Structural learning

Probabilistic Graphical Models

Jerónimo Hernández-González

Outline

- Structural learning: why?
- Structural learning based on scoring functions
- Structural learning based on conditional independence tests
- Structural learning for supervised classification

Model learning

Learning algorithm

Approximate P^* by learning a PGM, M, from data $D = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ which is assumed to be i.i.d. sampled from P^*

$$A : D \rightarrow M \equiv (G, \mathbf{\Theta})$$

We want to extract information from D about P^* to encode in:

- The structure of the PGM, G (Structural learning):
  - NP-complete combinatorial optimization problem
  - Efficient heuristics: Local search, genetic algorithms, ...
- The parameters of the PGM, $\mathbf{\Theta}$ (Parametric learning)

** It might be complemented with domain expert information

Structural learning

Importance of the right structure

Missing an edge

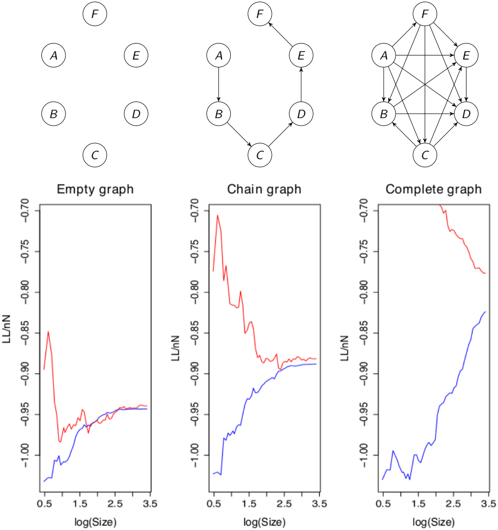
- Incorrect independences?
- Reduces no. parameters
- Might not learn the real distribution P^*
- Tends to better generalization

Adding an extra edge

- Spurious dependencies?
- Increases no. parameters
- Might correctly learn P^*
- Tends to worse generalization

Structural learning

Fitting and generalization



Training vs. validation normalized Log-Likelihood: $nLL\left(\mathcal{M}; D\right) = \frac{LL(\mathcal{M};D)}{v \cdot N}$

Structural learning

Fitting and generalization

Fitting - training LogLikelihood: $nLL(A(D_{tr}) = \mathcal{M}; D_{tr})$

- Tends to increase with number of parameters
- Tends to decrease with number of training cases

Same data is used for model training and scoring

Flexibility (model expressiveness) increases with no. parameters

Generalization - validation LogLikelihood: $nLL(\mathcal{M}; D_{va})$

- (Upper) constrained by the number of parameters
- Tends to increase with number of training cases

Fitting is an upper bound

Note that generalization is unknown!! (might be estimated, e.g., by CV)

Overfitting

$$nLL(\mathcal{M}, D_{tr}) - nLL(\mathcal{M}, D_{va})$$

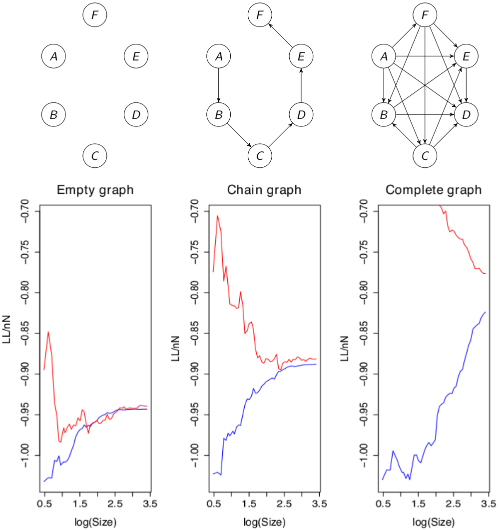
- Fitting and generalization, unbalanced
- Need to learn too many parameters with not enough training samples

Trade off!

Number of parameters vs. number of training cases

Structural learning

Fitting and generalization



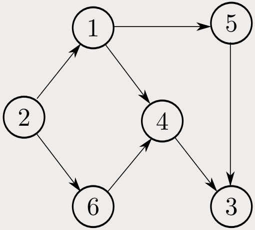
Training vs. validation normalized Log-Likelihood: $nLL\left(\mathcal{M}; D\right) = \frac{LL(\mathcal{M};D)}{v \cdot N}$

Exercise

Counting parameters

Let $X_1, \ldots, X_6$ be binary random variables. Calculate the increase in the no. parameters after the addition of the edges:

- $(2, 5)$
- $(2, 4)$



Structural learning

What?

Find the best structure (which fixes the no. parameters) with a dataset D sampled from a distribution of interest, P^*.

