Preventing Overfitting

Problem:

- We don’t want to these algorithms to fit to ``noise’’
- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result is in poor accuracy for unseen samples

Avoid Overfitting in Classification

- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Reduced-error pruning :

breaks the samples into a training set and a test set. The tree is induced completely on the training set.

Working backwards from the bottom of the tree, the subtree starting at each nonterminal node is examined.

If the error rate on the test cases improves by pruning it, the subtree is removed. The process continues until no improvement can be made by pruning a subtree, The error rate of the final tree on the test cases is used as an estimate of the true error rate.

Decision Tree Pruning:

```
physician fee freeze = n:
|   adoption of the budget resolution = y: democrat (151.0)
|   adoption of the budget resolution = u: democrat (1.0)
|   adoption of the budget resolution = n:
|   |   education spending = n: democrat (6.0)
|   |   education spending = y: democrat (9.0)
|   |   education spending = u: republican (1.0)
physician fee freeze = y:
|   synfuels corporation cutback = n: republican (97.0/3.0)
|   synfuels corporation cutback = u: republican (4.0)
|   synfuels corporation cutback = y:
|   |   duty free exports = y: democrat (2.0)
|   |   duty free exports = u: republican (1.0)
|   |   duty free exports = n:
|   |   |   education spending = n: democrat (5.0/2.0)
|   |   |   education spending = y: republican (13.0/2.0)
|   |   |   education spending = u: democrat (1.0)
physician fee freeze = u:
|   water project cost sharing = n: democrat (0.0)
|   water project cost sharing = y: democrat (4.0)
|   water project cost sharing = u:
|   |   mx missile = n: republican (0.0)
|   |   mx missile = y: democrat (3.0/1.0)
|   |   mx missile = u: republican (2.0)
```

```
Simplified Decision Tree:

physician fee freeze = n: democrat (168.0/2.6)
physician fee freeze = y: republican (123.0/13.9)
physician fee freeze = u:
|   mx missile = n: democrat (3.0/1.1)
|   mx missile = y: democrat (4.0/2.2)
|   mx missile = u: republican (2.0/1.0)
```

```
Evaluation on training data (300 items):

 Before Pruning          After Pruning
----------------   --------------------------
Size      Errors   Size      Errors   Estimate

  25    8( 2.7%)      7   13( 4.3%)    ( 6.9%)   <
```

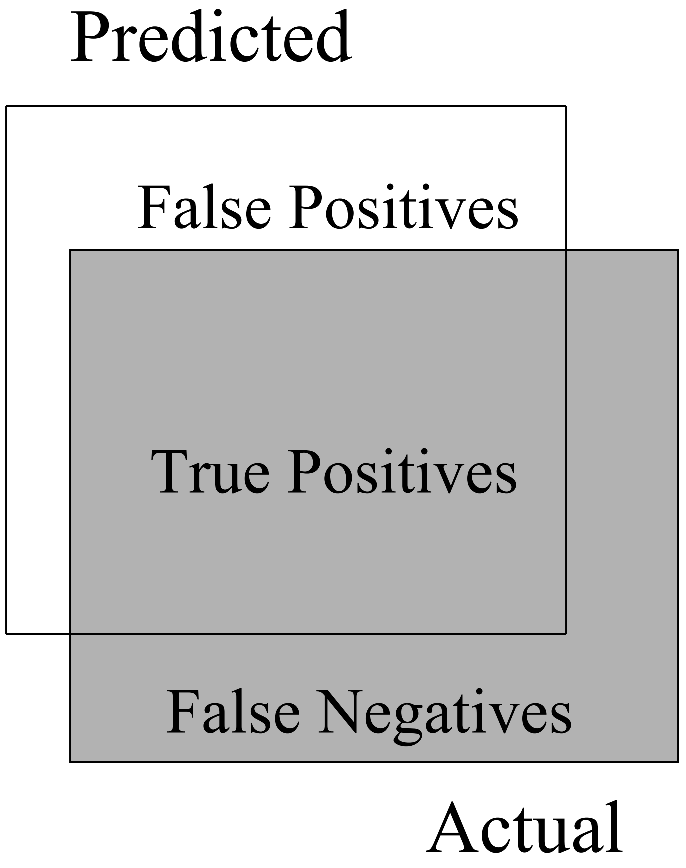
Evaluation of Classification Systems

Training Set: examples with class values for learning.

