



University of Toronto

Learning Pattern Classification

Navid Azizi
ECE1390



Outline

- **Problem Statement & Background**
- **Classifier Performance**
- **Algorithms**
 - **Nearest Neighbour Classifiers**
 - **Kernel Classifiers**
 - **Histograms and Classification Trees**
 - **Neural Networks**
- **Comparison of Algorithms**
- **Conclusion**



Problem Statement

■ Who is Learning?

- Anything that is immersed and interacting with an environment

■ How do you Learn?

- With
 - ◆ Data gathered from the environment
 - ◆ Prior knowledge and assumptions about environment
- Form internal representation or model of the environment

■ What do you do with Learning?

- You predict future outcomes



What is Learning Used For

- Learning algorithms have been widely used in computer science such as
 - Language Identification
 - Regression Estimation
 - Pattern Classification
 - Etc.

- The rest of this presentation (and course) will deal with pattern classification
 - Two-class pattern classification



Two-Class Pattern Classification

- There are two classes of objects

- Class 0
- Class 1

$$y \in \{0,1\}$$

- Information about Objects summarized through

- Real-valued measurements called features (usually d of them)

- All the features together comprise a feature vector $x \in \mathcal{R}^d$

- To model uncertainty about class objects

- Assume a priori probabilities

- ◆ P_0
- ◆ P_1



Relationship between Class and the Feature Set

- To model the relationship between the class to which the object belongs and the feature vector
 - Class-conditional distribution function

$$F_y(x)$$

$$F_0(x), F_1(x)$$

- Decision rule that takes a feature set and decide which class the object belongs to

$$g : \mathcal{R}^d \rightarrow \{0,1\}$$



Performance of Decision Rules

- Given a classifier g , an object with known class Y , and a feature vector for the object X , the performance can be given by

$$L(g) = P\{g(X) \neq Y\}$$

- What is the performance of a good decision rule?



Best Decision Rule

- If the a priori probabilities P_0, P_1 and the conditional distributions $F_0(y), F_1(y)$ are known
 - We have the optimal decision rule
 - ◆ Minimizes the probability of error
 - Bayes decision rule g^*

- *Compute a posteriori probabilities*

$$\eta_0(x) = P\{Y = 0 \mid X = x\}$$

$$\eta_1(x) = P\{Y = 1 \mid X = x\}$$

- *Pick the class with the largest probability*

$$g^*(x) = \arg \min_{y \in \{0,1\}} \eta_y(x)$$

- *Performance of g^**

$$L^* = L(g^*) = E[\min\{n_0(X), n_1(X)\}]$$



But ...

- We don't know $P_0, P_1, F_0(x), F_1(x)$
- Instead we have previous labeled observations which are correct

$$D_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$$

- Observations are assumed
 - to be independent
 - Y_k distributed according to $\{P_0, P_1\}$
 - X_k drawn according to $F_{Y_k}(x)$
- So now our decision rule depends on D_n
 - D_n is random



Performance

■ Aims for a Classifier

- Close to optimal Bayes error rate
- Consistent

$$(n \rightarrow \infty) \quad L(g_n) \rightarrow L^*$$

- ◆ Even if a classifier is consistent the convergence may be arbitrarily slow



Nearest Neighbour Classifiers

- Assign the input feature vector to the class indicated by the label of the nearest vector
 - To what are we comparing?
 - ◆ To all the samples in the reference sample

$$D_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$$

- How do define distance?
 - ◆ Choose a metric $d(X, X')$ that returns a scalar
 - ◆ More on this later



Nearest Neighbour Example

- Say our two sets are insects and dinosaurs
 - Feature vector is a two-dimensional vector
 - ◆ [weight, height]

- Training Set
 - $D = \{([1750, 10], \text{dinosaur}), ([2000, 5], \text{dinosaur}), ([0.05, 0.1], \text{insect})\}$
- Sample $X = [1800, 2]$

- Metric $D(X, X') = |X.w - X'.w| + |X.h - X'.h|$
 - Distances are 58, 203, 1801.85
- Our sample is nearest to $([1750, 10], \text{dinosaur})$
 - Designate our sample as a dinosaur



k-Nearest Neighbour Classifier

■ More general than Nearest Neighbour

- Assign a feature vector X to the pattern class that appears most frequently among the k nearest neighbours

■ Mathematical Formulation

- Assuming, indices of the label feature vectors in D_n are permuted to satisfy

$$d(X, X_1) \leq d(X, X_2) \leq \dots \leq d(X, X_k)$$

$$d(X, X_j) \geq d(X, X_k) \text{ for } j = k + 1, \dots, n$$

- The k -nearest neighbours are

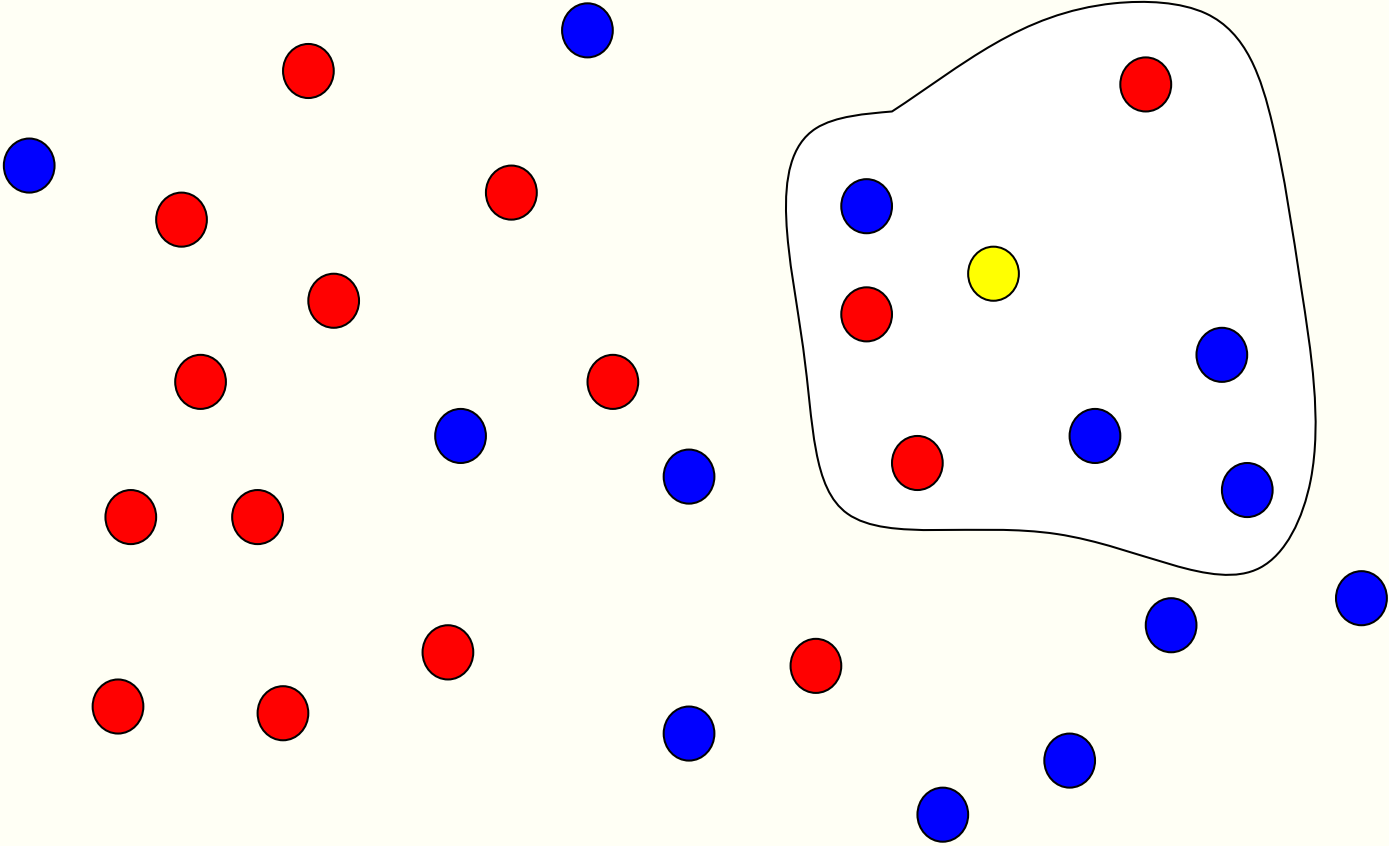
$$\{(X_1, Y_1), (X_2, Y_2), \dots, (X_k, Y_k)\}$$

