On Nearest Neighbours

CS189/289A: Introduction to Machine Learning

Stella Yu

UC Berkeley

31 October 2017

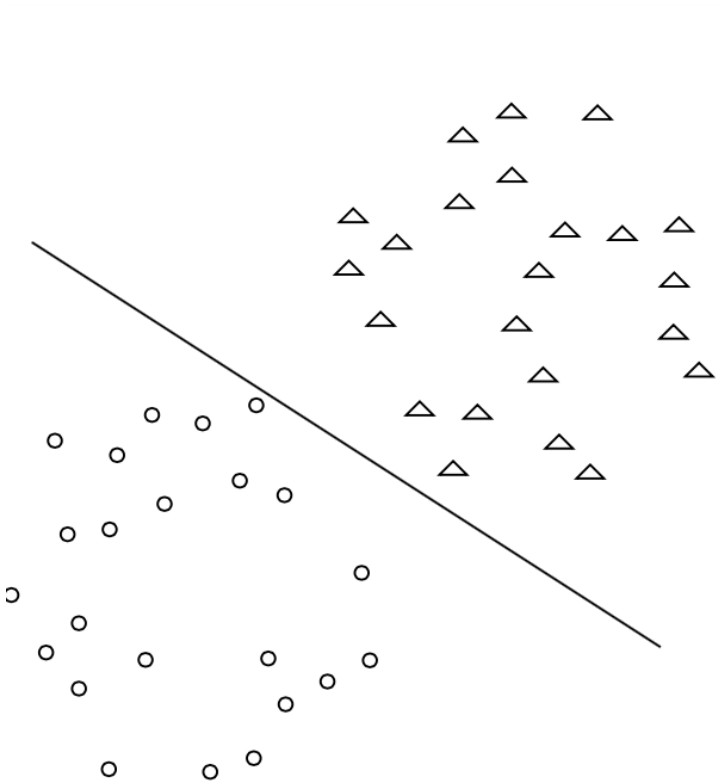
Outline

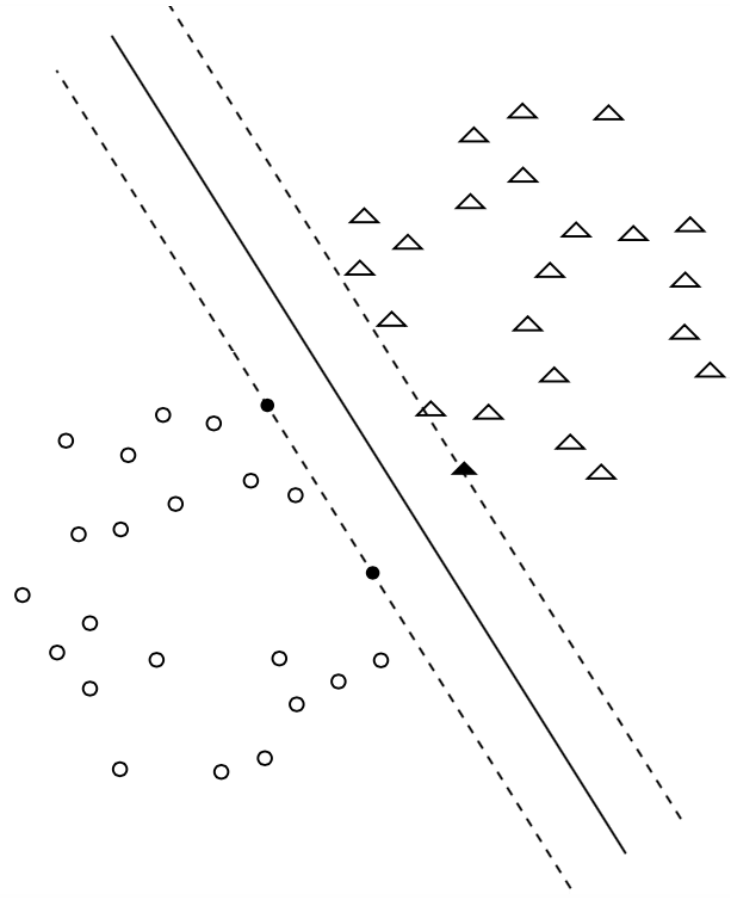
- Kernel SVM review

- *k*—Nearest neighbour (k-NN) algorithms
- k-NN decision regions
- Theoretical results on k-NN

- k-NN for image understanding
- k-NN: more data
- k-NN: fewer effective dimensions
- k-NN: better distance functions

Maximum Margin Principle for the Primal SVM





Primal and Dual SVMs

Primal SVM: (1)

$$\min_{w,t,\xi} \quad \varepsilon(w,t,\xi) = \frac{1}{2}|w|^2 + C\sum_{i=1}^{n}\xi_i \tag{2}$$

$$\text{s. t.} \quad y_i(w\cdot x_i - t) \geq 1-\xi_i, \tag{3}$$

$$\xi_i \geq 0, \quad i = 1,2,\ldots,n \tag{4}$$

Lagrangian: (5)

$$L(w,t,\xi,\alpha,\beta) = \frac{1}{2}|w|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(w\cdot x_i - t) - 1 + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i$$

Dual SVM: (6)

$$\max_{\alpha} \quad L(\alpha) = \alpha' 1 - \frac{1}{2}\alpha' G\alpha \tag{7}$$

$$\text{where} \quad G_{ij} = y_i(x_i\cdot x_j)y_j \tag{8}$$

$$\text{s. t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C,\ i = 1,\ldots,n. \tag{9}$$

KKT Optimality Conditions for SVM

- Stationarity

$$\frac{\partial L}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{10}$$

$$\frac{\partial L}{\partial t} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{11}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \quad \Rightarrow \quad C - \alpha_i - \beta_i = 0 \tag{12}$$

- Feasibility

$$y_i(w \cdot x_i - t) \geq 1 - \xi_i \tag{13}$$

$$\xi_i \geq 0, \quad \alpha_i \geq 0, \quad \beta_i \geq 0 \tag{14}$$

- Complementary slackness

$$\alpha_i \cdot (y_i(w \cdot x_i - t) - (1 - \xi_i)) = 0 \tag{15}$$

$$\beta_i \cdot \xi_i = 0 \tag{16}$$

SVM Solution: From the Dual to the Primal

- Solve α in the SVM dual:

$$\max_{\alpha} \quad L(\alpha) = \alpha' 1 - \frac{1}{2}\alpha' G \alpha \tag{17}$$

$$\text{where} \quad G_{ij} = y_i (x_i \cdot x_j) y_j \tag{18}$$

$$\text{s. t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \le \alpha_i \le C, i = 1, \ldots, n. \tag{19}$$

- Solve w, t in the SVM Primal:

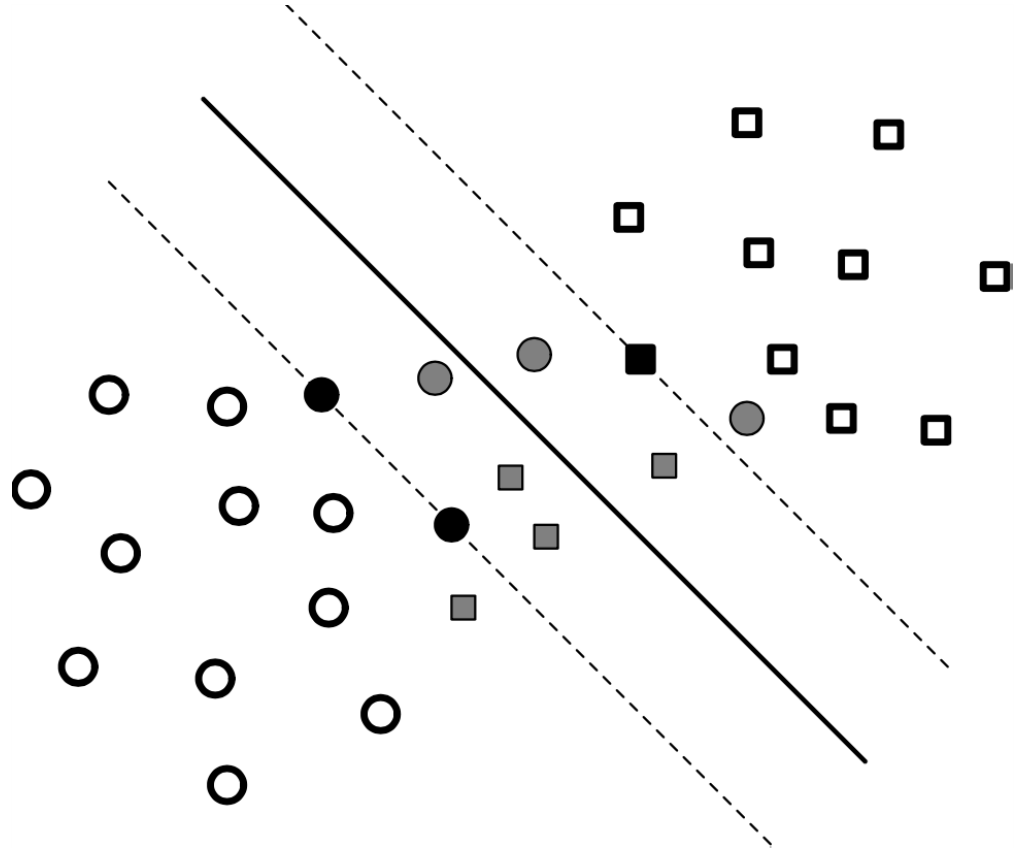
$$w = \sum_{i=1}^{n} \alpha_i y_i x_i = \sum_{\alpha_i > 0} \alpha_i y_i x_i \tag{20}$$

$$t = \text{mean}(\{w \cdot x_i - y_i : \quad 0 < \alpha_i < C\}) \tag{21}$$

$$\xi_i = \begin{cases} 1 - y_i(w \cdot x_i - t), & \alpha_i = C \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

- Instances with $\alpha_i > 0$ are support vectors.

Support Vectors and Their Dual Variables α



$$\alpha_i = 0 \Rightarrow \quad y_i f(x_i) \geq 1: \text{ on or outside the margin} \tag{23}$$

$$0 < \alpha_i < C \Rightarrow \quad y_i f(x_i) = 1: \text{ on the margin} \tag{24}$$

$$\alpha_i = C \Rightarrow \quad y_i f(x_i) \leq 1: \text{ on or inside the margin} \tag{25}$$

$$\alpha_i = 0 \Leftarrow \quad y_i f(x_i) > 1: \text{ outside the margin} \tag{26}$$

$$\alpha_i = C \Leftarrow \quad y_i f(x_i) < 1: \text{ inside the margin} \tag{27}$$

Only A Kernel Function Is Needed in Linear SVM

$$\text{SVM dual:} \quad \max_{\alpha} \quad L(\alpha) = \alpha' 1 - \frac{1}{2}\alpha' G \alpha \tag{28}$$

$$\text{where} \quad G_{ij} = y_i (x_i \cdot x_j) y_j \tag{29}$$

$$\text{s. t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, n. \tag{30}$$

$$\text{decision function:} \quad f(x) = w \cdot x - t = \left(\sum_{i=1}^{n} \alpha_i y_i x_i \right) \cdot x - t \tag{31}$$

$$= \sum_{\alpha_i > 0} \alpha_i y_i (x_i \cdot x) - t \tag{32}$$

All we need is some measure of pairwise similarity (a scalar) between features: $x \cdot z$, not the features x and z themselves:

$$K(x, z) = x \cdot z \quad \Leftarrow \text{kernel function} \tag{33}$$

From the Linear SVM to the Kernel SVM

$$\text{SVM dual:} \quad \max_{\alpha} \quad L(\alpha) = \alpha' 1 - \frac{1}{2}\alpha' G \alpha \tag{34}$$

$$\text{where} \quad G_{ij} = y_i \, (\phi(x_i) \cdot \phi(x_j)) \, y_j \tag{35}$$

$$\text{s. t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, n. \tag{36}$$

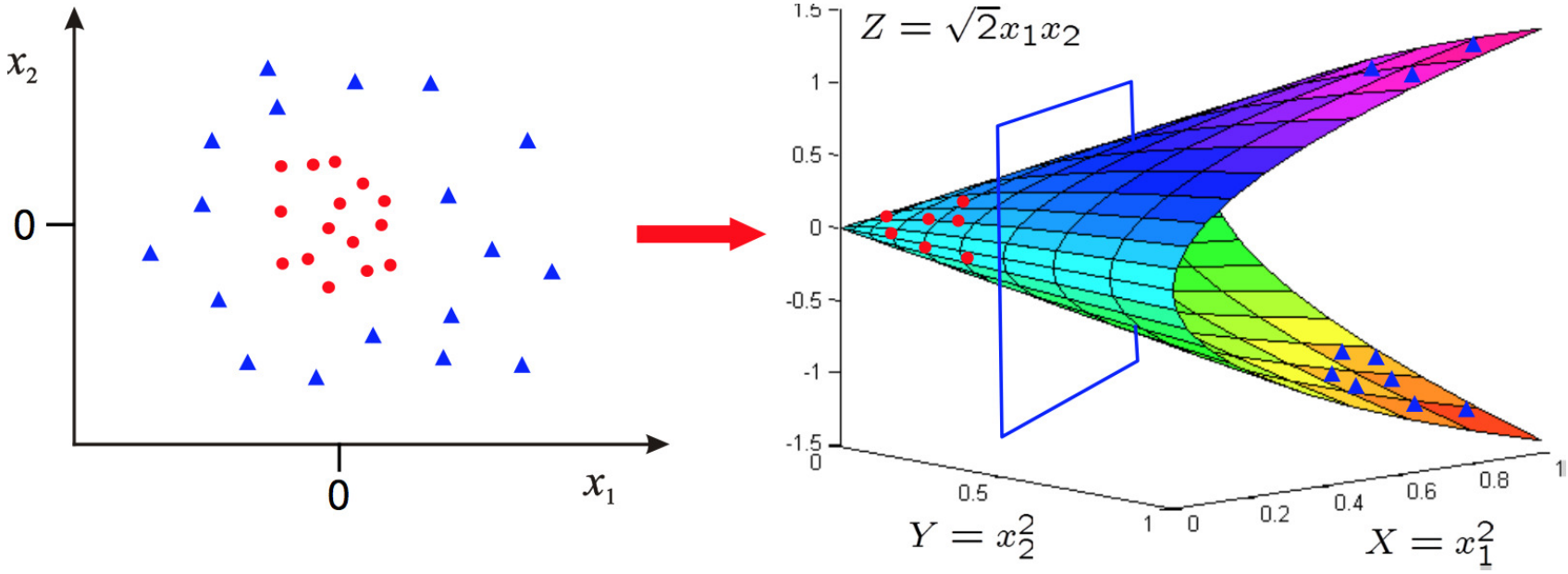
$$\text{decision function:} \quad f(x) = w \cdot \phi(x) - t = \left(\sum_{i=1}^{n} \alpha_i y_i \, \phi(x_i) \right) \cdot \phi(x) - t$$

$$= \sum_{\alpha_i > 0} \alpha_i y_i \, (\phi(x_i) \cdot \phi(x)) - t \tag{37}$$

- Option 1: Linear SVM in the $\phi(x)$ feature space, need $\phi(x)$.
- Option 2: Kernel SVM in the sample space of x, need $K(x, z)$:

$$K(x, z) = \phi(x) \cdot \phi(z) \quad \Leftarrow \text{kernel function} \tag{38}$$

