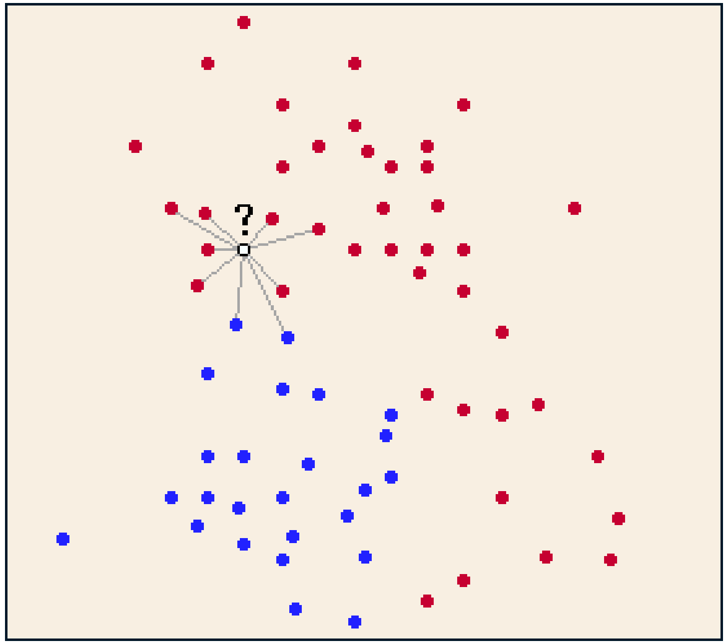
Knn & Recommender Systems

Praphul Chandra

K-Nearest Neighbor

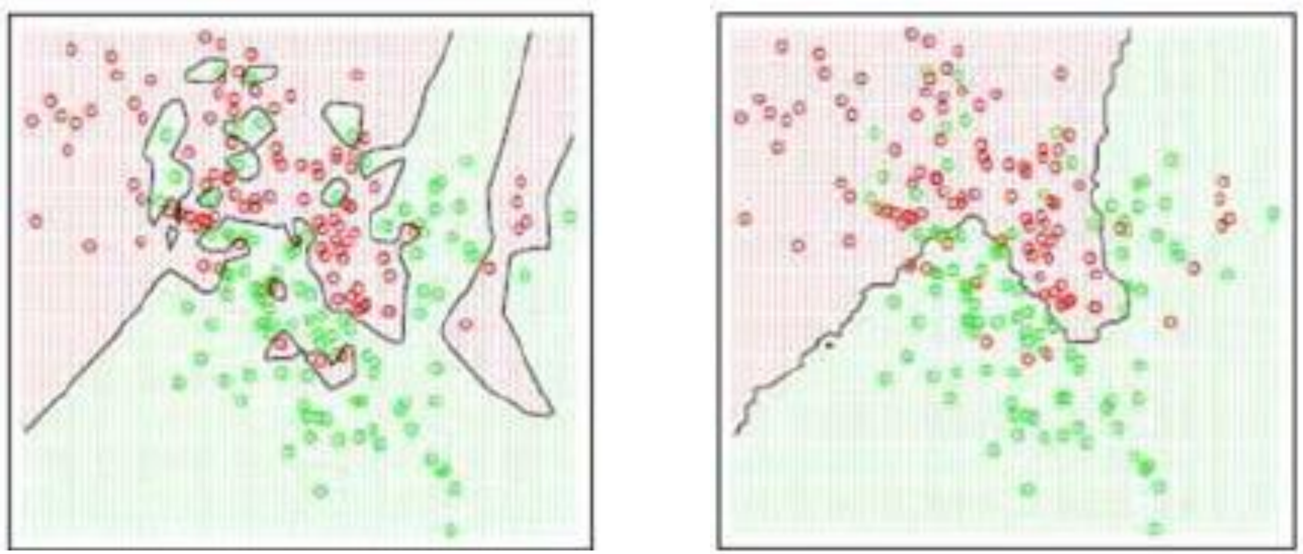
- Statistical Decision Theory
  - The best prediction of Y at an point X=x is the conditional mean. (L2 loss) $f(x) = \mathbb{E}[Y|X = x]$
  - knn: At each point x, approximate y by averaging all y_i with input x_i near x $\hat{f}(x) = Ave(y_i|x_i \in N_k(x))$
- Two approximations
  - Expectation is approximated by averaging over sample data.
  - Conditioning at a point x is relaxed to conditioning on some region “close” to x
- Note
  - Model Free *(No assumption on form of f)*
  - Computational Complexity *(Time, Space)*
  - Locally constant
- Behavior
  - Large k : Smoother boundaries
  - Large N : Large storage req. (space complexity)
  - Large p : lower accuracy (curse of dimensionality)
  - Choice of distance metric (Euclidean, Manhattan, Gower)



Knn : Choosing k

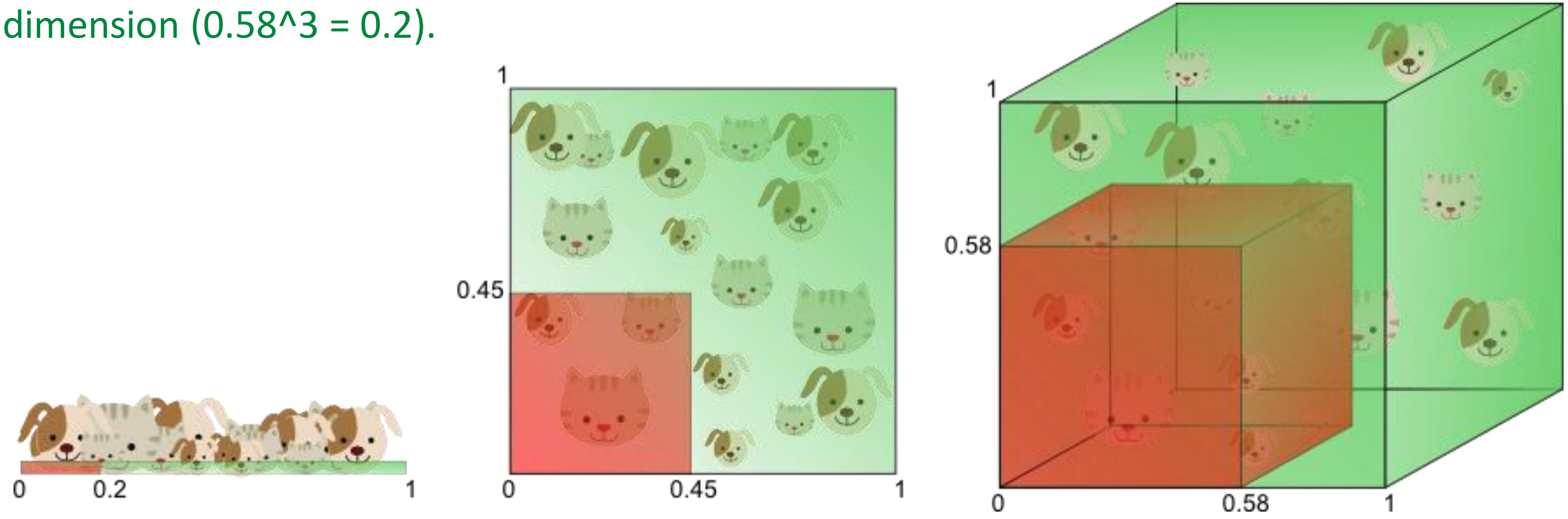
- Larger k
  - Smoother boundaries
  - Higher error (Train or test?)

