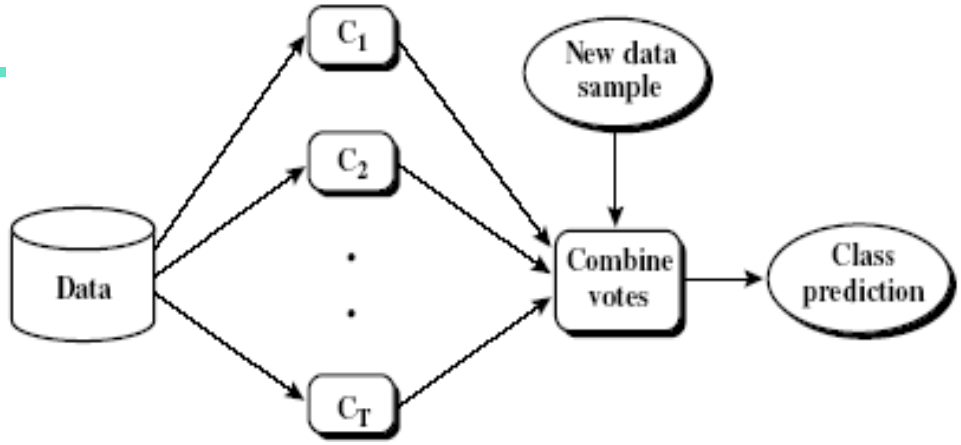
CS699
Lecture 7
Other Classifiers

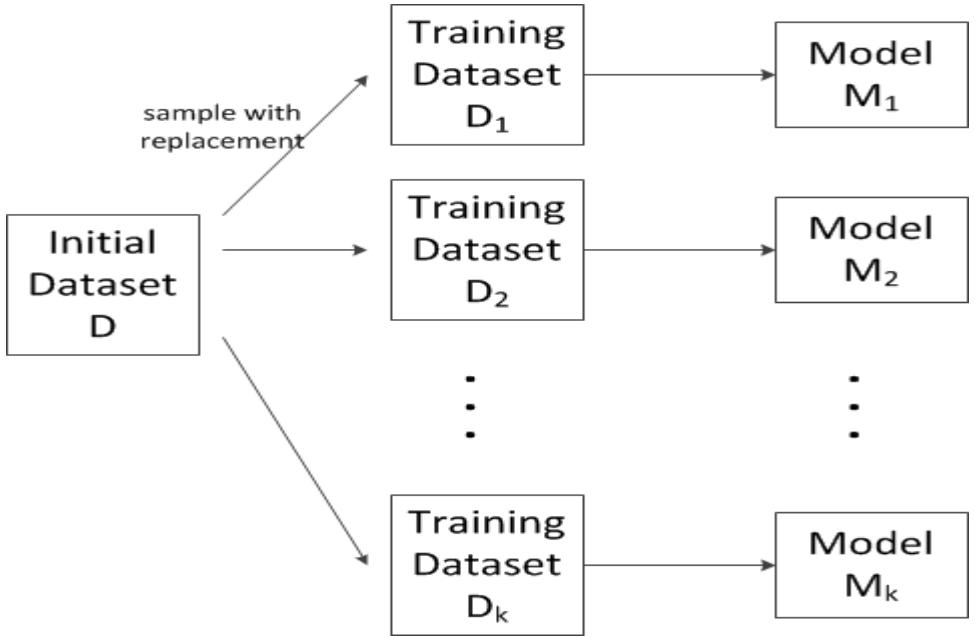
Ensemble Methods: Increasing the Accuracy



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Bagging: Boostrap Aggregating

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap).
  - A classifier model M_i is learned for each training set D_i (using the same classifier method,. e.g., decision tree).

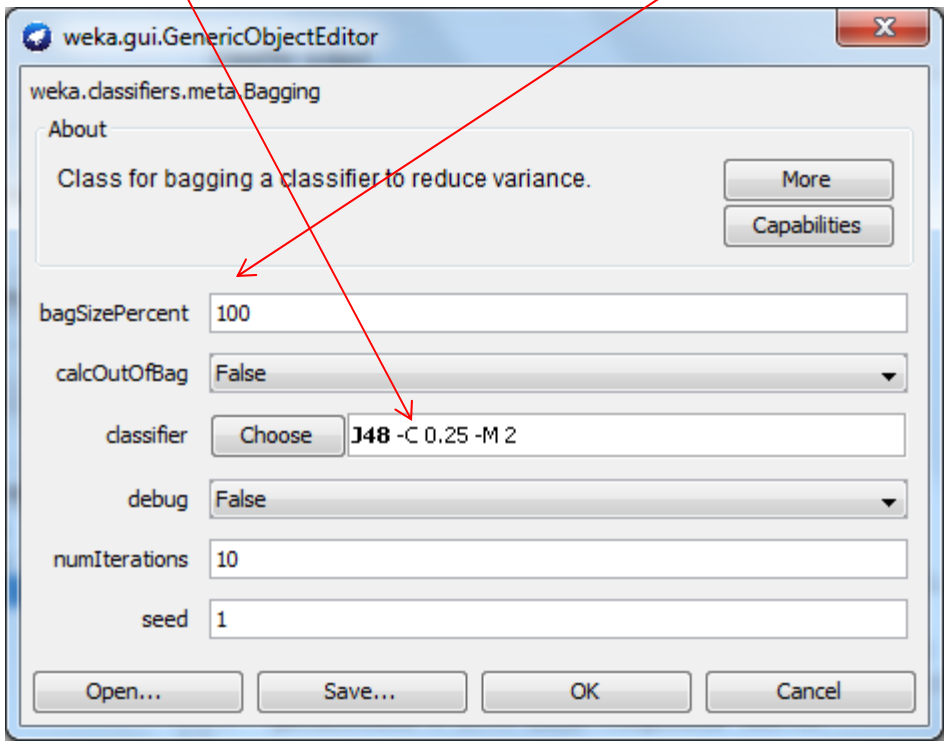


Bagging: Boostrap Aggregating

- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Bagging on Weka

- Under Meta classifiers
- Select dataset size per each iteration (bagSizePercent)
- Select base classifier



Boosting

- Builds and combines multiple classifier models.

