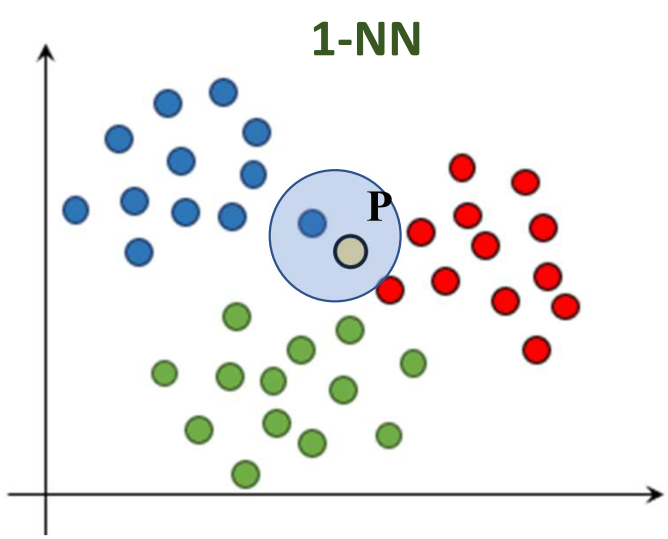
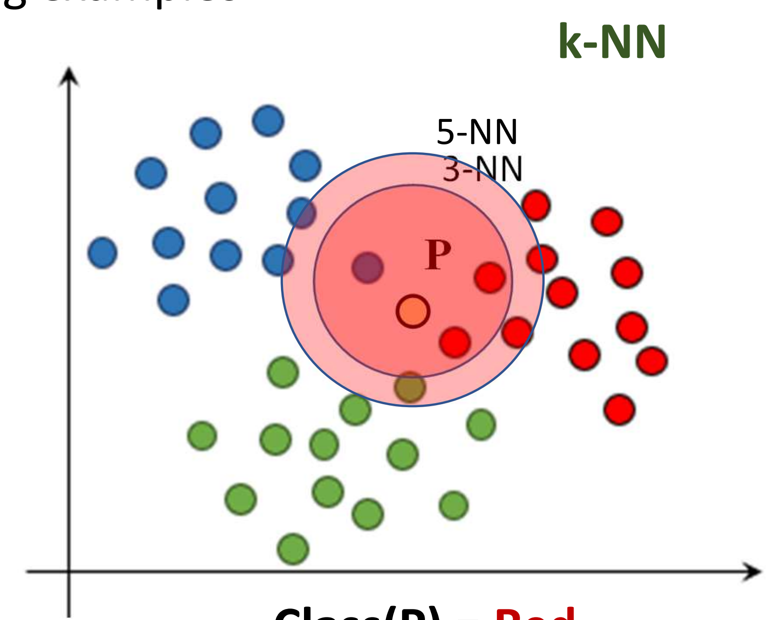
Revisiting Nearest Neighbors

k-Nearest Neighbour

- Given training data $\mathcal{D} = \{(\boldsymbol{x_1}, \boldsymbol{y_1}), \dots, (\boldsymbol{x_N}, \boldsymbol{y_N})\}$ and a test point
- Prediction: Look at the **k** most similar training examples



Class(P) = Blue

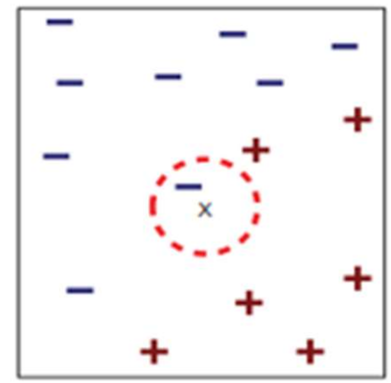


Class(P) = Red

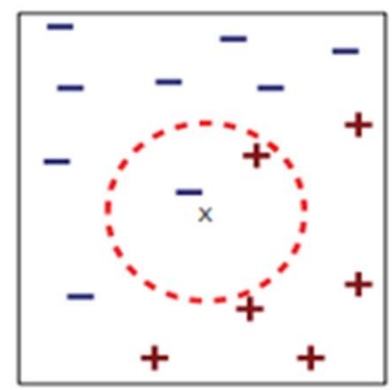
Class(P): Majority label among kNN

The k-NN Algorithm

- Amongst the simplest of all machine learning algorithms. No explicit training required to learn mapping function
- Compute the test point's distance from each training point
- Sort the distances in ascending (or descending) order
- Use the sorted distances to select the k nearest neighbors
- Use the majority rule (for classification) or averaging (for regression)



(a) 1-nearest neighbor



(b) 2-nearest neighbor

Components of a k-NN Classifier

- The k hyperparameter

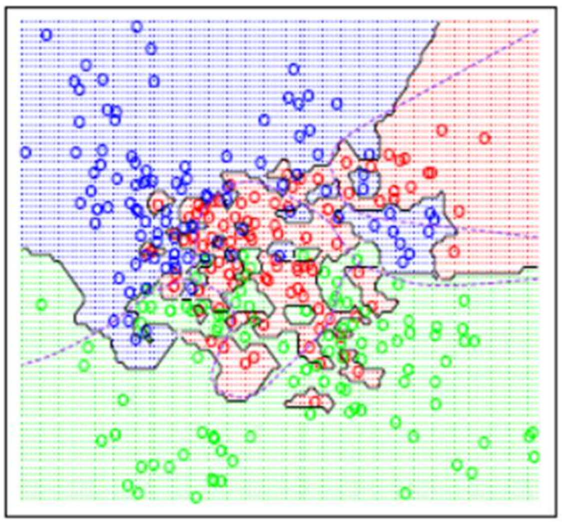
  ✓ Determines how large a neighbourhood should we consider?
  ✓ Decides the complexity of the hypothesis space
    - If k =1, every training example has its own neighbourhood
    - If k =N, the entire feature space is one neighbourhood

