Statistical Methods in AI (CSE 471)
Lecture4: Nearest Neighbour Classification

Vineet Gandhi
Centre for Visual Information Technology (CVIT)

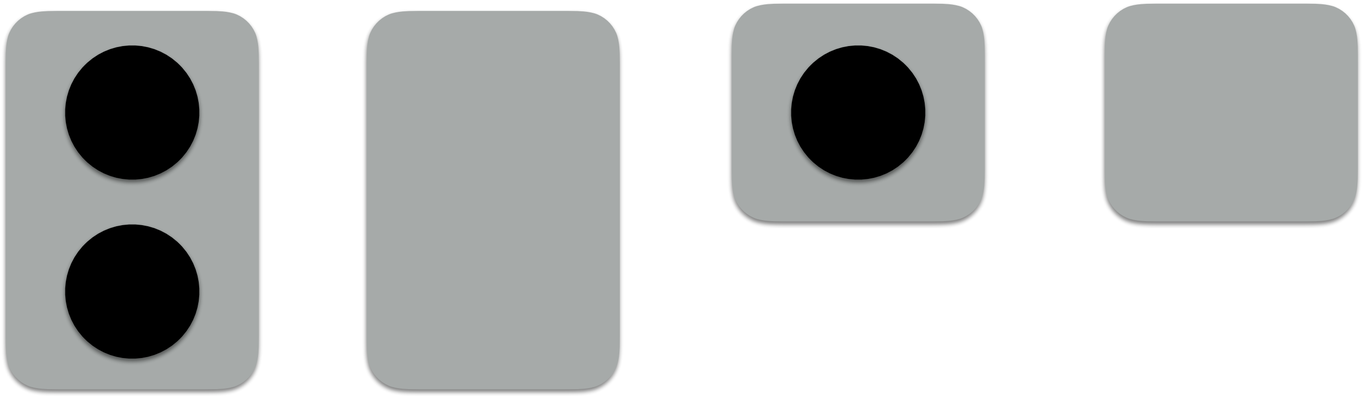
Todays class

- Nearest neighbour method
  - classification and regression
  - practical issues
  - optimality and assumptions
- Making kNN fast
  - Kd trees
  - Inverted indices
  - Locally sensitive hashing