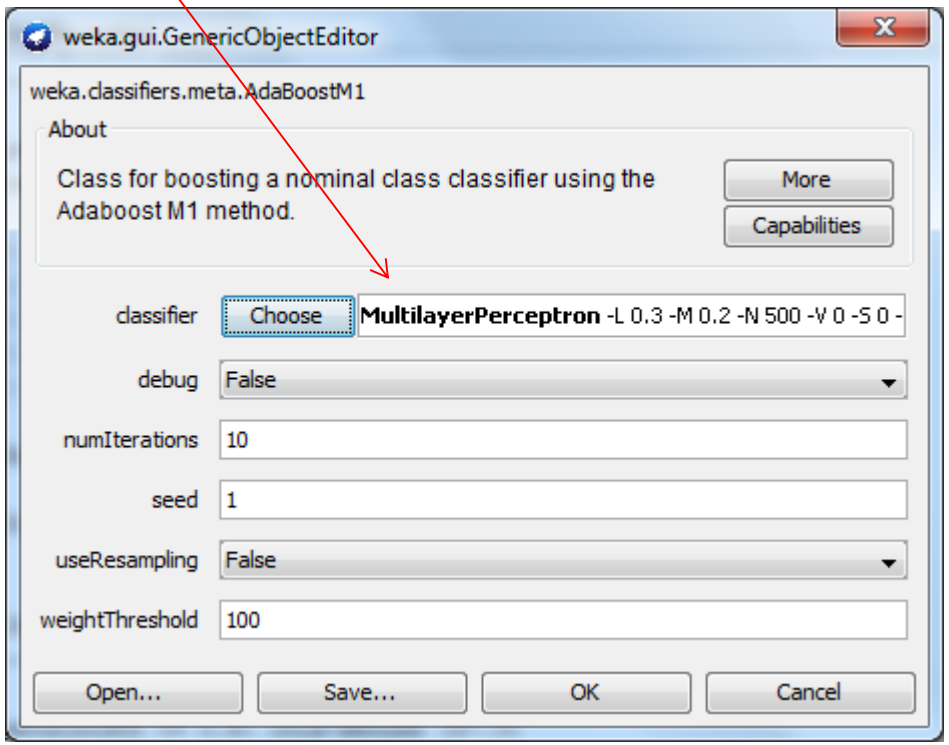
- Similar to bagging:
  - Sampling with replacement is used
  - The same classifier method is used.
  - Voting is used.

Boosting

- Different from bagging:
  - Models are built sequentially and a model is influenced by previously built models.
  - A new model “pays more attention” to tuples that were misclassified by previous models.
  - Each training tuple has a weight.
  - If a tuple is misclassified, the weight of the tuple is increased.
  - The probability that a tuple is sampled is based on the weight of the tuple (i.e., if a tuple was misclassified in a previous model, it is more likely to be sampled).
  - When combining models, a model with a higher performance is given a higher weight.

Boosting on Weka

- AdaBoostM1 under Meta classifiers
- Select base classifier



Class Imbalance Problem

- The main class of interest (the positive class) is represented by a very small fraction of dataset.
- Example: 10% positive tuples and 90% negative tuples
- As discussed earlier, an accuracy is not a good measure.
- Other measures, such as sensitivity, specificity, ROC curves can be used to better understand the performance of class-unbalanced data.
- We will discuss two approaches to improve classification accuracy on class-unbalanced data.

Class Imbalance Problem

- Oversampling and undersampling
- Both approaches manipulate the data.

- Example: 100 positives and 1000 negatives
- Oversampling: sample positive tuples, with replacement, until the number of positive tuples becomes 1000.
- Undersampling: randomly remove tuples from negative tuples until the number of negative tuples becomes 100.

Lazy Learner

- Does not build a model from the training dataset
- When a new tuple comes in, it uses the stored training dataset and classifies the new tuple.
- Also called instance based learner.
- K-nearest-neighbor classifier (KNN):
  - The class label of a new tuple is determined by the class labels of k nearest neighbor tuples, by majority voting.

Lazy Learner

- KNN Example, with k = 3, Manhattan distance

  Training dataset

OID	X	Y	CLASS
1	2	2	Y
2	3	4	N
3	4	2	Y
4	5	5	Y
5	6	3	N

Classify an object <X=3,Y=2>

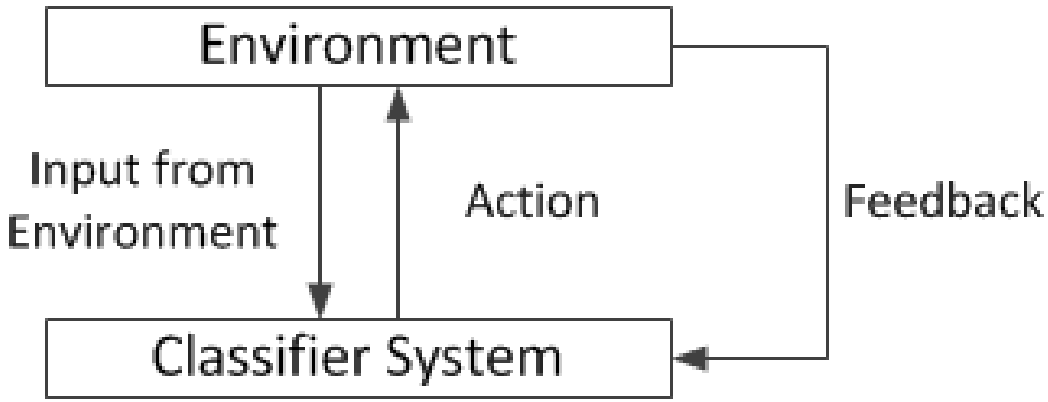
Objects 1, 2, and 3 are 3 nearest neighbors.

Y is the majority class among these 3 objects.