- Optimal k?
  - Hyper-parameter optimization: Heuristic or Cross Validation

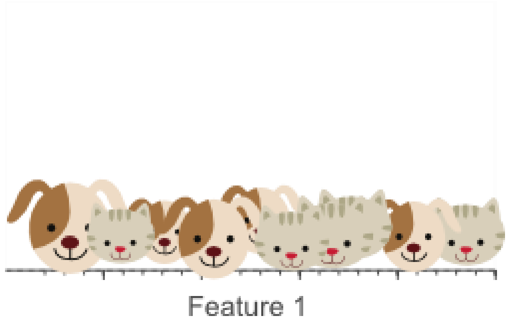


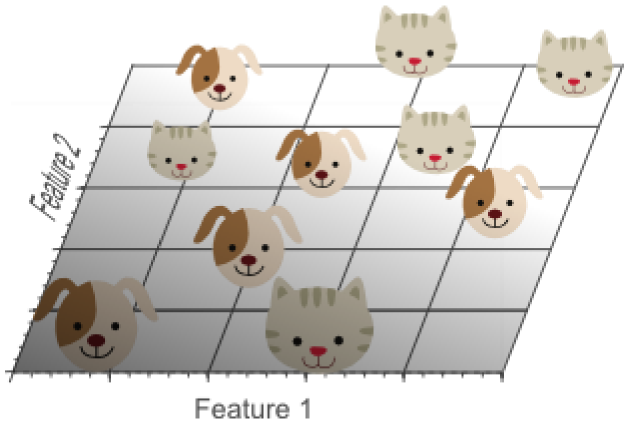
Knn: Dealing with the curse of dimensionality

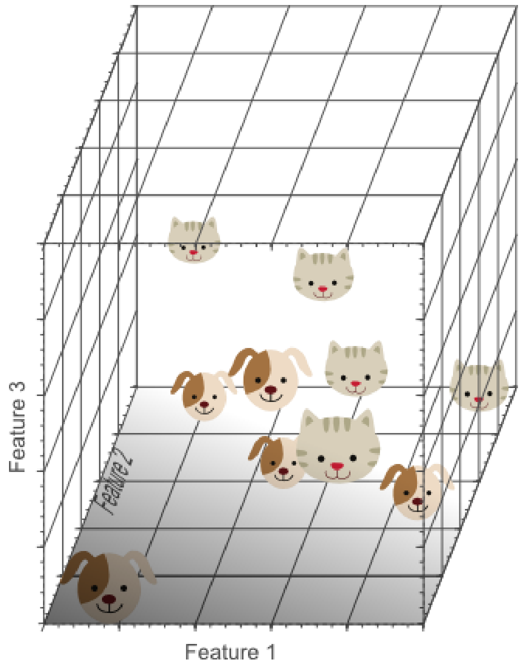
- The more features we use, the more sparse the data becomes
  - Sparseness is not uniformly distributed over the space.
  - The amount of training data needed grows exponentially with the number of dimensions.
- If we want our training data to cover 20% of this range,
  - In 1D: then the amount of training data needed is 20% of the complete population of cats and dogs.
  - In 2D, to cover 20% of the 2D feature range, we now need to obtain 45% of the complete population of cats and dogs in each dimension (0.45^2 = 0.2).
  - In 3D: to cover 20% of the 3D feature range, we need to obtain 58% of the population in each dimension (0.58^3 = 0.2).



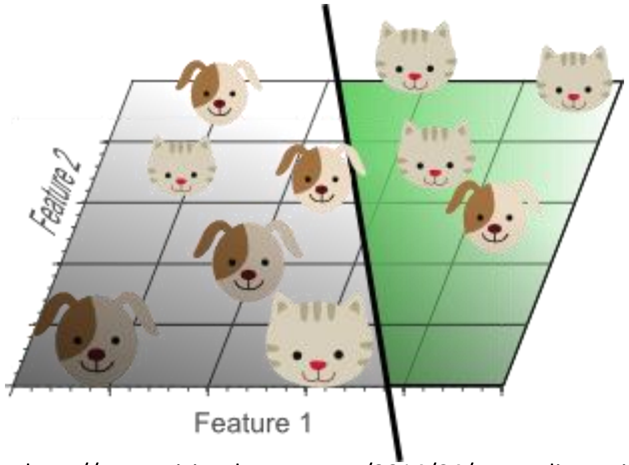
On the flip side: The boon of dimensionality

