機械学習特論

~ 理論とアルゴリズム ~

(CCA and Recommendation systems)

講師：西郷浩人

Matrix Decomposition（行列分解）and Dimensionality reduction（次元削減）

- Unsupervised dimensionality reduction methods

$$X'X = \lambda_1 \boldsymbol{w}_1 \boldsymbol{w}_1' + \lambda_2 \boldsymbol{w}_2 \boldsymbol{w}_2' \ldots \lambda_p \boldsymbol{w}_p \boldsymbol{w}_p' \rightarrow \lambda_1 \boldsymbol{w}_1 \boldsymbol{w}_1' + \lambda_2 \boldsymbol{w}_2 \boldsymbol{w}_2' \ldots \lambda_k \boldsymbol{w}_k \boldsymbol{w}_k'$$

(k < p)

  - PCA(Principal Component Analysis)
- Supervised dimensionality reduction methods

$$X'Y = \lambda_1 \boldsymbol{w}_1 \boldsymbol{w}_1' + \lambda_2 \boldsymbol{w}_2 \boldsymbol{w}_2' \ldots \lambda_p \boldsymbol{w}_p \boldsymbol{w}_p' = \lambda_1 \boldsymbol{w}_1 \boldsymbol{w}_1' + \lambda_2 \boldsymbol{w}_2 \boldsymbol{w}_2' \ldots \lambda_k \boldsymbol{w}_k \boldsymbol{w}_k'$$

(k < p)

  - PLS(Partial Least Squares)
  - CCA(Canonical Correlation Analysis)
  - Reduced-rank LDA
- In both cases, the number of dimensions k is chosen by a user.

CCA(Canonical Correlation Analysis)

PLS

$$max_{\boldsymbol{a},\boldsymbol{b}} cor^2(\boldsymbol{X}\,\boldsymbol{a}, \boldsymbol{Y}\,\boldsymbol{b})$$

$$\boxed{max_{\boldsymbol{w}} cov^2(\boldsymbol{X}\,\boldsymbol{w}, \boldsymbol{y})}$$

- Objective looks similar to that of PLS.
- CCA maximizes correlation instead of covariance.
  - Correlation is a normalized value which takes between -1 and 1, while covariance is not.
  - The following relationship holds between correlation and covariance.

$$cor^2(\boldsymbol{X}\,\boldsymbol{a}, \boldsymbol{Y}\,\boldsymbol{b}) = \frac{cov^2(\boldsymbol{X}\,\boldsymbol{a}, \boldsymbol{Y}\,\boldsymbol{b})}{var(\boldsymbol{X}\,\boldsymbol{a})\,var(\boldsymbol{Y}\,\boldsymbol{b})}$$

  - Maximizing correlation amounts to solving generalized eigenproblem.

Example: wine data

- X matrix (predictors) measured by anyone.

Wine	Price	Sugar	Alcohol	Acidity
1	7	7	13	7
2	4	3	14	7
3	10	5	12	5
4	16	7	11	3
5	13	3	10	3

- Y matrix (responses) measured by experts.

Wine	Hedonic	Goes with meat	Goes with dessert
1	14	7	8
2	10	7	6
3	8	5	5
4	2	4	7
5	6	2	4