Nonlinearly Separable in x, Linearly Separable in $\phi(x)$



$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \phi(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix} \tag{39}$$

$$\phi(x) \cdot \phi(z) = (x_1z_1)^2 + (x_2z_2)^2 + 2(x_1z_1x_2z_2) = (x \cdot z)^2 \tag{40}$$

Either a linear SVM in the $\phi(x)$ feature space, or equivalently a kernel SVM in the sample space of x, with kernel K:

$$K(x, z) = (x \cdot z)^2 \tag{41}$$

Linear SVM vs. Kernel SVM in General

	Linear SVM	**Kernel SVM**
training	$x_i, \quad i = 1, \ldots, n$	$K(x_i, x_j), \quad i, j = 1, \ldots, n$
testing	w	$K(x, x_i), \quad \alpha_i > 0$
memory	1 normal vector	#? support vectors
decision	feature space	sample space
boundary	linear	nonlinear

Commonly Used Kernels

- Linear kernels

$$K(x,z) = x \cdot z \tag{42}$$

- polynomial kernels

$$K(x,z) = (1 + x \cdot z)^d, \quad d > 0 \tag{43}$$

- Sigmod kernels

$$K(x,z) = \tanh(a(x \cdot z) + b) \tag{44}$$

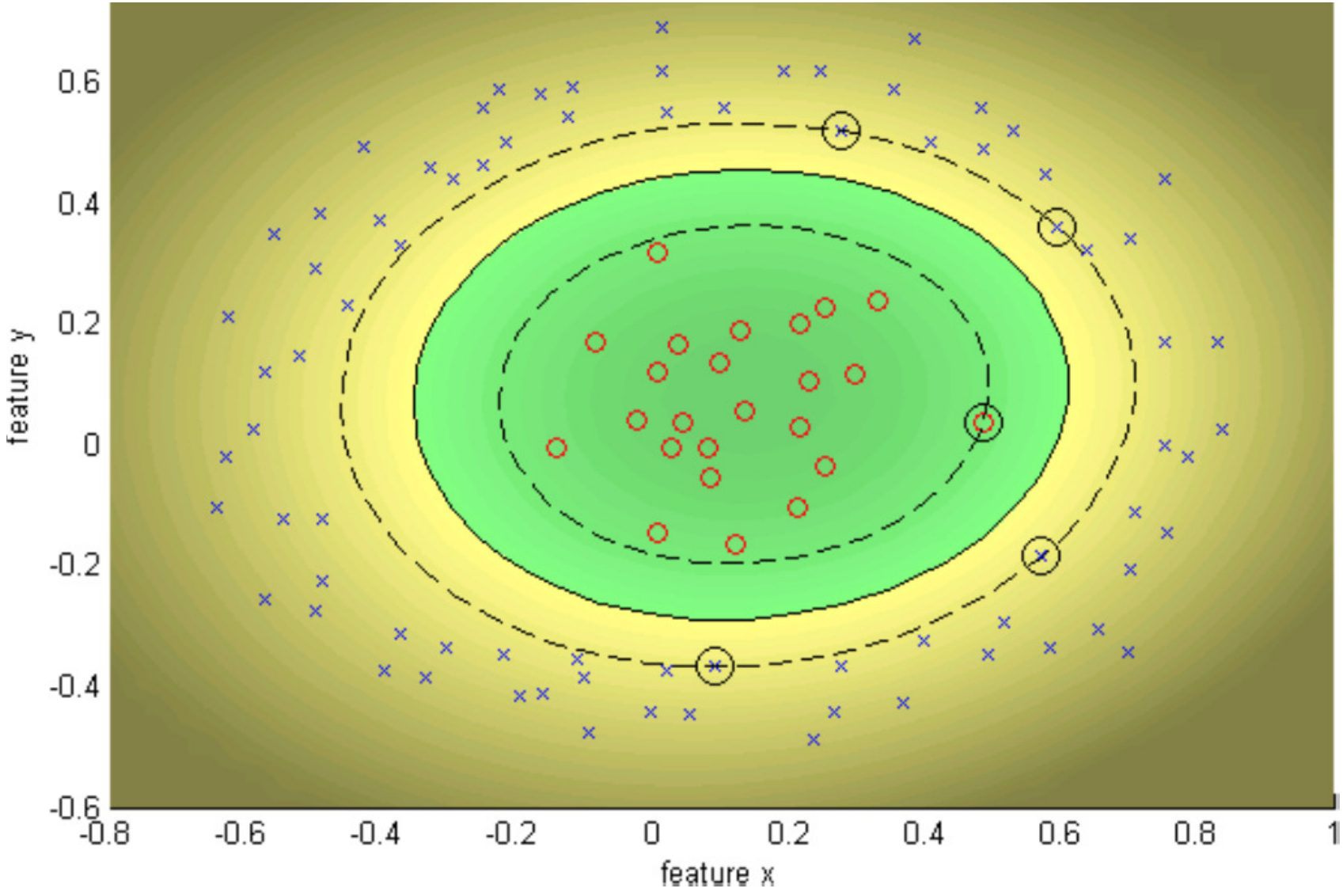
- Gaussian kernels or radial basis kernel

$$K(x,z) = \exp\left(-\frac{(x-z)^2}{2\sigma^2}\right) \tag{45}$$

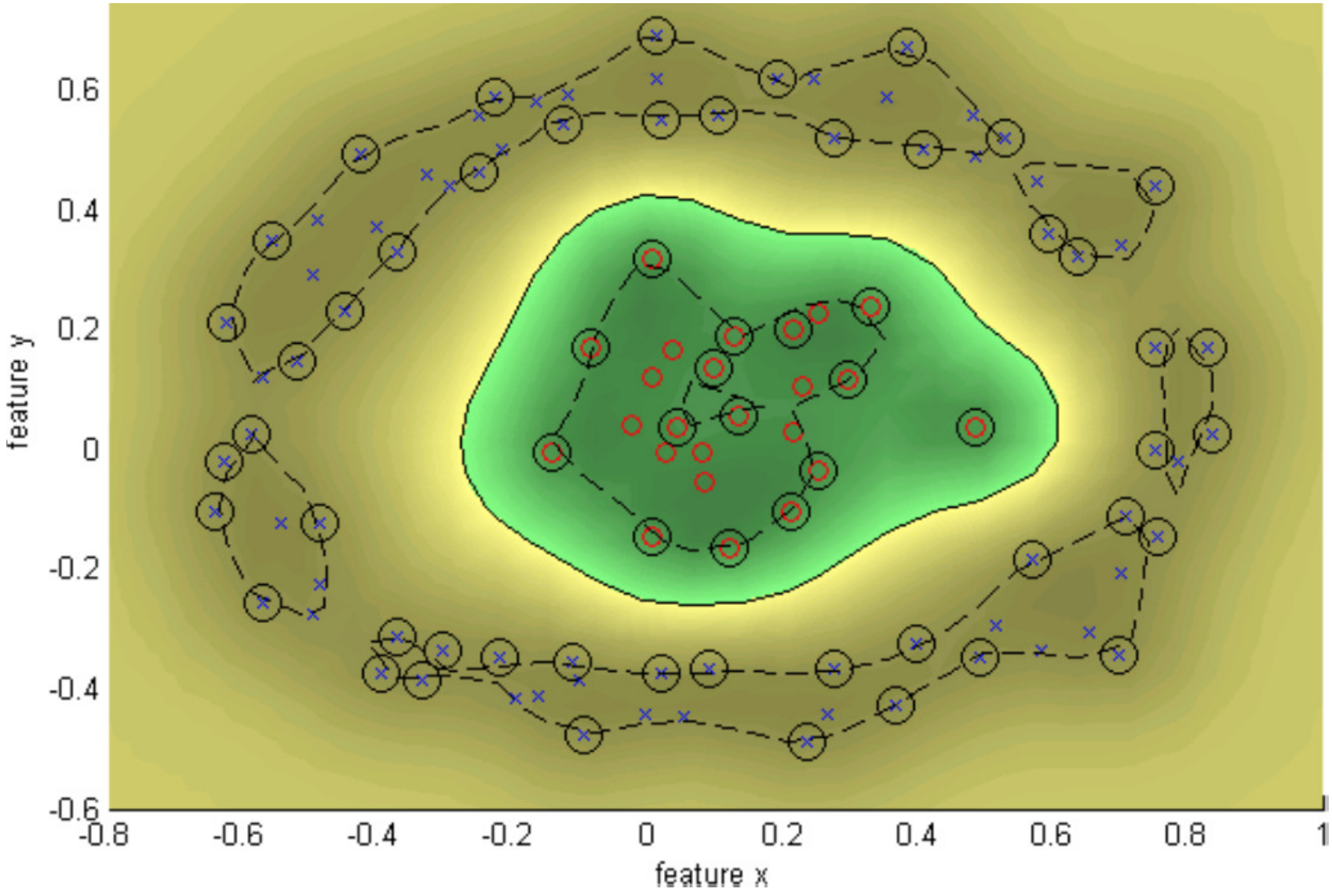
Infinite dimensional feature space, e.g. for $d = 1$,

$$\phi(x) = \exp(-\frac{x^2}{2\sigma^2})\left[1 \quad \frac{x}{\sigma\sqrt{1!}} \quad \frac{x^2}{\sigma^2\sqrt{2!}} \quad \frac{x^3}{\sigma^3\sqrt{3!}} \quad \cdots\right]' \tag{46}$$

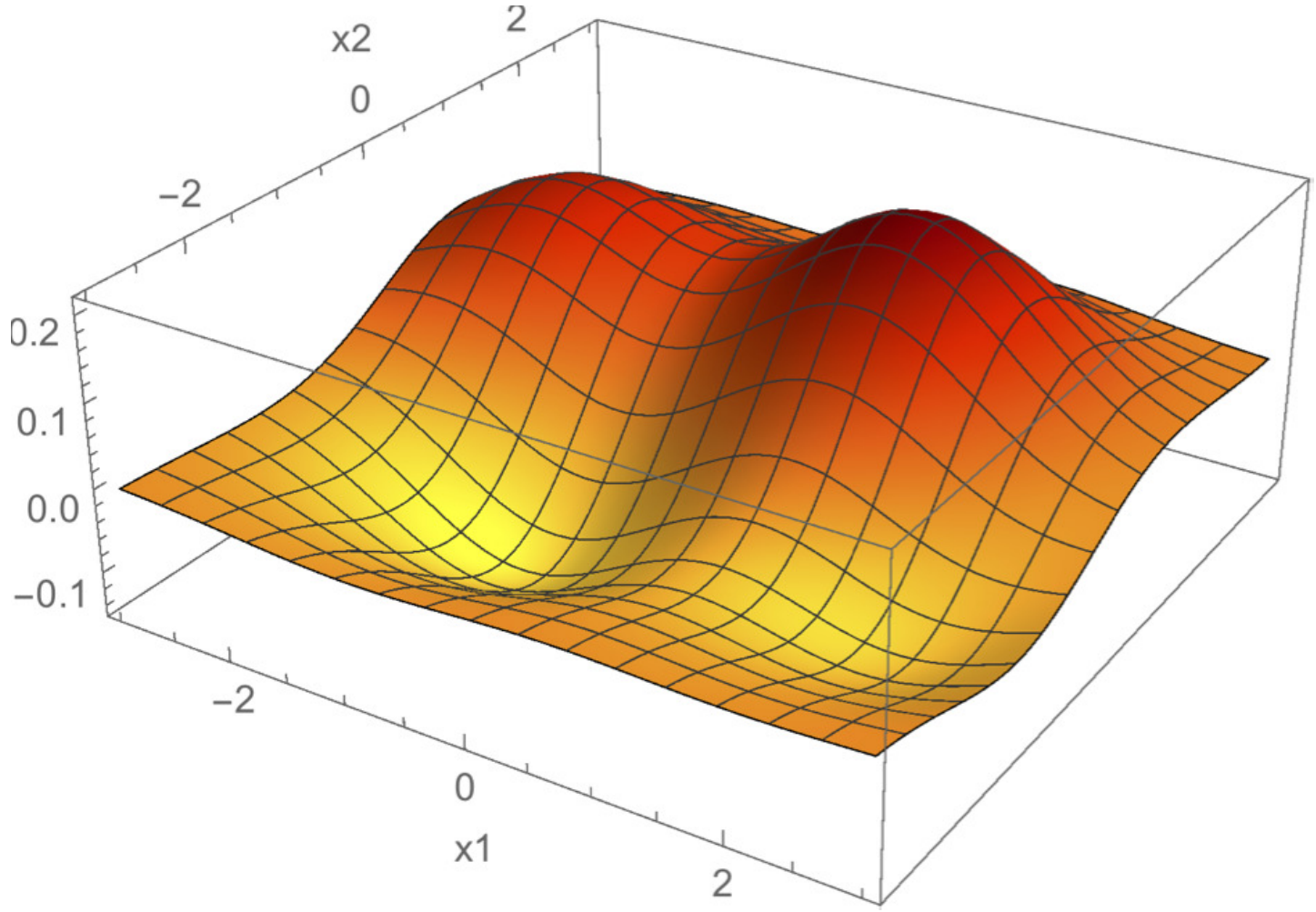
SVM Classifier with Gaussian Kernel $\sigma = 1$