- Distance Metric

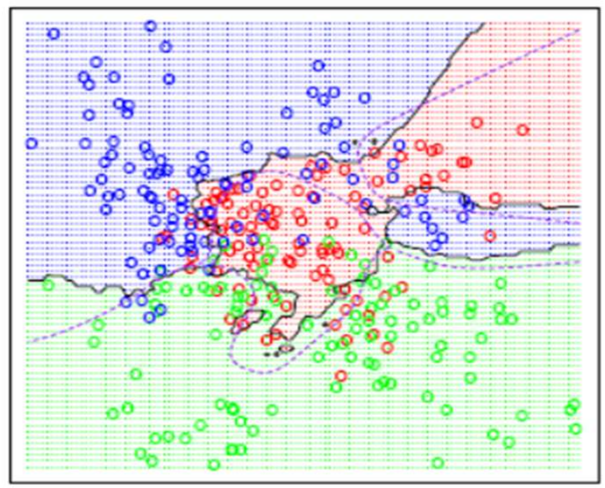
  ✓ How do we measure distance between instances?
  ✓ Determines the layout of the example space

Decision Boundary of the Classifier

- The decision boundary is the line that separates the classes in the feature space
- It helps to visualize how examples will be classified for the entire feature space
- It helps to see the complexity of the learned model
- The more examples that are stored, the more complex the decision boundaries can become



1-NN

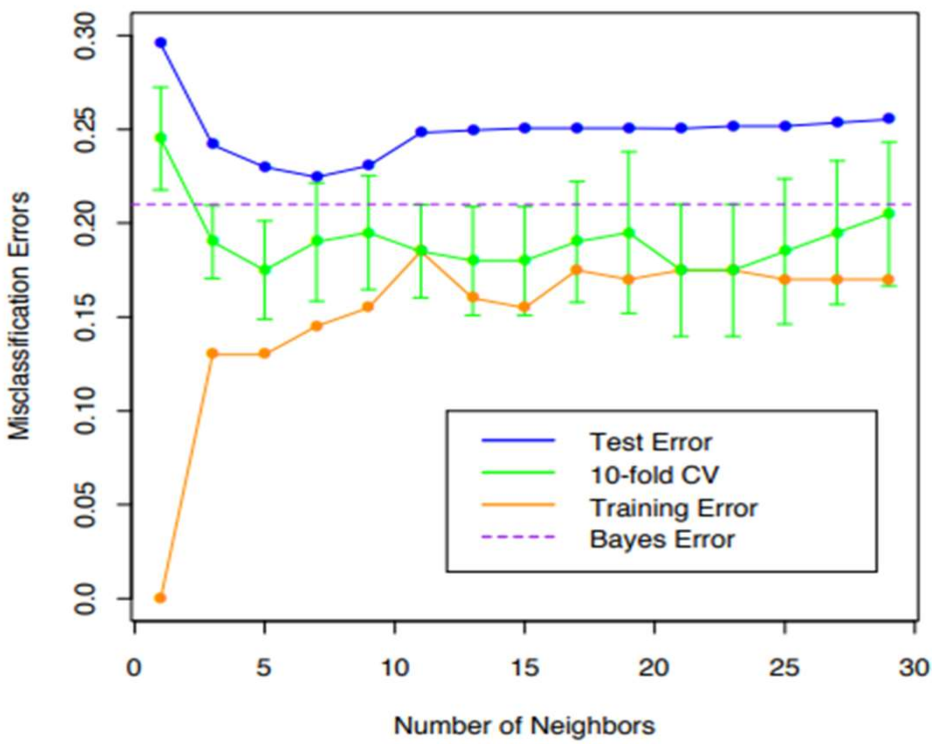


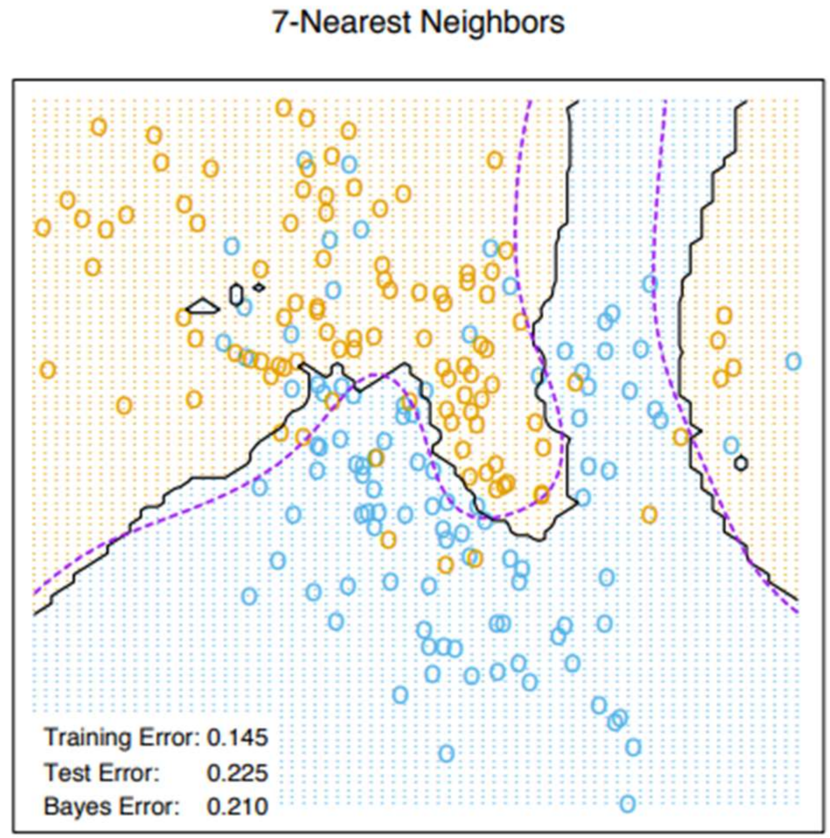
k-NN – Smoother more continuous decision boundaries

Choice of k – Neighbourhood Size

- Small k
  - Creates many small regions for each class
  - May lead to non-smooth decision boundaries and overfit
- Large k
  - Creates fewer larger regions
  - Usually leads to smoother decision boundaries (caution: too smooth decision boundary can underfit)
- Choosing k
  - Often data dependent and heuristic based
  - Use cross-validation (More on next class)
  - Remember in general, **a k too small or too big is bad!**

Example results for k-NN





[Figures from Hastie and Tibshirani, Chapter 13]

