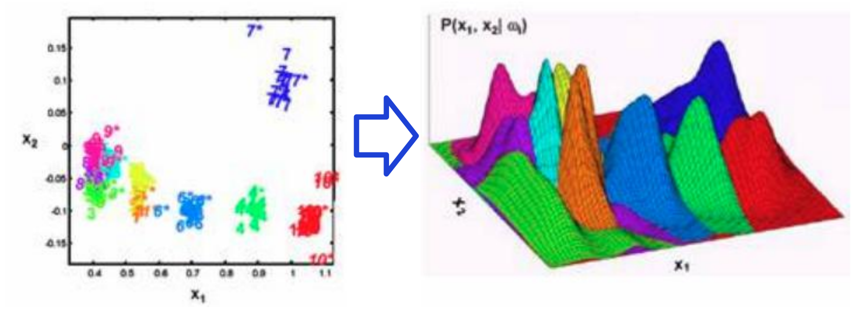
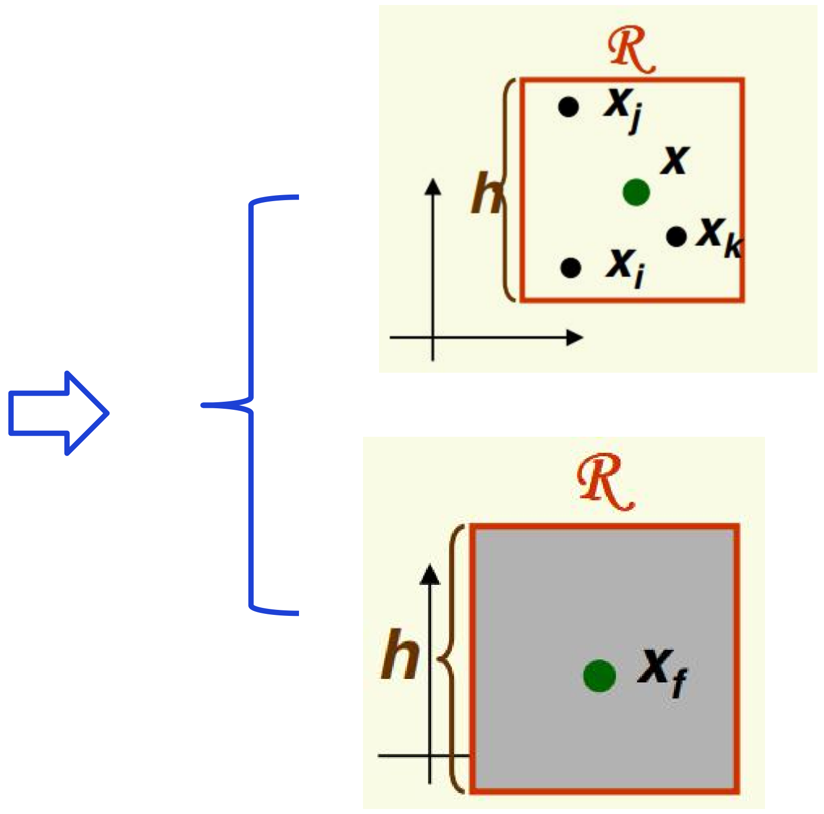
Rensselaer

Lecture 7: Non-Parametric Methods – KNN

Dr. Chengjiang Long
Computer Vision Researcher at Kitware Inc.
Adjunct Professor at RPI.
Email: **longc3@rpi.edu**

Recap Previous Lecture





Outline

- K–Nearest Neighbor Estimation
- The Nearest–Neighbor Rule
- Error Bound for K-Nearest Neighbor
- The Selection of K and Distance
- The Complexity for KNN
- Probabilistic KNN

Outline

- **K–Nearest Neighbor Estimation**
- The Nearest–Neighbor Rule
- Error Bound for K-Nearest Neighbor
- The Selection of K and Distance
- The Complexity for KNN
- Probabilistic KNN

k-Nearest Neighbors

- Recall the generic expression for density estimation

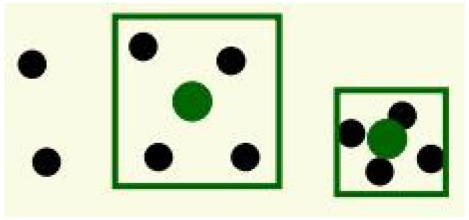
$$p(x) \approx \frac{k/n}{V}$$

- In Parzen windows estimation, we fix V and that determines k, the number of points inside V

- In k–nearest neighbor approach we fix k, and find V that contains k points inside

k-Nearest Neighbors

- kNN approach seems a good solution for the problem of the “best” window size
  - Let the cell volume be a function of the training data
  - Center a cell about x and let it grows until it captures k samples
  - k are called the k nearest neighbors of x

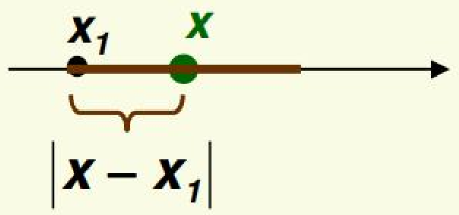


- Two possibilities can occur:
  - Density is high near x; therefore the cell will be small which provides a good resolution
  - Density is low; therefore the cell will grow large and stop until higher density regions are reached

k-Nearest Neighbor

- Of course, now we have a new question
  - How to choose k?
- A good "rule of thumb" is $k = \sqrt{n}$
  - Can prove convergence if n goes to infinity
  - Not too useful in practice, however
- Let's look at 1-D example
  - we have one sample, i.e. n = 1

$$p(x) \approx \frac{k/n}{V} = \frac{1}{2|x - x_1|}$$



- But the estimated p(x) is not even close to a density function:

$$\int_{-\infty}^{\infty} \frac{1}{2|x - x_1|} dx = \infty \neq 1$$

Nearest Neighbour Density Estimation

- Fix k, estimate V from the data.
- Consider a hypersphere centred on x and let it grow to a volume V* that includes k of the given n data points. Then

$$p(x) \approx \frac{k/n}{V}$$

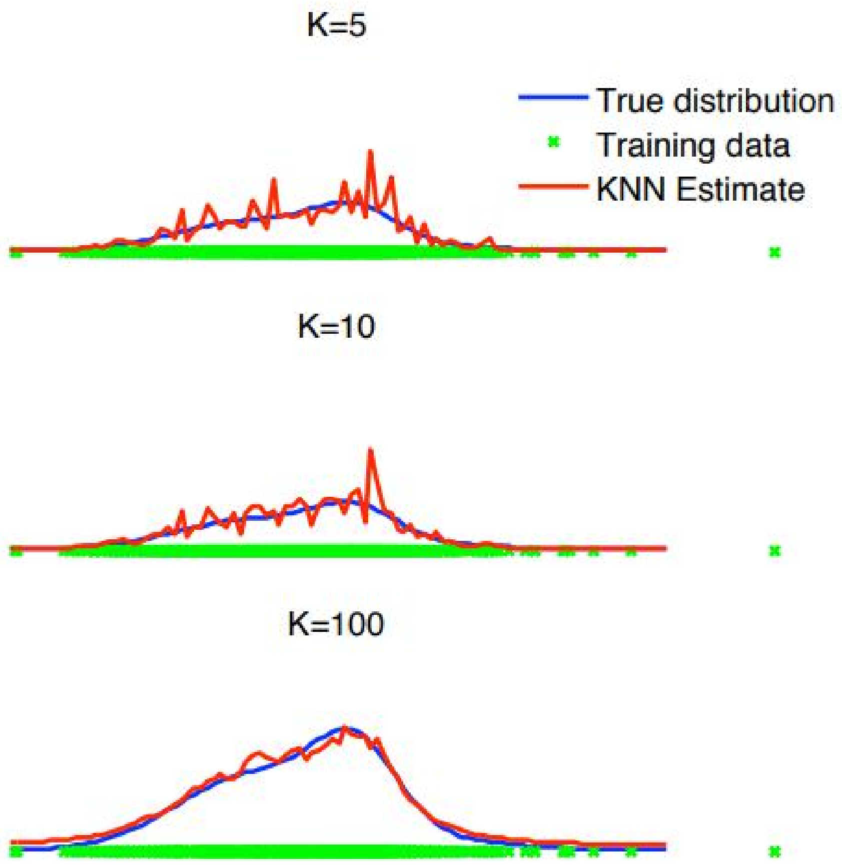


Illustration: Gaussian and Uniform plus Triangle Mixture Estimation (1)