So, the new object is classified as Y.

Learning Classifier System (LCS)

- Machine learning technique which combines
  - Reinforcement learning
  - Evolutionary computing
  - Other heuristics to produce adaptive system



Learning Classifier System (LCS)

- Reinforcement learning
  - Learning through trial and error via the reception of a numerical reward
  - Maximize future reward

Learning Classifier System (LCS)

- Evolutionary computing
  - Search algorithm based on the mechanisms of Natural selection and genetics
  - Apply Darwin's principle of the survival of the fittest among the computational structure
  - Apply stochastic process of gene mutation, recombination, etc.

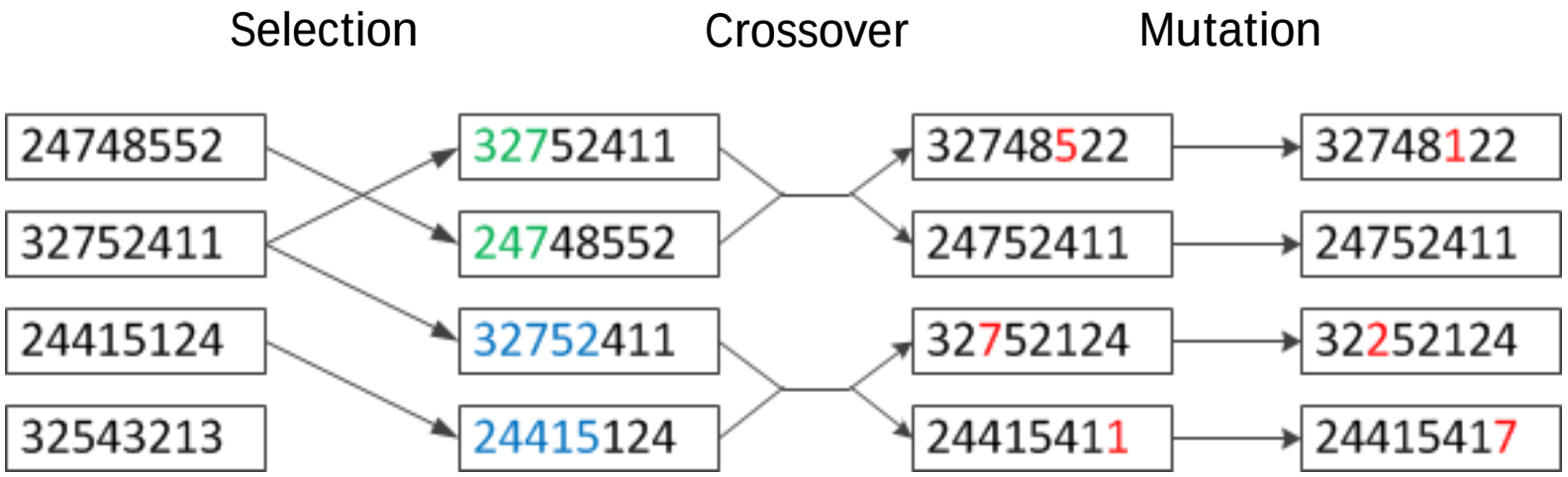
Genetic Algorithm

- A local search algorithm (searching for a solution to an optimization problem, for example)
- Based on evolutionary biology

- Next generation individuals are generated from a population by
  - Selection
  - Crossover
  - Mutation

Genetic Algorithm

- An example (with digit strings)



Parents selected based on fitness function

Parents are crossed over to generate offsprings

A digit is randomly selected and randomly mutated

Note: Selection, crossover, and mutation can be done in different ways.

XCS

- A learning classifier system
- A model is represented as a set of rules
- A rule has an antecedent and a consequent
- A rule is usually represented as a bit string
- Set of rules (classifier model) is generated based on a genetic algorithm

XCS

- Example of a rule representation

  (From the Buys_computer dataset)

- First, attribute values are encoded into binary values
  - Age has three values so three bits are used:

    100: <= 30

    010: 31..40

    001: > 40

XCS

- Income has three values so three bits are used:
  - 100: low
  - 010: medium
  - 001: high
- Student has two values so two bits are used:
  - 10: no
  - 01: yes
- Credit_rating has two values so two bits are used:
  - 10: fair
  - 01: excellent

XCS

- Buys_computer (class label) has two values so two bits are used:

  10: no

  01: yes

- Then, the following rule

  If age >40 AND Income = high AND Student = no

  AND Credit_rating = excellent,

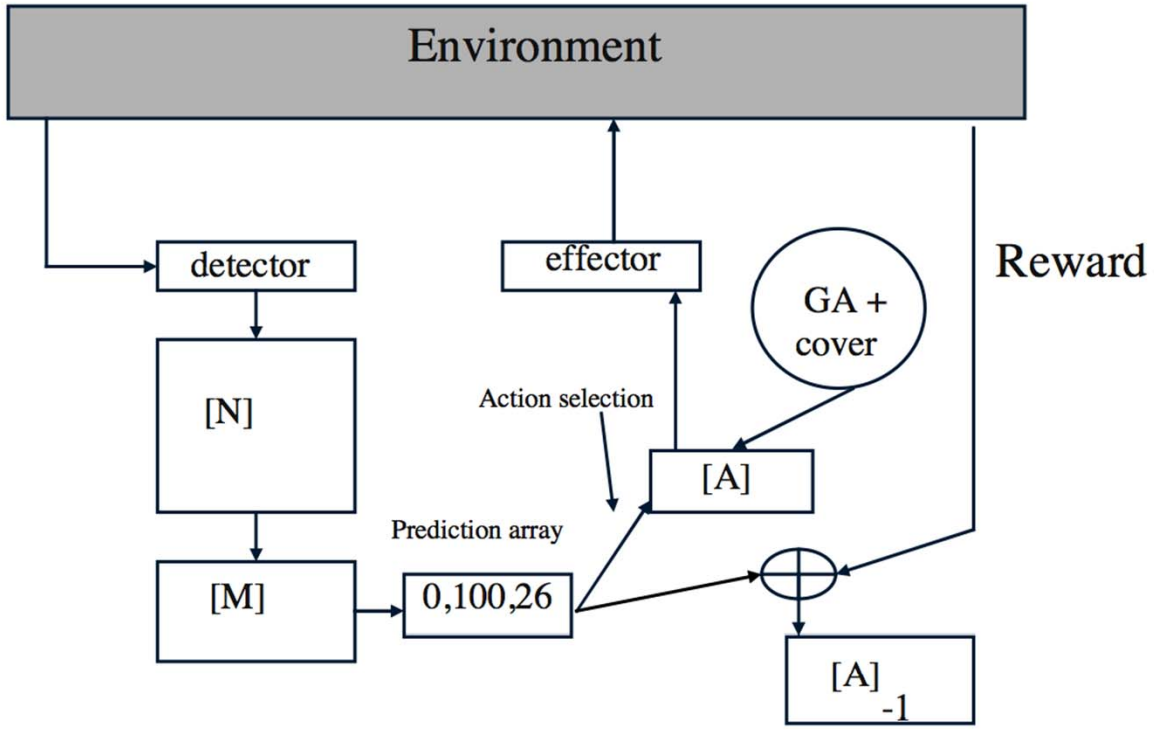
  Then buys_computer = no

  Is represented as: 00100110**01** 10

- NOTE: There are other representations.

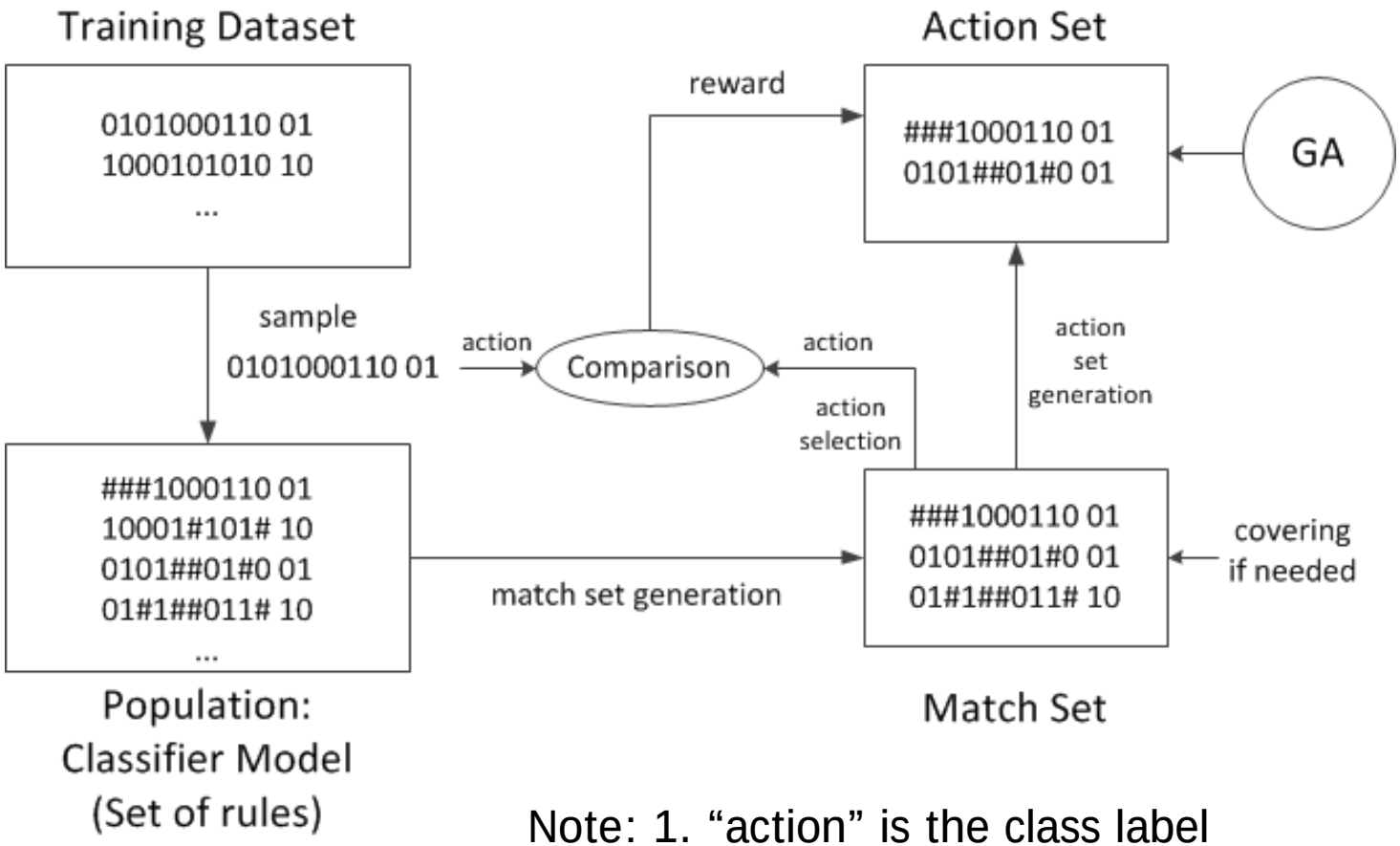
XCS

- Schematic (general)