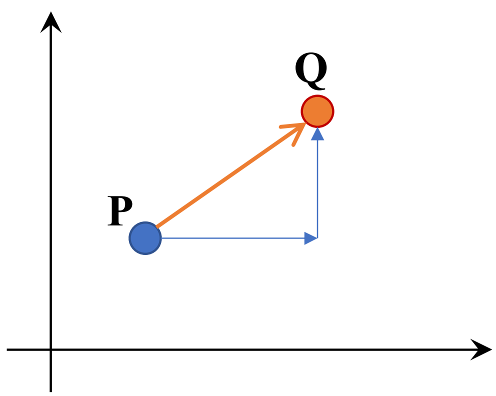
K-NN: Computing the distances

- The k-NN algorithm requires computing distances of the test example each of the training examples
- Several ways to compute distances
- The choice depends on the type of the features in the data
- Euclidean distance commonly used in case of real-valued features

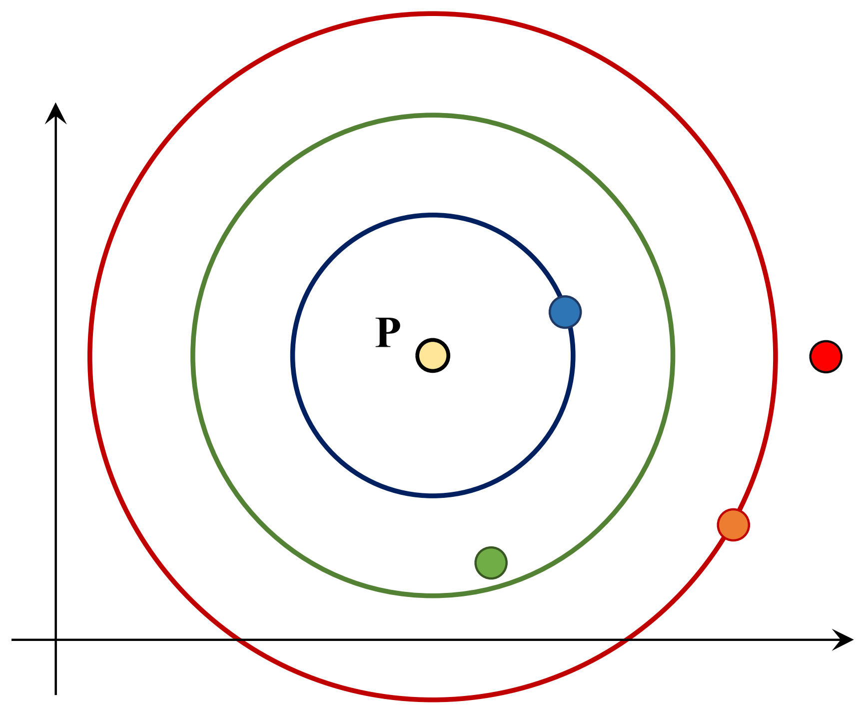
Euclidean Distance [L2]

$$d(P,Q)^2 = \sum_{i=1}^{d} (p_i - q_i)^2$$

$$d(P,Q)^2 = (P-Q)^T(P-Q)$$

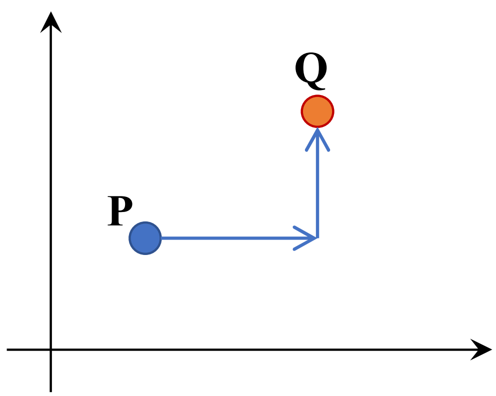


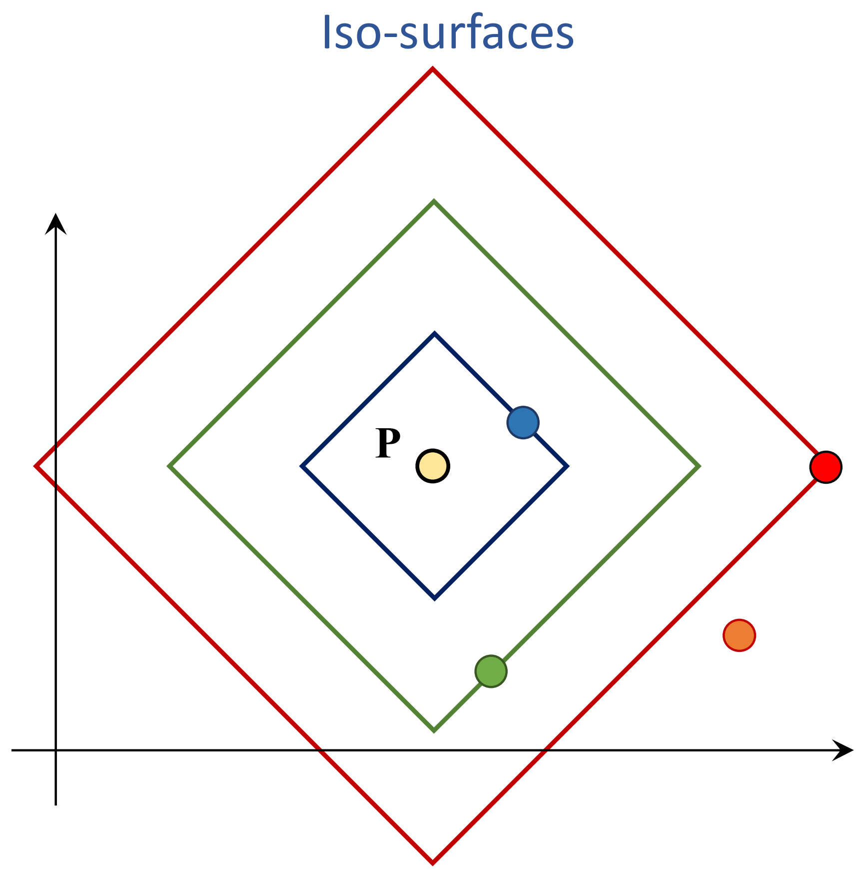
Iso-surfaces



Manhattan Distance [L1]