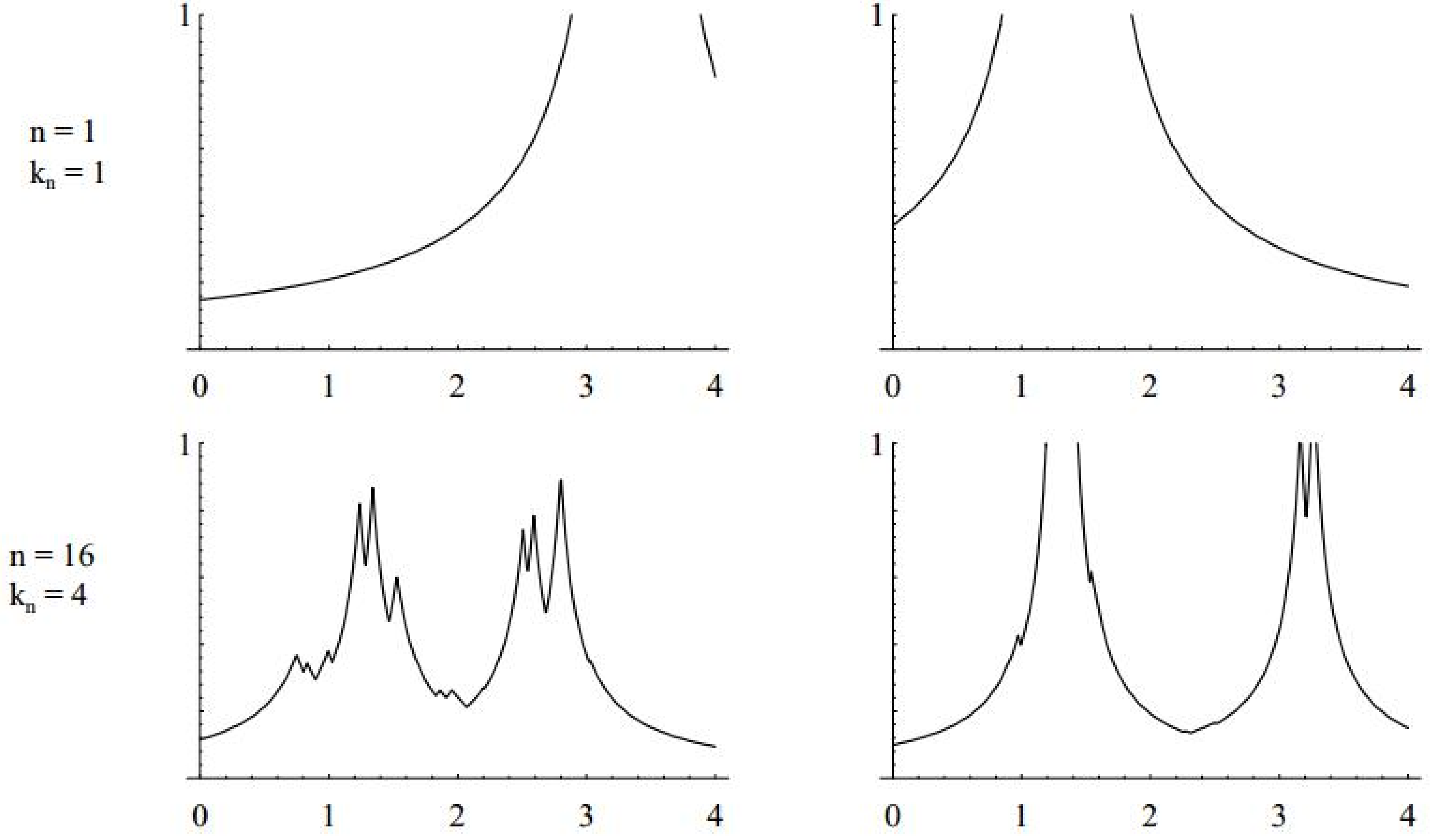
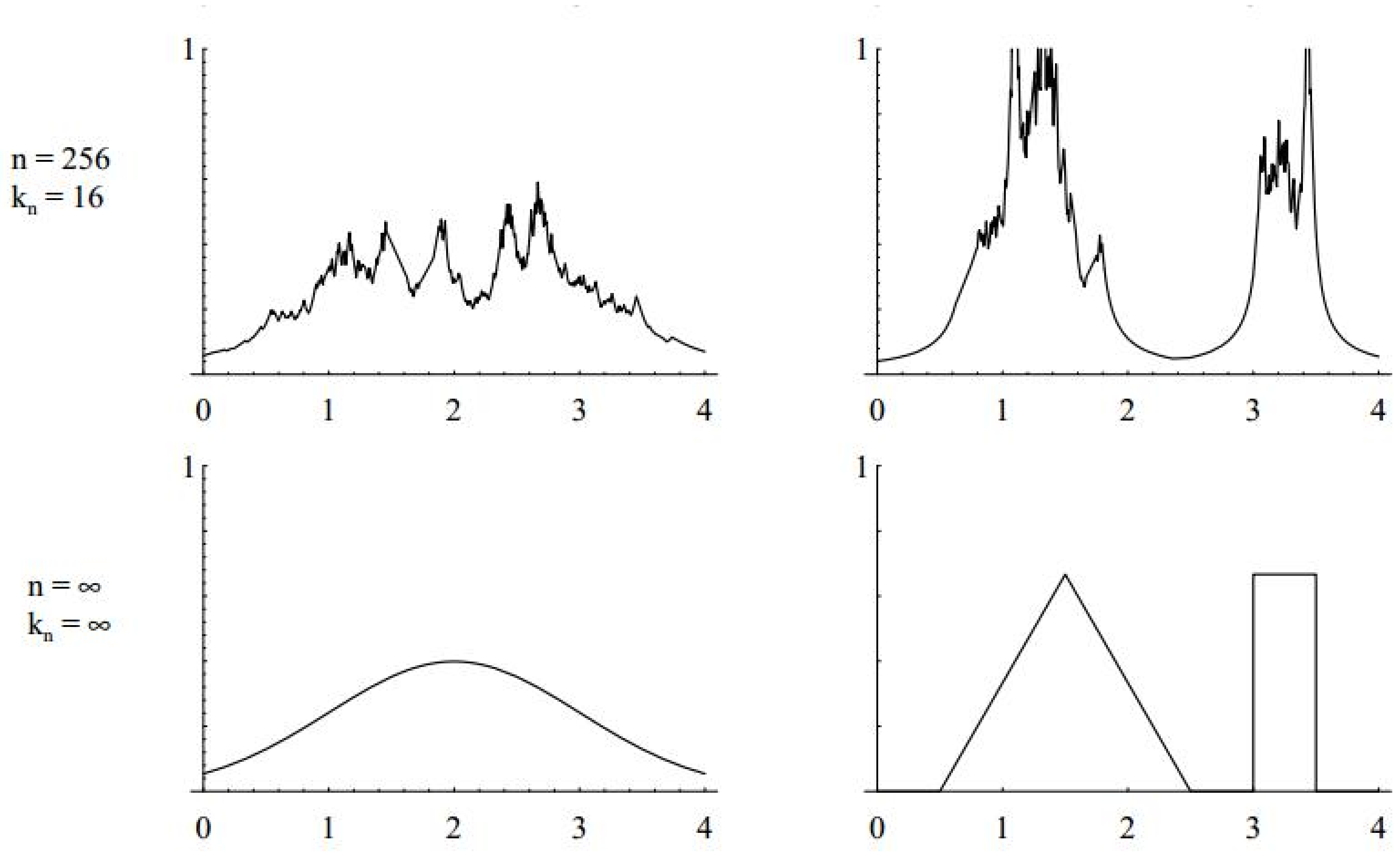


Illustration: Gaussian and Uniform plus Triangle Mixture Estimation (2)



k-Nearest Neighbor

- Thus straightforward density estimation p(x) does not work very well with kNN approach because the resulting density estimate
  ① Is not even a density
  ② Has a lot of discontinuities (looks very spiky, not differentiable)

Notice in the theory, if infinite number of samples is available, we could construct a series of estimates that converge to the true density using kNN estimation. However this theorem is not very useful in practice because the number of samples is always limited

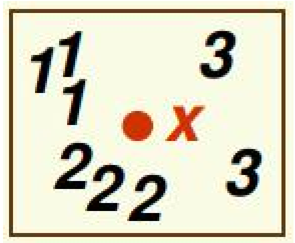
k-Nearest Neighbor

- However we shouldn't give up the nearest neighbor approach yet
- Instead of approximating the density $p(x)$, we can use kNN method to approximate the posterior distribution $P(c_i|x)$
  - We don't even need $p(x)$ if we can get a good estimate on $P(c_i|x)$

k-Nearest Neighbor

- How would we estimate P(ci|x) from a set of n labeled samples?
- Recall our estiamte for density: $p(x) \approx \frac{k/n}{V}$
- Let's place a cell of volume V around x and capture k samples.
  - ki samples amongest k labeled ci then

$$p(c_i, x) \approx \frac{k_i/n}{V}$$



- Using conditional probability, let's estimate posterior:

$$p(c_i \mid x) = \frac{p(x,c_i)}{p(x)} = \frac{p(x,c_i)}{\sum_{j=1}^{m} p(x,c_j)} \approx \frac{k_i/n}{V \sum_{j=1}^{m} \frac{k_j/n}{V}} = \frac{k_i}{\sum_{j=1}^{m} k_j} = \frac{k_i}{k}$$

k-Nearest Neighbor

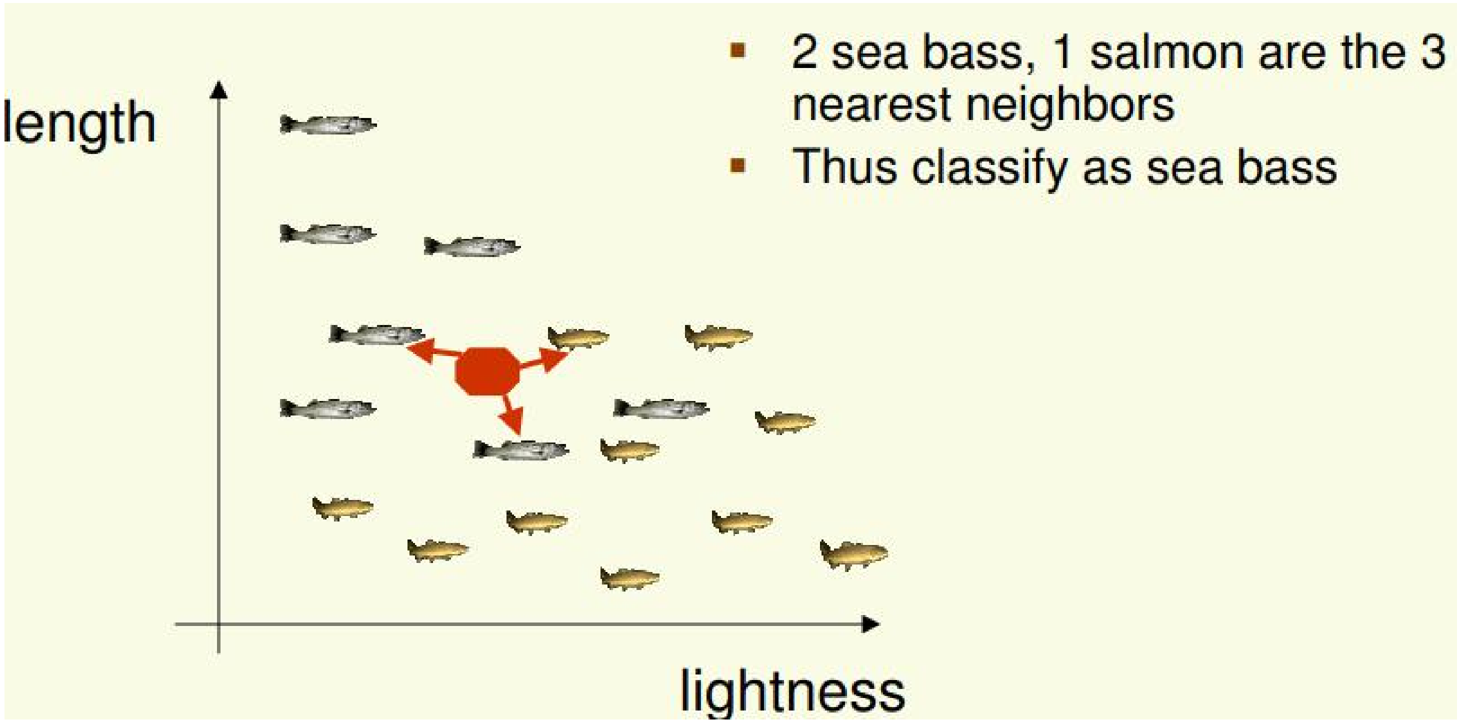
- Thus our estimate of posterior is just the fraction of samples which belong to class c_i:

$$p(c_i \mid x) = \frac{k_i}{k}$$

- This is a very simple and intuitive estimate
- Under the zero–one loss function (MAP classifier) just choose the class which has the largest number of samples in the cell
- Interpretation is: given an unlabeled example (that is x), find k most similar labeled examples (closest neighbors among sample points) and assign the most frequent class among those neighbors to x

k-Nearest Neighbor: Example

- Back to fish sorting
  - Suppose we have 2 features, and collected sample points as in the picture
  - Let k $= 3$



- 2 sea bass, 1 salmon are the 3 nearest neighbors
- Thus classify as sea bass

Outline

- K–Nearest Neighbor Estimation
- **The Nearest–Neighbor Rule**
- Error Bound for K-Nearest Neighbor
- The Selection of K and Distance
- The Complexity for KNN
- Probabilistic KNN

The Nearest–Neighbor Rule

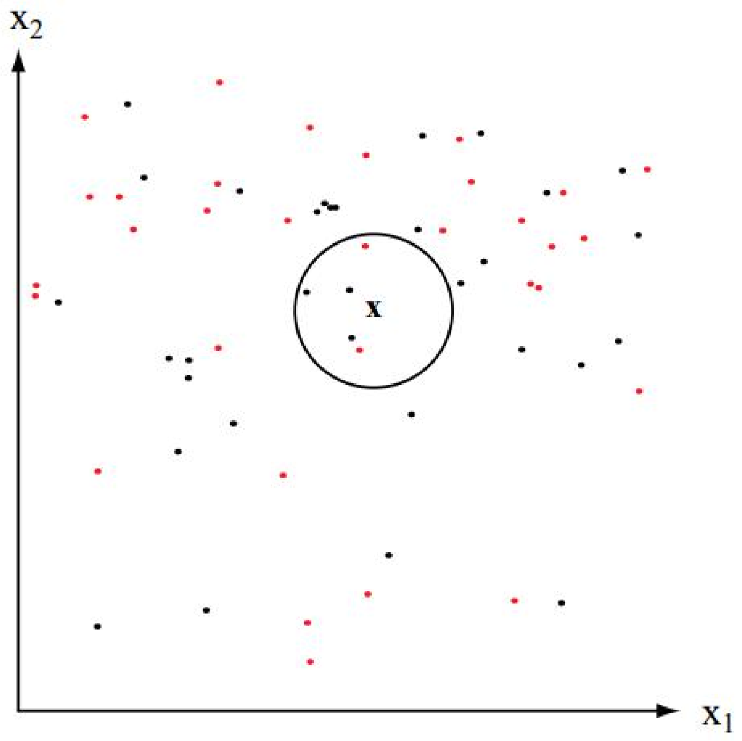
- Let $\mathcal{D}^n = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ be a set of n labeled prototypes
- Let $\mathbf{x}' \in \mathcal{D}^n$ be the closest prototype to a test point x then the nearest neighbor rule for classifying x is to assign it the label associated with x’
- If $n \rightarrow \infty$, it is always possible to find x’ sufficiently close so that:

$$P(\omega_m|\mathbf{x}) = \max_i P(\omega_i|\mathbf{x})$$

- kNN rule is certainly simple and intuitive. **If we have a lot of samples, the kNN rule will do very well ！**

The k–Nearest-Neighbor Rule

- Goal: Classify x by assigning it the label most frequently represented among the k nearest samples
- Use a voting scheme



The k-nearest-neighbor query starts at the test point and grows a spherical region until it encloses k training samples, and labels the test point by a majority vote of these samples

Voronoi tesselation

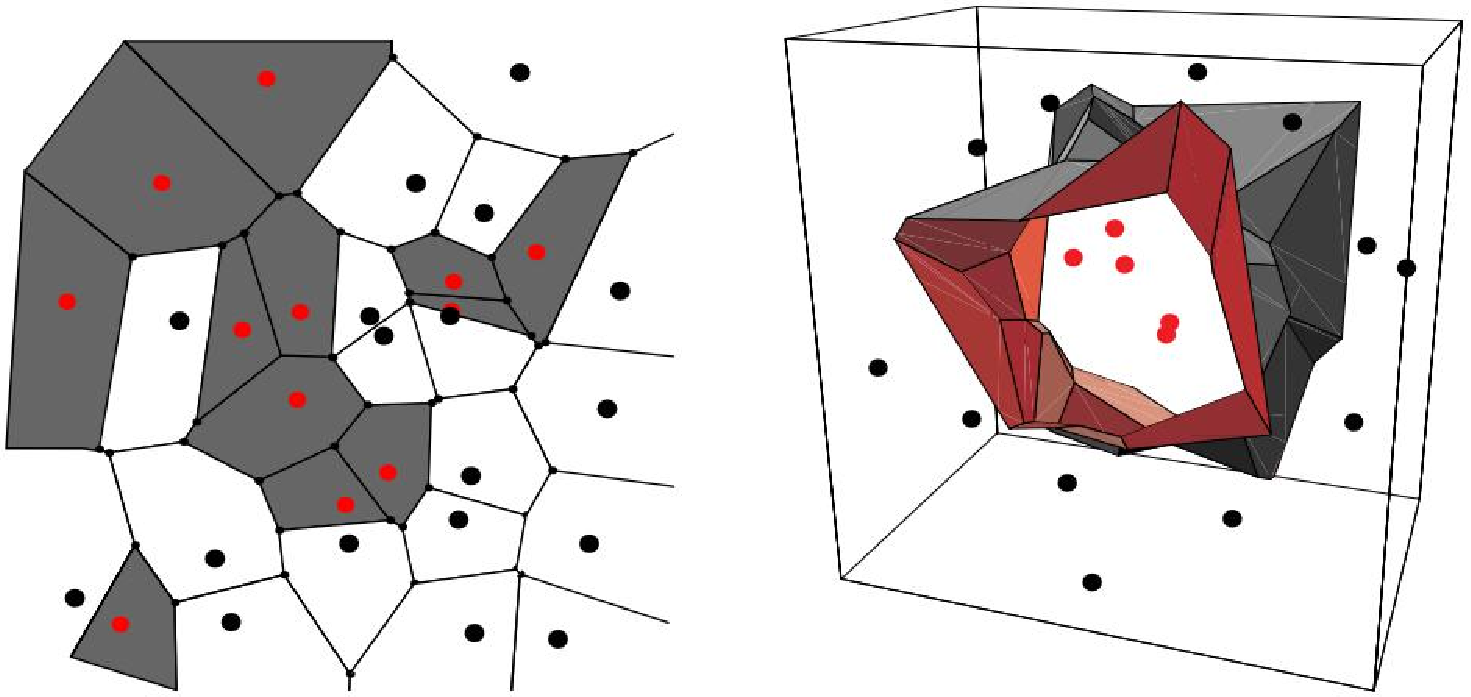
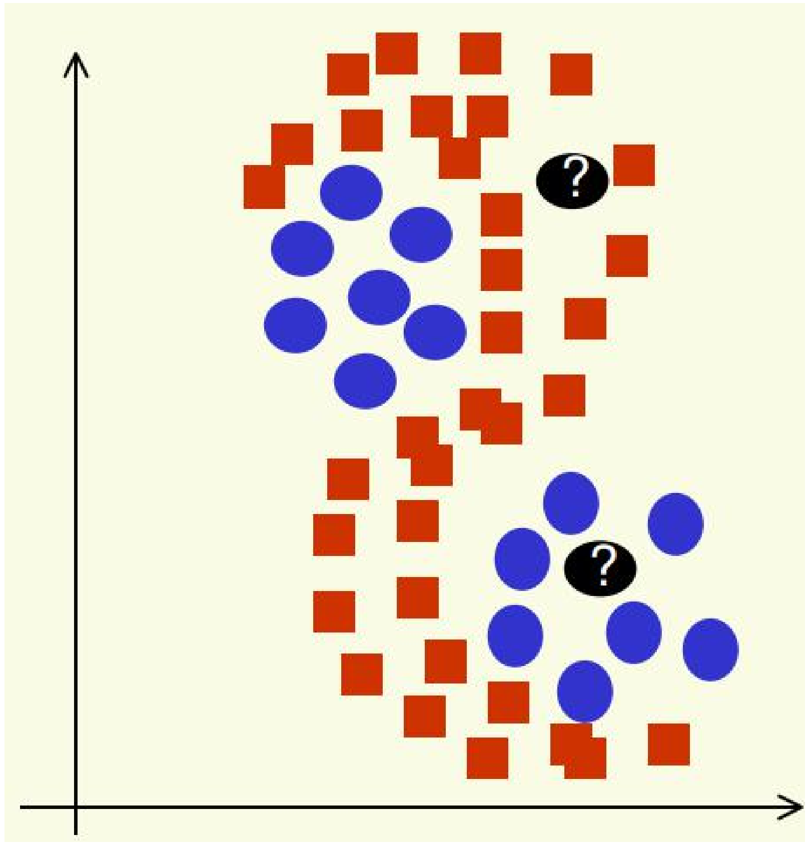


Figure 4.13: In two dimensions, the nearest-neighbor algorithm leads to a partitioning of the input space into Voronoi cells, each labelled by the category of the training point it contains. In three dimensions, the cells are three-dimensional, and the decision boundary resembles the surface of a crystal.

kNN: Multi-modal Distributions

- Most parametric distributions would not work for this 2 class classification problem

- Nearest neighbors will do reasonably well, provided we have a lot of samples



Outline

- K–Nearest Neighbor Estimation
- The Nearest–Neighbor Rule
- **Error Bound for K-Nearest Neighbor**
- The Selection of K and Distance
- The Complexity for KNN
- Probabilistic KNN

Notation

- ω_m is class with maximum probability given a point

$$P(\omega_m|x) = \max_i P(\omega_i|x)$$

Bayes Decision Rule always selects class which results in minimum risk (i.e. highest probability), which is ω_m

- P$*$ is the minimum probability of error, which is Bayes Rate.

