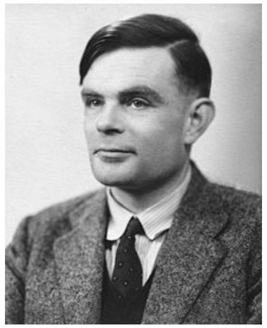
Lecture 9: Machine learning

Torgeir R. Hvidsten

Professor
Norwegian University of Life Sciences

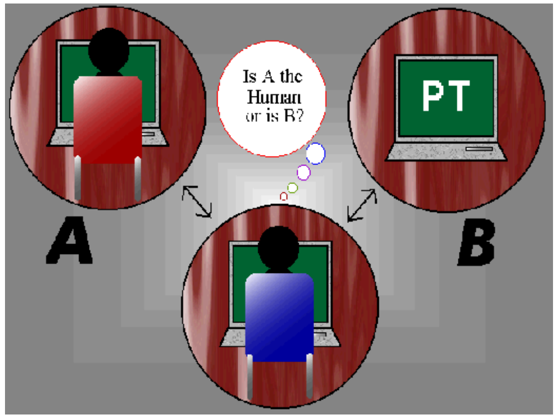
Guest lecturer
Umeå Plant Science Centre
Computational Life Science Cluster (CLiC)

Artificial intelligence: The Turing test


1912-1954

- Turing proposed that a computer program show intelligent behavior if is able to fool a human interrogator

- The Turing test: the computer is interrogated by a human via a teletype, and passes the test if the interrogator cannot tell if there is a computer or a human at the other end
  - natural language processing
  - knowledge representation
  - automated reasoning
  - <u>machine learning</u>



AI techniques

- **Logics**
- **Knowledge representation**
- **Search**
- **Machine learning**
- **Pattern recognition**
- **Automatic theorem proving**
- **Planning**
- **Machine vision**
- **Natural language processing**

"…making a machine behave in ways that would be called intelligent if a human were so behaving"

- John McCarthy, August 31, 1955

"The subfield of computer science concerned with the concepts and methods of symbolic inference by computer and symbolic knowledge representation for use in making inferences."

- The Free On-line Dictionary of Computing (September 27, 2003)

Machine learning

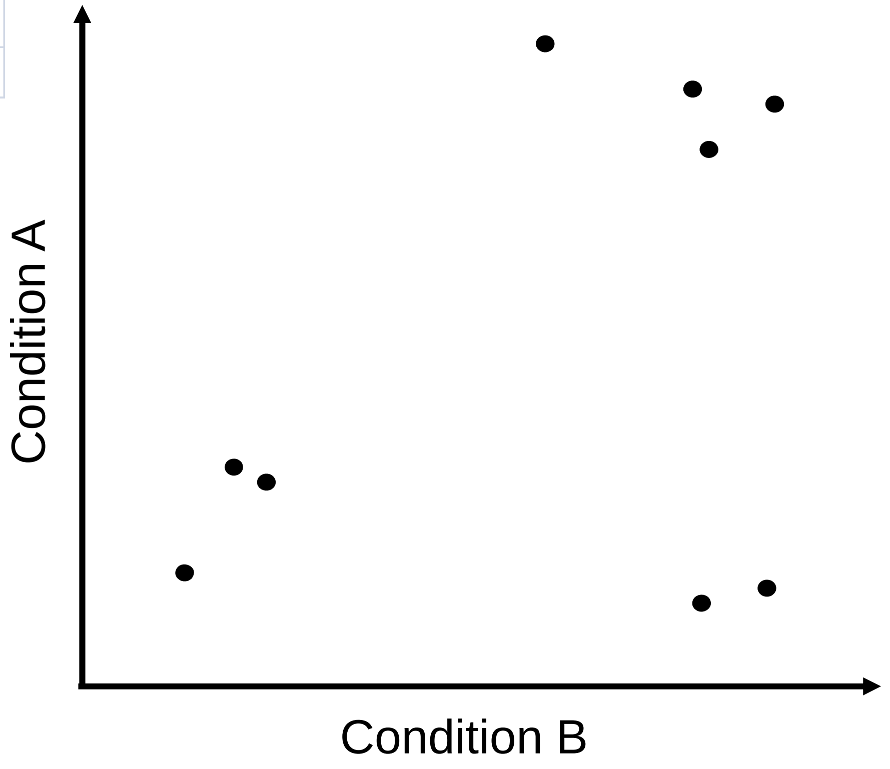
- Supervised learning; used to learn a model from a set of examples with predefined classes (training set)
- Unsupervised learning (clustering, class discovery); used to “discover” natural groups observations

Conditions/tissues/time

Genes/metabolites/proteins

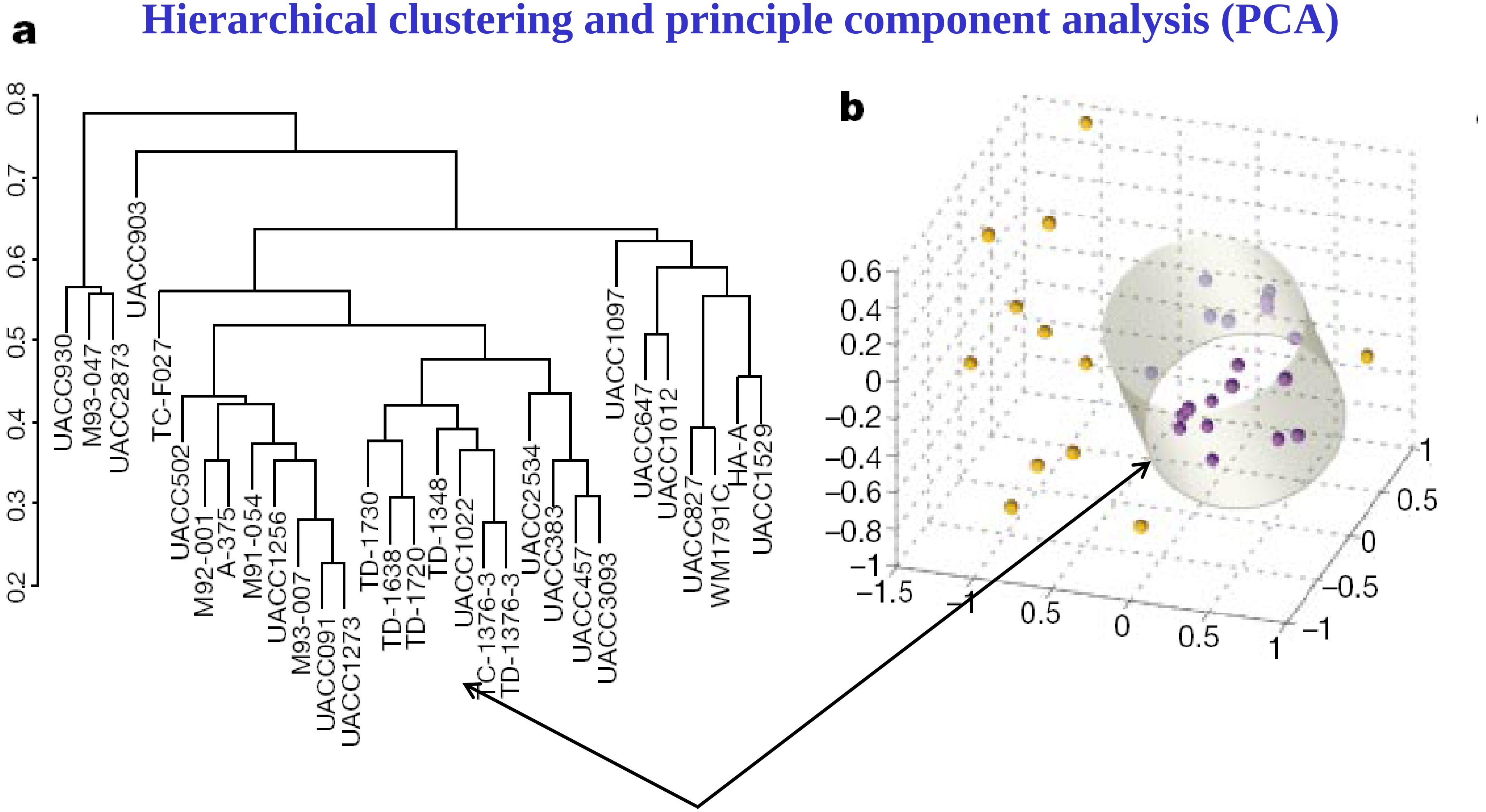
0.54	0.53	0.16	0.14	0.20	-0.34	-0.38	-0.36
-0.47	-3.32	-0.81	0.11	-0.60	-1.36	-1.03	-1.84
0.66	0.07	0.20	0.29	-0.89	-0.45	-0.29	-0.29
0.14	-0.04	0.00	-0.15	-0.58	-0.30	-0.18	-0.38
-0.04	0.00	-0.23	-0.25	-0.47	-0.60	-0.56	-1.09
0.28	0.37	0.11	-0.17	-0.18	-0.60	-0.23	-0.58
0.54	0.53	0.16	0.14	0.20	-0.34	-0.38	-0.36
0.20	0.14	0.00	0.11	-0.34	-0.03	0.04	-0.76
0.40	0.43	0.18	0.00	-0.14	0.29	0.07	-0.79
0.01	0.46	0.28	-0.34	-0.23	-0.36	-0.45	-0.64
…	…	…	…	…	…	…	…
-0.23	0.04	0.00	-0.30	-0.29	-0.45	-0.97	-2.06

Time series versus
Feature space



Unsupervised learning: Looking into more than 3D:

Hierarchical clustering and principle component analysis (PCA)



19 melanomas of all 31 cutaneous melanoma samples (Bitter et al. *Nature.* 406: 536, 2000)

Supervised learning: Training examples

M < 100

Gene/Expr	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	…	EM
G1	-0.47	-3.32	-0.81	0.11	-0.60	-1.36	-1.03	-1.84	-1.00	-0.60	…	-0.94
G2	0.66	0.07	0.20	0.29	-0.89	-0.45	-0.29	-0.29	-0.15	-0.45	…	-0.42
G3	0.14	-0.04	0.00	-0.15	-0.58	-0.30	-0.18	-0.38	-0.49	-0.81	…	-1.12
G4	-0.04	0.00	-0.23	-0.25	-0.47	-0.60	-0.56	-1.09	-0.71	-0.76	…	-0.62
G5	0.28	0.37	0.11	-0.17	-0.18	-0.60	-0.23	-0.58	-0.79	-0.29	…	-0.74
G6	0.54	0.53	0.16	0.14	0.20	-0.34	-0.38	-0.36	-0.49	-0.58	…	-1.47
G7	0.20	0.14	0.00	0.11	-0.34	-0.03	0.04	-0.76	-0.81	-1.12	…	-1.36
G8	0.40	0.43	0.18	0.00	-0.14	0.29	0.07	-0.79	-0.81	-0.92	…	-1.22
G9	0.01	0.46	0.28	-0.34	-0.23	-0.36	-0.45	-0.64	-0.79	-1.22	…	-1.09
…	…	…	…	…	…	…	…	…	…	…	…	…
GN	-0.23	0.04	0.00	-0.30	-0.29	-0.45	-0.97	-2.06	-0.89	-1.22	…	-0.97

Cell growth

Transcripton

N > 10000

WT

Transgenic

Machine learning

- Supervised methods
  - Bayes decision rule
  - Nearest neighbor approaches
  - Decision tree learning/rule-based learning
  - Linear/non-linear classifiers
  - Neural networks
  - Genetic algorithms/programming
- Concepts
  - Classification versus regression
  - Curse of dimensionality
  - Overfitting
  - Validation

Example: Decision tree learning

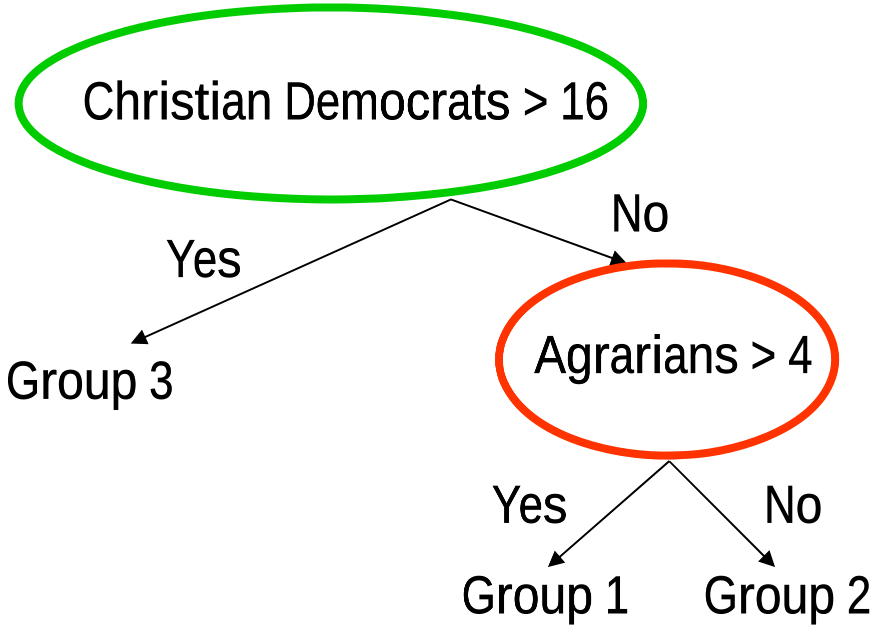
Country	Communists	Socialists	Greens	Social Democrats	Liberals	Agrarians	Subnational, regional and ethnic parties	Christian Democrats	Conservatives	Extreme Right
Norway	0	7	0	38	4	8	0	9	24	6
Sweden	6	0	2	43	10	17	0	2	18	1
Denmark	4	9	0	33	13	14	0	3	15	9
Finland	15	0	2	24	3	25	5	3	21	0
Iceland	0	18	3	16	4	22	0	0	36	0
UK	0	0	9	39	15	0	4	0	42	0
Netherlands	2	5	0	30	23	0	0	37	0	0
Belgium	2	0	4	27	19	0	14	31	0	2
Luxembourg	6	1	3	31	21	0	0	34	0	1
Switzerland	2	2	7	22	23	11	0	22	3	5
Austria	1	0	2	48	0	0	0	41	0	8
Germany	1	0	3	40	9	0	0	46	0	1
France	15	2	2	28	20	0	0	0	25	5
Italy	29	0	3	15	4	0	3	35	2	6
Greece	10	0	0	39	6	0	0	0	44	0
Spain	8	0	0	39	16	0	10	0	21	0
Portugal	15	0	1	31	38	0	0	1	11	0