Exercise 1: CCA

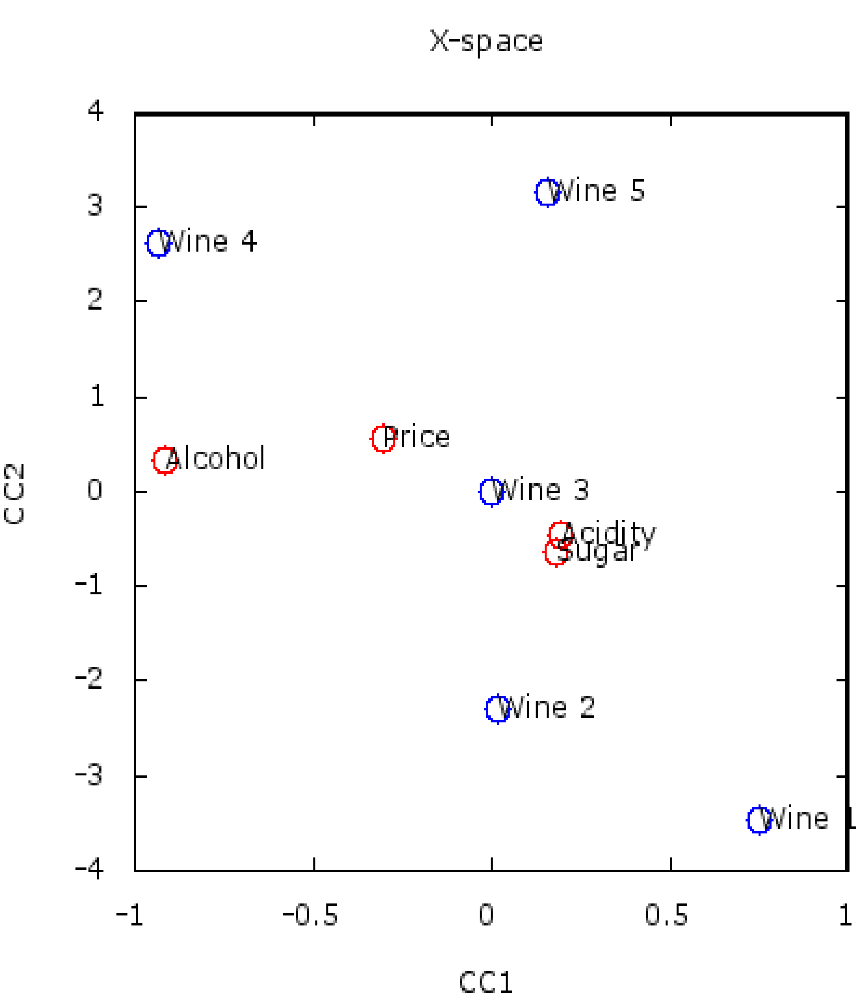
- Download today's data and type

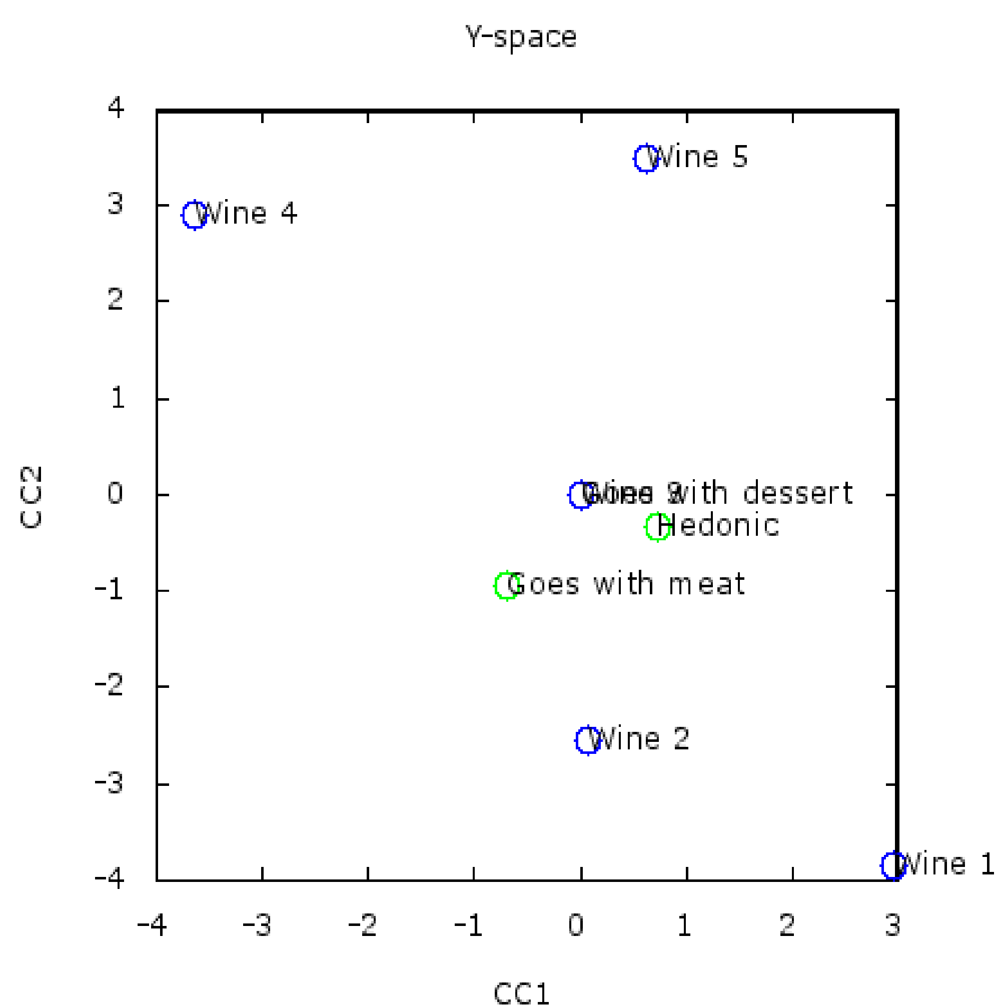
  > ccaExample

  - shows demonstration of CCA in the Wine data.

