

Machine Learning Models For Lung Cancer Classification Using Array Comparative Genomic Hybridization

D. Hardin Ph.D.¹ , C.F. Aliferis M.D., Ph.D.²

¹Department of Mathematics,

**²Discovery Systems Laboratory, Department of Biomedical Informatics,
Vanderbilt University**

5-4-2002

Overview:Goals

- *Goals:* Array comparative genomic hybridization (array CGH) is a recently introduced technology that measures gene copy number changes of hundreds of genes in a single experiment. The primary goal of this study was to determine whether we could use machine learning methods to create models that classify non-small Lung Cancers according to histopathology types. A second goal was to compare several machine learning methods in this learning task.

Overview: Methods

- **Methods:** DNA from tumors of 37 patients (21 squamous carcinomas, and 16 adenocarcinomas) were extracted and hybridized onto a 452 BAC clone array (printed in quadruplicate) carrying genes of potential importance in cancer. The following machine learning algorithms were run to create classification models: KNN, Decision Tree Induction, Linear and non-linear SVMs, and Feed-Forward Neural Networks. Performance was measured as classification accuracy using a leave-one-out methodology.

Overview: Results & Implications

- **Results:** The best multi-gene model found had a leave-one-out accuracy of 89.2%. Decision Trees performed poorer than the other methods in this learning task and dataset.
- **Implications:** Gene copy numbers as measured by array CGH are, *collectively*, an excellent indicator of histological subtype. Several interesting research directions are discussed.

What is *Machine Learning* (ML)? How is it different than *Statistics* and *Data Mining*?

- Machine Learning is the branch of Computer Science (Artificial Intelligence in particular) that studies systems that learn.
- Systems that learn = systems that improve their performance in some problem solving tasks without having been programmed by humans how to solve the tasks.
- ML has practically replaced Knowledge Acquisition for building Decision Support (“Expert”) Systems.

What is *Machine Learning* (ML)? How is it different than *Statistics* and *Data Mining*?

- Typical tasks:
 - ✓ image recognition,
 - ✓ medical diagnosis, medical prognosis, treatment recommendation,
 - ✓ elicitation of possible causal structure of problem domain,
 - ✓ game playing,
 - ✓ various optimization tasks,
 - ✓ prediction of structure or function of biomolecules,
 - ✓ text categorization,
 - ✓ identification of relevant variables, etc.

Indicative Example applications of ML IN Biomedicine

1. Bioinformatics

- Prediction of Protein Secondary Structure
- Prediction of Signal Peptides
- Gene Finding and Intron/Exon Splice Site Prediction
- Diagnosis using cDNA and oligonucleotide array gene expression data
- Identification of molecular subtypes of patients with various forms of cancer

2. Clinical problem areas

- Survival after Pneumonia (CAP)
- Survival after Syncope
- Diagnosis of Acute M.I.
- Diagnosis of Prostate Cancer
- Diagnosis of Breast Cancer
- Prescription and monitoring in hemodialysis
- Prediction of renal transplant graft failure

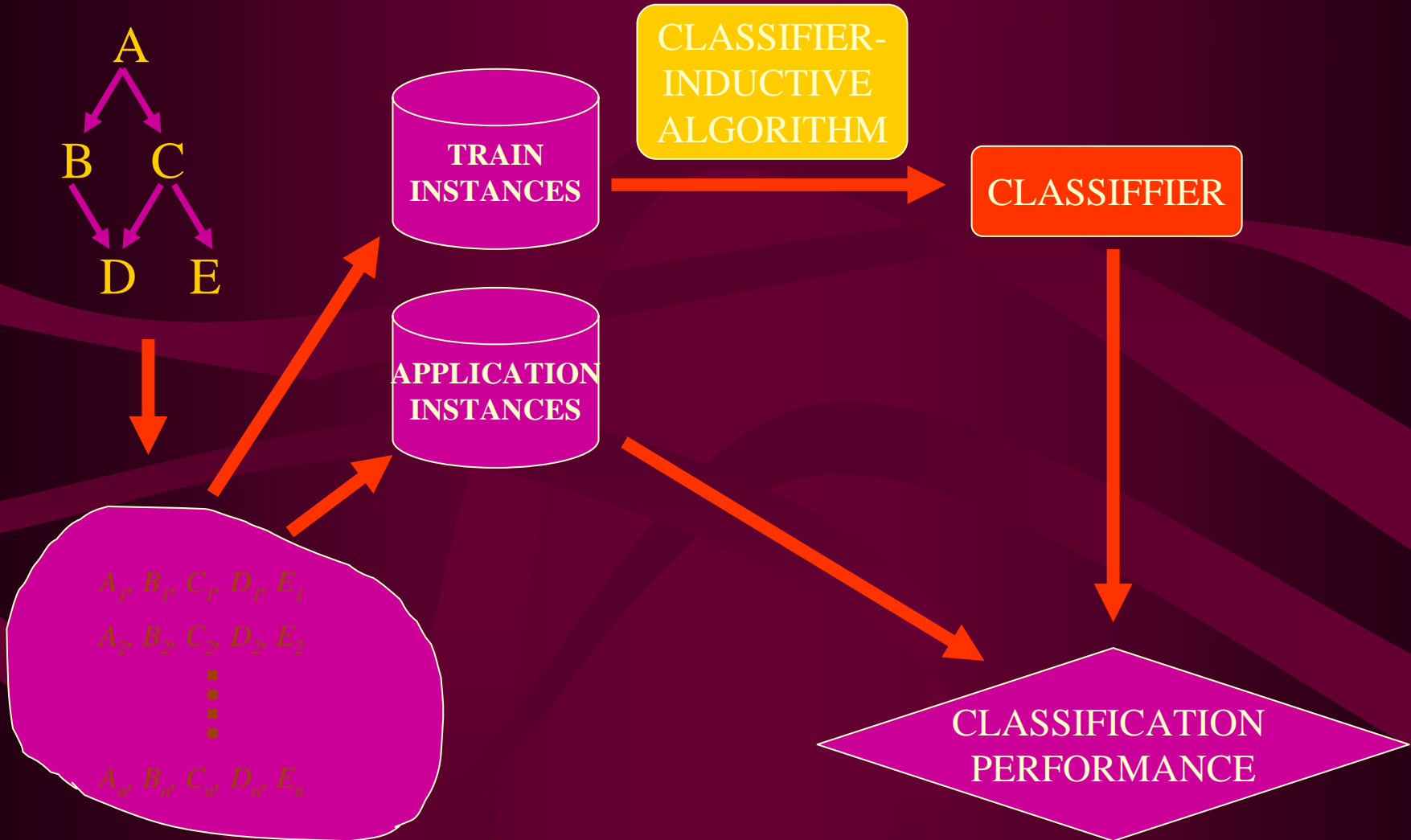
What is *Machine Learning* (ML)? How is it different than *Statistics* and *Data Mining*?

- Broadly speaking ML, DM, and Statistics have similar goals (modeling for classification and hypothesis generation or testing).
- **Statistics** has traditionally emphasized models that can be solved analytically (for example various versions of the Generalized Linear Model – GLM). To achieve this both restrictions in the expressive power of models and their parametric distributions are heavily used.
- **Data Mining** emphasizes very large-scale data analysis, trading off soundness and/or completeness for problem size.
- **Machine Learning** seeks to use computationally powerful approaches to learn very complex non- or quasi-parametric models of the data. Some of these models are closer to human representations of the problem domain per se (or of problem solving in the domain)

What is the difference between *supervised* and *unsupervised* ML methods?

- Supervised learning:
 - Give to the learning algorithm several instances of input-output pairs; the algorithm learns to predict the correct output that corresponds to some inputs (not only previously seen but also previously *unseen* ones (“generalization”)).
 - Example: show to learning algorithm patient cases (i.e., findings vector and a correct diagnosis for each case); then the algorithm induces a classifier that can classify a previously unseen patient to the correct diagnostic category given the findings observed in that patient)

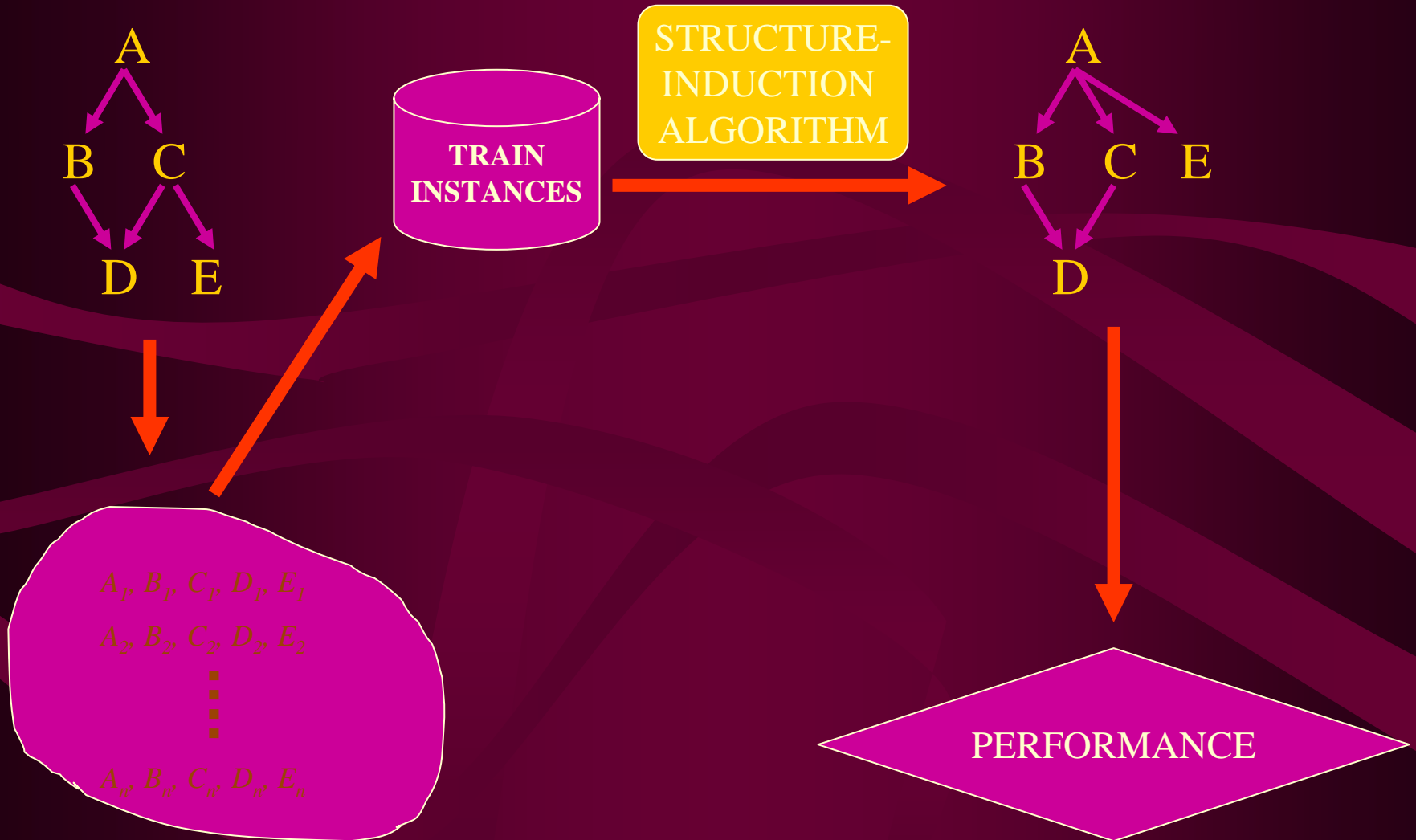
Classification



What is the difference between *supervised* and *unsupervised* ML methods?

