Nearest Neighbour or Instance-based Learning

Akito Sakurai

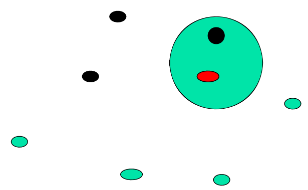
Instance-based Learning

- Concept
  - Remember all the training data $<x_i,f(x_i)>$ (will not consider what and how to predict until we need to do so).
  - If we asked, we will do the best at the time
- Technology belonging to this class
  - Nearest neighbor
  - *k*-Nearest neighbor
  - Locally weighted regression
  - Radial basis functions
- Called "Lazy" technique. What is "eager," then?

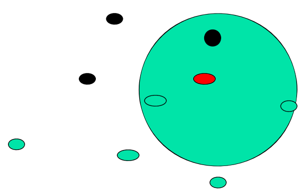
Nearest neighbor

- Basic Concept
  - For a query x_q , find out a nearest point x_n, and answer with a reply $f(x_q) \leftarrow f(x_n)$.
- *k*-Nearest neighbor
  - Find out (not one but) *k* nearest neighbors, and make a reply based on *majority* of their replies.
  - Average of *k* nearest neighbors is also used

1-Nearest Neighbor



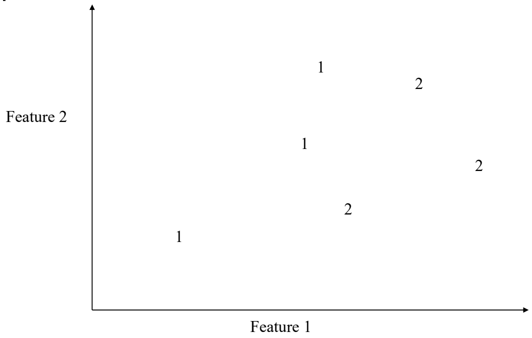
3-Nearest Neighbor



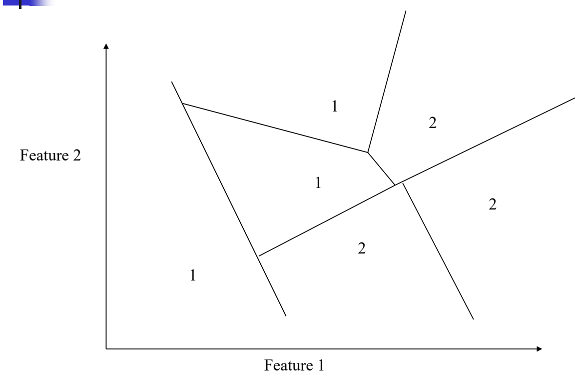
Features

- k-NN is appropriate when
  - Feature vector could be seen as a point in R^n
  - # of feature is not large (less than a few dozens)
  - You have a large amount of data
- K-NN is
  - Fast to learn
  - Could represent a complicated target function
  - Will not lose information contained in training data
- K-NN is
  - Slow to answer (predict)
  - Is easily fooled by irrelevant features