- So we classify X to be

$$Y = \text{maj}(Y_1, \dots, Y_k)$$



7-Nearest Neighbour Example





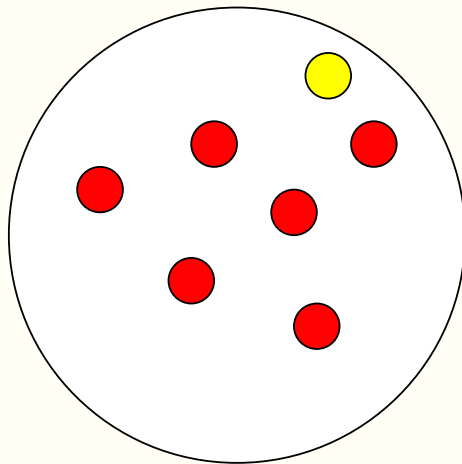
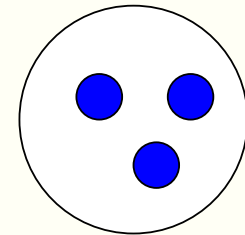
k-Nearest Neighbour Example

■ Characteristics of example

- Non-overlapping distribution
- Distance between classes is larger than diameter of classes

■ For infinite sample space

$$L_{\infty}(k) = L^* = 0$$



Makes a mistake if $n_0 < k$

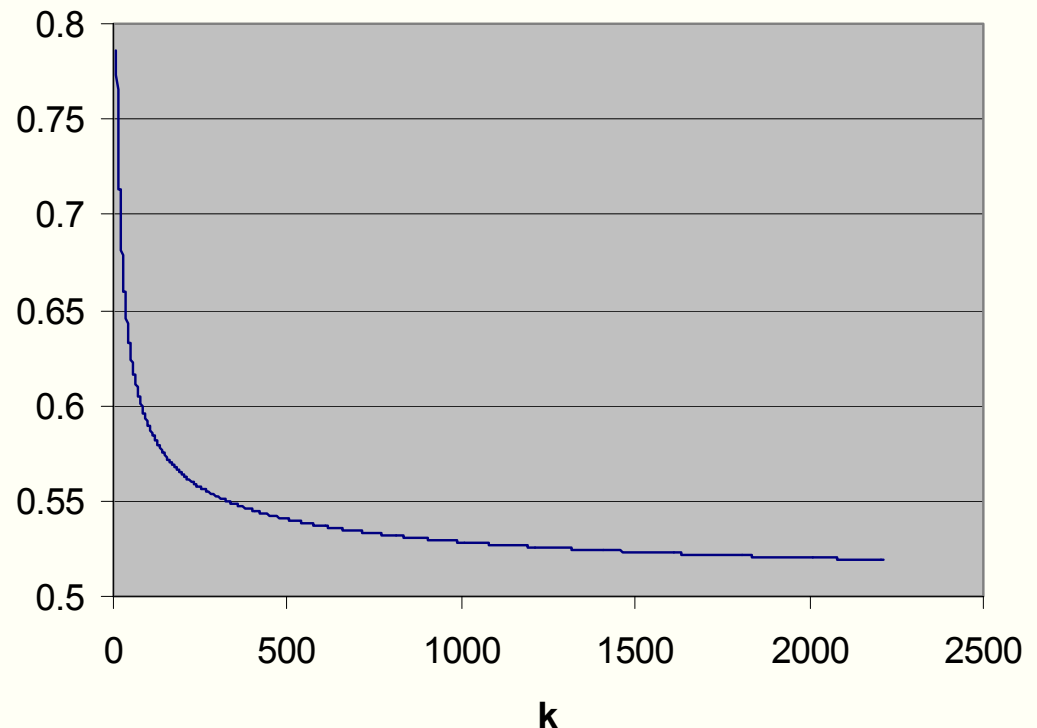


k-Nearest Neighbour Characteristics Infinite Sample Space

- How do we choose k?

$$L_{\infty}(k) \leq 2L^*(1 - L^*)$$

$$L_{\infty}(k) \leq L^* + L_{\infty}(1) \sqrt{\frac{2}{\pi \lceil k/2 \rceil}}$$





k-Nearest Neighbour Characteristics Finite Sample Space

- How do you choose the number of sample points

$$L_n(k) = L_\infty(k) + O(n^{-2/d})$$

- Larger the sample space, the closer we are to the ideal
- Complexity grows exponentially with dimension d

- Remark

- For fixed k , increasing n , error converges to $L_\infty(k)$
- Then as k increases, error converges to L^*



k-Nearest Neighbour Characteristics Finite Sample Space

- Is there a way of selecting k as the sample size increases

if $k \rightarrow \infty$ and $n \rightarrow \infty$ such that $k/n \rightarrow 0$

then $L_n(k) \rightarrow L^*$

- The idea is that the number of neighbours k considered in the decision should grow with sample size n but not too fast
- Under certain distributions $k=1$ is optimal (previous example)
- Also if L^* is small, no advantage choosing $k>3$



Choice of Metric

■ Non-trivial

- Components of the input vector may represent quantities that are quite different in nature
- Different units

■ Some metrics

- Global Metrics
 - ◆ Weighted Euclidean metric
- Local Metrics
 - ◆ Local Linear metric
- An optimal metric
 - ◆ A is a $d \times d$ matrix which is a nonsingular linear transformation