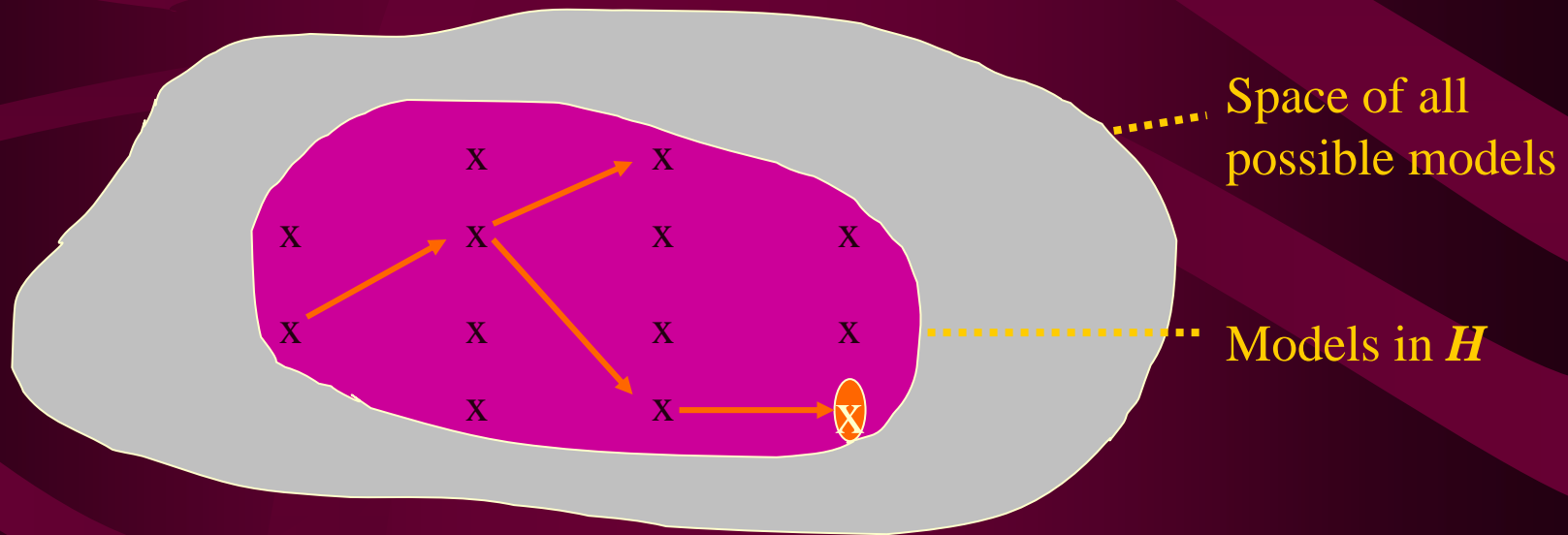
- Unsupervised learning:
 - Discover the categories (or other structural properties of the domain)
 - Example: give the learning algorithm gene expression measurements of patients with Lung Cancer; the algorithm finds sub-types (“molecular profiles”) of patients that are very similar to each other, and different to the rest of the types. Or another algorithm may discover how various genes interact among themselves to determine development of cancer.

Discovery



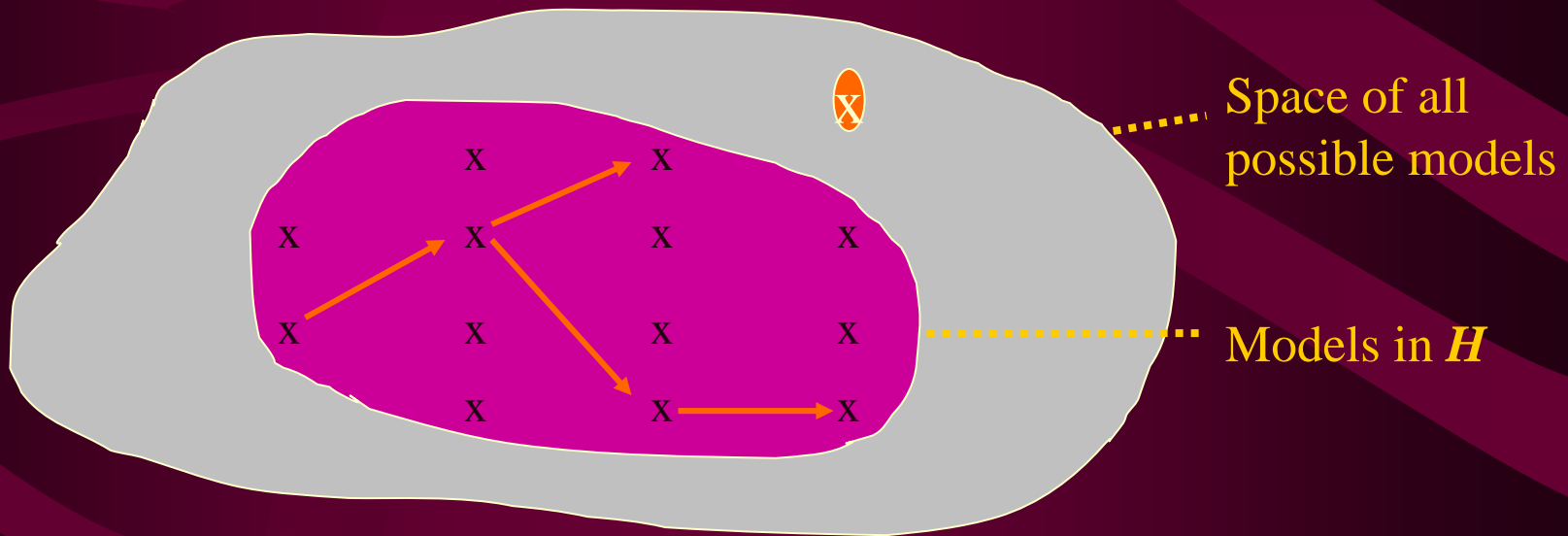
What is the theoretical basis of supervised Inductive ML?

- Inductive Machine Learning algorithms can be designed and analyzed using the following framework:
 - A language L in which we express models. The set of all possible models expressible in L constitutes our hypothesis space H
 - A scoring metric M tells us how good is a particular model
 - A search procedure S helps us identify the best model in H



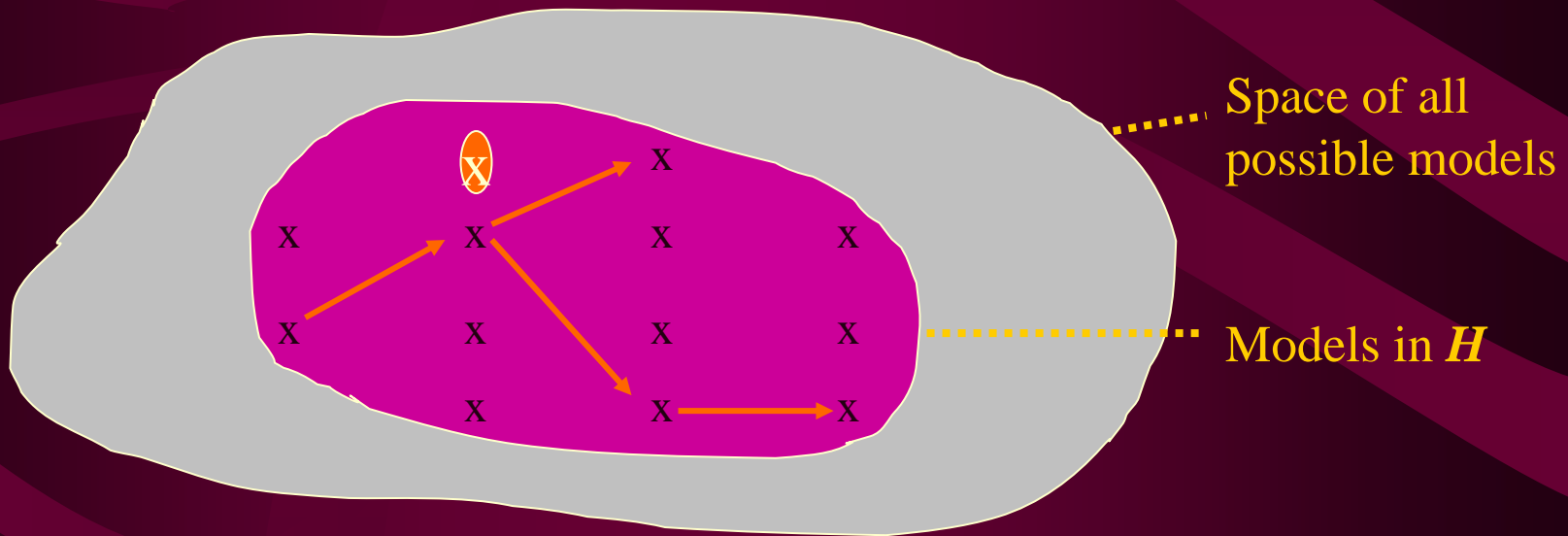
How can ML methods fail?

- Wrong language Bias: best model is not in H
- Example: we look for linear models, and the domain is non-linear



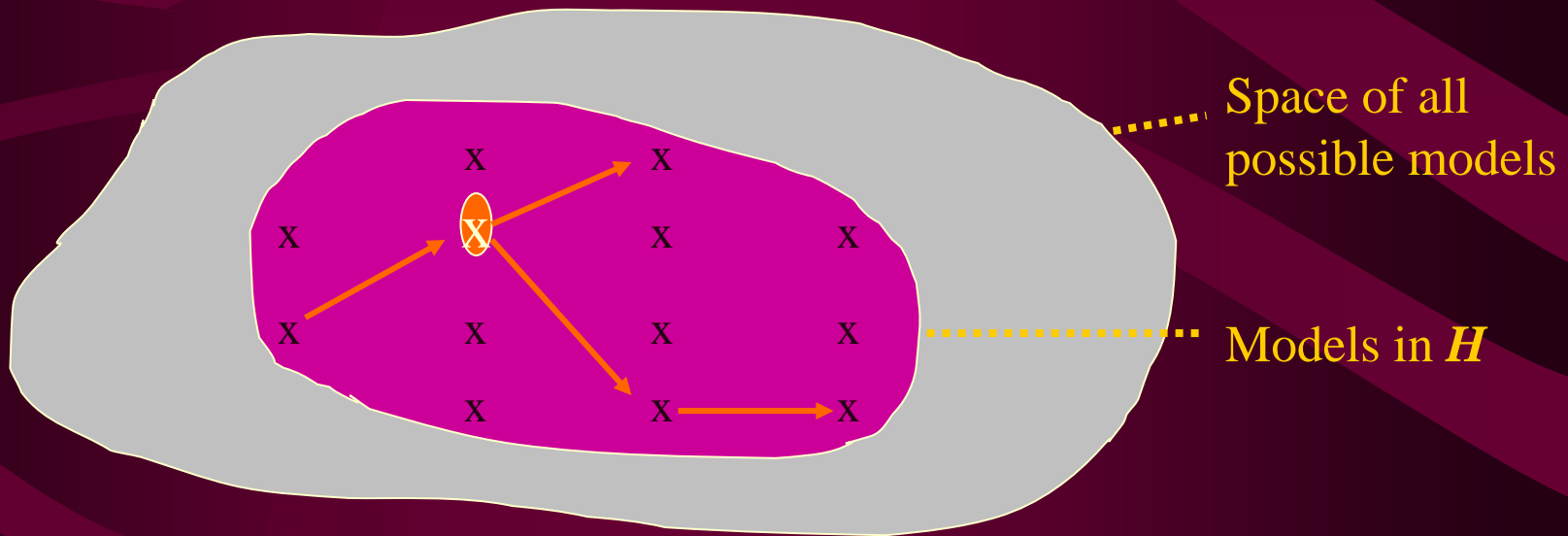
How can ML methods fail?

- Search Failure: best model is in H but search fails to examine it
- Example we use a steepest-ascent search in a multi-modal fitness landscape



How can ML methods fail?