SVM Classifier with Gaussian Kernel $\sigma = 0.1$



Gaussian SVM: Linear Combinations of RBFs



$$f(x) + t = \sum_{i=1}^{n} \alpha_i y_i K(x_i \cdot x) = \sum_{y_i=+1}^{n} \alpha_i K(x_i \cdot x) - \sum_{y_i=-1}^{n} \alpha_i K(x_i \cdot x) \quad (47)$$

Larger $\sigma \rightarrow$ smoother $f(x) \rightarrow$ more bias, less variance

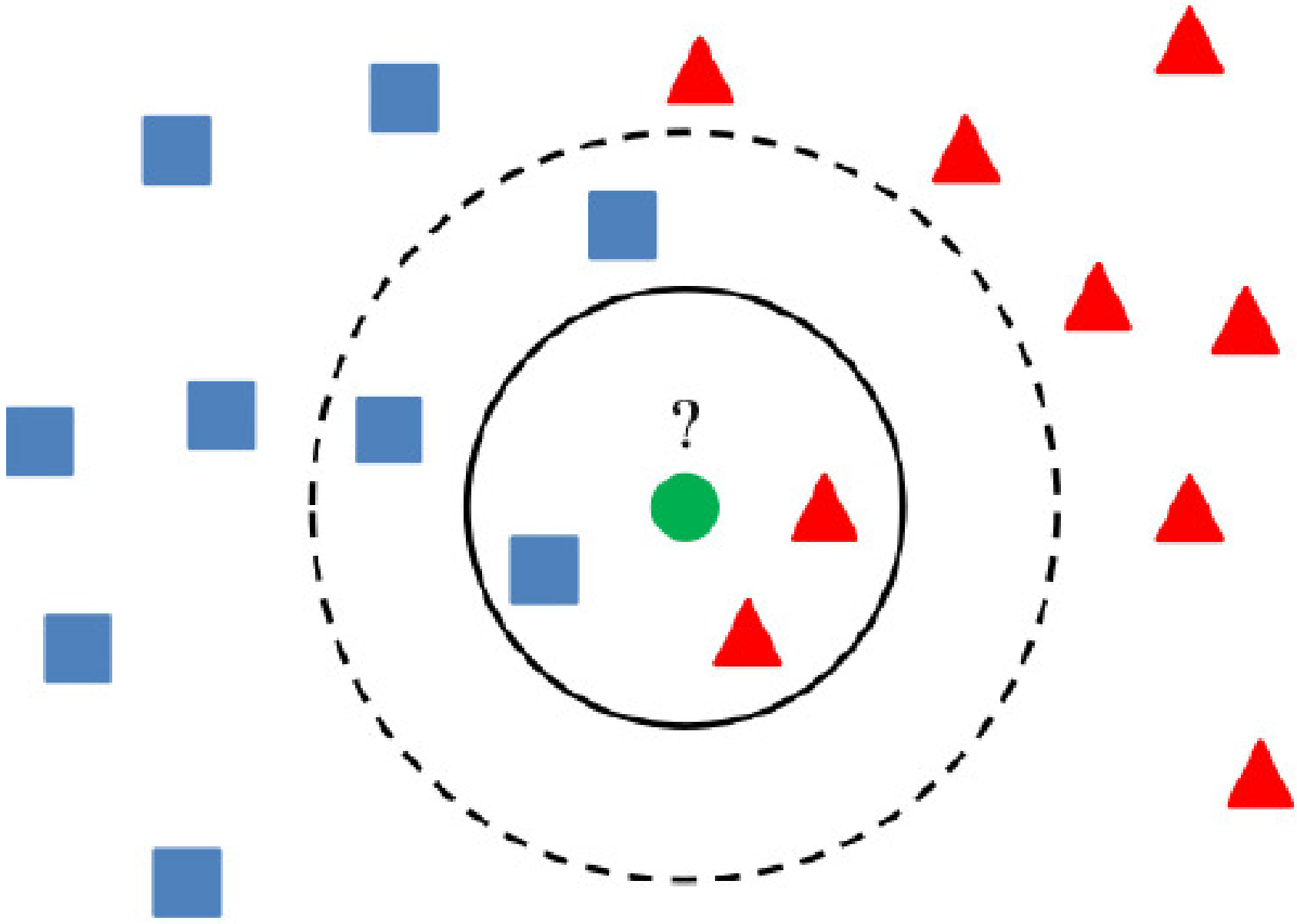
Nearest Neighbour Classifiers

- Key idea: Just store all the training examples $(x_i, f(x_i))$.
- k-Nearest neighbour classifier: Given test example x, first locate k nearest training examples x_i's, then estimate $f(x)$ with the majority vote among them:

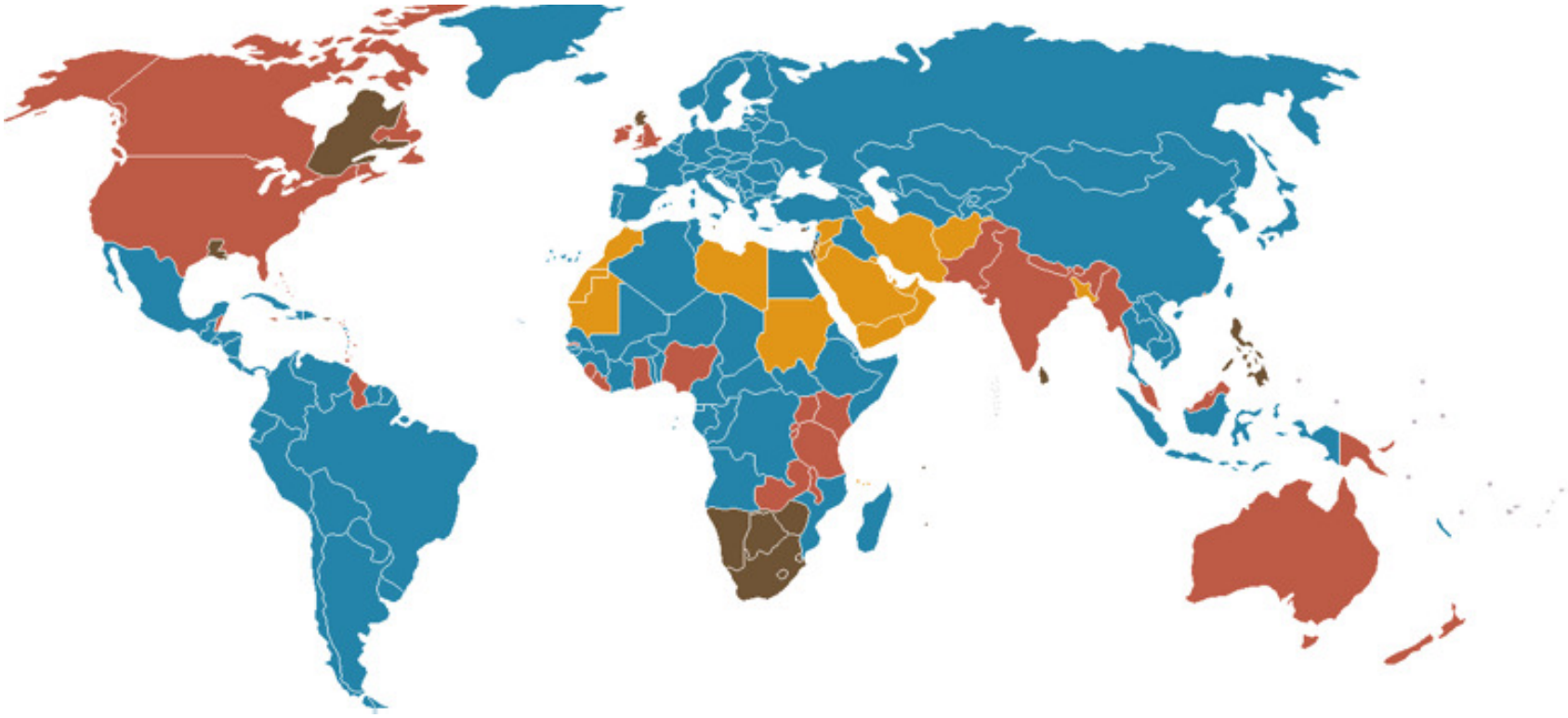
$$f(x) \Leftarrow \frac{1}{k} \sum_{i=1}^{k} f(x_i) \tag{48}$$

- k is usually an odd number to facilitate tie breaking.
- Two degrees of freedoms: n points and k neighbours.

