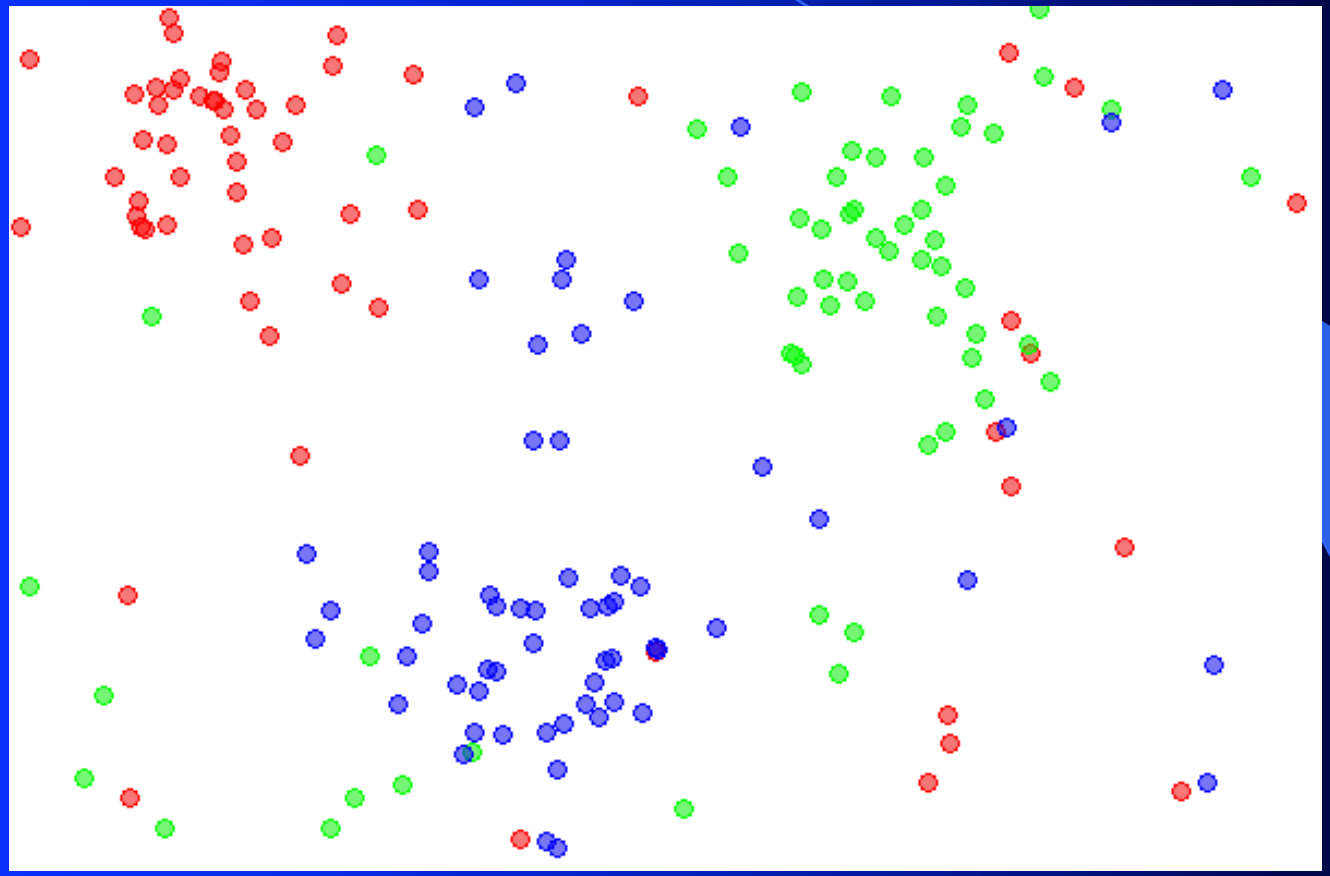
Nearest Neighbor Learning (Instance Based Learning)

- Classify based on local similarity
- Ranges from simple *nearest neighbor* to case-based and analogical reasoning
- Use local information near the current query instance to decide the classification of that instance
- As such can represent quite complex decision surfaces in a simple manner
  - Local model vs a model such as an MLP which uses a global decision surface

k-Nearest Neighbor Approach

- Simply store all (or some representative subset) of the examples in the training set
- When desiring to generalize on a new instance, measure the *distance* from the new instance to all the stored instances and the nearest ones *vote* to decide the class of the new instance
- No need to pre-process a specific hypothesis (Lazy vs Eager learning)
  - Fast learning
  - Can be slow during execution and require significant storage
  - Possible to index the data or reduce the instances stored to enhance efficiency