XCS

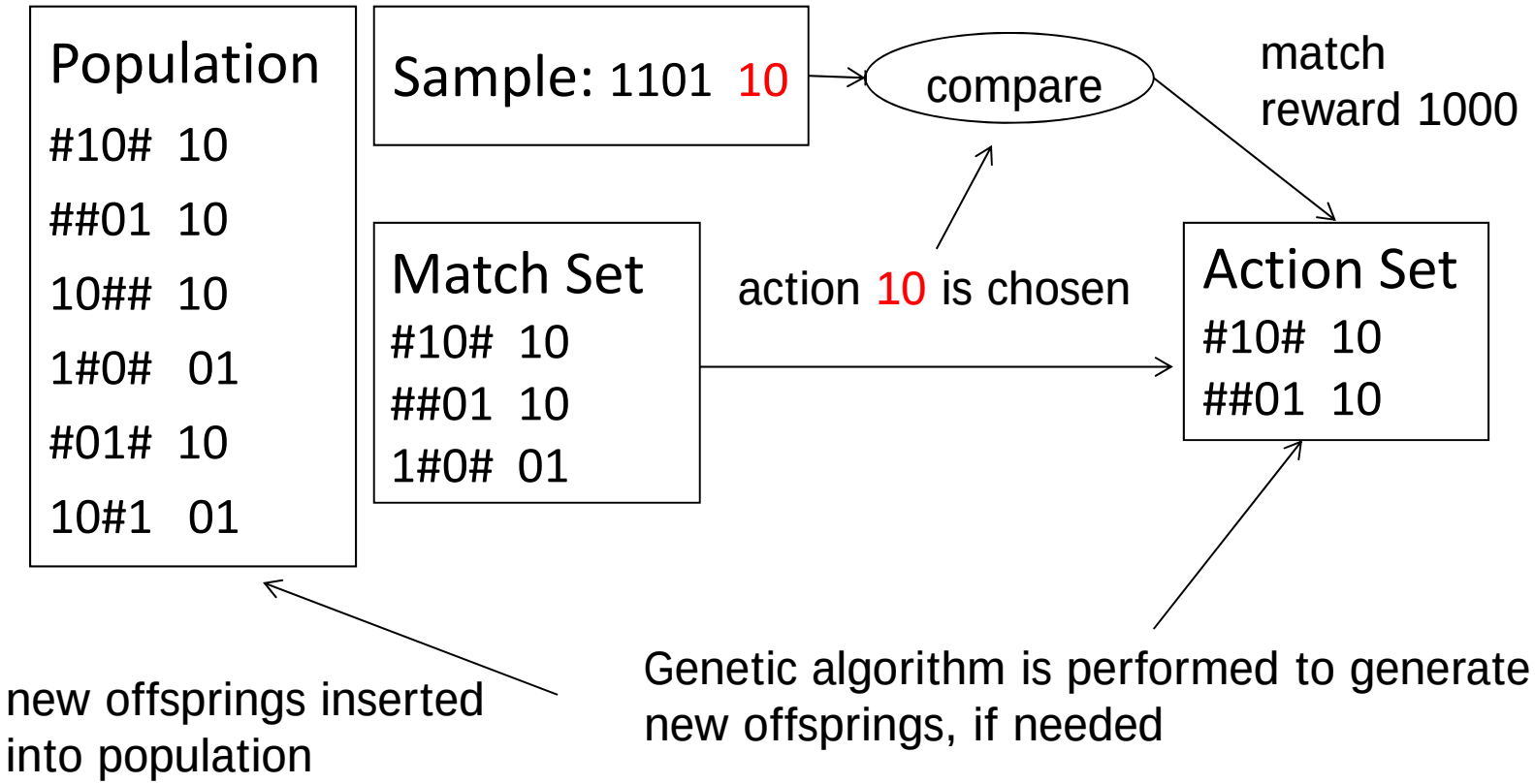
- Simplified Schematic (as a classifier)