Class knowledge:

Group 1: Nordic countries

Group 2: UK, France, Greece, Spain, Portugal

Group 3: Benelux countries, Switzerland, Austria, Italy, Germany



Machine learning terminology

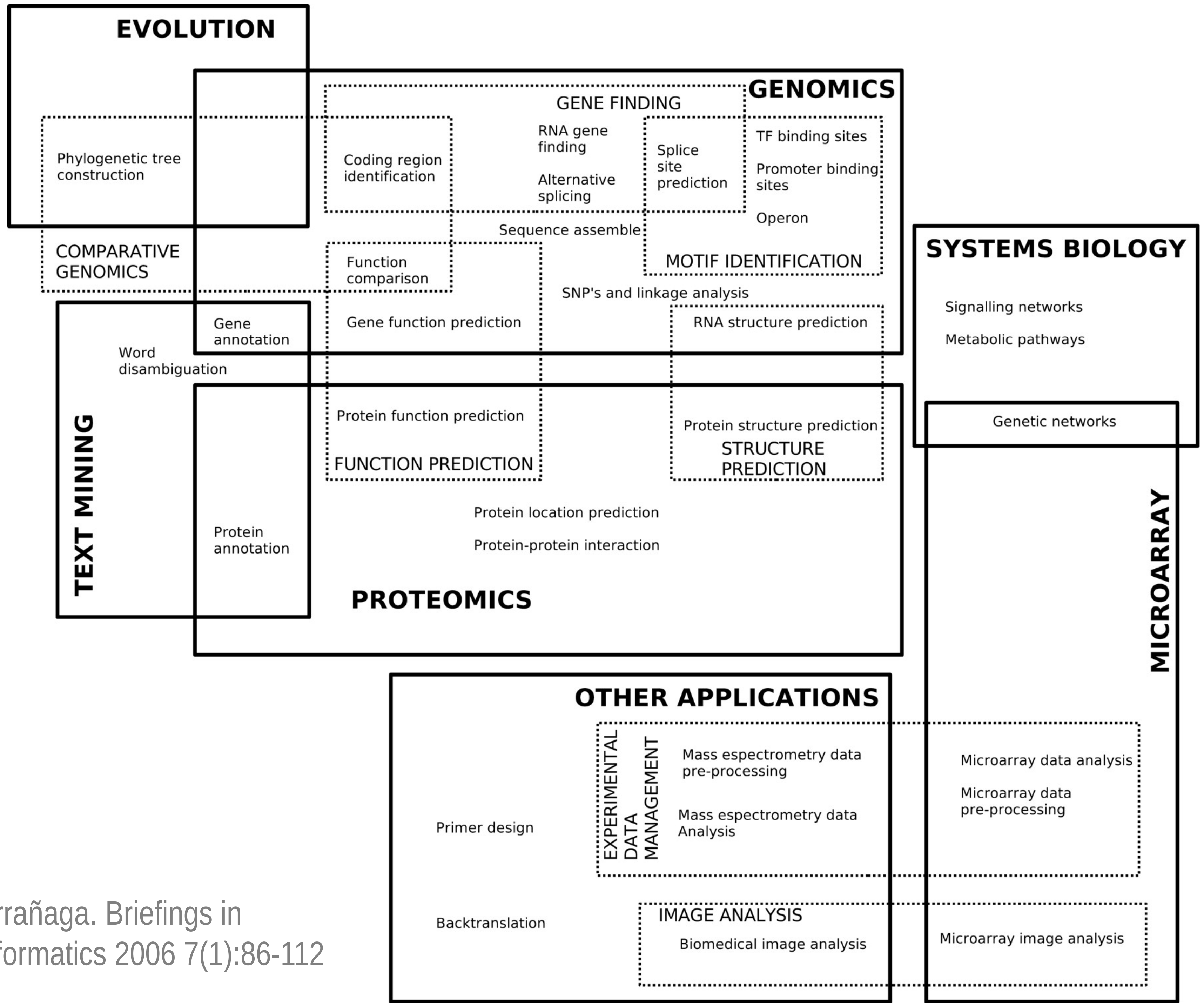
Some concepts:

1. Data: Observations collected from the real world (e.g. the voting pattern in Sweden). Observations consist of a number of features (e.g. communist votes)
2. Examples: Observations labeled with class information (e.g. Sweden belong to group 1).
3. Model: A general representation of the data (e.g. the decision tree)

Models are induced!

1. Induction: Using specific information/data to arrive at general knowledge (e.g. from examples to a decision tree).
2. Deduction: Using general knowledge to say something about a specific case (e.g. using a decision tree to predict the group of a new country).

Models can be predictive and/or descriptive.



P. Larrañaga. Briefings in Bioinformatics 2006 7(1):86-112

Bayes decision rule

Prior Probability

- w - classes, e.g.
  - w_1 the object is a fish, w_2 the object is a bird, etc.

- *A priori* probability (or prior) P(w_i)

Class-conditional probability

- Given class information (training data), we observe x, e.g.
  - The objects has wings
  - The object has eyes

- Class-conditional probability p(x|w)

Bayes decision rule

Suppose the priors $P(\omega_j)$ and conditional densities $p(x|\omega_j)$ are known

likelihood prior

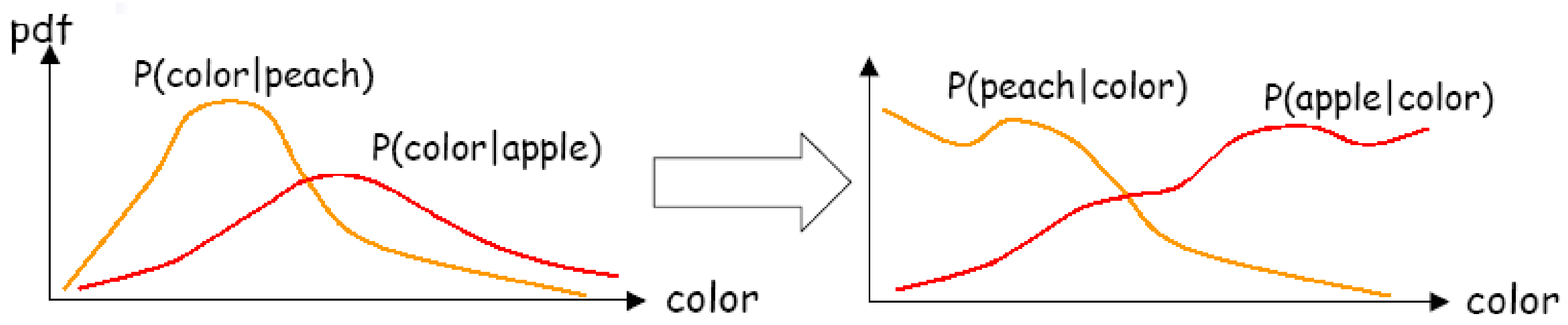
$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j)P(\omega_j)}{p(x)}$$

posterior evidence

Bayes decision rule:

$P(\omega_1|x) > P(\omega_2|x)$ then choose ω_1, else choose ω_2.

Example

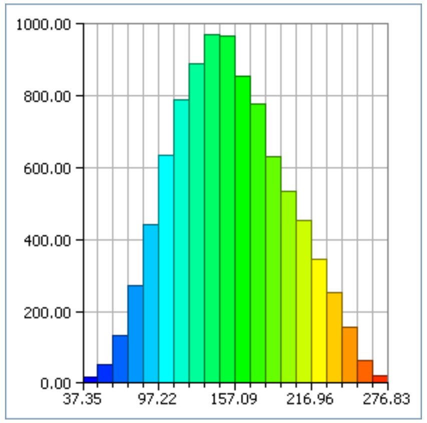


- Bayes Decision Rule
  - If P(apple | color) > P(peach | color) then choose apple

- Note that the evidence p(color) is only necessary for normalization purposes; it does not affect the decision rule

So, what about the data?

- Use the examples to estimate the probability distributions (training data):
  - $P(w_j)$ is easy.
  - $p(x|w_j)$: Histogram!



- One feature: bins are rectangles, Two features: cubes, *n*-features: hyper-cubes.
- More dimentions/features require more training data: Curse of dimensionality!
  - If we need 10 observations when we have one feature (to get a good histogram), then we need 10^n observations when we have *n*-features!
- If the true probability distributions are known, then Bayes decision rule is optimal (minimizes error rate).

Training examples

M < 100

Gene/Expr	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**	**…**	**EM**
G1	-0.47	-3.32	-0.81	0.11	-0.60	-1.36	-1.03	-1.84	-1.00	-0.60	…	-0.94
G2	0.66	0.07	0.20	0.29	-0.89	-0.45	-0.29	-0.29	-0.15	-0.45	…	-0.42
G3	0.14	-0.04	0.00	-0.15	-0.58	-0.30	-0.18	-0.38	-0.49	-0.81	…	-1.12
G4	-0.04	0.00	-0.23	-0.25	-0.47	-0.60	-0.56	-1.09	-0.71	-0.76	…	-0.62
G5	0.28	0.37	0.11	-0.17	-0.18	-0.60	-0.23	-0.58	-0.79	-0.29	…	-0.74
G6	0.54	0.53	0.16	0.14	0.20	-0.34	-0.38	-0.36	-0.49	-0.58	…	-1.47
G7	0.20	0.14	0.00	0.11	-0.34	-0.03	0.04	-0.76	-0.81	-1.12	…	-1.36
G8	0.40	0.43	0.18	0.00	-0.14	0.29	0.07	-0.79	-0.81	-0.92	…	-1.22
G9	0.01	0.46	0.28	-0.34	-0.23	-0.36	-0.45	-0.64	-0.79	-1.22	…	-1.09
…	…	…	…	…	…	…	…	…	…	…	…	…
GN	-0.23	0.04	0.00	-0.30	-0.29	-0.45	-0.97	-2.06	-0.89	-1.22	…	-0.97

N > 10000

Sick

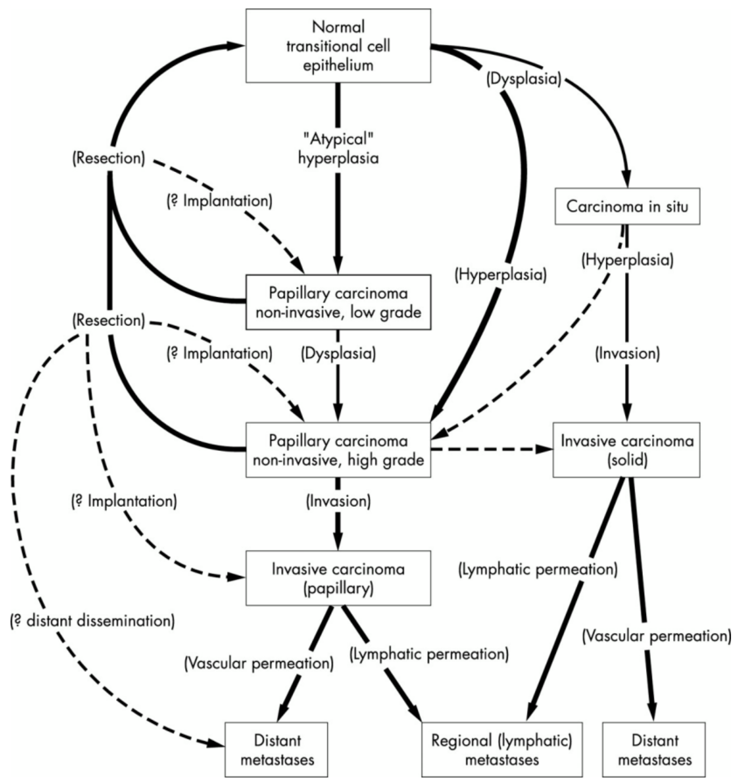
Healthy

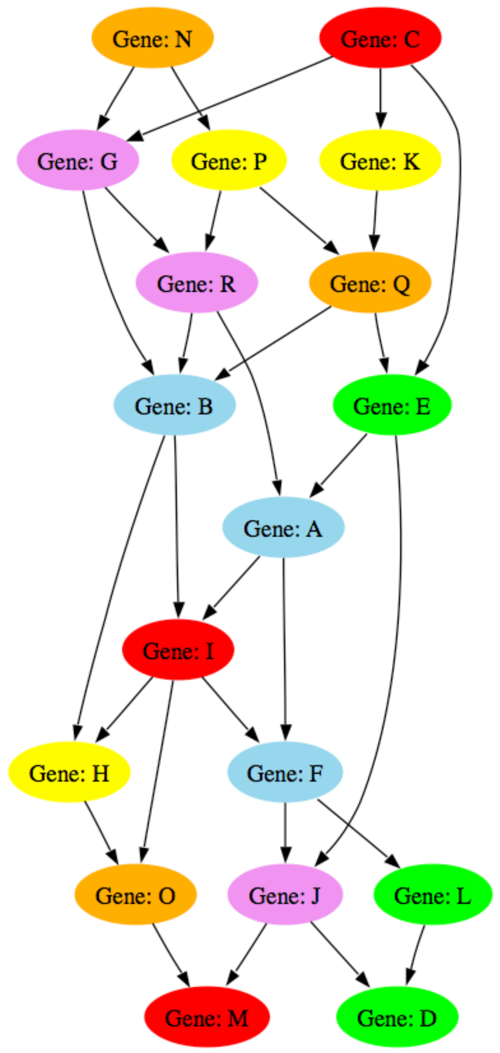
Feature selection

Feature selection is used to deal with the curse of dimensionality

- Ranking methods: compute a statistics (of the features discriminatory capability), rank the features and select the most discriminating ones
- Wrapper methods: select a subset of features, induce and validate the resulting model and repeat. Computationally expensive!
- Dimensionality reduction: map your features into a smaller features space (e.g. PCA)

