

CSC 411: Lecture 05: Nearest Neighbors

Ethan Fetaya, James Lucas and Emad Andrews

University of Toronto

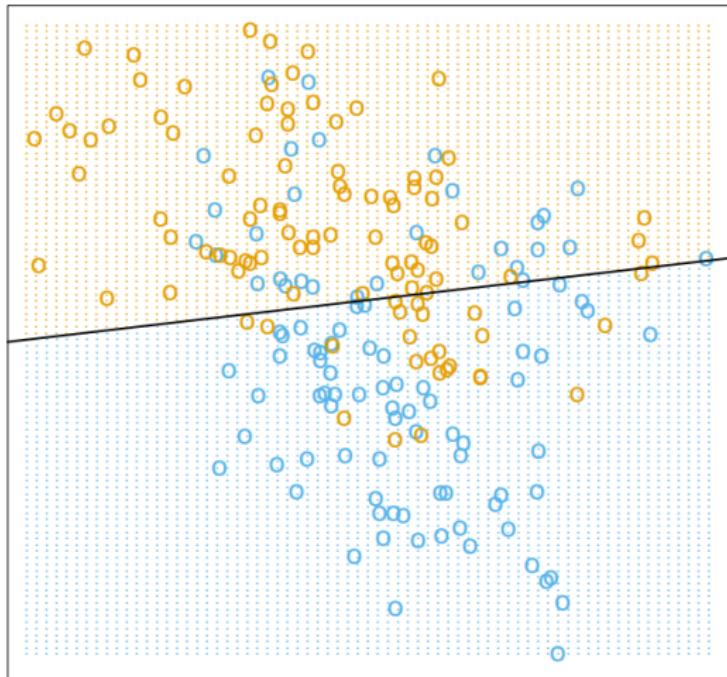
Today

- Non-parametric models
 - ▶ distance
 - ▶ non-linear decision boundaries

Note: We will mainly use today's method for classification, but it can also be used for regression

Nonlinear Classification

We can construct a linear decision boundary $y = \text{sign}(w_0 + w_1x_1 + w_2x_2)$



[Image credit: "The Elements of Statistical Learning"]

What is the meaning of "linear" classification

- **Linear classification** means that the decision boundary is linear

$$\mathbf{w}^T \mathbf{x} = 0$$

- Most problems are not linear
 - ▶ "Classification of mathematical problems as linear and nonlinear is like classification of the Universe as bananas and non-bananas."
- This can be solved with good features but... how do you find good features?
(hint: Deep learning)

Instance-based Learning

- Alternative to parametric models are **non-parametric** models
- These are typically simple methods for approximating discrete-valued or real-valued target functions (they work for classification or regression problems)
- NN Learning amounts to simply **memorizing** training data
- Test instances classified using **similar** training instances
- Embodies often sensible underlying **assumptions**:
 - ▶ Output varies smoothly with input
 - ▶ Data occupies sub-space of high-dimensional input space



Parametric vs. Non-parametric Models

- **Parametric**
 - ▶ Fixed Number of parameters
- **Non-parametric**
 - ▶ No fixed number of parameters
 - ▶ Parameters grow with the number of training data points

Non-parametric does not mean no parameters!

Nearest Neighbors

- Training example in Euclidean space: $\mathbf{x} \in \Re^d$
- Idea: The value of the target function for a new query is estimated from the known value(s) of the nearest training example(s)
- Distance typically defined to be Euclidean:


$$\|\mathbf{x}^{(a)} - \mathbf{x}^{(b)}\|_2 = \sqrt{\sum_{j=1}^d (x_j^{(a)} - x_j^{(b)})^2}$$

Algorithm:

1. Find example (\mathbf{x}^*, t^*) (from the stored training set) closest to the test instance \mathbf{x} . That is:

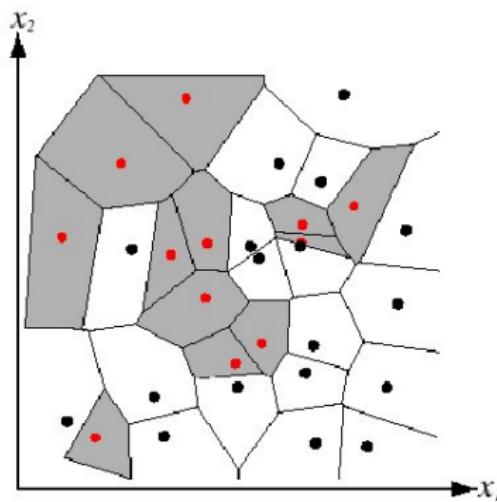
$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\operatorname{argmin}} \text{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. Output $y = t^*$