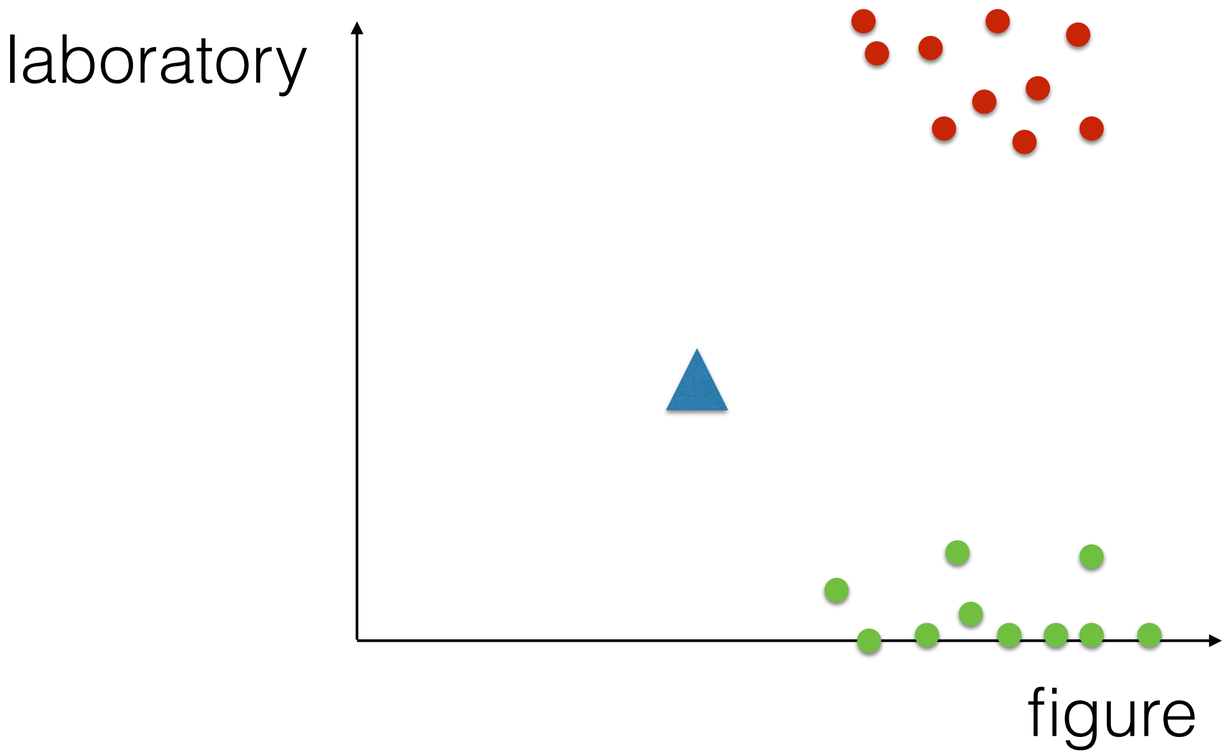
Assumption

- Similar points have similar labels

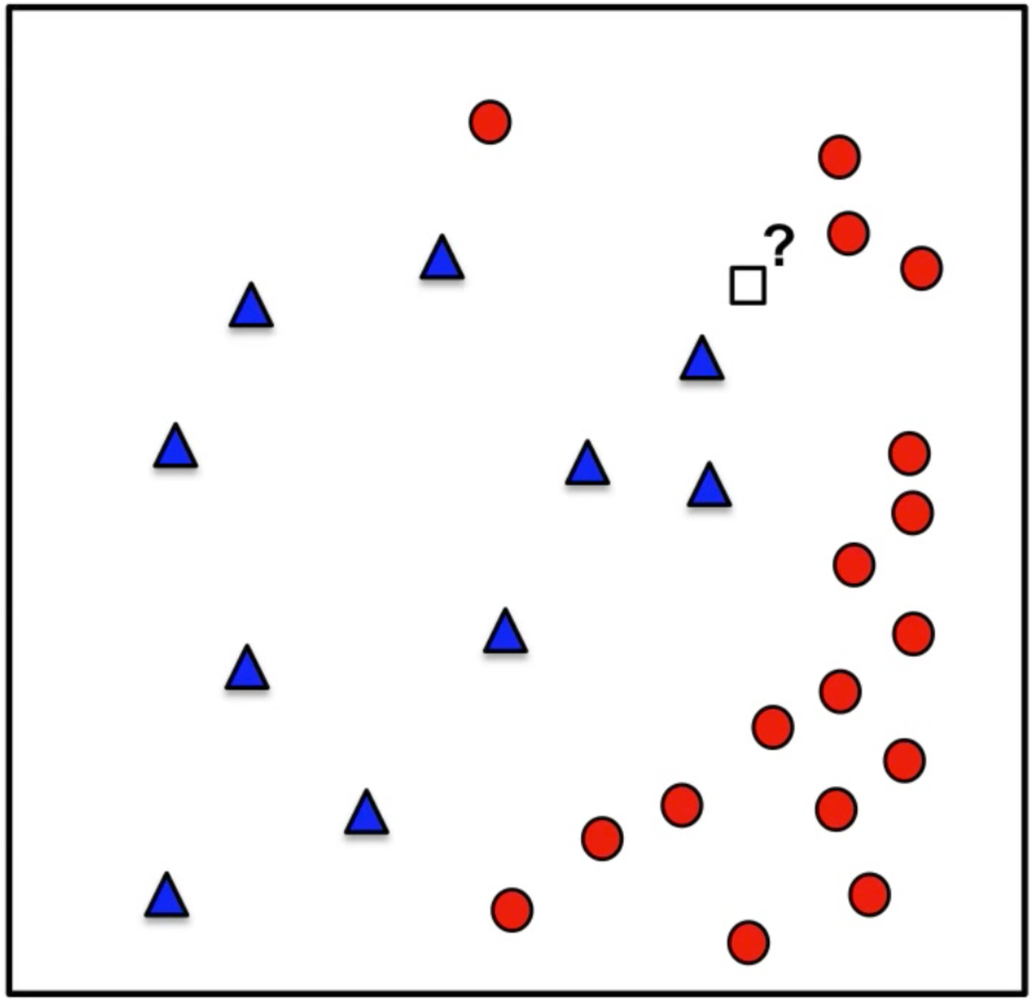
Intuition- sorting example



Intuition - document classification

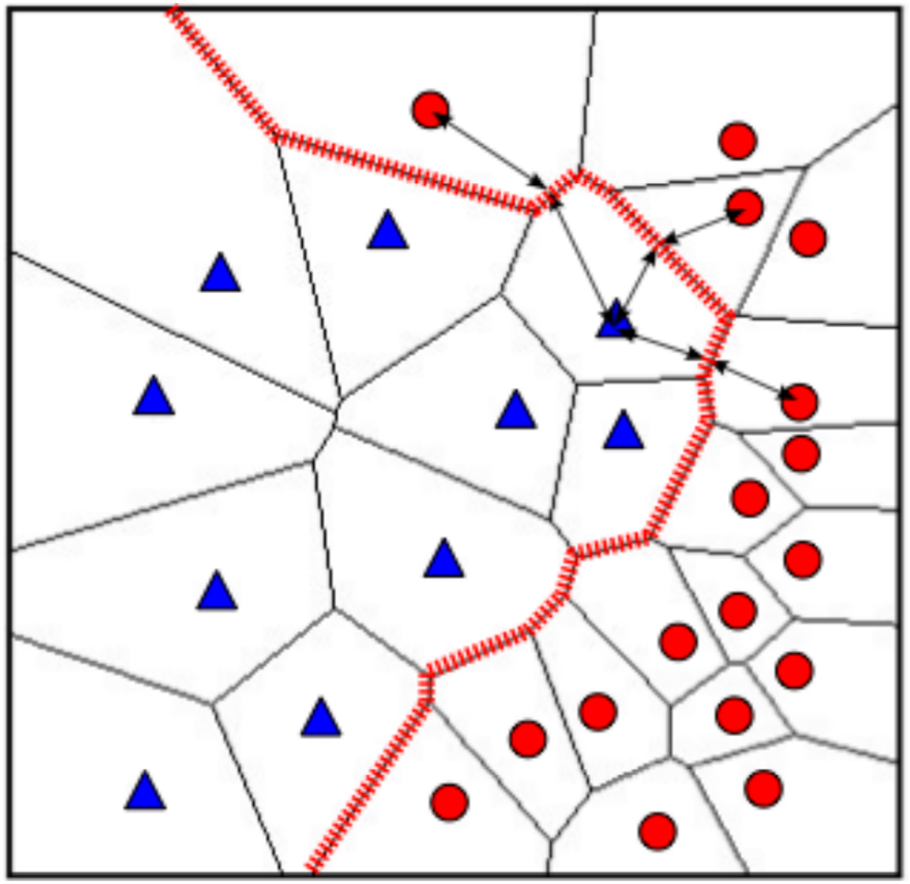


Nearest neighbour classification



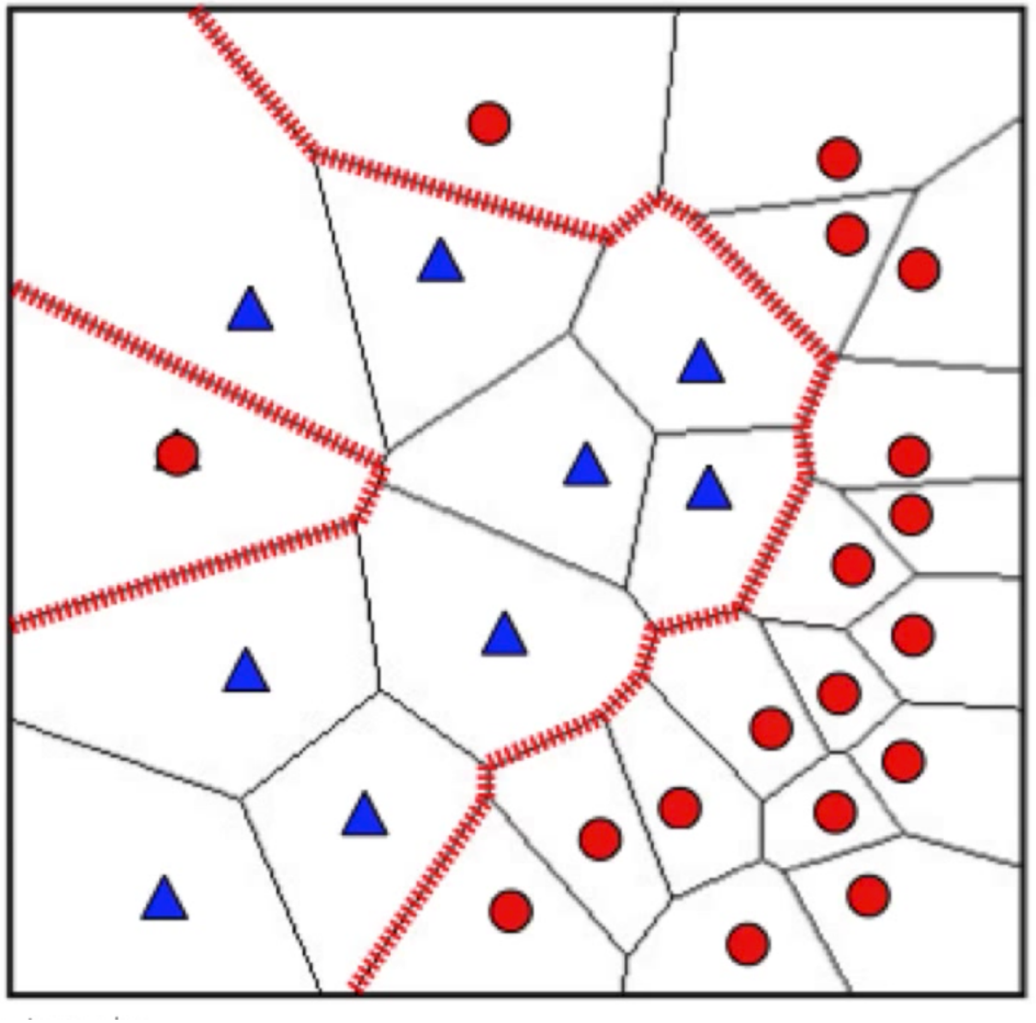
Nearest neighbour classification

- Voronoi tessellation
  - partition space in regions
  - boundary: points at same distance from two different training examples
- Decision boundary
  - non linear
  - compare with other classification approaches
  - impressive for such a simple approach



Issues: outlier

- Sensitive to outliers
  - single mislabeled example can significantly change the boundary
- Idea:
  - use more than one nearest neighbours to make decision



kNN classification

- Given pairs of training examples $\{x_i, y_i\}$, we want to classify a testing point x

- Algorithm
  - Compute distance $D(x_i, x_j)$ to every training example xj
  - Select k closest instances $x_{i1} \ldots\ldots x_{ik}$ and their labels $y_{i1} \ldots\ldots y_{ik}$
  - Output the class y* which is most frequent in $y_{i1} \ldots\ldots y_{ik}$

No training as such

Knn

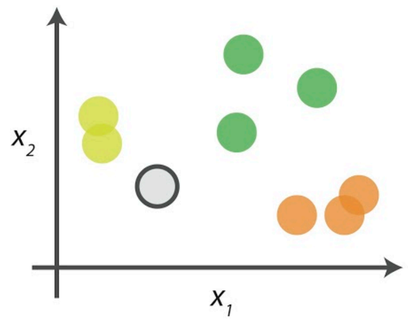
Algorithm (kNN):

1. Find k examples $\{\mathbf{x}^{(i)}, t^{(i)}\}$ closest to the test instance $\mathbf{x}$
2. Classification output is majority class

$$y = arg\max_{t^{(z)}} \sum_{r=1}^{k} \delta(t^{(z)}, t^{(r)})$$

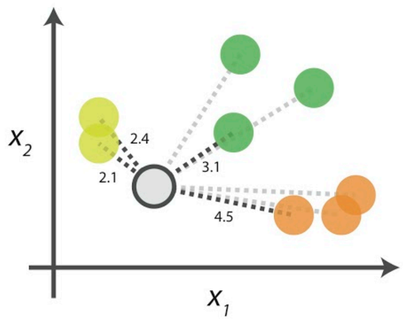
Knn

0. Look at the data



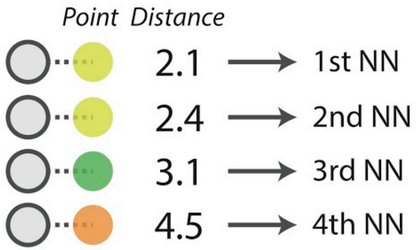
Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

1. Calculate distances



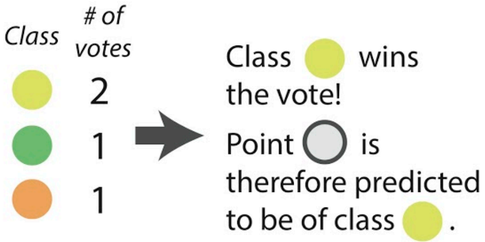
Start by calculating the distances between the grey point and all other points.

2. Find neighbours



Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

3. Vote on labels



Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

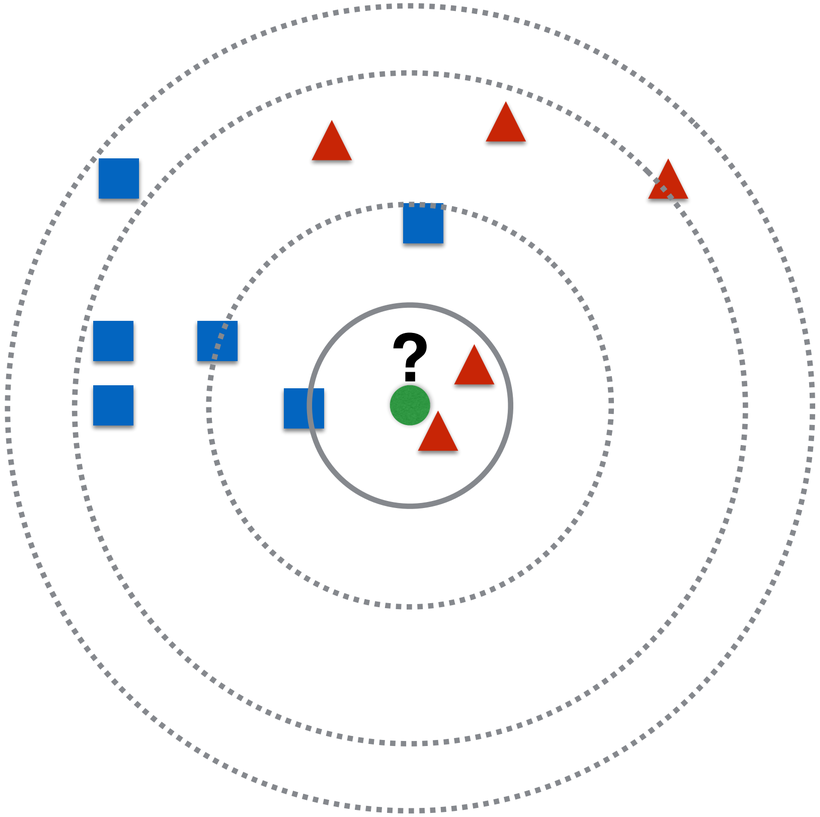
kNN function learning

- Given pairs of training examples $\{x_i, y_i\}$, we want to output $y_i \in R$ for a given input x