» Note the constrain relationship

When domain expertise cannot provide it

Why?

- Because we need a model to answer future queries
- Because we are interested into structure discovery: the structure is a goal itself

How? A NP-complete problem in the general case

- Based on a scoring function
- Based on conditional independence tests

» Exact polynomial algorithms in specific conditions. Otherwise: **heuristic** search.

Structural learning

Probabilistic Graphical Models

Jerónimo Hernández-González

Score based Structural Learning

General formulation

1. Define a scoring function that evaluates the ability of a structure to describe the observed data

   E.g.: Likelihood, penalized likelihood, ...

2. Search for the structure that maximizes the value of the scoring function

   E.g.: Hill-climbing search, Genetic algorithms, ...

Score based Structural Learning

General formulation

1. **Define a scoring function that evaluates the ability of a structure to describe the observed data**

   E.g.: Likelihood, penalized likelihood, ...

2. Search for the structure that maximizes the value of the scoring function

   E.g.: Hill-climbing search, Genetic algorithms, ...

Objective: Maximum likelihood learning

Select the model $\mathcal{M}$ that maximizes:

$$L(\mathcal{M}; D) = p(D; \mathcal{M}) = \prod_{\boldsymbol{x} \in D} p_{\mathcal{M}}(\boldsymbol{x}) = \prod_{\boldsymbol{x} \in D} \prod_{i=1}^{v} p(x_i | \boldsymbol{pa}_i; \boldsymbol{\Theta}_i)$$

or

$$\log L(\mathcal{M}; D) = N \cdot \sum_{i=1}^{v} -\mathrm{H}(X_i) + \mathrm{MI}(X_i; \boldsymbol{PA}_i)$$

For the purpose of model selection (structure comparison):

$$\arg \max_{\mathcal{M}} \log L(\mathcal{M}; D) \propto \arg \max_{\mathcal{M}} \sum_{i=1}^{v} \mathrm{MI}(X_i; \boldsymbol{PA}_i)$$

» The score of a structure is $score(\mathcal{G}; D) = \max_{\Theta} score((\mathcal{G}, \Theta); D)$

Score based Structural Learning

Scoring function: Likelihood score

Likelihood score

- If the likelihood is used as scoring function, it reduces to the sum of mutual information for each factor

$$\begin{aligned} score_L(\mathcal{G}; D) &= N \cdot \sum_{i=1}^{v} -\mathrm{H}(X_i) + \mathrm{MI}(X_i; \boldsymbol{PA}_i) \\ &\approx \sum_{i=1}^{v} \mathrm{MI}(X_i; \boldsymbol{PA}_i) \end{aligned}$$

- Mutual information quantifies the **strength** of the dependence relationship $X_i \not\perp\!\!\!\perp \boldsymbol{PA}_i$

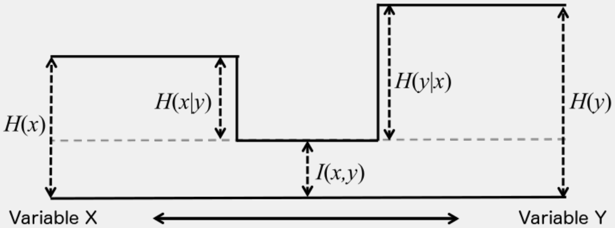
Likelihood score

Entropy

$$\mathrm{H}(X) = -\sum_{x \in X} p(x) \log p(x)$$

Mutual Information

$$\mathrm{MI}(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$
$$= \mathrm{H}(Y) - \mathrm{H}(Y|X) = \mathrm{H}(X) - \mathrm{H}(X|Y)$$



Score based Structural Learning

Scoring function: Likelihood score

Pros: Additively decomposable

$$score(\mathcal{G}; D) = \sum_{i=1}^{v} local\_score(\mathcal{G}_{X_i}; D)$$

- Local changes in the graph only affect locally to the score
- Efficient search heuristics

Cons: Monotone increasing

- Mutual information, $\mathrm{MI}(X;Y)$, is always ≥ 0

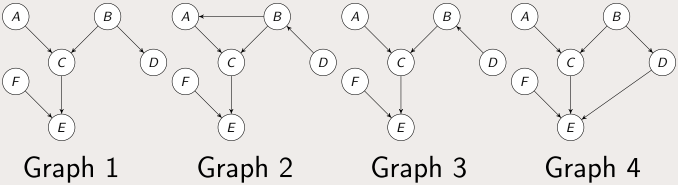
  Only $\mathrm{MI}(X;Y) = 0$, if X, Y are independent in the empirical distribution

- By adding edges, almost always the scoring value increases
- The complete graph maximizes this score $\rightarrow$ Overfitting!!