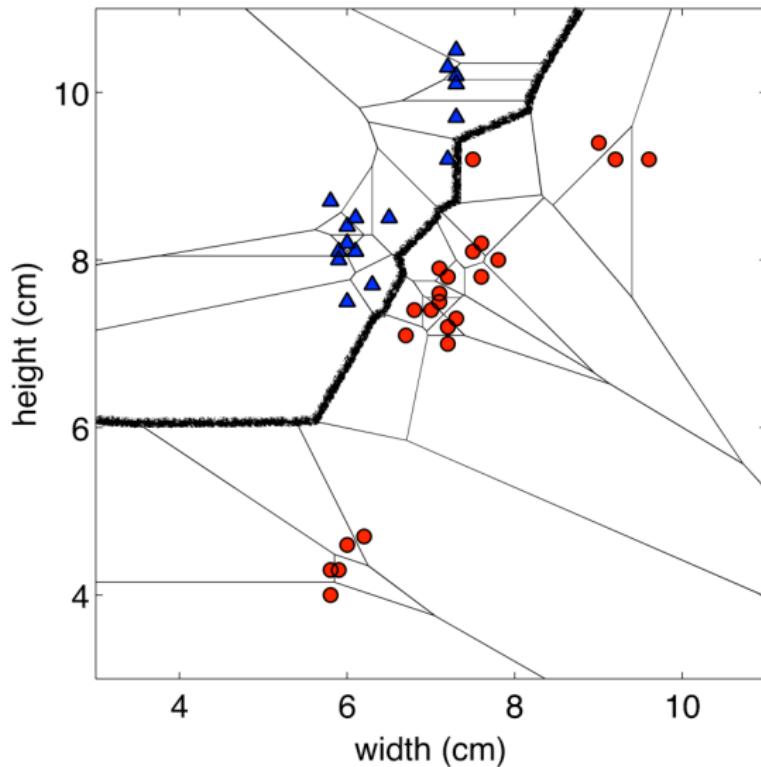
- Note: we don't really need to compute the square root. Why?

Nearest Neighbors: Decision Boundaries

- Nearest neighbor algorithm does not explicitly compute **decision boundaries**, but these can be inferred
- Decision boundaries: **Voronoi diagram** visualization
 - ▶ show how input space divided into classes
 - ▶ each line segment is equidistant between two points of opposite classes