Examples



k-Nearest Neighbor

- Naturally supports real valued attributes
- Typically use Euclidean distance

$$dist(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

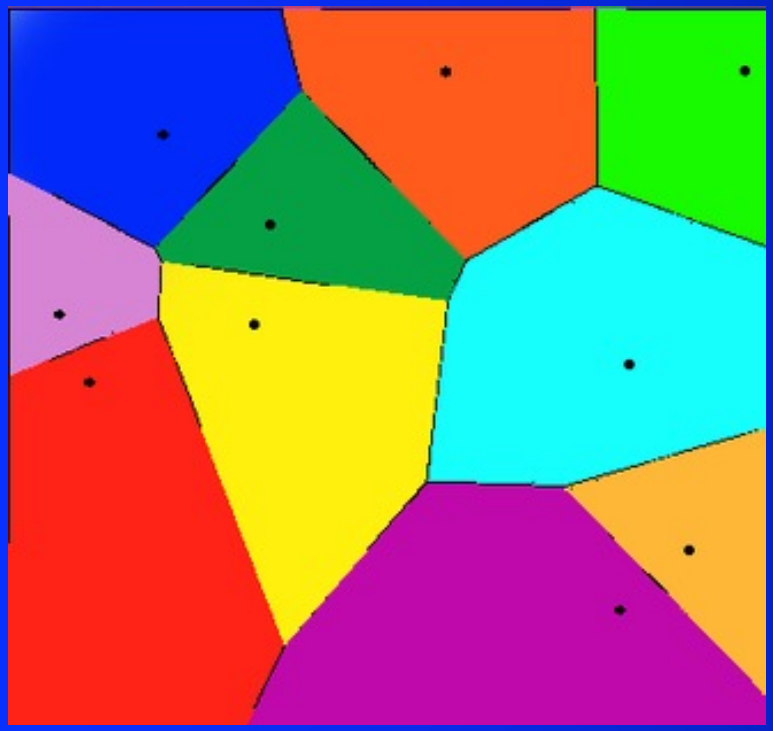
- Nominal/unknown attributes can just be a 1/0 distance (more on other distance metrics later)
- The *k* nearest neighbors each vote for their output class and we sum the votes
  - Assume three output classes A, B, C and $k = 3$ with summed votes (2, 1, 0)
  - A is the winning output class, or we could output all the votes, or normalize to a probability vector (.67, .33. 0)
- *k* greater than 1 is more noise resistant, but too large of *k* could lead to less accuracy as less relevant neighbors have more influence (common values: k=3, k=5)
  - Usually discover best *k* by trial and error (trying different values for a task)

Decision Surface

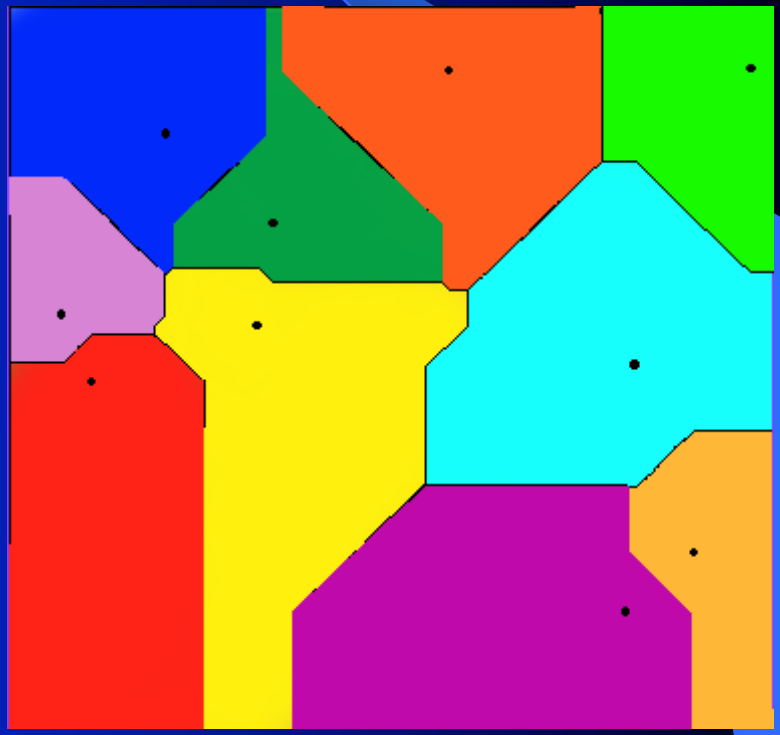
- Linear decision boundary between 2 closest points of different classes for 1-nn

Decision Surface

- Combining all the appropriate intersections gives a Voronoi diagram



Euclidean distance – each point a unique class



Same points - Manhattan distance

k-Nearest Neighbor (cont)

- Usually do distance weighted voting where each nearest neighbor vote is scaled inversely to its distance
- Inverse of distance squared is a common weight

$$w_i = \frac{1}{dist(x_q, x_i)^2}$$

- Gaussian is another common distance weight
- In this case the *k* value is more robust, could let *k* be even and/or be larger (even all points if desired), because the more distant points have negligible influence
- $w = 1$ for the non-weighted version

Challenge Question - *k*-Nearest Neighbor

- Assume the following data set
- Assume a new point (2, 6)
  - For nearest neighbor distance use Manhattan distance
  - What would the output be for 3-nn with no distance weighting? What is the total vote?
  - What would the output be for 3-nn with squared inverse distance weighting? What is the total vote?