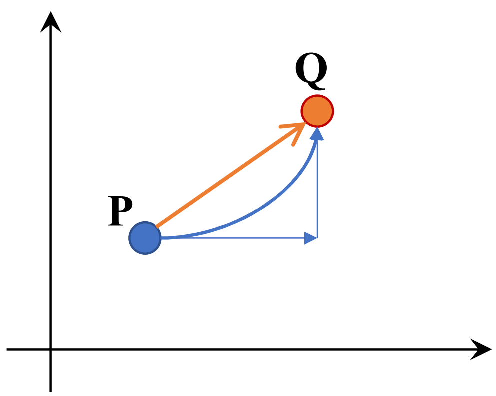
$$d(P,Q) = \sum_{i=1}^{d} |(p_i - q_i)|$$





Minkowski Distance

$$d(P,Q)^r = \sum_{i=1}^{d} |p_i - q_i|^r$$



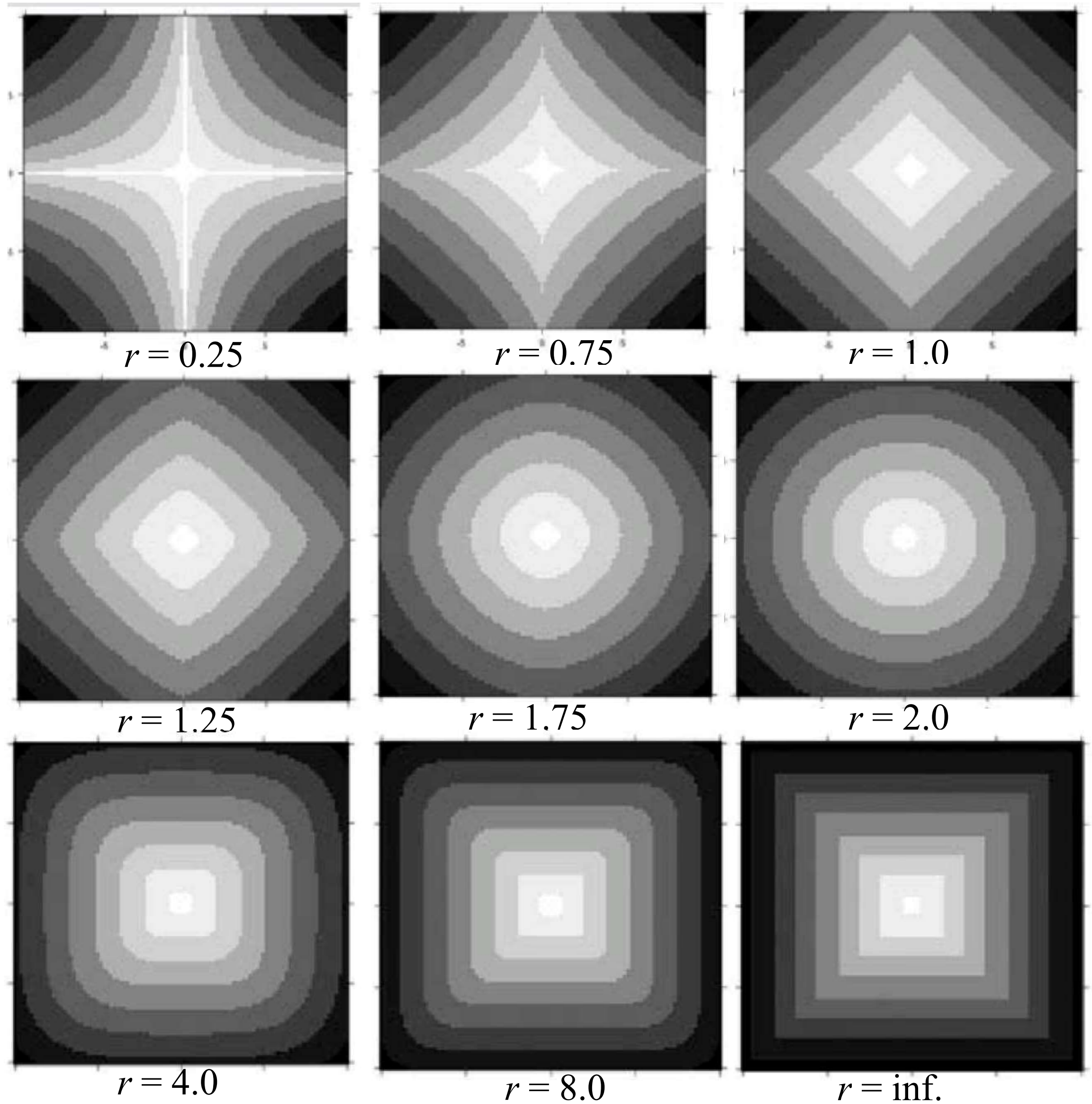
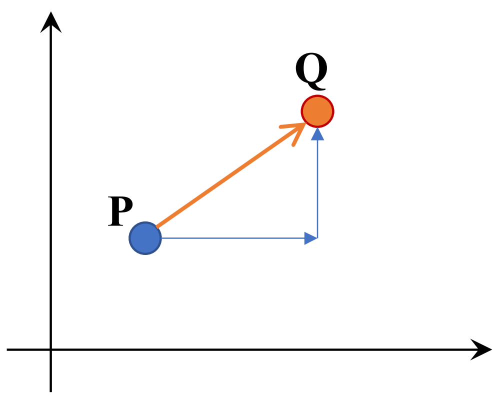
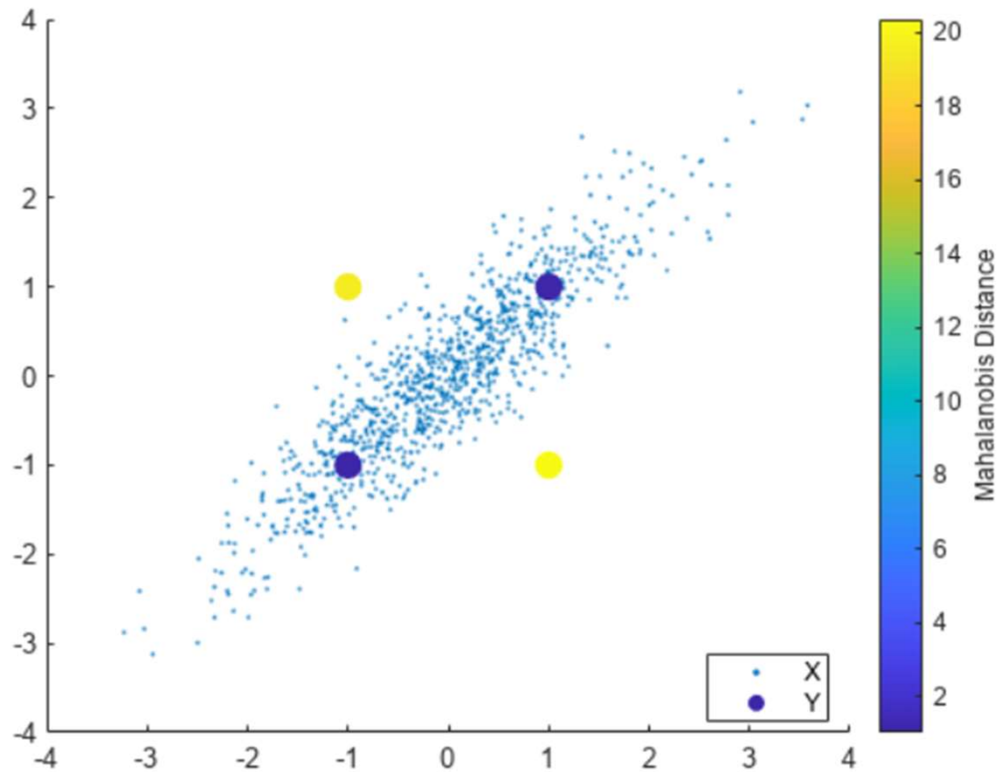