$$d(x, x') = \|A(x - x')\|_p$$

$$\|x\|_p = \sqrt[p]{|x_1|^p + \dots + |x_d|^p}$$



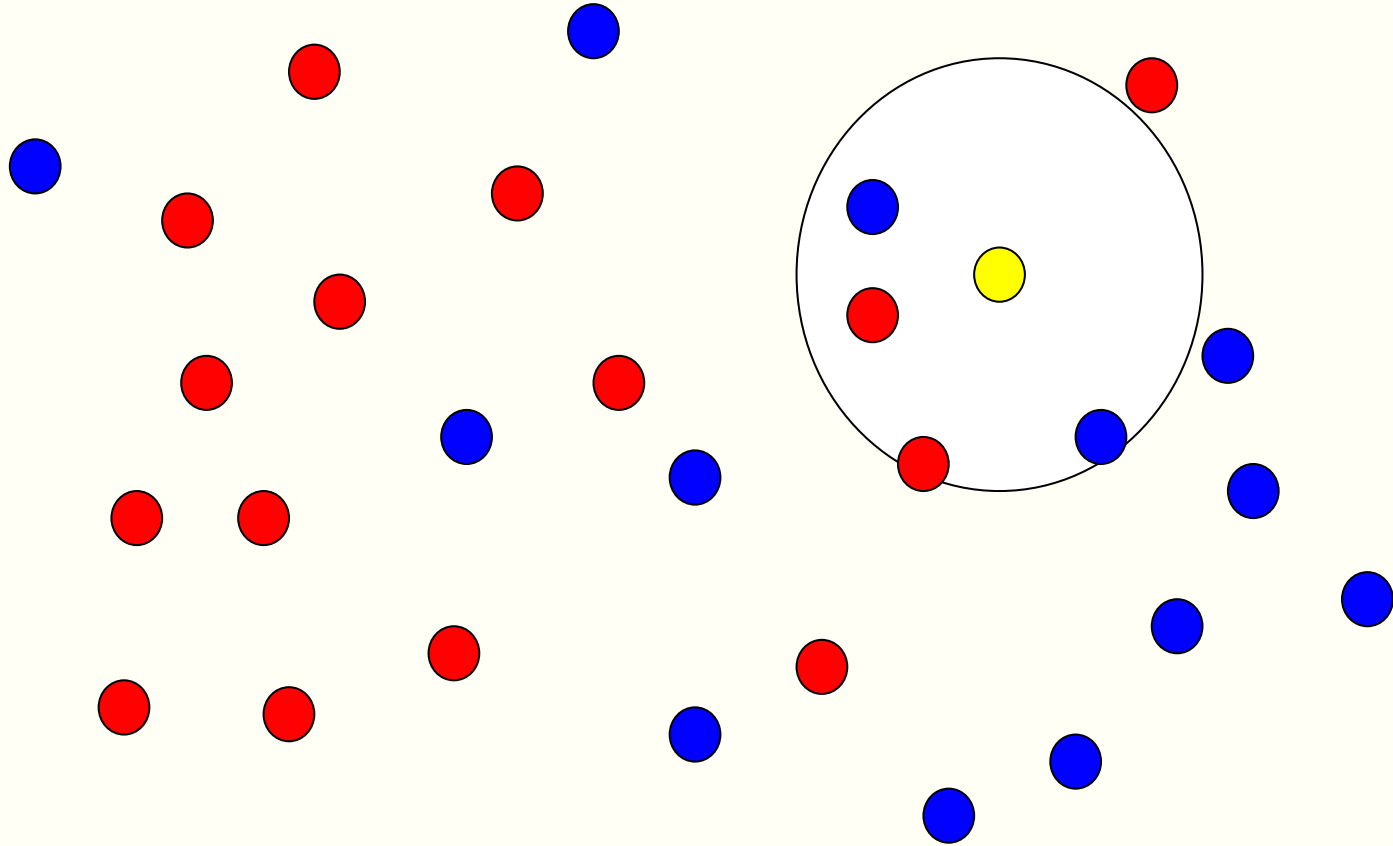
Kernel Classifiers

- **Instead of finding the k -nearest neighbours**
 - Find all the neighbours that are closer than some distance h

- **Moving Average Classifier**
 - The label assigned to X is 1 if and only if, among the training points within distance h to there are more points labeled by 1 than those labeled by 0



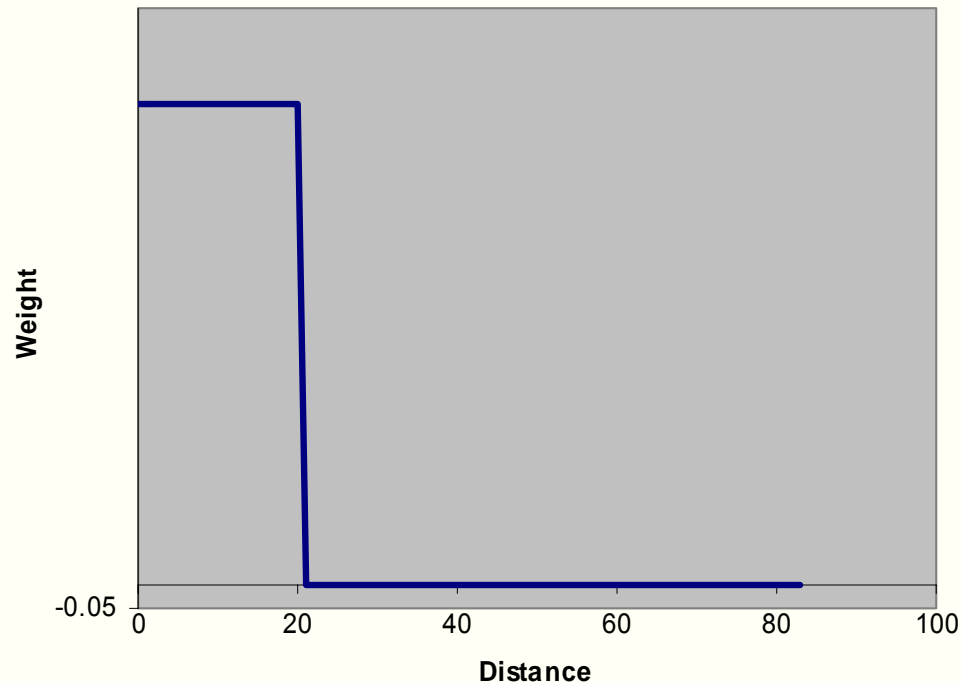
Moving Classifier Example





Moving Average Classifier

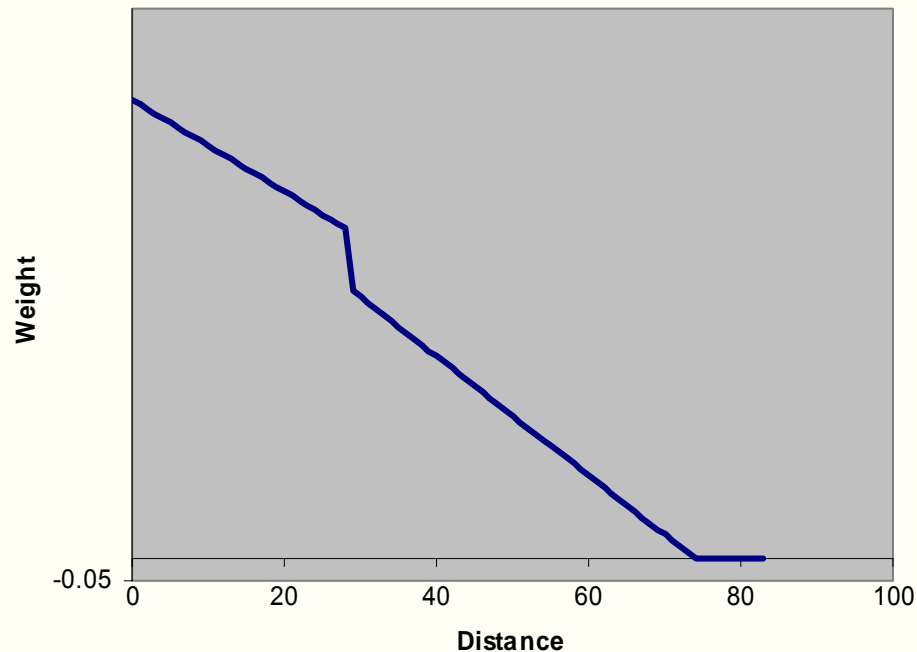
- The Moving Average Classifier is a specific example of the Kernel Classifiers
 - The Moving Average Classifier gives uniform weight to all points within distance h and zero weight to all other points.





