The Chow-Liu Algorithm


The Goal

Given a finite set of samples in a dataset, estimate the underlying n-dimensional discrete probability distribution using a tree model.

Trees

What is a tree?
• The variables in the dataset are the vertices V
• There are edges in the set E that connect the vertices
• We’ll assume the edges are undirected for now
• A graph (V,E) is a tree if it is connected and has no cycles

Technical point: We will allow our trees to be a forest i.e. the tree model we learn may be disconnected

Trees

• In a directed tree, we pick a vertex as the root
• We then turn the edges into directed edges and orient the edges away from the root
• This means that each vertex has at most one parent (but may have more than one child)

Tree Models

Notation:
• x (as in bold x) is an n-dimensional vector i.e. x = (x₁, x₂, ..., xₙ)
• Each xᵢ in x is a variable
• P(x) is a joint probability distribution of n discrete variables x₁, x₂, ..., xₙ

• We want to approximate the true joint probability distribution using tree models of the form:
  \[ P_t(x) = \prod_{i=1}^{n} P(x_i | x_{\pi(i)}) \]
• \( \pi(i) \) means “parent of variable i”
• If i is the root then \( \pi(i) \) is the empty set:
  \[ P(x_i | x_{\pi(i)}) = P(x_i) \]
Tree Models

\[ P_T(x) = \prod_{i=1}^{n} P(x_i|x_{\pi(i)}) \]

- Tree models consider the pairwise relationships between variables in the dataset.
- It is an improvement over just treating the variables independently of each other.

Closeness of approximation

- Let \( P(x) \) and \( P_T(x) \) be two probability distributions of \( n \) discrete variables \( x = (x_1, x_2, ..., x_n) \).
- Let

\[
KL(P, P_T) = \sum_x P(x) \log \frac{P(x)}{P_T(x)}
\]

Note: This summation is over all configurations of \( (x_1, x_2, ..., x_n) \).

The formula for \( KL(P, P_T) \) is called the Kullback-Leibler divergence (or KL divergence for short).

Kullback-Leibler Divergence

- We’ll rewrite the KL divergence as:

\[
KL(P, P_T) = \sum_x P(x) \log P(x) - \sum_x P(x) \log P_T(x)
\]

- The first term doesn’t depend on \( P_T \).
- The second term is known as the cross-entropy between \( P \) and \( P_T \).
- Properties of KL divergence:
  - \( KL(P, P_T) \geq 0 \)
  - \( KL(P, P_T) = 0 \) if and only if \( P(x) \equiv P_T(x) \) for all \( x \).

A Minimization Problem

Given:

- An \( n \)th-order probability distribution \( P(x_1, x_2, ..., x_n) \) with \( x_i \) being discrete.
- \( T_n \) - The set of all possible first-order dependence trees.

Find the optimal first-order dependence tree \( \tau \) such that \( KL(P, P_T) \leq KL(P, P_T) \) for all \( \tau \in T_n \).

Exhaustive Search

- Why not just search over all possible trees?
- Not feasible -- there are \( n^{(n-2)} \) possible trees with \( n \) vertices (from Cayley’s formula).
- We will turn the search into a maximum weight spanning tree (MWST) problem.

Mutual Information

- Define the mutual information \( I(x_i, x_j) \) between two variables \( x_i \) and \( x_j \) to be:

\[
I(x_i, x_j) = \sum_{x_i, x_j} P(x_i, x_j) \log \left( \frac{P(x_i, x_j)}{P(x_i)P(x_j)} \right)
\]

- Key insight: a probability distribution of tree dependence \( P_T(x) \) is an optimum approximation to \( P(x) \) iff its tree model has maximum weight.
- Proof to follow.
Proof

$$KL(P, P) = \sum_x P(x) \log P(x) - \sum_x P(x) \sum_{i=1}^{n} \log P(x_i|x_{\pi(i)})$$

$$= \sum_x P(x) \log P(x) - \sum_x P(x) \sum_{i=1, \root}^{n} \log \frac{P(x_i, x_{\pi(i)})}{P(x_{\pi(i)})}$$

$$= \sum_x P(x) \log P(x) - \sum_x P(x) \sum_{i=1}^{n} \log \frac{P(x_i, x_{\pi(i)})}{P(x_i)P(x_{\pi(i)})}$$

$$= -\sum_x P(x) \log P(x)$$

Proof (continued)