Exercise

Likelihood score

Which statement about the likelihood scores of these graphs is true?



a) $Score_L(G_1; D) = Score_L(G_3; D)$, for every dataset D

b) $Score_L(G_1; D) \geq Score_L(G_4; D)$, for every dataset D

c) $Score_L(G_2; D) \geq Score_L(G_3; D)$, for every dataset D

d) $Score_L(G_4; D) \geq Score_L(G_2; D)$, for every dataset D

Score based Structural Learning

Trade-off in overfitting (rev.)

Number of parameters vs. number of training cases

Avoiding overfitting

Restrict the hypothesis space by limiting the complexity of $\mathcal{M}$:

- Explicitly: To control it.
  E.g.:
  - Set a limit on the no. parents
  - Set a limit on the no. parameters as a function of no. cases
- Implicitly: To penalize it.
  E.g.:
  - Add a penalty per each additional edge
  - A Bayesian score that averages the score of all the parameters

Penalized log-likelihood score

$$score(\mathcal{G}; D) = score_L(\mathcal{G}; D) - f(dim(\mathcal{G}))$$

Add a penalization term which depends on the complexity of the graph $\mathcal{G}$. E.g.:

- No. parameters
- No. edges

Score based Structural Learning

Scoring function: BIC Score

BIC Score

Logarithmic penalization of the number of parameters

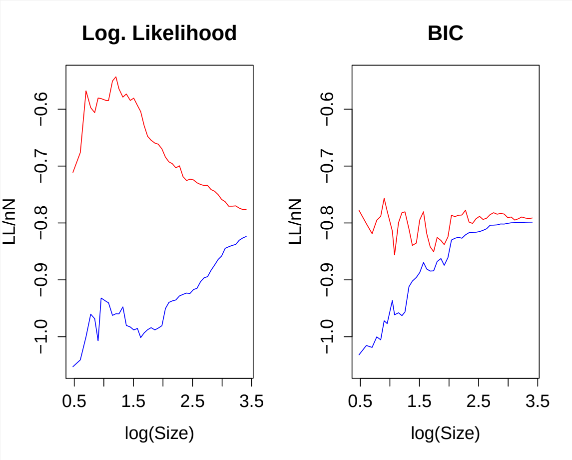
$$score_{BIC}(\mathcal{G}; D) = score_L(\mathcal{G}; D) - \frac{\log N}{2} \# param(\mathcal{G})$$

- As N grows, more emphasis is given to fit to data
- Asymptotically,
  - Required edges are added due to linear growth of likelihood term vs. logarithmic growth of complexity penalization term
  - Edges with spurious contribution to likelihood are left out
  - Any I-equivalent structure of the true $\mathcal{G}^*$ maximizes the score

Score based Structural Learning

Scoring function: BIC Score

Fitting and generalization

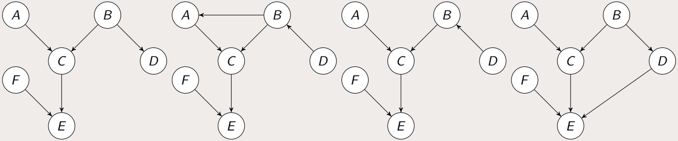


Training vs. validation normalized Log-Likelihood

Exercise

BIC score

Which statement about the BIC scores of these graphs is true?



Graph 1 Graph 2 Graph 3 Graph 4

a) $Score_{BIC}(G_1; D) = Score_{BIC}(G_3; D)$, for every dataset D

b) $Score_{BIC}(G_1; D) \geq Score_{BIC}(G_4; D)$, for every dataset D

c) $Score_{BIC}(G_2; D) \neq Score_{BIC}(G_3; D)$, for every dataset D

d) $Score_{BIC}(G_1; D) \geq Score_{BIC}(G_2; D)$, for every dataset D

Bayesian Score

Bayesian learning assumes a prior to obtain the posterior after observing D:

$$P(\mathcal{G}|D) \propto P(D|\mathcal{G})P(\mathcal{G})$$

We can define a score as:

$$score_B(\mathcal{G}; D) = P(D|\mathcal{G})P(\mathcal{G})$$

where

$$P(D|\mathcal{G}) = \int P(D|\mathcal{G}, \Theta_{\mathcal{G}})P(\Theta_{\mathcal{G}}|\mathcal{G})d\Theta_{\mathcal{G}}$$

** In the general case, this integral is difficult to assess.
Easier if conjugate priors are used: e.g., Dirichlet-Multinomial

The prior over structures, $P(\mathcal{G})$

- Uniform (uninformative)
- Penalize no. edges $P(\mathcal{G}) \propto c^{\#edges(\mathcal{G})}$ with $0 < c < 1$
- Penalize no. parameters

Score based Structural Learning

General formulation

1. Define a scoring function that evaluates the ability of a structure to describe the observed data

   E.g.: Likelihood, penalized likelihood, ...