Bayesian networks

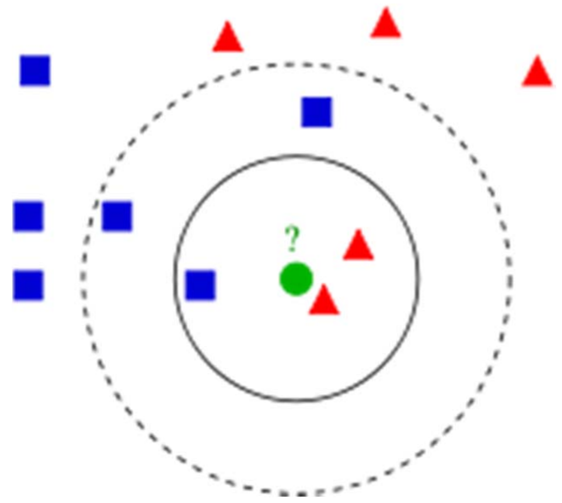




Nearest neigbour approaches

k-nearest neigboor

- The simplest of all machine learning algorithms.
- Each observation is a point in the *n*-dimensional space spanned by the features.
- An observation is assigned to the class most common amongst its *k* nearest neighbors.
- ”Nearest” can be defined differently: Euclidean distance, correlation, etc.
- Lazy learning where the function is only approximated locally and all computation is delayed until classification.



Decision tree learning
(rule-based learning)

Example: Cricket game

- Umpires' decision to play a cricket match
  - Data on three factors thought to influence the decision

Weather	Light	Ground condition	Umpires' decision
Sunny	Good	Dry	Play
Overcast	Good	Dry	Play
Raining	Good	Dry	No play
Overcast	Poor	Dry	No play
Overcast	Poor	Damp	No play
Raining	Poor	Damp	No play
Overcast	Good	Damp	Play
Sunny	Poor	Dry	Play

- Task: determine the rules the umpires are explicitly or implicitly using

Decision tree algorithm

- Aim: split the data so that each resulting subset belongs to one class
- Algorithm summary:
  1. For each feature, compute the goodness of the split
  2. Select the best feature and split the data according to the values in that feature
  3. If each of the subsets contains only one class, then stop. Otherwise reapply 1-3 on each of the subsets
  4. If the data is not completely classified, but there are no more splits available, then stop

Example split: Cricket game

- Need to divide the set of training examples into two smaller sets: 'Play' and 'No play'
- *Light* = *Good* yields four examples:

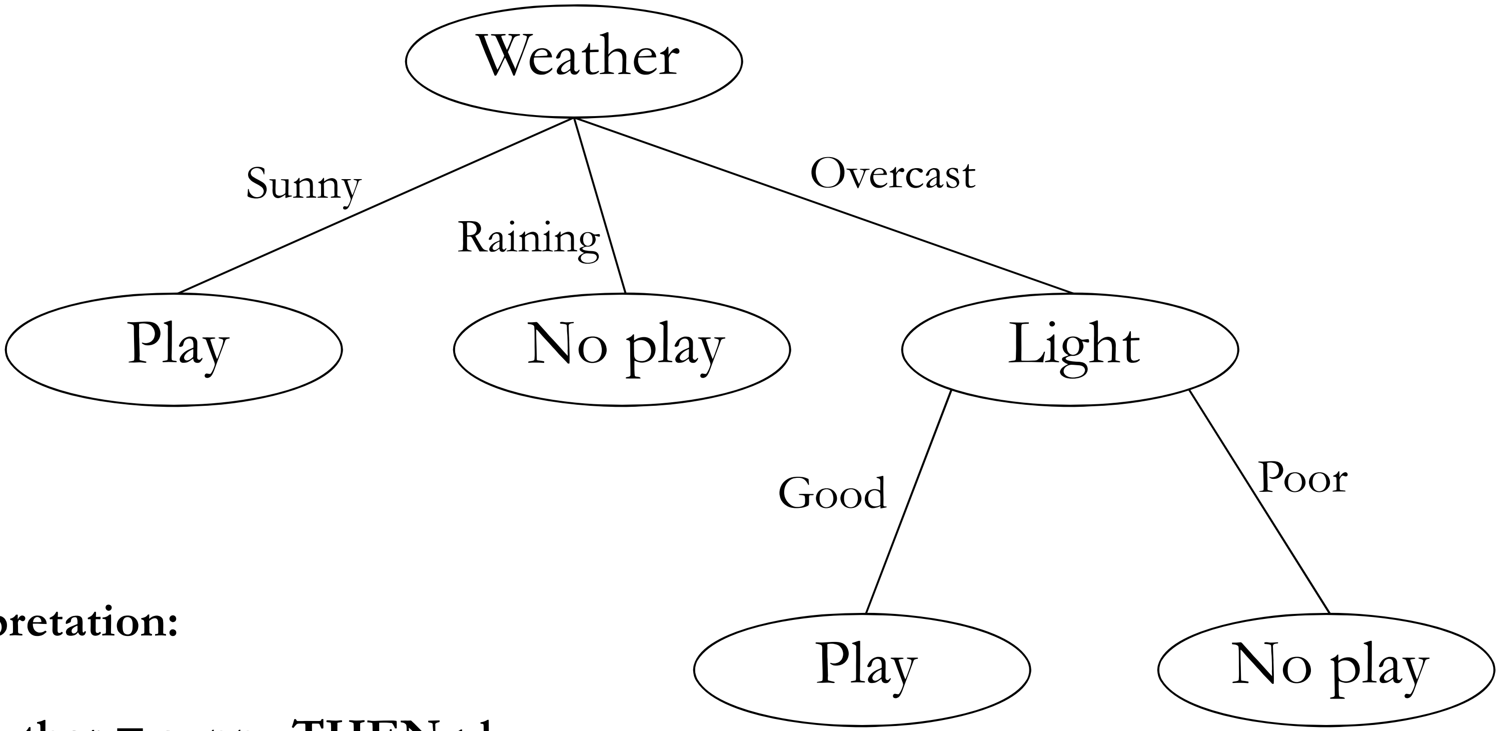
Sunny	**Good**	**Dry**	**Play**
Overcast	**Good**	**Dry**	**Play**
Raining	**Good**	**Dry**	**No play**
Overcast	**Good**	**Damp**	**Play**

- *Light* = *Poor* yields four examples:

Overcast	**Poor**	**Dry**	**No play**
Overcast	**Poor**	**Damp**	**No play**
Raining	**Poor**	**Damp**	**No play**
Sunny	**Poor**	**Dry**	**Play**

Cricket game

Final decision tree:



Interpretation:

IF weather = sunny THEN play
IF weather = raining THEN no play
IF weather = overcast AND light = good THEN play
IF weather = overcast AND light = poor THEN no play

Overfitting

- Overfitting: The method learns the random patterns in the data as well as the underlying process that created the data
  - Occurs because the alg. tries to reduce the classification error

- **To identify this phenomenon:**
  - **Split data into training data (≈75%) and test data (≈25%)**
  - **Build tree on the training data and test the model on the test data**

➢ **A decision tree X is overfitted if there exists a tree Y that do better on an unseen test set, but worse on the training set**

- "Solution": Prune complex branches of the tree

Occam’s razor

- William of Occam 14th century: *things should not be multiplied unnecessarily*
- Issac Newton (1687): *we are to admit no more causes of natural things than such as are both true and sufficient to explain their appearance*
- Albert Einstein (20th century): *everything should be made as simple as possible, but not simpler*

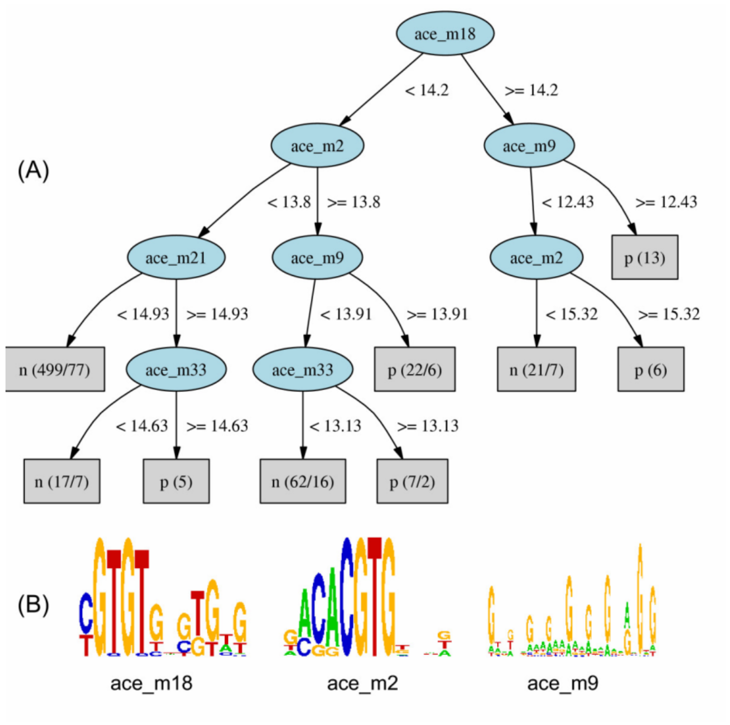
The simplest model that explains the data should be chosen

Decision trees: greedy algorithm

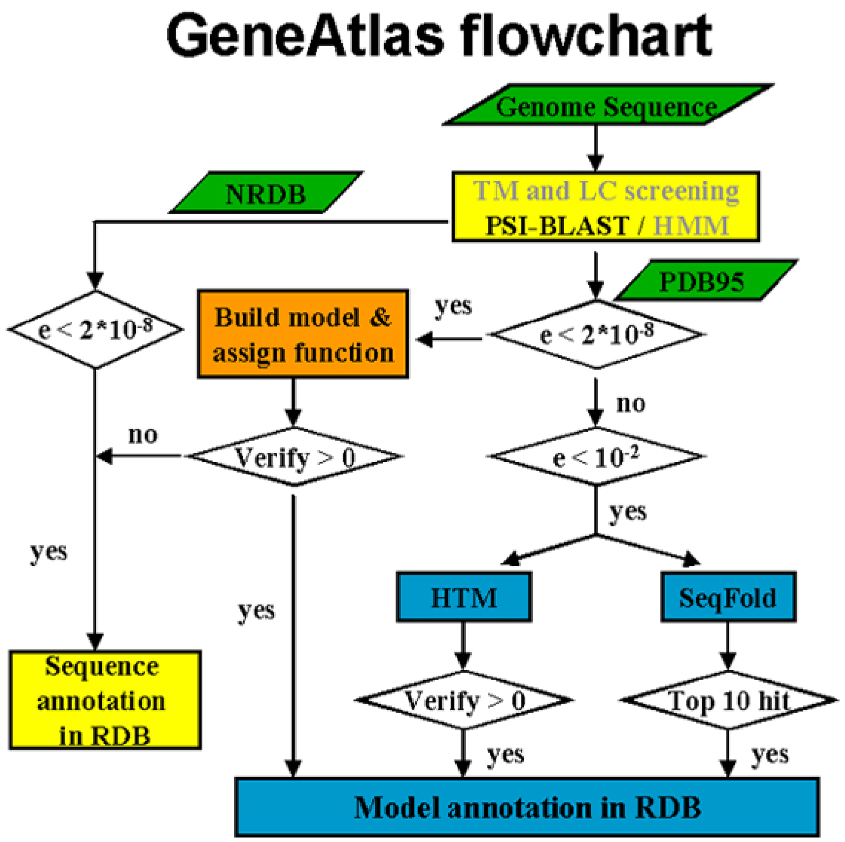
- Decision trees are built by iteratively splitting the training examples using the “best” feature: greedy

- Would benefit from some search strategy
  - A split could be evaluated in terms of its current ability to classify the data AND the accuracy of the splits later on in the algorithm run