A. A A
B. A B
C. B A
D. B B
E. None of the above

x	*y*	*Label*
1	5	A
0	8	B
9	9	B
10	10	A

$$w_i = \frac{1}{dist(x_q, x_i)^2}$$

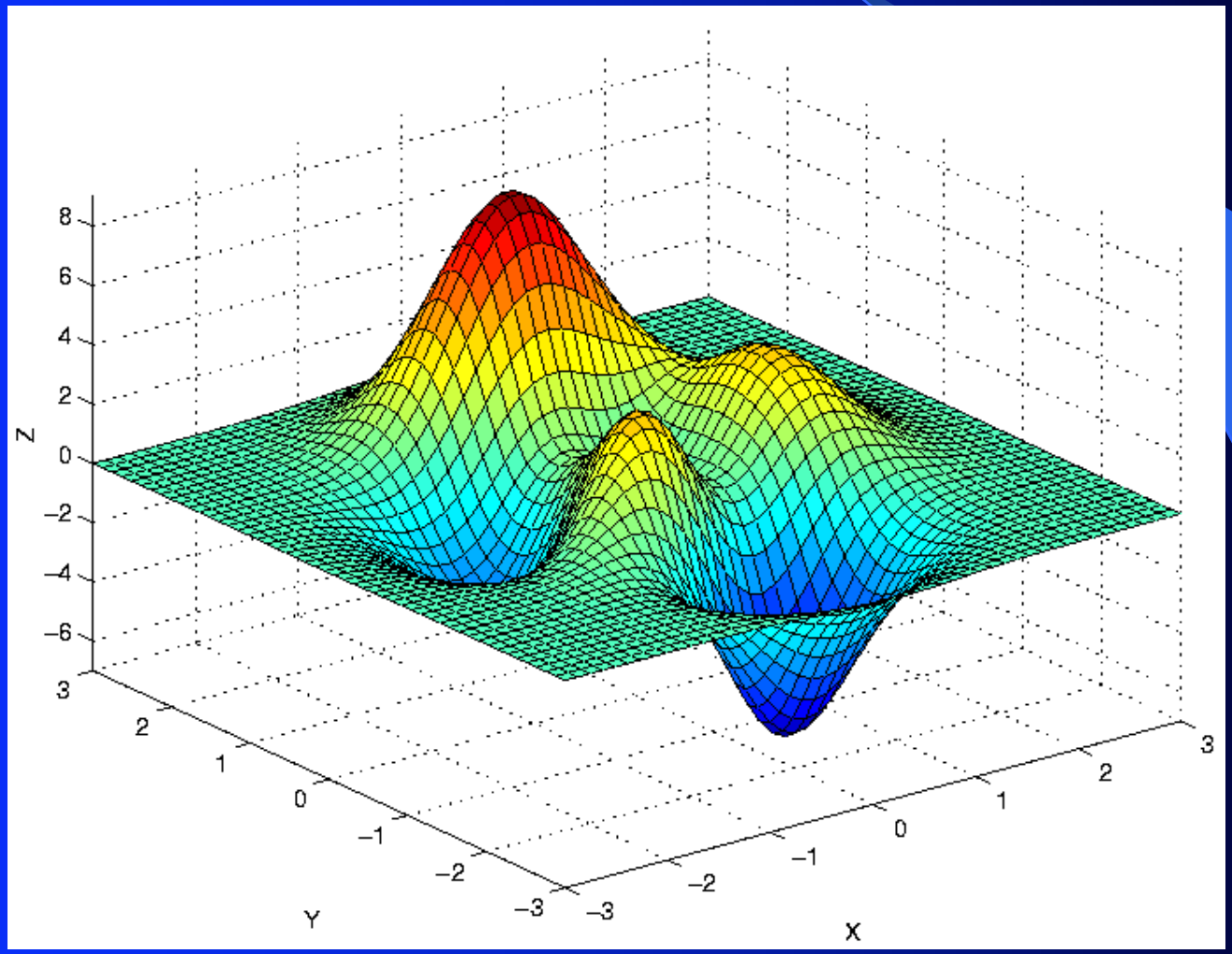
Challenge Question - *k*-Nearest Neighbor

- Assume the following data set
- Assume a new point (2, 6)
  - For nearest neighbor distance use Manhattan distance
  - What would the output be for 3-nn with no distance weighting? What is the total vote? – B wins with vote 2 out of 3
  - What would the output be for 3-nn with distance weighting? What is the total vote? A wins with vote .25 vs B vote of .0625+.01=.0725

x	*y*	*Label*	*Distance*	*Weighted Vote*
1	5	A	1 + 1 = 2	$1/2^2 = .25$
0	8	B	2 + 2 = 4	$1/4^2 = .0625$
9	9	B	7 + 3 = 10	$1/10^2 = .01$
10	10	A	8 + 4 = 12	$1/12^2 = .0069$

Regression with *k*-nn

- Can do regression by letting the output be the mean of the *k* nearest neighbors



Weighted Regression with *k*-nn

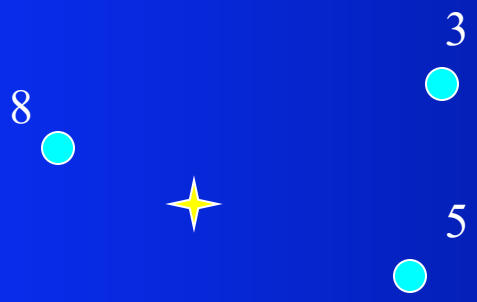
- Can do weighted regression by letting the output be the weighted mean of the *k* nearest neighbors
- For distance weighted regression

$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

$$w_i = \frac{1}{dist(x_q, x_i)^2}$$

- Where $f(x)$ is the output value for instance x
- $w = 1$ for non-weighted

Regression Example



$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

$$w_i = \frac{1}{dist(x_q, x_i)^2}$$

- What is the value of the new instance?
- Assume dist(x_q, n_8) = 2, dist(x_q, n_5) = 3, dist(x_q, n_3) = 4
- $f(x_q) = (8/2^2 + 5/3^2 + 3/4^2)/(1/2^2 + 1/3^2 + 1/4^2) = 2.74/.42 = 6.5$
- The denominator renormalizes the value

k-Nearest Neighbor Homework

- Assume the following training set
- Assume a new point (.5, .2)
  - Use Manhattan distance and show work
  - What would the output class for 3-nn be with no distance weighting?
  - What would the output class for 3-nn be with squared inverse distance weighting?
  - What would the 3-nn regression value be for the point if we used the regression values rather than class labels? Show results for *both* no distance weighting and squared inverse distance weighting.

x	*y*	*Class Label*	*Regression Label*
.3	.8	A	.6
-.3	1.6	B	-.3
.9	0	B	.8
1	1	A	1.2