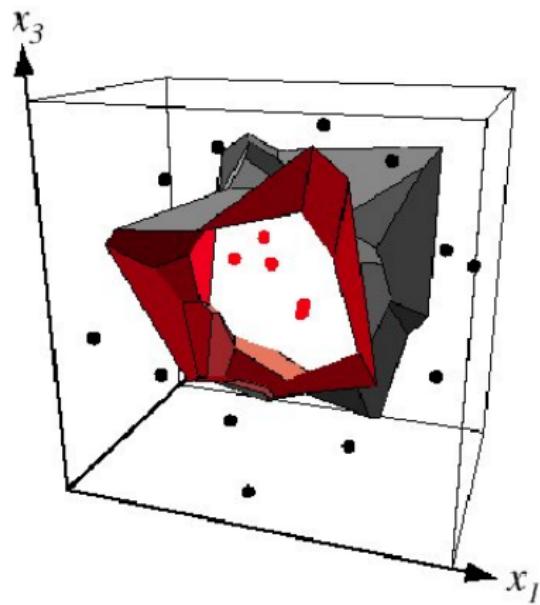


Nearest Neighbors: Decision Boundaries



Example: 2D decision boundary

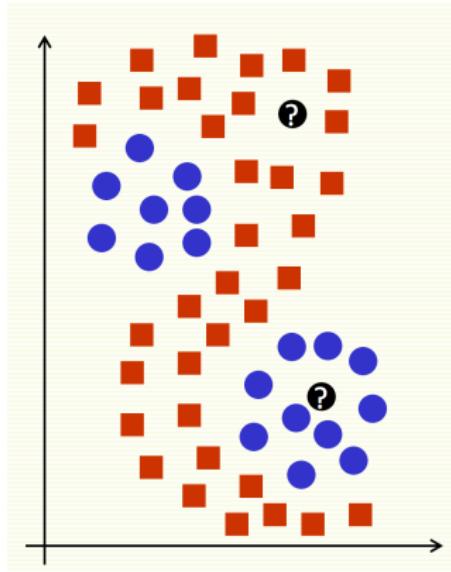
Nearest Neighbors: Decision Boundaries



Example: 3D decision boundary

Nearest Neighbors: Multi-modal Data

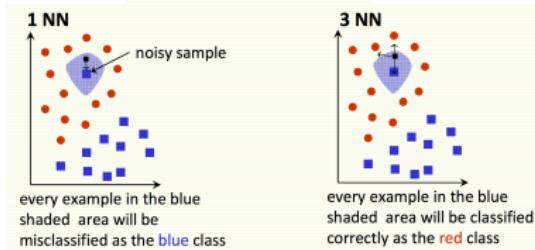
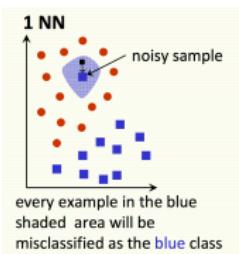
- Nearest Neighbor approaches can work with multi-modal data



[Slide credit: O. Veksler]

k-Nearest Neighbors

[Pic by Olga Veksler]



- Nearest neighbors **sensitive to noise or mis-labeled data** ("class noise").
Solution?
- Smooth by having k nearest neighbors vote

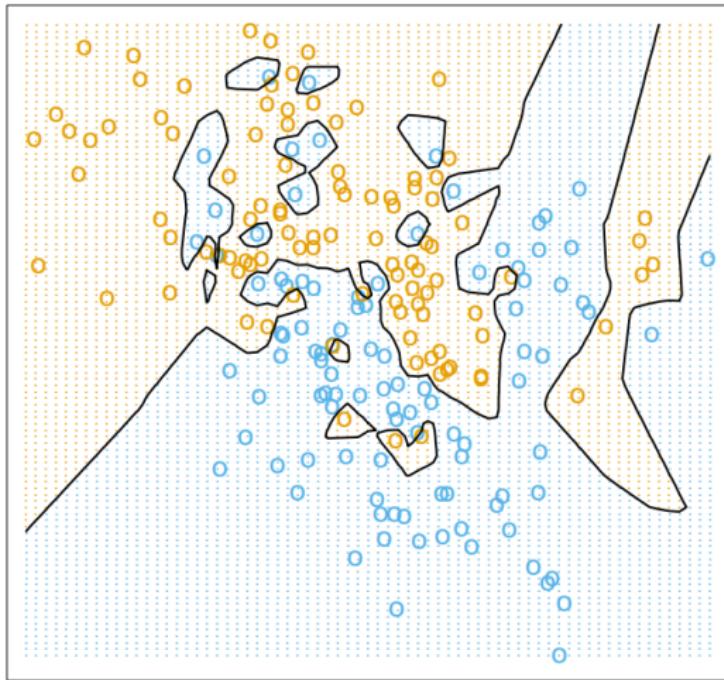
Algorithm (kNN):