Minimum error probability for a given x: $P^*(e|x) = 1 - P(\omega_m|x)$

Minimum average error probability for x: $P^* = \int P^*(e|x)p(x)dx$

Nearest Neighbor Error

- We will show:
  - The average probability of error is not concerned with the exact placement of the nearest neighbor.
  - The exact conditional probability of error is:

$$P = \int \left[1 - \sum_{i=1}^{c} P^2(\omega_i | x) \right] p(x) dx$$

  - The above error rate is never worse than 2x the Bayes Rate:

$$P^* \le P \le 2P^*$$

Approximate probability of error when all classes, c, have equal probability: $1 - (1/c)$

Convergence: Average Probability of Error

- Error depends on choosing the a nearest neighbor that shares that same class as x:

$$P(e|x) = \int P(e|x,x')p(x'|x)dx'$$

- As n goes to infinity, we expect p(x'|x) to approach a **delta function** (i.e. get indefinitely large as x' nearly overlaps x).
- Thus, the integral of p(x'|x) will evaluate to 0 everywhere but at x where it will evaluate to 1, so:

$$P(e|x) = P(e|x,x')$$

$$P(e|x) = P(e|x')$$

Error Rate: Conditional Probability of Error

- For each of n test samples, there is an error whenever the chosen class for that sample is not the actual class.
- For the Nearest Neighbor Rule:

✓ Each test sample is a random (x,θ) pairing, where θ is the actual class of x.
✓ For each x we choose x'. x' has class θ'.
✓ There is an error if θ ≠ θ'.

sum over classes being the same for x and x'

$$P_n(e|x,x'_n) = 1 - \sum_{i=1}^{c} P(\omega_i|x)P(\omega_i|x'_n)$$

$$\lim_{n\to\infty} P_n(e|x) = \int \left[1 - \sum_{i=1}^{c} P(\omega_i|x)P(\omega_i|x'_n)\right] \delta(x'_n - x)\,dx'_n$$

$$= 1 - \sum_{i=1}^{c} P^2(\omega_i|x)$$

delta function: $x \approx x'_n$

Error Rate: Conditional Probability of Error

- Error as number of samples go to infinity:

$$P = \lim_{n \to \infty} P_n(e)$$

$$P = \lim_{n \to \infty} \int P_n(e|x)\, p(x)\, dx$$

$$P = \int \left[1 - \sum_{i=1}^{c} P^2(\omega_i|x)\right] p(x)\, dx$$

Notice the squared term.
The lower the probability of correctly identifying a class given point x, the greater impact it has on increasing the overall error rate for identifying that point's class.

It's an exact result. How does it compare to Bayes Rate, P*?

Error Bounds

- Exact Conditional Probability of Error:

$$P = \int \left[1 - \sum_{i=1}^{c} P^2(\omega_i | x) \right] p(x) dx$$

How low can this get?
How high can the error rate get?

Expand: $$\sum_{i=1}^{c} P^2(\omega_i | x) = P^2(\omega_m | x) + \sum_{i \neq m}^{c} P^2(\omega_i | x)$$

Constraint 1: $$P(\omega_i | x) \geq 0$$

Constraint 2: $$\sum_{i \neq m} P(\omega_i | x) = 1 - P(\omega_m | x) = P^*(e | x)$$

The summed term is minimized when all the posterior probabilities but the m-th are equal:

$$P(\omega_i | x) = \begin{cases} \dfrac{P^*(e|x)}{c-1}, & i \neq m \\ 1 - P^*(e|x), & i = m \end{cases}$$

Non-m Posterior Probabilities have equal likelihood. Thus, divide by c-1

Error Bounds

- Finding the Error Bounds:

$$\sum_{i=1}^{c} P^2(\omega_i|x) = P^2(\omega_m|x) + \sum_{i \neq m}^{c} P^2(\omega_i|x)$$

$$\sum_{i=1}^{c} P^2(\omega_i|x) \geq (1 - P^*(e|x))^2 + \frac{P^{*2}(e|x)}{c-1}$$

Plug in minimizing conditions and make inequality

$$\sum_{i=1}^{c} P^2(\omega_i|x) \geq \left[1 - 2P^*(e|x) + P^{*2}(e|x)\right] + \frac{P^{*2}(e|x)}{c-1}$$

Factor

$$\sum_{i=1}^{c} P^2(\omega_i|x) \geq \left[1 - 2P^*(e|x)\right] + P^{*2}(e|x)\left(\frac{c-1}{c-1}\right) + \frac{P^{*2}(e|x)}{c-1}$$

Combine terms

$$\sum_{i=1}^{c} P^2(\omega_i|x) \geq \left[1 - 2P^*(e|x)\right] + \frac{c}{c-1} P^{*2}(e|x)$$

Simplify

$$2P^*(e|x) - \frac{c}{c-1} P^{*2}(e|x) \geq 1 \sum_{i=1}^{c} P^2(\omega_i|x)$$

Rearrange expression

$$1 - \sum_{i=1}^{c} P^2(\omega_i|x) \leq 2P^*(e|x) - \frac{c}{c-1} P^{*2}(e|x)$$

Error Bounds

- Finding the Error Bounds:

$$1-\sum_{i=1}^{c} P^2(\omega_i|x) \le 2P^*(e|x) - \cancel{\frac{c}{c-1} P^{*2}(e|x)}$$

$$P = \int \left[1-\sum_{i=1}^{c} P^2(\omega_i|x)\right] p(x)dx$$

$$P^* = \int P^*(e|x)p(x)dx$$

$$\Rightarrow \quad P \le 2P^*$$

Thus, the error rate is less than twice the Bayes Rate

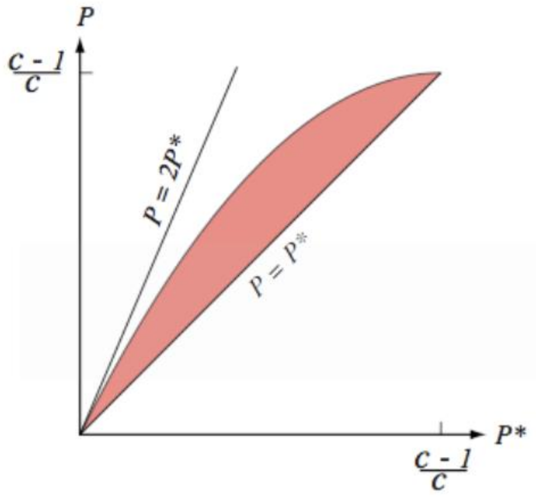
- Tightest upper bounds:

$$P^* \le P \le P^*\left(2 - \frac{c}{c-1}P^*\right)$$

Found by keeping the right term.

Error Bounds

- Bounds on the Nearest Neighbor error rate.



$$P^* \le P \le P^*\left(2 - \frac{c}{c-1}P^*\right)$$

Take P* = 0 and P* = 1 to get bounds for P*

$$0 \le P^* \le (c-1)/c$$

With infinite data, and a complex decision rule, you can at most cut the error rate in half.

- When Bayes Rate, P^*, is small, the upper bound is approximately 2x Bayes Rate.
- Difficult to show Nearest Neighbor performance convergence to asymptotic value

Outline

- K–Nearest Neighbor Estimation
- The Nearest–Neighbor Rule
- Error Bound for K-Nearest Neighbor
- **The Selection of K and Distance**
- The Complexity for KNN
- Probablistical KNN

kNN: How to Choose k?