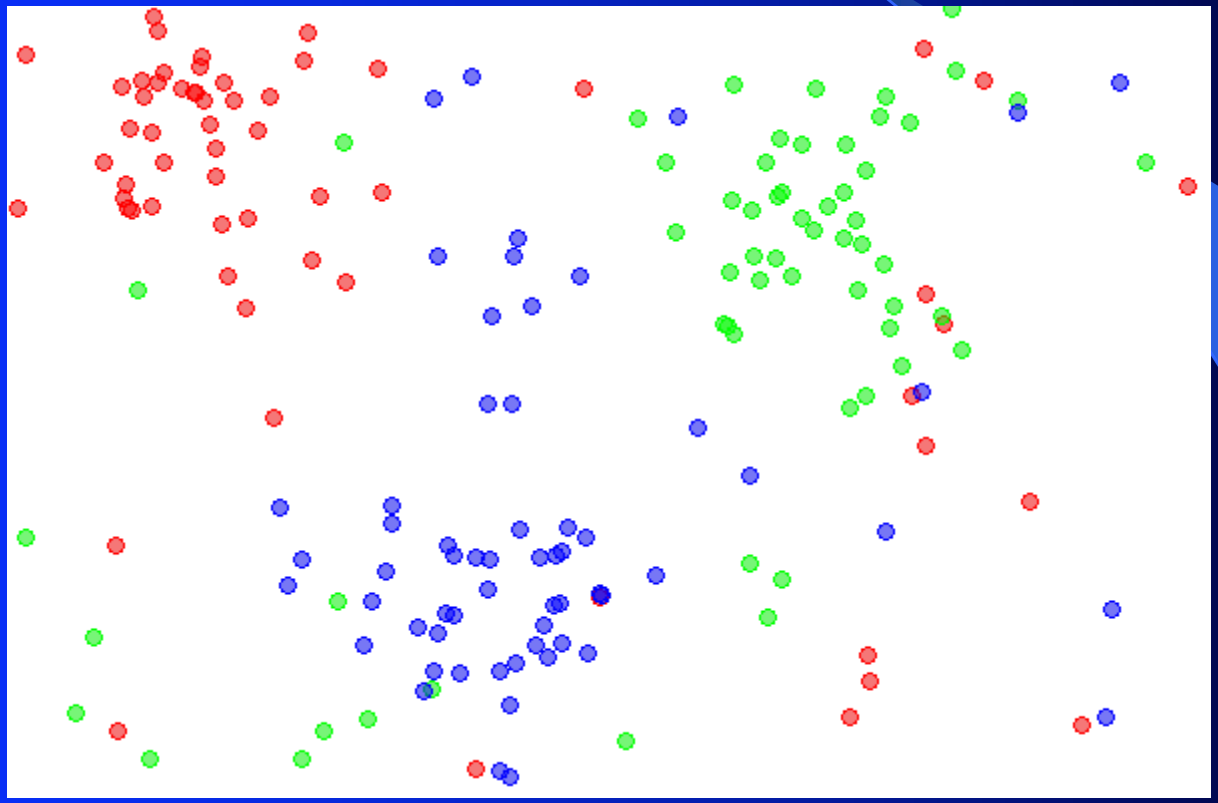
Attribute Weighting

- Normalize Features!
- One of the main weaknesses of nearest neighbor is irrelevant features, since they can dominate the distance
  - Example: assume 2 relevant and 10 irrelevant features
- Most learning algorithms weight the attributes (e.g. MLP and Decisions Trees do higher order weighting of features)
- Could do attribute weighting - No longer lazy evaluation since you need to come up with a portion of your hypothesis (attribute weights) before generalizing
- Still an open area of research
  - Higher order weighting – 1st order helps, but not enough
  - Even if all features are relevant features, all distances become similar as number of features increases, since not all features are relevant at the same time, and the currently irrelevant ones can dominate distance
  - An issue with all pure distance based techniques, need higher-order weighting to ignore *currently* irrelevant features
  - Dimensionality reduction can be useful (feature pre-processing, PCA, NLDR, etc.)

Reduction Techniques

- Create a subset or other representative set of prototype nodes
  - Faster execution, and could even improve accuracy if noisy instances removed
- Approaches
  - Leave-one-out reduction - Drop instance if it would still be classified correctly
  - Growth algorithm - Only add instance if it is not already classified correctly - both order dependent, similar results
  - More global optimizing approaches
  - Just keep central points – lower accuracy (mostly linear Voronoi decision surface), best space savings
  - Just keep border points, best accuracy (pre-process noisy instances – Drop5)
  - Drop 5 (Wilson & Martinez) maintains good accuracy with approximately 15% of the original instances
    - Wilson, D. R. and Martinez, T. R., Reduction Techniques for Exemplar-Based Learning Algorithms, *Machine Learning Journal,* vol. **38**, no. 3, pp. 257-286, 2000.

Reduction Techniques



Algorithm	Clean	Size%	Noisy	Size%
kNN	**82.11**	100.00	**78.93**	100.00
CNN	**76.48**	26.69	**68.14**	38.29
SNN	**75.44**	31.63	**74.60**	48.60
IB2	**76.58**	26.64	**67.80**	38.27
IB3	**78.85**	16.13	**72.09**	18.85
DEL	**80.65**	16.88	**78.16**	8.57
DROP1	**72.65**	8.41	**71.24**	8.00
DROP2	**81.07**	14.03	**79.99**	14.75
DROP3	**81.14**	14.31	**80.00**	11.49
DROP4	**81.11**	17.30	**79.57**	14.74
DROP5	**81.39**	16.09	**79.95**	15.52
ENN	**80.95**	83.34	**80.19**	74.91
RENN	**80.09**	80.92	**79.65**	72.53
AllKNN	**81.01**	77.44	**79.98**	64.58
ELGrow	**75.68**	1.83	**73.67**	1.88
Explore	**79.24**	2.01	**77.96**	2.03
Average	**79.06**	32.54	**76.36**	32.83