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

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

K-Nearest neighbors

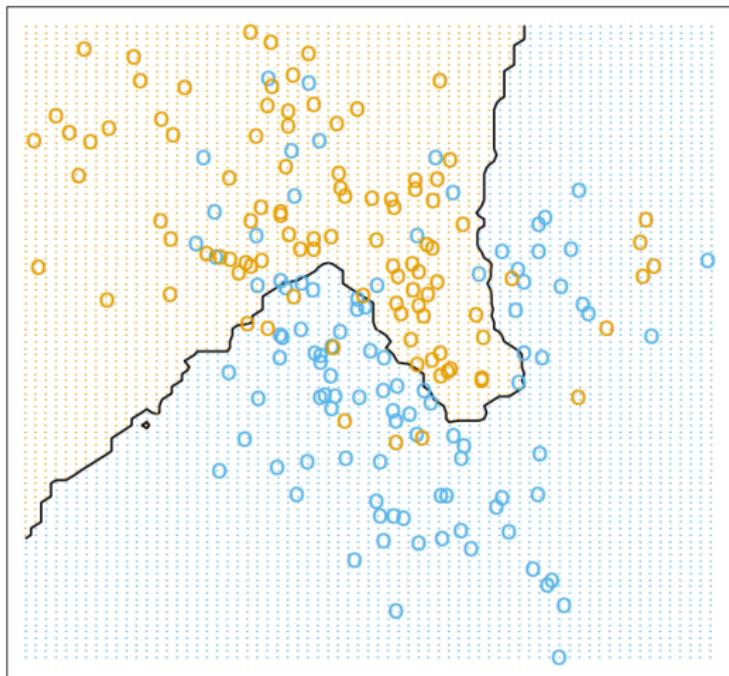
$k=1$



[Image credit: "The Elements of Statistical Learning"]

K-Nearest neighbors

$k=15$



[Image credit: "The Elements of Statistical Learning"]

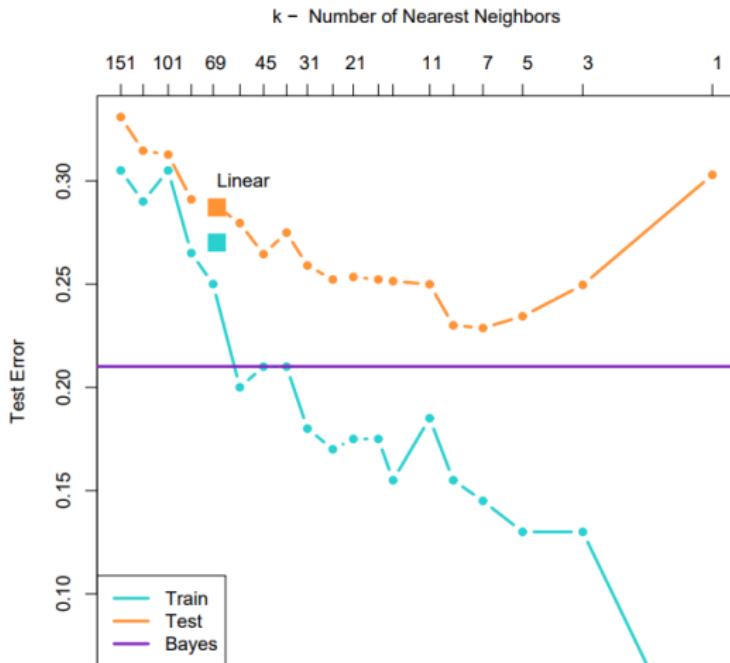
k-Nearest Neighbors

How do we **choose k ?**

- Larger k may lead to better performance
- But if we set k too large we may end up looking at samples that are not neighbors (are far away from the query)
- We can use validation set / cross-validation to find k
- Rule of thumb is $k < \sqrt{n}$, where n is the number of training examples

[Slide credit: O. Veksler]

K-Nearest neighbors



[Image credit: "The Elements of Statistical Learning"]

NN Theory in a nutshell *

- Can we **generalize** just by memorizing?
- Assumptions:
 - ▶ NN assumes smoothness - we will assume $p(y = 1|x)$ is **c-Lipschitz**.
 - ▶ The inputs are in unit cube $x \in [0, 1]^d$
- Definition:
 - ▶ The **Bayes-optimal classifier** is (for binary prediction)

$$h^*(x) = \begin{cases} 1 & \text{if } p(y = 1|x) \geq 0.5 \\ -1 & \text{if } p(y = 1|x) < 0.5 \end{cases}$$

- ▶ Given a sample S of size m , h_S is the 1-NN classifier.