Fig. by Lu et al., "The Minkowski Approach for Choosing the Distance Metric in Geographically Weighted Regression"

Mahalanobis Distance

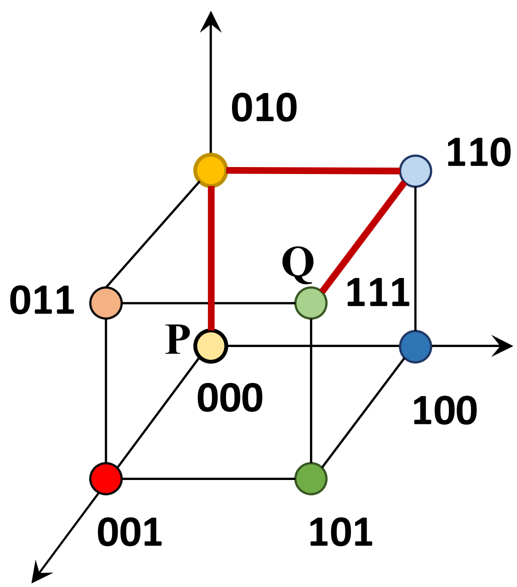
$$d(P,Q)^2 = (P-Q)^T \mathbf{S}^{-1}(P-Q)$$

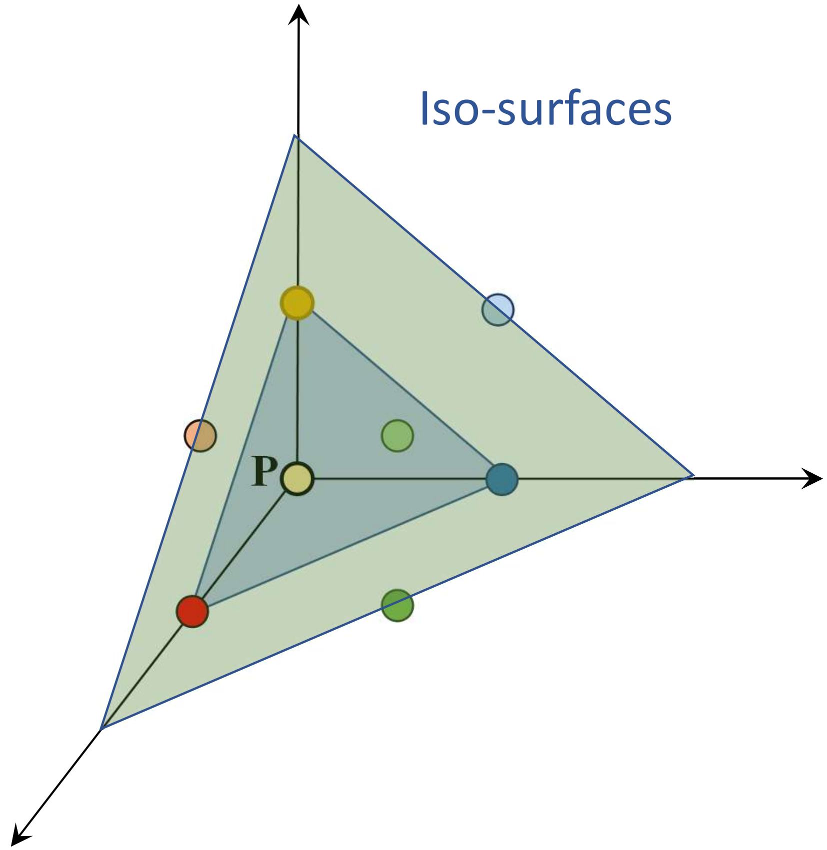




Hamming Distance

$$d(P,Q) = \sum_{i=1}^{d} \mathrm{I}(p_i \neq q_i)$$



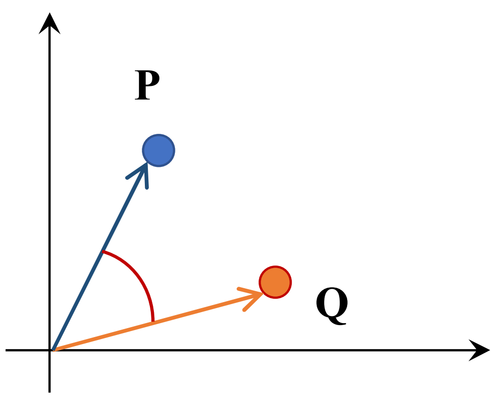


0	1	1	0	1	1	0
0	0	1	0	1	0	0

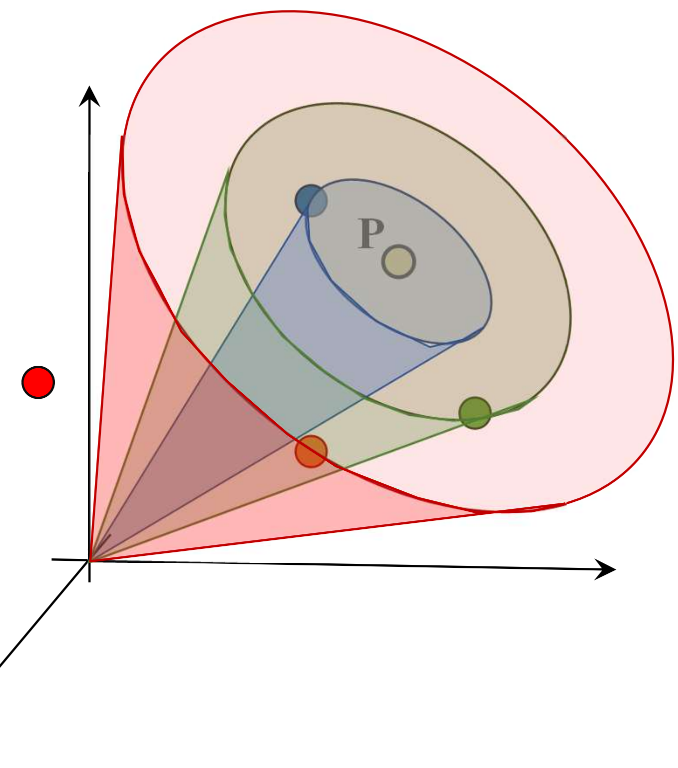