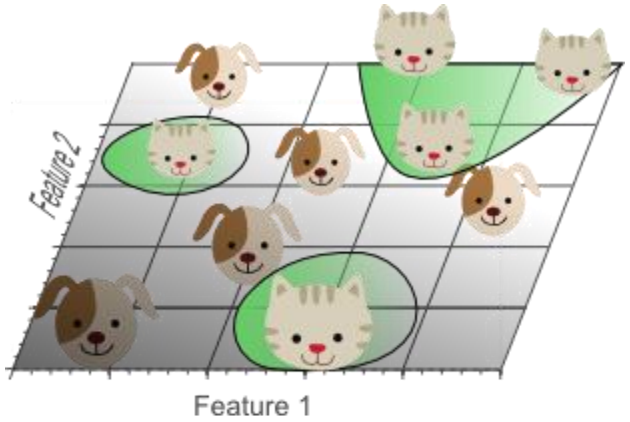


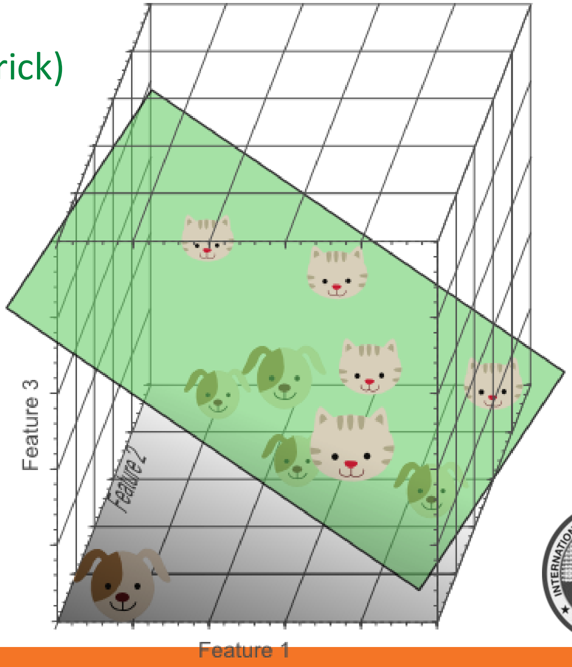




- More features ➔ Sparser Data (nearest neighbors not near (similar) enough)
- More feature ➔ More space between classes ➔ Separability
  - Linear separability in high dimensions ➔ Non-linear separability in fewer dimensions (kernel trick)
  - Guard against overfitting?



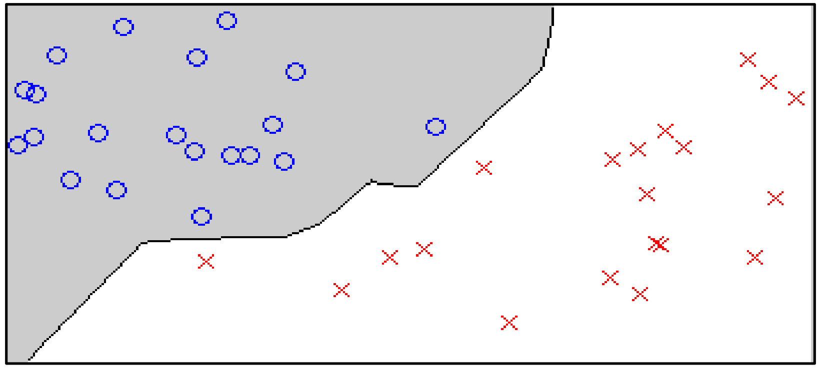




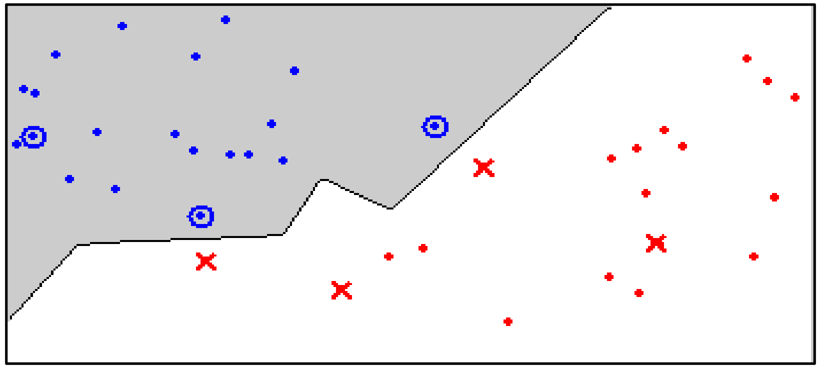
http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/#comment-241

Knn: Dealing with Large n

- Computational complexity
  - Critically depends on n
- Do we really need all the n points?
  - Can we drop some of the points?
  - Yet achieve the same accuracy?
  - a.k.a. Data reduction
- Each point / element / row / tuple is
  - Either a prototype
    - Needed for correct classification
  - Or an absorbed point
    - Not needed for correct classification given the prototypes
  - Or an outlier
    - Must be removed to improve generalization (smoother boundaries)

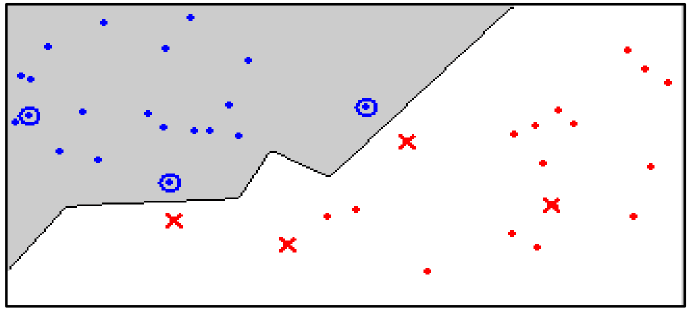


Original data



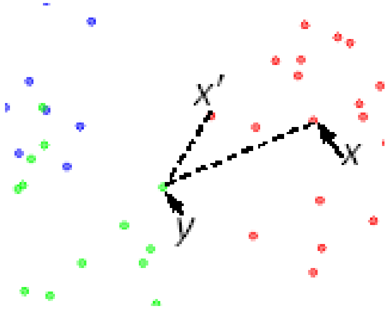
Condensed data

Knn: Dealing with Large n (cont'd) : Condensed knn / cnn / Hart's alg.