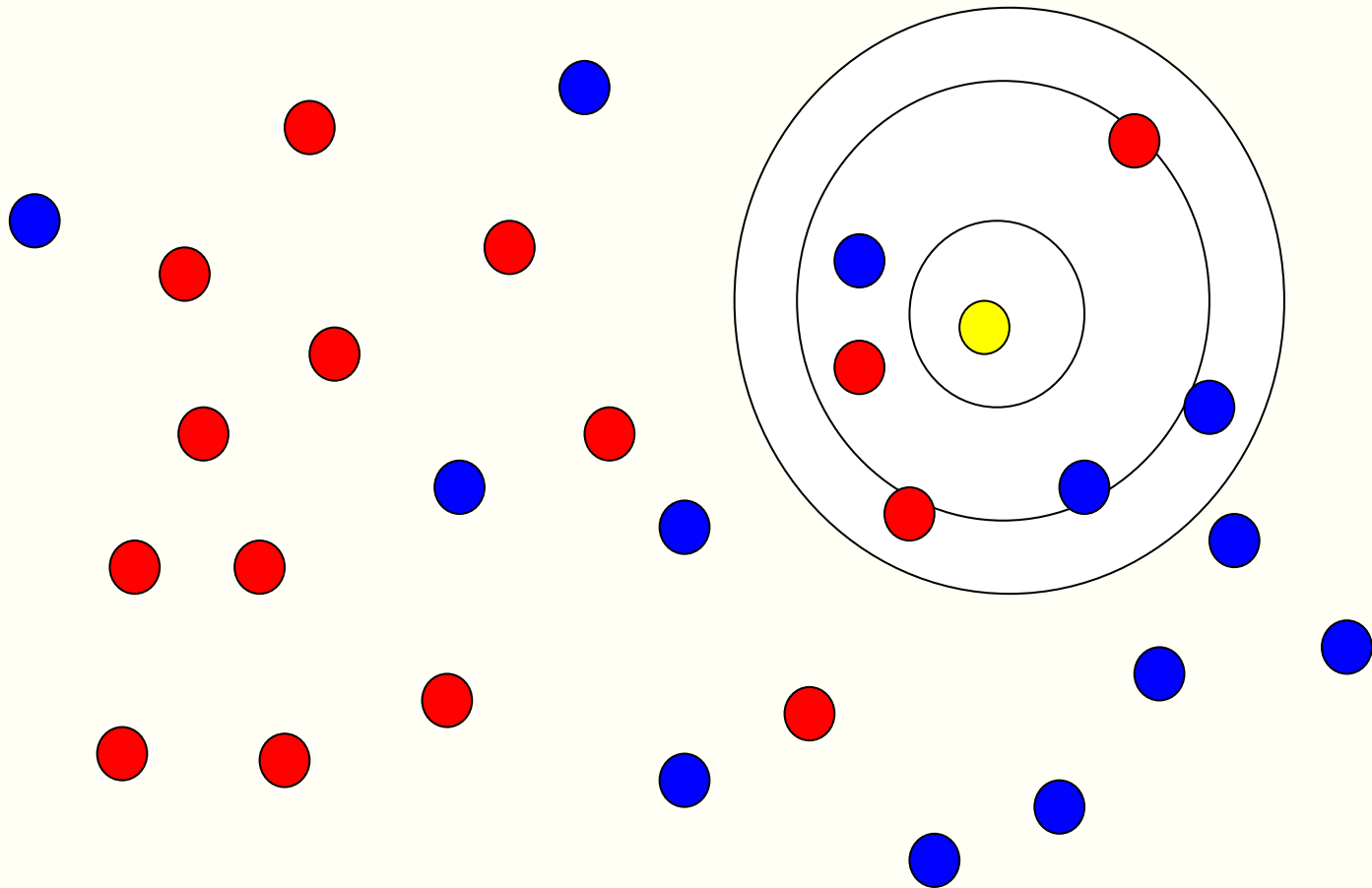
Kernel Classifier Rule

- The Kernel Classifier has a more general function
 - The kernel function is non-negative and monotonically decreasing





Kernel Classifier Example





Kernel Classifiers

■ Some Kernel Functions

- Gaussian $K(x) = e^{-\left\|\frac{x}{h}\right\|^2}$

- Cauchy $K(x) = 1 / \left(1 + \left\|\frac{x}{h}\right\|^{d+1}\right)$

■ h is the most important parameter

- h small: large relative points near X
 - ◆ Decision is very local
- h large: more points are considered
 - ◆ Decision is more stable



Radial Base Function Classifiers

■ Closely Related to Neural Network Classifiers

$$\sum_{i=1}^k a_i K\left(\frac{X - x_i}{h_i}\right)$$

■ Difference from Normal Kernel Classifiers

- Use k instead of all n points from the Training Set, $k \ll n$
- x_i , a_i and h_i are based on the training data

■ Example

- Group training data into k concentrated groups
- x_i 's are chosen as cluster centers



Performance of Kernel Classifiers

- What choice of K and h leads to consistent classification rule

$$\lim_{n \rightarrow \infty} L(g_n) = L^* \quad \text{if} \quad \lim_{n \rightarrow \infty} h = 0$$

- For high-dimensional spaces need very large values of h to guarantee that sufficiently many data points are taken into account at the decision to compensate for statistical variation
 - But then the local nature of the decision is lost
- Nontrivial



Histogram Classifiers

■ Partition the feature space

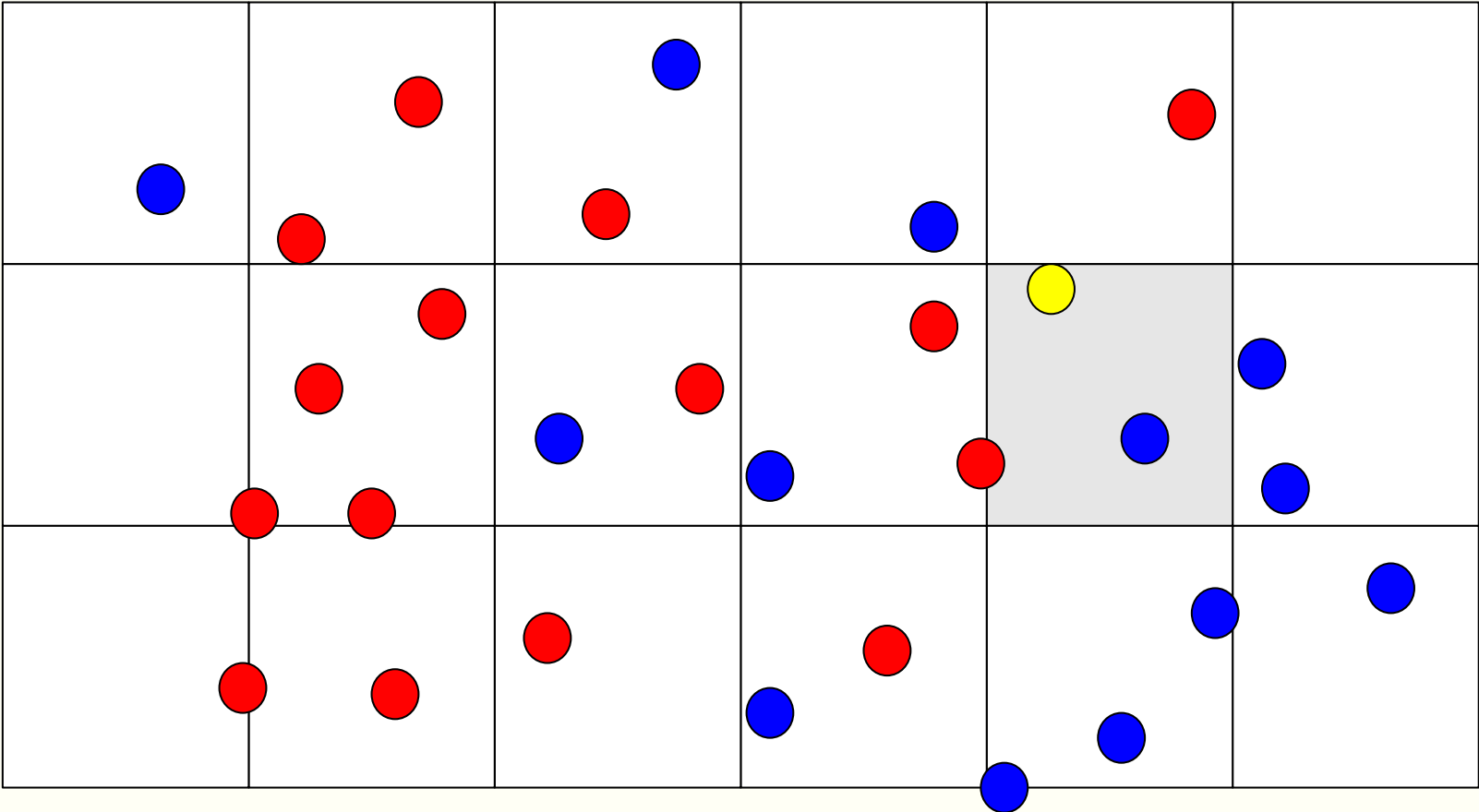
- Classification is made according to majority vote within the cell of the partition which the point falls