Distance Metrics

- Wilson, D. R. and Martinez, T. R., Improved Heterogeneous Distance Functions, *Journal of Artificial Intelligence Research,* vol. **6**, no. 1, pp. 1-34, 1997.
- Normalization of features - critical
- Don't know values in novel or data set instances
  - Can do some type of imputation and then normal distance
  - Or have a distance (between 0-1) for don't know values
- Original main question: How best to handle nominal features

Minkowsky:

$$D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r \right)^{1/r}$$

Euclidean:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:

$$D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T Q(\boldsymbol{x} - \boldsymbol{y}) = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} (x_i - y_i) q_{ji} \right) (x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m} (\boldsymbol{x} - \boldsymbol{y})^{\mathrm{T}} V^{-1} (\boldsymbol{x} - \boldsymbol{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i} \left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}} \right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_x$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{i-1} \mathrm{sign}(x_i - x_j)\,\mathrm{sign}(y_i - y_j)$$

sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

Figure 1. Equations of selected distance functions. ($\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of m attribute values).

Value Difference Metric

- Assume a 2-output class task (A, B)
- Attribute 1 = Shape (Round, Square, Triangle, etc.)
- 10 total round instances
  - 6 class A and 4 class B
- 5 total square instances
  - 3 class A and 2 class B
- Since both attribute values suggest the same probabilities for the output class, the distance between Round and Square would be 0
  - If triangle and round suggested very different outputs, triangle and round would have a large distance
- Distance of two attribute values is a measure of how similar they are in inferring the output class

Value Difference Metric (VDM)

[Stanfill & Waltz, 1986]

Providing appropriate distance measurements for nominal attributes.

$$vdm_a(x,y) = \sum_{c=1}^{C} \left(\frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right)^2$$

$N_{a,x}$ = # times attribute a had value x

$N_{a,x,c}$ = # times attribute a had value x and class was c

C = # output classes

Two values are considered closer if they have more similar classifications, i.e., if they have more similar correlations with the output classes.

VDM Example

$$vdm_a(x,y) = \sum_{c=1}^{C} \left(\frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right)^2$$

$N_{a,x}$ = # times attribute a had value x
$N_{a,x,c}$ = # times attribute a=x and class was c
C = # output classes

Color →	Class
red	0
green	0
red	1
blue	0
blue	1
red	1
blue	1
blue	1

x	$N_{1,x}$	class	$N_{1,x,c}$	$N_{1,x,c}/N_{1,x}$
red	3	0:	1	1/3=.33
		1:	2	2/3=.67
green	1	0:	1	1/1=1.0
		1:	0	0/1=0.0
blue	4	0:	1	1/4=.25
		1:	3	3/4=.75

vdm_{color}(red,green) = 0.889
vdm_{color}(red,blue) = 0.014
vdm_{color}(green,blue)=1.125

The Value Difference Metric (VDM) was introduced by Stanfill and Waltz (1986) to provide an appropriate distance function for nominal attributes. A simplified version of the VDM (without the weighting schemes) defines the distance between two values x and y of an attribute a as:

$$vdm_a(x,y) = \sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^q = \sum_{c=1}^{C} \left| P_{a,x,c} - P_{a,y,c} \right|^q \quad (8)$$

where

- $N_{a,x}$ is the number of instances in the training set T that have value x for attribute a;
- $N_{a,x,c}$ is the number of instances in T that have value x for attribute a and output class c;
- C is the number of output classes in the problem domain;
- q is a constant, usually 1 or 2; and
- $P_{a,x,c}$ is the conditional probability that the output class is c given that attribute a has the value x, i.e., $P(c \mid x_a)$. As can be seen from (8), $P_{a,x,c}$ is defined as:

In this section, we define a heterogeneous distance function *HVDM* that returns the distance between two input vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. It is defined as follows:

$$HVDM(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a,y_a)} \quad (11)$$

where m is the number of attributes. The function $d_a(x,y)$ returns a distance between the two values x and y for attribute a and is defined as:

$$d_a(x,y) = \begin{cases} 1, & \text{if } x \text{ or } y \text{ is unknown; otherwise...} \\ normalized\_vdm_a(x,y), & \text{if } a \text{ is nominal} \\ normalized\_diff_a(x,y), & \text{if } a \text{ is linear} \end{cases} \quad (12)$$