k—Nearest Neighbour Classifiers

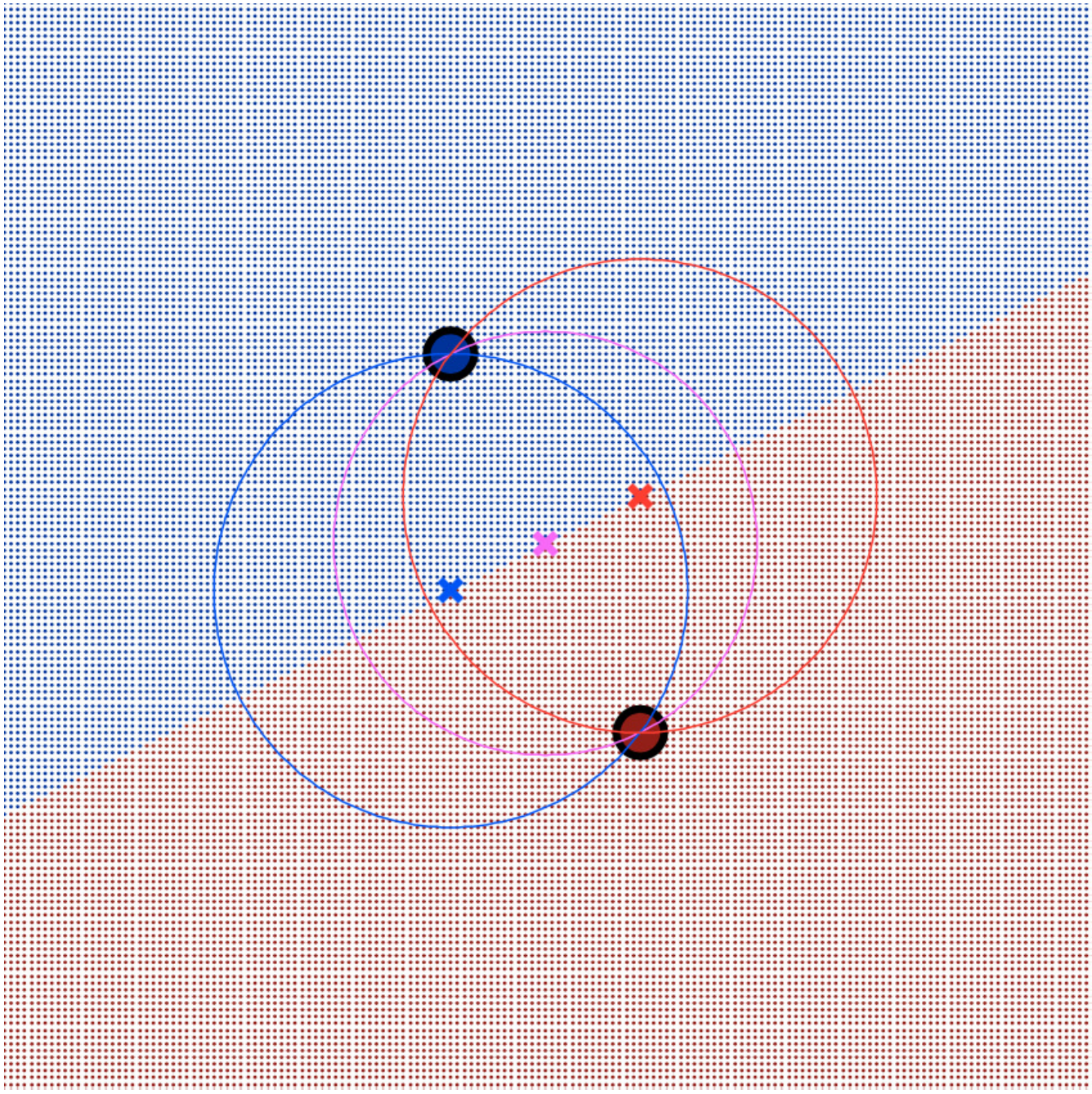


Analogy: Legal Systems

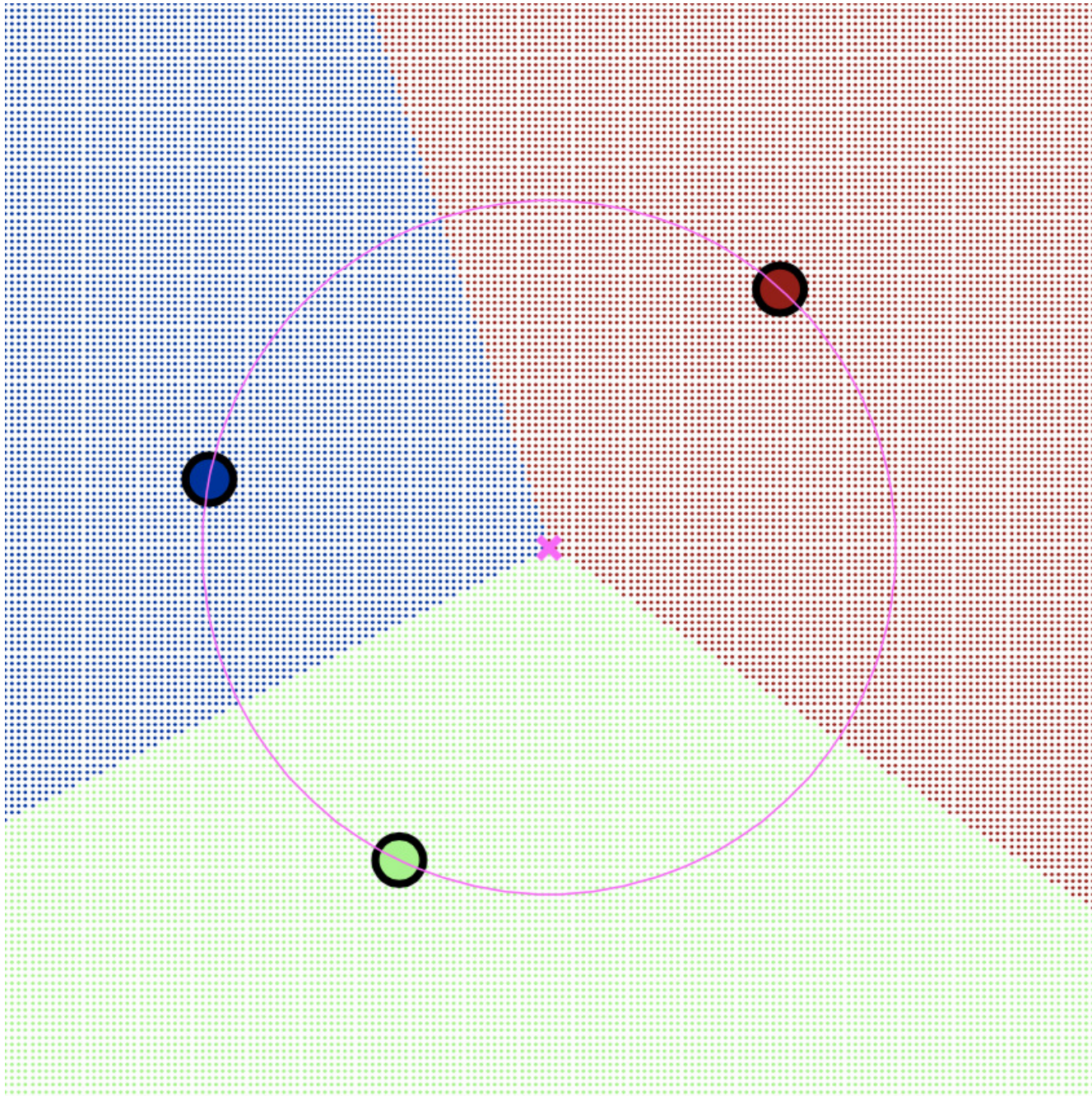


- Civil law / Codified law / Continental law
  "organized laws that attempt to cover exhaustively the various legal domains"
- Common law / Case law / Precedent law

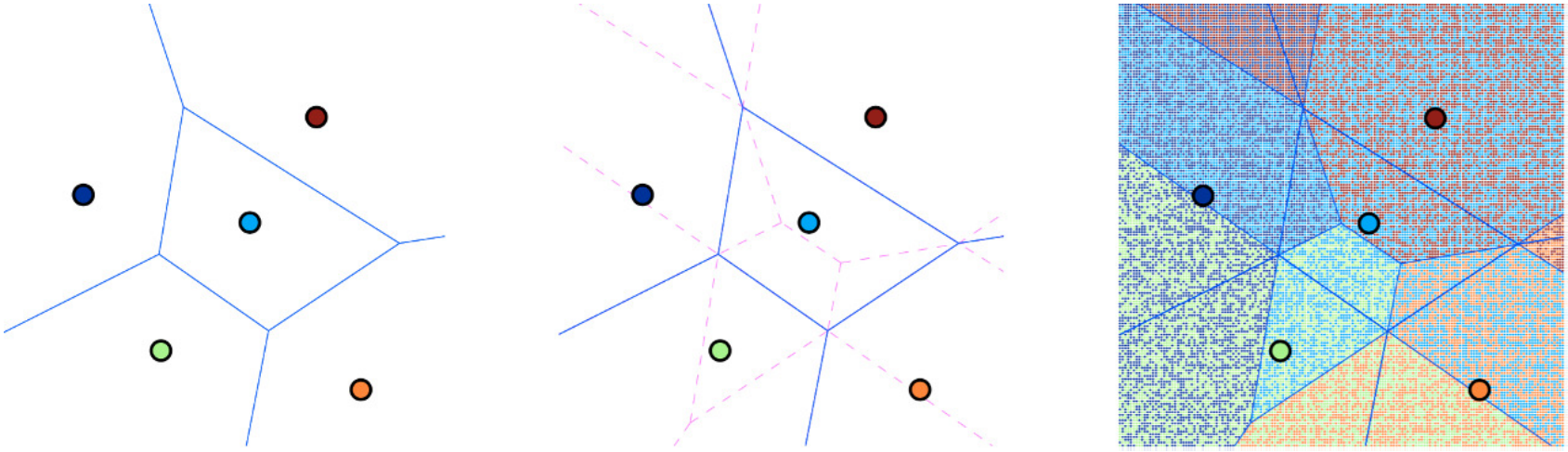
1-NN Decision Regions for 2 Exemplars



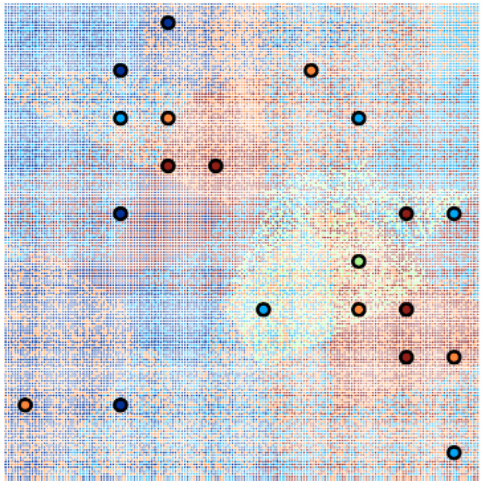
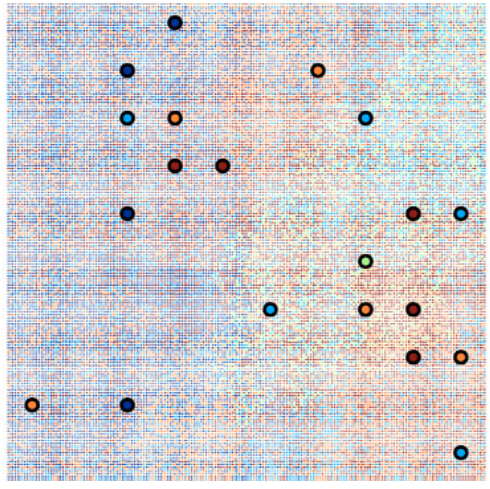
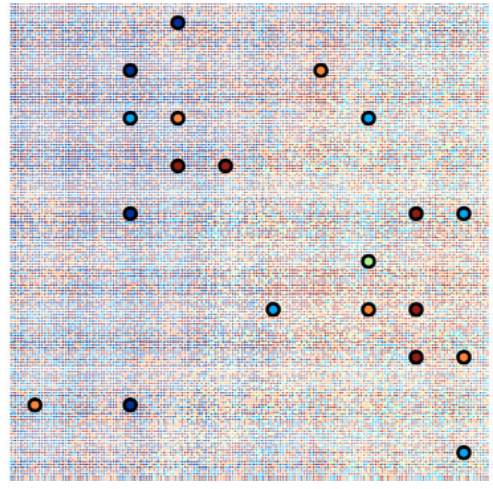
1-NN Decision Region for 3 Exemplars



1-NN Voronoi Diagram → 2-NN Subdivision



k-NN Decision Regions: $k = 3, 5, 7$

Nearest-Neighbour Rule: Pros and Cons

1. Also known as:
   - instance-based learning
   - memory-based learning
   - exemplar-based learning
   - lazy learning
2. What do you do during training?
   - Nothing! The only $O(0)$ algorithm in this class!
   - But prediction during testing is $O(n)$.
3. 1-NN perfectly separates training data $\rightarrow$ low bias high variance
4. As k increases, we increase bias and decrease variance. $k \rightarrow n$?
5. Can also output probabilities when $k > 1$
6. Easily adapt to other forms of $f(x)$
7. OK in low dimensions d, but not so good in high dimensions
8. A type of non-parametric method