■ Regular Partition Mode

- Equal size cubes of size h
- h should be
 - ◆ Small enough to have “local” decisions
 - ◆ Large enough to average out statistical variation



Regular Histogram Classifier



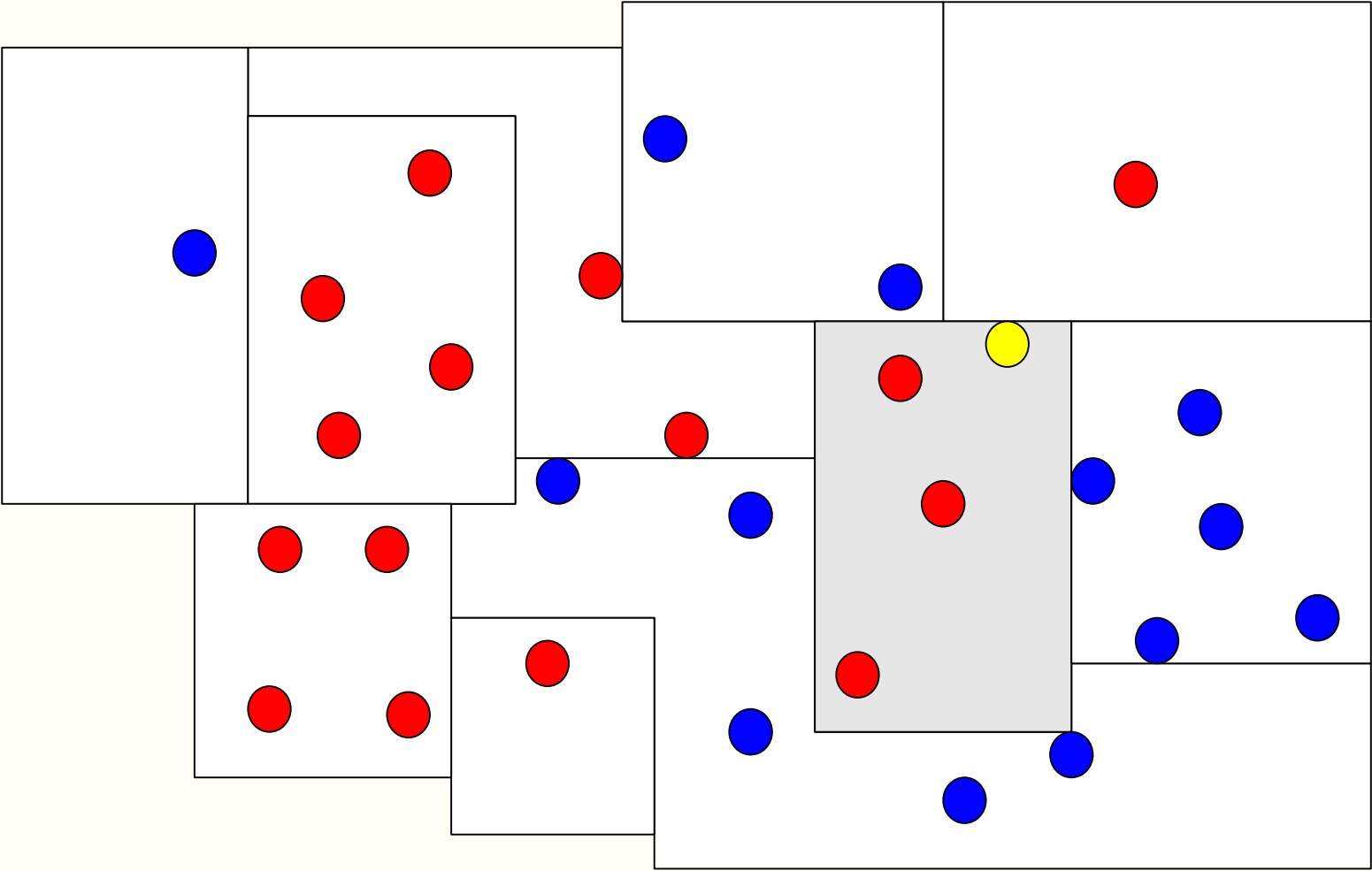


Regular Histogram Mode

- **For large dimensions conditions of h cannot be met**
 - Ex. $d=10$, X is distributed in $[0,1]^{10}$, $h=1/4$
 - Generates over a million cells
 - For practical sizes, h cannot be both small enough to be local or large enough to be statistically correct
- **Look at data and determine region sizes**
 - Some large cells
 - Some finer cells



Histogram Classifier Example





Binary Tree Classifiers

■ Partitions built recursively

- Split each cell respective to one co-ordinate
- Each split should represent a simple question
 - ◆ Is the i th component of X less than a ?

■ Advantages

- After tree has been built, classification is simple
- The classifier is easy to interpret



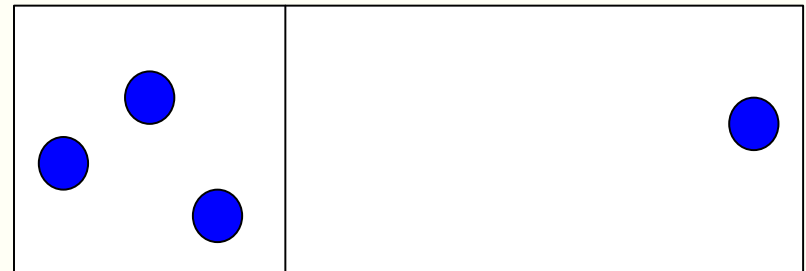
Binary Tree Classifiers

■ How do you choose where to perform the split?