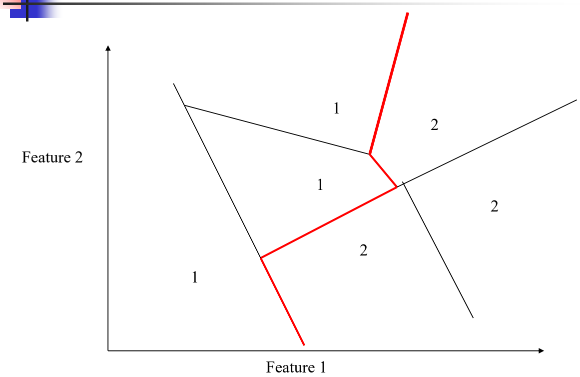
Geometrical interpretation



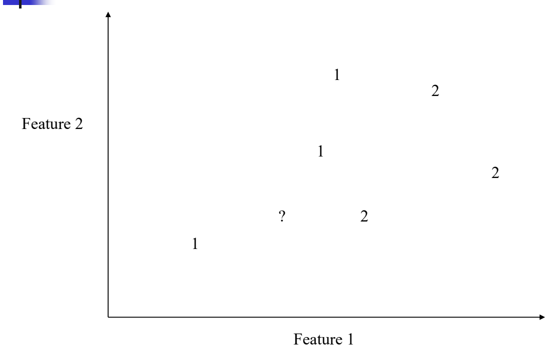
boundaries



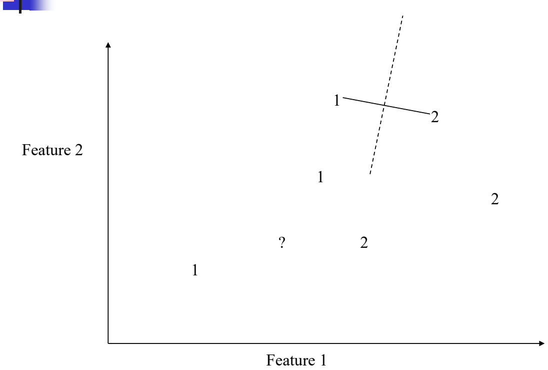
boundaries



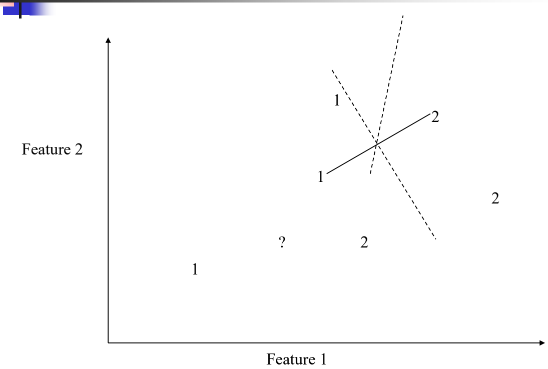
To draw boundaries



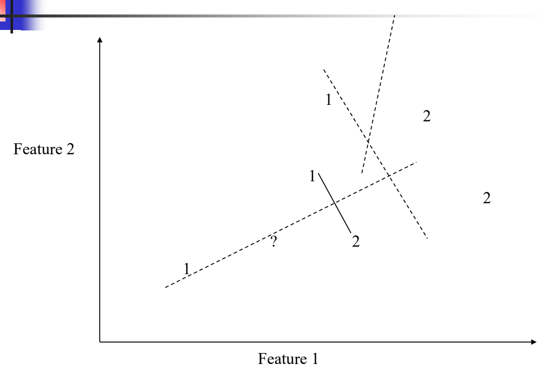
To draw boundaries



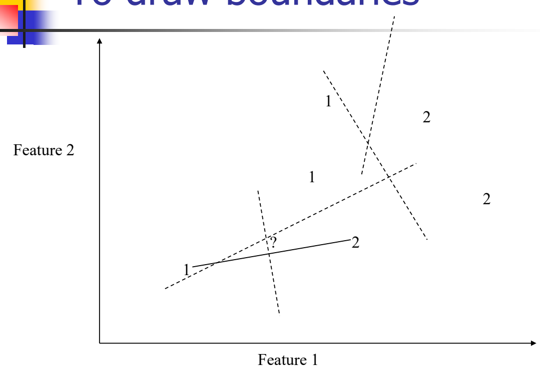
To draw boundaries



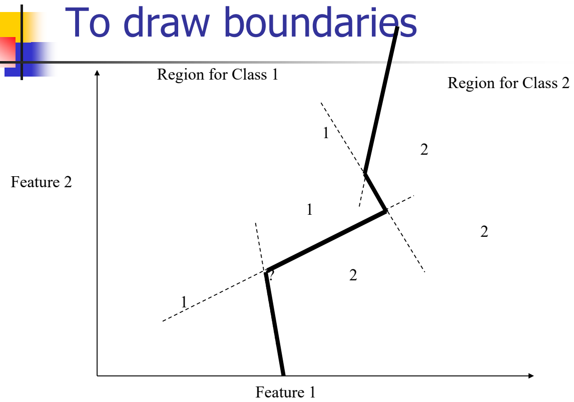
To draw boundaries



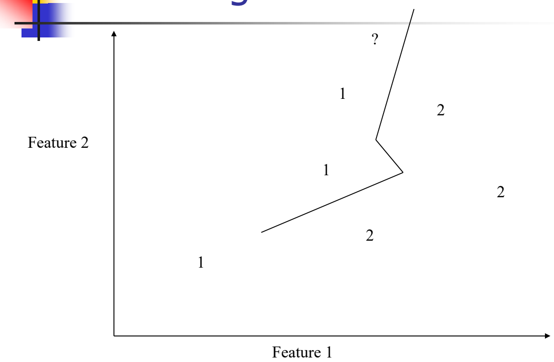
To draw boundaries



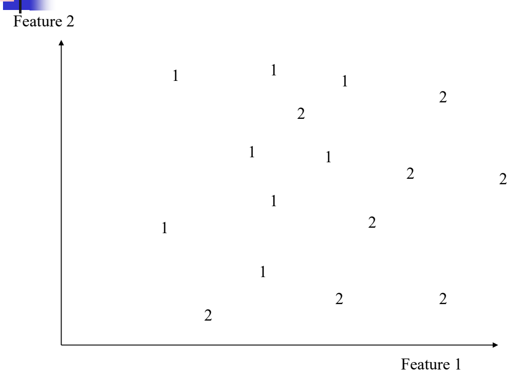
To draw boundaries



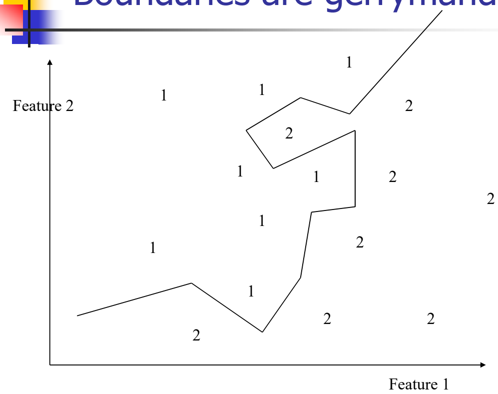
1-NN in geometrical int.