Decision tree and motifs learned for ABA-responsive genes in Arabidopsis



Pipeline for protein structure prediction and function annotation



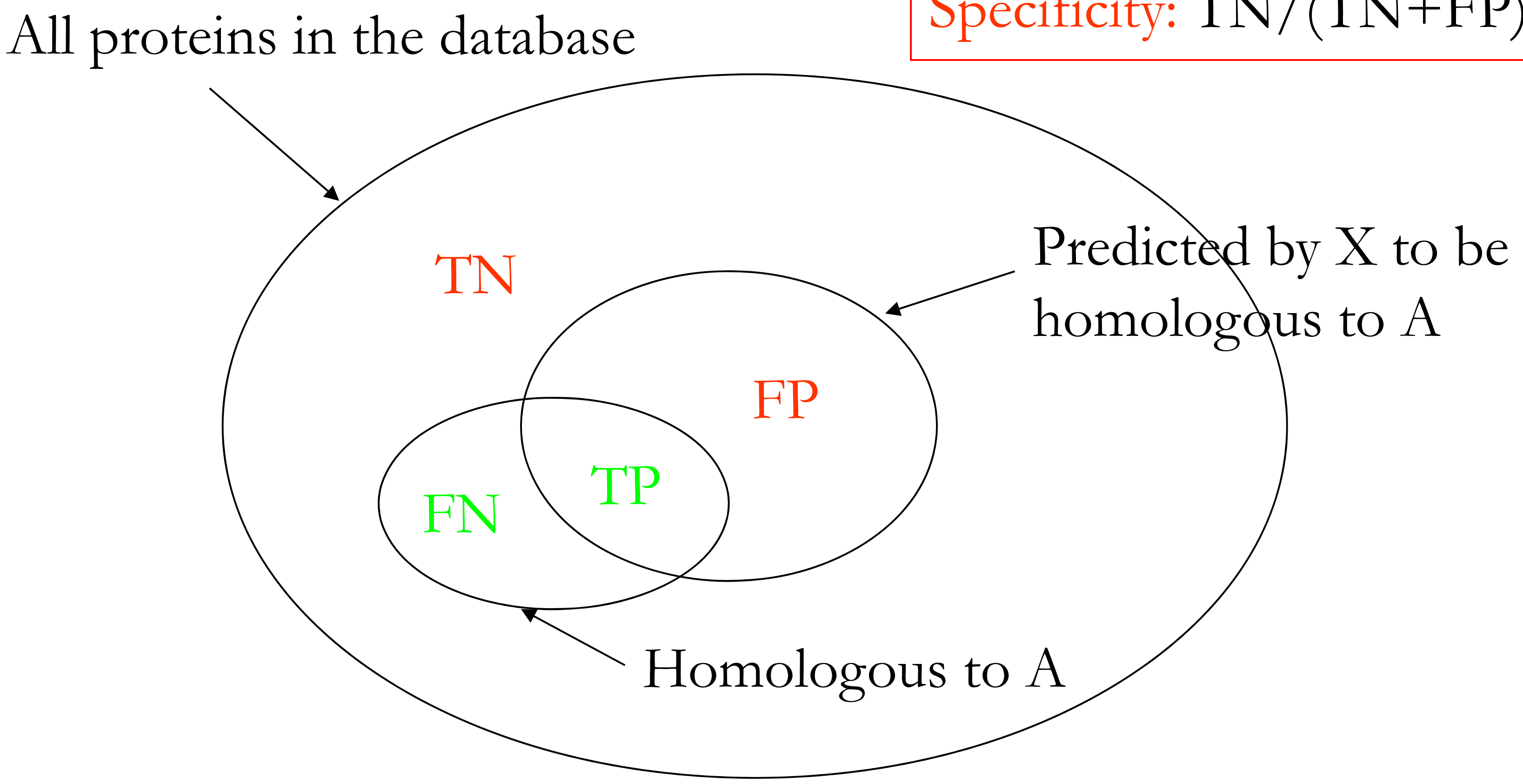
Model validation

Method power

You want to find homologous proteins to a specific protein A using some computational method X:

Sensitivity: TP/(TP+FN)

Specificity: TN/(TN+FP)

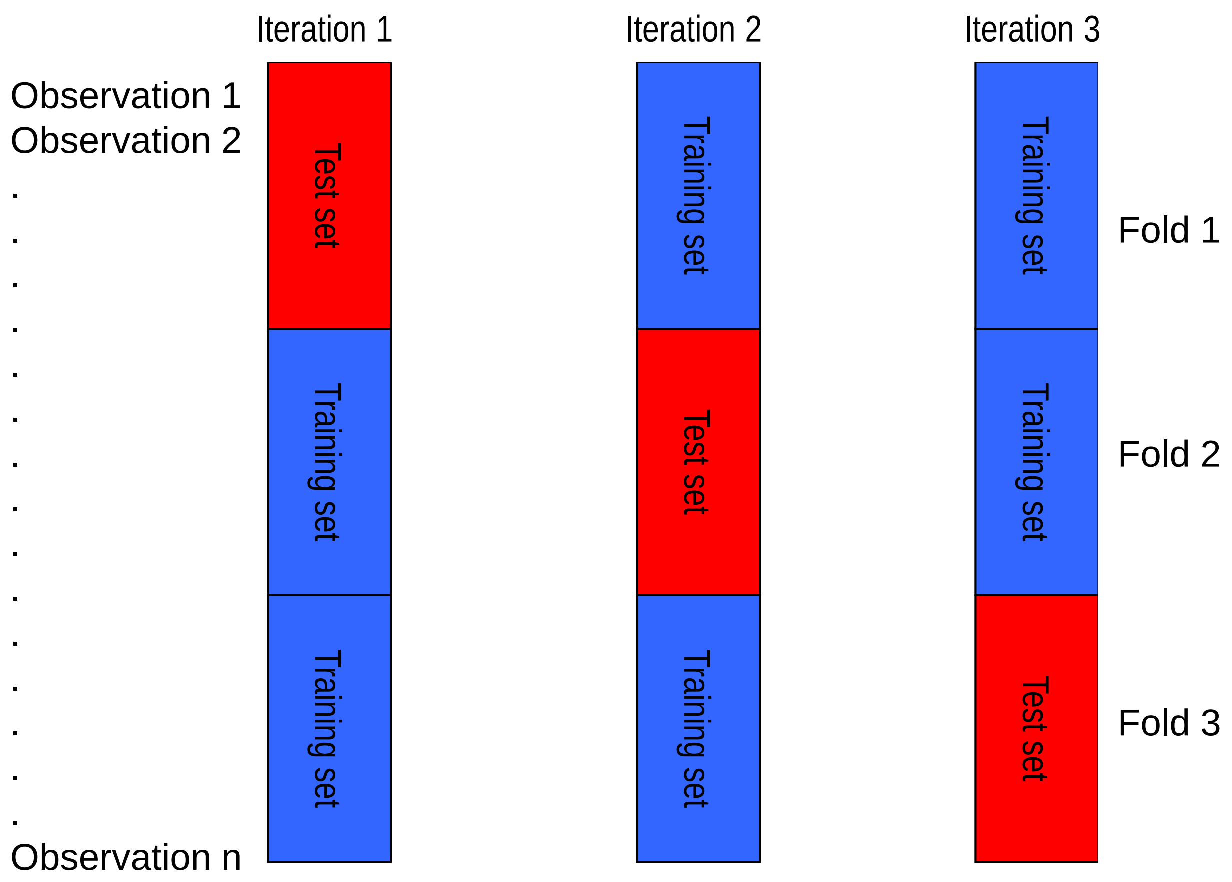

All proteins in the database
TN
Predicted by X to be homologous to A
FP
TP
FN
Homologous to A

Evaluation

- Classifications can be
  - True positives (TP)
  - False negatives (FN)
  - True negatives (TN)
  - False positives (FP)
- Evaluation measures:
  - accuracy = (TP+TN)/(TP+FN+TN+FP)
  - sensitivity = TP/(TP+FN)
  - specificity = TN/(TN+FP)
- Confusion matrix:

		Predicted	
		Class 0	Class 1
Actual	Class 0	TN	FP
	Class 1	FN	TP

Cross validation



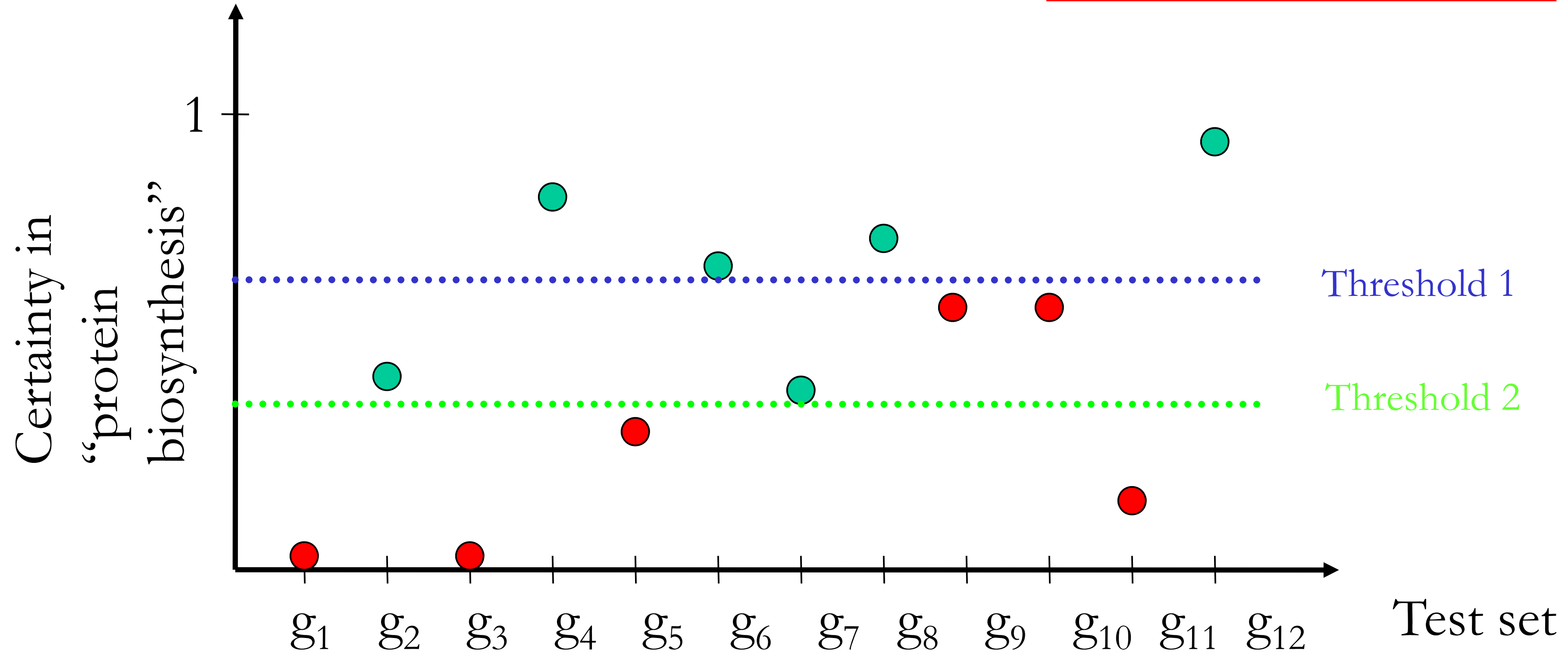
- *k*-fold cross validation: *k* iterations
- Leave-one out cross validation: *n* iterations

Threshold selection

● Gene with function "protein biosynthesis"

● Gene with a different function

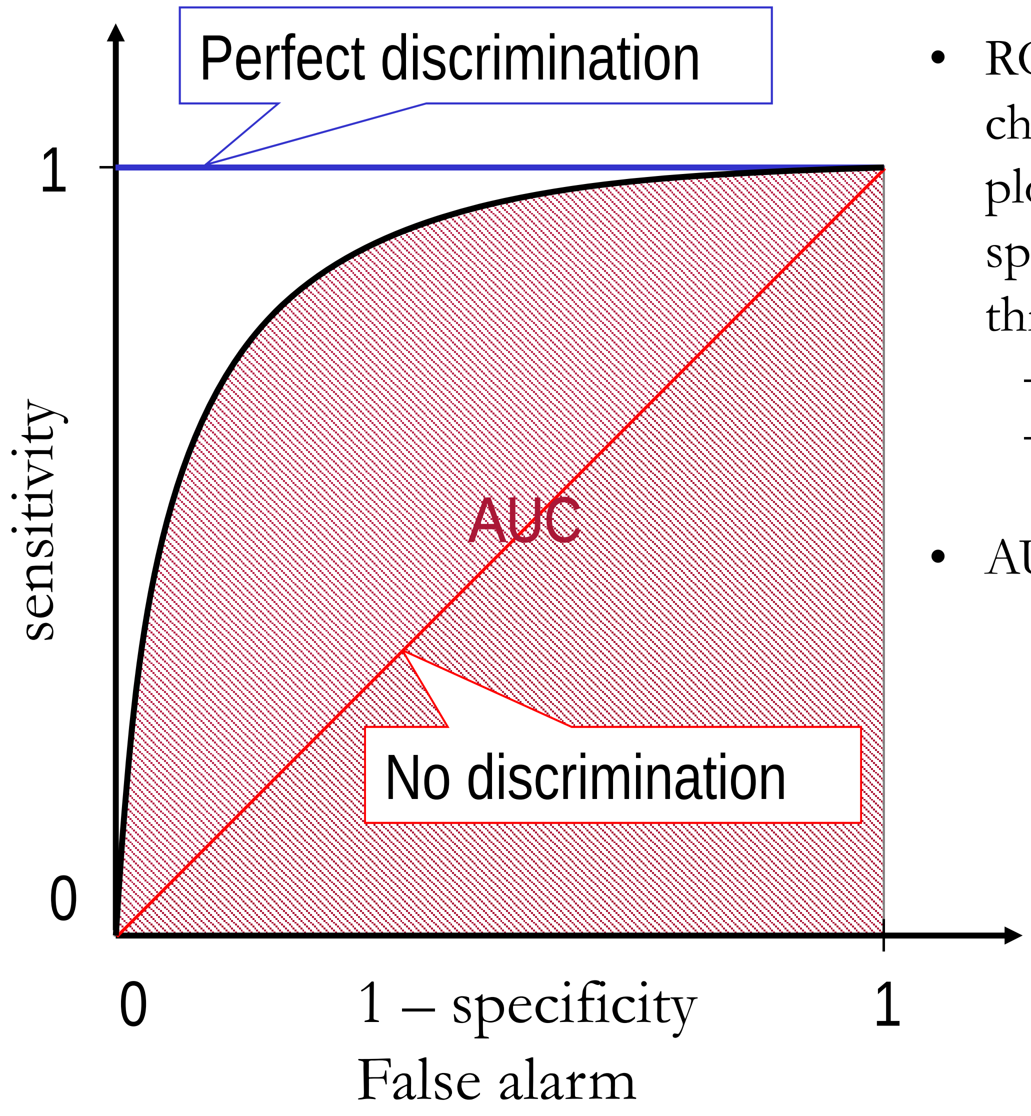
sensitivity:
TP/(TP+FN)
specificity:
TN/(TN+FP)



