by a **linear function** plus some **noise**:
- $\vec{y} = h_\beta(\mathrm{X}) + \vec{\epsilon}$
- Where **h** is linear in a parameter $\boldsymbol{\beta}$.
- Which functions below are linear in $\boldsymbol{\beta}$?
    - **h(x) = $\boldsymbol{\beta_1}$x**

Assumptions:

- 1.The relation between **x** (independent variable) and **y** (dependent variable) is linear in a parameter $\boldsymbol{\beta}$.
- 2.$\boldsymbol{\epsilon_i}$ are independent, identically distributed random variables following a **N(0, $\boldsymbol{\sigma^2}$)** distribution. (Note: **σ** is constant)

Goal:

- Given these assumptions, let's try to solve the max and min problems we defined earlier!
- Q: What does solving these mean?
- A: Finding $\boldsymbol{\beta}$ is equivalent to finding **h**

Least Squares:

- $$\begin{aligned}\beta_{LS} &= \arg\min \sum_i d(h_\beta(x_i), y_i) \\ &= \arg\min \|\vec{y} - h_\beta(\mathrm{X})\|_2^2 \\ &= \arg\min \|\vec{y} - \beta\mathrm{X}\|_2^2\end{aligned}$$
- $$\frac{\partial}{\partial\beta}(y - \beta\mathrm{X})^T(y - \beta\mathrm{X}) = 0$$
$$\frac{\partial}{\partial\beta}(y^Ty - y^TX\beta - \beta^TX^Ty - \beta^TX^TX\beta) = 0$$
$$\frac{\partial}{\partial\beta}(y^Ty - 2\beta^TX^Ty - \beta^TX^TX\beta) = 0$$
$$-2X^Ty - X^TX\beta = 0$$
$$X^TX\beta = X^Ty$$
$$\beta_{LS} = (X^TX)^{-1}X^Ty$$

Maximum Likelihood:

- Since **$\boldsymbol{\epsilon}$ ~ N(0, $\boldsymbol{\sigma^2}$)** and **Y = X$\boldsymbol{\beta}$ + $\boldsymbol{\epsilon}$** then **Y ~ N(X$\boldsymbol{\beta}$, $\boldsymbol{\sigma^2}$)**.
- $$\begin{aligned}\beta_{MLE} &= \arg\max \frac{1}{\sqrt{(2\pi)^n\sigma^n}} \exp(-\frac{\|y - X\beta\|_2^2}{2\sigma^2}) \\ &= \arg\max \exp(-\|y - X\beta\|_2^2) \\ &= \arg\max - \|y - X\beta\|_2^2 \\ &= \arg\min \|y - X\beta\|_2^2 \\ &= \beta_{LS} = (X^TX)^{-1}X^Ty\end{aligned}$$

An unbiased estimator:

- $\boldsymbol{\beta_{LS}}$ is an unbiased estimator of the true $\boldsymbol{\beta}$. That is **E[$\boldsymbol{\beta_{LS}}$]=$\boldsymbol{\beta}$**.

- $$\begin{aligned} E[\beta_{LS}] &= E[(X^T X)^{-1} X^T y] \\ &= (X^T X)^{-1} X^T E[y] \\ &= (X^T X)^{-1} X^T E[X\beta + \epsilon] \\ &= (X^T X)^{-1} X^T X\beta + E[\epsilon] \\ &= \beta \end{aligned}$$