Theorem::

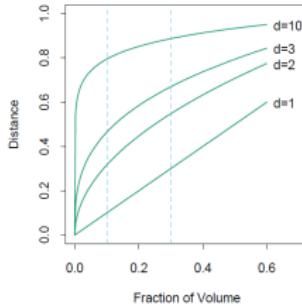
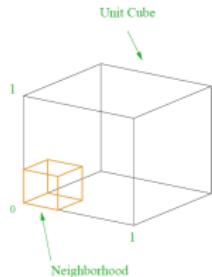
Under these assumptions

$$\mathbb{E}_S[L_{\mathcal{D}}(h_S)] \leq 2 \cdot L_{\mathcal{D}}(h^*) + \frac{4c\sqrt{d}}{m^{1/(d+1)}}$$

Converges to twice the optimal loss but exponentially slow in d .

The curse of dimensionality

- If we want the nearest neighbor to be closer than ϵ , how many points do we need to guarantee it?
- The volume of a single ball of radius ϵ is $\mathcal{O}(\epsilon^d)$ 
- The total volume of $[0, 1]^d$ is 1.
- Therefore $\mathcal{O}\left((\frac{1}{\epsilon})^d\right)$ balls are needed to cover the volume.



[Image credit: "The Elements of Statistical Learning"]

k-Nearest Neighbors: Issues & Remedies

- If some attributes (coordinates of x) have larger ranges, they are treated as more important
 - ▶ normalize scale
 - ▶ Simple option: Linearly scale the range of each feature to be, e.g., in range [0,1]
 - ▶ Linearly scale each dimension to have 0 mean and variance 1 (compute mean μ and variance σ^2 for an attribute x_j and scale: $(x_j - \mu)/\sigma$)
 - ▶ be careful: sometimes scale matters
- Irrelevant, correlated attributes add noise to distance measure
 - ▶ eliminate some attributes
 - ▶ or vary and possibly adapt weight of attributes
- Non-metric attributes (symbols)
 - ▶ Hamming distance
- High Dimensional Data: “Curse of Dimensionality”
 - ▶ Required amount of training data increases exponentially with dimension
 - ▶ Computational cost also increases

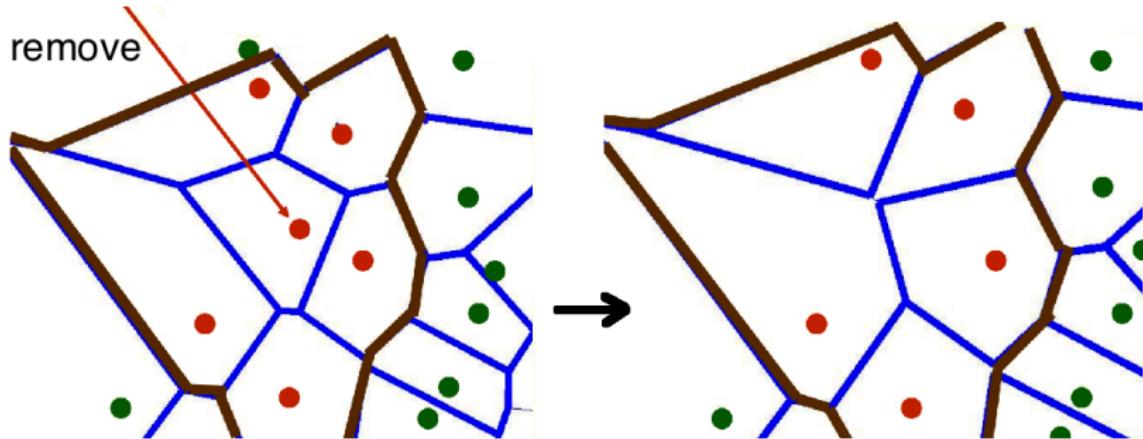
k-Nearest Neighbors: Issues (Complexity) & Remedies

- **Expensive at test time:** To find one nearest neighbor of a query point \mathbf{x} , we must compute the distance to all N training examples. Complexity: $O(kdN)$ for kNN
 - ▶ Use subset of dimensions
 - ▶ Pre-sort training examples into fast data structures (e.g., kd-trees)
 - ▶ Compute only an approximate distance (e.g., LSH)
 - ▶ Remove redundant data (e.g., condensing)
- **Storage Requirements:** Must store all training data
 - ▶ Remove redundant data (e.g., condensing)
 - ▶ Pre-sorting often increases the storage requirements

[Slide credit: David Claus]

k-Nearest Neighbors Remedies: Remove Redundancy

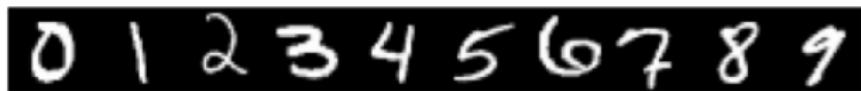
- If all Voronoi neighbors have the same class, a sample is useless, remove it



[Slide credit: O. Veksler]

Example: Digit Classification

- 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

Fun Example: Where on Earth is this Photo From?