- Algorithm
  - Compute distance $D(x_i, x_j)$ to every training example xj
  - Select k closest instances $x_{i1} \ldots\ldots x_{ik}$ and their labels $y_{i1} \ldots\ldots y_{ik}$
  - Output the weighted mean of $y_{i1} \ldots\ldots y_{ik}$

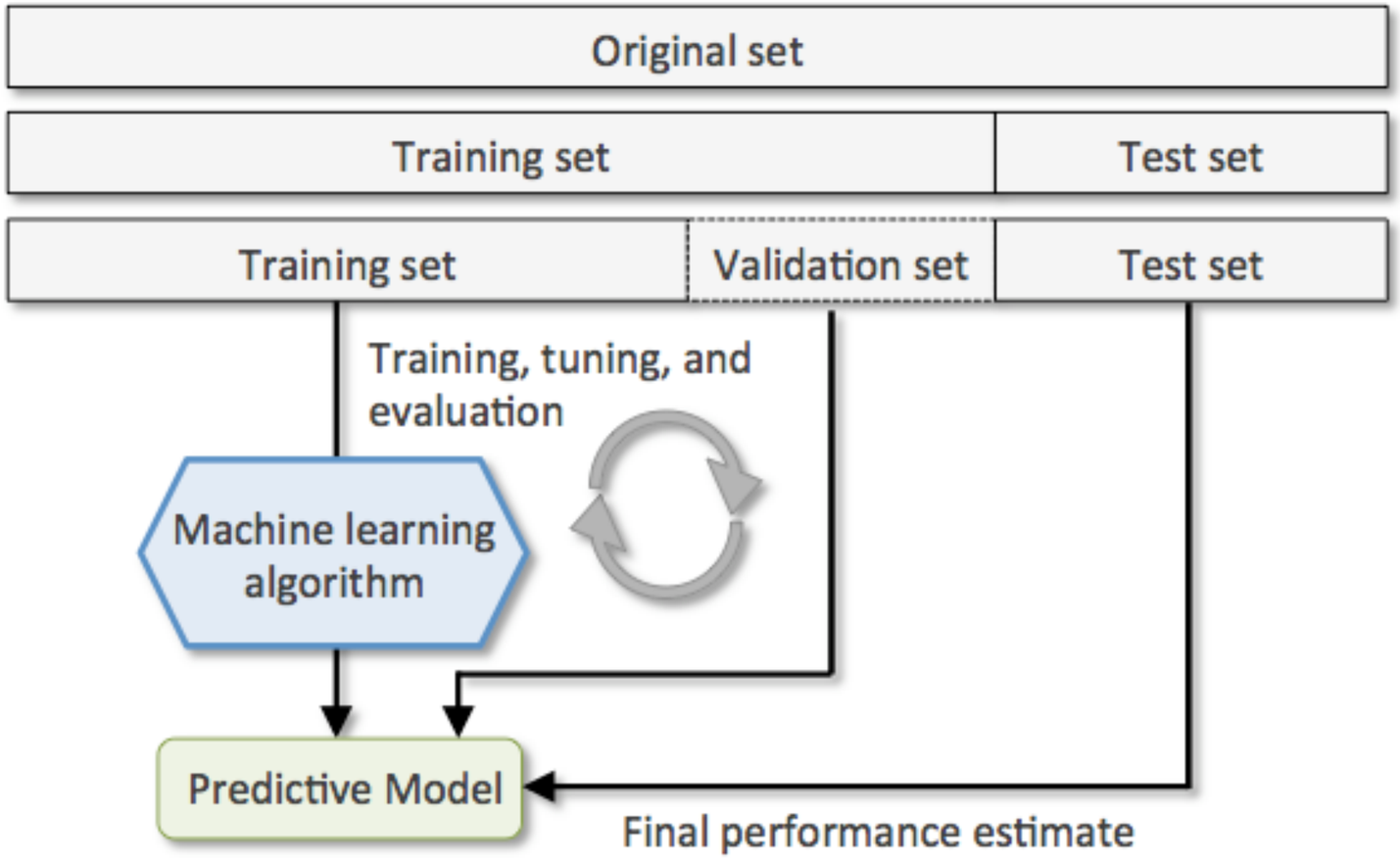
Number of nearest neighbours?

- Value of k has strong effect on kNN performance
- large k -> everything classified as most common class
- small k -> highly variable unstable decision boundaries
  - small changes in training set, large changes in classification
- Set aside a validation set (portion of training set)



How to choose k?

Rule of thumb: $k < \sqrt{n}$, where n is the number of training examples



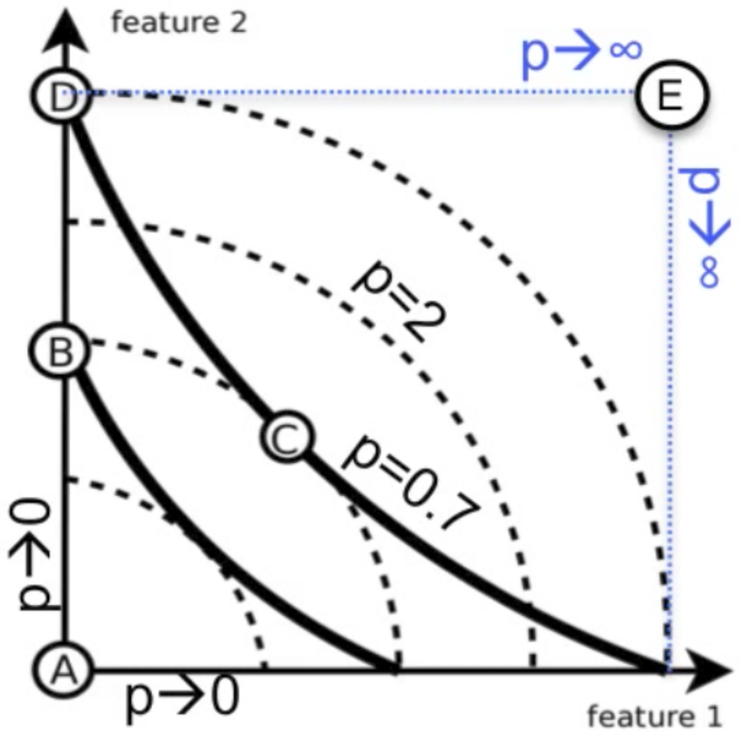
Distance measure?

- Performance highly dependent on distance measure
- Euclidean $D(x, x') = \sqrt[2]{\sum_i (x_i - x'_i)^2}$
  - symmetric, spherical, treat all dimensions equally
  - sensitive to large difference in single attribute
- Better version: $D(x, x') = \sqrt[2]{\sum_i \frac{(x_i - x'_i)^2}{\sigma_i^2}}$
- Even better $D(X, X') = (X - X')^T \sum^{-1} (X - X')$

Distance measure?

- Minkowski norm (p-norm)
  - p=1 —> Manhattan
  - p=2 —> Euclidean
  - p -> ∞
  - p -> 0

$$D(x, x') = \sqrt[p]{\sum_i (x_i - x'_i)^p}$$

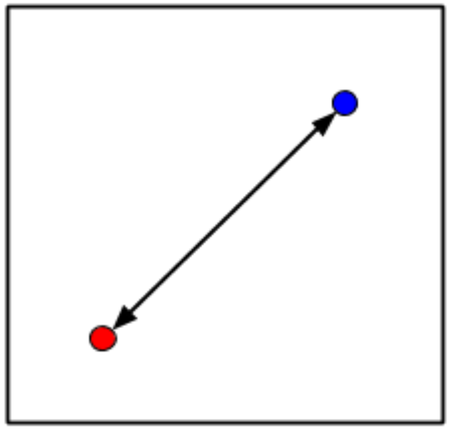


Distance measure?

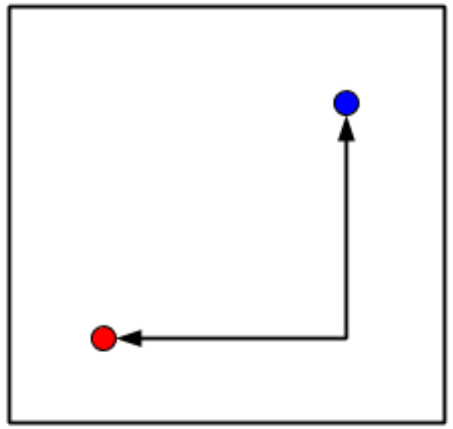
- Cosine distance
- Hamming distance $D(x, x') = \sum_i 1_{x_i \neq x'_i}$
- Kullback-Leibler (KL) divergence $D(x, x') = -\sum_i x_i \log \frac{x_i}{x'_i}$
  - Non symmetric
- Custom distance measures (BM25 for text)

Distance measure?