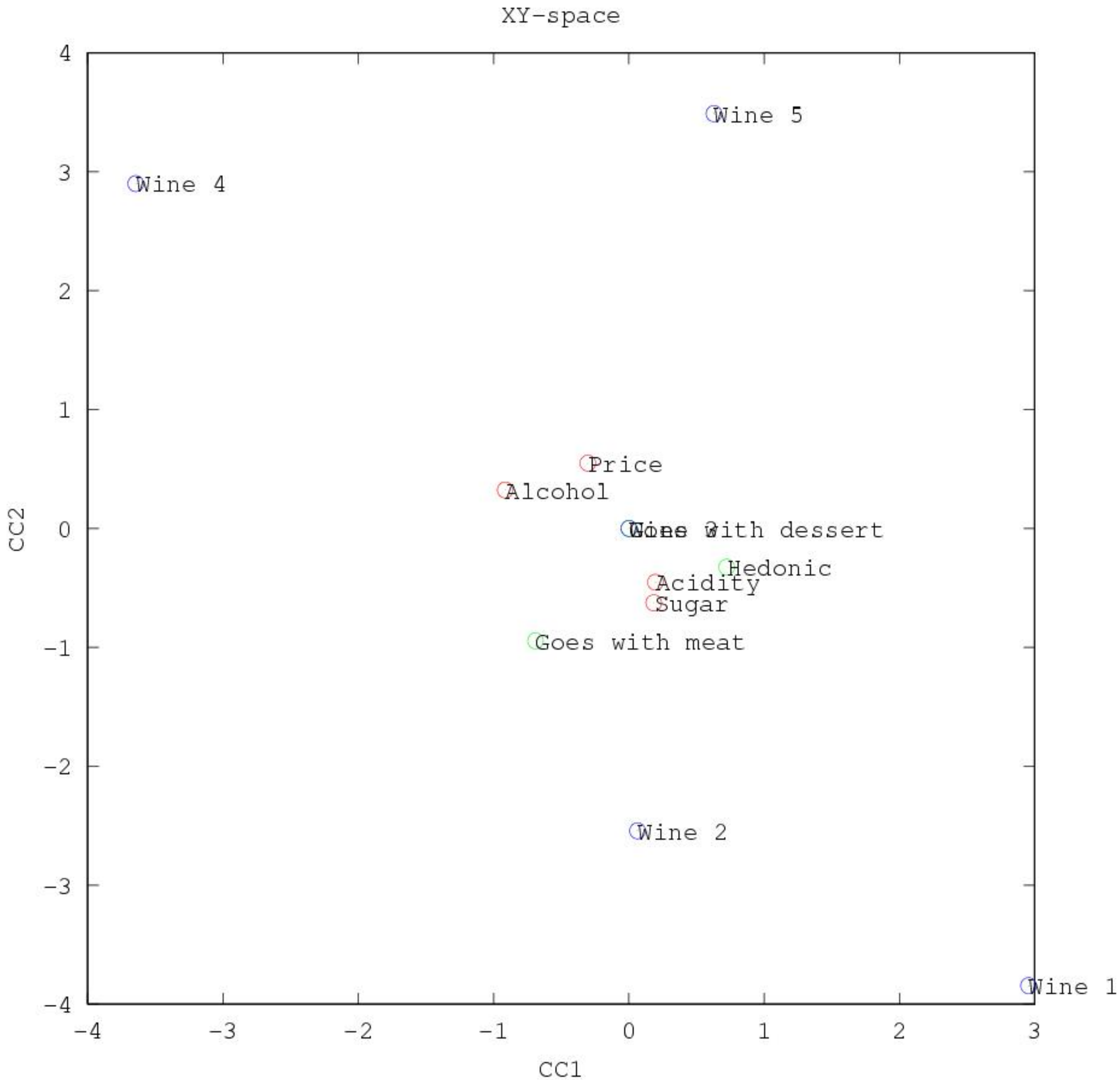
Visualizing Wine data through CCA

- Notice that blue data points are mapped similarly in the two spaces. By superimposing the two space, we can interpret that Hedonic is related to Acidity and Sugar.





XY combined space

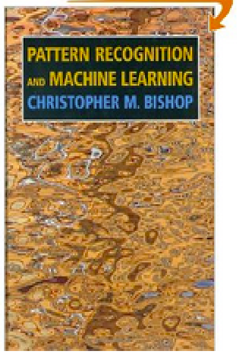


Recommendation system in Amazon.com

www.amazon.co.jp/Pattern-Recognition-Learning-Information-Statistics/dp/0387310738/ref=sr_1_2?ie=UTF8&qid=1358854008&sr=8-2

洋書 詳細検索 ジャンル一覧 Amazonランキング 初めての洋書 英語学習 ペーパーバック 専門書 バーゲン

クリック なか見!検索



自分のイメージを掲載する
この本の中身を閲覧する

Pattern Recognition And Machine Learning (Information Science and Statistics) [ハードカバー]

Christopher M. Bishop (著)

★★★★★ (1 件のカスタマーレビュー) いいね (8)

参考価格: ~~¥ 8,983~~
価格： ¥ 8,853 通常配送無料 詳細
OFF : ¥ 130 (1%)

18点在庫あり。（入荷予定あり） 在庫状況について
この商品は、Amazon.co.jp が販売、発送します。ギフトラッピングを利用できます。

1/23 水曜日 にお届けします。 今から16 時間 と 30 分以内に「お急ぎ便」または「当日お急ぎ便」を選択して注文を確定してください（有料オプション。Amazonプライム会員は無料）

新品の出品：21¥ 7,999より　中古品の出品：3¥ 6,800より

Amazon Student
Amazon Studentに登録して10%Amazonポイント
Amazon Student会員なら、この商品は10%Amazonポイント還元（ポイントが表示されている場合は、表示ポイント＋10%還元）。本10%Amazonポイント還元について。
› その他の情報を見る

Would you like to see this page in English? Click here.

数量: 1
お急ぎ便無料 で配送
Amazonプライム無料体験に登録
ショッピングカートに入れる
または
1-Clickで注文する場合は、サインインをしてください。
ほしい物リストに追加する

こちらからも買えますよ
24の新品/中古品の出品を見る
: ¥ 6,800より
この商品をお持ちですか？
マーケットプレイスに出品する
シェアする

内容紹介

発売日: 2006/8/17

This is the first textbook on pattern recognition to present the Bayesian viewpoint. The book presents approximate inference algorithms that permit fast approximate answers in situations where exact answers are not feasible. It uses graphical models to describe probability distributions when no other books apply graphical models to machine learning. No previous knowledge of pattern recognition or machine learning concepts is assumed. Familiarity with multivariate calculus and basic linear algebra is required, and some experience in the use of probabilities would be helpful though not essential as the book includes a self-contained introduction to basic probability theory.