- Split at the median point

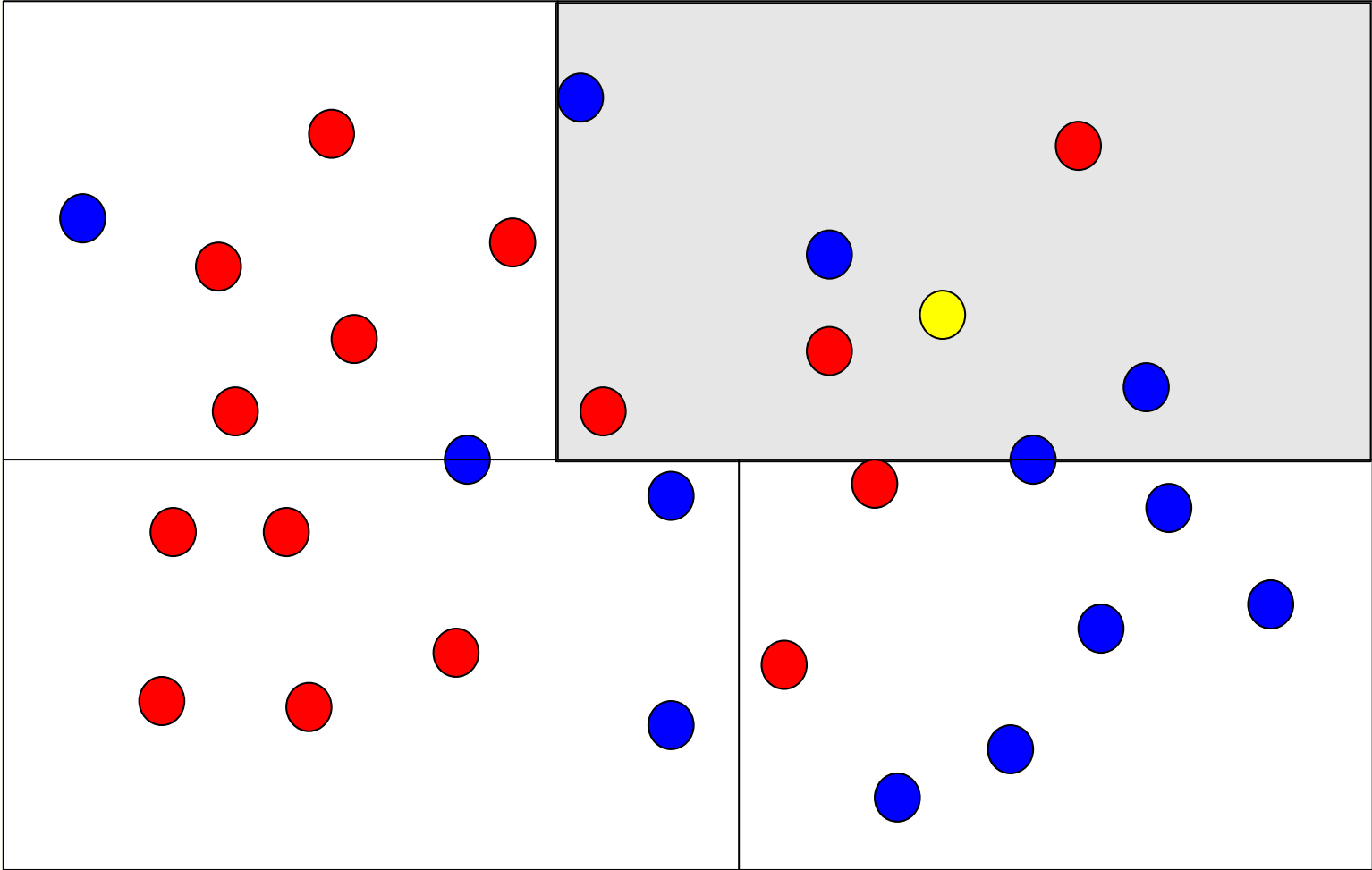


- Split to minimize deviations from the centroid
 - ◆ Can lead to consistent classification rule





Binary Tree Classifier Example



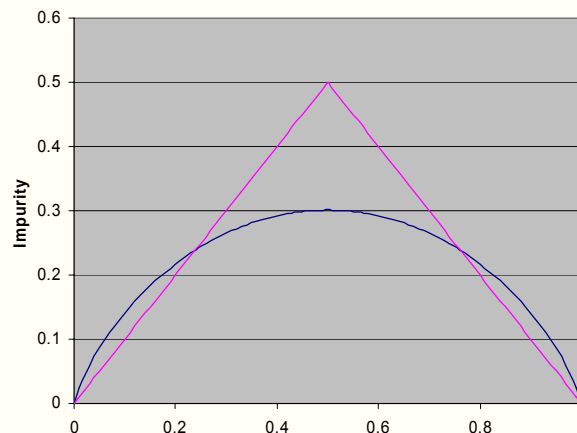


Splitting Criteria

- Disadvantages of above two splitting criteria
 - Discarding label information during tree-building phase

- Create Impurity function

- $[0,1] \rightarrow [0,\infty)$
- Symmetric around $\frac{1}{2}$
- $\Phi(0) = \Phi(1) = 0$



- Impurity

$$I_A = N \phi\left(\frac{N_0}{N}\right)$$

- Make cut that maximizes the difference $I_A - (I_{A'} - I_{A''})$



Splitting Criteria

■ Dangers of impurity-based growing

- Cells containing very few points will be preferentially split as improvement is much easier to achieve there
- Hack: Let tree grow, and then prune it to get best subtree

■ Better Solution

- Look ahead 2d steps, and select splits that minimize the training error
- Can be consistent



Vapnik-Chervonenkis Theory

- More math needed to understand neural networks
- Let C be a collection of decision rules ϕ
- Let L_C^* be the performance of the best rule from C
- Since we have limited training data we don't know the exact performance of a specific decision rule, but we hope for

$$L(\phi) \approx L_C^*$$

- Let ϕ_n^* be the best rule given the specific training data



More Vapnik-Chervonenkis

- Remember L^* is the performance of the optimal rule
- So we want to minimize the difference between our rule and the best rule. The difference can be rewritten as so

$$L(\phi_n^*) - L^* = (L(\phi_n^*) - L_C^*) + (L_C^* - L^*)$$

Estimation Error

Approximation Error



More Vapnik-Chervonenkis

■ Approximation Error

- How much we lose in performance by restricting our classifier to come from C

■ Estimation Error

- How much we lose in performance by our lack of knowledge of the exact performance of the various rules from C
- The loss we incur by estimating the actual performance by the empirical performance

■ Tradeoff in Rich class of rules C

- Reduce approximation error
- Greater danger that some rule from C just happens to fit very well with the particular data observed so far



More Vapnik-Chervonenkis

- **Need to know bound of Estimation Error**
 - Vapnik-Chervonenkis dimension of class V_C places bounds on error
- V_C of the class C is the largest integer n such that there exists a set of cardinality n that is shattered by C .
 - Won't go into definition

- **Bound**

$$L(\phi_n^*) - L_C^* \leq c \max\left(\sqrt{\frac{L_C^* V_C \log n}{n}}, \frac{V_C \log n}{n}\right)$$

- Idea: want V_C to be small
- But small V_C limits us to a non-rich class of functions, which may cause approximation error to increase



Neural Networks

- **An artificial neural network is an assembly of formal computation elements operating collectively and interconnected in various ways with the outputs of some elements functioning as inputs to others**
 - **A graph**

- **Described by the**
 - **Elementary computation units**
 - **Graph Interconnections**