2. **Search for the structure that maximizes the value of the scoring function**

   E.g.: Hill-climbing search, Genetic algorithms, ...

Score based Structural Learning

Search method: Finding the ***tree*** *with optimal score*

Score equivalence

It means that the $score(X \to Y) = score(Y \to X)$, and

$$\text{Equivalent } I\text{-structures} \implies \text{Same score}$$

$score_L$, $score_{BIC}$, and $score_B$ satisfy score equivalence

Optimal forest structure

Using a decomposable score,

1. Build the complete graph and weigh each edge (X, Y) with

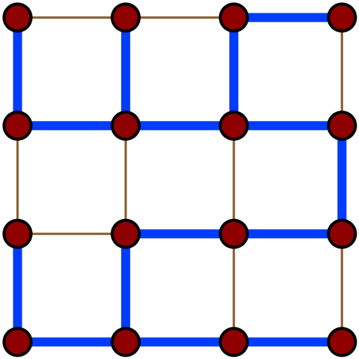
$$\text{máx}(score(X \to Y), 0)$$

2. Find the Maximum Spanning Tree on that graph
3. Remove edges with weight 0

Minimum Spanning Tree

Given a connected graph $\mathcal{U}$ with weighted undirected edges,

a spanning tree of $\mathcal{U}$ is a **tree subgraph** with all the vertices of $\mathcal{U}$ and the minimum possible number of edges



Score based Structural Learning

Search method: Minimum Spanning Tree

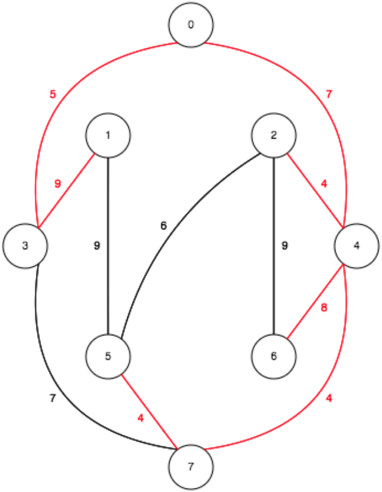
Minimum Spanning Tree

Given a connected graph $\mathcal{U}$ with weighted undirected edges,

a spanning tree of $\mathcal{U}$ is a **tree subgraph** with all the vertices of $\mathcal{U}$ and the minimum possible number of edges

a minimum spanning tree is a **spanning tree** of $\mathcal{U}$ having minimum weight

If $\mathcal{U}$ is unconnected, a minimum spanning forest is the union of a MST for each connected component



Kruskal's algorithm

Given an undirected edge-weighted graph $\mathcal{U}$ with n nodes,

1. Let L be the sorted list of edges of $\mathcal{U}$ by increasing weight
2. Set $\mathcal{T} = \emptyset$, where $\mathcal{T}$ is the subgraph of the MST
3. Pull the top edge from L and add it to $\mathcal{T}$
4. Remove from L any edge that forms a loop in $\mathcal{T}$
5. If $\mathcal{T}$ has $n-1$ edges, return $\mathcal{T}$ (a tree)
   ElseIf $L = \emptyset$, return $\mathcal{T}$ (a forest)
   Otherwise, go to step 3

`https://www.cs.usfca.edu/~galles/visualization/Kruskal.html`

Exercise

Recovering directionality

After we find an undirected spanning tree, which is the most efficient way to transform it into a directed spanning tree?

a) Evaluate all possible directions for the edges (at most, 2^n possible sets of edge directions) by iterating over them (within $O(2^n)$ time).

b) Exploit score decomposability to evaluate all possible directions for the edges (within $O(n)$ time).

c) Pick any arbitrary direction for each edge (within $O(n)$ time). Due to score equivalence, all possible directed versions of the optimal undirected spanning forest have the same score.

d) Pick any arbitrary root, and direct all edges away from it (within $O(n)$ time).

Score based Structural Learning

*Search method: Finding the **tree** with optimal score*

Score equivalence

It means that the $score(X \to Y) = score(Y \to X)$, and

$$\text{Equivalent } I\text{-structures} \implies \text{Same score}$$

$score_L$, $score_{BIC}$, and $score_B$ satisfy score equivalence

Optimal forest structure

Using a decomposable score,

1. Build the complete graph and weigh each edge (X, Y) with
$$\text{máx}(score(X \to Y), 0)$$
2. Find the Maximum Spanning Tree on that graph
3. Remove edges with weight 0

Finding the maximal scoring network with at most $k > 1$ parents is NP-hard!