Cosine Distance

$$s(P, Q) = \cos\theta = \frac{\mathbf{P}.\mathbf{Q}}{\|\mathbf{P}\|\|\mathbf{Q}\|}$$

$$d(P, Q) = 1 - s(P, Q)$$

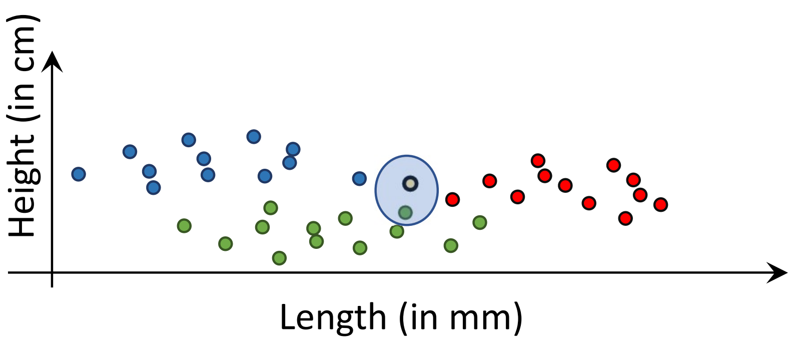


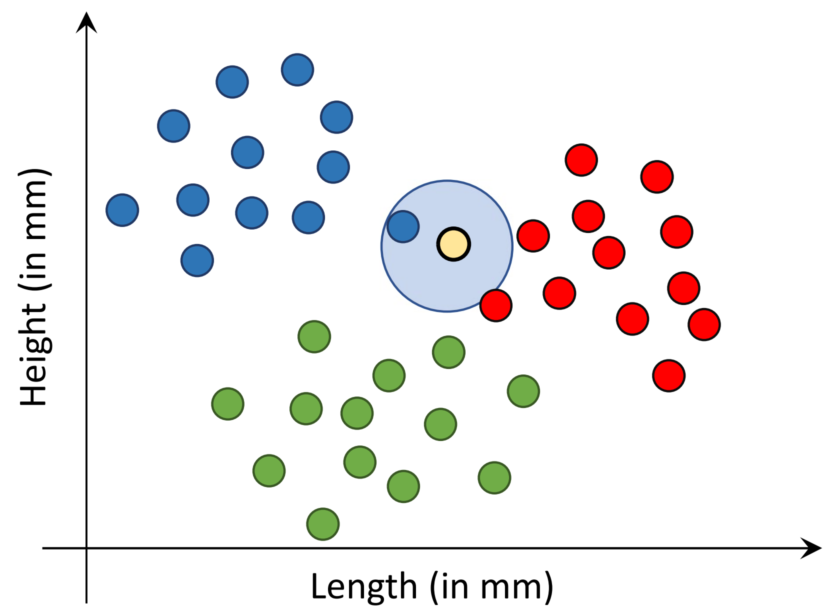
Iso-surfaces



What is the inductive bias of k-NN?

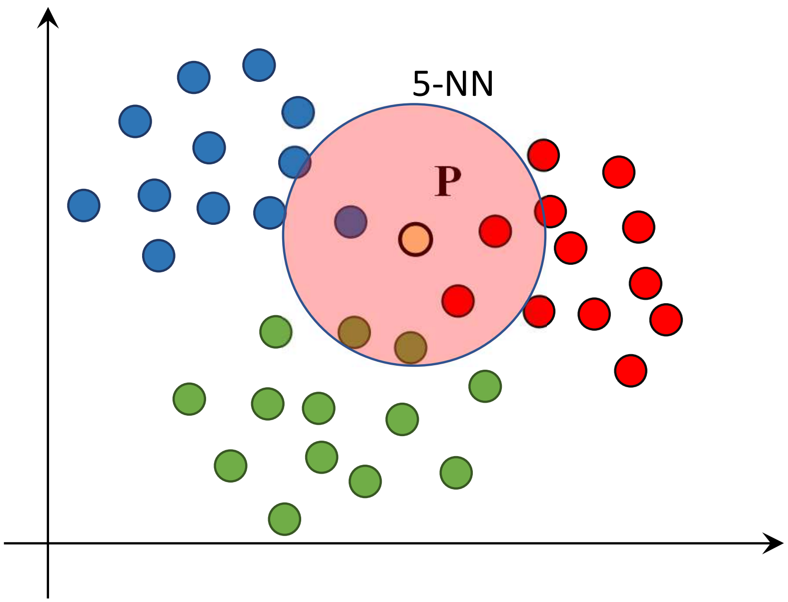
- Nearby instances should have the same label
- All features are equally important
- Complexity is tuned by the k parameter