キャンペーンおよび追加情報

- 本と合わせ買いで割引 対象商品：最大5000円OFF「PCソフト」 | 1000円OFF「アドビPCソフト」
- 掲載画像とお届けする商品の表紙が異なる場合があります。ご了承ください。

よく一緒に購入されている商品

合計価格: ¥ 17,362
両方カートに入れる
在庫状況の表示

☑ 対象商品: Pattern Recognition And Machine Learning (Information Science and Statistics) Christopher M. Bishop ハードカバー ¥ 8,853
☑ The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Springer Series in Statistics) Trevor Hastie ハードカバー ¥ 8,509

この商品を買った人はこんな商品も買っています

ページ: 1 / 10

高速文字列解析の世界—— | 入門 機械学習 | パターン認識と機械学習 | パターン認識と機械学習 | The Elements of Statist ... | Machine Learning: A | わかりやすいパターン認識 | 情報理論 (ちくま学芸文庫) | Machine Learning: An Al

chen.pptx | 灯油調査票(フォーマット).xls | 振込依頼書 (1).xls | 阿久津先生謝金.pdf | (発表と討論2012) 121....xls | 3'修論一覧12.xlsx | すべてを表示

Near-neighbor algorithm

- Let r_{i,k} be the rating of user i on item k, and I_{i} be items for which user i has generated ratings
- Mean rate for user i is

$$\mu_i = \frac{1}{|I_i|} \sum_{j \in I_i} r_{i,j}$$

- Predicted vote for user i on item j is a weighted sum

$$r_{i,j} = \mu_i + C \sum_{k=1}^{K} w_{i,k} \left(r_{k,j} - \mu_k \right)$$

  – Where C is a normalization constant, w_{i,k} are weights of K similar users, where K can be optimized on a validation set.

Near-Neighbor Weighting

- Nearest Neighbor
  - $w_{i,k}=1$ if is k is a neighbor of i, 0 otherwise
- Correlation between user i and k
  - $w_{i,k}=cor(r_i,r_k)$

Exercise 2: Near-neighbor algorithm for movie recommendation

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	3	4	0	0	0
User 2	0	3	5	0	0	5
User 3	0	0	4	5	0	5
User 4	0	0	3	0	0	0
User 5	0	0	3	0	0	0
User 6	2	0	0	2	0	2
User 7	0	0	0	0	5	0
User 8	0	2	1	0	0	1
User 9	0	3	0	0	3	0
User 10	1	0	0	0	0	0

Near-Neighbor Weighting

- Type

  >amazonNNAExample

    loads data, then type

  >amazonNNA(X,1)

    shows result for k=1

k=1

> amazonNNA(X,1)

users: 1 2 3 4 5 6 7 8 9 10

NN(1): 4 8 2 5 4 10 9 2 7 6

recommendation: 3 2 3 3 3 1 2 3 5 1

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	3	4	0	0	0
User 2	0	3	5	0	0	5
User 3	0	0	4	5	0	5
User 4	0	0	3	0	0	0
User 5	0	0	3	0	0	0
User 6	2	0	0	2	0	2
User 7	0	0	0	0	5	0
User 8	0	2	1	0	0	1
User 9	0	3	0	0	3	0
User 10	1	0	0	0	0	0

k=2

```
> amazonNNA(X,2)
users: 1   2   3   4   5   6   7   8   9  10
NN(1): 4   8   2   5   4  10   9   2   7   6
NN(2): 5   4   6   1   1   3   7   1   8  10
recommendation: 3  3  6  3  3  4  2  3  5  1
```

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	3	4	0	0	0
User 2	0	3	5	0	0	5
User 3	0	0	4	5	0	5
User 4	0	0	3	0	0	0
User 5	0	0	3	0	0	0
User 6	2	0	0	2	0	2
User 7	0	0	0	0	5	0
User 8	0	2	1	0	0	1
User 9	0	3	0	0	3	0
User 10	1	0	0	0	0	0

k=3

> amazonNNA(X,3)

users: 1 2 3 4 5 6 7 8 9 10

NN(1): 4 8 2 5 4 10 9 2 7 6

NN(2): 5 4 6 1 1 3 7 1 8 10

NN(3): 8 5 4 2 2 6 10 9 1 1

recommendation: 3 3 3 3 3 4 2 3 5 4

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	3	4	0	0	0
User 2	0	3	5	0	0	5
User 3	0	0	4	5	0	5
User 4	0	0	3	0	0	0
User 5	0	0	3	0	0	0
User 6	2	0	0	2	0	2
User 7	0	0	0	0	5	0
User 8	0	2	1	0	0	1
User 9	0	3	0	0	3	0
User 10	1	0	0	0	0	0

Issues

- Increasing k concentrates on *popular* titles for everyone.
- Prediction is difficult for users with less ratings.

Reference

Amazon.com Recommendations
Item-to-Item Collaborative Filtering
Greg Linden, Brent Smith, and Jeremy York