Test Set: examples with class values for evaluating.

Evaluation: Hypotheses are used to infer classification of examples in the test set; inferred classification is compared to known classification.

Accuracy: percentage of examples in the test set that are classified correctly.



Model Evaluation

- Analytic goal: achieve understanding
  - Exploratory evaluation : understand a novel area of study
  - Experimental evaluation : support or refute some models
- Engineering goal : solve a practical problem
- Estimator of classifiers : accuracy
  - Accuracy : how well does a model classify
  - Higher accuracy does not necessarily imply better performance on target task

Confusion Metrics

Actual Class

		+	-
Predicted class	Y	A: True +	B : False +
	N	C : False -	D : True -

Entries are counts of correct classifications and counts of errors

Other evaluation metrics

- True positive rate (TP) = A/(A+C)= 1- false negative rate
- False positive rate (FP)= B/(B+D) = 1- true negative rate
- Sensitivity = true positive rate
- Specificity = true negative rate
- Positive predictive value = A/(A+B)
- Recall = A/(A+C) = true positive rate = sensitivity
- Precision = A/(A+B) = PPV

Probabilistic Interpretation of CM

P (+) :
P (-)

prior probabilities approximated by class frequencies

Class Distribution

Defined for a particular training set

Posterior probabilities

P(+ | Y)
P(- | N)

likelihoods

P(Y |+)
P(Y |-)

approximated using error frequencies

Confusion matrix

Defined for a particular classifier

More Than Accuracy

- Cost and Benefits
  - Medical diagnosis : cost of falsely indicating “cancer” is different from cost of missing a true cancer case
  - Fraud detection : cost of falsely challenging customer is different from cost of leaving fraud undetected
  - Customer segmentation : Benefit of not contacting a non-buyer is different from benefit of contacting a buyer

Model Evaluation within Context

- Must take costs and distributions into account
- Calculate expected profit:

profit = P(+)*(TP*B(Y, +) + (1-TP)*C(N, +))
+ P(-)*((1-FP)*B(N, -) + FP*C(Y, -))