- In theory, when the infinite number of samples is available, the larger the k, the better is classification (error rate gets closer to the optimal Bayes error rate)

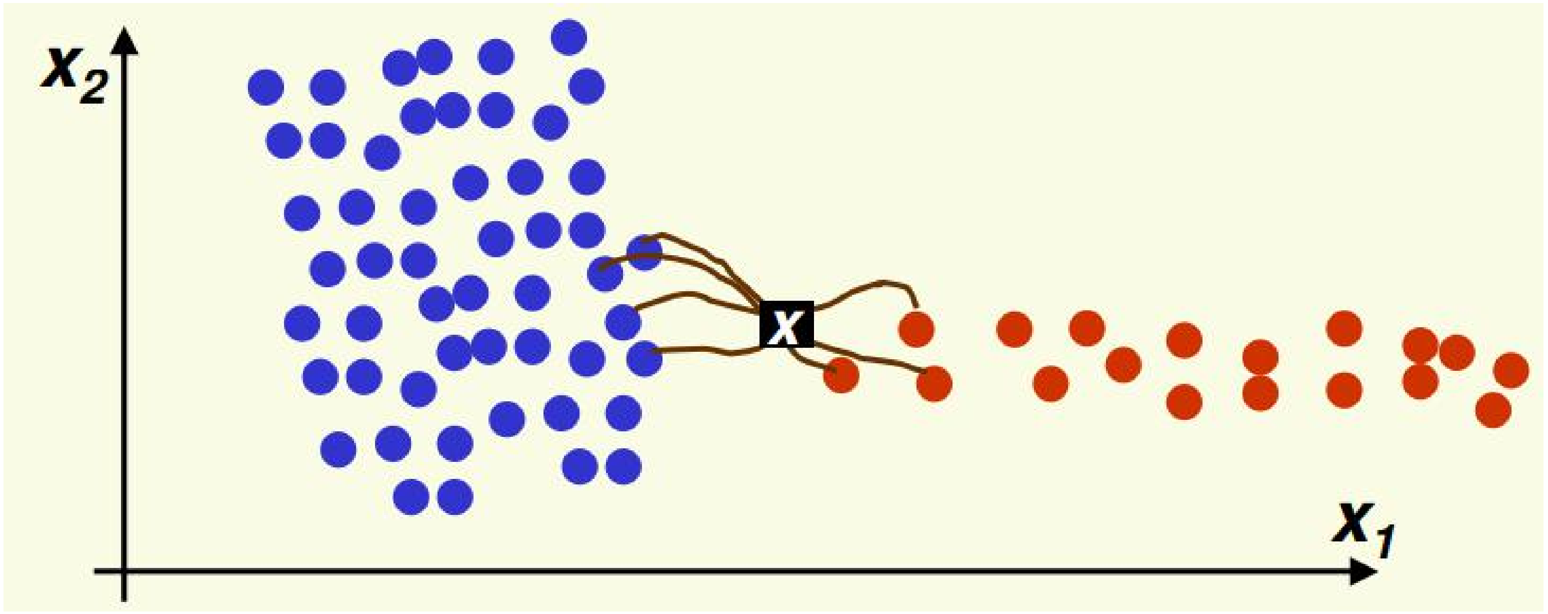
- But the caveat is that **all k neighbors have to be close to x**
  - Possible when infinite # samples available
  - Impossible in practice since # samples is finite

kNN: How to Choose k?

- In practice
  - 1. k should be large so that error rate is minimized
    - k too small will lead to noisy decision boundaries
  - 2. k should be small enough so that only nearby samples are included
    - k too large will lead to over smoothed boundaries

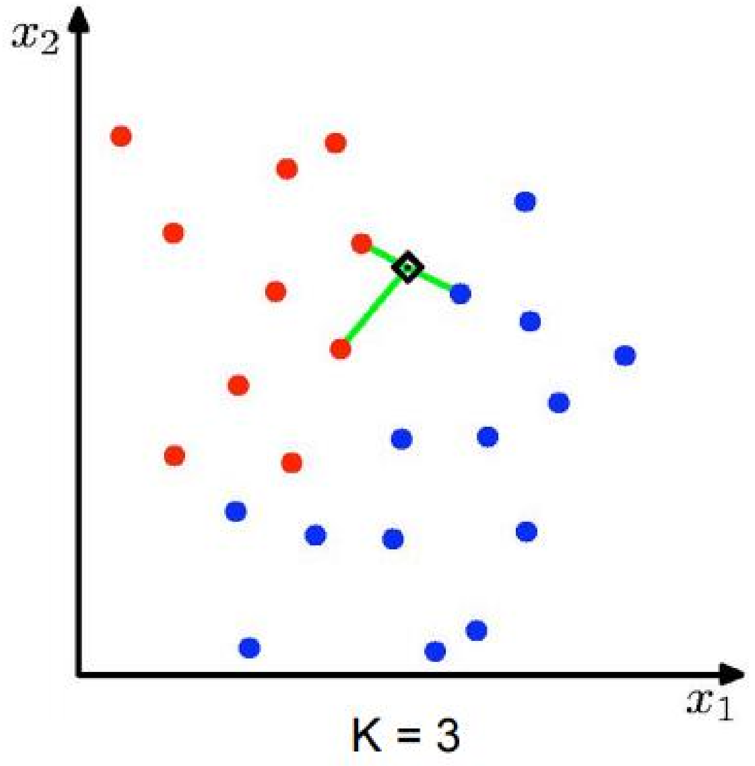
- Balancing 1 and 2 is not trivial
- This is a recurrent issue, need to smooth data, but not too much

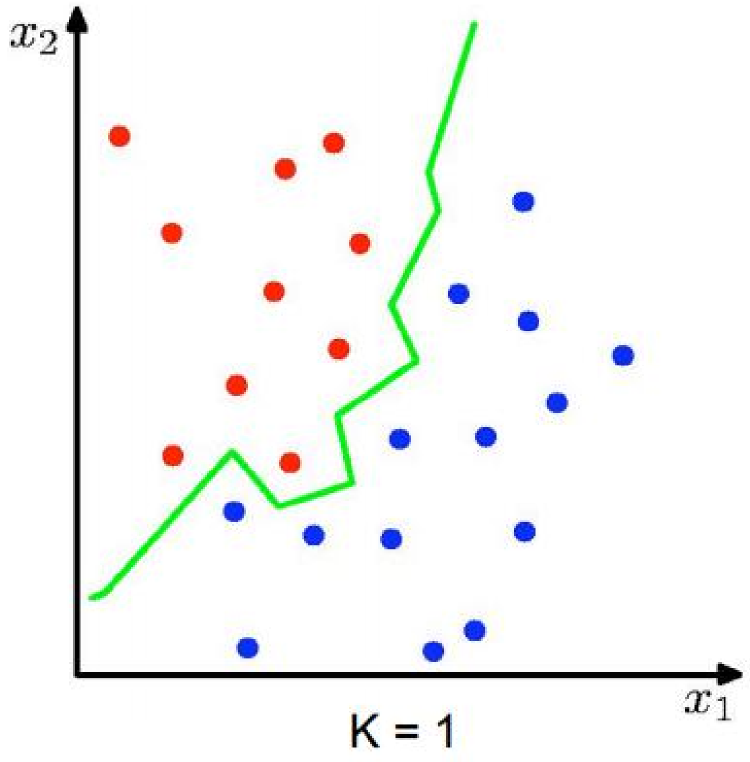
kNN: How to Choose k?



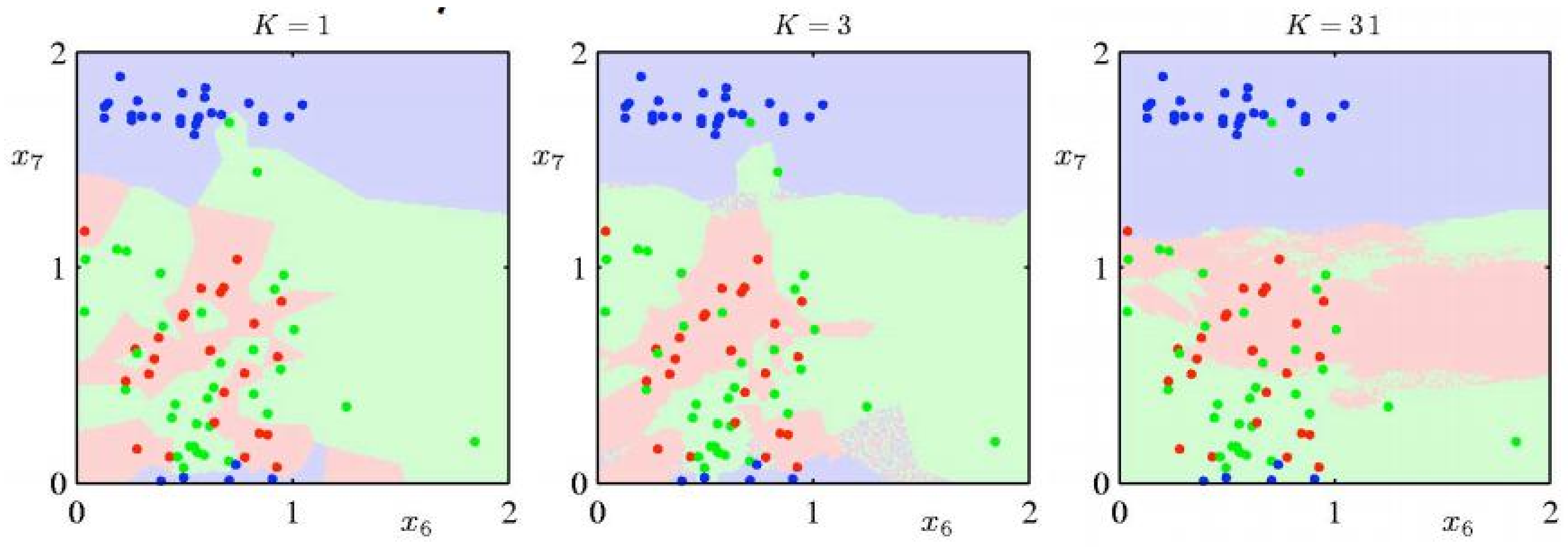
- For k $= 1, \ldots, 7$ point x gets classified correctly
  - red class
- For larger k classification of x is wrong
  - blue class

K-Nearest-Neighbours for Classification





K-Nearest-Neighbours for Classification



- K acts as a smother
- As $N \to \infty$, the error rate of the 1-nearestneighbour classifier is never more than twice the optimal error (obtained from the true conditional class distributions).