Recommendation system based on SVD

~ book recommendation example ~

Extracting terms from book titles

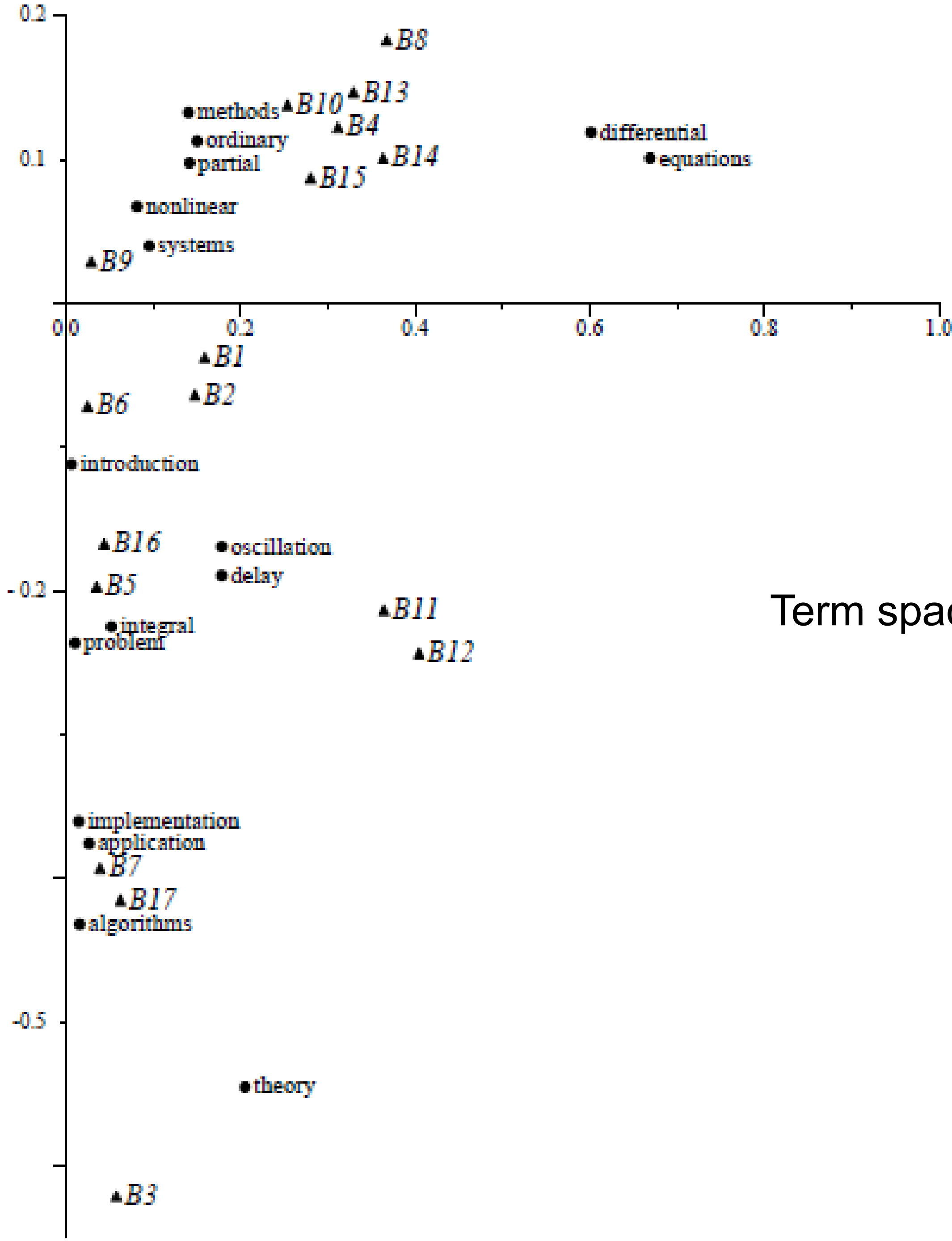
Label	Titles
B1	A Course on Integral Equations
B2	Attractors for semigroups and Evolution Equations
B3	Automatic Differentiation of Algorithms: Theory, Implementation and Applications
B4	Geometircal Aspects of Partial Defferential Equations
B5	Ideals, Varieties, and Algorithms - An Introduction to Computational Algebraic Gometry and Commutative Algebra
B6	Introduction to Hamiltonian Dynamycal Systems and the N-Body Problem
B7	Knapsack Problems: Algorithm and Computer Implementations
B8	Methods of Solving Singular Systems of Ordinary Differential Equations
B9	Nonlinear Systems
B10	Ordinary Differential Equasions
B11	Oscillation Theory for Neutral Differential Equations with Delay
B12	Oscillation Theory of Delay Differential Equations
B13	Pseudodifferential Operators and Nonlinear Partial Differential Equations
B14	Sine Methods for Quadrature and Differential Equations
B15	Stability of Stochastic Differential Equations with Respect to Semi-Martingales
B16	The Boundary Integral Approach to Static and Dynamic Contact Problems
B17	The Double Mellin-Barens Type Integrals and Their Applications to Convolution Theory

Term-document matrix

Documents

Terms																	
	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17
algorithms	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
application	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
delay	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
differential	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
equations	1	1	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
implementation	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
integral	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
introduction	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
methods	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
nonlinear	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
ordinary	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
oscillation	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
partial	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
problem	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
systems	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
theory	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1

Result of SVD



Term space vs Document space

Searching a book by terms “application” and ”theory”

Documents

Terms	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17
algorithms	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
application	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
delay	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
differential	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
equations	1	1	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
implementation	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
integral	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
introduction	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
methods	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
nonlinear	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
ordinary	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
oscillation	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
partial	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
problem	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
systems	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
theory	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1

Searching a book based on terms “application” and ”theory”

- Recall that usual SVD of **X** is $X = U \Sigma V'$
- Now search for a direction **v** that satisfies $q = U \Sigma v'$

  where q =[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]'