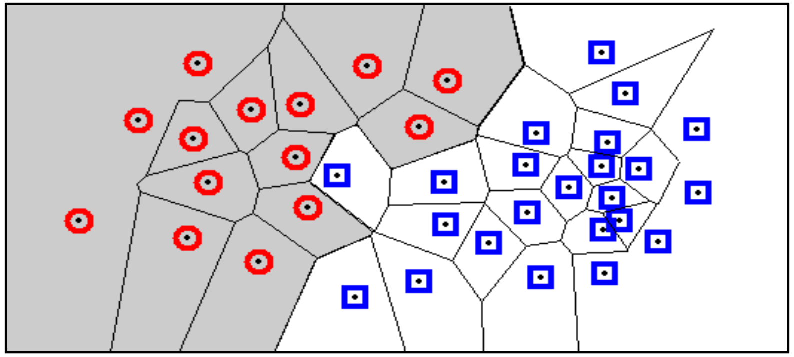
Condensed data

- Key Idea
  - Data reduction followed
  - Prototype selection
- Prototype Set (U)
  - Select subset U of X s.t. 1nn performs equally well on U and X
- Algorithm
  - Initialize U randomly with one element
  - **Scan** X, looking for an element x whose nearest prototype from U has a different label than x.
  - Remove x from X and add it to U
  - Iterate till no more prototypes are added to U.
- Efficiency
  - **Scan** the training examples in order of decreasing border ratio $a(x) = \frac{\|x'-y\|}{\|x-y\|}$
  - Denominator: Distance of x to the closest example *y* having a different color than *x*,
  - Numerator : Distance from *y* to its closest example *x'* with the same label as *x*

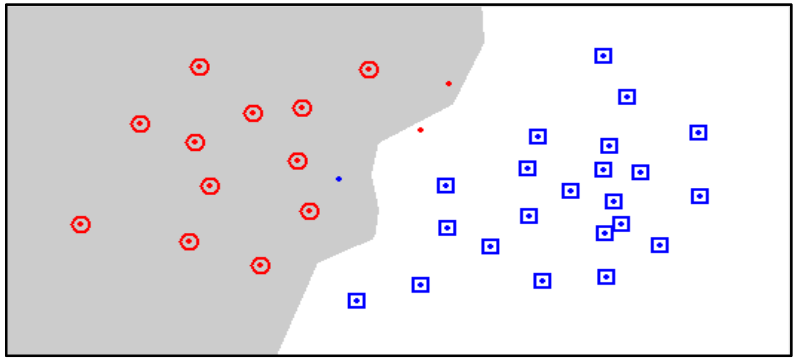


Improving (Smoothing) knn | Avoid overfitting

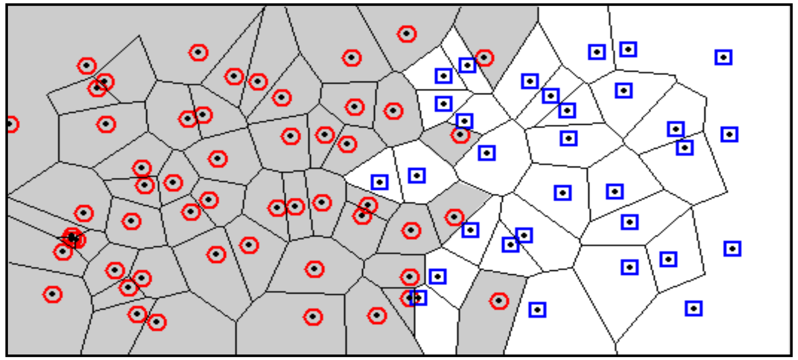
- Wilson editing : Remove points that do not agree with the majority of their k nearest neighbours (Class outliers)
- Edited NN



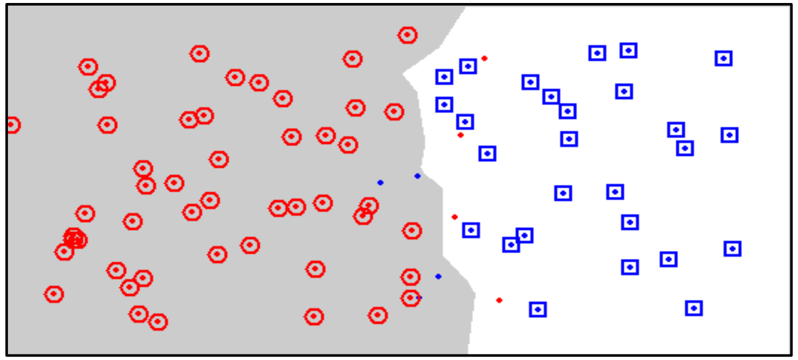
Original data



Wilson editing with k=7



Original data



Wilson editing with k=7

knn: Summary

- The best prediction of Y at an point X=x is the conditional mean. (L2 loss)
- At each point x, approximate y by averaging all y_i with input x_i near x

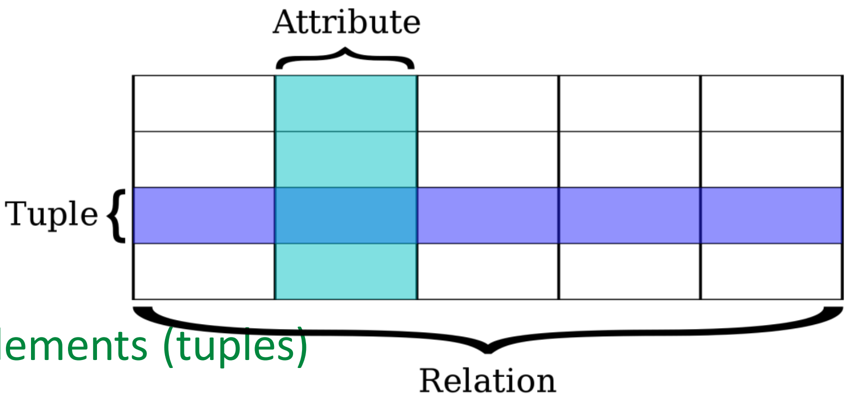
- Lazy | Model Free *(No assumption on form of f)*
- Computational Complexity *(Time, Space)*

- Distance based algorithm
  - Scaling attributes is important
  - Attributes with larger range can dominate e.g., Age versus Salary
  - May not be suitable for high dimensional data
- Categorical variables and Ordinal variables need to be appropriately measured
- Can be used for both regression and classification

Recommender Systems

Praphul Chandra

Association Rules



- What?
  - Are statements about relations among features (attributes) : across elements (tuples)
  - People who buy diapers are likely to beer
- How?
  - Use a transaction-itemset data model
- Why?
  - Used to make recommendations
  - Can we do better?