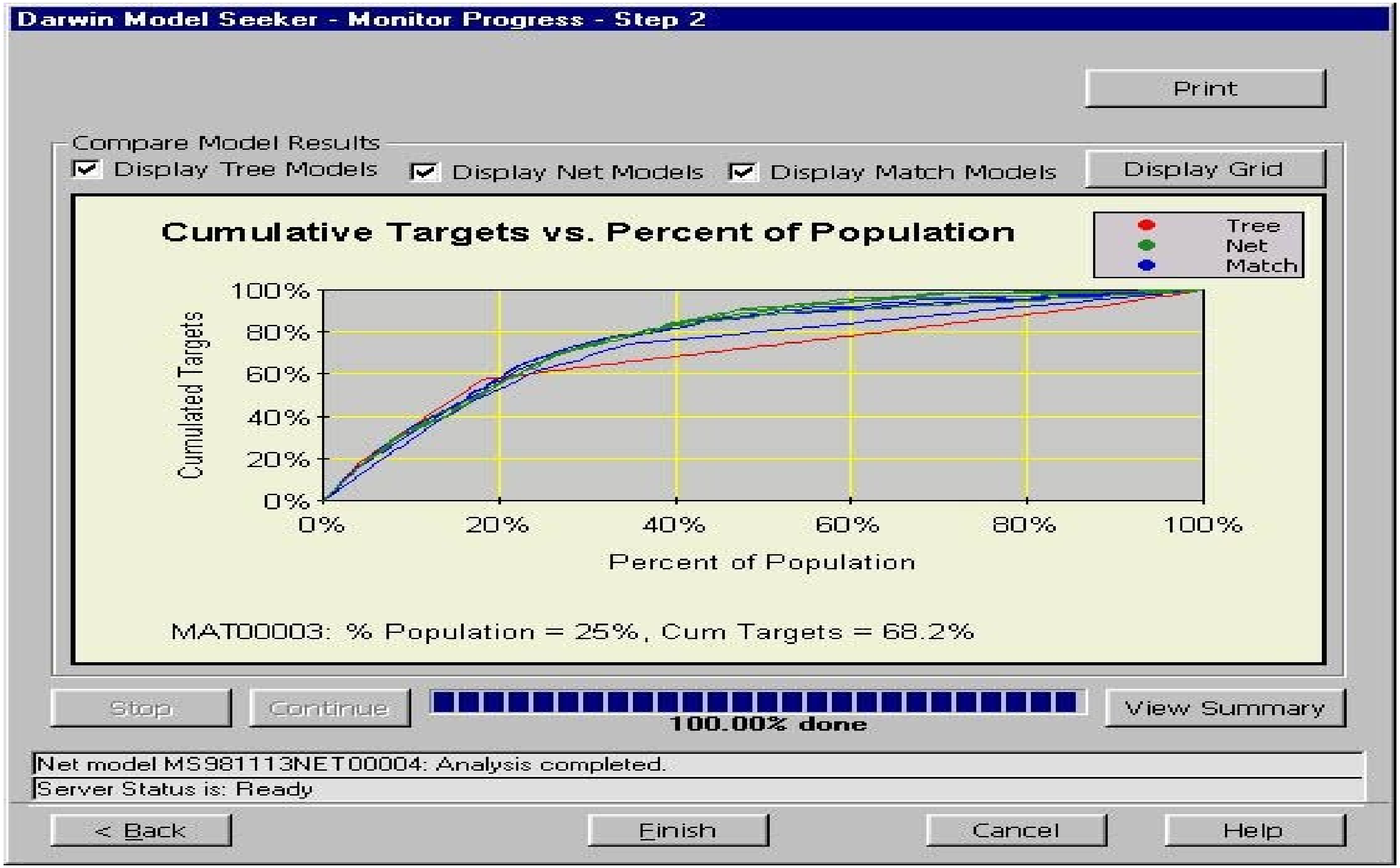
Benefits of correct classification

costs of incorrect classification

- Choose the classifier that maximises profit

Lift & Cumulative Response Curves

Lift = P(+ | Y)/P(+) : How much better with model than without