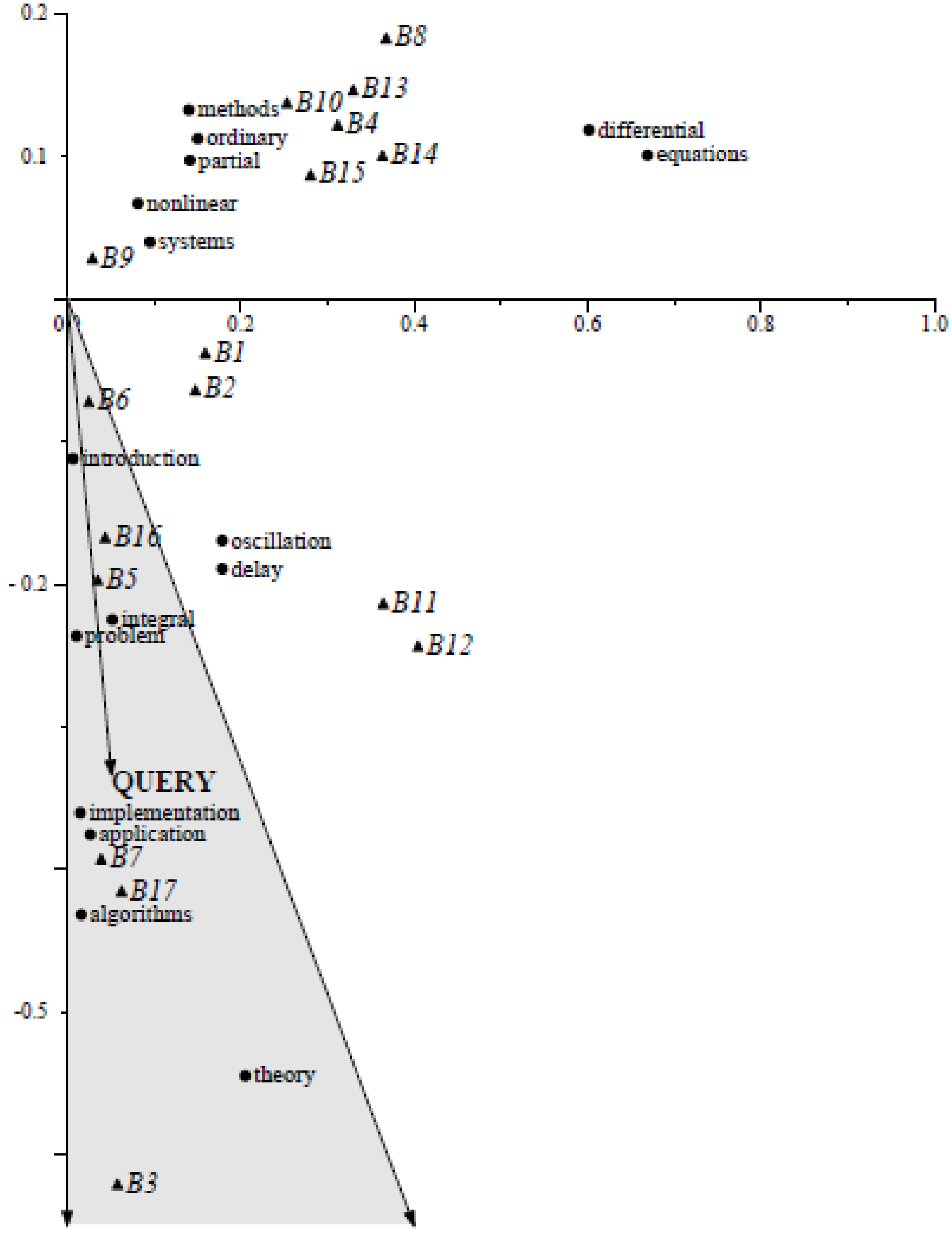
  1 in the second column correspond to “application” and 1 in the last column corresponds to “theory”
- Such **v** can be obtained as $v = q' U \boldsymbol{\Sigma}^{-1}$
  - since $v = \left(\boldsymbol{\Sigma}^{-1} U^{-1} q\right)' = q' U \boldsymbol{\Sigma}^{-1}$

Searching a book based on terms “application” and ”theory”

$$\boldsymbol{v} = \boldsymbol{q}'\boldsymbol{U}\boldsymbol{\Sigma}^{-1}$$

$$\begin{pmatrix} 0.0511 & -0.3337 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}^{t} \begin{pmatrix} 0.0159 & -0.4317 \\ 0.0266 & -0.3756 \\ 0.1785 & -0.1692 \\ 0.6014 & 0.1187 \\ 0.6691 & 0.1209 \\ 0.0148 & -0.3603 \\ 0.0520 & -0.2248 \\ 0.0066 & -0.1120 \\ 0.1503 & 0.1127 \\ 0.0813 & 0.0672 \\ 0.1503 & 0.1127 \\ 0.1785 & -0.1692 \\ 0.1415 & 0.0974 \\ 0.0105 & -0.2363 \\ 0.0952 & 0.0399 \\ 0.20251 & -0.5448 \end{pmatrix} \begin{pmatrix} 4.5314 & 0 \\ 0 & 2.7582 \end{pmatrix}^{-1}$$

探索方向 v


0.2
0.1
-0.2
-0.5
0.0
0.2
0.4
0.6
0.8
1.0
B8
B13
methods
B10
B4
differential
ordinary
equations
partial
B14
B15
nonlinear
systems
B9
B1
B2
B6
introduction
B16
oscillation
delay
B5
B11
integral
problem
B12
QUERY
implementation
application
B7
B17
algorithms
theory
B3