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

	Beer	Bread	Milk	Diaper	Eggs	Coke
T_1	0	1	1	0	0	0
T_2	1	1	0	1	1	0
T_3	1	0	1	1	0	1
T_4	1	1	1	1	0	0
T_5	0	1	1	1	0	1

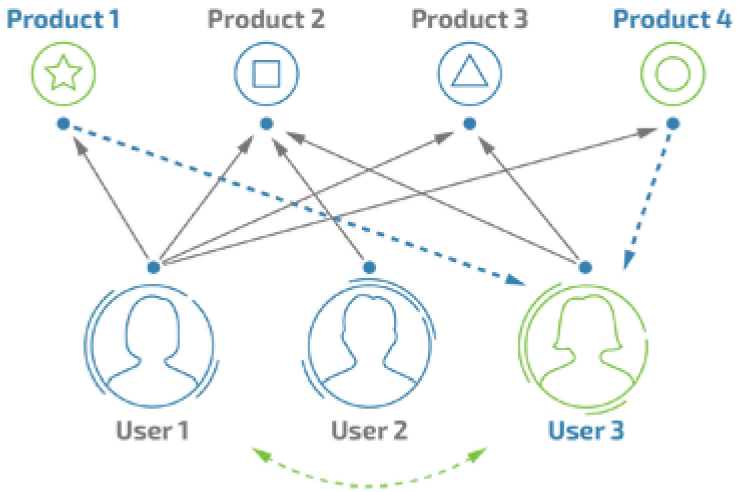
Personalized Recommendations

CRITIC	TITANIC	BATMAN	INCEPTION	SUPERMAN RETURNS	SPIDERMAN	MATRIX
MICHEL	2.5	3.5	3	3.5	2.5	3
SATYA	3	3.5	1.5	5	3	3.5
PARANAV	2.5	3	N/A	3.5	N/A	4
SURESH	N/A	3.5	3	4	2.5	4.5
TOM	3	4	2	3	2	3
LEO	3	4	N/A	5	3.5	3
CHAN	N/A	4.5	N/A	4	1	N/A

- Recommender Systems
  - Recommend items (content) based on user ratings of item
  - "Ratings" may be
    - Explicit, e.g. buying or rating an item
    - Implicit, e.g. browsing time, no. of mouse clicks
- Collaborative filtering
  - Input
    - User-Rating Matrix (Incomplete : Sparse)
  - Output
    - For a particular user, complete the row
  - If user-u likes item-j, recommend item-j' that was **liked by other users like him** : User-Based
  - If user-u likes item-j, recommend item-j' that **is similar to item-j** : Item-Based
- Others
  - Matrix Factorization
  - Content based (e.g. Text)
  - Hybrid (Formulate as a Supervised Learning)

User Based Collaborative Filtering

- Input
  - User-Rating Matrix (Incomplete : Sparse)
- Output
  - For a particular user, complete the row



- Key Idea
  - If u likes j,
  - recommend j' that was liked by other users like him
  - Quantify user-user similarity
  - Use user-user similarity to 'impute' missing values

CRITIC	TITANIC	BATMAN	INCEPTION	SUPERMAN RETURNS	SPIDERMAN	MATRIX
MICHEL	2.5	3.5	3	3.5	2.5	3
SATYA	3	3.5	1.5	5	3	3.5
PARANAV	2.5	3	N/A	3.5	N/A	4
SURESH	N/A	3.5	3	4	2.5	4.5
TOM	3	4	2	3	2	3
LEO	3	4	N/A	5	3.5	3
CHAN	N/A	4.5	N/A	4	1	N/A

User_sim for CHAN		TITANIC	INCEPTION	MATRIX		TITANIC	INCEPTION	MATRIX
0.7125006		2.5	3	3		1.7812515	2.1375	2.1375
0.760215	*	3	1.5	3.5	==>	2.280645	1.1403	2.66075
0.6831639		2.5	N/A	4		1.7079098	N/A	2.73266
0.7028414		N/A	3	4.5		N/A	2.1085	3.16279
0.7341787		3	2	3		2.2025361	1.4684	2.20254
0.80555		3	N/A	3		2.41665	N/A	2.41665
1		N/A	N/A	N/A		N/A	N/A	N/A

UBCF : Dis(similarity)

CRITIC	TITANIC	BATMAN	INCEPTION	SUPERMAN RETURNS	SPIDERMAN	MATRIX
MICHEL	2.5	3.5	3	3.5	2.5	3
SATYA	3	3.5	1.5	5	3	3.5
PARANAV	2.5	3	N/A	3.5	N/A	4
SURESH	N/A	3.5	3	4	2.5	4.5
TOM	3	4	2	3	2	3
LEO	3	4	N/A	5	3.5	3
CHAN	N/A	4.5	N/A	4	1	N/A

- Pearson correlation
  - Ignore items that one user has rated but the other has not.
  - ➔ Users with few rated items in common will have very high similarities
  - What if each user has rated many items but have rated only two overlapping items?
- Cosine similarity (mean reduced)
  - Sum over Intersection = Ignore items that one user has rated but the other has not = Pearson
  - Sum over Union : no rating = 0 rating : $(r_{u,j} - \bar{r}_u)$= 0
  - Significance weighting : Numerator does not increase for unshared items but denominator increases