Score based Structural Learning

*Search method: **Heuristic** search*

Local search

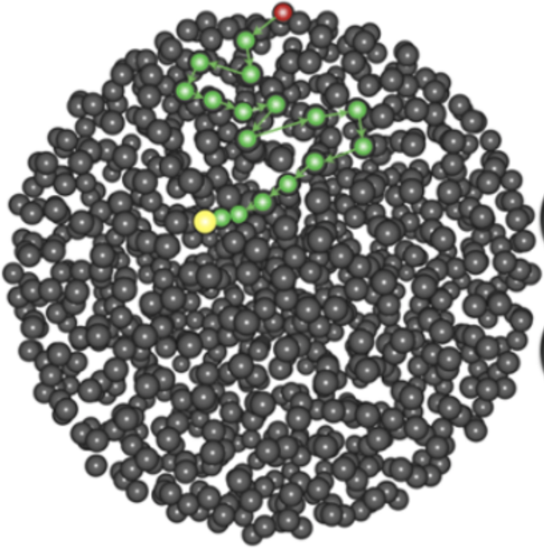
- Based on the concept of neighborhood: each graph $\mathcal{G}$ has *a few* neighbor graphs $\mathcal{G}'$

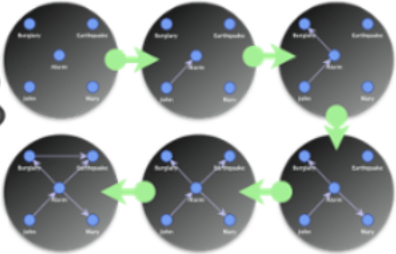
  Defined by operators such as: edge addition, removal or reversal
- Search technique

  Greedy hill-climbing, Best first search, Genetic, Simulated annealing, ...
- Efficient: Polynomial complexity
- Suboptimal solutions: Local optima

Score based Structural Learning

*Search method: **Heuristic** search*





Greedy hill climbing

1. Start with an initial network. E.g.,
   - the empty or a random network
   - best tree
   - prior knowledge
2. **Repeat** at each iteration:
   2.1 Calculate the score$^{(**)}$ of any possible structure obtained by a single local change. E.g.,
      - add an edge
      - remove an edge
      - reverse an edge

   2.2 Apply the change that most improves the score

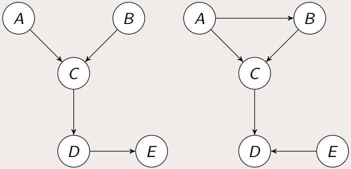
   Until no change improves the score # Local maxima reached

Thanks to score$^{(**)}$ decomposability, we can assess the scoring value by reevaluating only the part affected by the local change

Exercise

Calculating Likelihood Differences

We need to choose between these two graphs with likelihood score:



Graph 1 Graph 2

What is $Score_L(G_1; D) - Score_L(G_2; D)$ given dataset D of size N?

a) $N \cdot [\mathrm{MI}_{\hat{p}}(C; A, B) + \mathrm{MI}_{\hat{p}}(D; C) + \mathrm{MI}_{\hat{p}}(E; D) - \mathrm{MI}_{\hat{p}}(B; A) - \mathrm{MI}_{\hat{p}}(D; C, E)]$

b) $N \cdot [\mathrm{MI}_{\hat{p}}(D; C) + \mathrm{MI}_{\hat{p}}(E; D) - \mathrm{MI}_{\hat{p}}(A; B) - \mathrm{MI}_{\hat{p}}(D; C, E) - \mathrm{H}_{\hat{p}}(A, B, C, D, E)]$

c) $N \cdot [\mathrm{MI}_{\hat{p}}(D; C) + \mathrm{MI}_{\hat{p}}(E; D) - \mathrm{MI}_{\hat{p}}(B; A) - \mathrm{MI}_{\hat{p}}(D; C, E)]$,

d) $N \cdot [\mathrm{MI}_{\hat{p}}(A; B) - \mathrm{H}_{\hat{p}}(A, B)]$

e) $N \cdot \mathrm{MI}_{\hat{p}}(A; B)$

Structural learning

Probabilistic Graphical Models

Jerónimo Hernández-González

Structural learning

What?

Find the best structure (which fixes the no. parameters) with a dataset D sampled from a distribution of interest, P^*.

» Note the constrain relationship

When domain expertise cannot provide it

Why?