Weighted k-NN

- If there is a tie in majority labels, one can do weighted voting
  - Samples are weighted by inverse of distance to the point p
  - e.g., $w_i = \frac{1}{1+d(p,x_i)}$
  - Example:
    - Blue:
      - Distance: 1
      - Weight: 0.5
    - Green:
      - Distances: 1.5, 1.6
      - Weight: 0.4 + 0.38 = 0.78
    - Red:
      - Distances: 1.1, 1.3
      - Weight: 0.48 + 0.43 = 0.91
  - **Class(p)** = **Red**

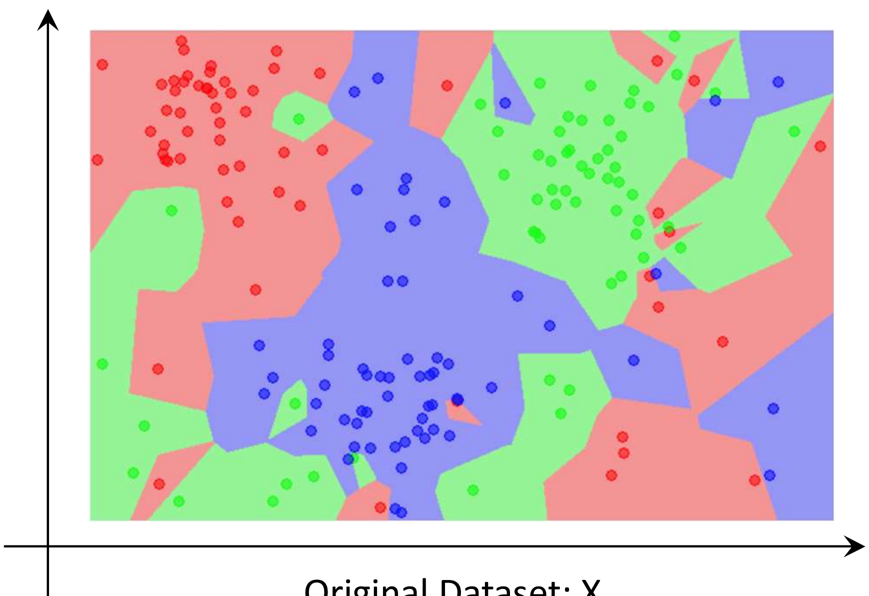


k-Nearest Neighbour: Properties

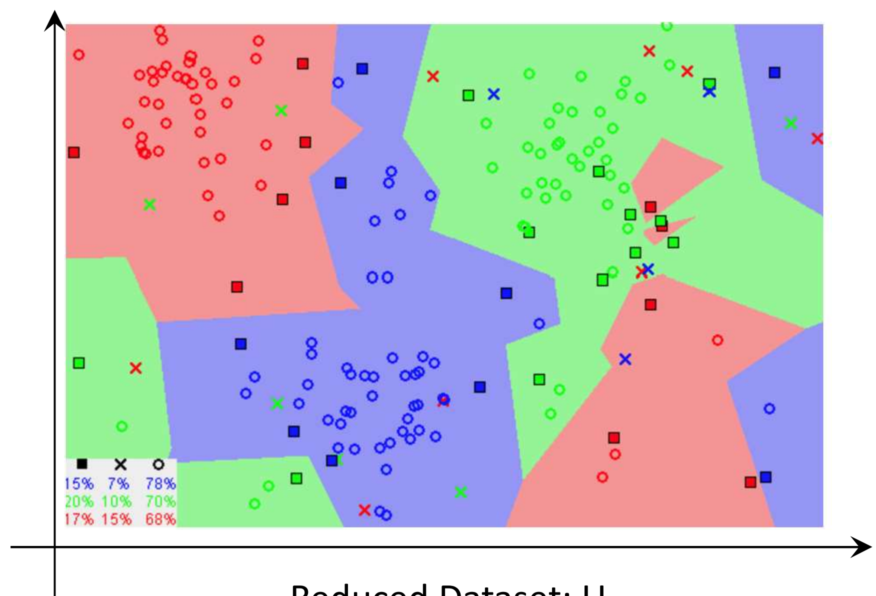
- What's nice:
  - Simple and intuitive; easily implementable
  - Asymptotically consistent (a theoretical property)
    - With infinite training data and large enough k, k-NN approaches the best possible classifier
- What's not so nice:
  - Stores all the training data in memory even at test time
    - Can be memory intensive for large training datasets
    - An example if non-parametric, or memory/instance-based methods
  - Expensive at test time: $\mathcal{O}(ND)$ computations for each test point
    - Have to search through all training data to find nearest neighbours
    - Distance computations with $\mathcal{N}$ training points (D features each)
  - Sensitive to noisy features (Not 1-NN)
  - May perform badly in high dimensions (curse of dimensionality)

Data Reduction: Condensed NN

- Given a training set X, and the condensed set U ={},
    1. Choose an x whose nearest prototype in U has a different label than x.
    2. Move x from X to U
    3. Repeat until no more prototypes are added to U.
- Use U instead of X for classification.