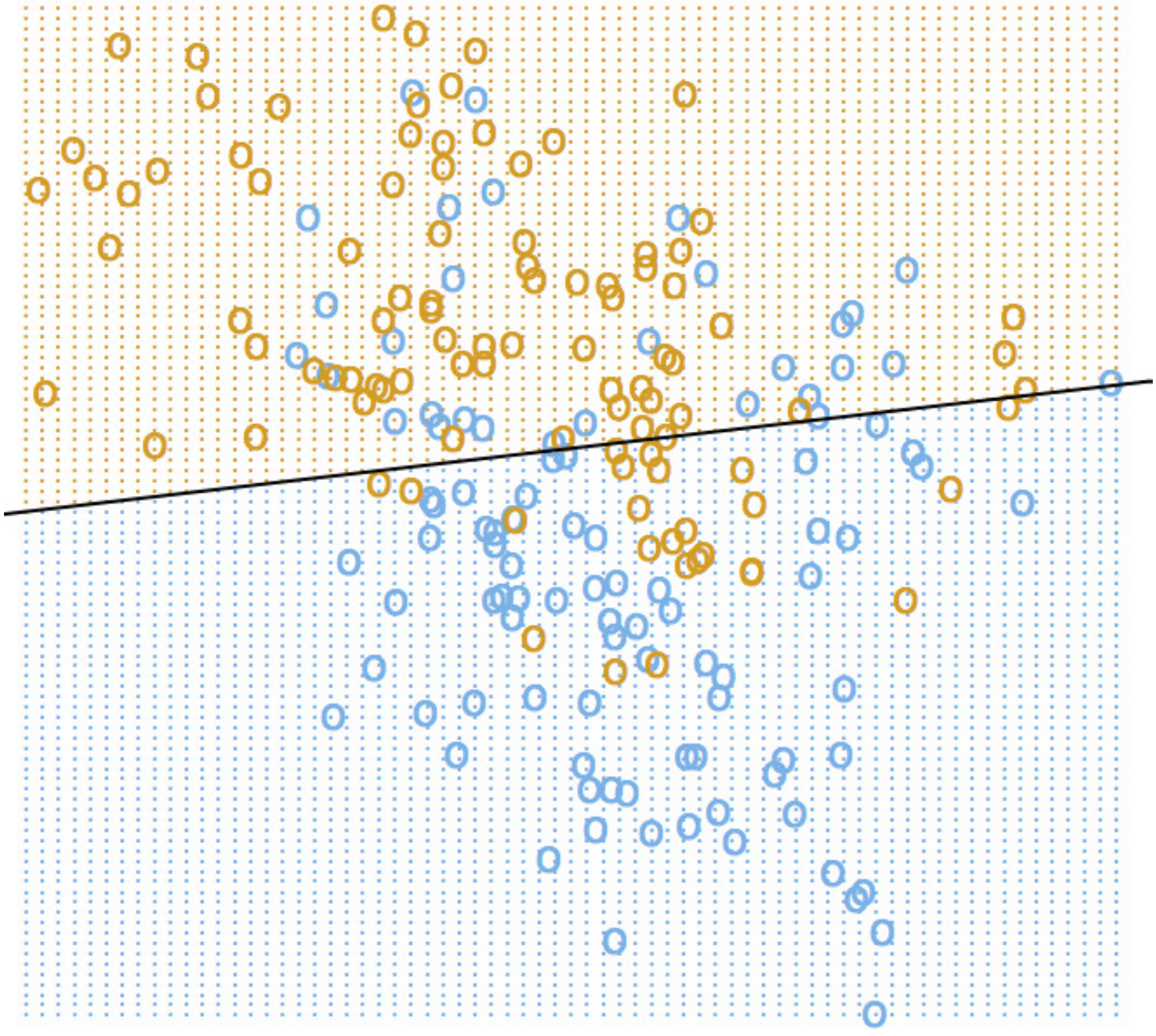
Parametric vs. Non-Parametric Methods

- Parametric
  # parameters is independent of n — # training instances

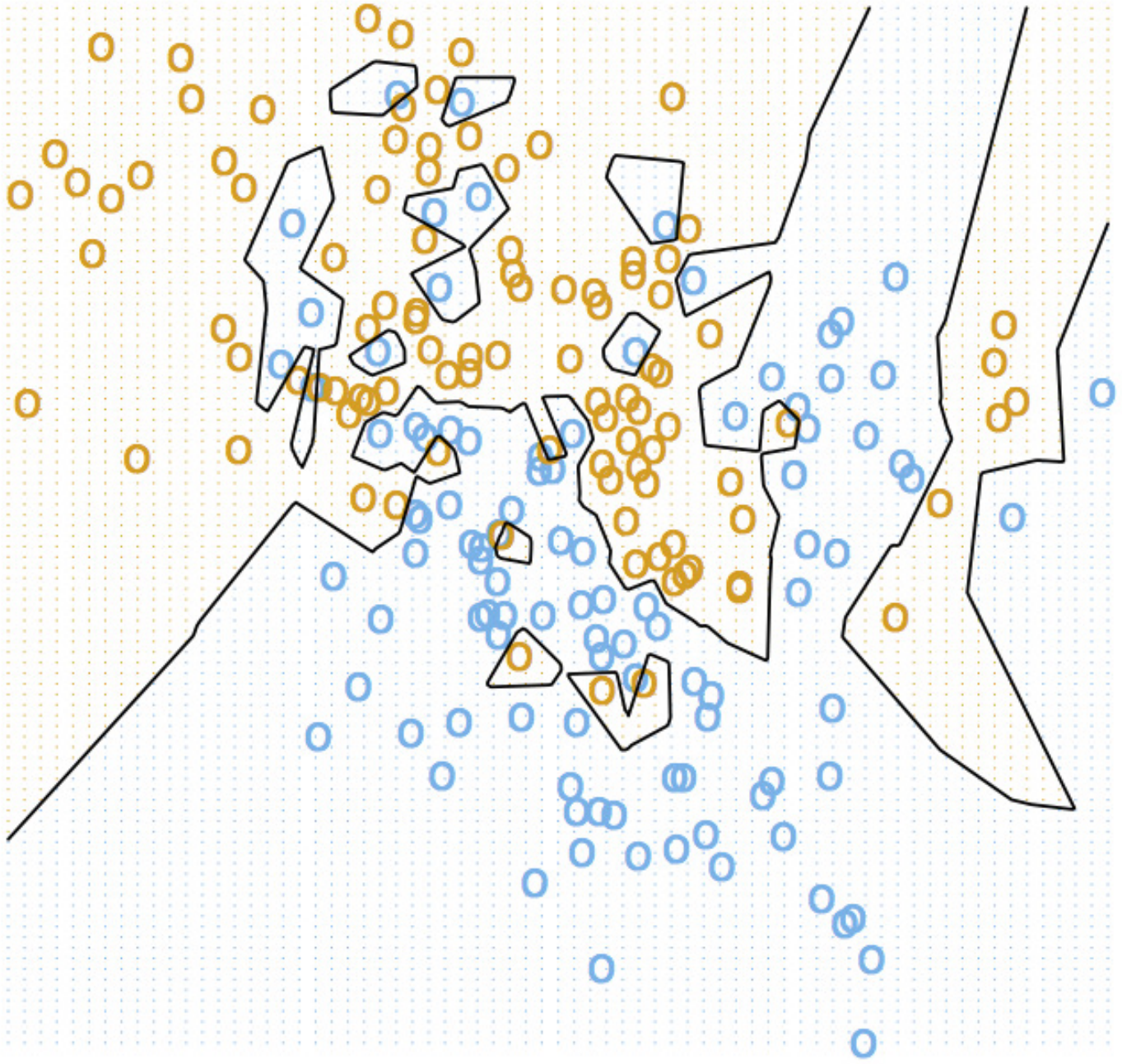
- Non-parametric
  # parameters grows with n

- Studied so far:
  - LS regression?
  - LDA?
  - SVM?
  - Kernel SVM?
  - kNN?
  - $\vdots$

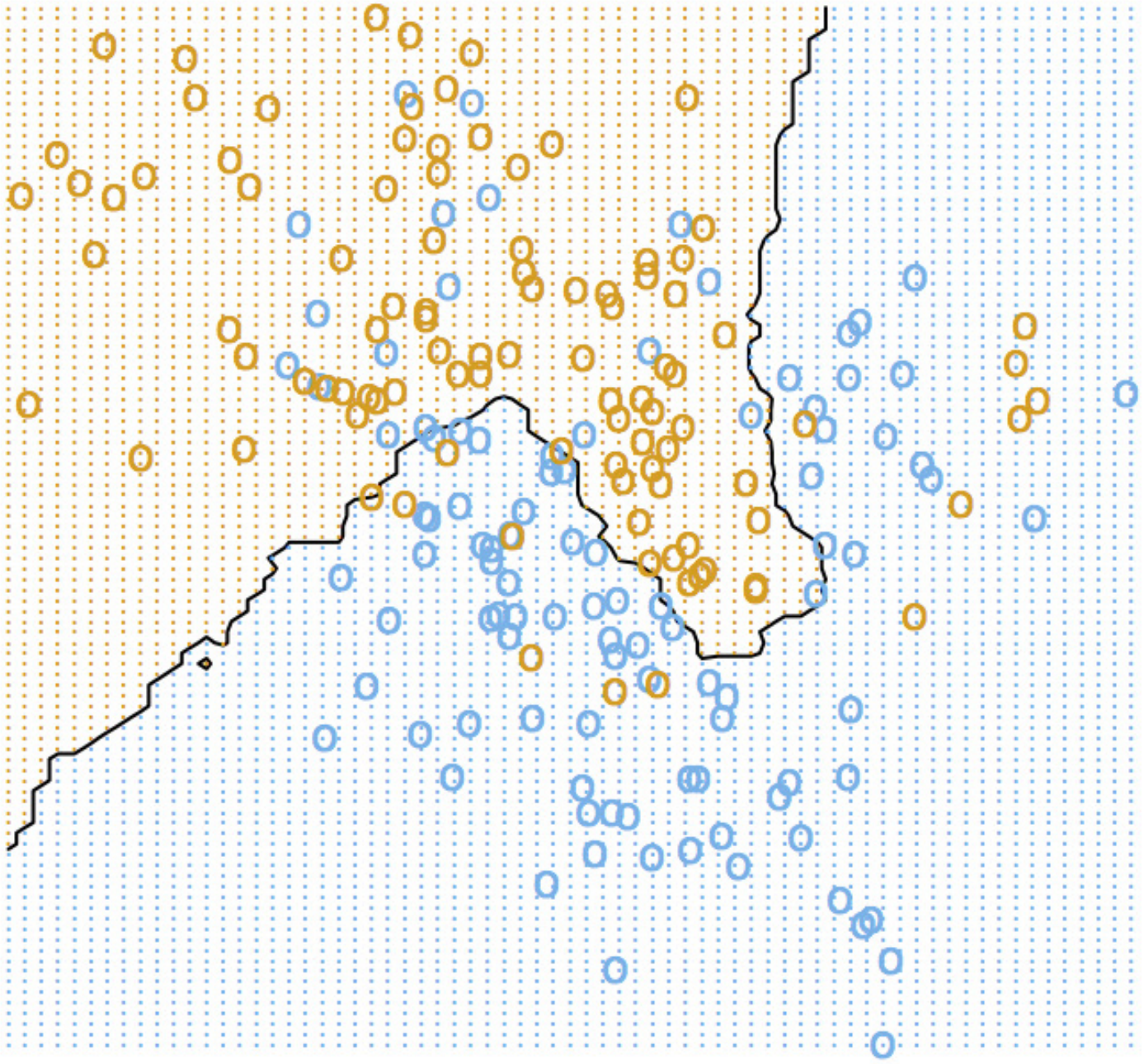
Linear Classifier



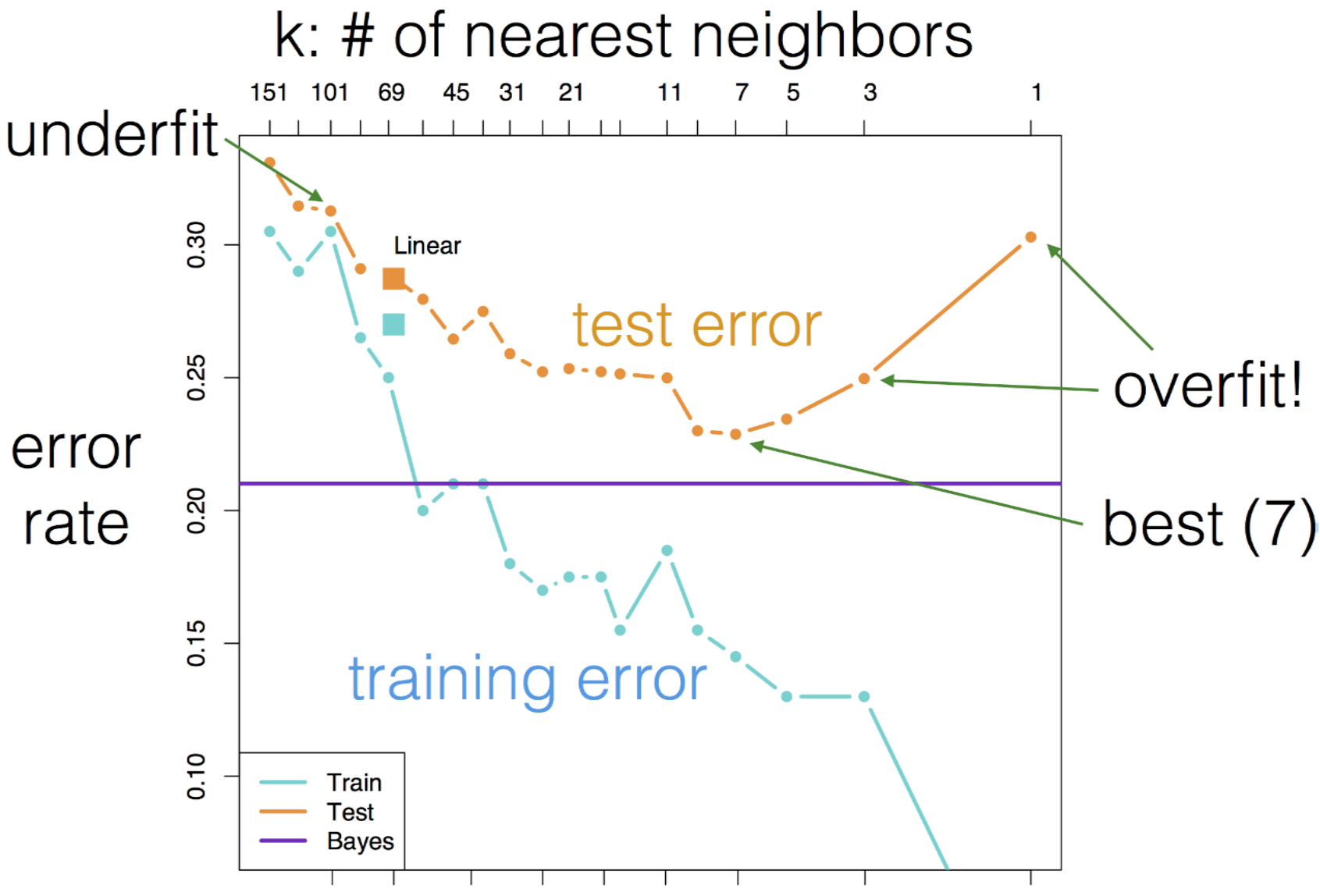
Nearest Neighbour Classifier: Non-Parametric



k-Nearest Neighbour Classifier: $k = 15$



Choose k: Overfitting vs. Underfitting



Theoretical Results about k-NN at the Limit

$$\varepsilon^*(x): \quad \text{error of optimal prediction} \tag{49}$$

$$\varepsilon^*_{\text{k-NN}}(x): \quad \text{error of k-NN} \tag{50}$$

- **Theorem(Cover & Hart, 1967):** $\lim_{n\to\infty} \varepsilon_{1-NN} \leq 2\varepsilon^*$

  If you have a lot of data, nearest neighbours can work very well. You will be at most worse by a factor of 2 from the optimal.

- **Theorem(Fix & Hodges, 1951):** $\lim_{n\to\infty, k\to\infty, \frac{k}{n}\to 0} \varepsilon_{k-NN} = \varepsilon^*$

  If you have a lot of data, and can afford many neighbours, k-NN can work very well, close to the optimal.