- Because we need a model to answer future queries
- Because we are interested into structure discovery: the structure is a goal itself

How? A NP-complete problem in the general case

- Based on a scoring function
- Based on conditional independence tests

» Exact polynomial algorithms in specific conditions. Otherwise: **heuristic** search.

Structural learning

Based on conditional independence tests

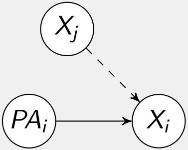
PGMs as models that fulfill a set of independencies

Objective: Model the most important dependencies of p

Iterative procedure:

- given the current structure $\mathcal{G}$
- test each possible inclusion of a new parent:

  should we move from $p(x_i|\boldsymbol{pa}_i)$ to $p(x_i|\boldsymbol{pa}_i \cup \{x_j\})$?



- i.e., check the statistical independence: $X_i \perp\!\!\!\perp X_j \mid \boldsymbol{PA}_i$

Independence test: χ squared

χ squared: independence test, $\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B$

A chi-square (χ^2) statistic measures the difference of the observed and expected frequencies of the outcomes of two sets of variables. Null hypothesis (H_0) assumes independence:

$$\forall \boldsymbol{x}, p(\boldsymbol{x}_{A,B}) = p(\boldsymbol{x}_A) \cdot p(\boldsymbol{x}_B)$$

Statistic:

$$X^2 = \sum_{i=1}^{r_A} \sum_{j=1}^{r_B} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

$$X^2 \sim \chi^2(v)$$

i.e., X^2 is distributed according to a χ^2 distribution with v degrees of freedom, where $v = (r_A - 1) \cdot (r_B - 1)$

Unlikelihood of the independence according to observed data

Independence test: χ squared

χ squared: independence test, $\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B$

A chi-square (χ^2) statistic measures the difference of the observed and expected frequencies of the outcomes of two sets of variables. Null hypothesis (H_0) assumes independence:

$$\forall \boldsymbol{x}, p(\boldsymbol{x}_{A,B}) = p(\boldsymbol{x}_A) \cdot p(\boldsymbol{x}_B)$$

Alternative statistic (likelihood-ratio):

$$G^2 = 2 \sum_{i=1}^{r_A} \sum_{j=1}^{r_B} O_{ij} \log \frac{O_{ij}}{E_{ij}} = 2N \cdot MI(\boldsymbol{X}_A; \boldsymbol{X}_B)$$

$$G^2 \sim \chi^2(v)$$

i.e., G^2 is distributed according to a χ^2 distribution with v degrees of freedom, where $v = (r_A - 1) \cdot (r_B - 1)$

Unlikelihood of the independence according to observed data

Example: political preferences on taxes

Assuming independence: which are the expected counts?

	favor	indifferent	opposed	total
democrat				285
republican				215
total	202	150	148	500

Example: political preferences on taxes

Assuming independence: which are the expected counts?

	favor	indifferent	opposed	total
democrat	$\frac{285}{500} \cdot \frac{202}{500} \cdot 500 = 115{,}14$	85,50	84,36	285
republican	86,86	64,50	63,64	215
total	202	150	148	500

Example: political preferences on taxes

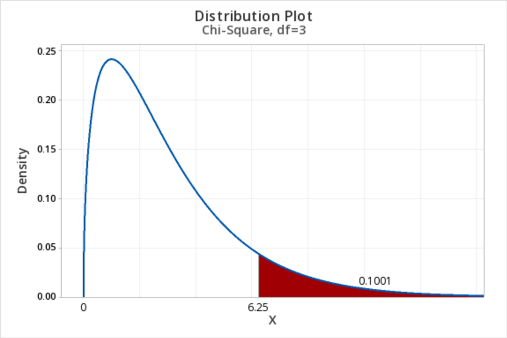
Observed counts vs. Expected counts (given indep. assumption)

	favor	indifferent	opposed	total
democrat	138 – 115,14	83 – 85,50	64 – 84,36	285
republican	64 – 86,86	67 – 64,50	84 – 63,64	215
total	202	150	148	500

$$X^2 = \frac{(138-115{,}14)^2}{115{,}14} + \frac{(83-85{,}50)^2}{85{,}50} + \frac{(64-84{,}36)^2}{84{,}36} + \frac{(64-86{,}86)^2}{86{,}86} + \frac{(67-64{,}50)^2}{64{,}50} + \frac{(84-63{,}64)^2}{63{,}64} = 22{,}152$$