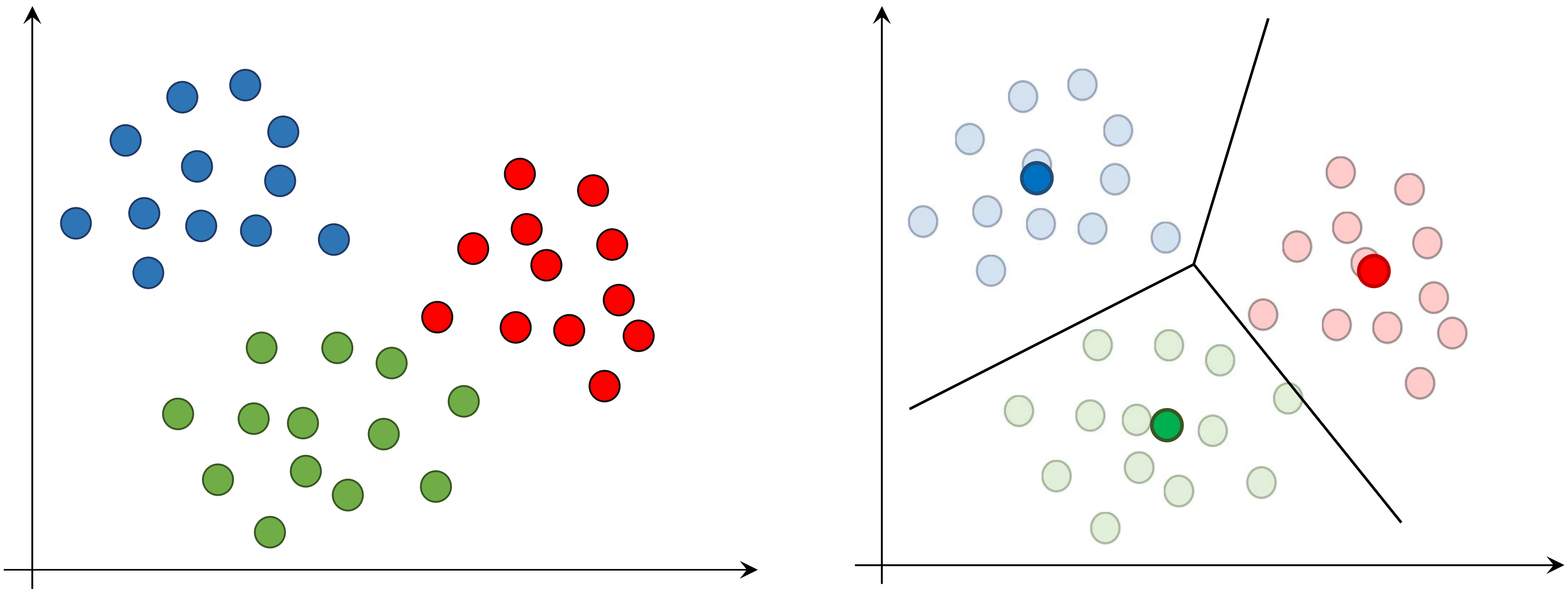
Original Dataset: X



Reduced Dataset: U

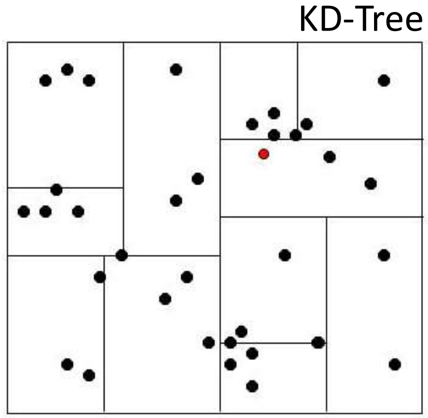
Nearest Mean Classifier

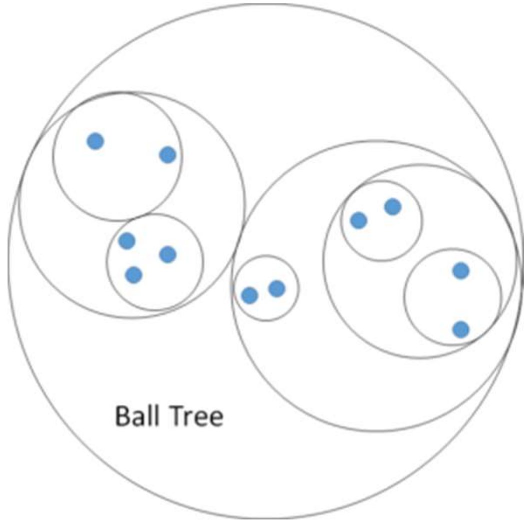
- Represent each class by a single prototype; Its Mean



Fast / Approximate Nearest Neighbor

- Quick search for NN with possible errors
- KD Tree
  - Binary space-partitioning trees with axis-parallel splits.
    Each node is a hyperplane
- Ball Tree
  - Samples are grouped by spheres. Each node has a specific center and radius





Example: Digit Classification

- Decent performance with lots of data

- Yann Le Cunn – MNIST Digit Recognition
  - Handwritten Digits of 28 X 28-pixel, d = 784
  - 60,000 training samples and 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

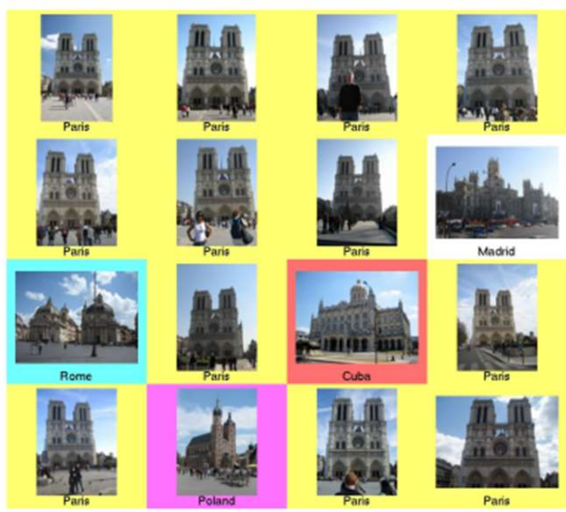
Where on Earth is this Photo From?

- Problem: Where was this picture taken?

- Link to the original paper: http://graphics.cs.cmu.edu/projects/im2gps/im2gps.pdf

Where on Earth is this Photo From?

- Problem: Where was this picture taken?
  - Get 6M images from Flickr with GPS info
  - Represent each image with meaningful features
  - Do kNN!



- Link to the original paper: http://graphics.cs.cmu.edu/projects/im2gps/im2gps.pdf