Application of k-NN to Image Understanding

Label Transfer

Tags: Sky, Water, Beach, Sunny, …
Time: 1pm, August, 2006, …
Location: Italy, Greece, Hawaii …
Photographer: Flickrbug21, Traveller2

A Large Geo-Tagged Training Image Set

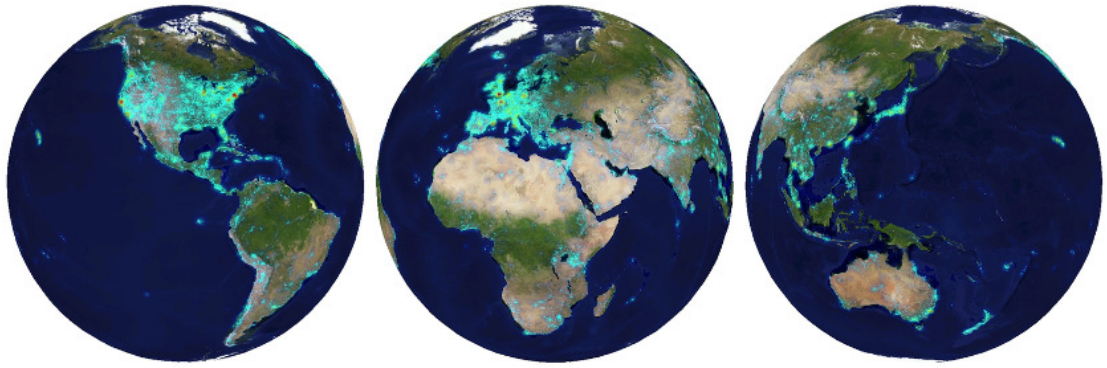
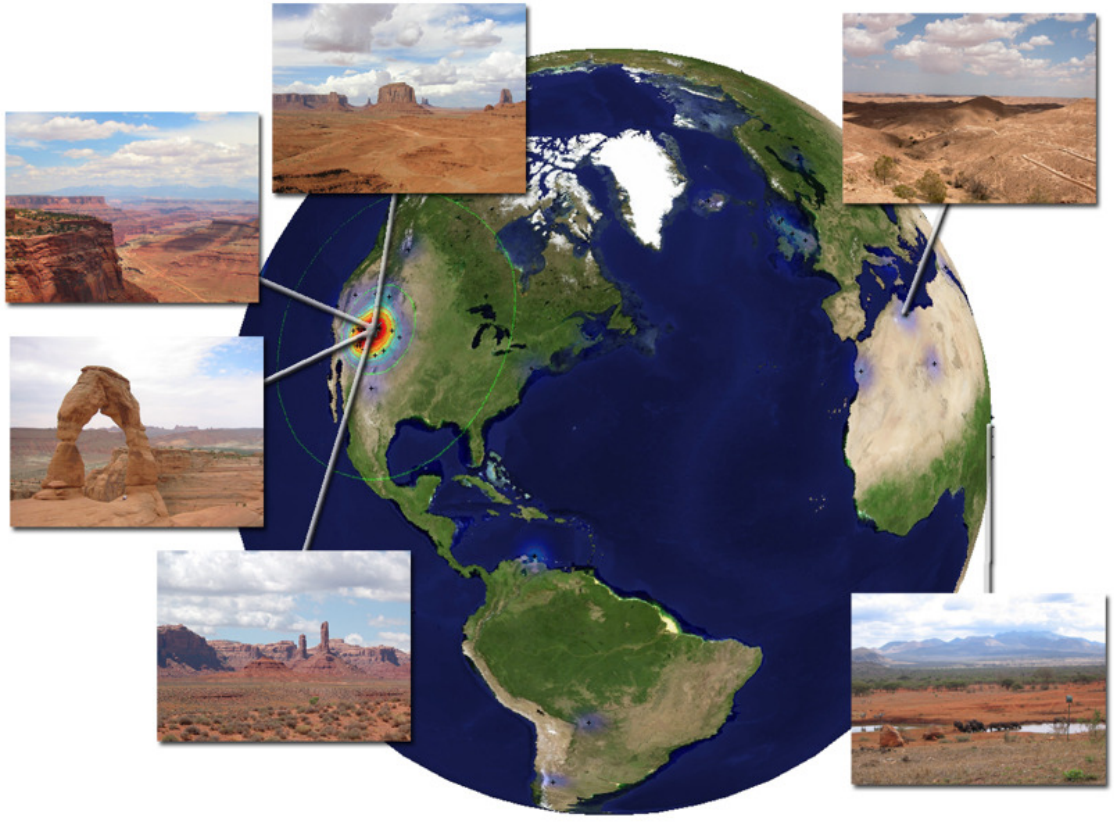


Image → GPS

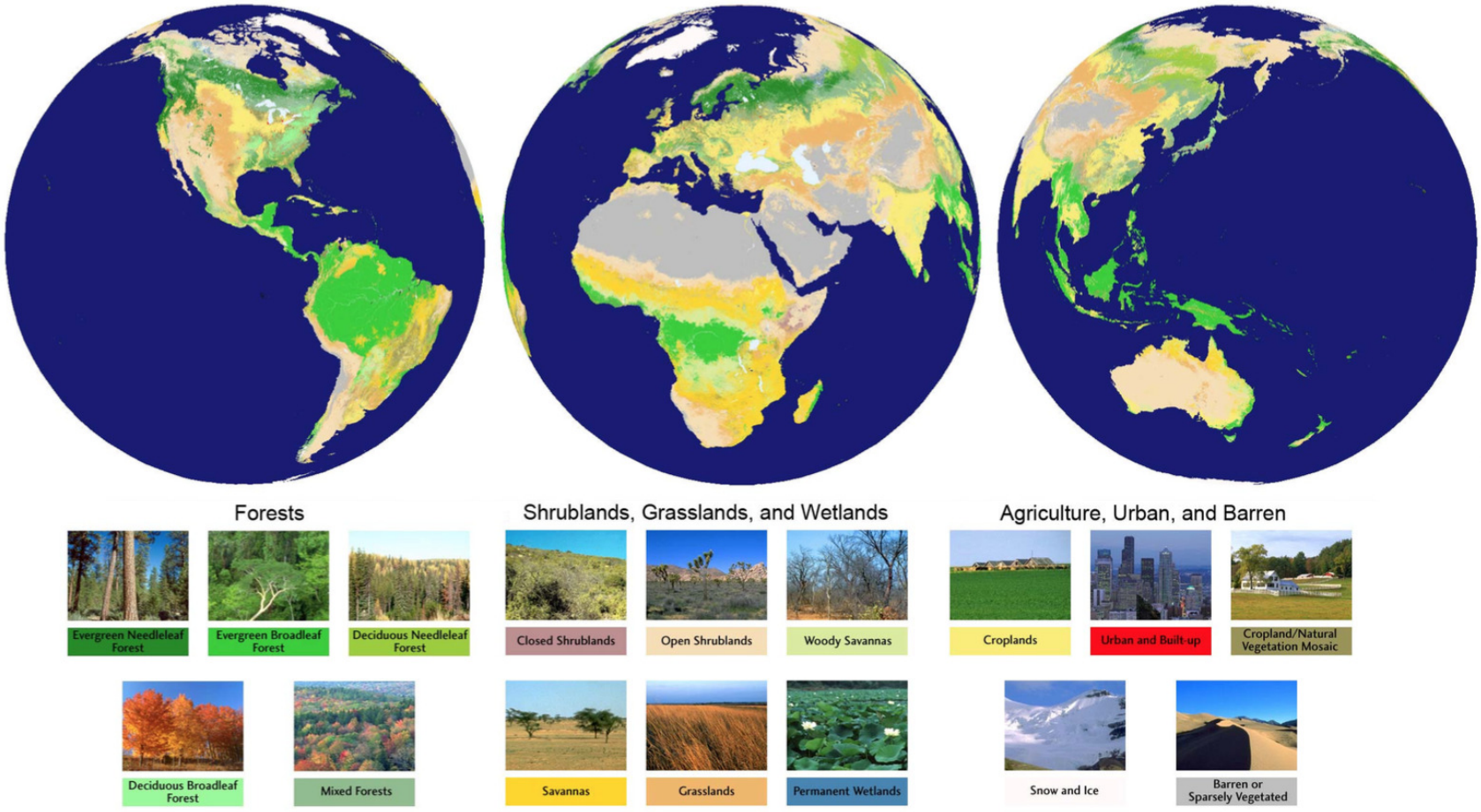
Query Photograph



Visually Similar Scenes

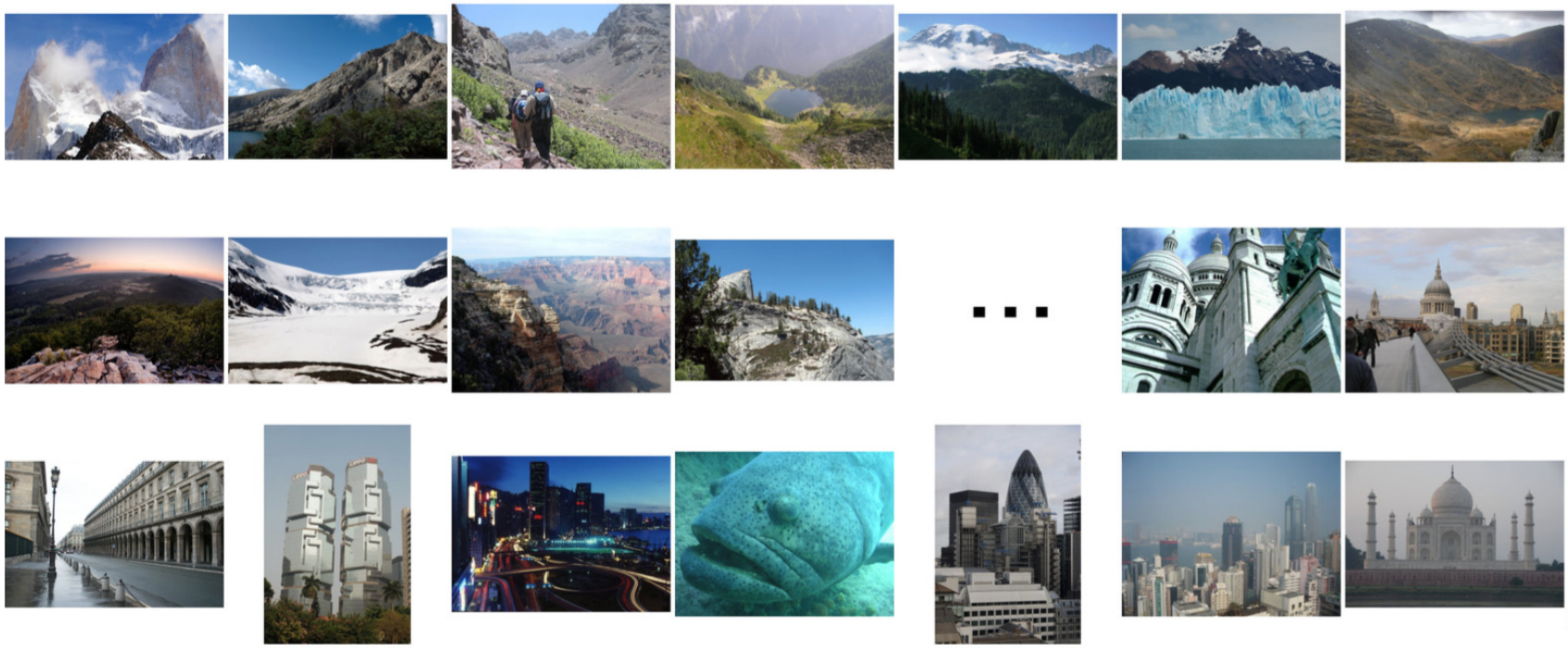
- *IM2GPS: Estimating Geographic Information From A Single Image*, James Hays and Alexei A. Efros, CVPR 2008.

Land Cover Classification



Population Density Ranking

Elevation Gradient Magnitude Ranking



k-NN Advantages and Disadvantages

- Training is very fast
- Learn complex target functions easily
- Don't lose information

- Slow at query time
- Lots of storage
- Easily fooled by irrelevant attributes

Curse of Dimensionality

1. Nearest neighbor is easily misled in high dimensions

2. Easy problems in low-D are hard in High-D
   "If we could see in high dimensions, there would be no need for Machine Learning"