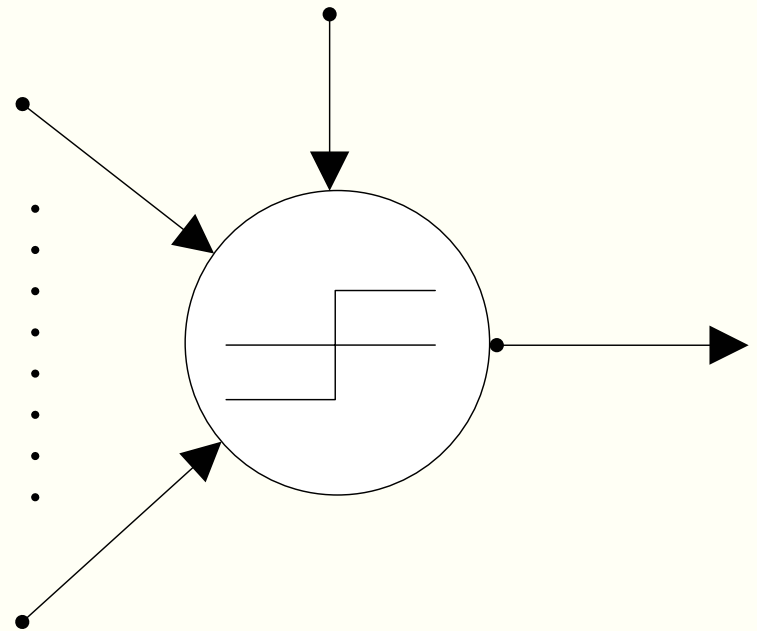


Neurons

■ Linear Threshold Model

- **d inputs**
- **Each has different weights**
- **Sum of all inputs**
- **+1 if larger than threshold**
- **-1 if smaller than threshold**

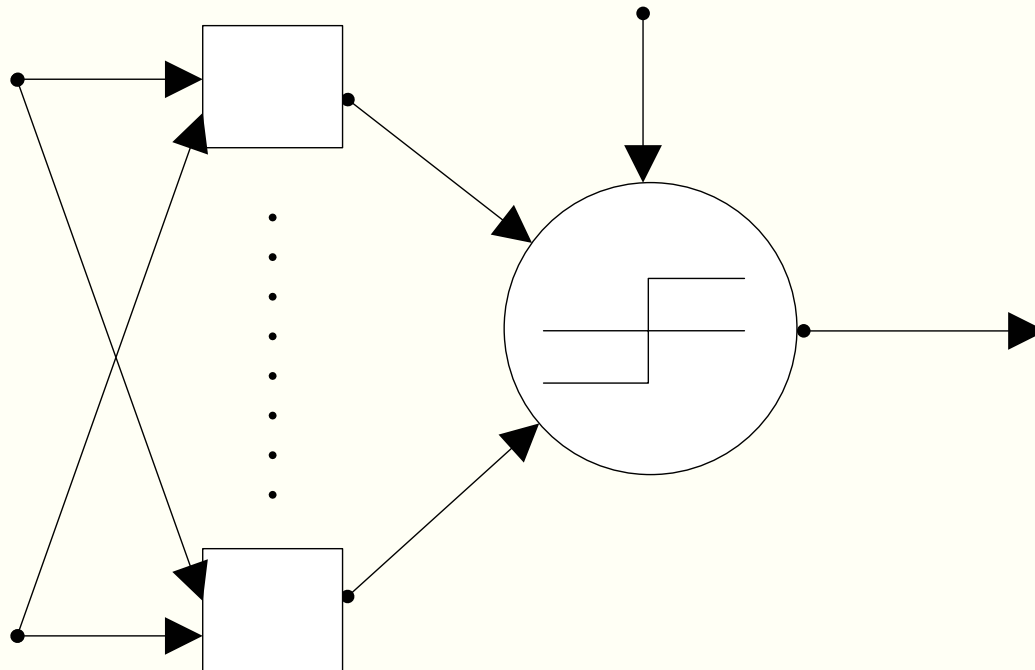
- **Can get rid of input if consider x_0 as input = -1 with w_0 as threshold**





Canonical Threshold Model

- Use functions of inputs



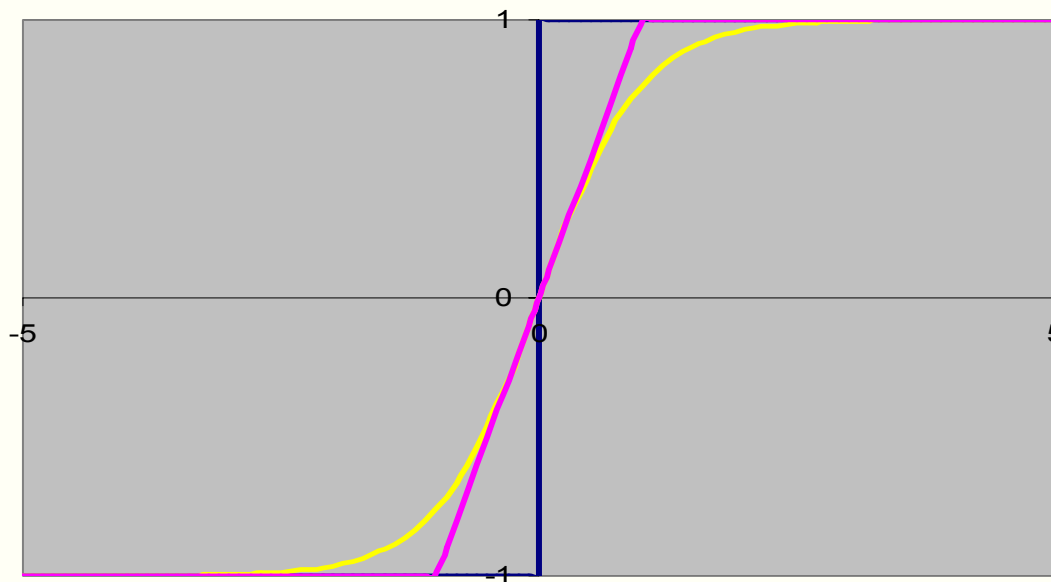


Sigmoid

- Instead of threshold function with binary output
 - Use activation function

$$\sigma(t) \rightarrow \begin{cases} -1, & \text{if } t \rightarrow -\infty \\ +1, & \text{if } t \rightarrow \infty \end{cases}$$

- Sigmoids are very popular since they are smooth and can be used for ad hoc learning

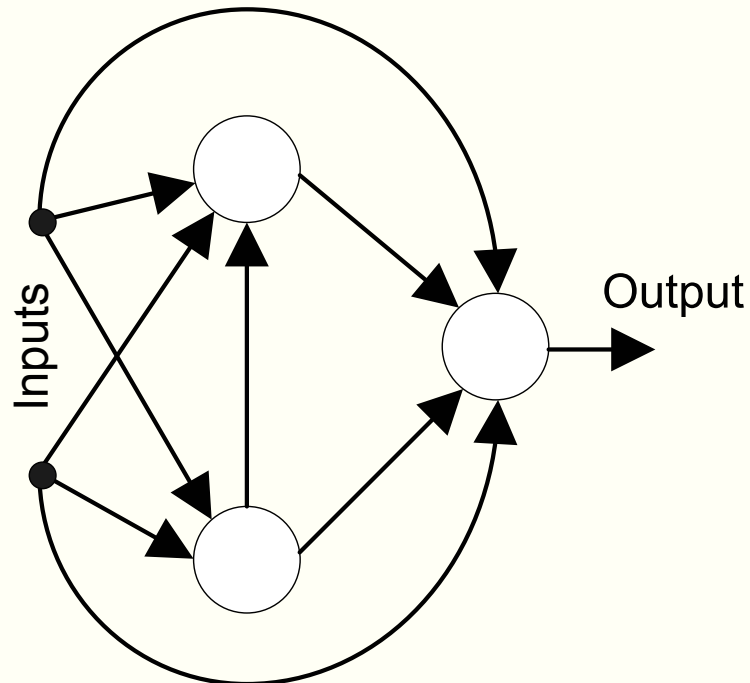




Networks

■ Graph

- **Set of vertices**
 - ◆ **Source nodes (external inputs)**
 - ◆ **Computation Nodes (neurons)**
 - ➔ **One of the computation nodes is the output node**
 - ➔ **Output node is a threshold function**
- **Set of edges**