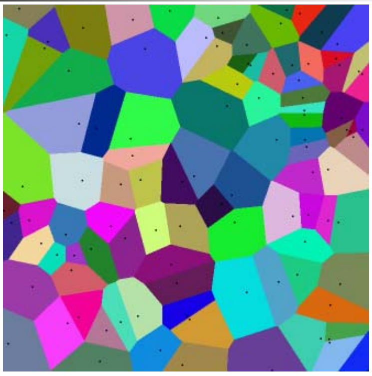
When there are many points

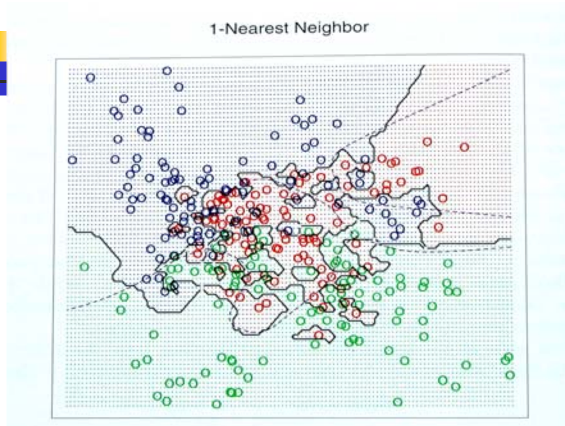


Boundaries are gerrymandering

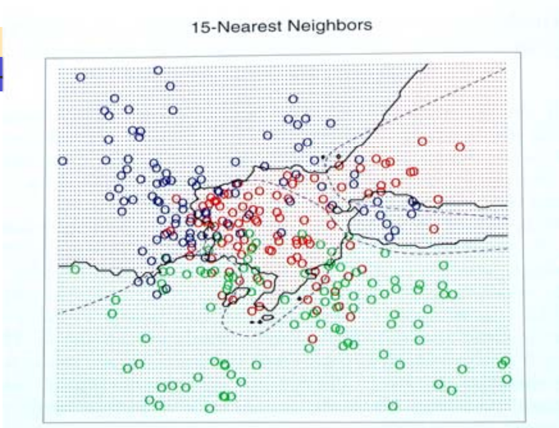


Voronoi diagram





From Hastie, Tibshirani, Friedman 2001 p418



From Hastie, Tibshirani, Friedman 2001 p418

From Hastie, Tibshirani, Friedman 2001 p419

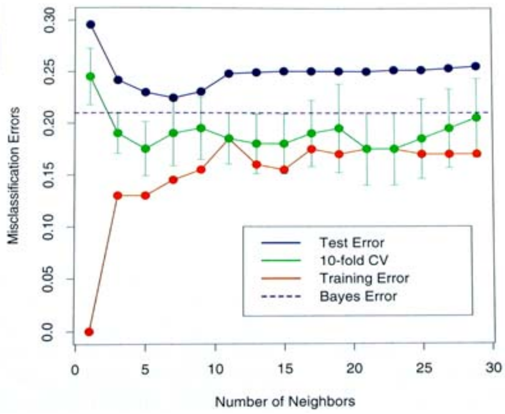


Table 6. Results summary of TC systems on Reuters versions 1–4.

System	Reuters version 1	Reuters version 2	Reuters version 3	Reuters version 4
WORD	—	.15 (Scut)	.31 (Pcut)	.29 (Pcut)
kNN	—	.69 (Scut)	.85 (Scut)	.82 (Scut)
LLSF	—	—	.85 (Scut)	.81 (Scut)
NNets.PARC (perceptron)	—	—	—	.82 (Pcut)
CLASSI (perceptron)	—	—	.80	—
RIPPER (DNF)	—	.72 (Scut)	.80 (Scut)	—
SWAP-1 (DNF)	—	—	.79	—
DTree IND	—	.67 (Pcut)	—	—
DTree C4.5	—	—	.79 (F_1)	—
CHARADE (DNF)	—	—	.78	—
EXPERTS (n-gram)	—	.75 (Scut)	.76 (Scut)	—
Rocchio	—	.66 (Scut)	.75 (Scut)	—
NaiveBayes	—	.65 (Pcut)	.71	—
CONSTRUE (Exp. Sys.)	.90	—	—	—

Yiming Yang, An Evaluation of Statistical Approaches to Text Categorization, Information Retrieval, vol.1, 69-90 (1999)

			#1	#2	#3	#4	#5
		# of documents	21,450	14,347	13,272	12,902	12,902
		# of training documents	14,704	10,667	9,610	9,603	9,603
		# of test documents	6,746	3,680	3,662	3,299	3,299
		# of categories	135	93	92	90	10
System	Type	Results reported by					
Word	(non-learning)	[Yang 1999]	.150	.310	.290		
	probabilistic	[Dumais et al. 1998]				.752	.815
	probabilistic	[Joachims 1998]					.720
	probabilistic	[Lam et al. 1997]	.443 (MF_1)				
PropBayes	probabilistic	[Lewis 1992]	.650				
Bim	probabilistic	[Li and Yamanishi 1999]				.747	
	probabilistic	[Li and Yamanishi 1999]				.773	
Nb	probabilistic	[Yang and Liu 1999]				.795	