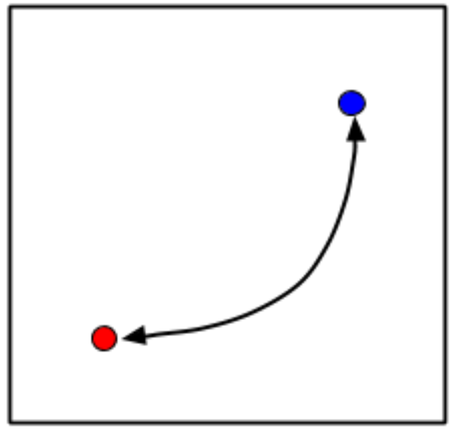
Euclidean



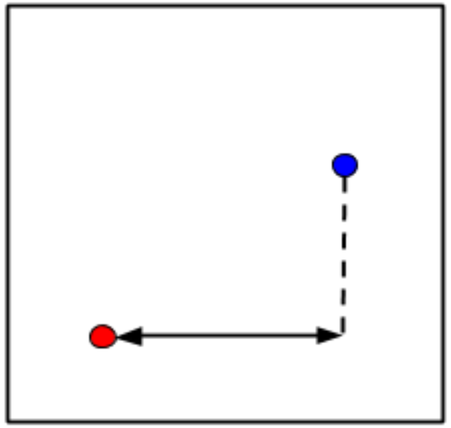
Manhattan



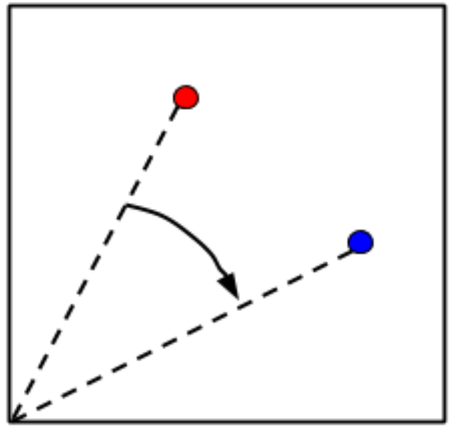
Minkowski



Chebychev



Cosine Similarity



Hamming

Distance measure?

Minkowski distance:

$$d_p = \|x_i - y_i\|_p = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$$

Mahalanobis distance:

$$d_M = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2 / s_i^2}$$

Chebyshev distance:

$$d_\infty = \|x_i - y_i\|_\infty = \max_{i=1..n} |x_i - y_i|$$

Euclidean distance:

$$d_2 = \|x_i - y_i\|_2 = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

Cosine distance:

$$d_{cos} = 1 - \frac{\sum_{i=1}^{n} x_i * y_i}{\sqrt{\sum_{i=1}^{n} (x_i)^2} * \sqrt{\sum_{i=1}^{n} (y_i)^2}}$$

Manhattan:

$$d_1 = \|x_i - y_i\|_1 = \sum_{i=1}^{n} |x_i - y_i|$$

Distance measure?

Minkowski distance:

$$d_p = \|x_i - y_i\|_p = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$$

Mahalanobis distance:

$$d_M = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2 / s_i^2}$$

Hellinger distance:

$$d_H = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{n} |\sqrt{x_i} - \sqrt{y_i}|^2}$$

Chebyshev distance:

$$d_\infty = \|x_i - y_i\|_\infty = \max_{i=1..n} |x_i - y_i|$$

Euclidean distance:

$$d_2 = \|x_i - y_i\|_2 = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

Cosine distance:

$$d_{cos} = 1 - \frac{\sum_{i=1}^{n} x_i * y_i}{\sqrt{\sum_{i=1}^{n} (x_i)^2} * \sqrt{\sum_{i=1}^{n} (y_i)^2}}$$

Manhattan:

$$d_1 = \|x_i - y_i\|_1 = \sum_{i=1}^{n} |x_i - y_i|$$

Bray-Curtis distance:

$$d_{BC} = \sum_{i=1}^{n} |x_i - y_i| \Big/ \sum_{i=1}^{n} |x_i + y_i|$$

Canberra:

$$d_C = \sum_{i=1}^{n} |x_i - y_i| / (|x_i| + |y_i|)$$

Distance based on Pearson correlation:

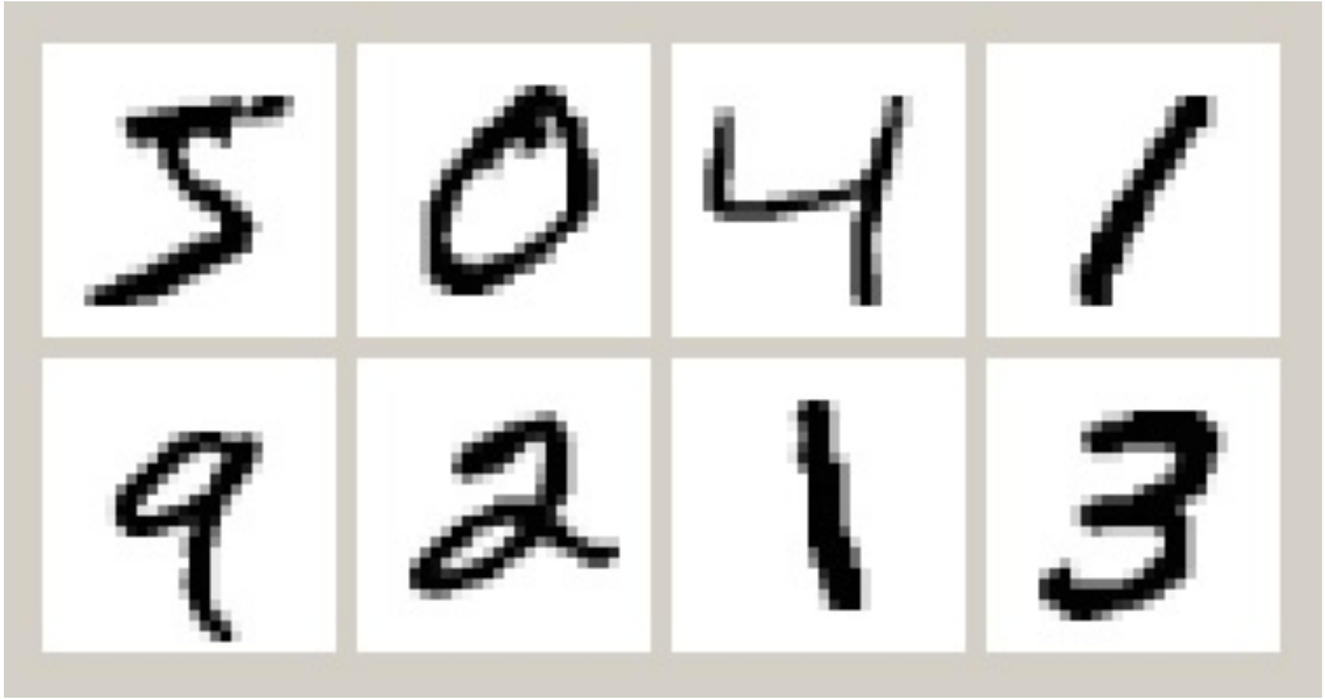
$$d_{corr} = 1 - \frac{\sum_{i=1}^{n} (x_i - \bar{x}) * (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} * \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}$$

kNN practical issues

- Ties?
  - pick odd k (what about multi class?)
  - flip a coin to decide +ve or -ve
  - pick class with greater prior
  - nearest: use 1nn to decide

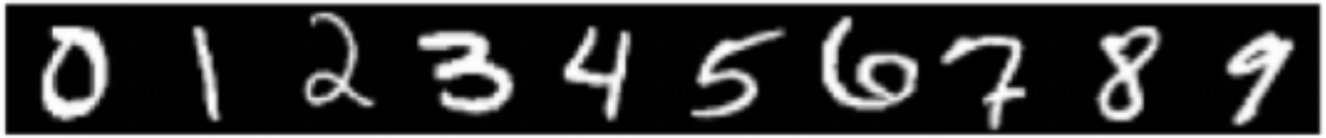
- Missing values
  - have to “fill in” (can’t compute distance otherwise)
  - should effect distance as less as possible
  - average across entire dataset

MNIST dataset



MNIST dataset

- Decent performance when lots of data



- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

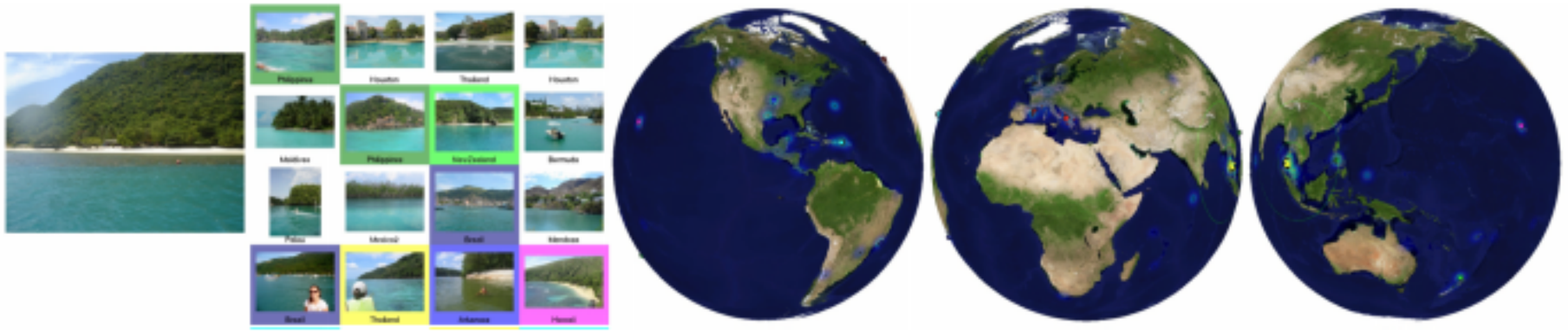
im2gps

- Problem: Where (e.g., which country or GPS location) was this picture taken?