- Metric Failure: best model is in H , search finds it but is deemed to be worse than an inferior model
- Example: Use strength of univariate association to infer plausible causation

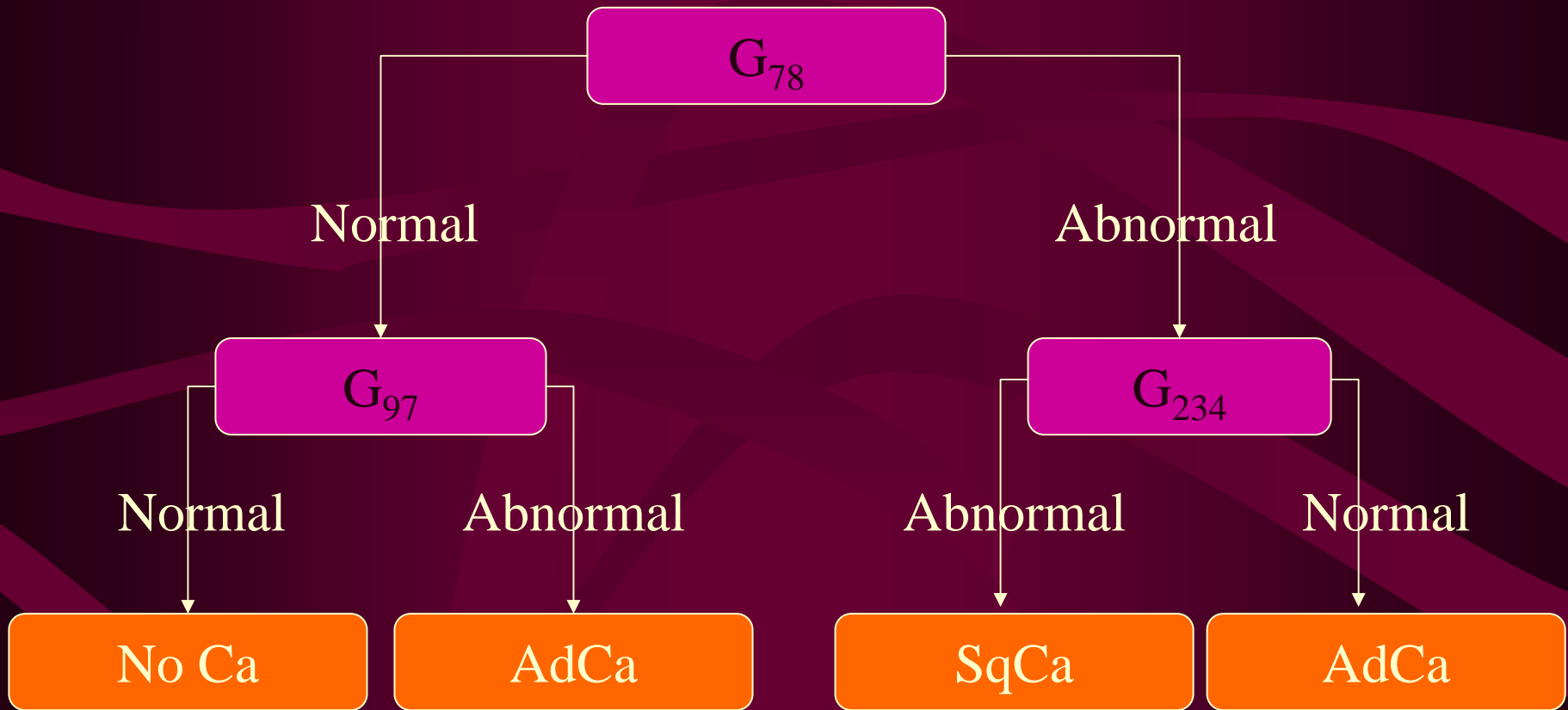


Prominent ML Methods

- KNN
 - Support Vector Machines
 - Variable Selection Algorithms
 - Decision Trees
 - Artificial Neural Networks
-
- Clustering
 - Bayesian Networks
 - Genetic Algorithms

Machine Learning Methods: Decision Tree Induction

- An example decision tree to solve the problem of classifying patient cases



Machine Learning Methods: Decision Tree Induction

- The Decision Tree can be, alternatively, thought as a collection of rules:

R1: $G_{78} = \text{normal}, G_{97} = \text{normal} \rightarrow \text{Decision} = \text{No Ca}$

R2: $G_{78} = \text{normal}, G_{97} = \text{abnormal} \rightarrow \text{Decision} = \text{Ad Ca}$

R3: $G_{78} = \text{abnormal}, G_{234} = \text{abnormal} \rightarrow \text{Decision} = \text{Sq Ca}$

R4: $G_{78} = \text{abnormal}, G_{234} = \text{normal} \rightarrow \text{Decision} = \text{Ad Ca}$

Machine Learning Methods: Decision Tree Induction

- The Decision Tree can yet be thought of as *concept learning*: For example the concept “Ad Ca” is the disjunction of the following conjunctions:

$(G_{78} = \textit{normal} \textbf{ and } G_{97} = \textit{abnormal}) \textbf{ or}$

$(G_{78} = \textit{abnormal} \textbf{ and } G_{234} = \textit{normal})$

Machine Learning Methods: Decision Tree Induction

- Decision Trees can be automatically learned from data using either information-theoretic criteria or a measure of classification performance.
- The induction procedure is very simple in principle:

1. Start with an empty tree
2. Put at the root of the tree the variable that best classifies the training examples
3. Create branches under the variable corresponding to its values
4. Under each branch repeat the process with the remaining variables

Notes:

- “best classifies” can be determined on the basis of maximizing homogeneity of outcome in the resulting subgroups, cross-validated accuracy, best-fit of some linear regressor, etc.

Machine Learning Methods: Decision Tree Induction

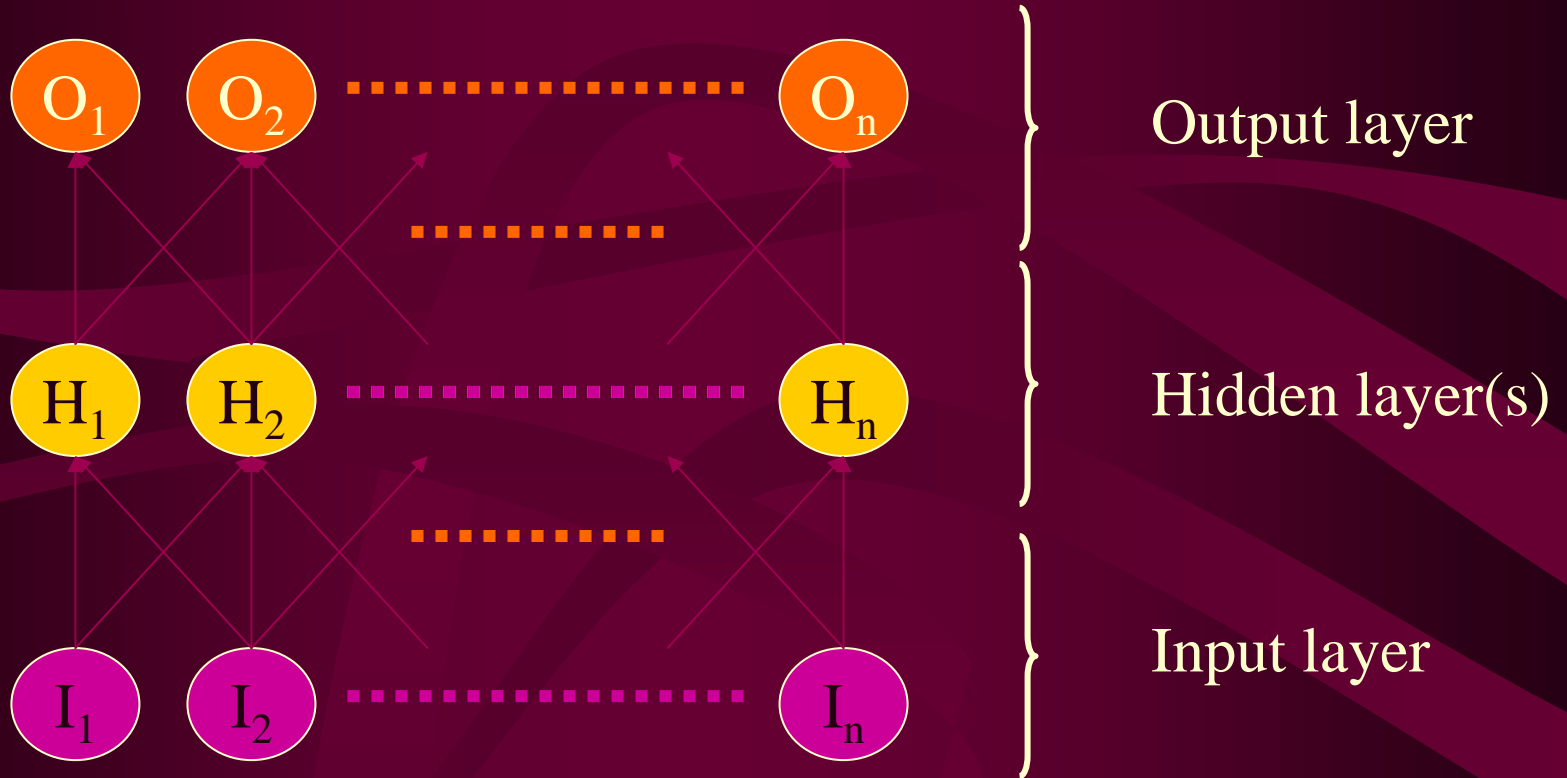
Notes (CNT'D):

- o DTI is best for:
 - Discrete domains
 - Target function has discrete outputs
 - Disjunctive/conjunctive descriptions required
 - Training data may be noisy
 - Training data may have missing values
- o DTI can represent any finite discrete-valued function
- o Extensions for continuous variables do exist
- o Search is clearly greedy and thus can be trapped in local minima
- o DTI is very sensitive to high feature-to-sample ratios; when many features contribute a little to classification DTI does not do well
- o DT models are highly intuitive, and easy to explain and use even without computing equipment available

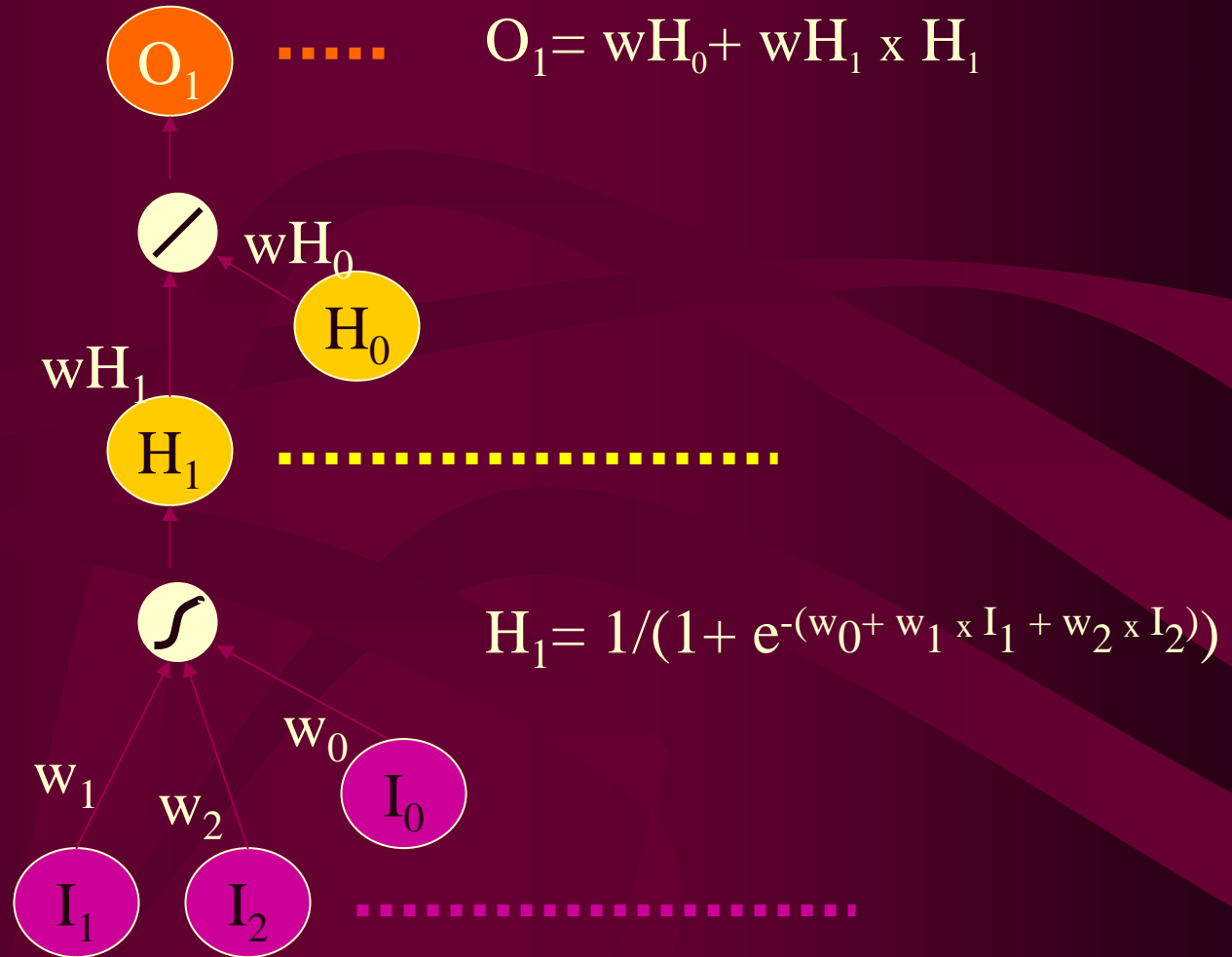
Machine Learning Methods: Neural Networks