kNN: Selection of Distance

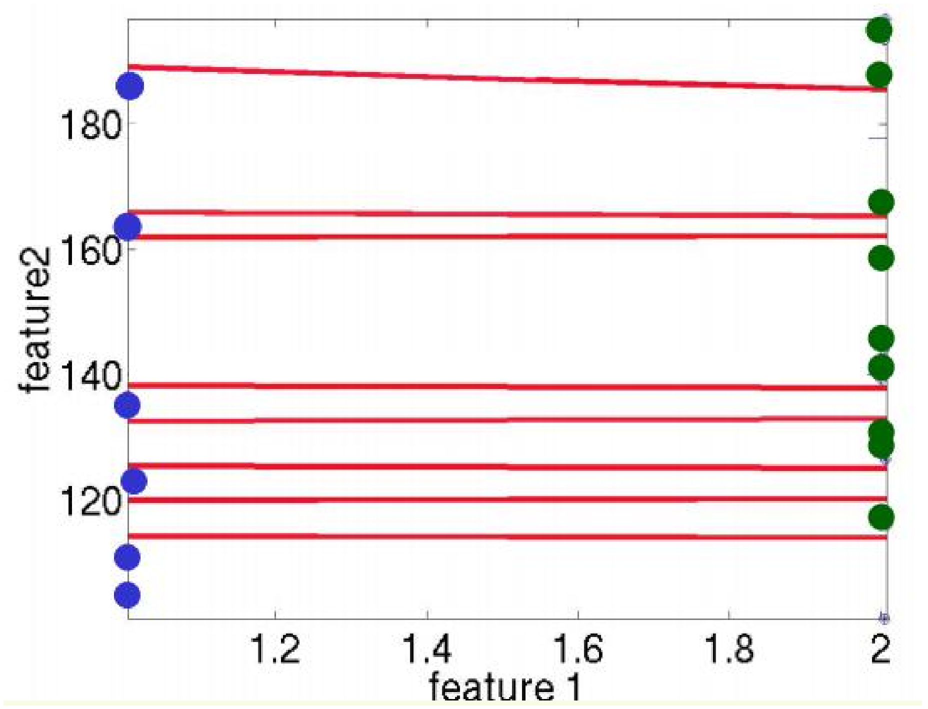
- So far we assumed we use Euclidian Distance to find the nearest neighbor:

$$D(a,b) = \sqrt{\sum_k (a_k - b_k)^2}$$

- However some features (dimensions) may be much more discriminative than other features (dimensions)

- Eucleadian distance treats each feature as equally important

kNN: Extreme Example of Distance Selection



- Decision boundaries for blue and green classes are in red
- These boundaries are really bad because
    - feature 1 is discriminative, but it's scale is small
    - feature 2 gives no class information $($noise$)$ but its scale is large

kNN: Selection of Distance

- Extreme Example
  - feature 1 gives the correct class: 1 or 2
  - feature 2 gives irrelevant number from 100 to 200
- Suppose we have to find the class of x=[1, 100] and we have 2 samples [1, 50] and [2, 110]

$$D\left(\begin{bmatrix}1\\100\end{bmatrix},\begin{bmatrix}1\\150\end{bmatrix}\right)=\sqrt{(1-1)^2+(100-150)^2}=50$$

$$D\left(\begin{bmatrix}1\\100\end{bmatrix},\begin{bmatrix}2\\110\end{bmatrix}\right)=\sqrt{(1-2)^2+(100-110)^2}=10.5$$

- x = [1, 100] is misclassified!
- The denser the samples, the less of the problem
  - But we rarely have samples dense enough

kNN: Selection of Distance

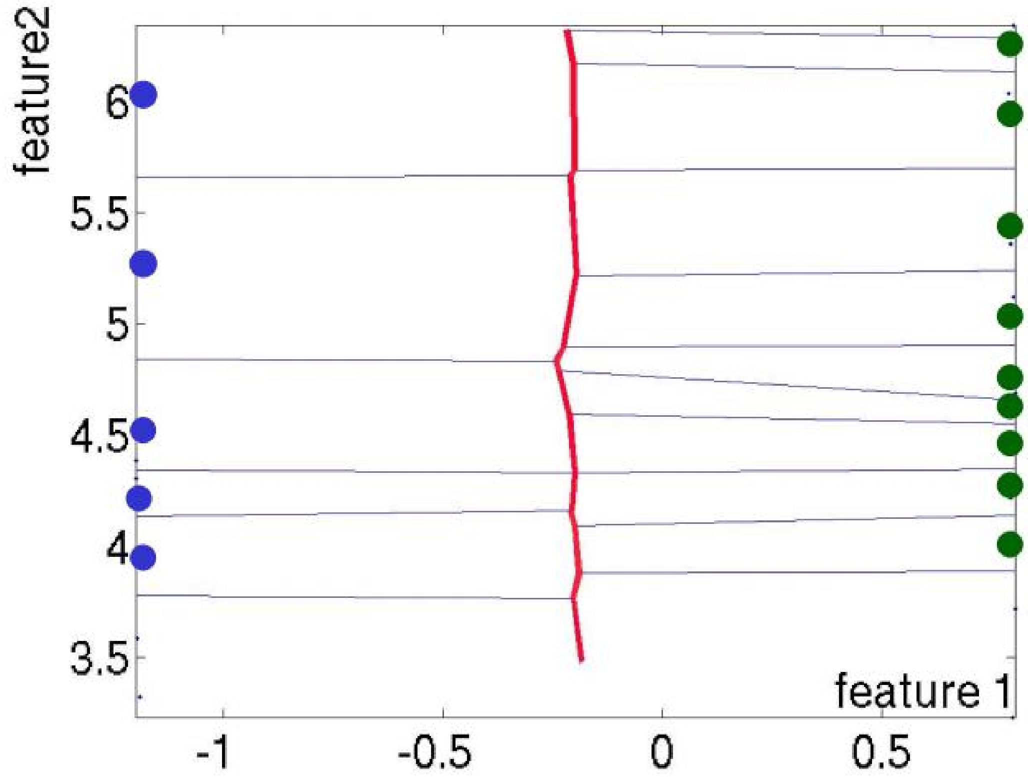
- Notice the 2 features are on different scales:
  - feature 1 takes values between 1 or 2
  - feature 2 takes values between 100 to 200
- We could normalize each feature to be between of mean 0 and variance 1

- If X is a random variable of mean μ and varaince σ^2, then $(X - \mu)/\sigma$ has mean 0 and variance 1

- Thus for each feature vector xi, compute its sample mean and variance, and let the new feature be

$$\frac{x_i - mean(x)}{\sigma}$$

kNN: Normalized Features



- The decision boundary (in red) is very good now!

kNN: Selection of Distance

- However in high dimensions if there are a lot of irrelevant features, normalization will not help

$$D(a,b) = \sqrt{\sum_k (a_k - b_k)^2} = \sqrt{\underbrace{\sum_i (a_i - b_i)^2}_{\text{discriminative feature}} + \underbrace{\sum_j (a_j - b_j)^2}_{\text{noisy features}}}$$

- If the number of discriminative features is smaller than the number of noisy features, Euclidean distance is dominated by noise

kNN: Feature Weighting

- Scale each feature by its importance for classification

$$D(a,b) = \sqrt{\sum_k w_k (a_k - b_k)^2}$$

- Can learn the weights wk from the training data
  - Increase/decrease weights until classification improves

kNN: Mahalanobis distance

- Mahalanobis distance lets us put different weights on different comparisons

$$\begin{aligned} D(u,v)^2 &= (u-v)^T \Sigma (u-v) \\ &= \sum_i \sum_j (u_i - v_i) \Sigma_{ij} (u_j - v_j) \end{aligned}$$

where Σ is a symmetric positive definite matrix

- Euclidean distance is $\Sigma = \mathrm{I}$

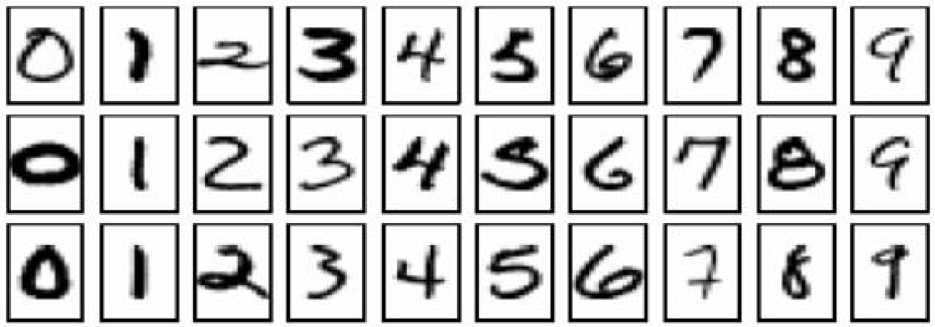
For more information about distance measures, please read this article: http://www.umass.edu/landeco/teaching/multivariate/readings/McCune.and.Grace.2002.chapter6.pdf

Outline

- K–Nearest Neighbor Estimation
- The Nearest–Neighbor Rule
- Error Bound for K-Nearest Neighbor
- The Selection of K and Distance
- **The Complexity for KNN**
- Probablistical KNN

Error rates on USPS digit recognition

- 7291 train, 2007 test
- Neural net: 0.049
- 1−NN/Euclidean distance: 0.055
- 1−NN/tangent distance: 0.026
- In practice, use neural net, since KNN too slow (lazy learning) at test time



Problems with kNN

- Can be slow to find nearest neighbor in high dim space

$$n^* = \arg\min_{n \in D} dist(x, x_n)$$

- Need to store all the training data, so takes a lot of memory
- Need to specify the distance function
- Does not give probabilistic output

Reducing run-time of kNN

- Takes O(Nd) to find the exact nearest neighbor
- Use a branch and bound technique where we prune points based on their partial distances

$$D_r(a,b)^2 = \sum_{i=1}^{r}(a_i - b_i)^2$$

- Structure the points hierarchically into a kd–tree (does offline computation to save online computation)
- Use locality sensitive hashing (a randomized algorithm)