Sensitivity = 2/3, Specificity=1

Sensitivity = 1, Specificity=2/3

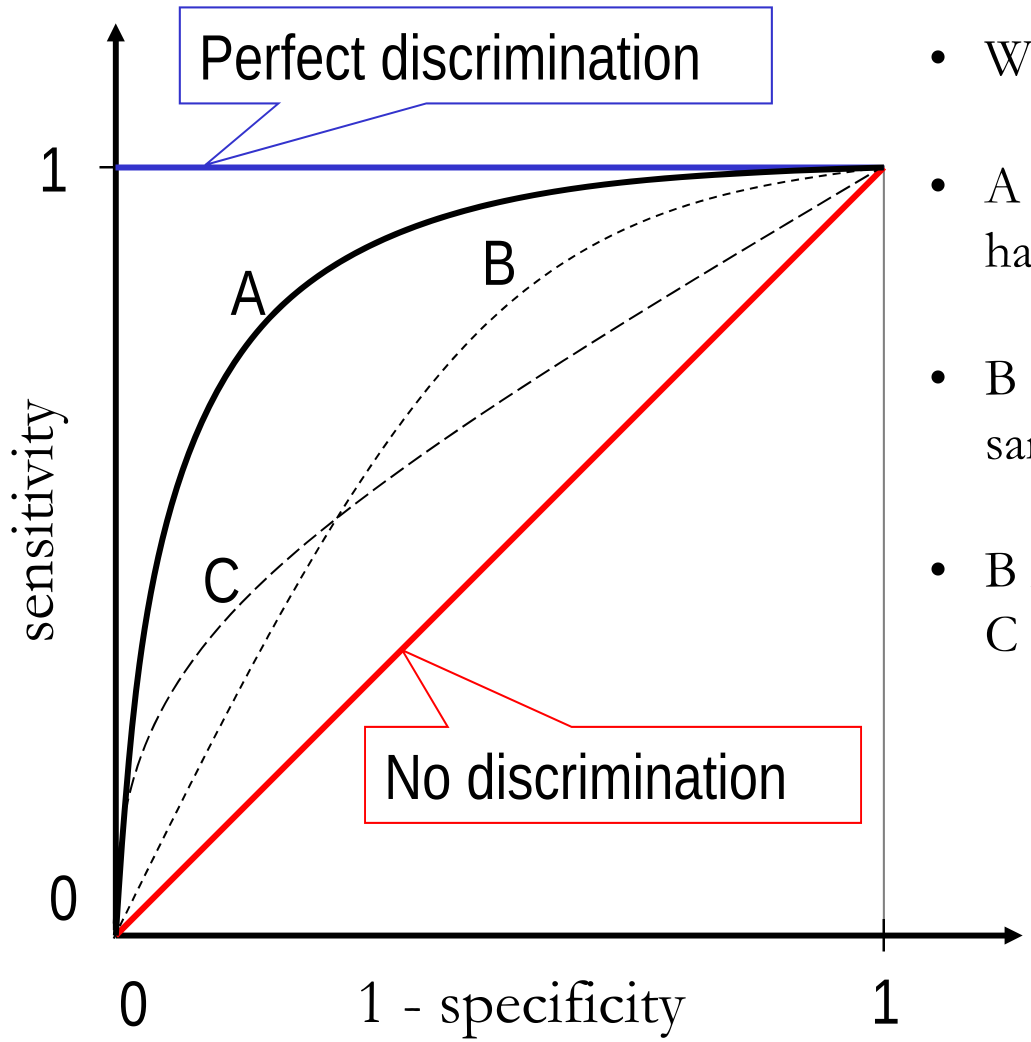
ROC analysis and classifier evaluation



- ROC: Receiver operating characteristics curve results from plotting sensitivity against specificity for all possible thresholds
  - sensitivity: TP/(TP+FN)
  - specificity: TN/(TN+FP)

- AUC: Area under the ROC curve

ROC analysis and classifier evaluation

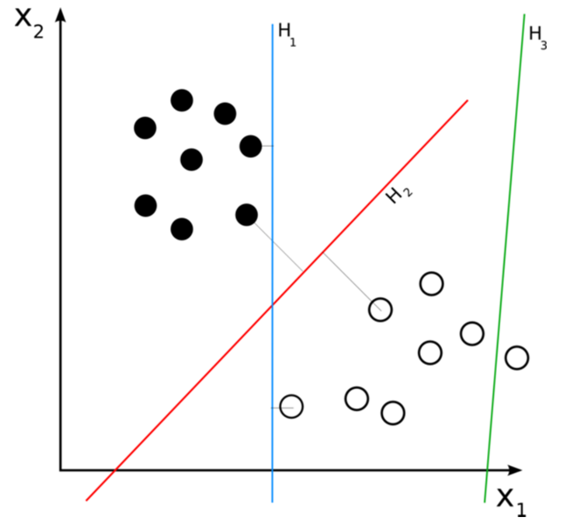


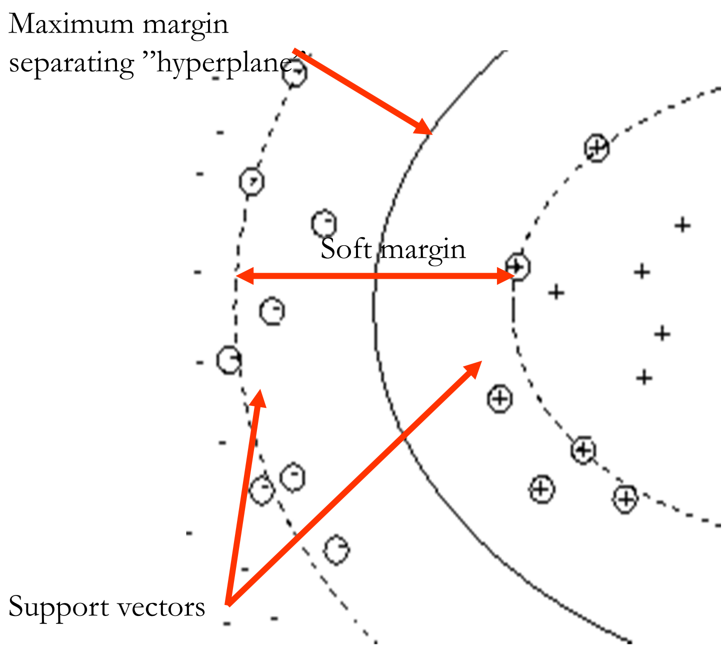
- Which ROC curve is better?
- A dominants B and C and clearly has a higher AUC
- B and C have approximately the same AUC
- B is better for some thresholds, C for others

Artificial neural networks (linear versus non-linear methods)

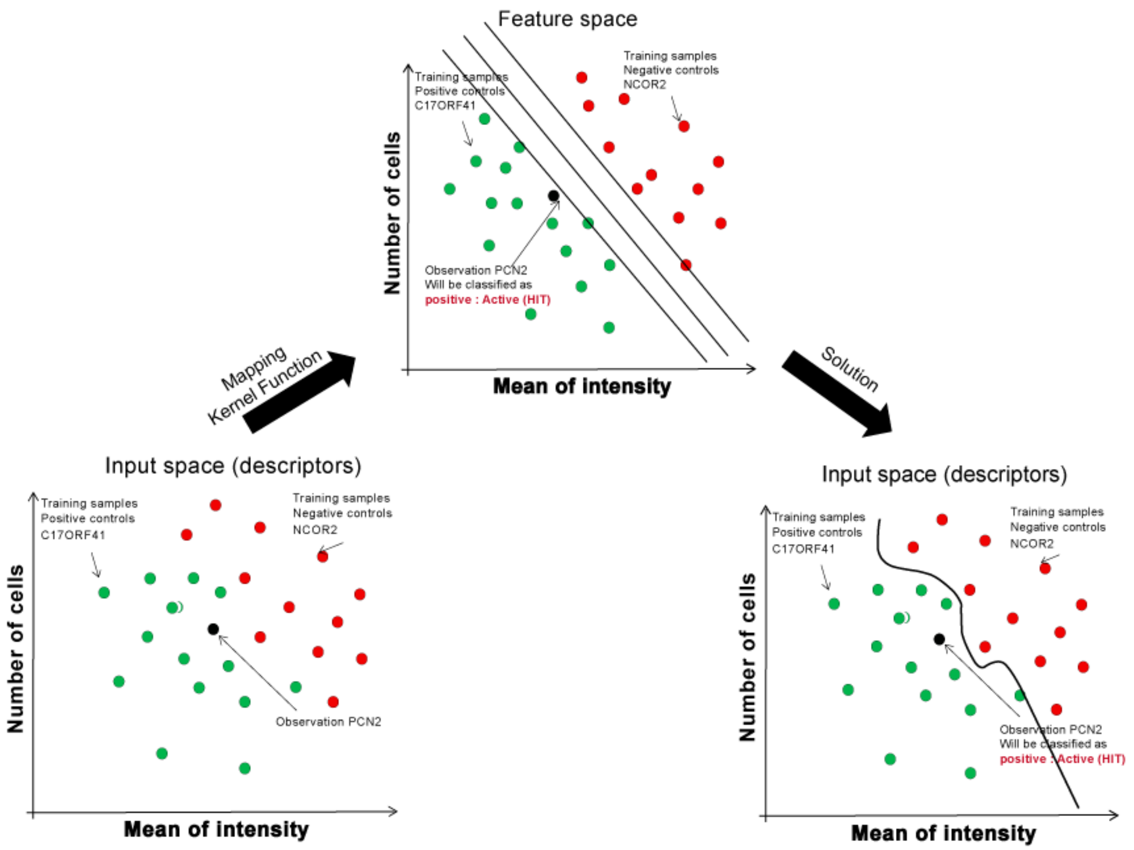
Linear versus non-linear classifiers

- Linear: Finds a hyperplane that separates the classes
  - In two dimensions: $w_0 + w_1 \cdot x_1 + w_2 \cdot x_2$
  - Use the examples $\boldsymbol{x}$ to estimate $\boldsymbol{w}$
- Non-linear:
  - Support vector machines uses the kernel trick: The kernel maps the observations into a higher dimensional space where the problem is linearly separable
  - Artifical neural networks





siRNA classification



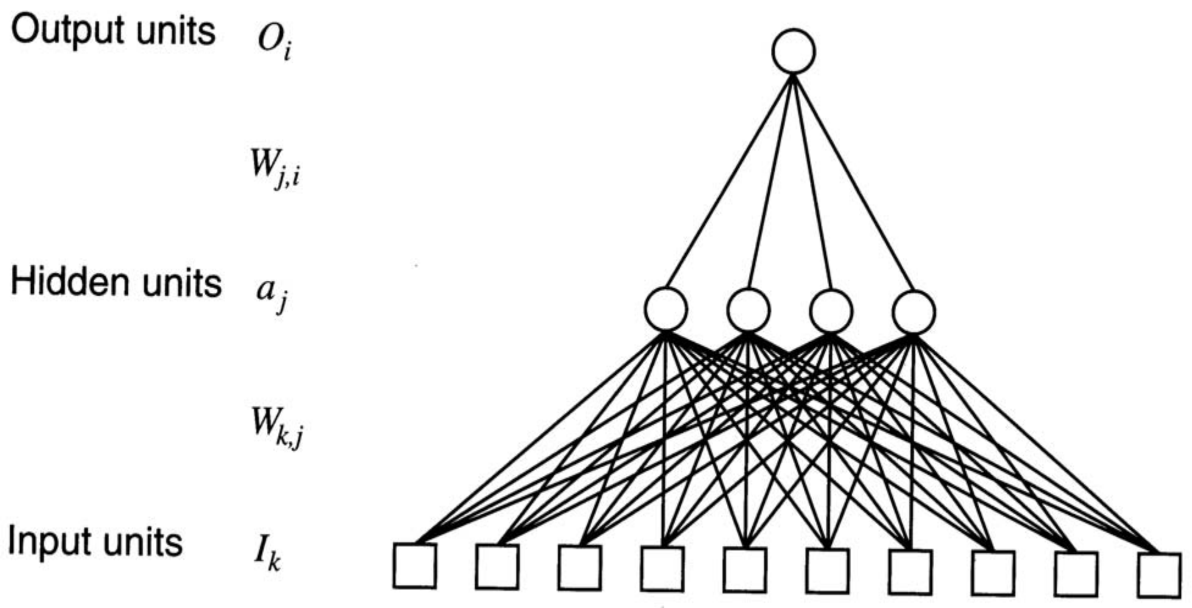
Artifical neural networks

- Inspired by how the brain works – a mathematical model for the operation of the brain