Darwin Model Seeker - Monitor Progress - Step 2

Compare Model Results

☑ Display Tree Models ☑ Display Net Models ☑ Display Match Models

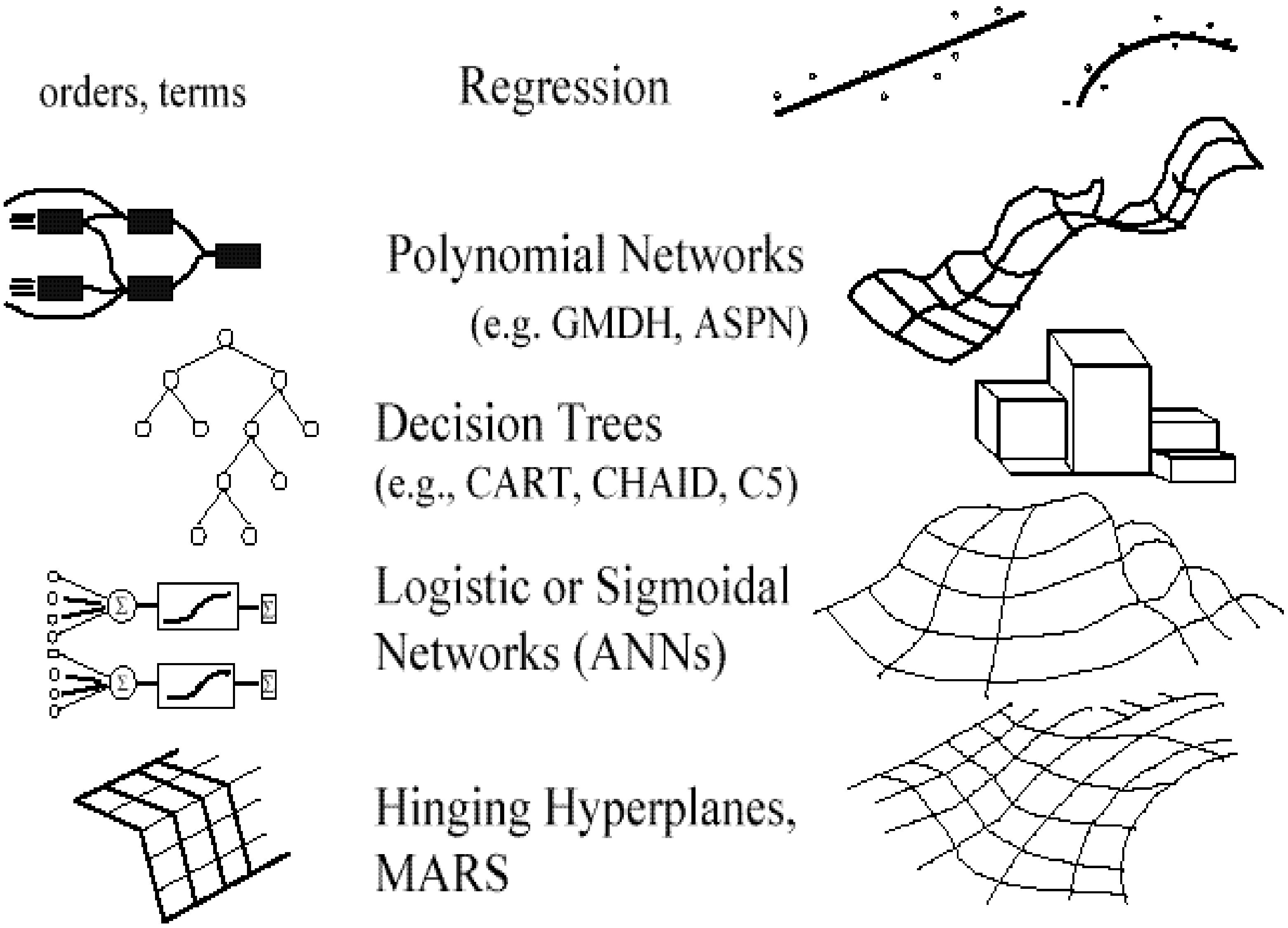
MAT00003: % Population = 25%, Cum Targets = 68.2%