Note that: $-\sum_x P(x) \log P(x) = -\sum_{x_i} P(x_i) \log P(x_i)$

To see this, suppose $x = (x_1, x_2)$, let all variables are binary, let $i=1$

$$= -\sum P(x) \log P(x_i)$$

$$= -(P(x_i = 0, x_2 = 0) \log P(x_i = 0) + P(x_i = 0, x_2 = 1) \log P(x_i = 0) + P(x_i = 1, x_2 = 0) \log P(x_i = 1) + P(x_i = 1, x_2 = 1) \log P(x_i = 1))$$

$$= -(P(x_i = 0) \log P(x_i = 0) + P(x_i = 1) \log P(x_i = 1))$$

$$= -\sum x_i P(x_i) \log P(x_i) = -\sum x_i P(x_i) \log P(x_i)$$

Proof (continued)

In the same way:

$$\sum_x P(x) \log \frac{P(x_i, x_{\pi(i)})}{P(x_{\pi(i)})}$$

$$= \sum_{x_i, x_{\pi(i)}} P(x_i, x_{\pi(i)}) \log \frac{P(x_{\pi(i)})}{P(x_{\pi(i)})} = I(x_i, x_{\pi(i)})$$

Proof (continued)

One more piece of notation:

$$H(x) = -\sum_x P(x) \log P(x)$$

$$H(x_i) = -\sum x_i P(x_i) \log P(x_i)$$

Substituting the expressions above and from pg 12 into the last line of pg 13:

$$KL(P, P) = -\sum_{i=1}^{n} I(x_i, x_{\pi(i)}) + \sum_{i=1}^{n} H(x_i) - H(x)$$

The algorithm

- First calculate all $n(n-1)/2$ pairwise mutual information measures
- Use Kruskal’s algorithm to construct maximum weight spanning tree:
  - Construct tree one edge at a time, in decreasing order of the weights
  - If all weights are $> 0$, you get one connected component
  - Running time is $O(n^2)$ for $n$ variables because you have to consider all $n(n-1)/2$ edges

Proof

$$KL(P, P) = -\sum_{i=1}^{n} I(x_i, x_{\pi(i)}) + \sum_{i=1}^{n} H(x_i) - H(x)$$

Mutual information is always $\geq 0$

Independent of the dependence tree

Minimizing $I(P, P)$ is the same as maximizing the total branch weight:

$$\sum_{i=1}^{n} I(x_i, x_{\pi(i)})$$
Estimation

• But in order to calculate mutual information \( I(x_i, x_j) \), you need the probability distribution \( P(x) \).
• Need to estimate the mutual information from a finite set of samples using maximum likelihood estimation.

Suppose you are given \( s \) independent samples \( x_1, x_2, \ldots, x_s \) of a discrete variable \( x \). Each sample is an \( n \)-component vector \( \mathbf{x} = (x_1, x_2, \ldots, x_n) \).

Define:

\[
\begin{align*}
n_{uv}(i, j) & = \# \text{ of samples with } x_i = u \text{ and } x_j = v \\
f_{uv}(i, j) & = \frac{n_{uv}(i, j)}{\sum_v n_{uv}(i, j)} \\
f_u(i) & = \sum_v f_{uv}(i, j)
\end{align*}
\]

Maximum Likelihood Estimator for \( P(x_i = u, x_j = v) \)

Maximum Likelihood Estimator for \( P(x_i = u) \)

Estimation

Calculate:

\[
\hat{I}(x_i, x_j) = \sum_{u,v} f_{uv}(i, j) \log \frac{f_{uv}(i, j)}{f_u(i) f_v(j)}
\]

Use \( \hat{I}(x_i, x_j) \) in Kruskal’s algorithm instead of \( I(x_i, x_j) \).

The entire algorithm

1. Compute marginal counts \( f_u(i) \) and pairwise counts \( f_{uv}(i, j) \).
2. Compute mutual information \( \hat{I}(x_i, x_j) \) for all pairs \( x_i \) and \( x_j \).
4. Set the parameters in the CPTs for each node to be their maximum likelihood estimates:

\[
P(x_i | x_{\pi(i)}) = \frac{f_{uv}(i, \pi(j))}{f_u(i)}
\]

Steps 1-3 dominate the complexity: they all take \( O(n^2) \) time.

References

• Chow, C. K. and Liu, C. N.
  “Approximating Discrete Probability Distributions with Dependence Trees”.