- An ANN is a number of nodes (units) connected by links. Each link is associated with a numerical weight.
  - Training set: $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$
  - Learning in an ANN is reduced to the process of using the training data to tune the weights so that the network represents the function f

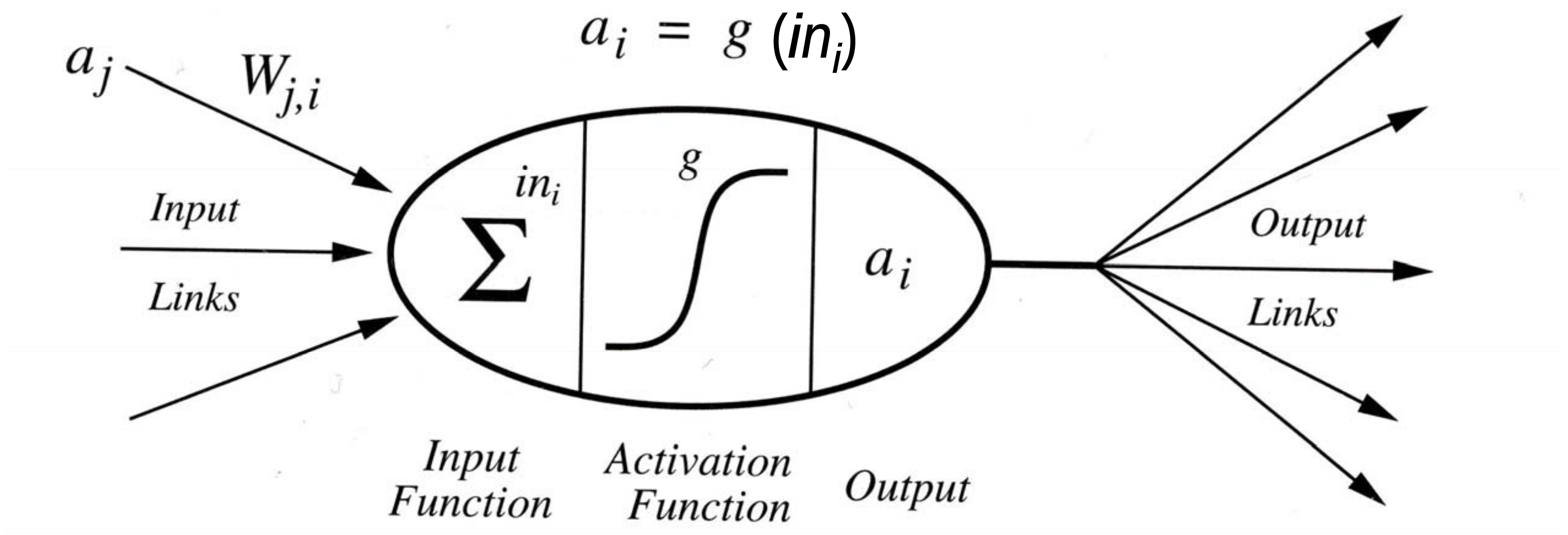
Network structure

- Feed-forward network: all units are connected to all units in the next layer
  - One (sufficiently large) hidden layer can represent any continuous function
  - More hidden layers can even represent discontinuous functions



- Recurrent network: feed back loops, internal states (memory):
  - E.g. The brain is clearly a recurrent network

Units



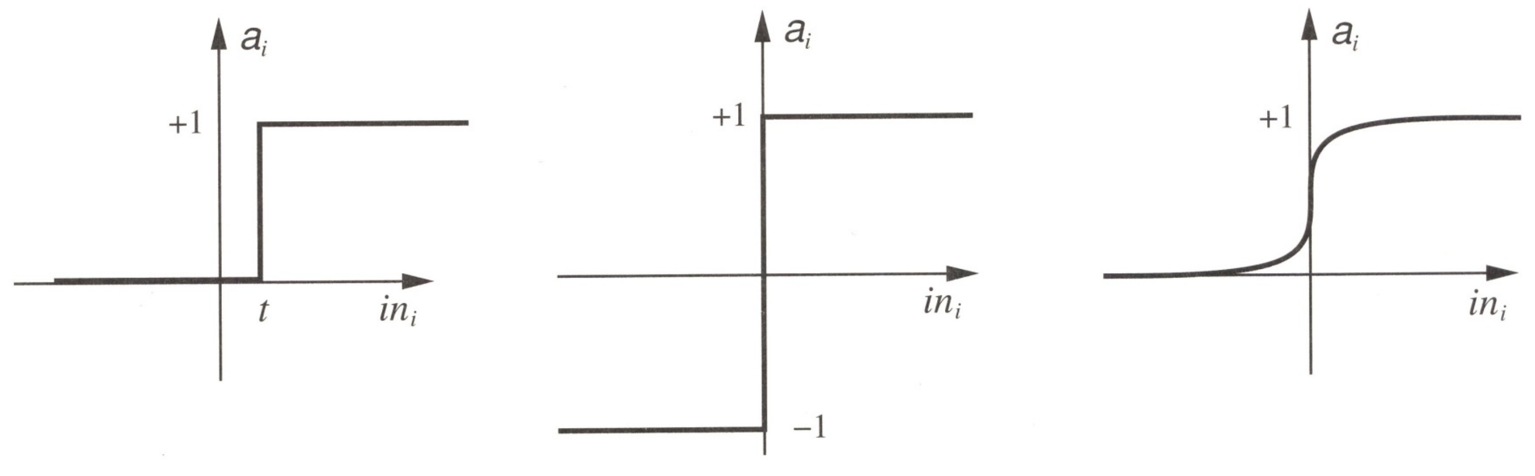
- Input function: linear component, in_i that compute the weighted sum of the units input values

$$in_i = \sum_j W_{j,i} \cdot a_j$$

- Activation function: nonlinear transformation, g, of the input function into the unit's activation value

$$a_i = g(in_i)$$

Activation functions



(a) Step function **(b) Sign function** **(c) Sigmoid function**

Three different activation functions for units.

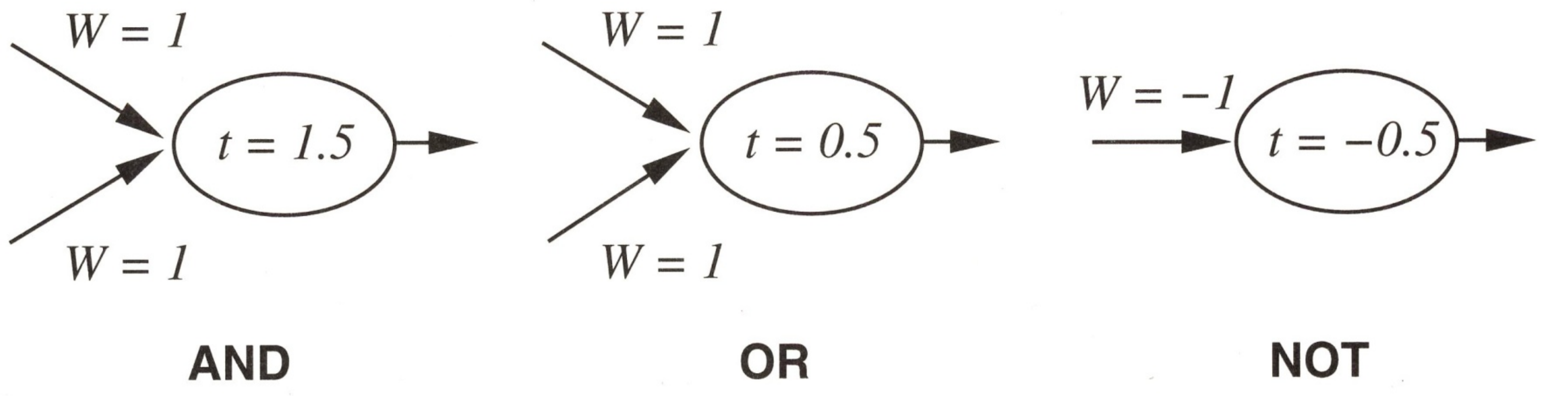
$$\text{step}_t(x) = \begin{cases} 1, & \text{if } x \geq t \\ 0, & \text{if } x < t \end{cases} \quad \text{sign}(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases} \quad \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

Typically, the threshold of the activation function is embedded in the input function:

$$a_i = step_t\left(\sum_{j=1}^{n} W_{j,i} \cdot a_j\right) = step_0\left(\sum_{j=0}^{n} W_{j,i} \cdot a_j\right), \text{where } W_{0,j} = t \text{ and } a_0 = -1$$

Boolean functions

- Units can represent the basic logical gates
- Thus, units can build networks that can represent any Boolean function

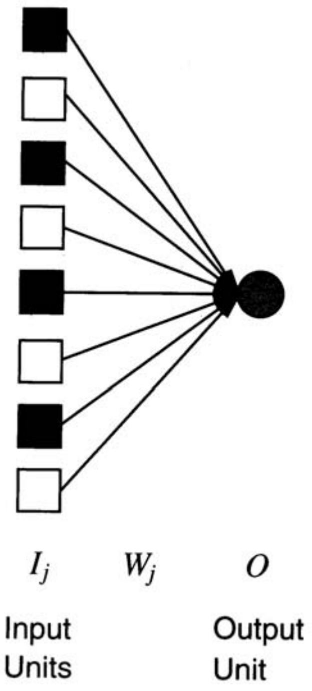


Optimal network structures

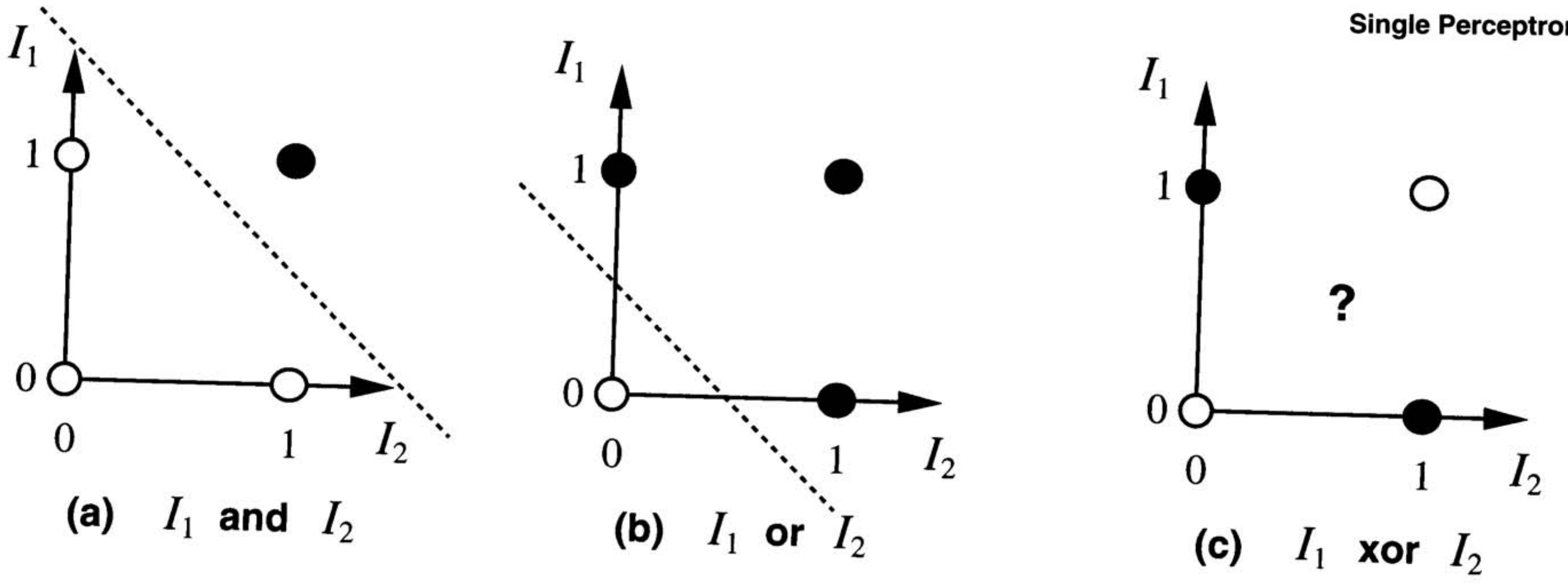
- Too small network: the network will be incapable of representing the desired function
- Too large network: the network can memorize all the examples by forming a lookup table
  - Overfitting!
- To identify this phenomenon:
  - Use training/test sets
  - Choose the simples model that explains the data! Occam's razor
- Finding the optimal network structure is itself a search problem
  - Potentially time-consuming since every state in this search involves training and evaluating a neural network of a particular size