$$r_{u,j} \quad : \quad \text{rating of user-i to item-j}$$

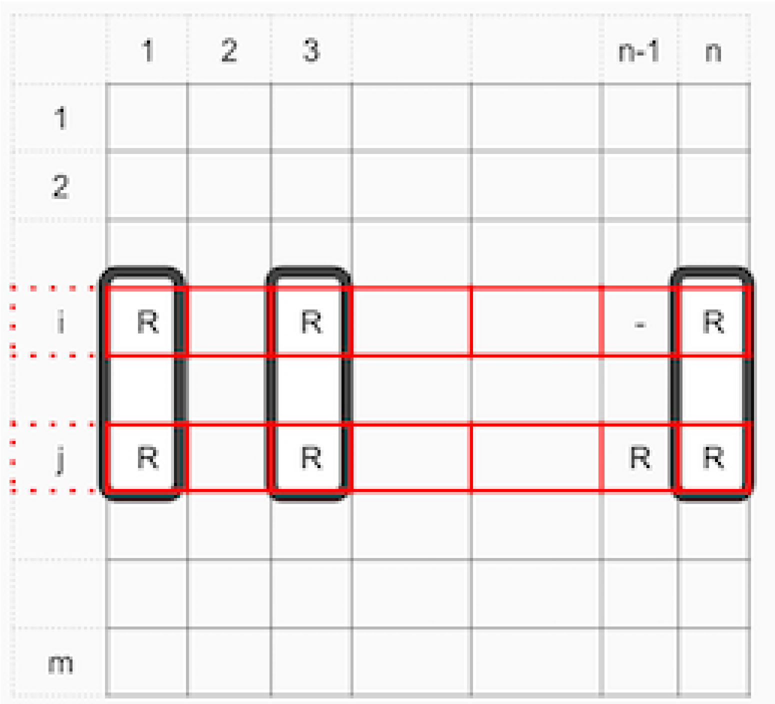
$$\bar{r}_u = \frac{1}{|X_u|} \sum_{j \in X_u} r_{u,j}$$

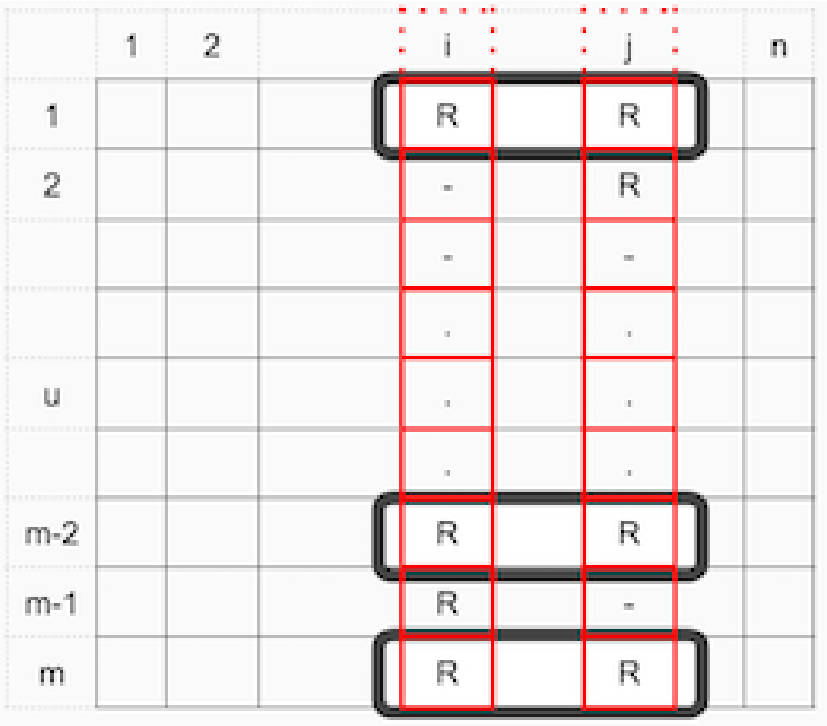
$$\hat{r}_{u,j} = \bar{r}_u + \kappa \sum_{i=1} w_{u,i}(r_{i,j} - \bar{r}_i)$$

$$w_{u,i} = \frac{cov(r_u, r_i)}{\sigma_{r_u}\sigma_{r_i}} = \frac{\sum_{j \in (X_u \cap X_i)} (r_{u,j} - \bar{r}_u)(r_{i,j} - \bar{r}_i)}{\sqrt{\sum_{j \in (X_u \cap X_i)} (r_{u,j} - \bar{r}_u)^2} \sqrt{\sum_{j \in (X_u \cap X_i)} (r_{i,j} - \bar{r}_i)^2}}$$

$$w_{u,i} = cos(r_u - \bar{r}_u, r_i - \bar{r}_i) = \frac{(r_u - \bar{r}_u)(r_i - \bar{r}_i)}{||r_u - \bar{r}_u||||r_i - \bar{r}_i||} = \frac{\sum_j (r_{u,j} - \bar{r}_u)(r_{i,j} - \bar{r}_i)}{\sqrt{\sum_j (r_{u,j} - \bar{r}_u)^2} \sqrt{\sum_j (r_{i,j} - \bar{r}_i)^2}}$$

Item Based Collaborative Filtering





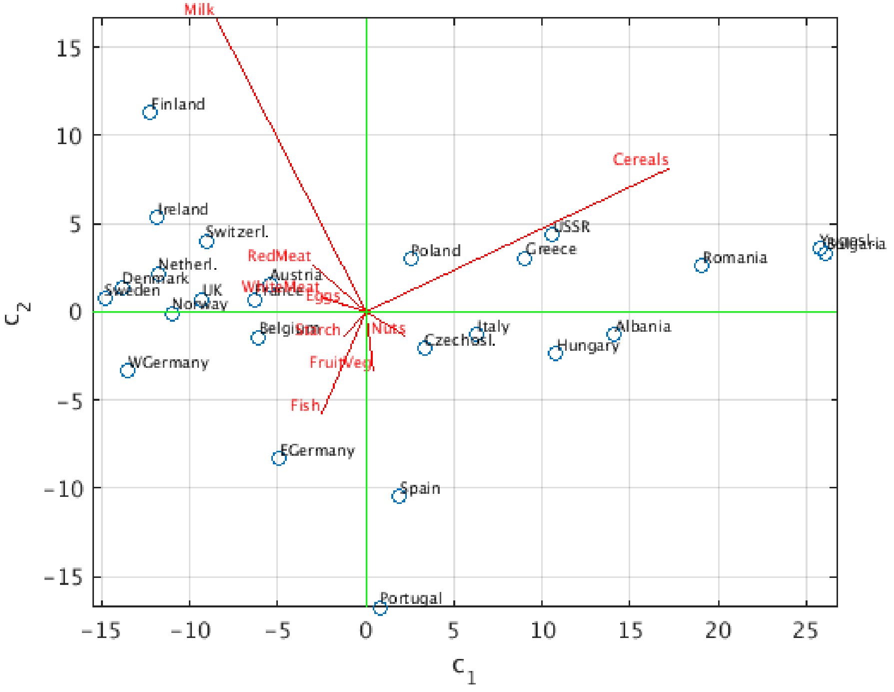
- UBCF
  - For a particular **user**, complete the <u>row</u>
- IBCF
  - For a particular **item**, complete the <u>column</u>