Recommendation system based on SVD

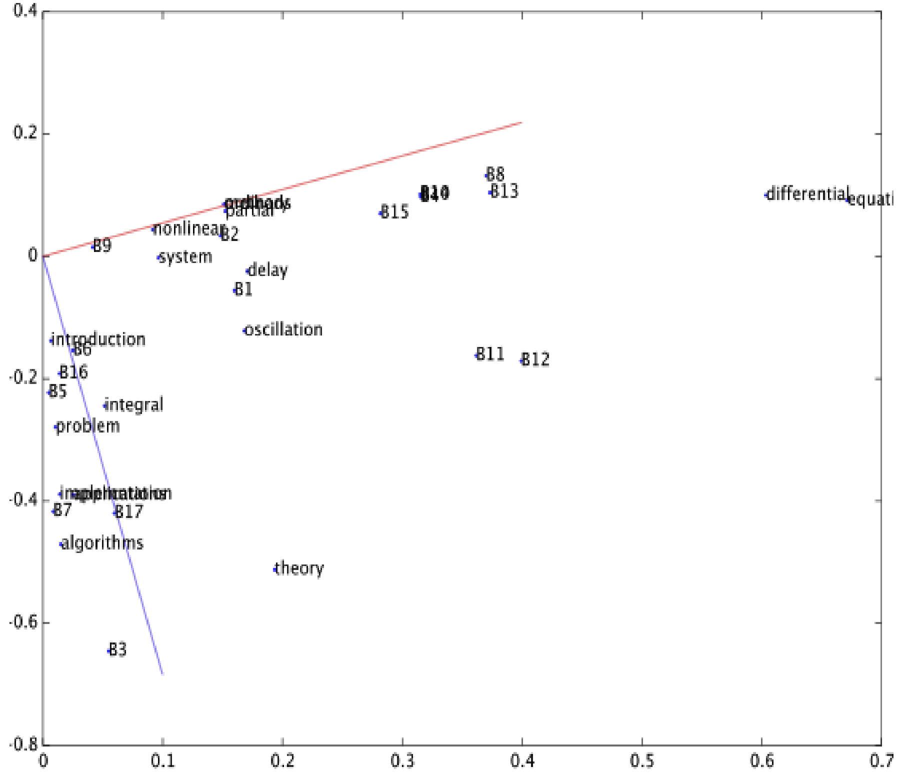
- Technically equivalent to PCA

- Also known as LSI (Latent Semantic Indexing) or LSA (Latent Semantic Analysis)

- Why it works well ? Probably because..
  - Synonyms such as “automobile” and “vehicle” could be guessed using other terms.

Exercise 3: LSI for book recommendation

- Let's try SVD (LSI_demo.m)
- In the last example we searched a book by terms. How can we search a recommendable book based on the documents (books) 4, 8, 10 ?
- The answer is shown as a red line.



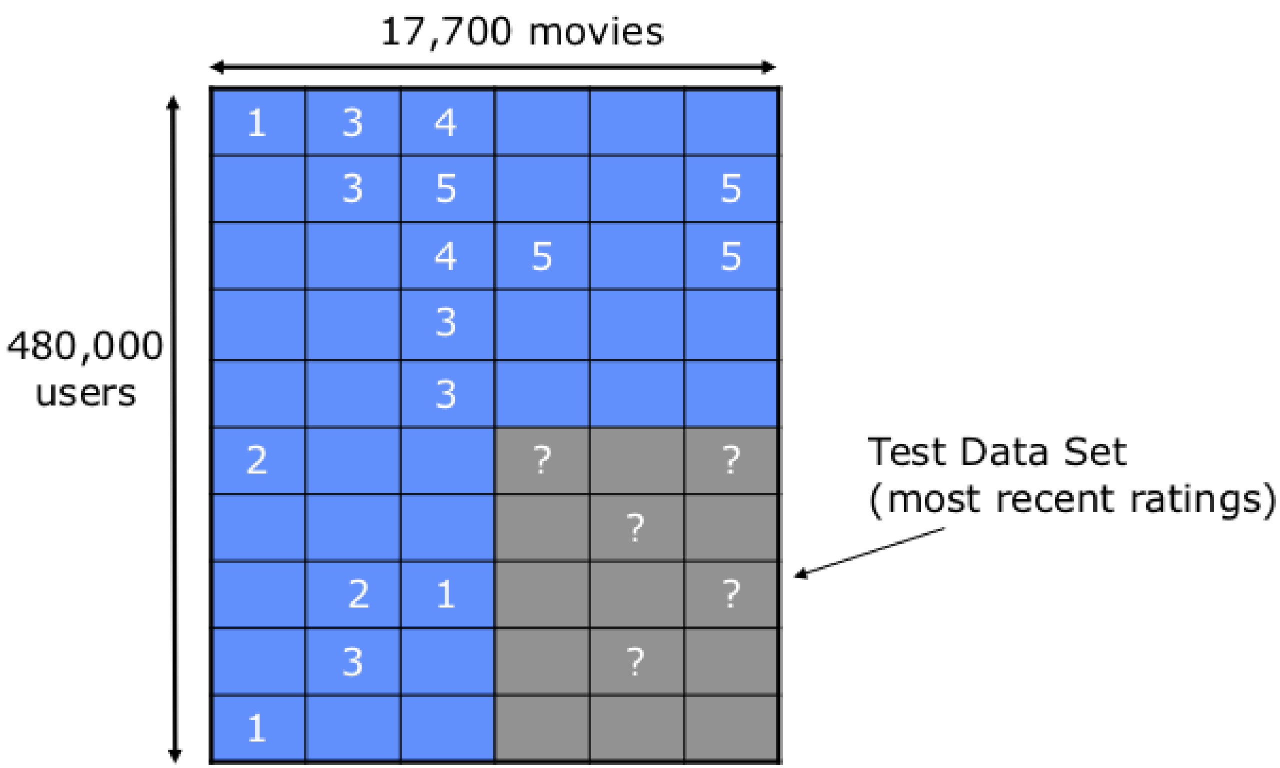
Reference

M.W.Berry, S.T.Dumais and G.W.O'Brien
Using Linear Algebra for Intelligent Information Retrieval

A Practical Approach to Movie Recommendation; Netflix Challenge

Netflix Challenge: Problem

- Goal is to make a predictor with 10% better performance than Netflix's recommendation system.