Perceptrons

- Perceptrons: single-layer, feed-forward networks
  - Majority function: outputs 1 if a majority of the *n* inputs are 1 (would require a decision tree with $O(2^n)$ nodes)
- A perceptron can only represent a function if there is a line that separates all the white dots (0s) from the black dots (1s), i.e. functions that are linearly separable



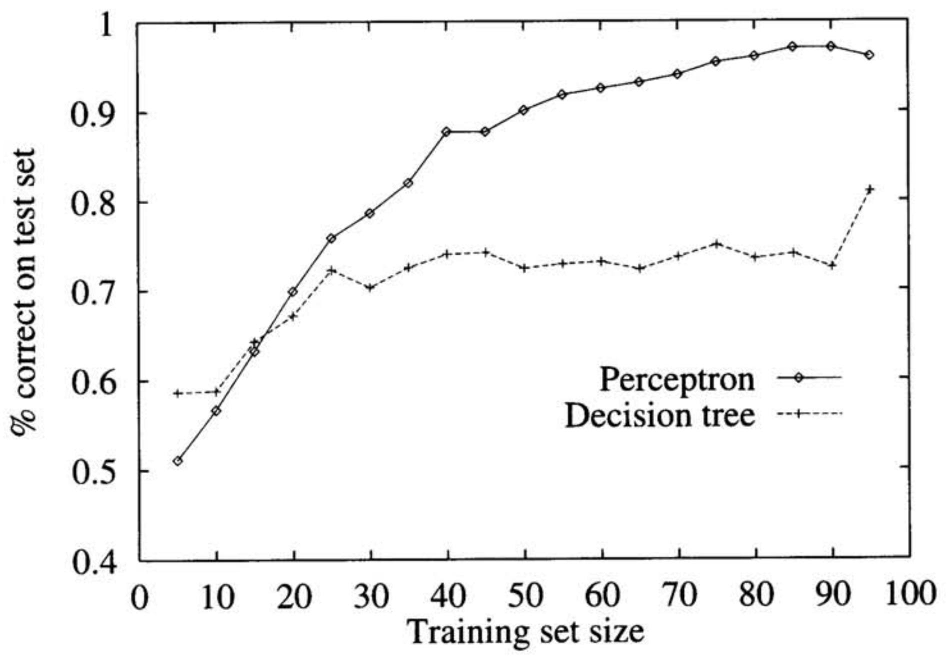
Single Perceptron



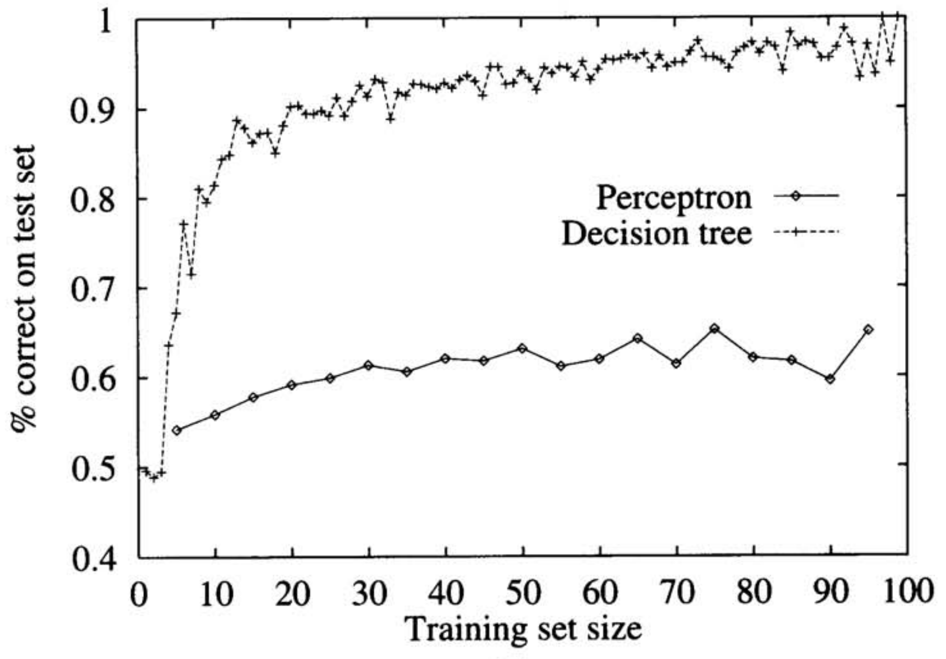
Learning linearly separable functions

- Only one unit: $O = Step_0(\sum_j W_j I_j)$
- Err = $T - O$, where T is the correct output
- Perceptron learning rule: $W_j \leftarrow W_j + \alpha \times I_j \times Err$
  - α is called the learning rate
- Learning algorithm:
  - Initiate weights, e.g. random values between 0 and 1
  - For each example
    - Compute O
    - Update weights with the learning rule
  - Repeat until all examples are correctly predicted
- Epoch: updating all weights for every example
- Note: the perceptron rule is guaranteed to learn a linearly separable function given enough examples!

Perceptrons versus decision trees: Example

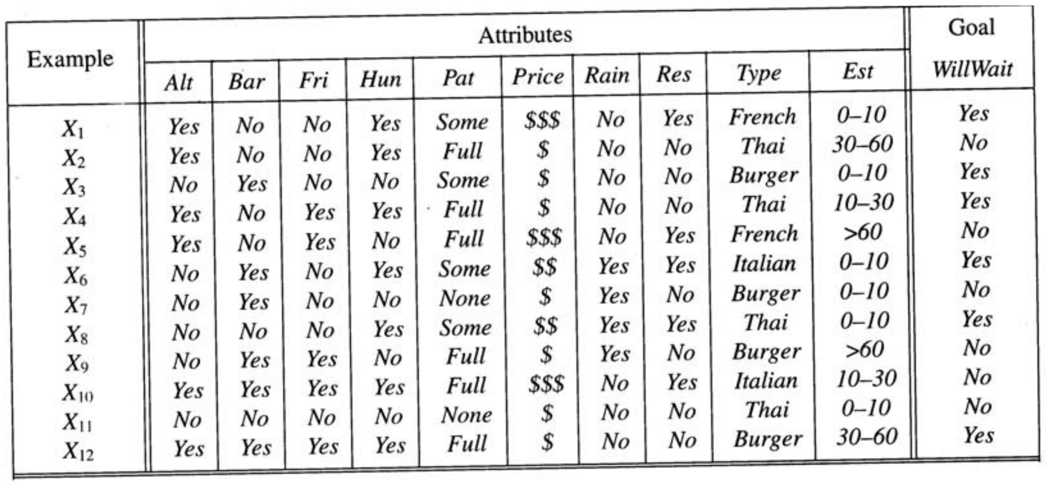


(a)



(b)

(a) Majority function
(b) Waiting problem

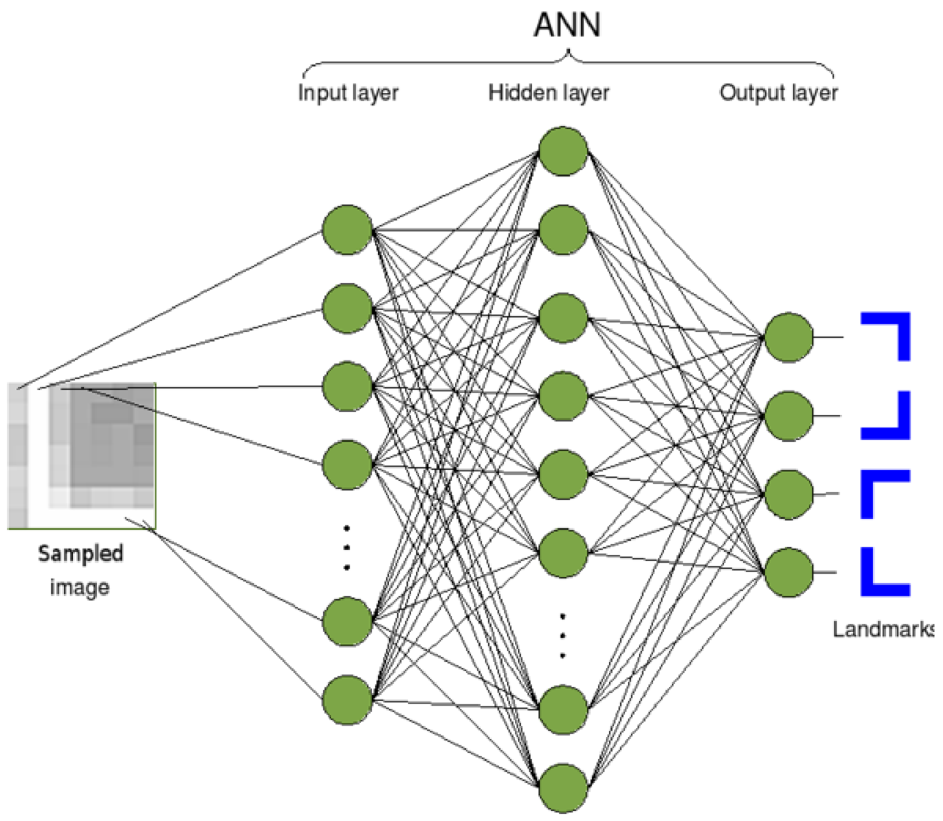


Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

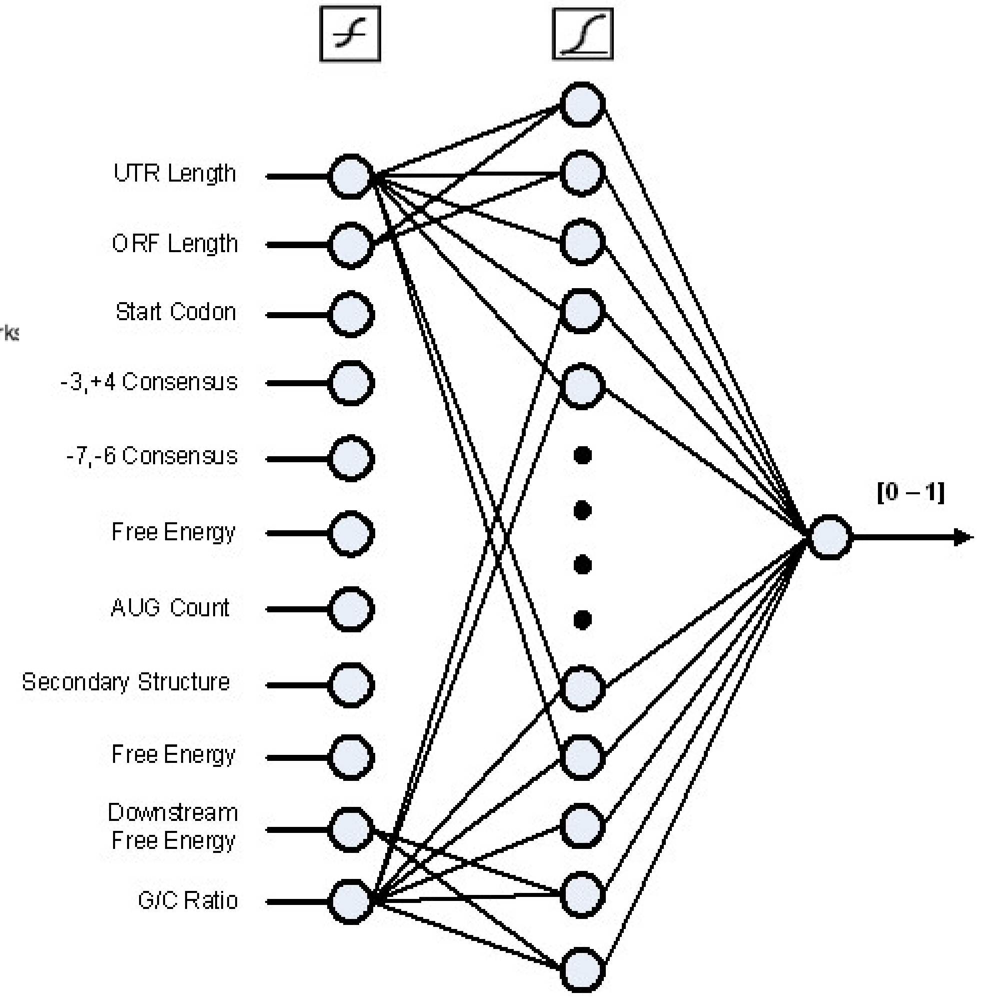
ANN discussion

- Very insensitive to noise
- ANNs are basically black box approach – unlike decision trees they do not provide a clue to how a prediction is made
- Difficult to incorporate prior biological knowledge
- Can also be used for clustering (unsupervised learning): self-organizing maps