	decision trees	[Dumais et al. 1998]					.884
C4.5	decision trees	[Joachims 1998]					.794
Ind	decision trees	[Lewis and Ringuette 1994]	.670				
Swap-1	decision rules	[Apté et al. 1994]		.805			
Ripper	decision rules	[Cohen and Singer 1999]	.683	.811		.820	
SleepingExperts	decision rules	[Cohen and Singer 1999]	**.753**	.759		.827	
DL-Esc	decision rules	[Li and Yamanishi 1999]				.820	
Charade	decision rules	[Moulinier and Ganascia 1996]		.738			
Charade	decision rules	[Moulinier et al. 1996]		.783 (F_1)			
Llsf	regression	[Yang 1999]		.855	.810		
Llsf	regression	[Yang and Liu 1999]				.849	
BalancedWinnow	on-line linear	[Dagan et al. 1997]	.747	.833			
Widrow-Hoff	on-line linear	[Lam and Ho 1998]				.822	
Rocchio	batch linear	[Cohen and Singer 1999]	.660	.748		.776	
FindSim	batch linear	[Dumais et al. 1998]				.617	.646
Rocchio	batch linear	[Joachims 1998]					.799
Rocchio	batch linear	[Lam and Ho 1998]				.781	
Rocchio	batch linear	[Li and Yamanishi 1999]				.625	
Classi	neural network	[Ng et al. 1997]		.802			
Nnet	neural network	[Yang and Liu 1999]				.838	
	neural network	[Wiener et al. 1995]			.820		
Gis-W	example-based	[Lam and Ho 1998]				.860	
k-NN	example-based	[Joachims 1998]					.823
k-NN	example-based	[Lam and Ho 1998]				.820	
k-NN	example-based	[Yang 1999]	.690	.852	**.820**		
k-NN	example-based	[Yang and Liu 1999]				.856	
	SVM	[Dumais et al. 1998]				.870	**.920**
SvmLight	SVM	[Joachims 1998]					.864
SvmLight	SVM	[Li and Yamanishi 1999]				.841	
SvmLight	SVM	[Yang and Liu 1999]				.859	
AdaBoost.MH	committee	[Schapire and Singer 2000]		.860			
	committee	[Weiss et al. 1999]				**.878**	
	Bayesian net	[Dumais et al. 1998]				.800	.850
	Bayesian net	[Lam et al. 1997]	.542 (MF_1)				