100.00% done

Net model MS981113NET00004: Analysis completed.

Server Status is: Ready

Parametric Models : Parametrically Summarise Data



Regression

Polynomial Networks
(e.g. GMDH, ASPN)

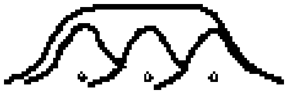
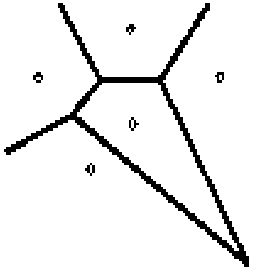
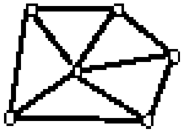
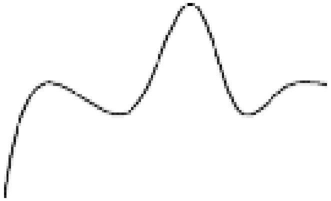
Decision Trees
(e.g., CART, CHAID, C5)

Logistic or Sigmoidal
Networks (ANNs)

Hinging Hyperplanes,
MARS

Contributory Models :

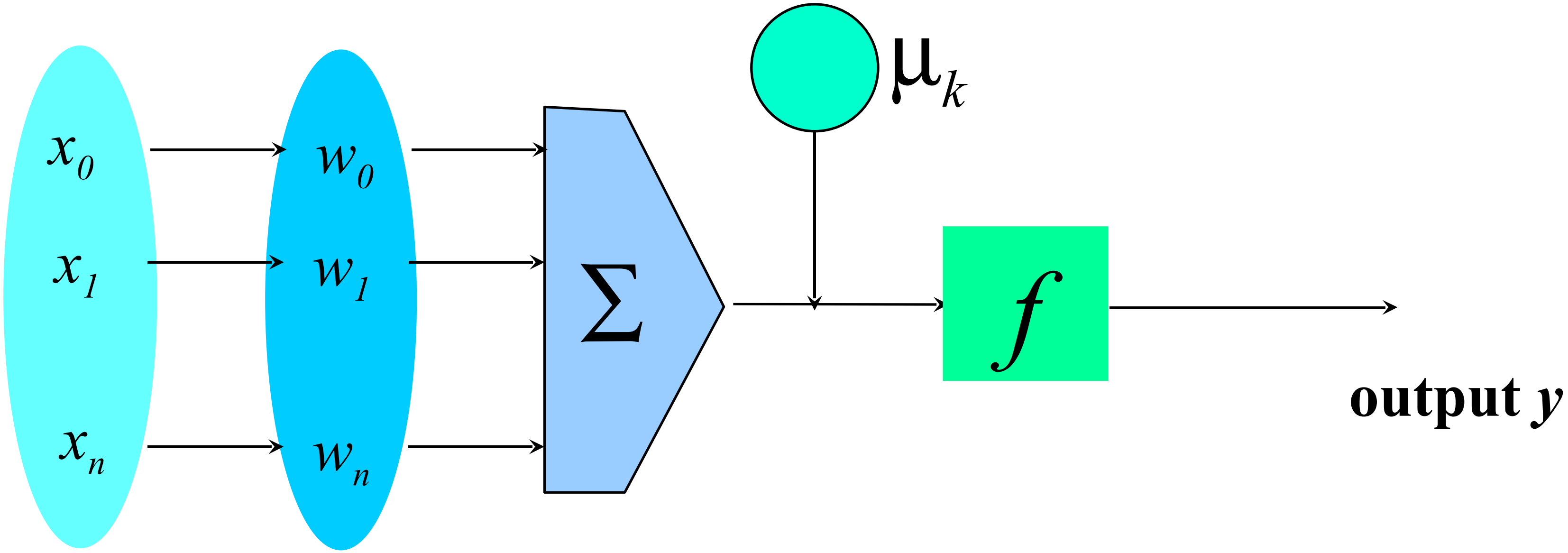
retain training data points; each potentially affects the estimation at new point

shape, spread	Kernels	
k, distance metric	k-Nearest Neighbor	
Goal, iterations	Delaunay Planes	
Spread, index	Projection Pursuit Regression	

Neural Networks

- Advantages
  - prediction accuracy is generally high
  - robust, works when training examples contain errors
  - output may be discrete, real-valued, or a vector of several discrete or real-valued attributes
  - fast evaluation of the learned target function
- Criticism
  - long training time
  - difficult to understand the learned function (weights)
  - not easy to incorporate domain knowledge