Example: political preferences on taxes

$$X^2 = \frac{(138-115{,}14)^2}{115{,}14} + \frac{(83-85{,}50)^2}{85{,}50} + \frac{(64-84{,}36)^2}{84{,}36} + \frac{(64-86{,}86)^2}{86{,}86} + \frac{(67-64{,}50)^2}{64{,}50} + \frac{(84-63{,}64)^2}{63{,}64} = 22{,}152$$



Upper-tail critical values of chi-square distribution with ν degrees of freedom

	Probability less than the critical value				
ν	0.90	0.95	0.975	0.99	0.999
1	2.706	3.841	5.024	6.635	10.828
2	4.605	5.991	7.378	9.210	13.816
3	6.251	7.815	9.348	11.345	16.266
4	7.779	9.488	11.143	13.277	18.467
5	9.236	11.070	12.833	15.086	20.515
6	10.645	12.592	14.449	16.812	22.458
7	12.017	14.067	16.013	18.475	24.322
8	13.362	15.507	17.535	20.090	26.125
9	14.684	16.919	19.023	21.666	27.877
10	15.987	18.307	20.483	23.209	29.588
11	17.275	19.675	21.920	24.725	31.264
12	18.549	21.026	23.337	26.217	32.910
13	19.812	22.362	24.736	27.688	34.528
14	21.064	23.685	26.119	29.141	36.123
15	22.307	24.996	27.488	30.578	37.697

Unlikelihood of the independence according to observed data

Conditional independence test, $X_i \perp\!\!\!\perp X_j | \boldsymbol{PA}_i$

A chi-square (χ^2) which assumes (null hypothesis, H_0) conditional independence:

$$\forall \boldsymbol{x}, p(x_i, x_j | \boldsymbol{x}_{\boldsymbol{PA}_i}) = p(x_i | \boldsymbol{x}_{\boldsymbol{PA}_i}) \cdot p(x_j | \boldsymbol{x}_{\boldsymbol{PA}_i})$$

Statistic:

$$X^2 = \sum_{l=1}^{r_i} \sum_{m=1}^{r_j} \sum_{k=1}^{r_{\boldsymbol{PA}_i}} \frac{(O_{lmk} - E_{lmk})^2}{E_{lmk}} \quad \text{with} \quad E_{lmk} = N_{\cdot mk} N_{l \cdot k} / N_{\cdot\cdot k}$$

$$X^2 \sim \chi^2(v)$$

i.e., X^2 is distributed according to a χ^2 distribution with v degrees of freedom, where $v = (r_A - 1) \cdot (r_B - 1) \cdot r_C$

Unlikelihood of the independence according to observed data

Independence test: χ squared

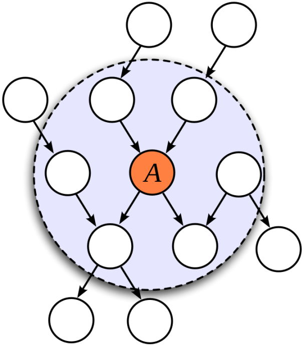
χ squared: independence test, $\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B$

Results depend on the magnitude of the difference between actual and observed frequencies (X^2 and/or G^2), the degrees of freedom (v), and the sample size (N).

- The **degrees of freedom** increase *exponentially* with the number of parents: $(r_i - 1)(r_j - 1)(r_{\boldsymbol{PA}_i})$
  The **test** becomes more demanding
- The **difference** X^2 (and G^2) increases *linearly* with the **sample size** N
  The **test** becomes less demanding

Structural learning

Specific approaches: Markov blanket



Markov blanket

Set of nodes including parents, children, and other parents of its children.

The Markov blanket renders a node independent of the rest of the network.

The joint distribution of the variables in its Markov blanket is sufficient for calculating the distribution of a node.

Structural learning

Probabilistic Graphical Models

Jerónimo Hernández-González

PGMs for supervised classification

A generative approach

Advantages of $p_M(x, c)$ (vs. of $p_M(c|\boldsymbol{x})$)

- Flexibility
- Discover relationships between vars.: $p(\boldsymbol{x}_A|c)p(\boldsymbol{x}_B|c)p(c)$
- Incorporate a priori knowledge: Bayesian statistics
- Obtain the conditional distribution $p_M(c|\boldsymbol{x})$
- Deal with missing values, outliers, ...
- Rejection region, $p_M(c|\boldsymbol{x}) > t$

Disadvantages of $p_M(x, c)$ (vs. of $p_M(c|\boldsymbol{x})$)

- Harder problem
- Models irrelevant information, $p_M(\boldsymbol{x})$, for classification
- Requires more parameters

Going discriminative: Modeling $p(C|\mathbf{x})$ but not $p(\mathbf{X})$

Capture specific knowledge about classification

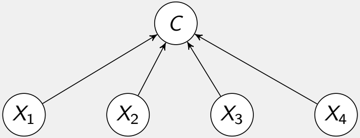
Few parameters with highly discriminative information:

- Model the dependencies of the form $X_i \not\perp\!\!\!\perp C|\mathbf{X}_S$
- Model the most relevant dependencies like $\mathbf{X}_A \not\perp\!\!\!\perp \mathbf{X}_B|C$
- Avoid redundancy $\{C \perp\!\!\!\perp \mathbf{X}_A|\mathbf{X}_{V\setminus A}\}$

PGMs for supervised classification

Structures biased towards classification: Informed Bayes

Informed Bayes



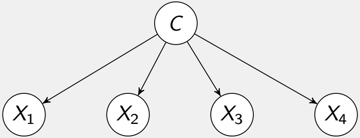
$$p_M(\boldsymbol{x}, c) = \hat{p}(c|\boldsymbol{x}) \prod_{i=1}^{v} \hat{p}(x_i)$$

- Models all and only the important dependencies
- Exponential number of parameters w.r.t. v
- Given enough data resembles the Bayes classifier
- In normal scenarios, tends to overfit!!

PGMs for supervised classification

Structures biased towards classification: Naive Bayes

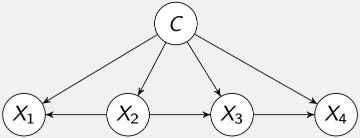
Naive Bayes



$$p_M(\boldsymbol{x}, c) = \hat{p}(c) \prod_{i=1}^{v} \hat{p}(x_i|c)$$

- Assumes that features are independent given the class: $\{X_i \perp\!\!\!\perp X_j | C\}$
- Models the most important dependences: $\{X_i \not\perp\!\!\!\perp C | \boldsymbol{X}_S\}$
- No. parameters is linear with No. variables
- Worse generalization with large training sets
- Low risk of overfitting

Tree-augmented naive Bayes



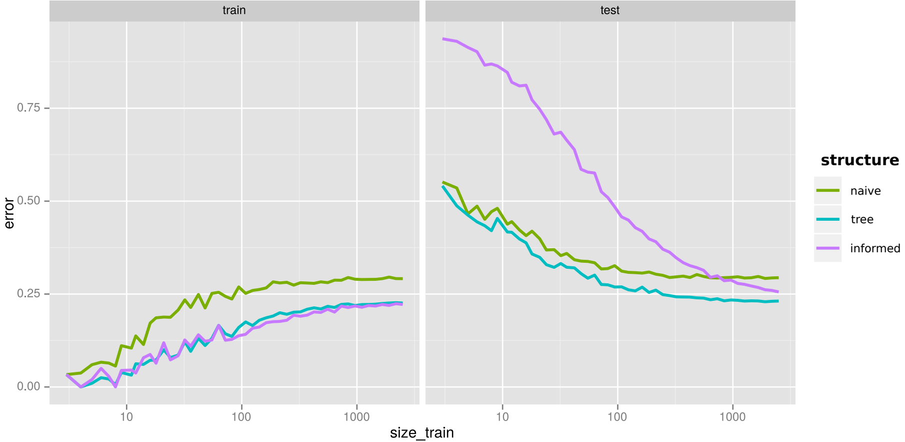
$$p_M(\boldsymbol{x}, c) = \hat{p}(c) \prod_{i=1}^{v} \hat{p}(x_i | x_j, c)$$

- Generalization of naive Bayes
- Breaks the strong conditional independence assumption of NB
- Apart from C, each feature can have a parent ($k = 1$) among the rest of features
  MST among features with weights $CMI(X_i, X_j | C)$

Further generalization: k-dependence Bayesian classifier restricts the maximum number of parents to k

Fitting and generalization

Training (left) and test (right) classification error



Exercise

Hidden variable

Consider a generative naive Bayes model with 10 binary-valued features $\{X_1, \ldots, X_{10}\}$, but the class variable C is not observed. Still, C is strongly correlated with its children.

Suppose we learn a structure directly on $\{X_1, \ldots, X_{10}\}$ (without C). Which structure are we likely to learn if we use the likelihood score as the structure learning criterion?

a) The empty network, i.e., a network consisting of only the variables but no edges between them.

b) Some connected network over $\{X_1, \ldots, X_{10}\}$ which is not fully connected nor empty.

c) A fully connected network, i.e., one with an edge between every pair of nodes.

Summary

Structural learning

- Usually done by scored based approaches and local search
- Many alternatives, some exact in controlled scenarios
- Structures predefined specific for supervised classification

Structural learning

Probabilistic Graphical Models

Jerónimo Hernández-González