Matrix Factorization

- Key Idea
  - Both Users and Items lie in some underlying space : Related by “taste”
  - Can we uncover this underlying dimension space in which users and items lie?
  - Can we create new dimensions (latent factors) in which users and items lie?
  - Recall : PCA & SVD

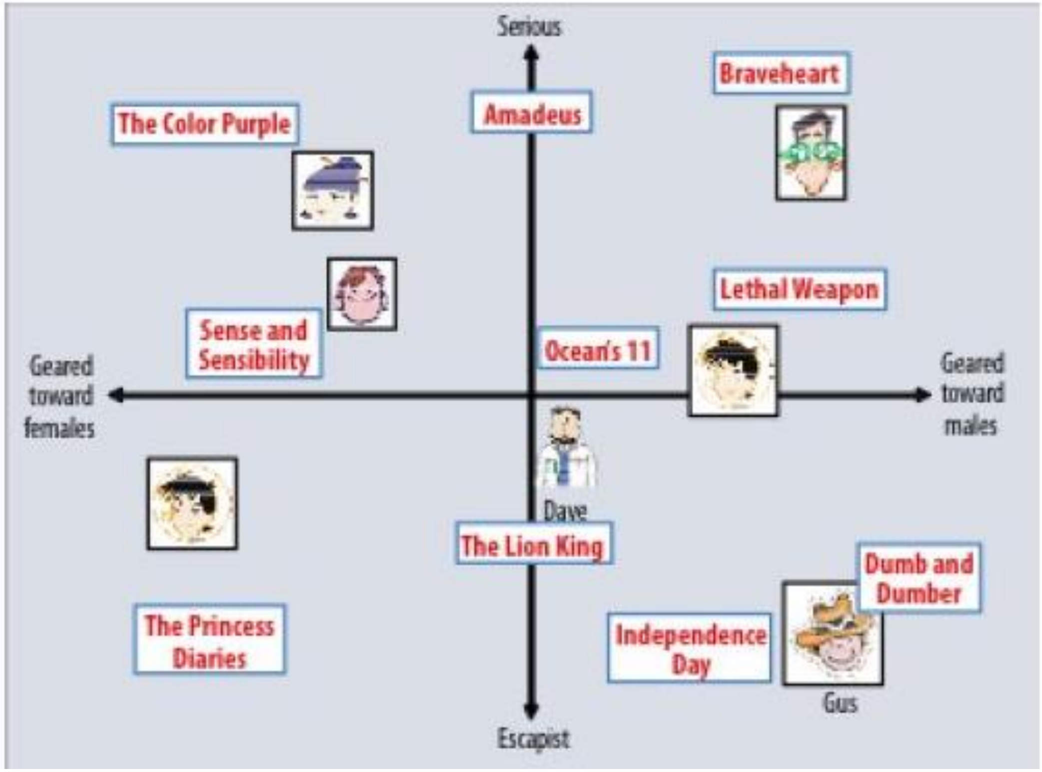
T =

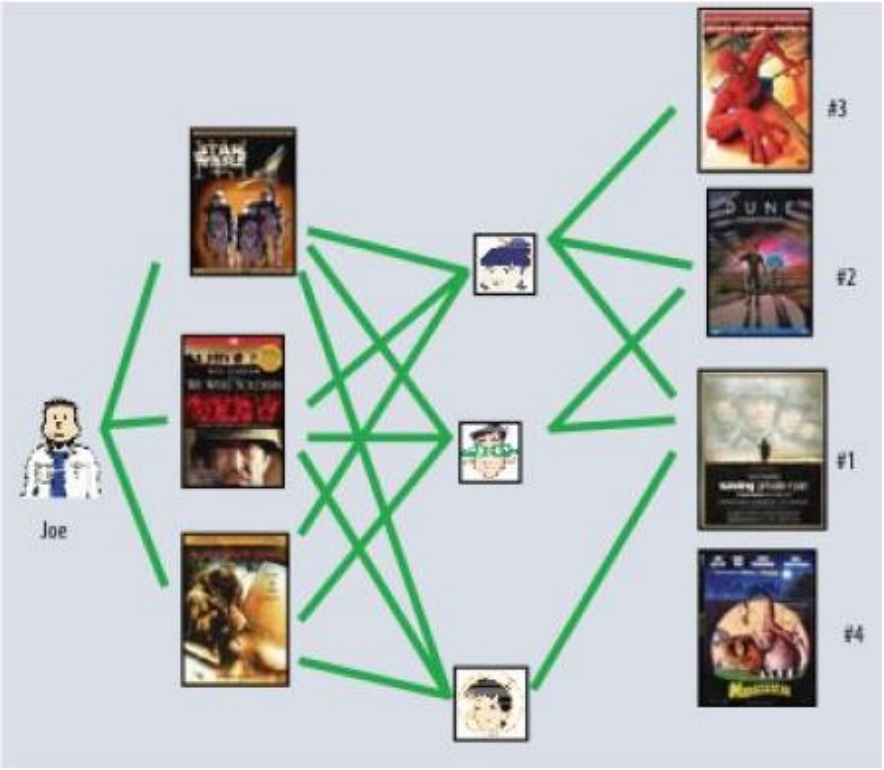
	RedMeat	WhiteMeat	Eggs	Milk	Fish	Cereals	Starch	Nuts	FruitVeg
Albania	10.1	1.4	0.5	8.9	0.2	42.3	0.6	5.5	1.7
Austria	8.9	14	4.3	19.9	2.1	28	3.6	1.3	4.3
Belgium	13.5	9.3	4.1	17.5	4.5	26.6	5.7	2.1	4
Bulgaria	7.8	6	1.6	8.3	1.2	56.7	1.1	3.7	4.2
Czechosl.	9.7	11.4	2.8	12.5	2	34.3	5	1.1	4
Denmark	10.6	10.8	3.7	25	9.9	21.9	4.8	0.7	2.4
EGermany	8.4	11.6	3.7	11.1	5.4	24.6	6.5	0.8	3.6
Finland	9.5	4.9	2.7	33.7	5.8	26.3	5.1	1	1.4
France	18	9.9	3.3	19.5	5.7	28.1	4.8	2.4	6.5
Greece	10.2	3	2.8	17.6	5.9	41.7	2.2	7.8	6.5
Hungary	5.3	12.4	2.9	9.7	0.3	40.1	4	5.4	4.2
Ireland	13.9	10	4.7	25.8	2.2	24	6.2	1.6	2.9
Italy	9	5.1	2.9	13.7	3.4	36.8	2.1	4.3	6.7
Netherl.	9.5	13.6	3.6	23.4	2.5	22.4	4.2	1.8	3.7
Norway	9.4	4.7	2.7	23.3	9.7	23	4.6	1.6	2.7
Poland	6.9	10.2	2.7	19.3	3	36.1	5.9	2	6.6
Portugal	6.2	3.7	1.1	4.9	14.2	27	5.9	4.7	7.9
Romania	6.2	6.3	1.5	11.1	1	49.6	3.1	5.3	2.8
Spain	7.1	3.4	3.1	8.6	7	29.2	5.7	5.9	7.2
Sweden	9.9	7.8	3.5	24.7	7.5	19.5	3.7	1.4	2
Switzerl.	13.1	10.1	3.1	23.8	2.3	25.6	2.8	2.4	4.9
UK	17.4	5.7	4.7	20.6	4.3	24.3	4.7	3.4	3.3
USSR	9.3	4.6	2.1	16.6	3	43.6	6.4	3.4	2.9
WGermany	11.4	12.5	4.1	18.8	3.4	18.6	5.2	1.5	3.8
Yugosl.	4.4	5	1.2	9.5	0.6	55.9	3	5.7	3.2