3. Low-D intuitions don't apply to High D
   Everything is far from everything else

4. Examples:
   - Points on hyper-grid
   - Hypersphere
   - Hypercube vs. hypersphere
   - High-dimensional Gaussian

The Geometry of High-Dimensional Spaces

$$\text{volume of outer ball} \propto r^d \tag{51}$$

$$\text{volume of inner ball} \propto (r-\varepsilon)^d \tag{52}$$

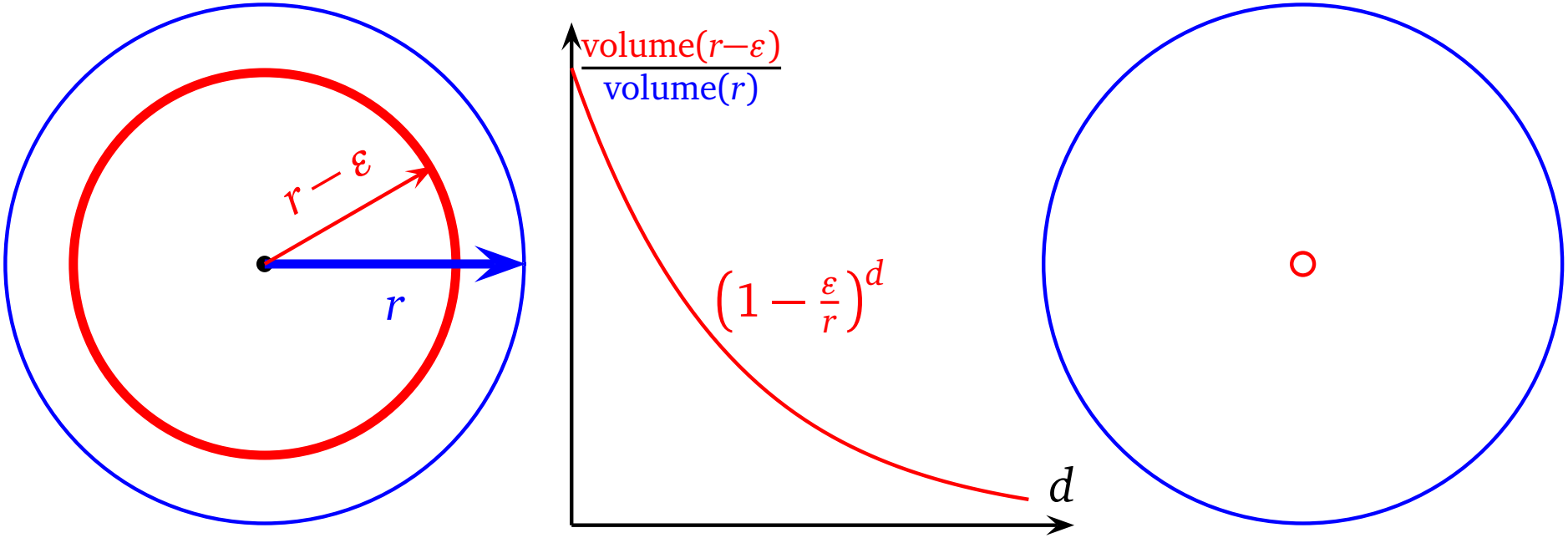
$$\text{ratio of inner-to-outer ball volume} = \tag{53}$$

$$\frac{(r-\varepsilon)^d}{r^d} = \left(1-\frac{\varepsilon}{r}\right)^d \approx \exp\left(-\frac{\varepsilon d}{r}\right) \to 0, \quad \text{as } d \to \infty \tag{54}$$

$\frac{\varepsilon}{r}$	$d=2$	$d=10$	$d=100$	$d=1000$
0.01	0.9801	0.9044	0.3660	$0.99^{1000} \approx 0.0000$
0.25	0.5625	0.0563	0.0000	0.0000
0.5	0.2500	0.0010	0.0000	0.0000

Uniform Distribution in High-Dimensional Balls

$$\lim_{d\to\infty} \frac{\text{volume}(r-\varepsilon)}{\text{volume}(r)} = 0$$

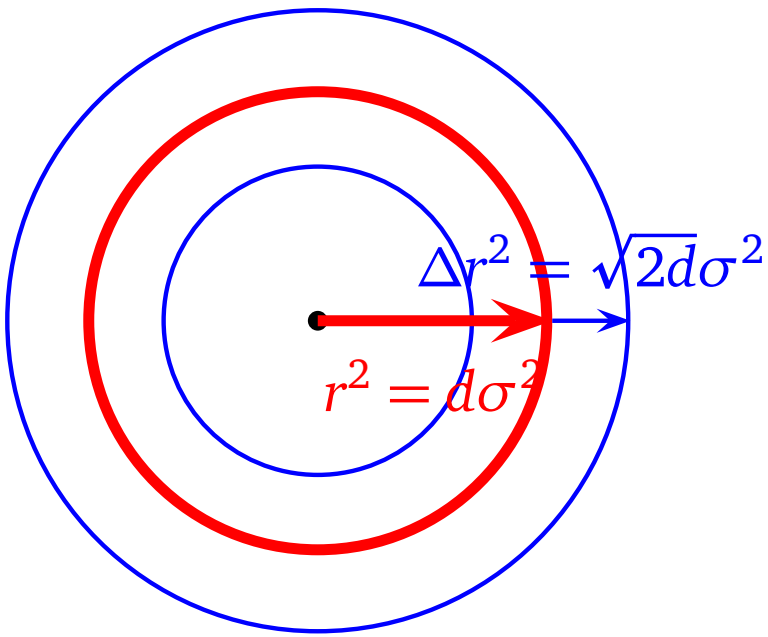


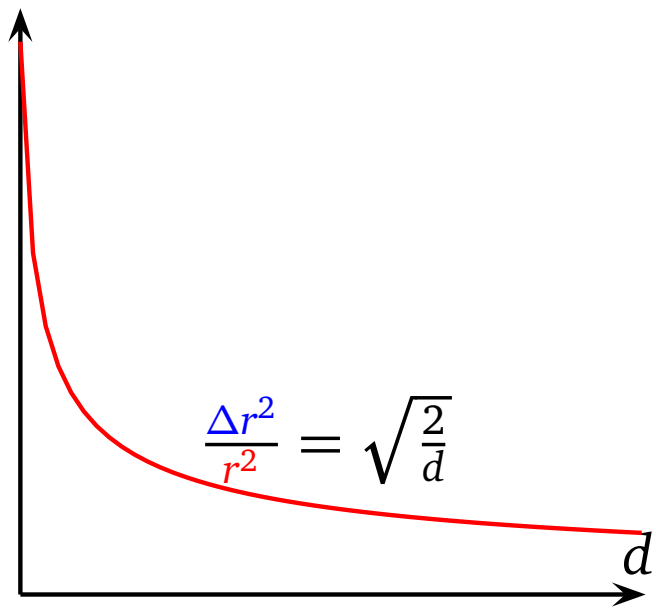
- Random points from uniform distributions in ball: nearly all are in outer shell.

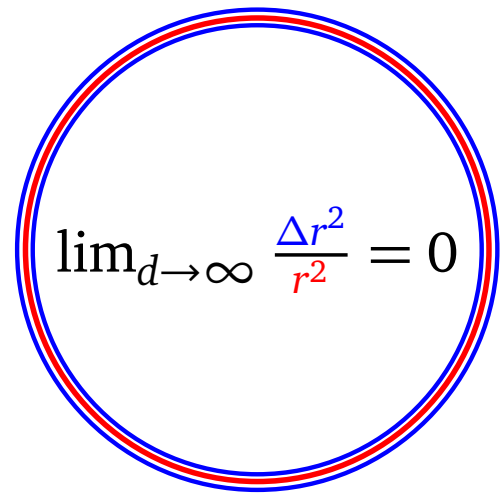
High-Dimensional Gaussians

$$X \sim \mathcal{N}(0, \sigma^2), \quad p(x) = \frac{1}{(2\pi\sigma)^{\frac{d}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{d} x_i^2\right) \tag{55}$$

$$r^2 = \sum_{i=1}^{d} x_i^2 \sim \mathcal{N}(d\sigma^2, (\sqrt{2d}\sigma^2)^2) \tag{56}$$







- Random points from Gaussian distributions in ball: nearly all are in some thin shell.
- Curse of Dimensionality: All k-NNs are actually not that near!

How to Deal with Curse of Dimensionality?

- Problem: D is on the order of or greater than N
- Remedy 1: Make N larger → more data.
- Remedy 2: Make D smaller → better features or distances

Application: Photo Editing

Photo Editing Mask

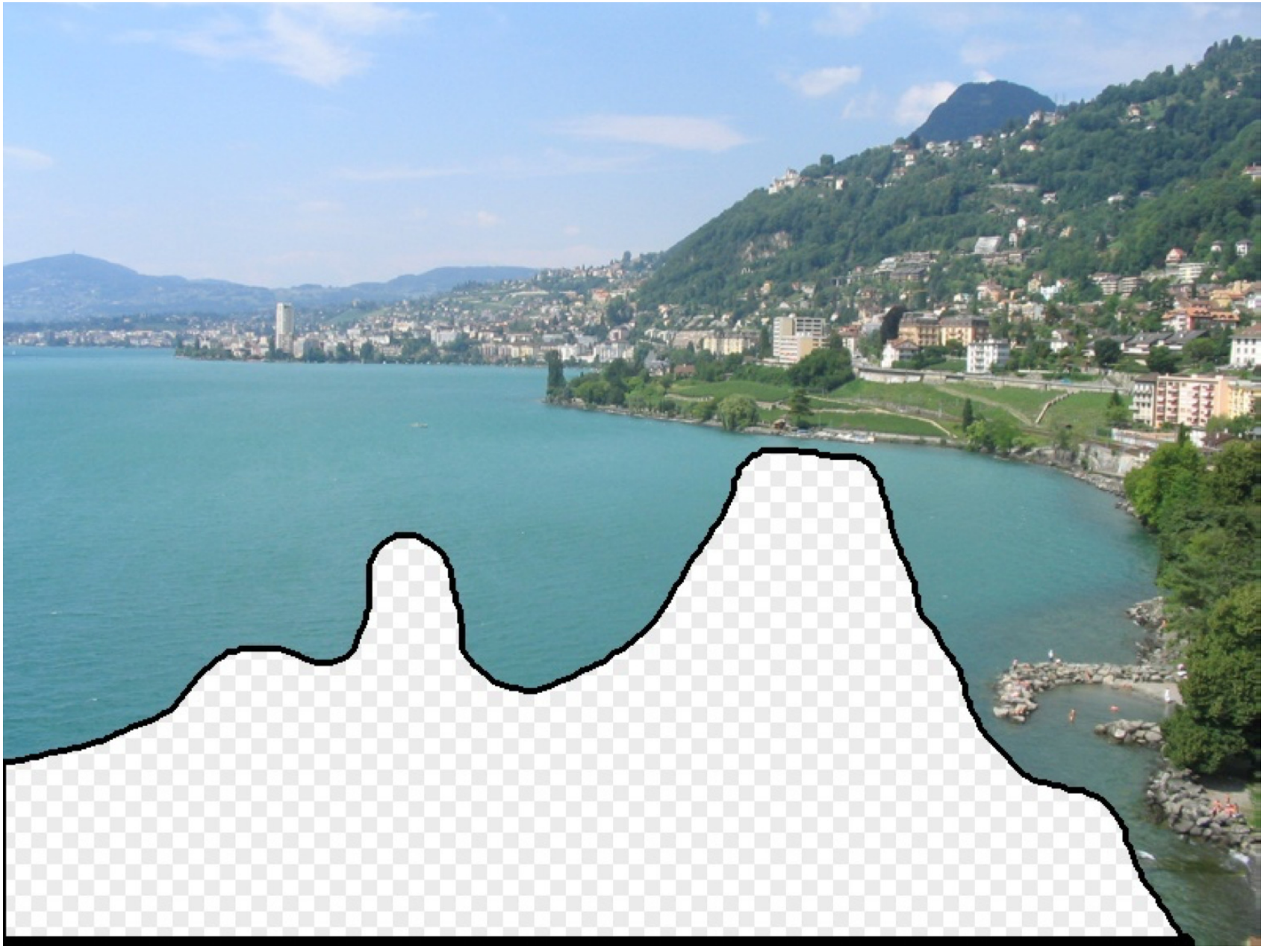
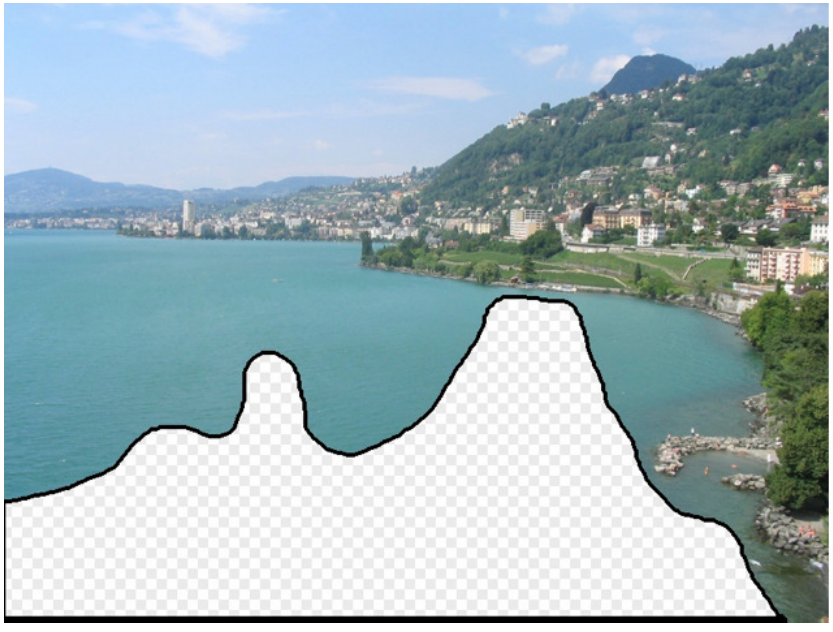


Photo Editing Target and Source Pair



- *Scene Completion Using Millions of Photographs*, James Hays and Alexei A. Efros, SIGGRAPH 2007.