- A very large, family of supervised and unsupervised learning methods motivated by the way biological neural systems are organized.
- Among the most effective and robust learning methods yet devised
- They have excellent documented results in a wide variety of problem areas including: image recognition, text categorization, medical diagnosis, structural genomic and proteomic classification, dynamic control, drug-dosage adjustment, cluster discovery, speech recognition, etc.
- On the other hand it is almost impossible to understand why a network performs the way it does (i.e., *what* it has learned) and *why* it gives the output that it does.
- We used here multi-layer feed-forward (acyclic) networks trained with gradient descent (Backpropagation algorithm)

Machine Learning Methods: Neural Networks



Machine Learning Methods: Neural Networks



Machine Learning Methods: Neural Networks

- Backpropagation algorithm:

1. Initialize all weights to small random numbers
2. Repeat
3. Propagate input forward:
 - a. Compute output of every unit given an input vector
4. Propagate errors backward:
 - a. Calculate error for each output unit
 - b. Calculate error for each hidden layer unit
 - c. Update each network weight
5. Until error small enough or other stopping criterion (independent test set error, number of train iterations, etc.)

Machine Learning Methods: Neural Networks

- Notes:
 - The choice of a smooth transfer function allows differentiation of the error function and thus derivation of an easy to compute stochastic gradient descent optimization rule employed in BP
 - BP may be trapped in local minima; in practice this is not a major problem (because of stochastic gradient descent, momentum, and very high-dimensional spaces)
 - A ANN with three layers of units can represent any function to arbitrary accuracy
 - ANNs have very good generalization behavior
 - Overfitting can be avoided by empirical (generalization error, weight decay) or theoretical means

Results

| METHOD | GENES | PARAMETERS & OBSERVATIONS | LEAVE-ONE-OUT ACCURACY (%) |
|--------------------------------|------------|--|-------------------------------------|
| Decision Tree Induction | 388 | default | 56.8 |
| KNN | 388 | k=1,2, 3 | 86.5, 75.7, 83.7 |
| Linear SVM | 388 | - | 83.8 |
| Polynomial-kernel SVM | 388 | degree=2 , cost=1, 10, 100, 1000 | 78.4, 83.8, 83.8, 83.8 |
| | | degree=3 , cost=1, 10, 100, 1000 | 78.4, 83.8, 83.8, 83.8 |
| | | degree=4,5 , cost=1, 10, 100, 1000 | 83.8 |
| | | degree=6, cost=1, 10, 100, 1000 | 81.1 |
| RBF-kernel SVM | 388 | gamma=1, 0.1, 0.05, 0.01, 0.001 | 62.2, 83.8, 81.1, 81.1, 56.8 |
| NNs | 388 | 500 epochs, 5 hidden units, variable learning rate (optimised separately –see text) | 83.8 |

Results

| METHOD | GENES | PARAMETERS & OBSERVATIONS | LOO ACCURACY (%) |
|-------------------|------------------|--|---------------------------------|
| DTI | 8, 15, 50 | 8, 15, 50 best univariate predictors | 70.3 |
| | 8, 15, 50 | 8, 15, 50 best univariate predictors + boosting | 78.4 |
| Linear SVM | 80 | 80 best genes according to weights in linear SVM trained with all genes | 89.2 |
| NN | | 388 genes → 1 to 36 Principal Components (27.7-100 % of variance explained) | 64.8(min)- 83.8(max) |

Conclusions

- The experiments presented here support the hypothesis that gene copy numbers as measured by array CGH are, *collectively*, **an excellent indicator of the histological subtype**.
- Gene copy number is a **more stable property** of cells than gene expression levels or protein concentrations. As such, array CGH has the potential to offer valuable complementary information to, for instance, cDNA array assays or MALDI mass-spectrometry measurements.
- An interesting next research direction is, therefore, **to combine all three types of data** together with clinical and traditional histopathology information and investigate its association to important clinical outcomes (such as response to treatment and prognosis).

Conclusions

- Other interesting directions:
 - **Discover possible new cancer classes** on the basis of molecular information. Such classes may carry more important information, clinically, than histopathology.
 - Analyze cases that were misclassified
 - Repeat analysis by removing strongest univariate predictor sets
 - Evaluate the benefits of feature selection methods (such as our recently developed Markov Blanket-based approaches)
 - Investigate effects of de-noising
 - Investigate methods for analytically removing effects of distribution mix (i.e., contamination of cancer samples with normal tissue due to biopsy technique)

Conclusions

- From a **computational perspective**, bio-informatics datasets, such as the one studied here, challenge the limits of the state-of-the-art applied machine learning methods in several ways:
 - Small sample size increases the danger of overfitting model parameters to the data and makes discovery of complex/causal models very difficult
 - High rates of missing values compromise the learning ability of machine learning algorithms,
 - Very large variable-to-sample ratios
 - With the exception of DTI, which was particularly vulnerable to the characteristics of the data, there were no clear “winners” among the algorithms tested.

Clustering

- Unsupervised class of methods
- Basic idea: group similar items together and different items apart
- Countless variations:
 - of what constitutes “similarity” (may be distance in feature space, may be other measures of association),
 - of what will be clustered (patients, features, time series, cell-lines, combinations thereof, etc.)
 - of whether clusters are “hard” (no multi-membership) or “fuzzy”
 - of how clusters will be build and organized (partitional, agglomerative, non-hierarchical methods)
- Uses:
 - Taxonomy (e.g., identify molecular subtypes of disease)
 - Classification (e.g., classify patients according to genomic information)
 - Hypothesis generation (e.g., if genes are highly “co-expressed” then this may suggest they are in same pathway)

Clustering

- **K-means clustering**: We want to partition the data into k most-similar groups

1. Choose k cluster centers (“centroids”) to coincide with k randomly chosen patterns (or arbitrarily-chosen points in the pattern space)
2. Repeat
3. Assign each pattern in data to cluster with the closest centroid
4. Recompute new centroids
5. Until convergence (i.e., few or no re-assignments or small decrease in error function such as total sum of squared errors of each pattern in a cluster from centroid of that cluster)

Variations:

- selection of good initial partitions
- Allow splitting/merging of resulting clusters
- Various similarity measures and convergence criteria

Clustering (k-means)

e.g., ($K=2$)

A B
2 3

C D
9 10

E F
11 12

Step 1: (arbitrarily)

[A B

C D]

[E F]

Centroid1=6, centroid2=11.5

Step 2:

[A B]

[C D

E F]

Centroid1=2.5, centroid2=10.25

----- (algorithm stops) -----

Clustering

Agglomerative Single Link:

1. Start with each pattern belonging to its own cluster
2. Repeat
3. Join these two clusters that have the smallest pair-wise distance
4. Until all patterns are in one cluster

Note:

- Inter-cluster distance between clusters A and B is computed as the minimum distance of all pattern pairs (a,b) s.t. a belongs to A and b to B

Clustering (ASL)

e.g.,

| | | |
|-----|-----|-------|
| A B | C D | E F |
| 1 2 | 5 7 | 11 12 |

Step 1: [A] [B] [C] [D] [E] [F] smallest distance [A] [B]=1 OR [E] [F]=1

Step 2: [A B] [C] [D] [E] [F] smallest distance [E] [F]=1

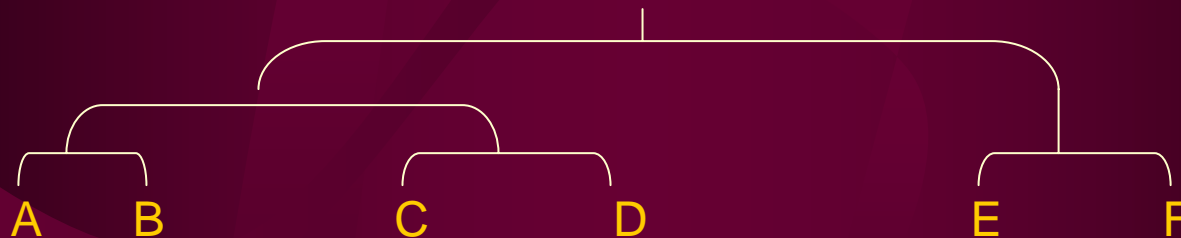
Step 3: [A B] [C] [D] [E F] smallest distance [C] [D]=2

Step 4: [A B] [C D] [E F] smallest distance [A B] [C D]=3

Step 5: [A B C D] [E F] smallest distance [A B C D] [E F]=4

Step 6: [A B C D E F] -----(algorithm stops)-----

Schematic representation via the “dendrogram”:



Clustering

Agglomerative Complete Link:

1. Start with each pattern belonging to its own cluster
2. Repeat
3. Join these two clusters that have the smallest pair-wise distance
4. Until all patterns are in one cluster

Note:

- Inter-cluster distance between clusters A and B is computed as the maximum distance of all pattern pairs (a,b) s.t. a belongs to A and b to B

Clustering (ACL)

e.g.,

| | | | | | |
|---|---|---|---|----|----|
| A | B | C | D | E | F |
| 1 | 2 | 5 | 7 | 11 | 12 |

Step 1: [A] [B] [C] [D] [E] [F] smallest distance [A] [B]=1 OR [E] [F]=1

Step 2: [A B] [C] [D] [E] [F] smallest distance [E] [F]=1

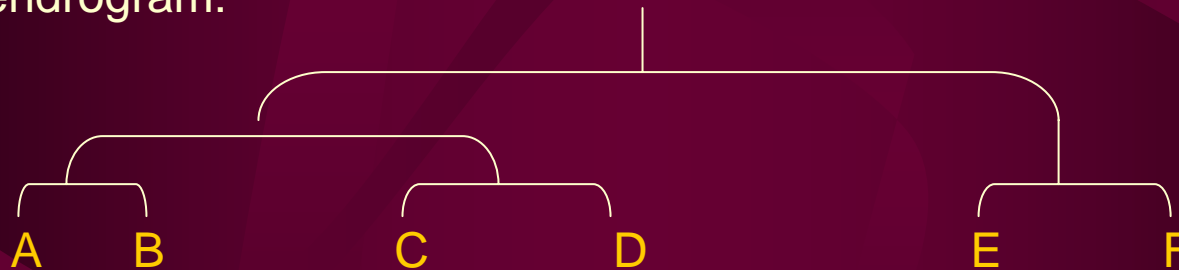
Step 3: [A B] [C] [D] [E] [F] smallest distance [C] [D]=2

Step 4: [A B] [C D] [E] [F] smallest distance [A B] [C D]=6

Step 5: [A B C D] [E] [F] smallest distance [A B C D] [E F]=11

Step 6: [A B C D E F] -----(algorithm stops)-----

With dendrogram:



Genetic Algorithms

- Evolutionary Computation (Genetic Algorithms & Genetic Programming) is motivated by the success of evolution as a robust method for adaptation found in nature
- The main algorithm is very simple:

1. Generate randomly a population P of p hypotheses
2. Compute the fitness of each member of P , h_i
3. Repeat
 - a. Create a random sample P_s from P by choosing each h_i with probability proportional to the relative fitness of h_i to the total fitness of all h_j
 - b. Augment P_s with cross-over offspring of the remaining hypotheses chosen with same probability as in step #4
 - c. Change members of P_s at random by bit-mutations
 - d. Replace P by P_s and compute new fitness of each member of P
4. Until enough generations have been created or a good enough hypothesis have been generated
5. Return best hypothesis