Networks

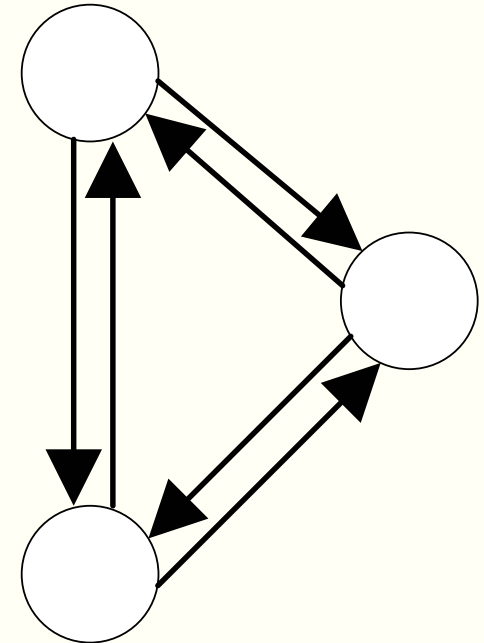
■ Networks can be

- **Acyclic**

- ◆ Rich family of classifiers can be engendered by such architectures by just varying the weights

- **Recurrent**

- ◆ Has cycles
- ◆ May have stability problems
 - ➔ Have undirected graph where connections are two-way





Universal Basis of Computation

- **Neurons for the universal basis of computation**
- **Finite problems**
 - $\text{NAND}(x_1, x_2) = \text{sgn}(-x_1 - x_2 + 1)$ $x_1, x_2 \in \{-1, +1\}$
 - NAND gates can perform any Boolean function in depth-two logic
- **Continuous problems**
 - It has been proved that a depth-two arbitrary-size network of with common activation function in the first layer and linear accumulation at the output can perform any continuous problem



Size of Neural Networks

- **Bayes classifier can be approximated arbitrarily well by networks of neurons**
- **For better approximation more neurons are needed**
 - Large networks have a lot of tunable parameters which must be learned from fixed amount of training data
 - May cause overfitting (no generalization takes place)
- **The VC dimension of the network determines it's generalization**
 - Tradeoff between approximation/generalization



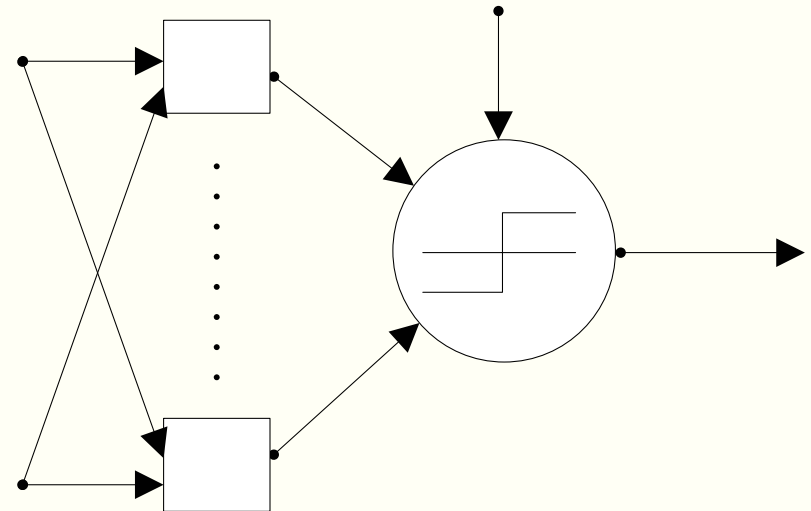
Some More Definitions

- φ -seperable if there exists

- Weights
- Functions

that can separate points according to Y

- Solution weight vector is the weights that allow φ -seperable given the functions



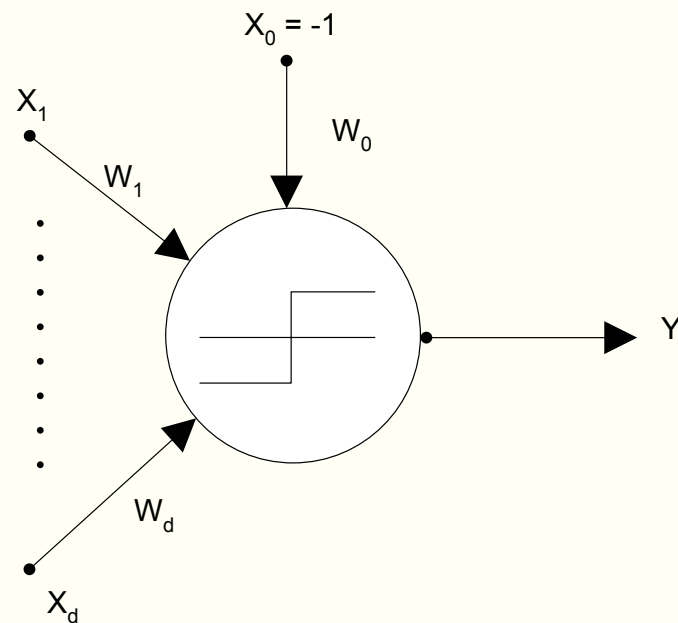


Single Neuron System

- The VC dimension of the class of decision functions computable by a φ -threshold element where $\varphi = (\varphi_1, \dots, \varphi_k)$ is given by $VC=k$

- VC of this neural network is $(d+1)$

- $\varphi_0 = -1$
- $\varphi_j = x_j \quad (1 \leq j \leq d)$





Single Neuron System Learning Algorithms

- **How to Compute Weights**
- **Assume there exists a solution weight vector**

- **Find the solution weight vector**
 - **Off-line techniques**
 - ◆ linear programming techniques
 - ◆ Not suitable for adaptive pattern recognition applications

 - **On-line techniques**
 - ◆ Learn on the “go”
 - ◆ Perceptron-based procedure
 - ◆ Gradient based approaches



Single Neuron System Perceptron

- Start with arbitrary weight vector
- At every step in time use one of the training data points to adjust the weight vector
 - Training data sequence is infinite by cyclically going through the sample points

- Update Rule

$$w(t+1) = \begin{cases} w(t), & \text{if correct output} \\ w(t) + \rho\phi(X(t)), & \text{if incorrect output} \end{cases}$$

- Whenever there is a misclassification, weight vector updates are in the direction of the positive half-space of the vector



Disadvantages of Perceptron

- **Has problems when the sample is not separable**
- **Not easily extended to general network architectures**
 - **What classifications should the internal neurons have**
- **Use gradient based approaches**
 - **Least mean-square seeks to minimize the squared error**
 - **Updates the weight vector in the direction of the gradient**



General Acyclic Networks

■ Threshold Neurons

- $c'K \log K \leq VC \leq cK \log K$
 - ◆ c, c' : positive constants
 - ◆ K : # of programmable parameters
- VC grows faster by addition nodes to existing layers (width) than by adding layers (depth)

■ Sigmoid Neurons

- Only lower bound on VC holds
- VC may be infinite



General Acyclic Networks

Learning Algorithms

- **Hard to find algorithms for learning from examples with provable performance**
 - Intractable for some architectures
- **Usually use gradient descent ideas**
 - Gradients are recursively generated starting from the output unit and working backward one layer at a time
 - **Problems:**
 - ◆ Inconsistent
 - ◆ Gets stuck in local minima
 - ◆ Convergence is slow
 - **Work because of good choice of**
 - ◆ Network architectures
 - ◆ Good initial conditions



More Neural Network Info

- **Because of difficulty of finding VC for general network structures neural networks cannot guarantee good performance**
 - Under some malicious distributions, other classifiers such as k-nearest neighbour are superior
- **Most Neural Networks minimize the mean-square error instead of the misclassification rate**
 - In some problems get surprisingly good performance