Note: 1. "action" is the class label
2. # matches either 0 or 1

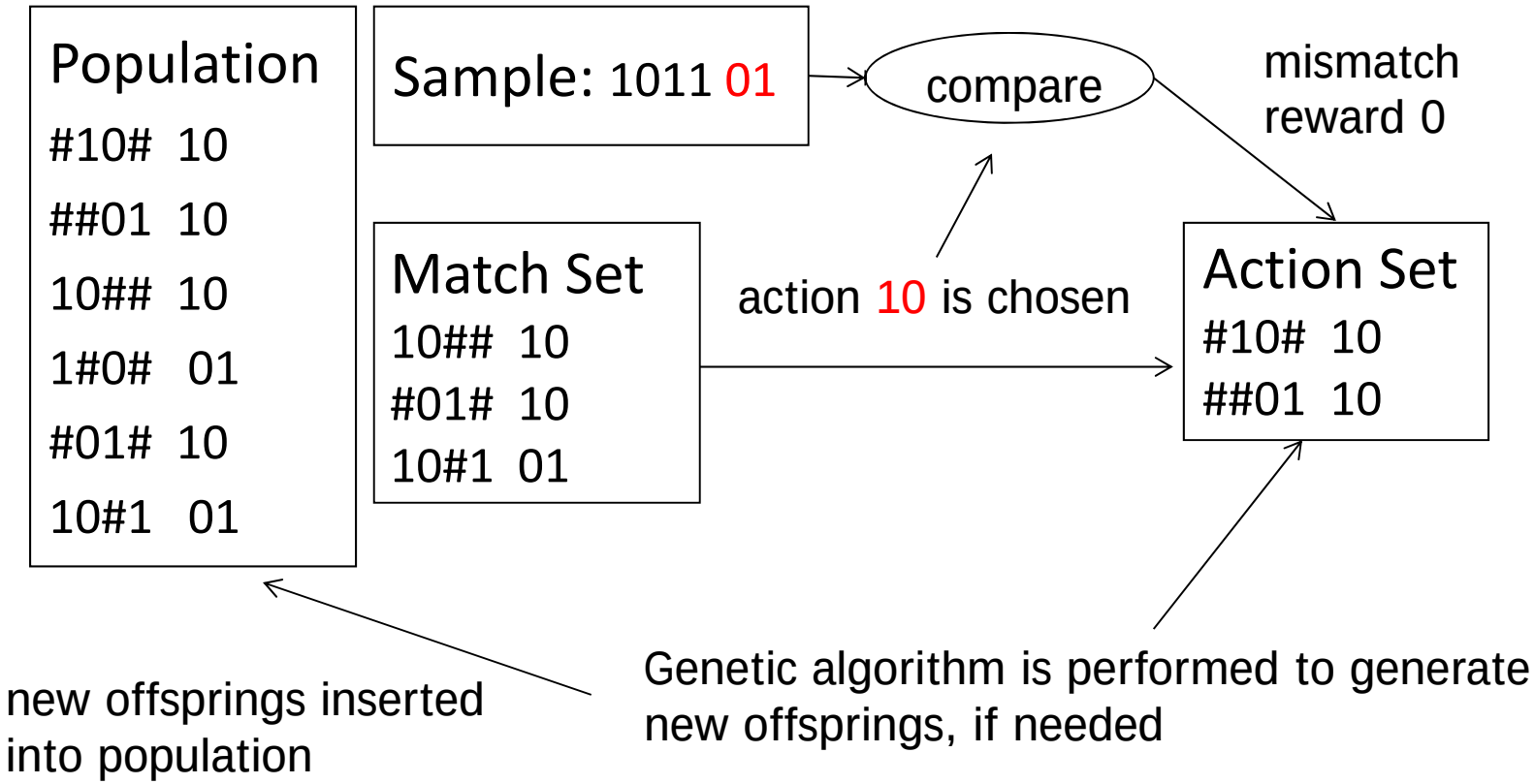
XCS

- Example


Population
#10# 10
##01 10
10## 10
1#0# 01
#01# 10
10#1 01
Sample: 1101 10
compare
match
reward 1000
Match Set
#10# 10
##01 10
1#0# 01
action 10 is chosen
Action Set
#10# 10
##01 10
new offsprings inserted
into population
Genetic algorithm is performed to generate
new offsprings, if needed

XCS

- Example


Population
#10# 10
##01 10
10## 10
1#0# 01
#01# 10
10#1 01
Sample: 1011 01
compare
mismatch
reward 0
Match Set
10## 10
#01# 10
10#1 01
action 10 is chosen
Action Set
#10# 10
##01 10
new offsprings inserted
into population
Genetic algorithm is performed to generate
new offsprings, if needed

XCS

- Experimental result (preliminary)

data Set	Naïve-Bayse	J48	Logistic	ANN	SVM	XCS
adult	83.48	86.22	85.18	83.27	75.90	81.78
diabetes	76.30	73.83	77.21	75.39	65.10	79.94
breast cancer	71.68	75.53	68.88	64.69	73.08	87.58
breast cancer wisconsin	95.99	94.56	96.57	95.28	95.71	99.57
cmc	49.29	53.22	51.60	54.51	56.21	56.62
ionosphere	82.62	91.45	88.89	91.17	93.45	92.00
iris	94.00	94.00	93.33	97.33	96.67	98.00
liver disorder	55.36	68.70	68.12	71.59	59.42	83.19
lymphography	37.84	36.49	28.38	39.86	43.92	52.03
lung cancer	78.13	78.13	75.00	65.63	71.88	80.48
primary tumor	54.98	47.91	48.23	45.02	44.05	53.05
voting	75.53	77.34	75.53	78.25	81.27	93.07
credit approval(crx)	77.68	86.09	85.22	82.75	54.78	88.26
balance-scale	23.84	32.80	26.72	21.60	14.08	78.88
heartDisease-cleveland	55.45	55.45	57.76	55.78	49.51	66.67
Average	67.48	70.11	68.44	68.14	65.00	79.41

XCS

- LCS applications
  - Data mining (besides classification)
    - Extract compact descriptions of interesting phenomenon usually described by multidimensional data
  - Optimization
    - Shape optimization
    - Predict next note in different type of music
    - The profit optimization of a batch chemical reaction

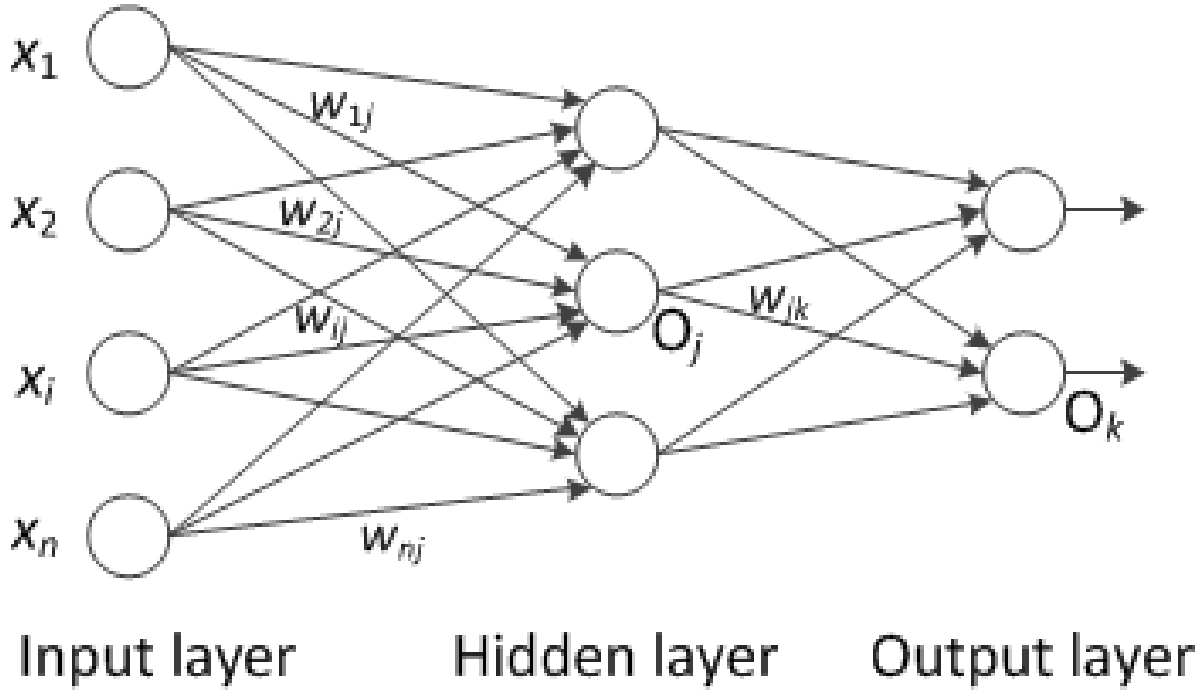
XCS

- LCS applications
  - Modeling
    - Some categorization process
    - Used in Computational Economics
  - Control
    - Control of traffic signal
    - Control of robot arms
    - Control of underwater vehicle
  - Many others