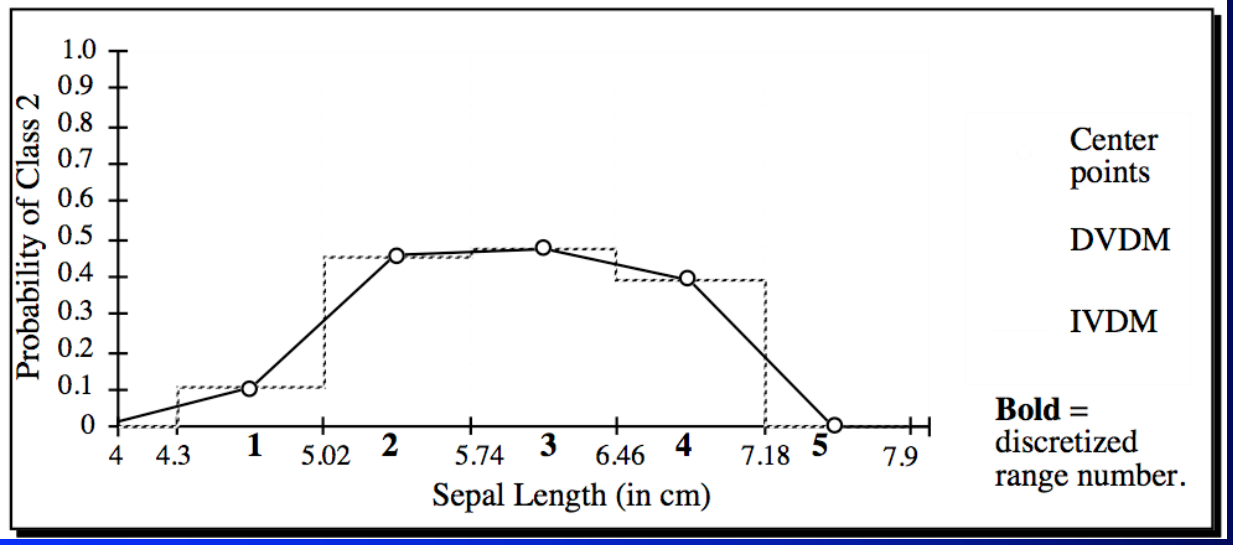
	Distance Function							# of inputs		
Database	Euclid	HOEM	**HVDM**	DVDM	**IVDM**	**WVDM**	#Inst.	Con	Int	Nom
Annealing	94.99	94.61	94.61	94.99	**96.11**	95.87	798	6	3	29
Audiology	60.50	72.00	**77.50**	**77.50**	**77.50**	**77.50**	200	0	0	69
Audiology (test)	41.67	75.00	**78.33**	**78.33**	**78.33**	**78.33**	26	0	0	69
Australian	80.58	81.16	81.45	**83.04**	80.58	82.46	690	6	0	8
Breast Cancer	94.99	95.28	94.99	**95.57**	**95.57**	**95.57**	699	0	9	0
Bridges	58.64	53.73	59.64	56.73	**60.55**	56.64	108	1	3	7
Credit Screening	78.99	81.01	80.87	80.14	80.14	**81.45**	690	6	0	9
Echocardiogram	94.82	94.82	94.82	**100.00**	**100.00**	98.57	132	7	0	2
Flag	48.95	48.84	55.82	**58.76**	57.66	58.74	194	3	7	18
Glass	**72.36**	70.52	**72.36**	56.06	70.54	71.49	214	9	0	0
Heart Disease	72.22	75.56	78.52	80.37	81.85	**82.96**	270	5	2	6
Heart (Cleveland)	73.94	74.96	76.56	79.86	78.90	**80.23**	303	5	2	6
Heart (Hungarian)	73.45	74.47	76.85	**81.30**	80.98	79.26	294	5	2	6
Heart (Long-Beach-Va)	**71.50**	71.00	65.50	71.00	66.00	68.00	200	5	2	6
Heart (More)	72.09	71.90	72.09	72.29	**73.33**	**73.33**	1541	5	2	6
Heart (Swiss)	**93.53**	91.86	89.49	88.59	87.88	88.72	123	5	2	6
Hepatitis	77.50	77.50	76.67	80.58	**82.58**	79.88	155	6	0	13
Horse-Colic	65.77	60.82	60.53	76.75	**76.78**	74.77	301	7	0	16
House-Votes-84	93.12	93.12	**95.17**	**95.17**	**95.17**	**95.17**	435	0	0	16
Image Segmentation	92.86	**93.57**	92.86	92.38	92.86	93.33	420	18	0	1
Ionosphere	86.32	86.33	86.32	**92.60**	91.17	91.44	351	34	0	0
Iris	94.67	95.33	94.67	92.00	94.67	**96.00**	150	4	0	0
LED+17 noise	42.90	42.90	**60.70**	**60.70**	**60.70**	**60.70**	10000	0	0	24
LED	**57.20**	**57.20**	56.40	56.40	56.40	56.40	1000	0	0	7
Liver Disorders	62.92	**63.47**	62.92	55.04	58.23	57.09	345	6	0	0
Monks-1	**77.08**	69.43	68.09	68.09	68.09	68.09	432	0	0	6
Monks-2	59.04	54.65	**97.50**	**97.50**	**97.50**	**97.50**	432	0	0	6
Monks-3	87.26	78.49	**100.00**	**100.00**	**100.00**	**100.00**	432	0	0	6
Mushroom	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	8124	0	1	21
Pima Indians Diabetes	71.09	70.31	71.09	**71.89**	69.28	70.32	768	8	0	0
Promoters	73.73	82.09	**92.36**	**92.36**	**92.36**	**92.36**	106	0	0	57
Satellite Image	90.21	**90.24**	90.21	87.06	89.79	89.33	4435	36	0	0
Shuttle	**99.78**	**99.78**	**99.78**	96.17	99.77	99.61	9253	9	0	0
Sonar	**87.02**	86.60	**87.02**	78.45	84.17	84.19	208	60	0	0
Soybean (Large)	87.26	89.20	90.88	**92.18**	**92.18**	**92.18**	307	0	6	29
Soybean (Small)	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	47	0	6	29
Thyroid (Allbp)	94.89	94.89	95.00	94.86	**95.32**	95.29	2800	6	0	22
Thyroid (Allhyper)	97.00	97.00	96.86	96.93	**97.86**	97.50	2800	6	0	22
Thyroid (Allhypo)	90.39	90.39	90.29	89.36	**96.07**	90.18	2800	6	0	22
Thyroid (Allrep)	96.14	96.14	96.11	96.86	**98.43**	97.07	2800	6	0	22
Thyroid (Dis)	98.21	98.21	98.21	**98.29**	98.04	98.00	2800	6	0	22
Thyroid (Hypothyroid)	93.42	93.42	93.36	93.01	**98.07**	96.96	3163	7	0	18
Thyroid (Sick-Euthyroid)	68.23	68.23	68.23	88.24	**95.07**	94.40	3163	7	0	18
Thyroid (Sick)	86.93	86.89	86.61	88.82	96.86	**97.11**	2800	6	0	22
Vehicle	**70.93**	70.22	70.93	63.72	69.27	65.37	846	18	0	0
Vowel	**99.24**	98.86	**99.24**	91.47	97.53	96.21	528	10	0	0
Wine	95.46	95.46	95.46	94.38	**97.78**	97.22	178	13	0	0
Zoo	97.78	94.44	**98.89**	**98.89**	**98.89**	**98.89**	90	0	0	16
Average:	80.78	81.29	83.79	84.06	**85.56**	85.24				