Table 6. Comparative results among different classifiers obtained on five different version of the **Reuters** collection. Unless otherwise noted, entries indicate the microaveraged breakeven point; within parentheses, "M" indicates macroaveraging and "F_1" indicates use of the F_1 measure. **Boldface** indicates the best performer on the collection.

Fabrizio Sebastiani, Machine learning in automated text categorization, ACM Computing Surveys, vol.34, no.1, 1-47 (2002)

Table VI. Comparative Results Among Different Classifiers Obtained on Five Different Versions of Reuters. (Unless otherwise noted, entries indicate the microaveraged breakeven point; within parentheses, "M" indicates macroaveraging and "F_1" indicates use of the F_1 measure; boldface indicates the best performer on the collection.)

			#1	#2	#3	#4	#5
		# of documents	21,450	14,347	13,272	12,902	12,902
		# of training documents	14,704	10,667	9,610	9,603	9,603
		# of test documents	6,746	3,680	3,662	3,299	3,299
		# of categories	135	93	92	90	10
System	Type	Results reported by					
Word	(non-learning)	[Yang 1999]	.150	.310	.290		
	probabilistic	[Dumais et al. 1998]				.752	.815
	probabilistic	[Joachims 1998]				.720	
	probabilistic	[Lam et al. 1997]	.443 (MF_1)				
PropBayes	probabilistic	[Lewis 1992a]	.650				
Bim	probabilistic	[Li and Yamanishi 1999]				.747	
	probabilistic	[Li and Yamanishi 1999]				.773	
Nb	probabilistic	[Yang and Liu 1999]				.795	
	decision trees	[Dumais et al. 1998]					.884
C4.5	decision trees	[Joachims 1998]				.794	
Ind	decision trees	[Lewis and Ringuette 1994]	.670				
Swap-1	decision rules	[Apte et al. 1994]		.805			
Ripper	decision rules	[Cohen and Singer 1999]	.683	.811		.820	
SleepingExperts	decision rules	[Cohen and Singer 1999]	**.753**	.759		.827	
DL-Esc	decision rules	[Li and Yamanishi 1999]				.820	
Charade	decision rules	[Moulinier and Ganascia 1996]		.738			
Charade	decision rules	[Moulinier et al. 1996]		.783 (F_1)			
Llsf	regression	[Yang 1999]		.855	.810		
Llsf	regression	[Yang and Liu 1999]				.849	
BalancedWinnow	on-line linear	[Dagan et al. 1997]	.747 (M)	.833 (M)			
Widrow-Hoff	on-line linear	[Lam and Ho 1998]				.822	
Rocchio	batch linear	[Cohen and Singer 1999]	.660	.748		.776	
FindSim	batch linear	[Dumais et al. 1998]				.617	.646
Rocchio	batch linear	[Joachims 1998]				.799	
Rocchio	batch linear	[Lam and Ho 1998]				.781	
Rocchio	batch linear	[Li and Yamanishi 1999]				.625	
Classi	neural network	[Ng et al. 1997]		.802			
Nnet	neural network	[Yang and Liu 1999]				.838	
	neural network	[Wiener et al. 1995]			**.820**		
Gis-W	example-based	[Lam and Ho 1998]				.860	
k-NN	example-based	[Joachims 1998]				.823	
k-NN	example-based	[Lam and Ho 1998]				.820	
k-NN	example-based	[Yang 1999]	.690	.852	**.820**		
k-NN	example-based	[Yang and Liu 1999]				.856	
	SVM	[Dumais et al. 1998]				.870	**.920**
SvmLight	SVM	[Joachims 1998]				.864	
SvmLight	SVM	[Li Yamanishi 1999]				.841	
SvmLight	SVM	[Yang and Liu 1999]				.859	
AdaBoost.MH	committee	[Schapire and Singer 2000]		**.860**			
	committee	[Weiss et al. 1999]				**.878**	
	Bayesian net	[Dumais et al. 1998]				.800	.850
	Bayesian net	[Lam et al. 1997]	.542 (MF_1)				