Classification by Backpropagation

- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples
- Also referred to as **connectionist learning** due to the connections between units

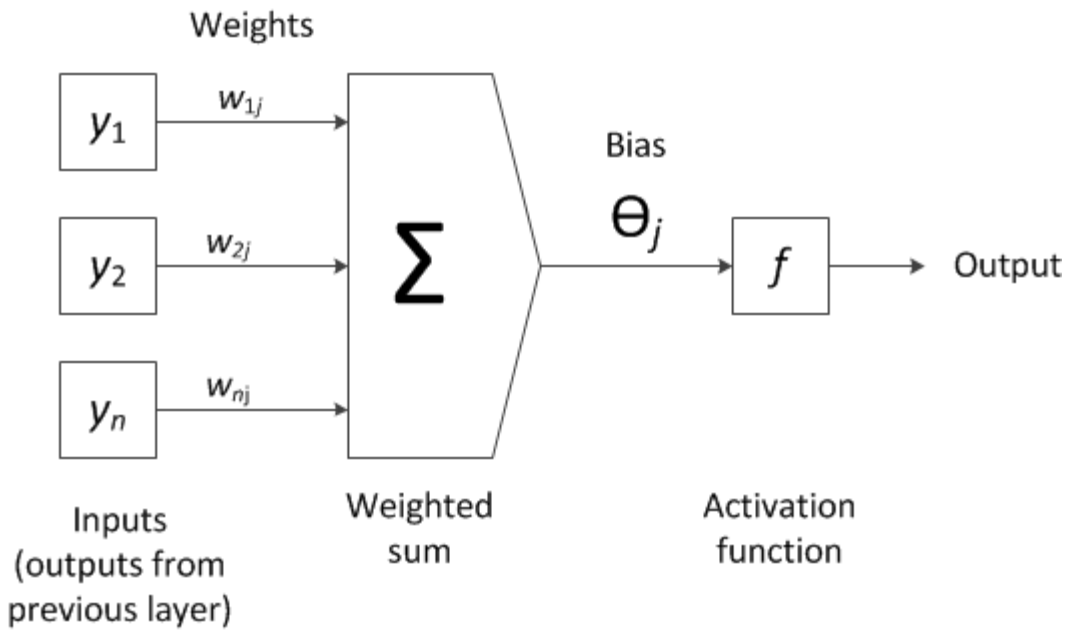
Multilayer Feed-forward Neural Network



How A Multi-Layer Neural Network Works

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- The network is **feed-forward**: None of the weights cycles back to an input unit or to an output unit of a previous layer

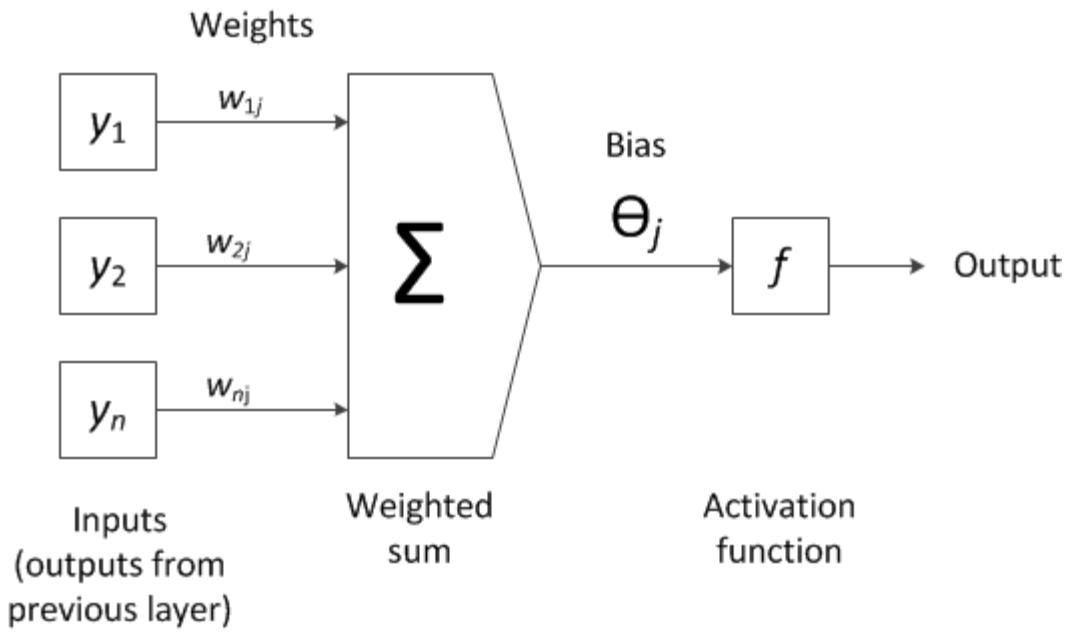
Neuron: A Hidden/Output Layer Unit



Hidden or output layer unit j

- An *n*-dimensional input vector **x** is mapped into variable y by means of the scalar product and a nonlinear function mapping
- The inputs to unit are outputs from the previous layer. They are multiplied by their corresponding weights to form a weighted sum, which is added to the bias associated with unit. Then a nonlinear activation function is applied to it.

Neuron: A Hidden/Output Layer Unit



Hidden or output layer unit j

- Net input to unit *j* is: $I_j = \sum_i w_{ij} O_i + \theta_j$
- Output of unit *j* is: $O_j = \frac{1}{1+e^{-I_j}}$

Here, activation function is *logistic*, or *sigmoid* function.