Image recognition



Identification of alternative translation initiation sites



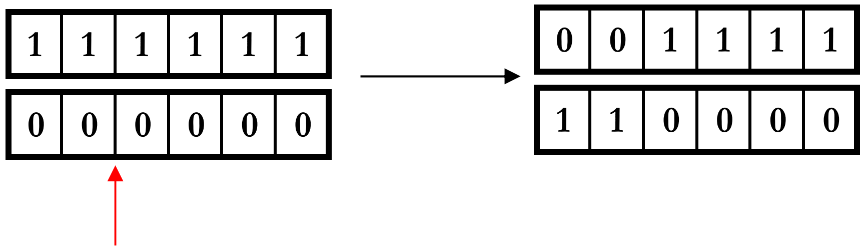
Genetic algorithms and programming

Genetic algorithms

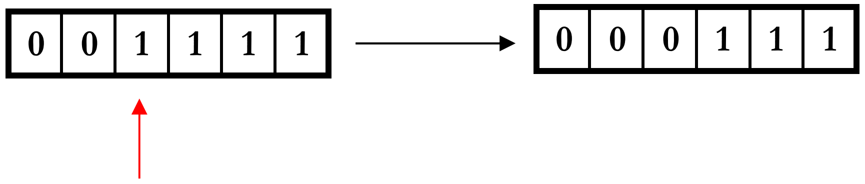
- Idea: use principles of evolution to discover better solutions to a problem given a random starting set of solutions
- Operators act on individuals (solutions) in the population (set of solutions) to yield a set of new solutions (next generation)
  - Reproduction
  - Selection
  - Crossover
  - Mutation
- Iteratively apply the operators to the population moving the algorithm from one generation to the next

Operators

- Cross over



- Mutation:

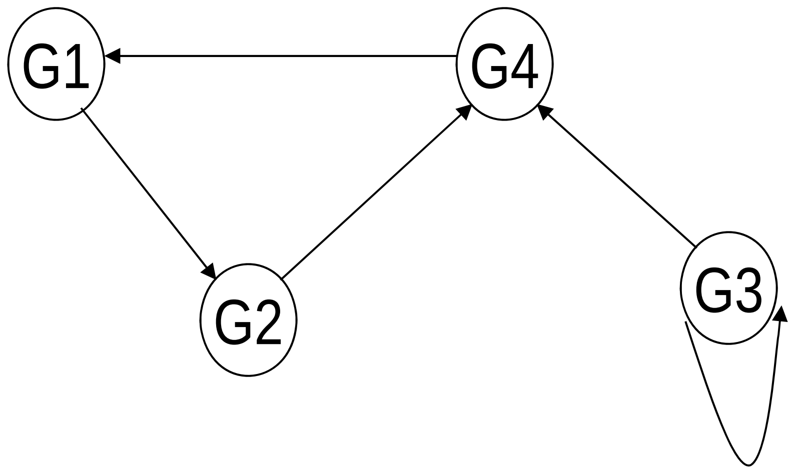


Algorithm

1. Generate a starting population randomly
2. Select two individuals from the current population randomly according to their fitness.
3. Combine the selected individuals using crossovers to form two new individuals
4. Mutate the two new individuals
5. Place the new indivudals in the next generation
6. Repeat 2-5 until the next generation is filled
7. Repeat 2-6 until no improvment is observed

Learn gene networks using GA

- Encode an individual as a matrix of gene interactions
- Fitness relates to how well the network describes microarray data



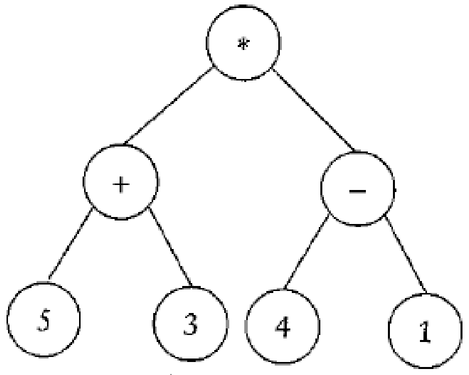
	G1	G2	G3	G4
G1	0	0	0	1
G2	1	0	0	0
G3	0	0	1	0
G4	0	1	1	0

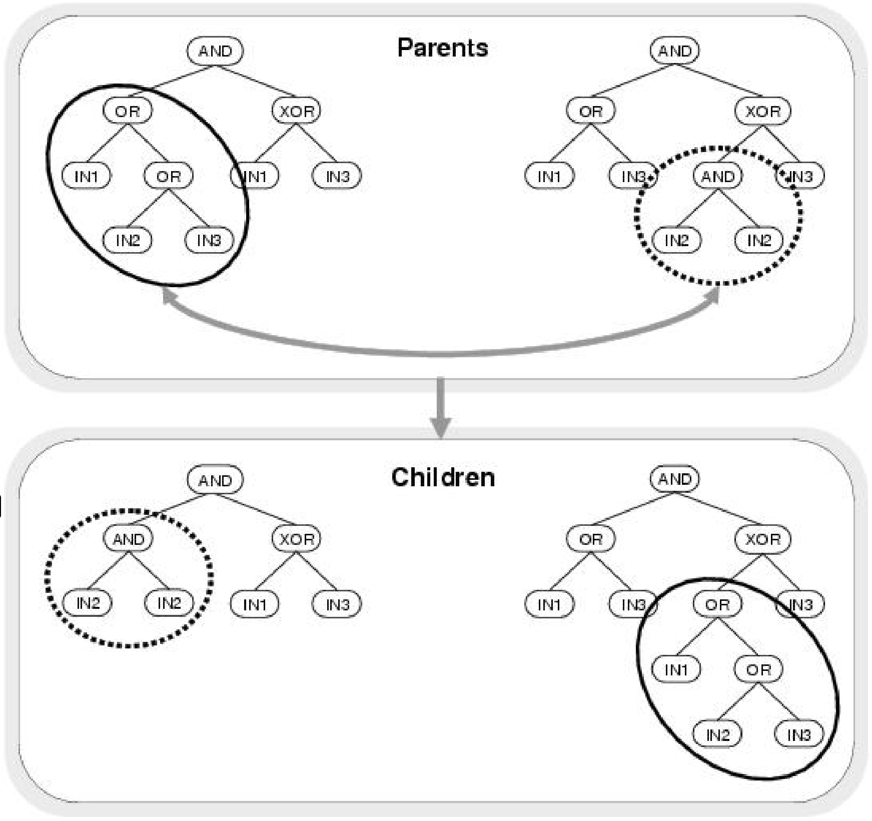
Genetic programming

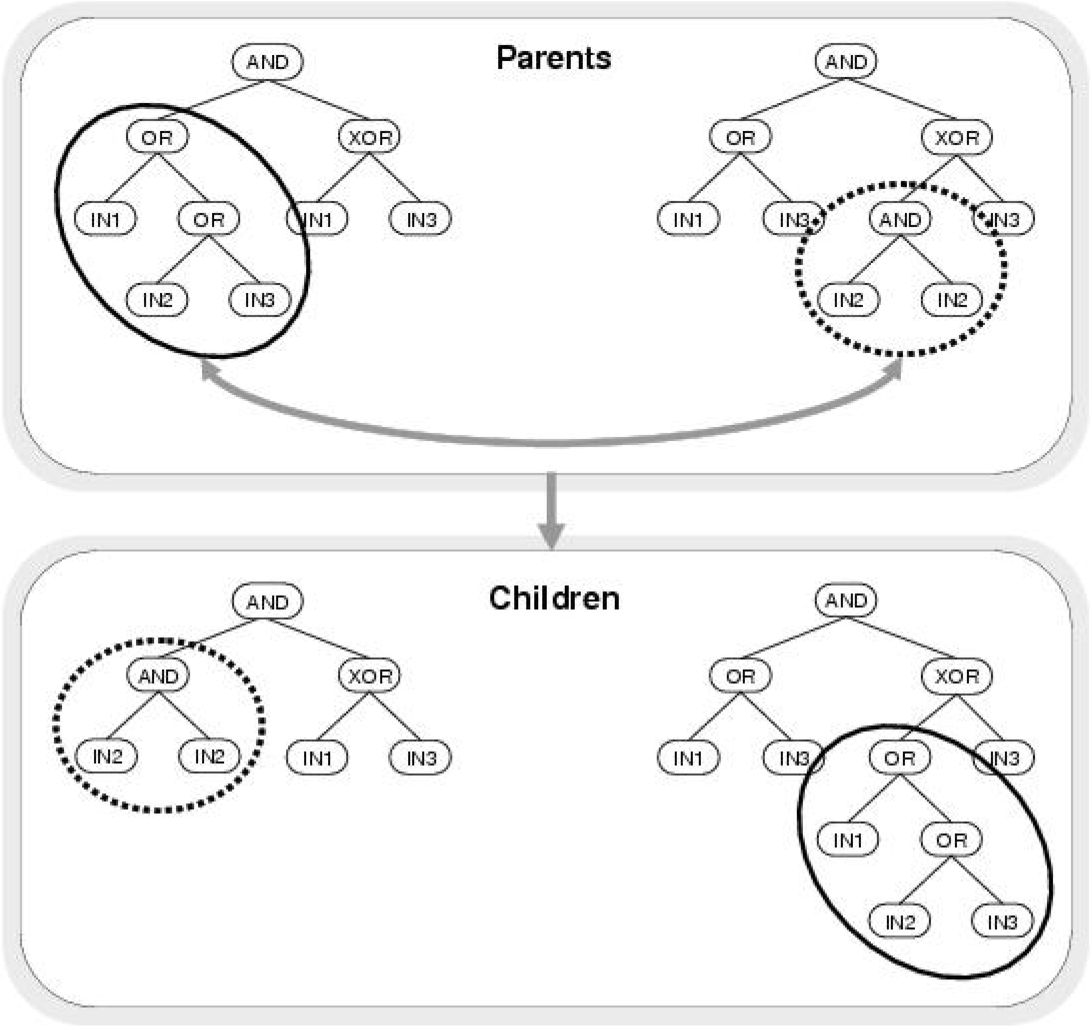
- One of the most recent techniques in AI
- Closely related to GA
- Solution is represented by a parse tree
- Originally designed for ‘automatic programming’
  - Method for computers programming themselves
- The programs they derive can be used to represent a range of functions which are based on the tree representation
  - E.g. a decision tree

Tree representation

- Terminals
  - Variables in a computer program
  - Constants
- Operators
  - Perform operations on terminals
  - Binary operators (+,-,*,…)
  - Unary operators (log10, exp, sqrt, …)






Parents
AND
OR
XOR
IN1
OR
IN1
IN3
IN2
IN3
AND
OR
XOR
IN1
IN3
AND
IN3
IN2
IN2
Children
AND
AND
XOR
IN2
IN2
IN1
IN3
AND
OR
XOR
IN1
IN3
OR
IN3
IN1
OR
IN2
IN3

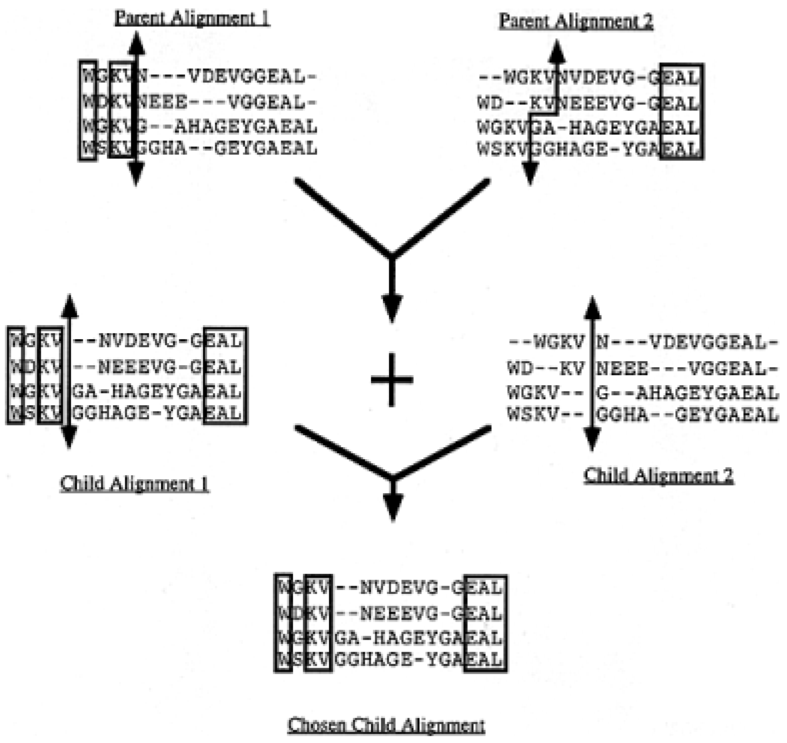
Application

- Machine learning:
  - Each tree is a decision tree or rule
  - Fitness is the classification accuracy of the tree
  - Examples: used in drug discovery and functional genomics
- Learning computer programs
  - Fitness is to what degree the program can be ran on a computer and produce the desired output

Limitations of GA and GP

- The problem most lend iteself to a genetic representation
- All solutions should be valid

SAGA: Sequence Alignment by Genetic Algorithm



Final remarks

Tools

- R Machine Learning:
  http://cran.r-project.org/web/views/MachineLearning.html
- mloss: Machine Learning Open Source Software
  http://mloss.org/about/
- RapidMiner:
  http://rapid-i.com/content/view/181/190/
- …

Summary

- Machine learning allows models with predictive and descriptive capabilities to be induced from examples

- Evaluation: training set, test set, cross validation, …

- Different approaches have different strengths and weaknesses
  - Linear versus non-linear
  - Interpretable versus black box
  - Regression versus classification

Summary

- Overfitting: you select a model A over a model B when A performs better on the training set, but worse on the unseen test set
  - Stop before overfitting occurs (e.g. before the decision tree is to long or when the performance of the neural network no longer improves)
  - Occam's razor: Select the simplest model that explains the data (do not use non-linear methods on a linearly separable problem)
- Course of dimensionality
  - Rule of thumb: You need more observations than features
  - Use dimensionality reduction methods (e.g. PCA) or feature selection (on the training set!)