Fabrizio Sebastiani, Machine learning in automated text categorization, ACM Computing Surveys, vol.34, no.1, 1-47 (2002)

Behavior in infinity

- $p(x)$: posterior prob. of x being 1 (positive)
- 1-Nearest neighbor:
  - when # of samples $\rightarrow \infty$, asymptotic to Gibbs
  - Gibbs predicts 1 with probability $p(x)$
- k-Nearest neighbor
  - # of smpls $\rightarrow \infty$ and k >>1, asymptotic to Bayes opt.
  - Bayes opt. : Summing up all the votes, if $p(x)$>0.5 then 1 else 0.

Note: Expected error of Gibbs is at most twice of that of Bays optimal

Gibbs classifier

Given a new instance,

1. Sample a hypothesis randomly according to P(h|D) over H
2. Classify the new instance by the hypothesis

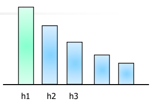
When the expectation is taken over the prior distribution P(h) of target concepts,

$E[error_{BayesOptimal}] \leq E[error_{Gibbs}] \leq 2E[error_{BayesOptimal}]$

(Haussler et al. 1994) or "Mitchell Machine Learning Chap. 6.8"

Useful when there exist many hypotheses and repetitive predictions

Bayes Optimal Classifier



$$\arg\max_{c_j \in \{+,-\}} \sum_{h_i \in H} P(c_j \mid h_i) P(h_i \mid D)$$

Note: Bayes-optimal classifier need not to be in the hypothesis space H.

Note: Many papers/reviews claim that it works well. But in reality, it is often not the case. To clarify conditions when it does is an interesting research topic.

Note: Is it feasible? When feasible, it takes long time to calculate.

Bayes optimal vs. MAP

Suppose our hypothesis space H has three functions h1, h2 and h3

- P(h1 | D) = 0.4, P(h2 | D) = 0.3, P(h3 | D) = 0.3
- What is the MAP hypothesis?
- For a new instance **x**, suppose h1(**x**) = +1, h2(**x**) = -1 and h3(**x**) = -1
- What is the most probable classification of **x**? -1 !

  P(+1 | x) = 0.4 P(-1| x) = 0.3 + 0.3
- The most probable classification is not the same as the prediction of the MAP hypothesis

Distance-weighted k-NN

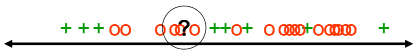
- The closer, the heavier

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}, \quad w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

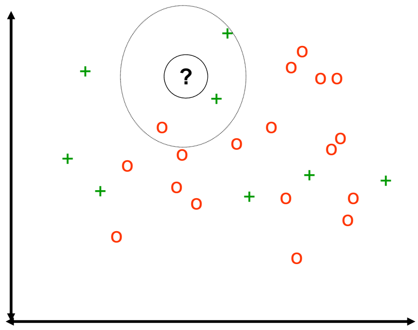
where $d(x_q,x_i)$ is the distance between x_q and x_i

- Using this, not only the "k samples" but also all the samples could be used $\Rightarrow$Shepard's method (1968)