Interpolated VDM (IVDM)

$$IVDM(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} ivdm_i(x_i, y_i)^2$$

$$ivdm_a(x,y) = \begin{cases} vdm_a(x,y), & \text{if } a \text{ is discrete} \\ \sum_{c=1}^{C} \left| p_{a,c}(x) - p_{a,c}(y) \right|^2, & \text{otherwise} \end{cases}$$

$$p_{a,c}(x) = P_{a,u,c} + \left(\frac{x - mid_{a,u}}{mid_{a,u+1} - mid_{a,u}} \right) * (P_{a,u+1,c} - P_{a,u,c})$$



IVDM

- Distance Metrics make a difference
- IVDM also helps deal with the many/irrelevant feature problem of *k*-NN, because features only add significantly to the overall distance if that distance leads to different outputs
- Two features which tend to lead to the same output probabilities (exactly what irrelevant features should do) will have 0 or little distance, while their Euclidean distance could have been significantly larger
- Need to take it further to find distance approaches taking into account higher order combinations between features in the distance metric

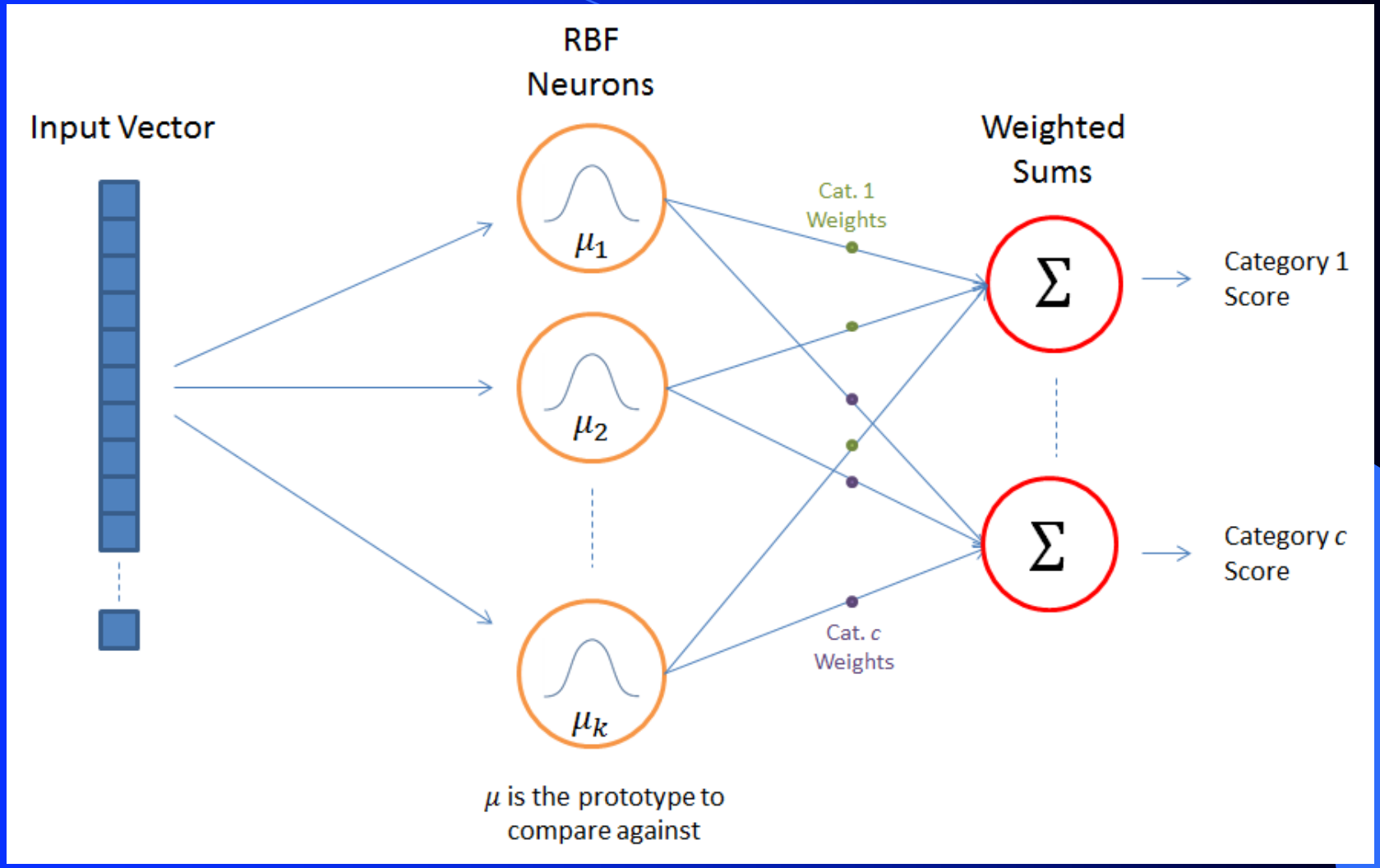
k-Nearest Neighbor Notes

- Note that full "Leave one out" CV is easy with *k*-nn
- Very powerful yet simple scheme which does well on many tasks
- Overfitting (less of an issue) handled by using larger *k*
- Struggles with irrelevant inputs
  - Needs better incorporation of feature weighting schemes
- Issues with distance with very high dimensionality tasks
  - Too many features wash out effects of the specifically important ones (akin to the irrelevant feature problem)
  - May need distance metrics other than Euclidean distances
- Also may be helpful to reduce total # of instances
  - Efficiency
  - Sometimes accuracy