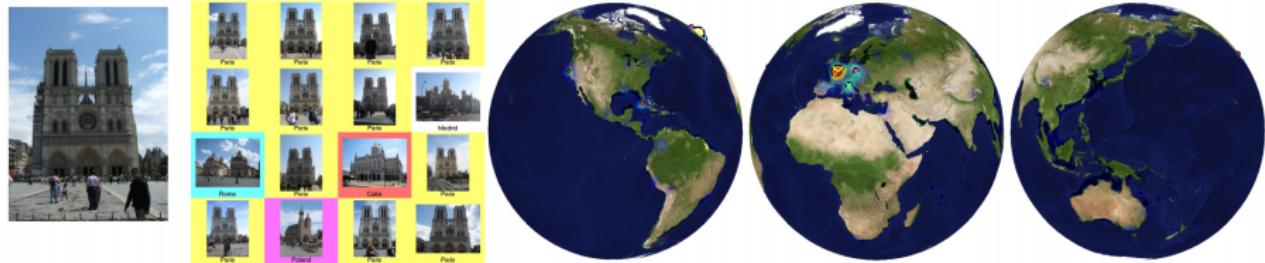
- Problem: Where (e.g., which country or GPS location) was this picture taken?



[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/>]

Fun Example: Where on Earth is this Photo From?

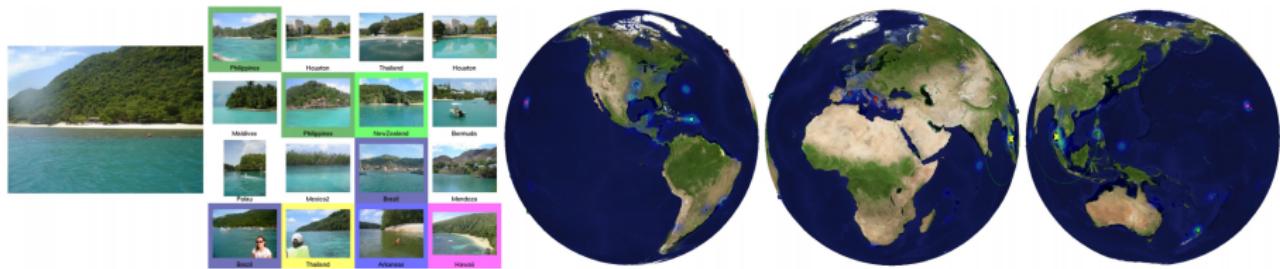
- Problem: Where (e.g., which country or GPS location) was this picture taken?
 - ▶ Get 6M images from Flickr with GPS info (dense sampling across world)
 - ▶ Represent each image with meaningful features
 - ▶ Do kNN!



[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/>]

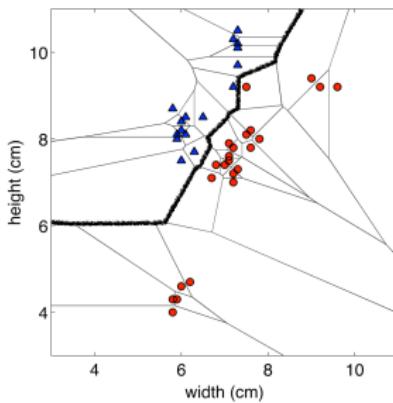
Fun Example: Where on Earth is this Photo From?

- 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/>]

K-NN Summary



- Naturally **forms complex decision boundaries**; adapts to data density
- If we have lots of samples, kNN typically works well
- Problems:
 - ▶ Sensitive to class noise
 - ▶ Sensitive to scales of attributes
 - ▶ Distances are less meaningful in high dimensions
 - ▶ Expensive at run time.
- **Inductive Bias:** What kind of decision boundaries do we expect to find?