Genetic Algorithms

- The population size, cross-over rate, and mutation rate are parameters that are set empirically
- Representation of hypotheses in GAs is typically a bitstring so that the mutation and cross-over operations can be achieved easily.
- E.g., consider encoding clinical decision-making rules:

variable1: fever {yes, no}

variable2: x_ray {positive, negative}

variable3: diagnosis {flu, pneumonia}

Rule1: fever=yes and x_ray=positive → diagnosis=pneumonia

Rule2: fever=unknown and x_ray=unknown → diagnosis= flu or pneumonia

Bitstring representation:

R1: 1 0 1 0 0 1

R2: 0 0 0 0 1 1

(note: we can constrain this representation by using less bits, the fitness function, and syntactic checks)

Genetic Algorithms

Let's cross-over these rules at a random point:

| | | | | |
|-----|-----|--|-----|-----|
| R1: | 1 0 | | 1 0 | 0 1 |
| R2: | 0 0 | | 0 0 | 1 1 |

Gives:

| | | | |
|------|-----|-----|-----|
| R1': | 1 0 | 0 0 | 1 1 |
| R2': | 0 0 | 1 0 | 0 1 |

And mutation at two random bits may give:

| | | | |
|-------|-----|-----|-----|
| R1'': | 1 0 | 0 1 | 1 1 |
| R2'': | 1 0 | 1 0 | 0 1 |

Which is interpreted as:

Rule1'': fever=yes and x_ray=negative → diagnosis=flu or pneumonia

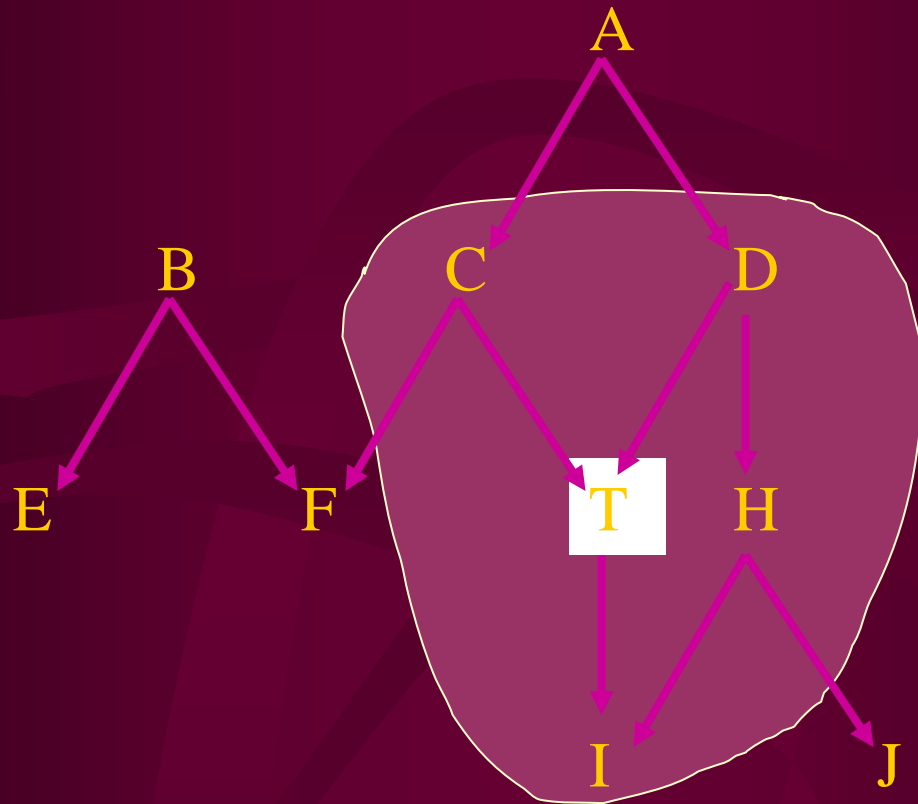
Rule2'': fever=yes and x_ray=positive → diagnosis=pneumonia

Genetic Algorithms

Notes:

- There exist many variations of how to do cross over, how to select hypotheses for mutation/cross-over, how to isolate subpopulations, etc.
- Although it may appear at first that the process of finding better hypotheses relies totally on chance, this is not the case. Several theoretical results (most famous one being the “Schema Theorem” prove that exponentially more short & better-fit hypotheses are being considered than worse-fit ones (to the number of generations).
- Furthermore, due to the discrete nature of optimization local minima will trap the algorithm less, but also it becomes more difficult to find the global optimum.
- It has been shown that GA perform an implicit parallel search in hypotheses templates without explicitly generating them (“Implicit Paralellism”).

Causal Probabilistic (Bayesian) Network Induction



Causal Probabilistic (Bayesian) Network Induction

- Causal Discovery with BNs: Assume faithfulness of the data-generating process to the data; assume causal sufficiency; assume data is random sample from all instances produced by the process. Then an algorithm that generates the correct causal network given the data is:

PC Algorithm (Outline)

Phase I: find direct edges by using the criterion that A has a direct edge to B iff for all subsets of features there is no subset S, s.t. independent(A, B | S).

Phase II: orient edges in “collider” triplets (i.e., of the type: A->B<-C) using the criterion that if there are direct edges between A, B and between B,C, but not between A, C, and there is no subset containing B s.t. independent (A,C |B) then A->B<-B

Phase III: enter a constraint-propagation loop for orienting edges further by adding orientations until no further orientations can be produced using the two following criteria: (a) if A->B...->C and A-C then A->C also, and (b) if A->B-C then B->C

K-Nearest Neighbors

- Say we want to predict outcome for a patient i that received treatment 1 and is of genotype class 2. KNN searches for the K most similar cases in the training data base (using Euclidean Distance or other similarity metric):

$$ED(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_k (x_{i,k} - x_{j,k})^2}$$

- For example patient #1 and the new patient have ED=

| Patient# | Treatment type | Genotype | Survival |
|----------|----------------|----------|----------|
| 1 | 1 | 1 | 1 |
| i | 1 | 2 | ? |

$$ED = \sqrt{((1-1)^2 + (1-2)^2)} = 1$$

K-Nearest Neighbors

- Similarly the distances of case i to all training cases are:

| Patient# | ED(Patient#, P_i) | Survival |
|----------|----------------------|----------|
| 1 | 1 | 1 |
| 2 | 0 | 2 |
| 3 | 1 | 1 |
| 4 | 0 | 2 |
| 5 | 1.4 | 2 |
| 6 | 1 | 1 |
| 7 | 1.4 | 2 |
| 8 | 1 | 1 |

- Now let's rank training cases according to distance to case i

K-Nearest Neighbors

| Patient# | ED(Patient#, P_i) | Survival |
|----------|----------------------|----------|
| 2 | 0 | 2 |
| 4 | 0 | 2 |
| 3 | 1 | 1 |
| 1 | 1 | 1 |
| 6 | 1 | 1 |
| 8 | 1 | 1 |
| 5 | 1.4 | 2 |
| 7 | 1.4 | 2 |

As we can see the training case most similar to i has outcome 2. The 2 training cases most similar to i have a median outcome 2. The 3 training cases most similar to i have a median outcome 2, and so on. We say that for $K=1$ the KNN predicted value is 2, for $K=2$ the predicted value is 2, and so on.

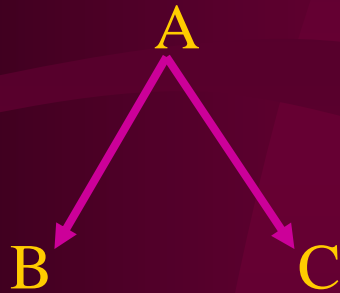
K-Nearest Neighbors

To summarize:

- KNN is based on a case-based reasoning framework.
- It has good asymptotic properties for $K > 1$.
- It is straightforward to implement (of course care has to be given to variable encoding, variable relevance, and distance metric); efficient encoding is not easy since it requires specialized data structures.
- It is used in practice as:
 - a baseline comparison for new methods
 - component algorithm for “wrapper” feature selection methods
 - Non-parametric density estimator

Causal Probabilistic (Bayesian) Network Induction

- Causal Probabilistic Networks provide a simple but very powerful language for representing causal processes
- Causal Bayesian Network = Graph (Variables (nodes), dependencies (arcs)) + Joint Probability Distribution + Markov Property
- Graph has to be DAG (directed acyclic) in the standard BN model



JPD

$$P(A+, B+, C+) = 0.006$$

$$P(A+, B+, C-) = 0.014$$

$$P(A+, B-, C+) = 0.054$$

$$P(A+, B-, C-) = 0.126$$

$$P(A-, B+, C+) = 0.240$$

$$P(A-, B+, C-) = 0.160$$

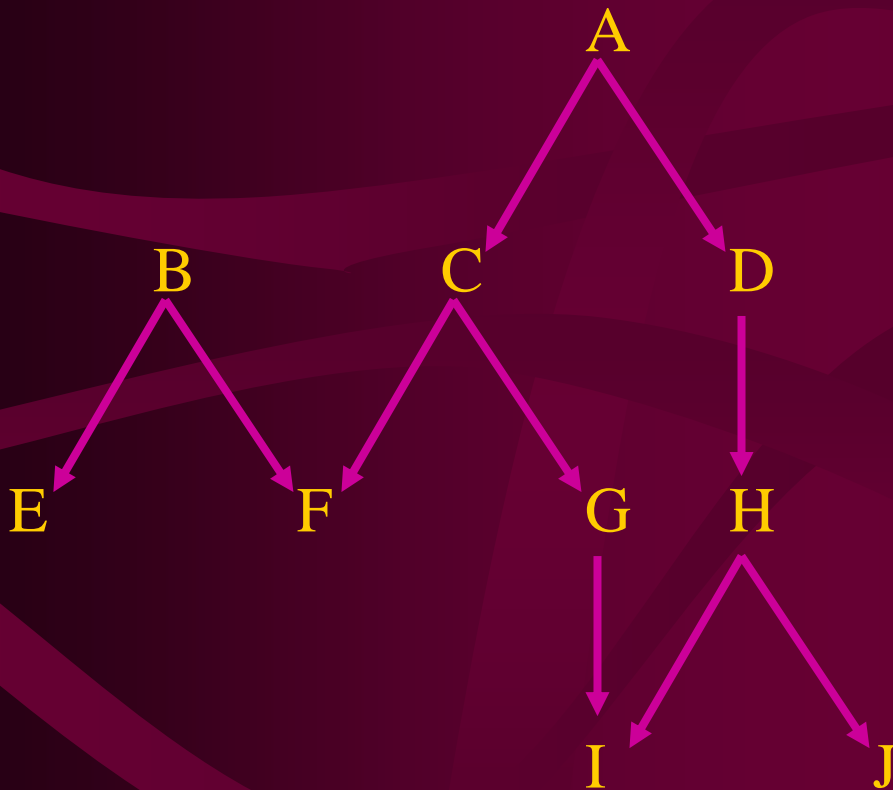
$$P(A-, B-, C+) = 0.240$$

$$P(A-, B-, C-) = 0.160$$

- Theorem 1 (Neapolitan): any JPD can be represented in BN form

Causal Probabilistic (Bayesian) Network Induction

- Markov Property: the probability distribution of any node N given its parents P is independent of any subset of the non-descendent nodes W of N



e.g., :

$$D \perp \{B, C, E, F, G \mid A\}$$

$$F \perp \{A, D, E, F, G, H, I, J \mid B, C\}$$

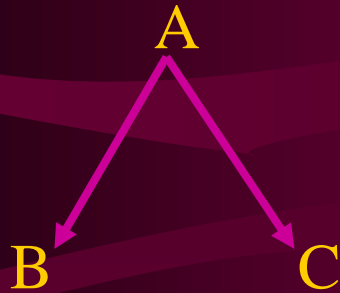
Causal Probabilistic (Bayesian) Network Induction

- Theorem 2 (Pearl): the Markov property enables us to decompose (factor) the joint probability distribution into a product of prior and conditional probability distributions

$$P(V_1, V_2, \dots, V_n) = \prod_i p(V_i | \text{Parents}(V_i))$$

Causal Probabilistic (Bayesian) Network Induction

- Theorem 3 (Pearl): A DAG and set of conditional probabilities of each node given its parents defines a BN with a unique and valid jpd.