Key Idea: Matrix Factorization; simplified SVD

- Decompose rating matrix R

$$\boldsymbol{R} = \boldsymbol{U}\,\Sigma\,\boldsymbol{V}\,'$$

  - where U corresponds to “user space”, and V corresponds to “movie space”. If we ignore singular value matrix, then matrix factorization of R is

$$\boldsymbol{R} = \boldsymbol{U}\,\boldsymbol{V}\,'$$

  - where each element represents rating by i-th user for j-th movie

$$R_{i,\,j} = \boldsymbol{u_i}\,' \times \boldsymbol{v_j}$$

  - where **u** and **v** are feature vectors with the same lengths.

Problem formulation

- Goal: Predict ranking by i-th user for j-th movie

$$y^{(i,j)} \approx r^{(i,j)} = \left(v^{(j)}\right)^T u^{(i)}$$

- Objective function

$$J\left(u^{(1)}, \ldots u^{(n_u)}, v^{(1)}, \ldots v^{(n_v)}\right) = \frac{1}{2} \sum_{(i,j):\, y(i,j) != 0} \left(\left(v^{(j)}\right)^T u^{(i)} - y^{(i,j)}\right)^2$$

Problem formulation

- Goal: Predict ranking by i-th user for j-th movie

$$y^{(i,j)} \approx r^{(i,j)} = \left(v^{(j)}\right)^T u^{(i)}$$

- Objective function

$$J\left(u^{(1)}, \dots u^{(n_u)}, v^{(1)}, \dots v^{(n_v)}\right) = \frac{1}{2} \sum_{\underline{(i,j): y(i,j) != 0}} \left(\left(v^{(j)}\right)^T u^{(i)} - y^{(i,j)}\right)^2$$

Sum over only existing ranks

Problem formulation

- Goal: Predict ranking by i-th user for j-th movie

$$y^{(i,j)} \approx r^{(i,j)} = \left(v^{(j)}\right)^T u^{(i)}$$

- Objective function

$$J\left(u^{(1)}, \ldots u^{(n_u)}, v^{(1)}, \ldots v^{(n_v)}\right) = \frac{1}{2} \sum_{\underline{(i,j):\, y(i,j) != 0}} \left(\left(v^{(j)}\right)^T u^{(i)} - y^{(i,j)}\right)^2$$

Sum over only existing ranks

$$+ \frac{\lambda}{2} \sum_{j=1}^{n_v} \sum_{k=1}^{n} \left(v_k^{(j)}\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_v} \sum_{k=1}^{n} \left(u_k^{(i)}\right)^2$$

Problem formulation

- Goal: Predict ranking by i-th user for j-th movie

$$y^{(i,j)} \approx r^{(i,j)} = \left(v^{(j)}\right)^T u^{(i)}$$

- Objective function

$$J\left(u^{(1)}, \dots u^{(n_u)}, v^{(1)}, \dots v^{(n_v)}\right) = \frac{1}{2} \sum_{\underline{(i,j): y(i,j) != 0}} \left(\left(v^{(j)}\right)^T u^{(i)} - y^{(i,j)}\right)^2$$

Sum over only existing ranks

$$\underline{+ \frac{\lambda}{2} \sum_{j=1}^{n_v} \sum_{k=1}^{n} \left(v_k^{(j)}\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_v} \sum_{k=1}^{n} \left(u_k^{(i)}\right)^2}$$

L2 regularization prevents overfitting

Derivative of the objective

- Can be obtained similarly to ridge regression as;

$$\frac{\partial J}{\partial u_k^{(i)}} = \sum_{(i,j):y(i,j)!=0} \left(\left(v^{(j)} \right)^T u^{(i)} - y^{(i,j)} \right) v_k^{(j)} + \lambda u_k^{(i)}$$

$$\frac{\partial J}{\partial v_k^{(j)}} = \sum_{(i,j):y(i,j)!=0} \left(\left(v^{(j)} \right)^T u^{(i)} - y^{(i,j)} \right) u_k^{(i)} + \lambda v_k^{(j)}$$

- Then, ready to use gradient-based optimizers such as gradient descent, SGD or conjugate gradient.

Exercise 4: Movie recommendation

- Add your movie recommendations to "myratings.m" by refererering to "movie_ids.txt"
  - Adding more ratings gives you more accurate personal movie recommendation.
- Then, run "netflix.m". After training, it will display personal movie recommendations.
- Try
  - Adding more ratings
  - Changing regularization parameter lambda.
- Code is used in the lecture by Andrew Ing.

Dataset (movielens 100k)

- Training data set
  - Y is a 1682x943 matrix, containing ratings (1-5) of 1682 movies on 943 users
  - Plus your personal recommendation

Practical Issues in ML/DS project

- Data normalization

$$x \leftarrow \frac{x - mean(x)}{std(x)}$$

- Data augmentation

$$x_{missing} \leftarrow mean(x_{known})$$

- Data imbalance
  - Problem: the number of positives differ a lot from the number of negatives.
  - Solution: Downsample the larger class.