Matrix Factorization : Example

- Input
  - User-Rating Matrix (Incomplete : Sparse)
- Output
  - A set of new dimensions where both users & items can be represented





PCA Dimensionality Reduction using Singular Value Decomposition

- PCA
  - U contains the principal **components** : directions aligned with variance
  - S contains a measure of how much **variance** is explained by each of the new components
  - V contains the linear combinations of these new components (**loadings**) to recover data in original basis

$$
\begin{aligned}
X &= USV^T \\
&= u_1 s_1 v_1^T + u_2 s_2 v_2^T + \ldots + u_p s_p v_p^T \\
&\approx u_1 s_1 v_1^T + \ldots + u_k s_k v_k^T
\end{aligned}
$$

$$\underbrace{X}_{n \times p} \approx \underbrace{U}_{n \times k}\ \underbrace{S}_{k \times k}\ \underbrace{V^T}_{k \times p}$$

$u_1, u_2, \ldots, u_k$ Orthonormal basis for the columns of X…

$s_1 \geq s_2 \geq \ldots s_k$ …sorted in order of explained variance

- SVD obtains the best low-rank approximation of X
  - k < p : Which k to keep?

$$\min_{rank_{A_k} \leq k} ||A - A_k||_F^2$$

$$||X||_2 = \left(\sum_{i=1}^{n} \sum_{j=1}^{p} |x_{ij}|^2 \right)^{\frac{1}{2}} = ||X||_F$$

Content based Filtering

- Collaborative Filtering Limitations
  - New item → No ratings (Cold Start)
  - Rich get Richer
  - Does not exploit other data e.g. item description, genre, author / manufacturer

- Input
  - User-Rating Matrix (Incomplete : Sparse)
  - Item features (description, synopses etc.)
- Output
  - For a particular user, complete the row

- Key Idea
  - Item-DescriptionWords as the data matrix; Like-or-not as the label (Binary classification)
  - Treat as a supervised learning problem (Model based unlike UBCF and IBCF)
  - Extended features : text description, author, genre, published review, customer comments etc.
  - Recommendation based on what the user has liked in the past (Not UBCF; Not IBCF)

Recommendation Systems : Summary

- A class of problems
  - Where the objective is to recommend “items” to “users”

- Multiple modelling approaches
  - Collaborative Filtering
    - UBCF, IBCF
    - (Dis) similarity measures : Pearson, Cosine
  - Matrix Factorization
    - Latent Factor
  - Content based
    - Use item data (description)
  - Hybrid
    - Use item & user data (description)

CRITIC	TITANIC	BATMAN	INCEPTION	SUPERMAN RETURNS	SPIDERMAN	MATRIX
MICHEL	2.5	3.5	3	3.5	2.5	3
SATYA	3	3.5	1.5	5	3	3.5
PARANAV	2.5	3	N/A	3.5	N/A	4
SURESH	N/A	3.5	3	4	2.5	4.5
TOM	3	4	2	3	2	3
LEO	3	4	N/A	5	3.5	3
CHAN	N/A	4.5	N/A	4	1	N/A

Q?

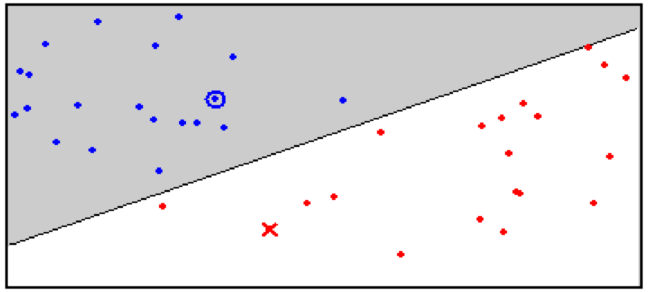
Praphul Chandra

Improving (Speeding up) knn

- Clustering as a pre-processing step
  - Eliminate most points (keep only cluster centroids)
  - Apply knn
- Condensed nn
  - Retain samples closest to "decision boundaries"
  - Decision Boundary Consistent – a subset whose nearest neighbour decision boundary is identical to the boundary of the entire training set
  - Minimum Consistent Set – the smallest subset of the training data that correctly classifies all of the original training data
- Reduced nn
  - Remove a sample if doing so does not cause any incorrect classifications

1. Initialize subset with a single training example
2. Classify all remaining samples using the subset, and transfer any incorrectly classified samples to the subset
3. Return to 2 until no transfers occurred or the subset is full

Cran library: Class



Minimum Consistent Set