Photo Editing Result

Nearest Neighbours from 20K Images

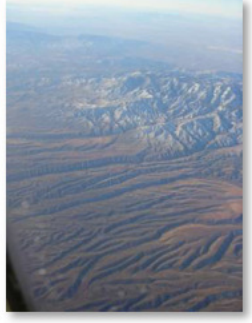




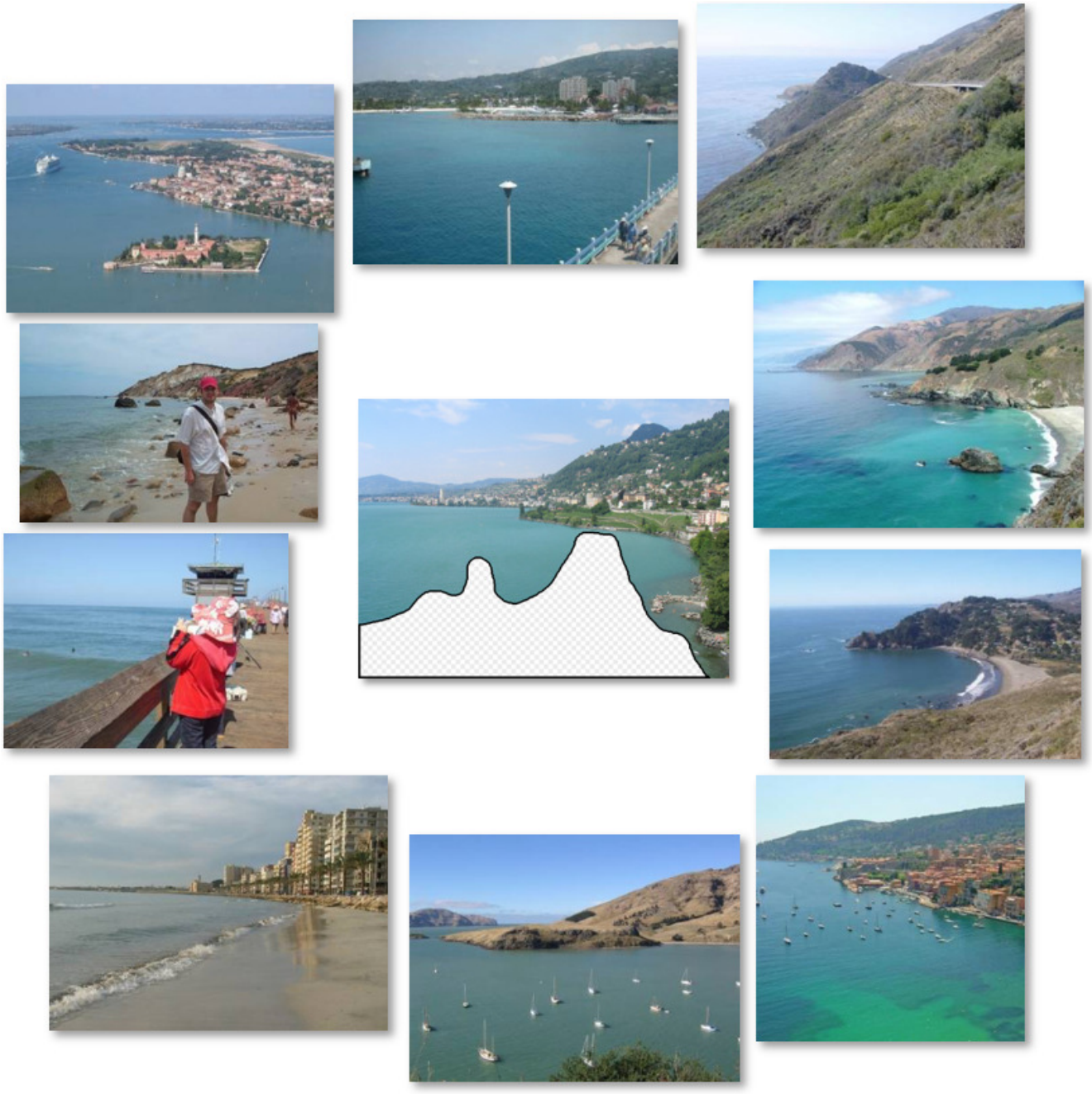








Nearest Neighbours from 2 Million Images



Data Reduction: Lower Image Resolution

Data Reduction: Bag-of-Words Models

- Orderless document representation
- Frequencies of words from a dictionary
- *Introduction to modern information retrieval*
  Gerard Salton and Michael J. McGill, 1983

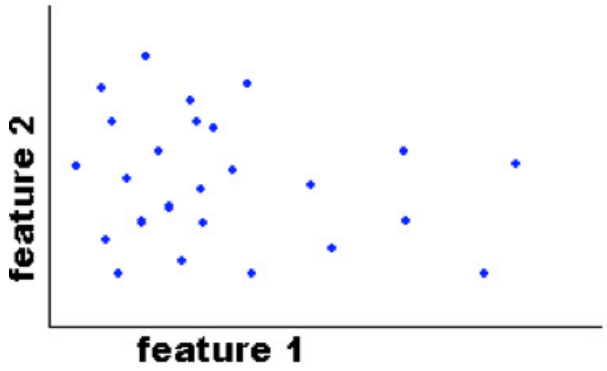
Bag-of-Words Models - State of the Union Address

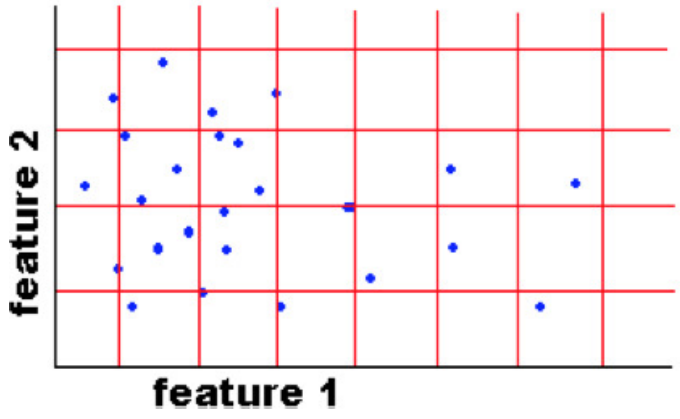
Bag-of-Words Models - Soviet Missiles in Cuba

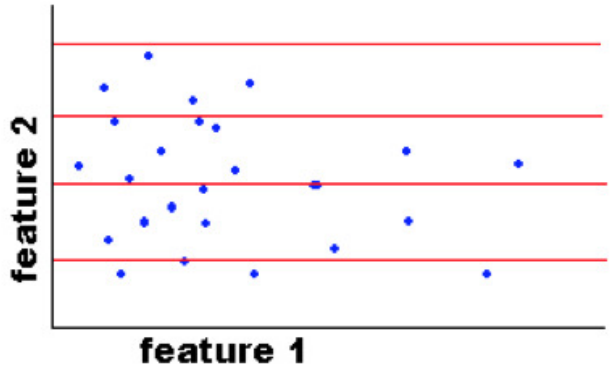
Bag-of-Words Models - Declaration of War

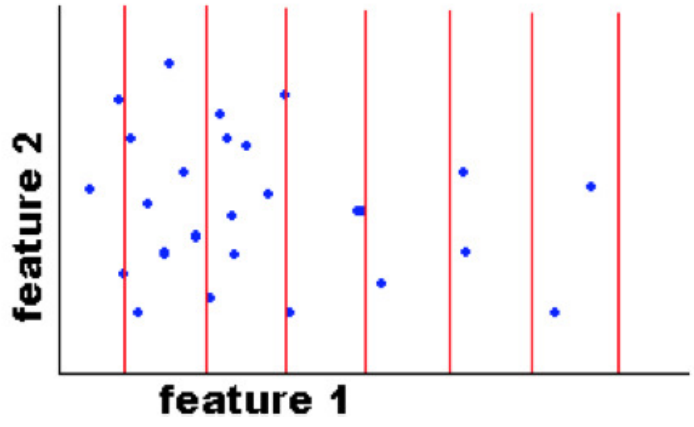
Data Reduction: Joint vs. Marginal Histograms

Images from Dave Kauchak









Common Distance Function: Minkowski Distances

$$D_p(x,z) = \left(\sum_{t=1}^{d} |x_t - z_t|^p \right)^{\frac{1}{p}} = \|x - z\|_p \tag{57}$$

- $p = 2$: Euclidean distance

$$D_2(x,z) = \sqrt{\sum_{t=1}^{d} (x_t - z_t)^2} \tag{58}$$

- $p = 1$: Manhattan / cityblock distance

$$D_1(x,z) = \sum_{t=1}^{d} |x_t - z_t| \tag{59}$$

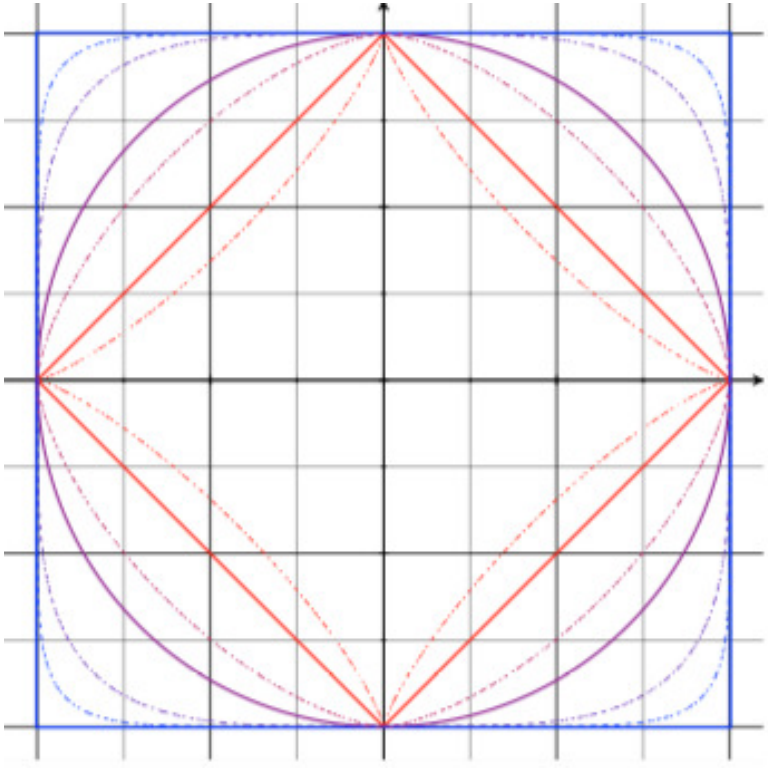
- $p = \infty$: Chebyshev distance

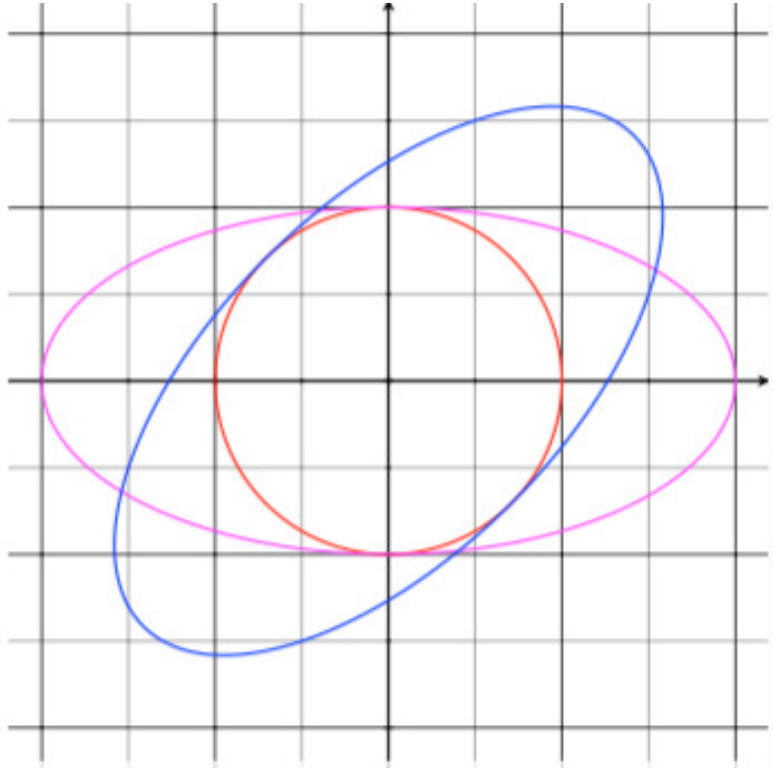
$$D_\infty(x,z) = \max_t |x_t - z_t| \tag{60}$$

- $p = 0$: Hamming distance or edit distance

$$D_0(x,z) = \sum_{t=1}^{d} (x_t - z_t)^0 = \sum_{t=1}^{d} 1(x_t = z_t) \tag{61}$$

Elliptical Distance: Mahalanobis Distance





- Minkowski distance at $p = 0.8, 1, 1.5, 2, 4, 8, \infty$.
- Mahalanobis Distance is the Euclidean distance weighted by the inverse of the data covariance matrix:

$$D_M(x, z) = \sqrt{(x - z)' \Sigma^{-1} (x - z)} \tag{62}$$

From Kernels to Distances

- Kernel

$$K(x,z) = \phi(x) \cdot \phi(z) \tag{63}$$

- Euclidean distance

$$\begin{aligned} D(x,z) &= \|\phi(x) - \phi(z)\|^2 && (64)\\ &= \sqrt{\phi(x)\cdot\phi(x) - 2\phi(x)\cdot\phi(z) + \phi(z)\cdot\phi(z)} && (65)\\ &= \sqrt{K(x,x) - 2K(x,z) + K(z,z)} && (66) \end{aligned}$$

- Cosine distance

$$\begin{aligned} D(x,z) &= 1 - \cos(\theta) && (67)\\ &= 1 - \frac{\phi(x)\cdot\phi(z)}{\sqrt{(\phi(x)\cdot\phi(x))(\phi(z)\cdot\phi(z))}} && (68)\\ &= 1 - \frac{K(x,z)}{\sqrt{K(x,x)K(z,z)}} && (69) \end{aligned}$$

Summary

- Don't underestimate the humble Nearest Neighbor
- Not glamorous, but works remarkably well
- Always a good baseline

- A larger issue: still just interpolating training data
- Need to extrapolate, generalize