$$P(\mathbf{V}) = \prod_i p(V_i | \text{Pa}(V_i))$$

The original JPD:

| |
|-------------------------|
| $P(A+, B+, C+) = 0.006$ |
| $P(A+, B+, C-) = 0.014$ |
| $P(A+, B-, C+) = 0.054$ |
| $P(A+, B-, C-) = 0.126$ |
| $P(A-, B+, C+) = 0.240$ |
| $P(A-, B+, C-) = 0.160$ |
| $P(A-, B-, C+) = 0.240$ |
| $P(A-, B-, C-) = 0.160$ |

Becomes :

| |
|--------------------|
| $P(A+) = 0.8$ |
| $P(B+ A+) = 0.1$ |
| $P(B+ A-) = 0.5$ |
| $P(C+ A+) = 0.3$ |
| $P(C+ A-) = 0.6$ |

**Up to
Exponential
Saving in
Number of
Parameters!**

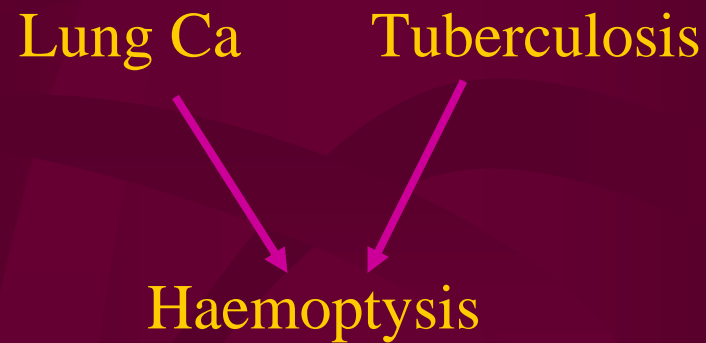
Causal Probabilistic (Bayesian) Network Induction

- The Markov property captures causality:
 - Confounders



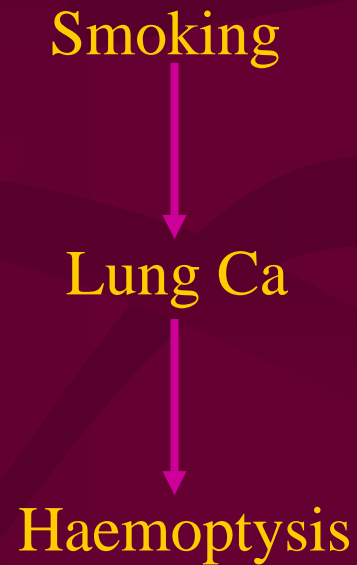
Causal Probabilistic (Bayesian) Network Induction

- The Markov property captures causality:
 - Modeling “explaining away”
 - Modeling/understanding selection bias



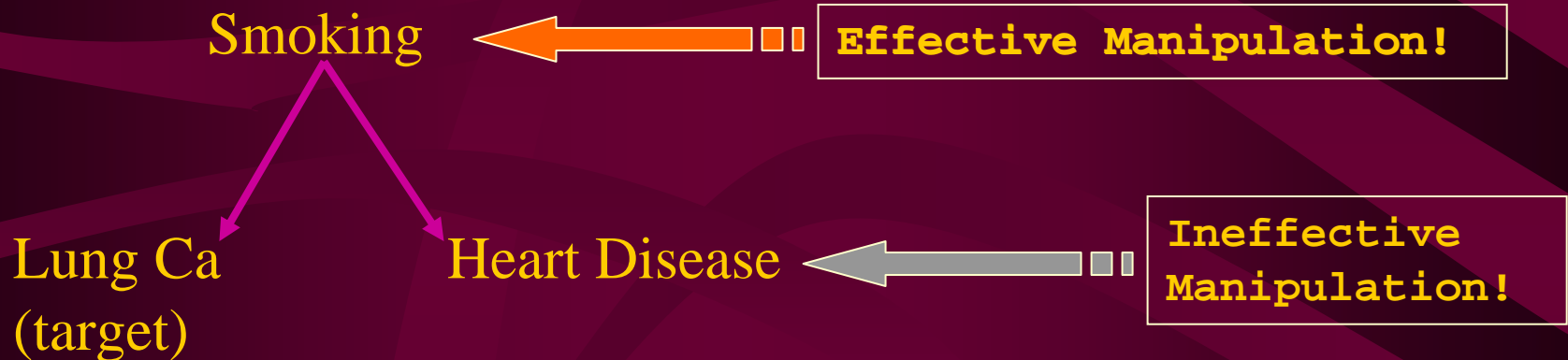
Causal Probabilistic (Bayesian) Network Induction

- The Markov property captures causality:
 - Modeling causal pathways



Causal Probabilistic (Bayesian) Network Induction

- The Markov property captures causality:
 - Manipulation in the presence of confounders



Causal Probabilistic (Bayesian) Network Induction

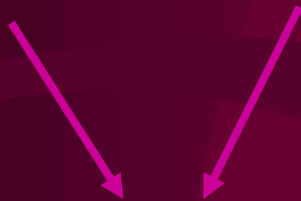
- The Markov property captures causality:
 - Manipulation in the presence of selection bias

Lung Ca (target)

Tuberculosis

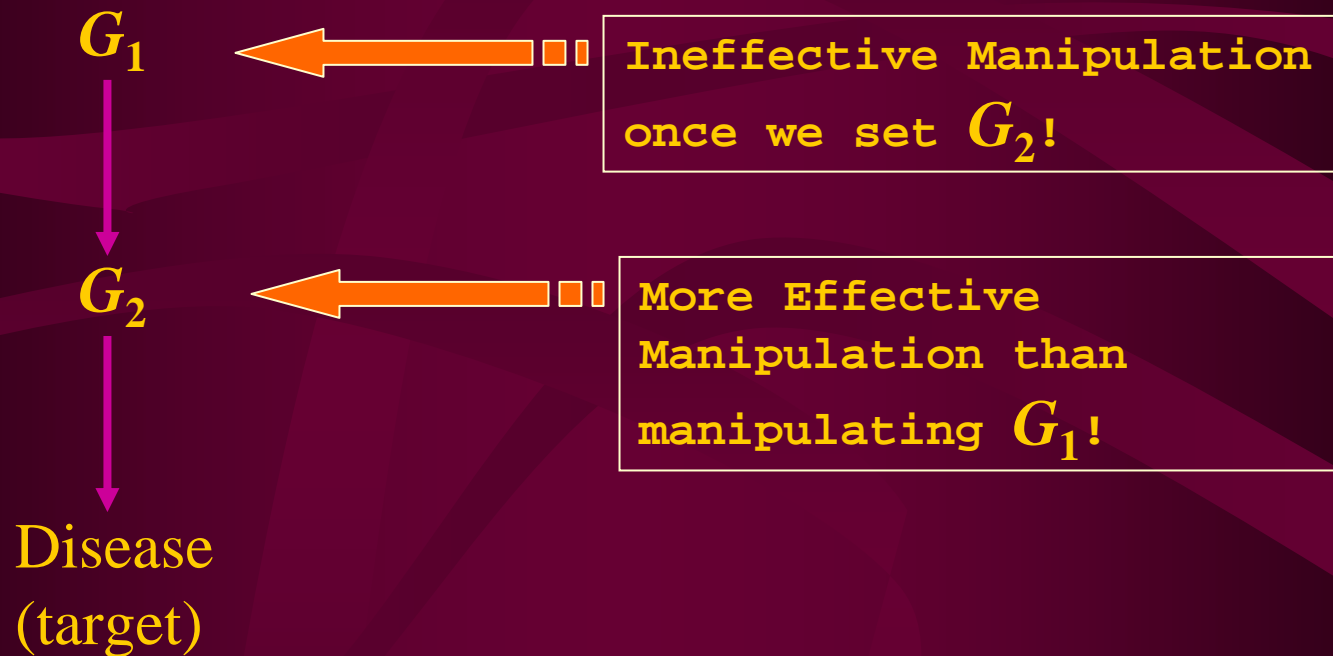
Haemoptysis

Ineffective
Manipulation!



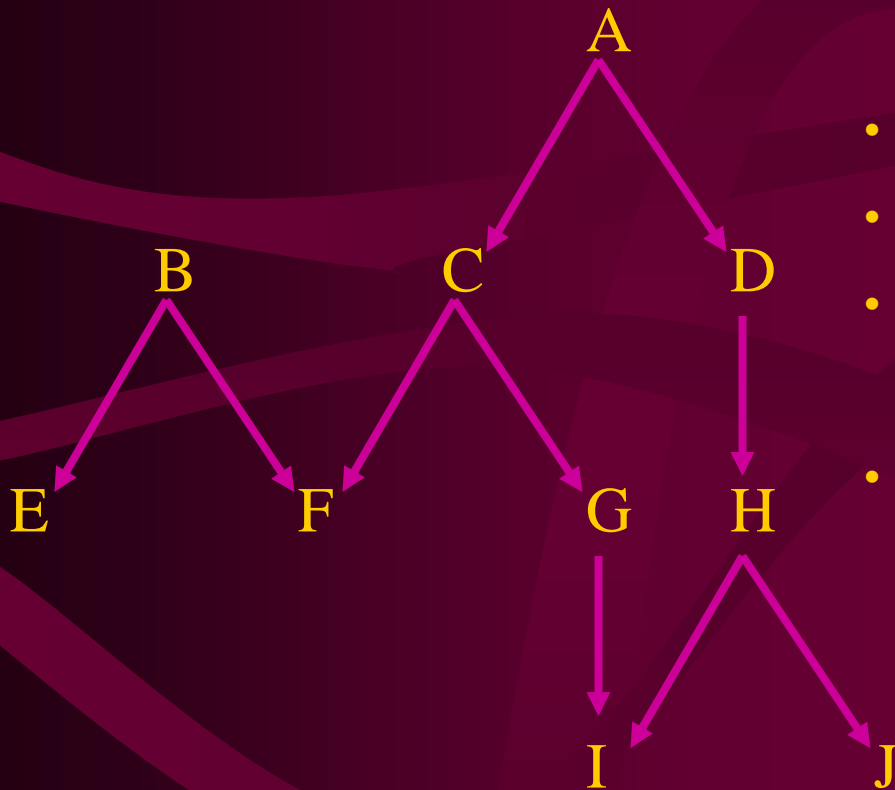
Causal Probabilistic (Bayesian) Network Induction

- The Markov property captures causality:
 - Identifying targets for manipulation in causal chains



Causal Probabilistic (Bayesian) Network Induction

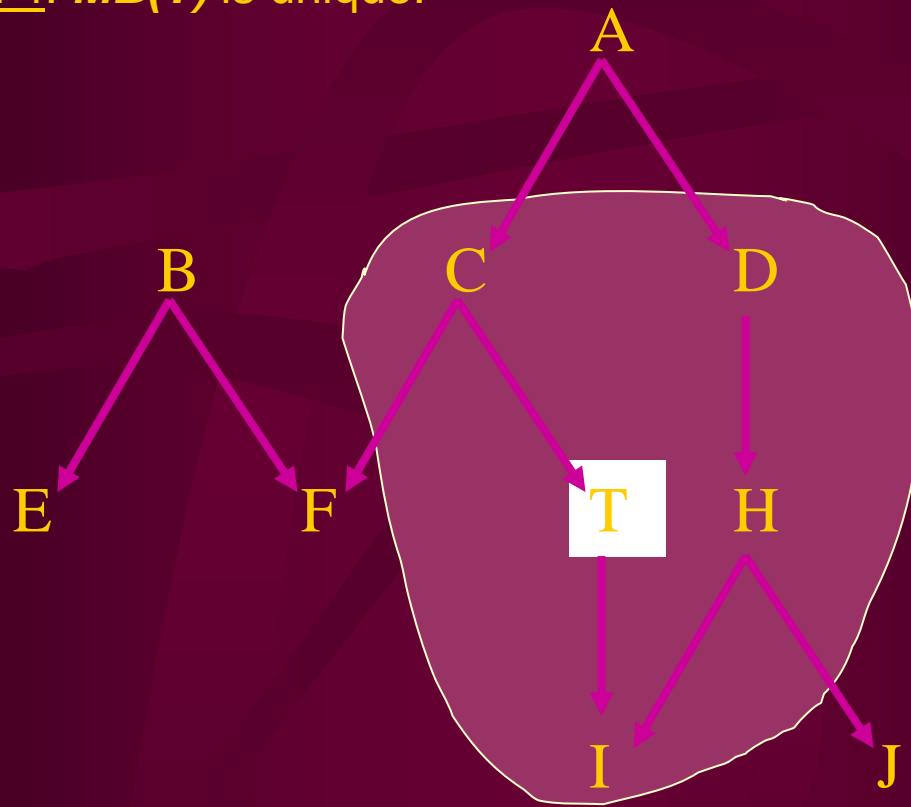
- Inference: Once we have a BN model of some domain we can ask questions:



- Forward: $P(D+, I- | A+) = ?$
- Backward: $P(A+ | C+, D+) = ?$
- Forward & Backward:
 $P(D+, C- | I+, E+) = ?$
- Arbitrary abstraction/Arbitrary predictors/predicted variables

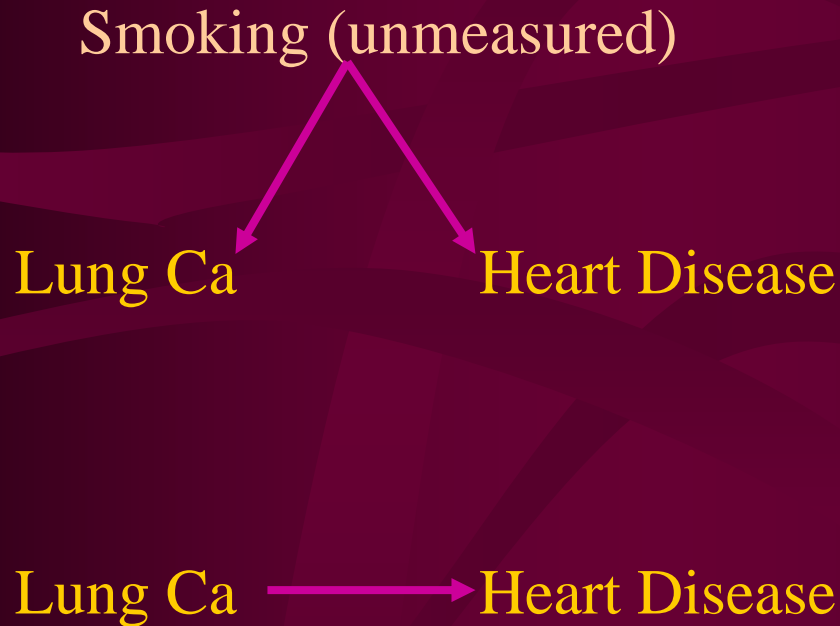
Causal Probabilistic (Bayesian) Network Induction

- Markov Blanket of a feature T : The smallest feature subset conditioned on which all other features are independent of T .
- Theorem 4. $MB(T)$ is unique.



Causal Probabilistic (Bayesian) Network Induction

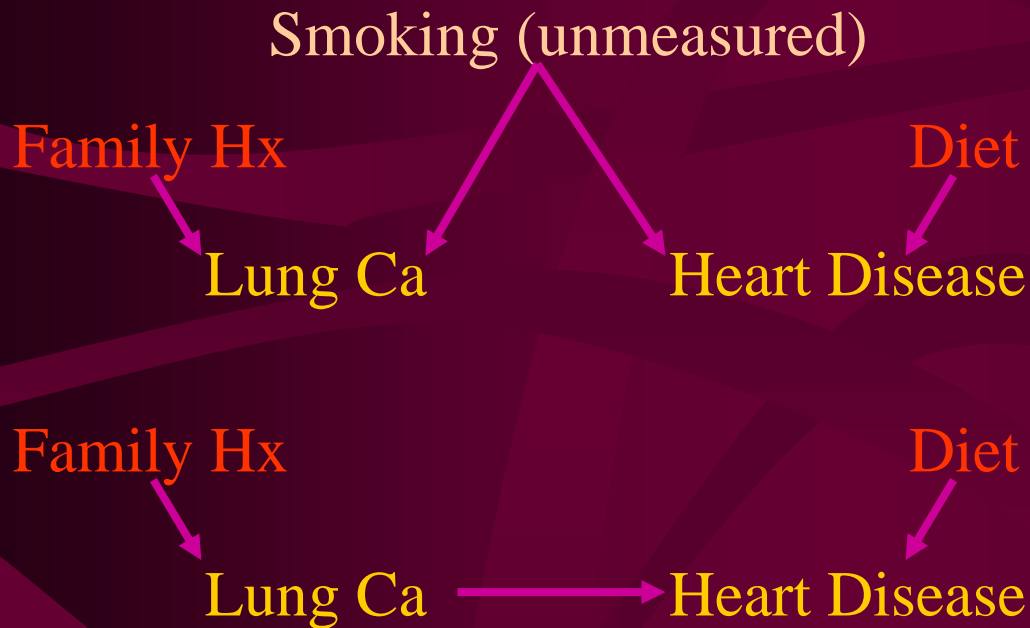
- CPNs can help us learn causal relationships without doing experiments!



But Fisher says these two causal graphs are not distinguishable without doing an experiment (!?)

Causal Probabilistic (Bayesian) Network Induction

- CPNs can help us learn causal relationships without doing experiments!



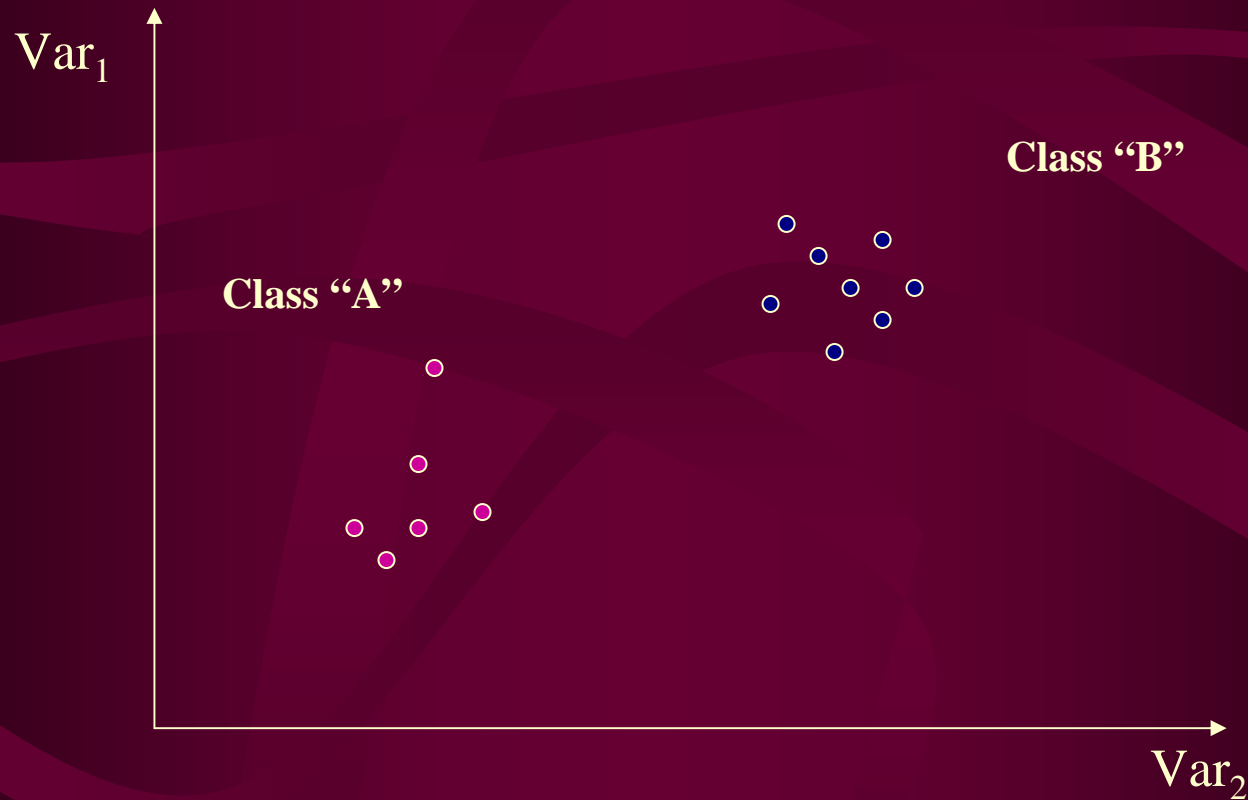
Fisher is right of course; however if we know a cause of each variable of interest then, in many cases, we can derive causal associations without an experiment

Causal Probabilistic (Bayesian) Network Induction

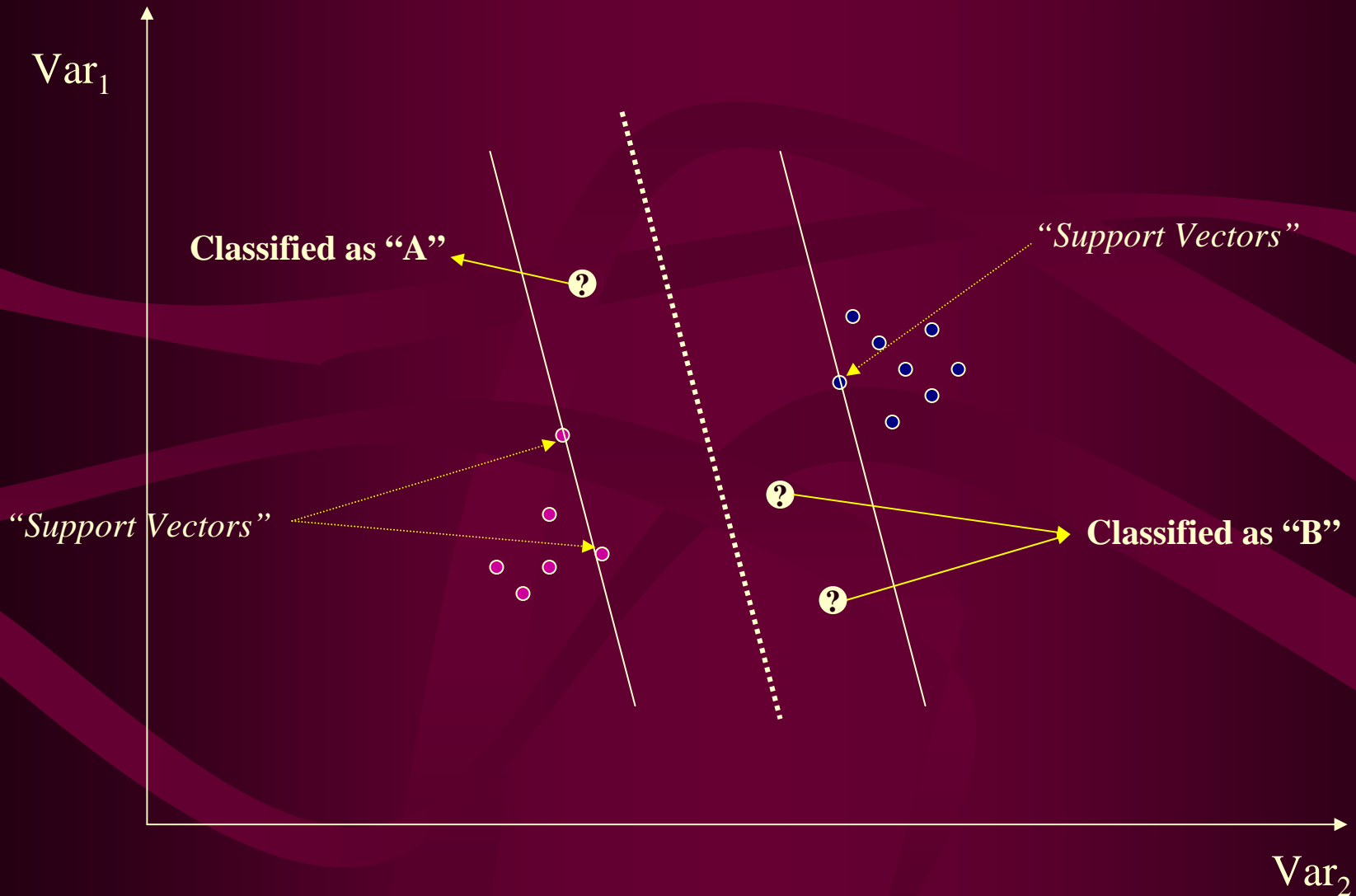
- Faithfulness. The graph G of some CPN C is faithful to a joint probability distribution J over feature set V if and only if every conditional dependence P entailed by J is also entailed by G . (Note that every conditional dependence P entailed by G is also already entailed by J because of the Markov Condition).
- We say that a data-generating process K is faithfully represented by C' , if K in the sample limit produces data with joint probability distribution D , and C' is faithful to D .
- Causal Sufficiency. For every pair of measured variables in the training data, all their common parents are also measured. (Note algorithms for non-causally sufficient datasets also exist, however their theoretical basis is much more complex and will not be discussed here)

Machine Learning Methods: Support Vector machines

- SVMs are classifiers that search for hyperplanes that separate between data points in the data set.