K-nn Lab

- See Learning Suite

Radial Basis Function Networks

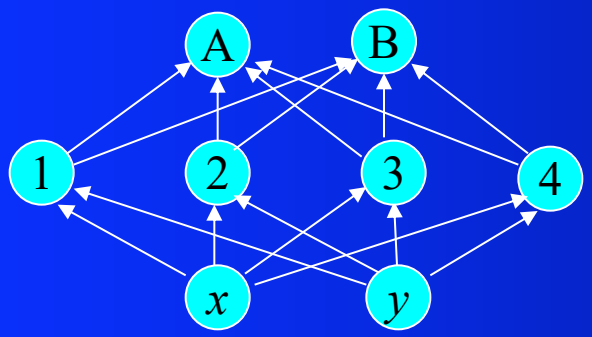


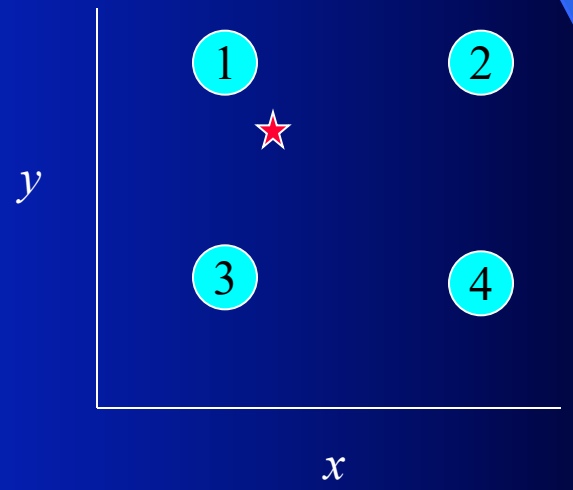
$$f(x) = w_0 + \sum_{i=1}^{h} w_i K(d(\mu_i, x))$$

Where $f(x)$ is the output of each linear output node, K is RBF (kernel function) and d is distance

Radial Basis Function (RBF) Networks

- One linear output node per class – with weights and bias
- Each hidden (prototype) node computes the distance from itself to the input instance (Gaussian is common) – *not* like an MLP hidden node
- An arbitrary number of prototype nodes form a hidden layer in the Radial Basis Function network – prototype nodes typically non-adaptive
- The prototype layer expands the input space into a new prototype space. Translates the data set into a new set with more features
- Output layer weights are learned with the linear model delta rule
  - Not a preset label vote like in k-nearest neighbor
- Thus, output nodes learn 1st order prototype weightings for each class





$$\Delta w_i = c(t - net)x_i$$

Radial Basis Function Networks

- Neural Network variation of nearest neighbor algorithm
- Output layer execution and weight learning
  - Highest node/class net value wins (or can output confidences)
  - Each node collects weighted votes from prototypes – unlearned weighting from distance (prototype activation – like *k*-nn), but unlike *k*-nn, vote value is learned and all nodes always have a vote (weight) for every output class
  - Weight learning - Delta rule variations or direct matrix weight calculation – linear or non-linear node activation function
    - Could use an MLP at the top layer if desired
- Key Issue: How many prototype nodes should there be and where should they be placed (means)
- Prototype node sphere of influence – Kernel basis function (deviation) – like choosing *k* for *k*-NN
  - Too small – less generalization, should have some overlap
  - Too large - saturation, lose local effects, longer training

Node Placement

- Random Coverage - Prototypes potentially placed in areas where instances don't occur, Curse of dimensionality
- One prototype node for each instance of the training set
- Random subset of training set instances
- Clustering - Unsupervised or supervised - *k*-means style vs. constructive
- Genetic Algorithms
- Node adjustment – Adaptive prototypes (Competitive Learning style)
- Dynamic addition and deletion of prototype nodes

RBF Homework – Not currently required

- Assume you have an RBF with
  - Two inputs
  - Three output classes A, B, and C (linear units)
  - Three prototype nodes at (0,0), (.5,1) and (1,.5)
  - The radial basis function of the prototype nodes is
    - max(0, 1 – Manhattan distance between the node and the instance in question)
  - Assume no bias and initial weights of .6 into output node A, -.4 into output node B, and 0 into output node C
  - Assume top layer training is the delta rule with LR = .1
- Assume we input the single instance .6 .8
  - Which class would be the winner?
  - What would the weights be updated to if it were a training instance of .6 .8 with target class B? (thus B has target 1 and A and C have target 0)

$$\Delta w_i = c(t - net)x_i$$

RBF vs. BP

- Line vs. Sphere - mix-and-match approaches
  - Multiple spheres still create Voronoi decision surfaces
- Potential Faster Training - nearest neighbor localization - Yet more data and hidden nodes typically needed
- Local vs Distributed, less extrapolation (ala BP), have reject capability (avoid false positives)
- RBF will have problems with irrelevant features just like nearest neighbor (or any distance based approach which treats all inputs equally)
  - Could be improved by adding learning into the prototype layer to learn attribute weighting

Distributed vs Local

- MLP vs K-NN (RBF) – exponential vs linear representation potential – but how useable is it? – overfit, exponential training data? Which is best is an open question.
- Below are decision surfaces for MLP with 3 hidden nodes, and K-NN with 3 nodes