[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

im2gps

- Problem: Where (eg, which country or GPS location) was this picture taken?
  - Get 6M images from Flickr with gps info (dense sampling across world)
  - Represent each image with meaningful features
  - Do kNN (large k better, they use $k = 120$)!



[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

im2text

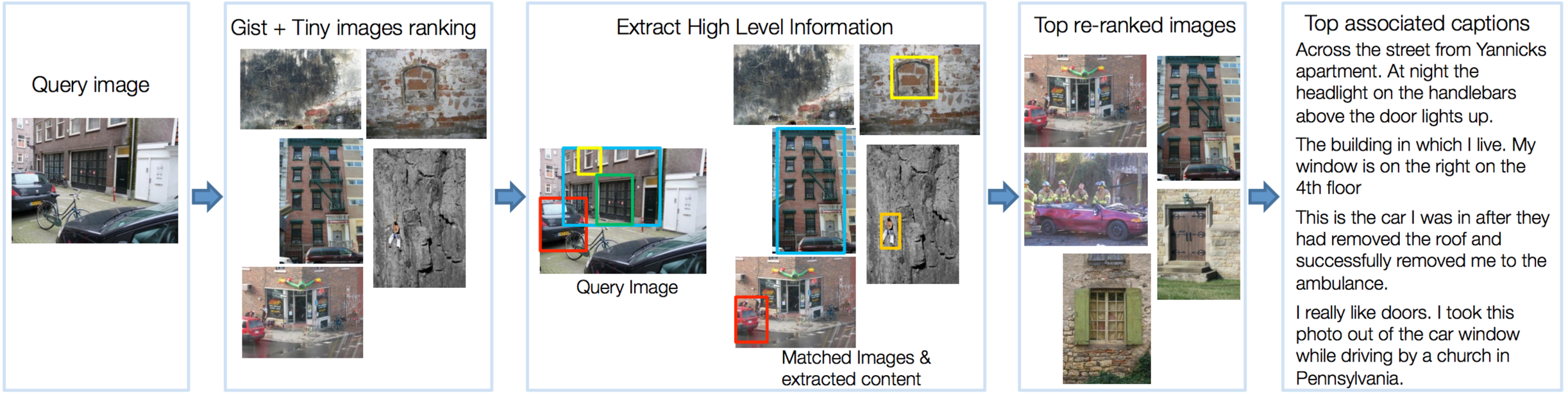
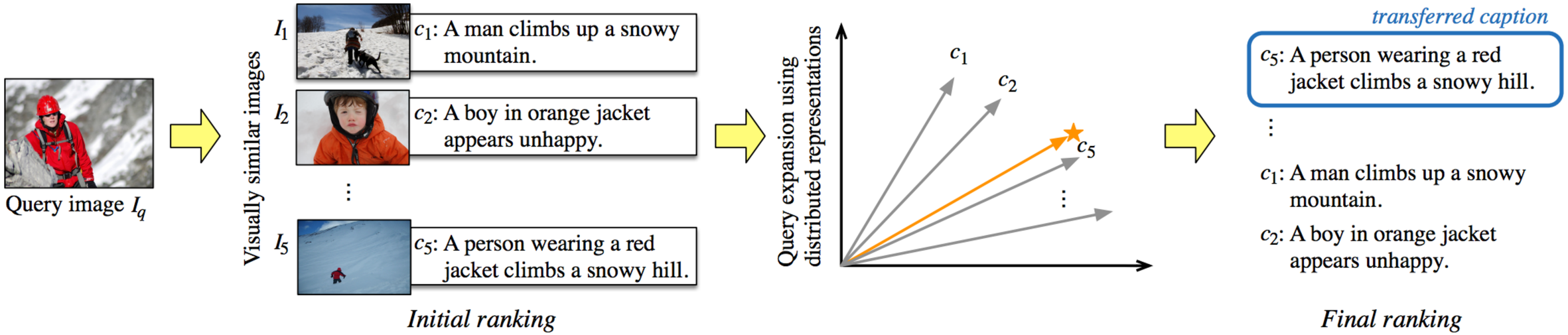


Figure 2: **System flow:** 1) Input query image, 2) Candidate matched images retrieved from our web-scale captioned collection using global image representations, 3) High level information is extracted about image content including objects, attributes, actions, people, stuff, scenes, and tfidf weighting, 4) Images are re-ranked by combining all content estimates, 5) Top 4 resulting captions.

Ordonex et al. Im2text: Describing images using 1 million captioned photographs, NIPS 2011

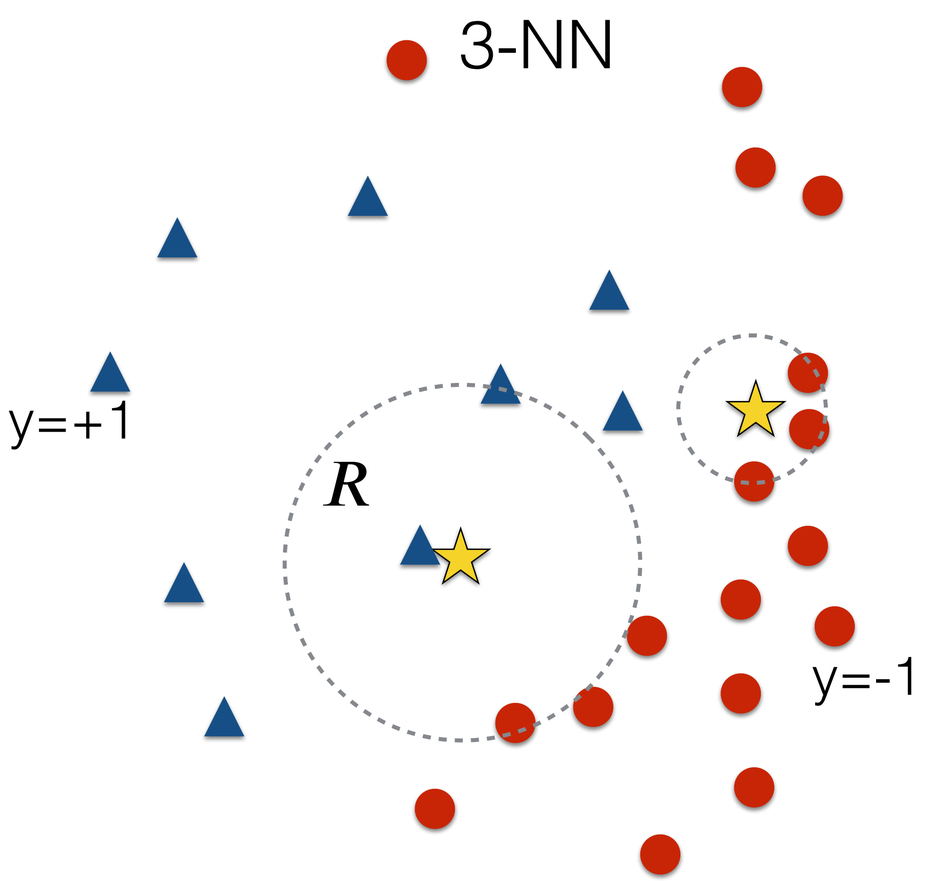
im2text



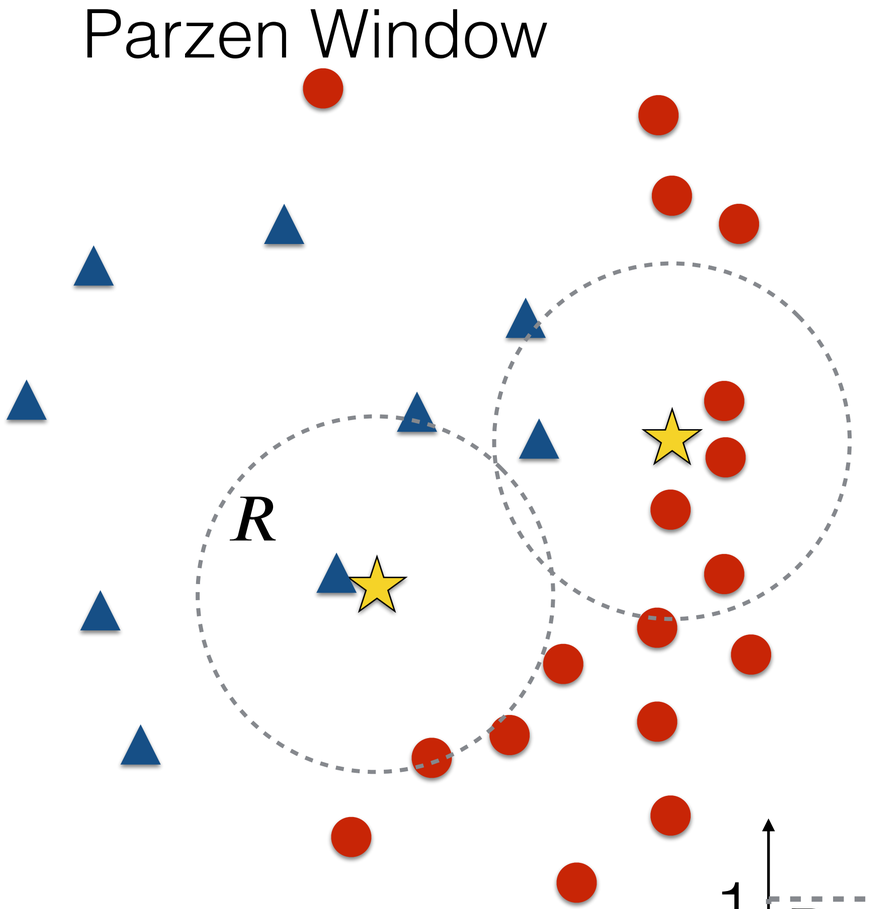
$$q = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} sim(I_q, I_i) \cdot c_i^j$$