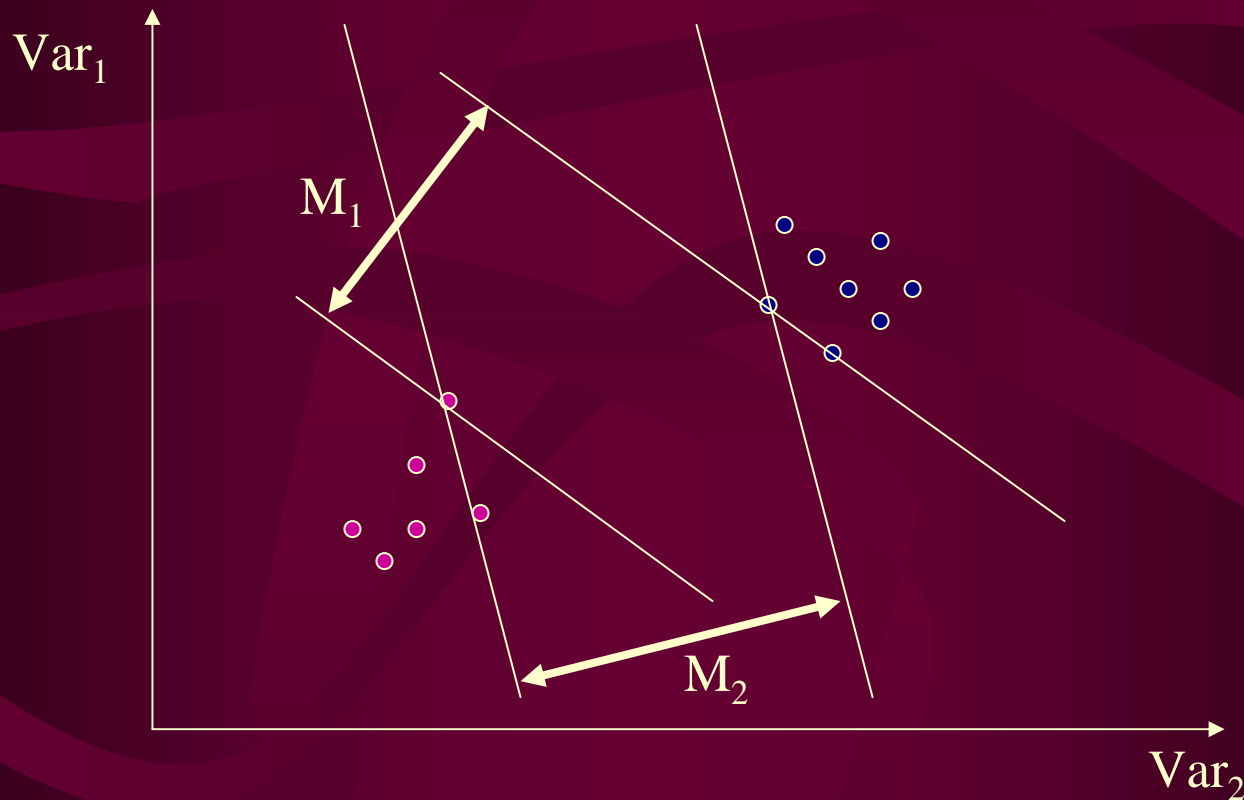


Machine Learning Methods: Support Vector machines



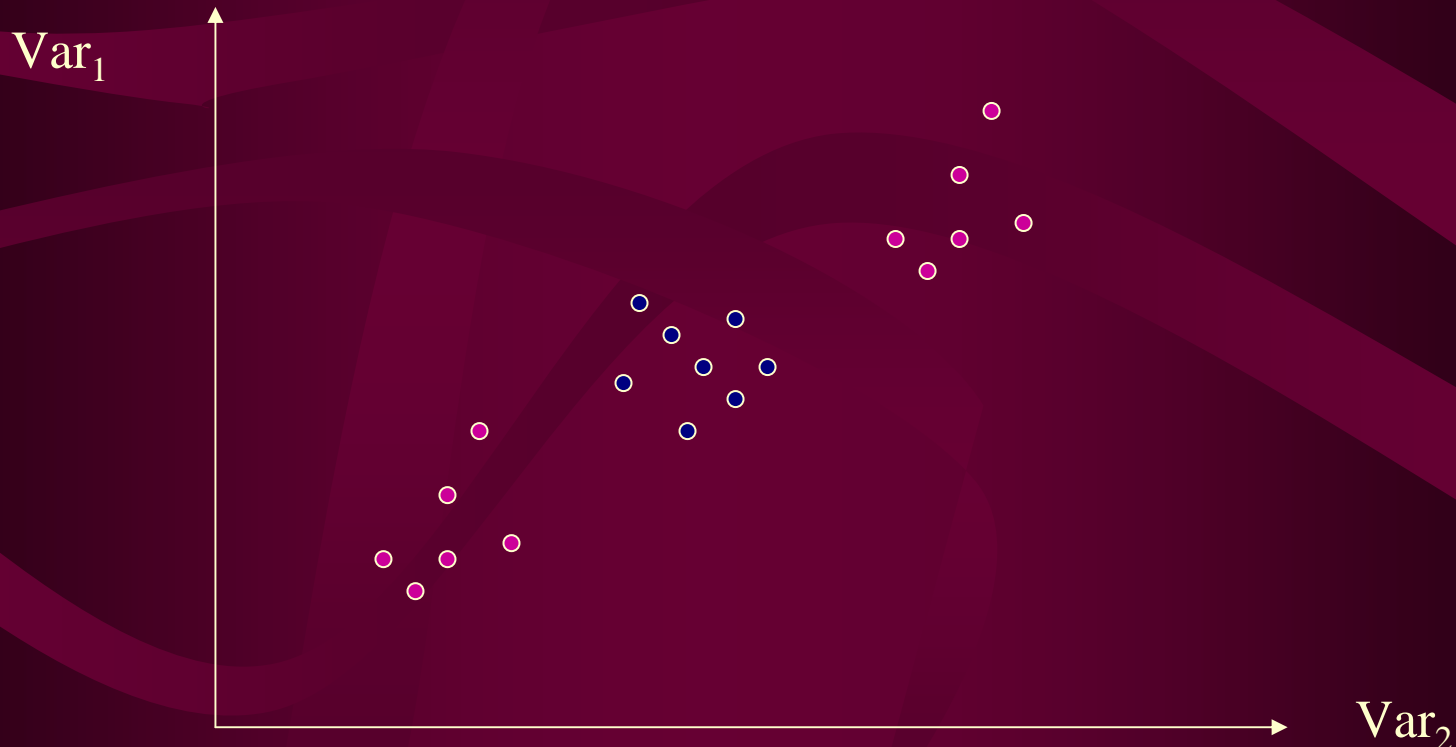
Machine Learning Methods: Support Vector machines

- The separation seeks to maximize the distance between selected (boundary) data points of the different classes (the “support vectors”). This has been shown to improve generalization performance.

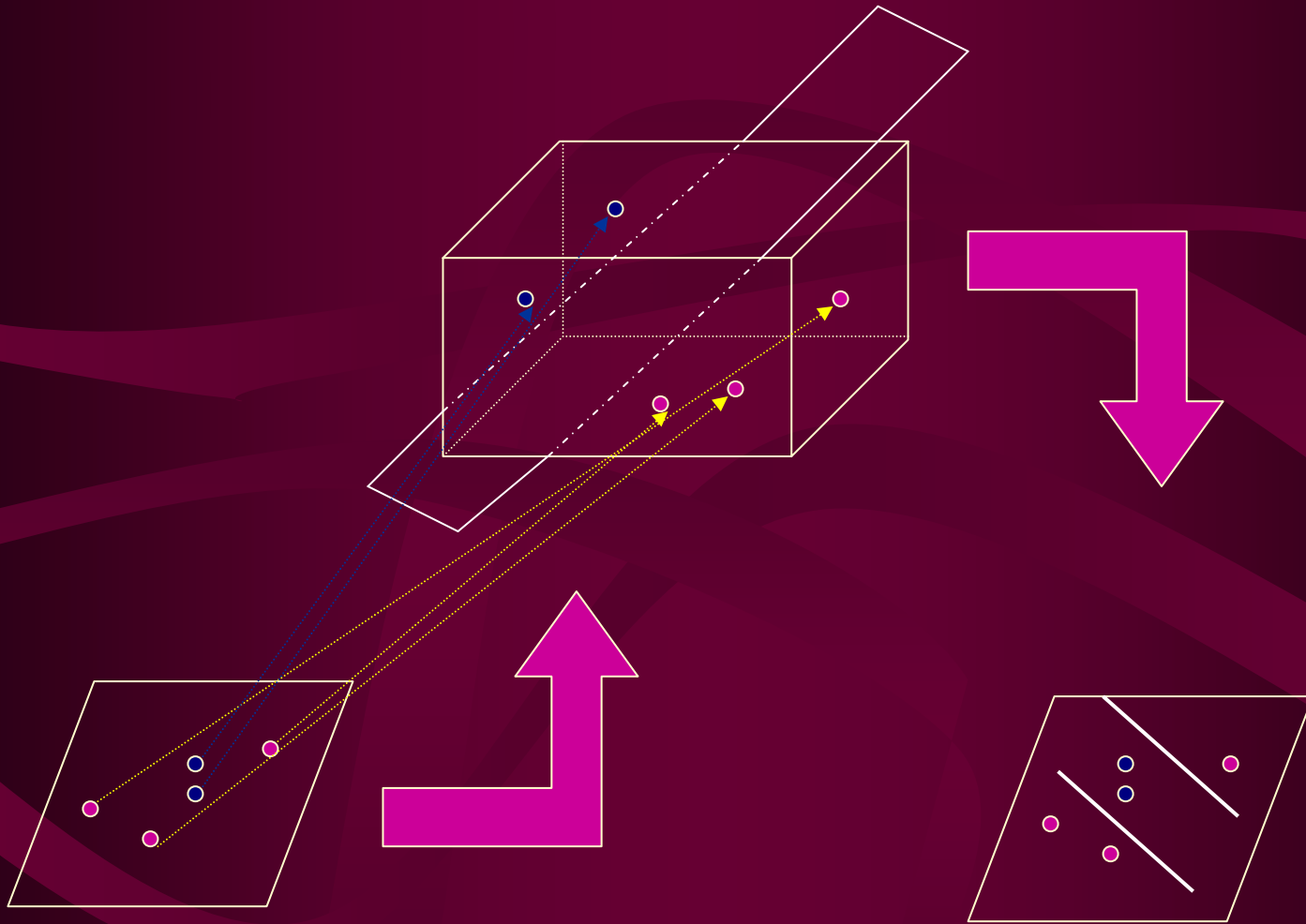


Machine Learning Methods: Support Vector machines

- Non-linearly-separable data are transformed by projection to a higher-dimensional space in which separating hyperplanes can be found. The transformation is achieved through the use of functions (“kernels”) such that optimization can be achieved by using only dot products of the original data vectors and not the whole data, which increases computational efficiency significantly.



Machine Learning Methods: Support Vector machines



Machine Learning Methods: Support Vector machines

- Widely-used kernels are polynomial kernels and Gaussian Radial Basis Function kernels. In contrast to most machine learning methods where optimization is based on steepest-ascent/descent hill climbing heuristic methods, in SVMs the optimization typically achieves a global maximum (for a chosen SVM class).
- Empirical evidence verifies that SVMs are in general, very robust to a low sample-to-feature ratio.