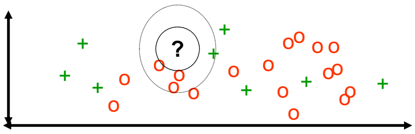
K-NN and irrelevant features



K-NN and irrelevant features



K-NN and irrelevant features



Problems with distance

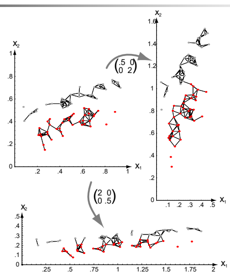
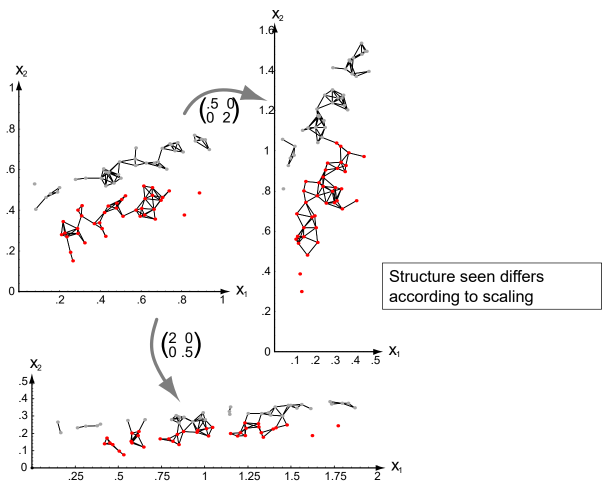


FIGURE 10.8. Scaling axes affects the clusters in a minimum distance cluster method. The original data and minimum-distance clusters are shown in the upper left; points in one cluster are shown in red, while the others are shown in gray. When the vertical axis is expanded by a factor of 2.0 and the horizontal axis shrunk by a factor of 0.5, the clustering is altered (as shown at the right). Alternatively, if the vertical axis is shrunk by a factor of 0.5 and the horizontal axis is expanded by a factor of 2.0, smaller more numerous clusters result (shown at the bottom). In both these scaled cases, the assignmen of points to clusters differ from that in the original space. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification* . Copyright 2001 by John Wiley & Sons, Inc.



Curse of dimensionality

- Suppose that we have 20 features but only tow of them are meaningful.
- Curse of dimensionality :
  - *k*-NN gives us any conclusion by the 18 features
- A solution:
  - Give weight z_j to the j-th feature, where z_j is chosen so that the prediction error is minimal
  - cross-validation would determine z_j.

Locally weighted regression

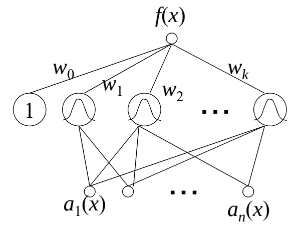
- k-NN is understood to locally approximate f around a query x_q.
- How about explicitly constructing an approximation of $f(x)$ around x_q ?
  - Linear regression to k-NN ?
  - Second order regression ?
  - Spline?
- There are candidates of errors to be minimized

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in x_q \text{ の k-NN}} \left(f(x) - \hat{f}(x_q)\right)^2$$

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} \left(f(x) - \hat{f}(x_q)\right)^2 K(d(x_q, x))$$

Radial Basis Function Network

- Linear combination of local approximators
- A kind of neural networks
- Similar to distance-weighted regression
  - Not lazy but eager



$$f(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x))$$

An example of $K_u(d(x_u, x))$

$$K_u(d(x_u, x)) \equiv e^{-\frac{1}{2\sigma^2} d(x_u, x)^2}$$

Learning of RBF

- To Determine x_u of $K_u(d(x_u,x))$
  - Scatter them uniformly in the sample space
  - From training samples
- To Learn weights (supposing K_u is Gaussian)
  - Determine sd and mean of K_u.
    - E.g. EM
  - Fixing K_u, determine linear part
    - Linear regression is fast

Lazy vs. eager

- Lazy: does not generalize examples but think it over when queried.
  - k-Nearest Neighbor
- Eager: does generalize examples before queries
  - *Learning*-type algorithm, ID3, regression, RBF, etc.
- Any difference?
  - Eager: in many cases, creates a global approximation
  - Lazy: creates a local approximation when needed
  - For the same hypothesis space, lazy would create more complex hypothesis globally
    - Possible over-fitting
    - Flexible to combine complex regions and simple regions.

Summary

- Instance-base approach
  - Does not assume a global structure
    - Admits any structure
  - Susceptible to noise (could not utilize global information to smooth it locally)
  - Curse of dimensionality