Yagcioglu et al., "*A Distributed Representation Based Query Expansion Approach for Image Captioning*", ACL 2015

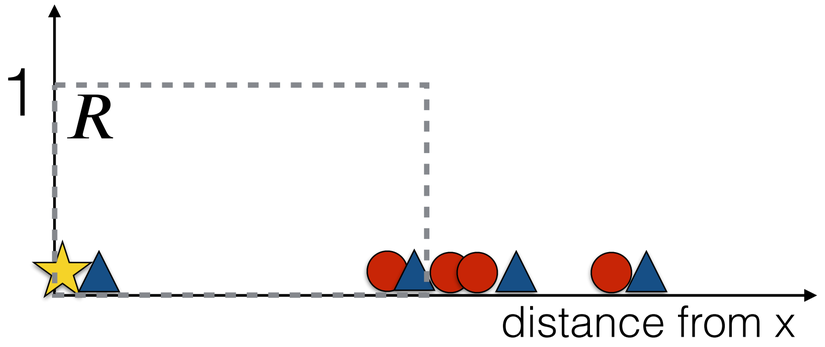
kNN, Parzen Windows and Kernels



$$h(x) = \text{sgn}[\sum_i y_i \cdot 1_{||x_i - x|| \leq R}]$$

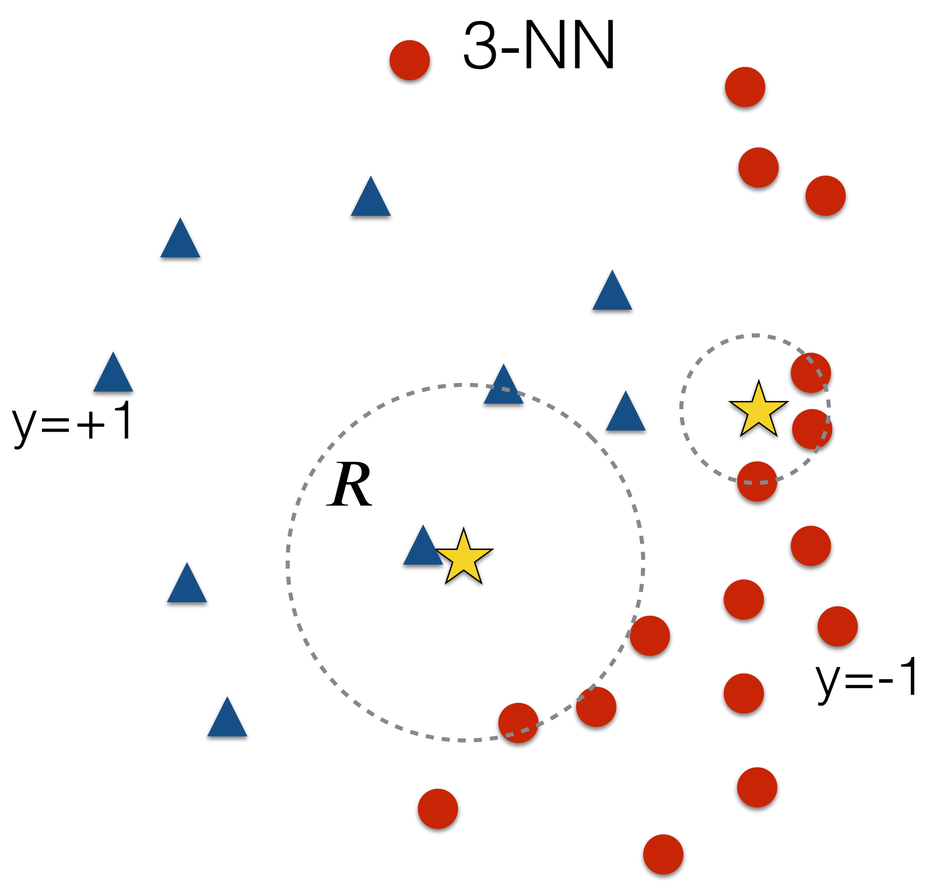


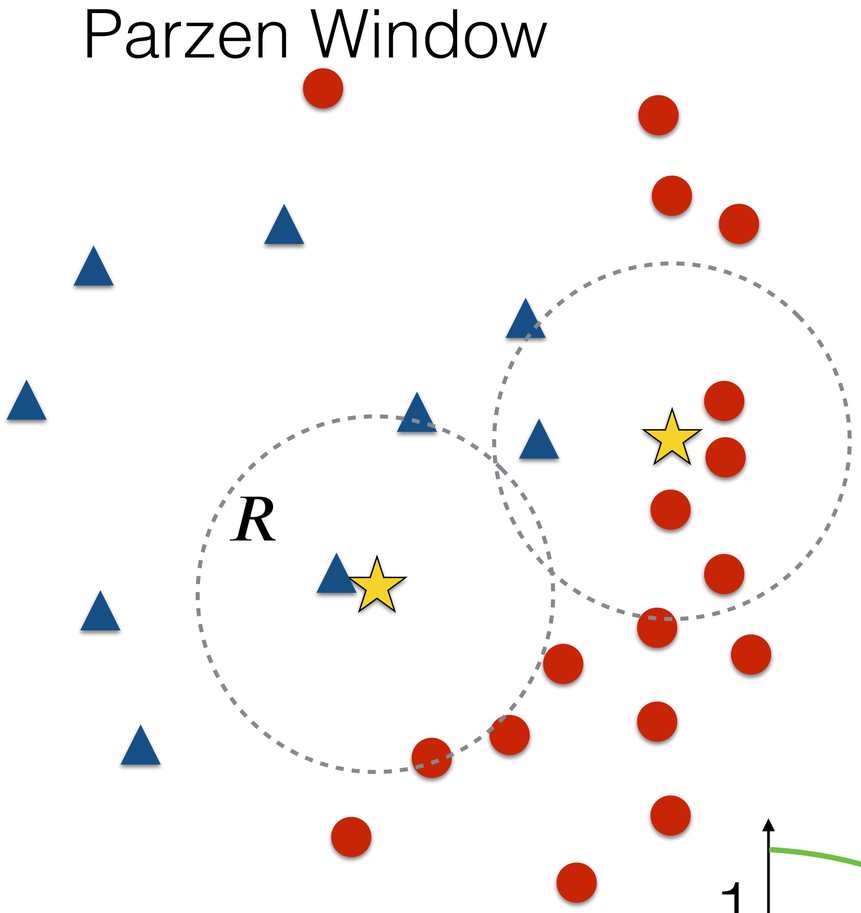
$$h(x) = \text{sgn}[\sum_{i:x_i \in R(x)} y_i]$$



slide credit: Victor Lavrenko

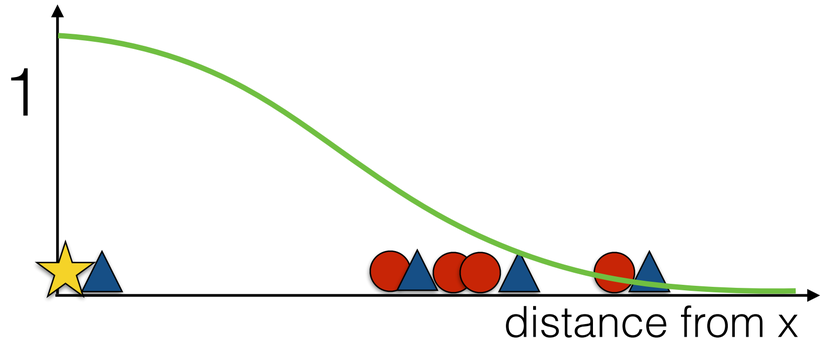
kNN, Parzen Windows and Kernels





$$h(x) = \text{sgn}[\sum_{i:x_i \in R(x)} y_i]$$

$$h(x) = \text{sgn}[\sum_i y_i \cdot 1_{||x_i - x|| \leq R}] = \text{sgn}[\sum_i y_i \cdot K(x_i, x)]$$



slide credit: Victor Lavrenko

kNN Pros and Cons

- Pros
  - No assumption about the data
  - Non parametric (let the data speak for itself)
  - Only two choices are k and D (should be guided by the kind of data)
  - Easy to update in online setting

- Cons
  - Need to handle missing values
  - Sensitive to outliers
  - Computationally expensive (while testing) —> O(nd)
    - n: number of examples, d: cost of computing distance
    - The system will become slower and slower as n grows

Making kNN fast

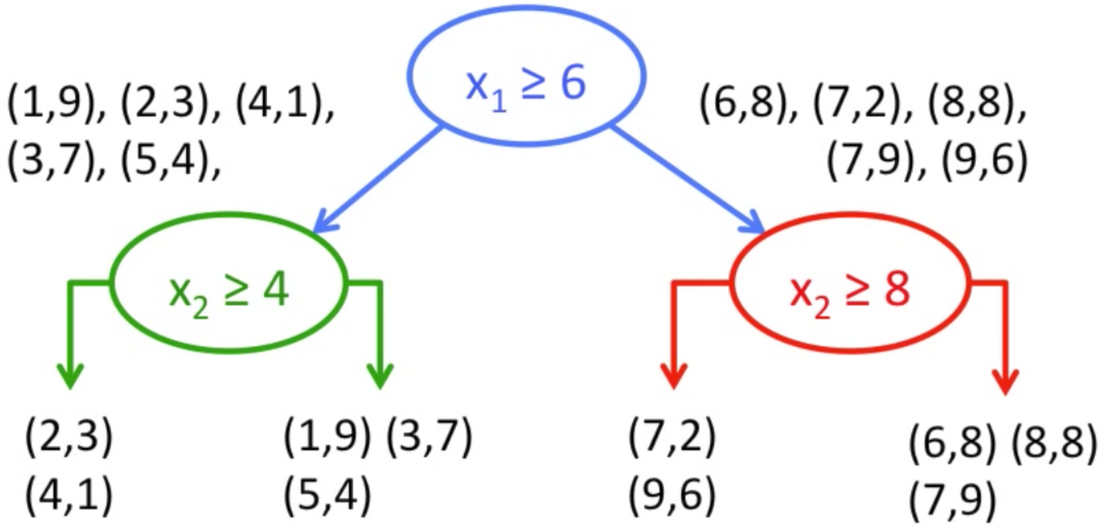
- What algorithm sees
  - {(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)}
  - Testing instance (7,4)