A Neuron



output *y*

Input vector *x*

weight vector *w*

weighted sum

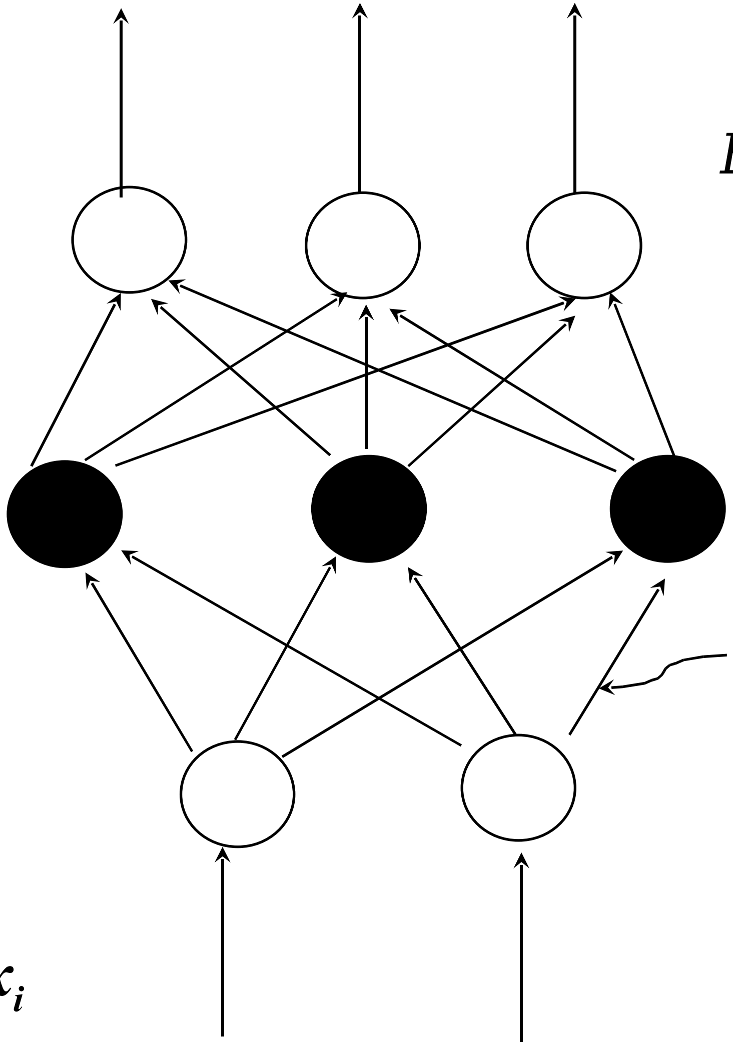
Activation function

- The *n*-dimensional input vector *x* is mapped into variable *y* by means of the scalar product and a nonlinear function mapping

Network Training

- The ultimate objective of training
  - obtain a set of weights that makes almost all the tuples in the training data classified correctly
- Steps
  - Initialize weights with random values
  - Feed the input tuples into the network one by one
  - For each unit
    - Compute the net input to the unit as a linear combination of all the inputs to the unit
    - Compute the output value using the activation function
    - Compute the error
    - Update the weights and the bias

Multi-Layer Perceptron



Output vector

Output nodes

Hidden nodes

Input nodes

Input vector: *x_i*

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

$$\theta_j = \theta_j + (l)\,Err_j$$

$$w_{ij} = w_{ij} + (l)Err_j O_i$$

$$Err_j = O_j(1-O_j)(T_j - O_j)$$

w_{ij}

$$O_j = \frac{1}{1+e^{-I_j}}$$

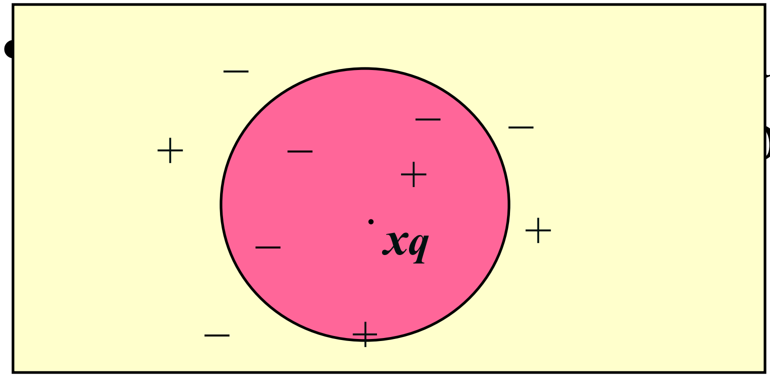
$$I_j = \sum_i w_{ij} O_i + \theta_j$$

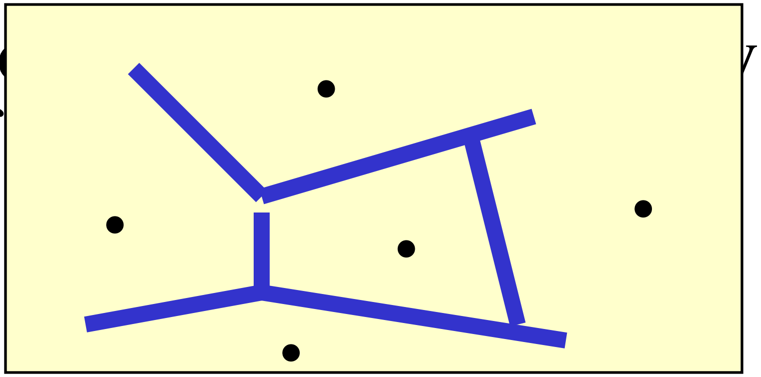
Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing (“lazy evaluation”) until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

The *k*-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space.
- The nearest neighbor are defined in terms of Euclidean distance.
- The target function could be discrete- or real-valued.
- For discrete-valued, the *k*-NN returns the most common value among the k training examples nearest to x_q.





Discussion on the *k*-NN Algorithm

- The k-NN algorithm for continuous-valued target functions
  - Calculate the mean values of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the k neighbors according to their distance to the query point x_q
    - giving greater weight to closer neighbors
  - Similarly, for real-valued target functions

$$w\text{º}\frac{1}{d(x_q, x_i)^2}$$

- Robust to noisy data by averaging k-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes.
  - To overcome it, axes stretch or elimination of the least relevant attributes.