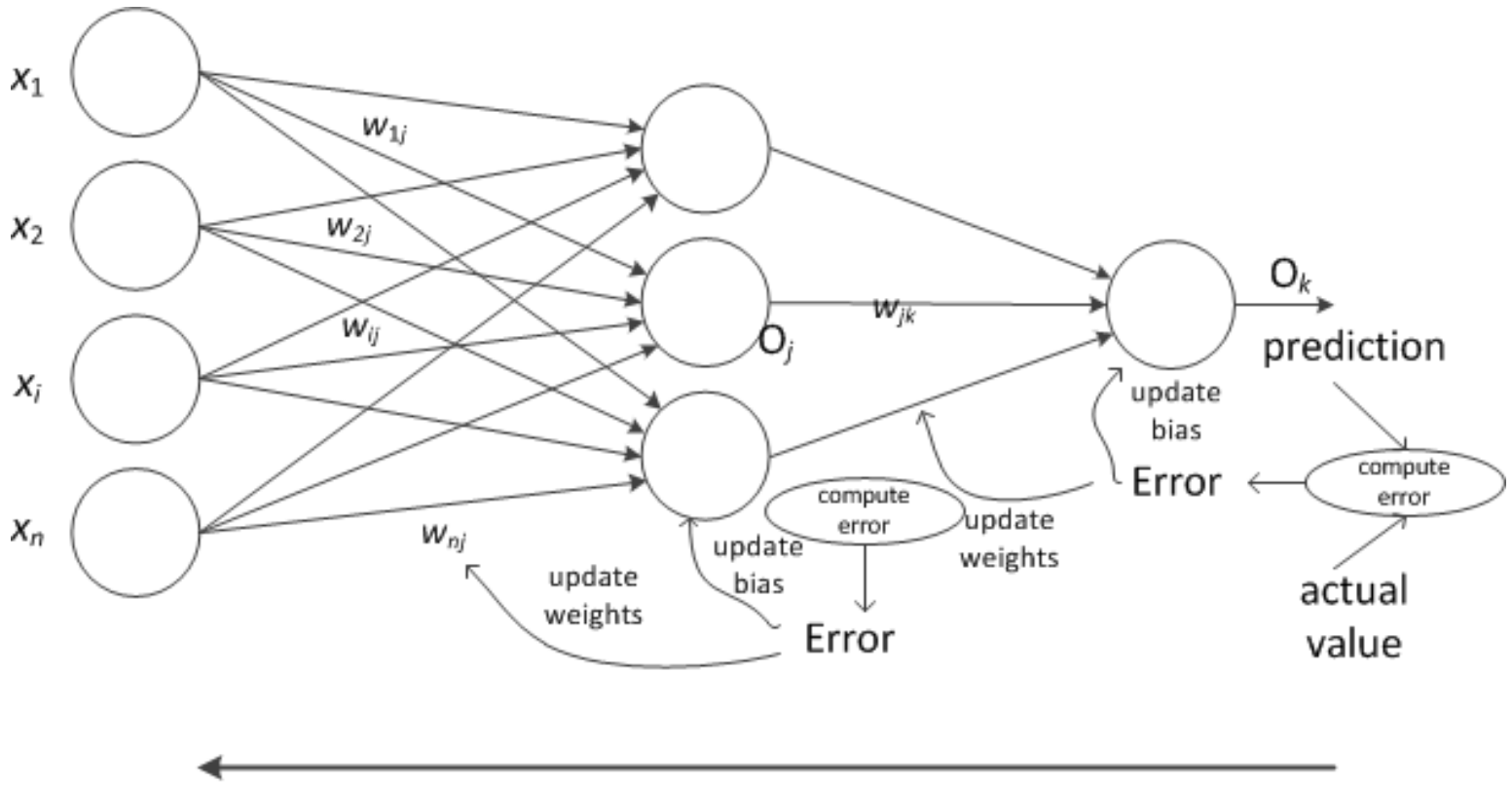
Defining a Network Topology

- Decide the **network topology:** Specify # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Numeric attribute: Normalize the input values for each attribute measured in the training tuples to [0.0—1.0]
- Categorical attribute: One **input** unit per domain value, each initialized to 0
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a *different network topology* or a *different set of initial weights*

Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the "**backwards**" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "**backpropagation**"
- Steps
  - Initialize weights to small random numbers, associated with biases
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

Backpropagation



error is propagated backward by updating biases and weights

Neural Network as a Classifier

- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or “structure.”
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of “hidden units” in the network
- Strength
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs *and outputs*
  - Successful on an array of real-world data, e.g., hand-written letters
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks

Other Classifiers

- Support Vector Machine
  - Relatively recent (1992)
  - Highly accurate
  - Especially for nonlinear boundary decisions
  - Less prone to overfitting
  - Slow
- Logistic regression (for classification)
- Random tree
- Random forest

References

- Han, J., Kamber, M., Pei, J., “Data mining: concepts and techniques,” 3rd Ed., Morgan Kaufmann, 2012
- http://www.cs.illinois.edu/~hanj/bk3/
- Class imbalance problem: Han, J., Kamber, M., Pei, J., “Data mining: concepts and techniques,” 3rd Ed., Morgan Kaufmann, 2012, pp.383-385.
- Genetic algorithm: S. Russell and P. Norvig, “Artificial intelligence a modern approach,” 3rd Ed., 2010, Prentice Hall, pp. 126 – 129.

References

- LCS and XCS
  - J.H. Holland, “Adaptation in natural and artificial systems,” 1975, University of Michigan Press.
  - S.W. Wilson, “Classifier fitness based on accuracy,” Evolutionary Computation, 3, 1995, pp. 149-175.
  - S.W. Wilson, “Generalization in the XCS Classifier System,” Genetic Programming 1998, Proc. 3rd Annual Conference.
  - M Butz and S.W. Wilson, "An Algorithmic Description of XCS," IlliGAL Report No. 2000017, May 2000, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign
- Neural Network: Han, J., Kamber, M., Pei, J., “Data mining: concepts and techniques,” 3rd Ed., Morgan Kaufmann, 2012, pp.398-408.