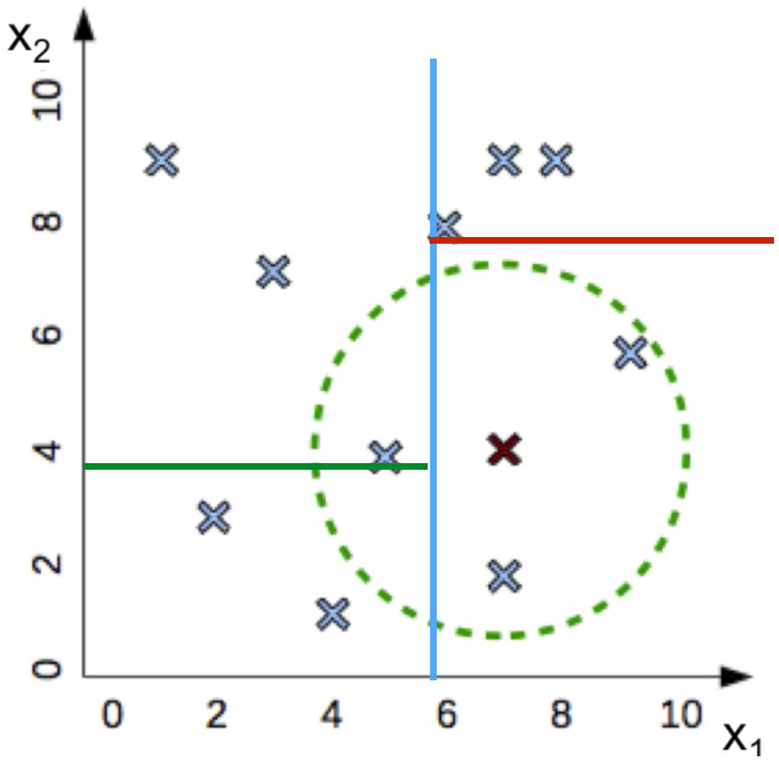
- Ideas:
  - Reduce d : dimensionality reduction
  - Reduce n : don’t compare with all training examples
    - kd trees $O(d \log_2 n)$
    - locally sensitive hashing
    - inverted indices

Making kNN fast (low dimensional data)

- Building kd tree
  - pick random dimension, find median, split data, repeat

{(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)}



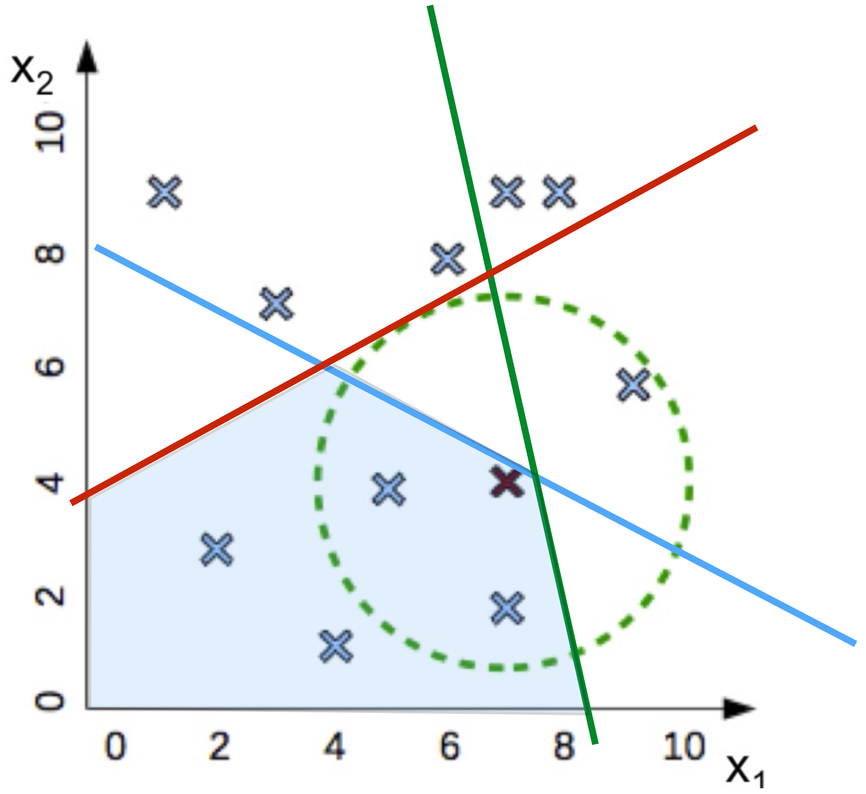


$N >> 2^d$

slide credit: Victor Lavrenko

Making kNN fast (high dimensional data)

- Locally sensitive hashing (LSH)
- Random hyper-planes $h_1,\ldots,h_k$.
  - space sliced into 2^k regions
- Complexity $O(kd + dn/2^k)$

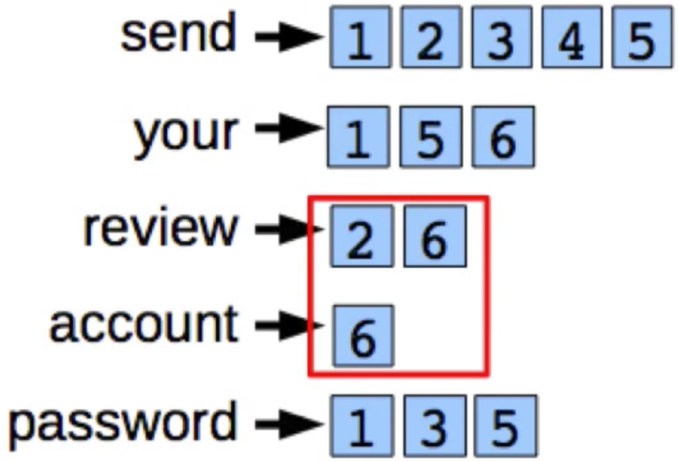


Making kNN fast (high dimensional +sparse data)

- Data structure used by search engines (Google etc.)

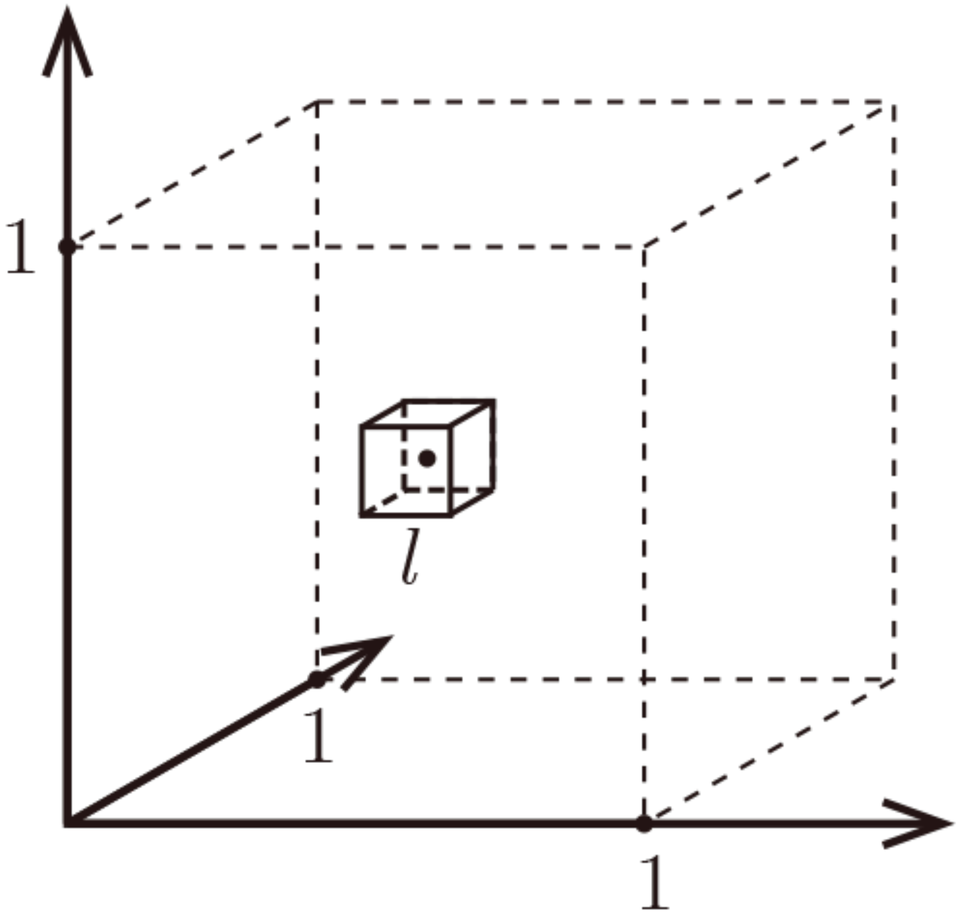
new email: “account review”