Reducing space requirements of kNN

- Various heuristic algorithms have been proposed to prune/ edit/ condense “irrelevant” points that are far from the decision boundaries
- Later we will study sparse kernel machines that give a more principled solution to this problem

kNN: Computational Complexity

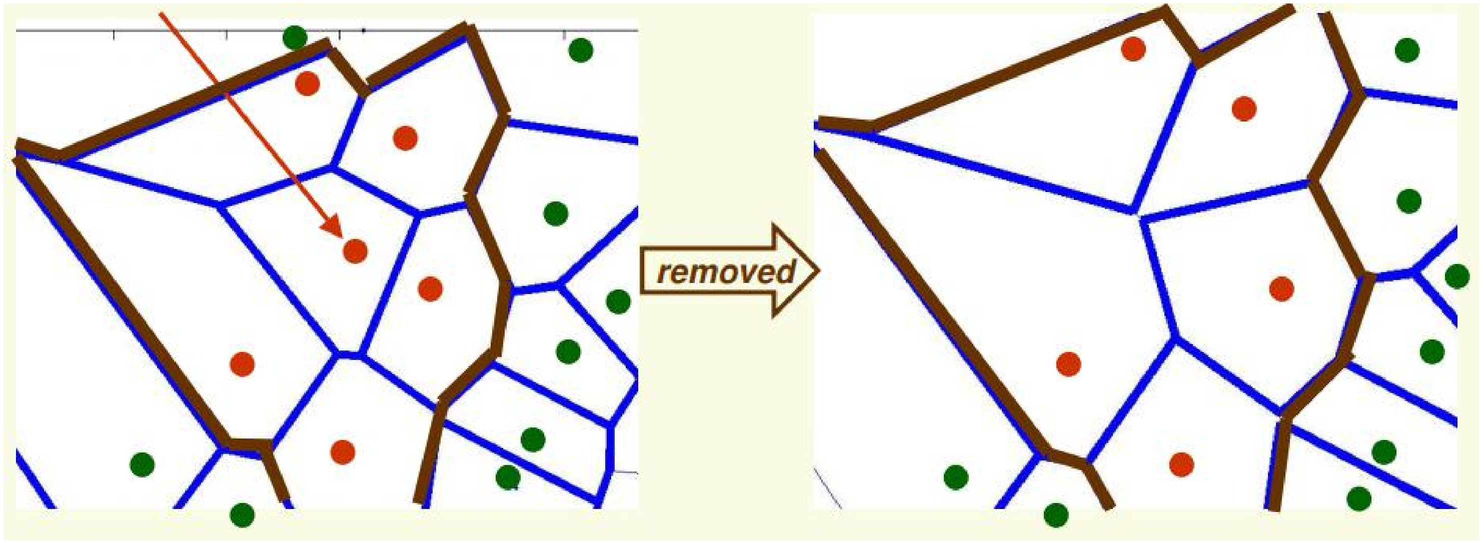
- Basic kNN algorithm stores all examples. Suppose we have n examples each of dimension k
  - O(d) to compute distance to one example
  - O(nd) to find one nearest neighbor
  - O(knd) to find k closest examples examples
  - Thus complexity is O(knd)

- This is prohibitively expensive for large number of samples

- But we need large number of samples for kNN to work well!

Reducing Complexity: Editing 1NN

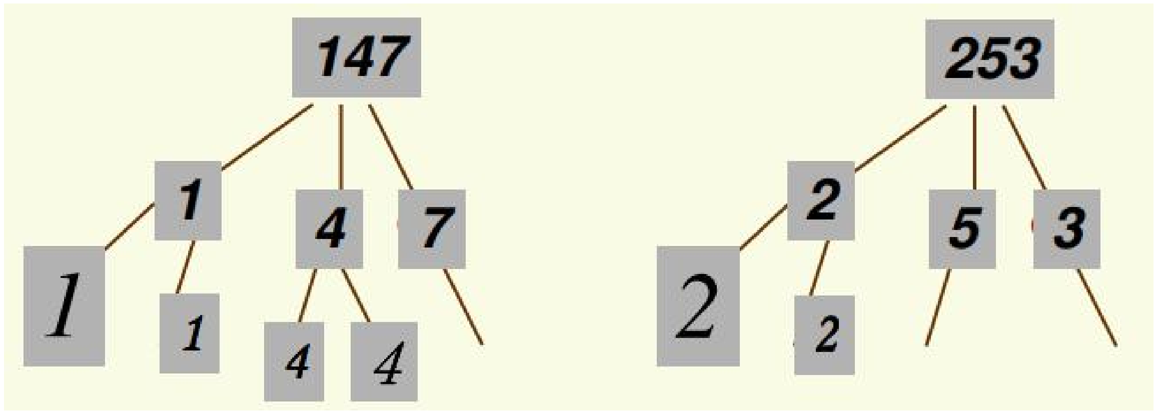
- If all voronoi neighbors have the same class, a sample is useless, we can remove it:



- Number of samples decreases
- We are guaranteed that the decision boundaries stay the same

Reducing Complexity: kNN prototypes

- Explore similarities between samples to represent data as search trees of prototypes



- Advantages: Complexity decreases
- Disadvantages:
  - finding good search tree is not trivial
  - will not necessarily find the closest neighbor, and thus **not** guaranteed that the decision boundaries stay the same

Outline

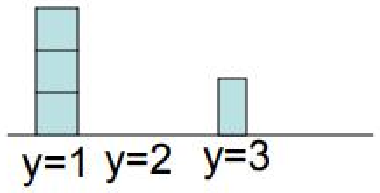
- K–Nearest Neighbor Estimation
- The Nearest–Neighbor Rule
- Error Bound for K-Nearest Neighbor
- The Selection of K and Distance
- The Complexity for KNN
- **Probabilistic KNN**

Probabilistic kNN

- We can compute the empirical distribution over labels in the K–neighborhood
- However, this will often predict 0 probability due to sparse data

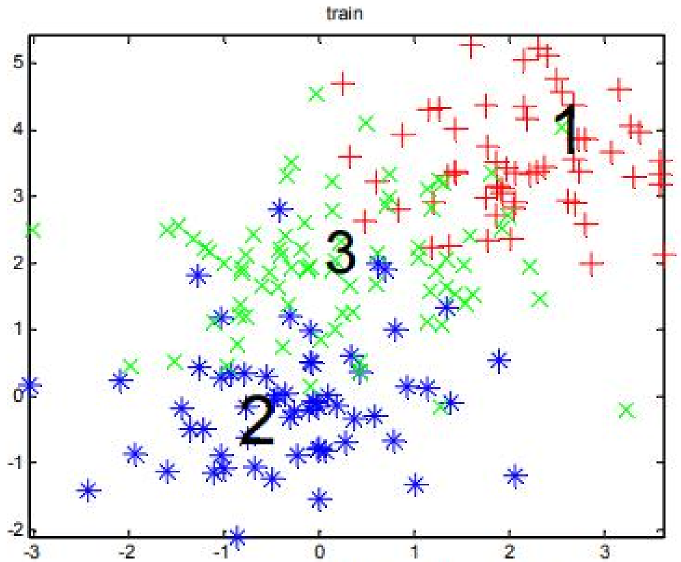
$$p(y|x, D) = \frac{1}{K} \sum_{j \in nbr(x,K,D)} I(y = y_j)$$

K=4, C=3

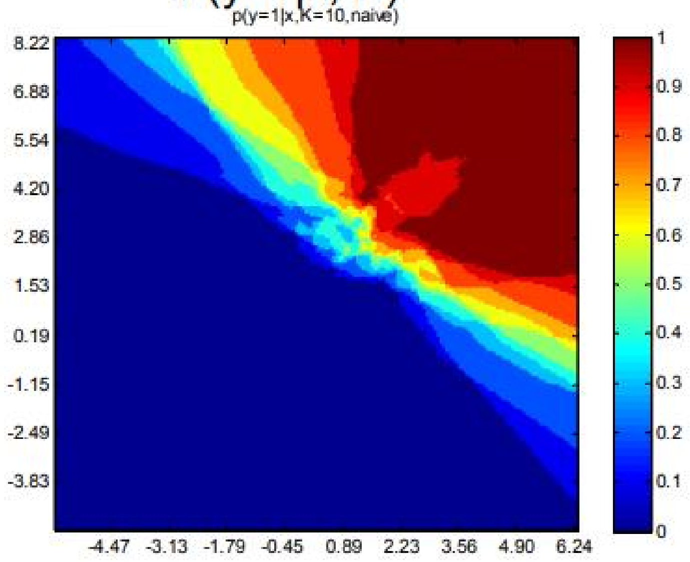


P = [3/4, 0, 1/4]

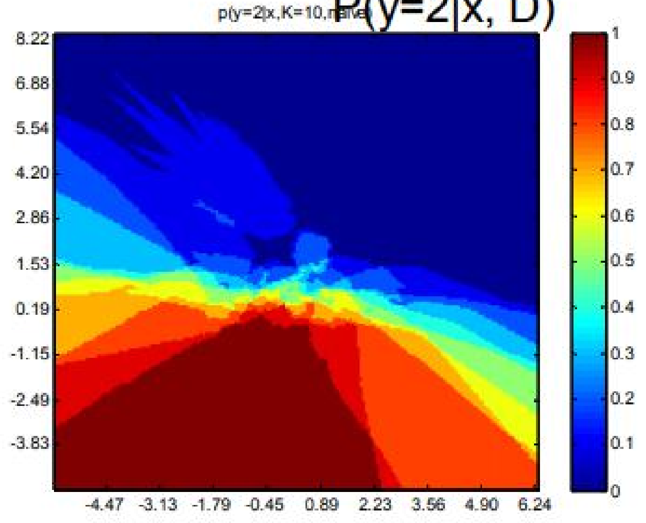
Probabilistic kNN



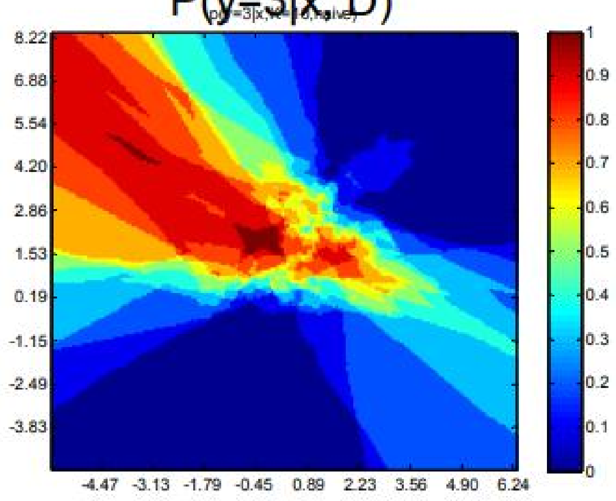
P(y=1|x, D)