D1: “send your password”	spam
D2: “send us review”	ham
D3: “send us password”	spam
D4: “send us details”	ham
D5: “send your password”	spam
D6: “review your account”	ham



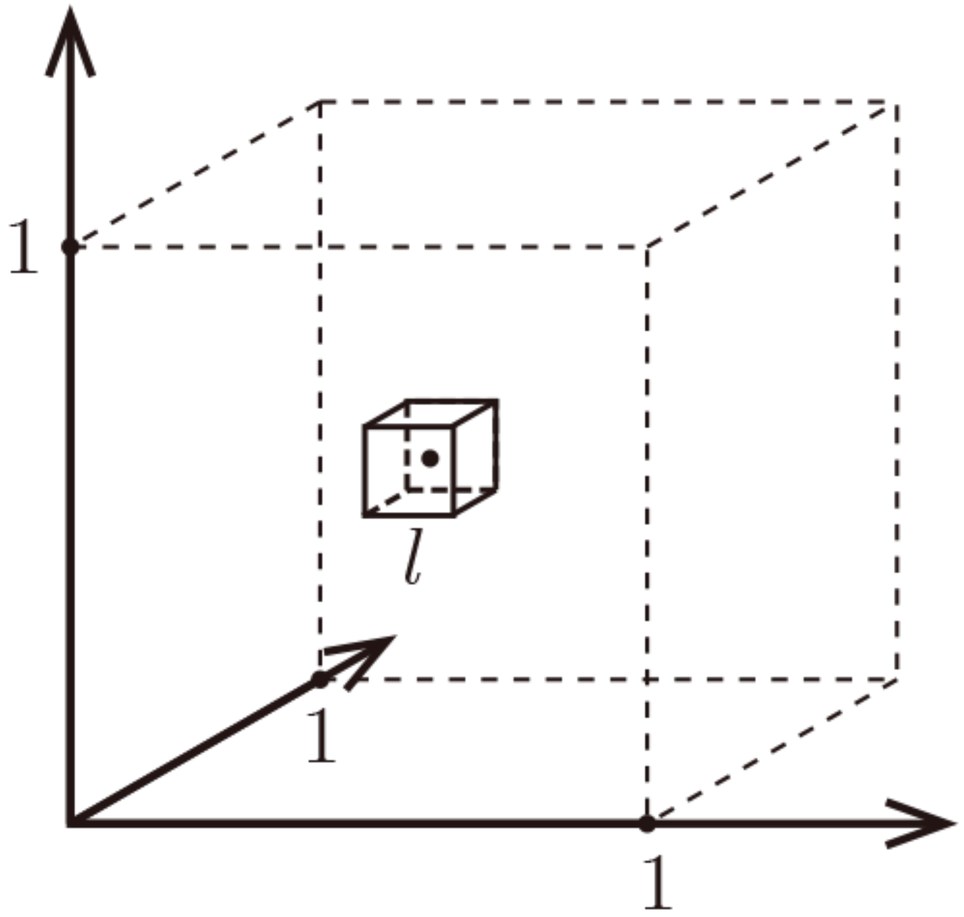
slide credit: Victor Lavrenko

The curse of dimensionality



- The kNN classifier makes the assumption that similar points share similar labels.
- Unfortunately, in high dimensional spaces, points that are drawn from a probability distribution, tend to never be close together.

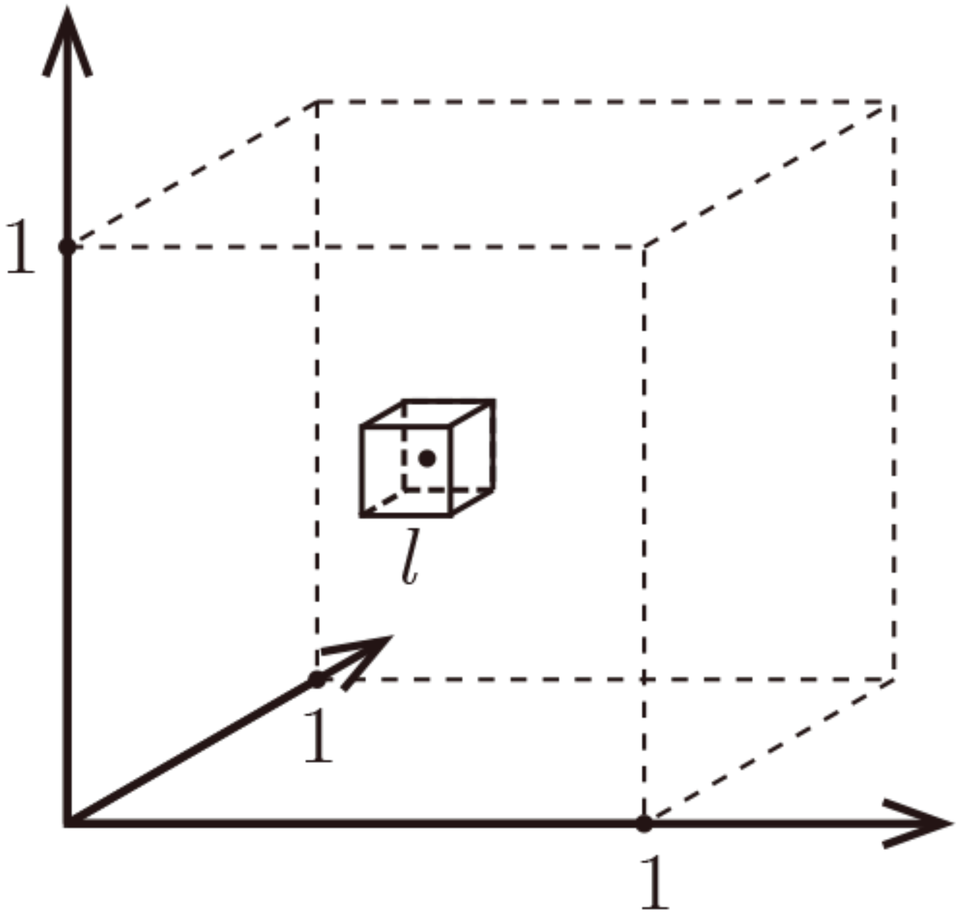
The curse of dimensionality



Let ℓ be the edge length of the smallest hyper-cube that contains all k-nearest neighbor of a test point.

$$\ell^d \approx \frac{k}{n} \text{ and } \ell \approx \left(\frac{k}{n}\right)^{1/d}.$$

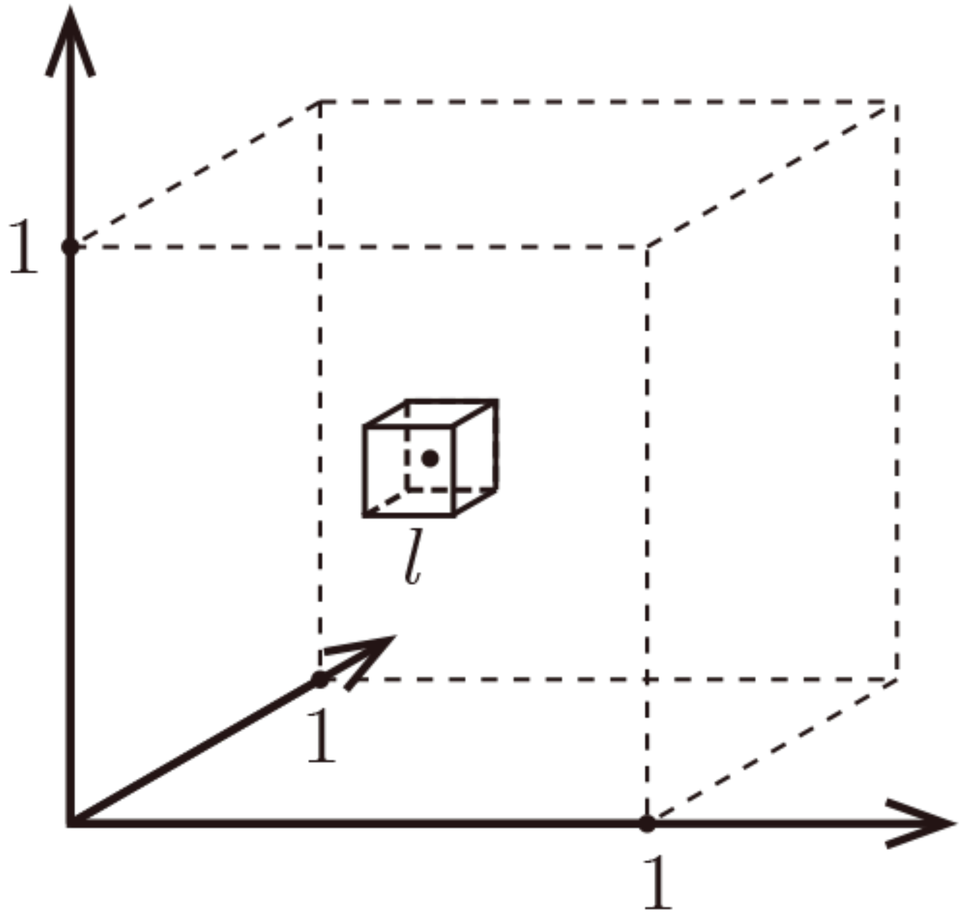
The curse of dimensionality



$$\ell^d \approx \frac{k}{n} \text{ and } \ell \approx \left(\frac{k}{n}\right)^{1/d}.$$

d	ℓ
2	0.1
10	0.63
100	0.955
1000	0.9954

The curse of dimensionality



- So as $d \gg 0$ almost the entire space is needed to find the 10-NN
- This breaks down the k-NN assumptions, because the k-NN are not particularly closer (and therefore more similar) than any other data points in the training set.

Manifold to rescue