P(y=2|x, D)



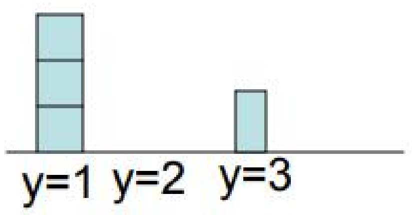
P(y=3|x, D)



Smoothing empirical frequencies

- The empirical distribution will often predict 0 probability due to sparse data
- We can add pseudo counts to the data and then normalize

K=4, C=3



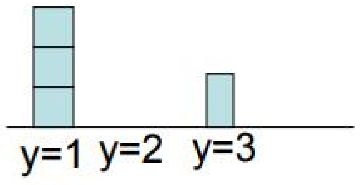
P = [3 + 1, 0 + 1, 1 + 1] / 7 = [4/7, 1/7, 2/7]

Softmax (multinomial logit) function

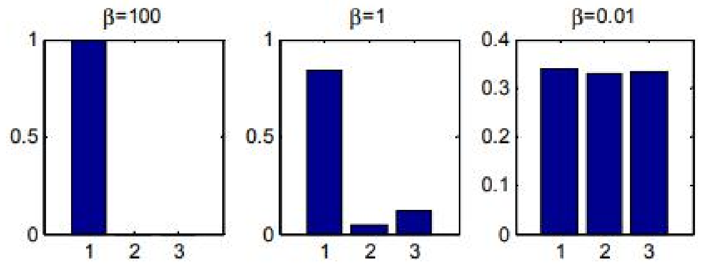
- We can "soften" the empirical distribution so it spreads its probability mass over unseen classes
- Define the softmax with inverse temperature β

$$S(x,\beta)_i = \frac{\exp(\beta x_i)}{\sum_j \exp(\beta x_j)}$$

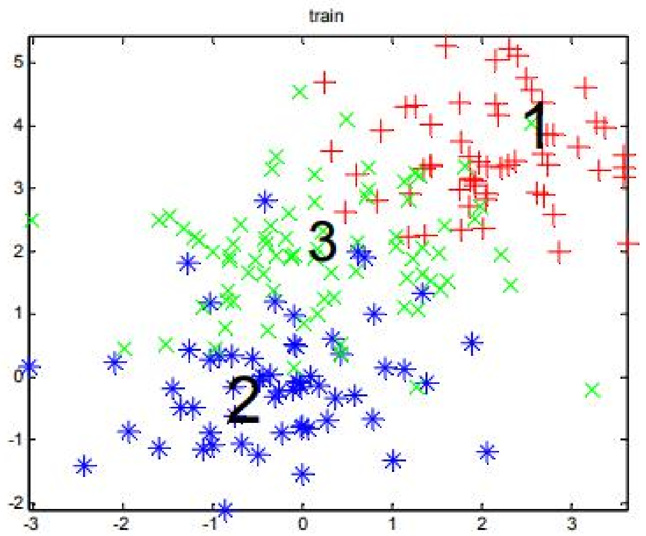
- Big beta = cool temp = spiky distribution
- Small beta = high temp = uniform distribution

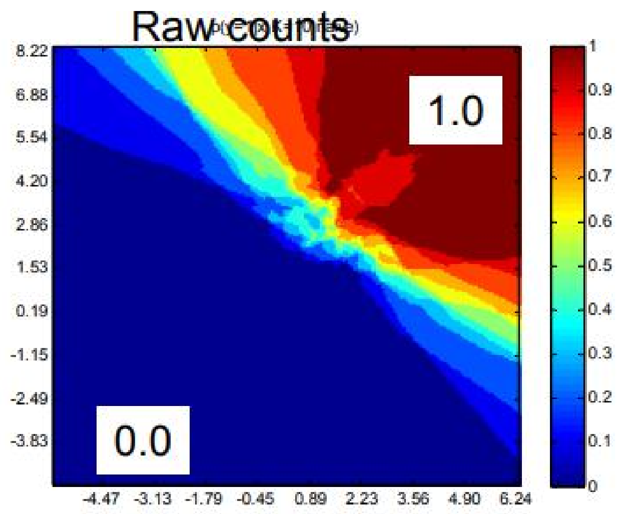


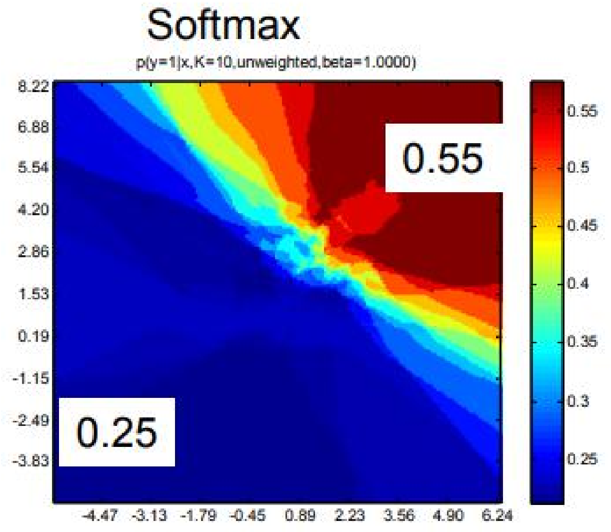
X = [3 0 1]



Softened Probabilistic kNN



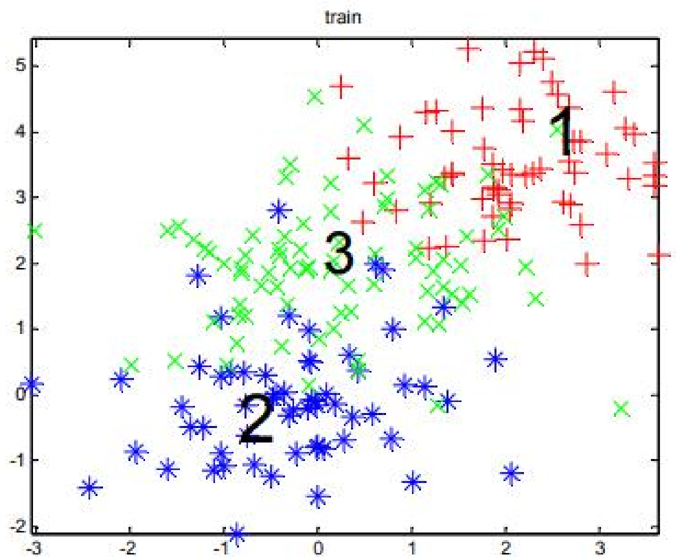


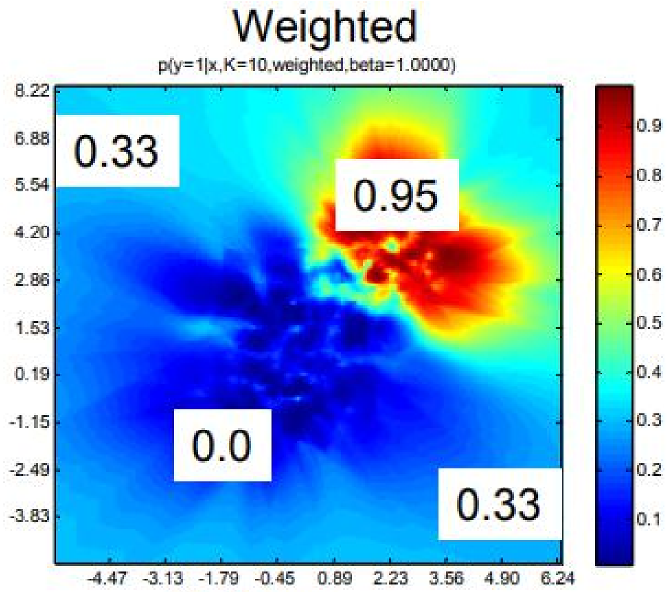


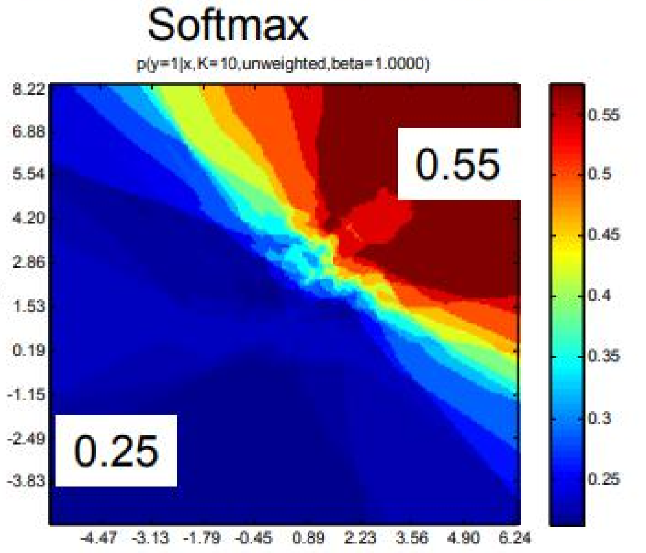
$$p(y|x,D,K,\beta) = \frac{\exp[(\beta/K)\sum_{j\sim x} I(y=y_j)]}{\sum_{y'}\exp[(\beta/K)\sum_{j\sim x} I(y'=y_j)]}$$

Sum over Knn

Weighted Probabilistic kNN







Weighted sum over Knn

$$p(y|x, D, K, \beta) = \frac{\exp[(\beta/K)\sum_{j\sim x} w(x,x_j) I(y=y_j)]}{\sum_{y'} \exp[(\beta/K)\sum_{j\sim x} w(x,x_j) I(y'=y_j)]}$$

Local kernel function

kNN Summary

- Advantages
  - Can be applied to the data from any distribution
  - Very simple and intuitive
  - Good classification if the number of samples is large enough
- Disadvantages
  - Choosing best k may be difficult
  - Computationally heavy, but improvements possible
  - Need large number of samples for accuracy
  - Can never fix this without assuming parametric distribution

Q & A