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 ie. 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:

- \( \mathbf{x} \) (as in bold x) is an n-dimensional vector ie. \( \mathbf{x} = (x_1, x_2, ..., x_n) \)
- Each \( x_i \) in \( \mathbf{x} \) is a variable
- \( P(\mathbf{x}) \) is a joint probability distribution of \( n \) discrete variables \( x_1, x_2, ..., x_n \)

- \( \pi(i) \) 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

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

- 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 nth-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 \( KL(P, P_\tau) \leq KL(P, P_t) \) for all \( t \in T_n \).
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 $P_t(\mathbf{x})$ is an optimum approximation to $P(\mathbf{x})$ iff its tree model has maximum weight
- Proof to follow
Proof

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

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

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

\[ - \sum_x P(x) \sum_{i=1}^n \log P(x_i) \]

Proof (continued)

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

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

\[ - \sum_x P(x) \log P(x_i) \]

\[ = - [P(x_1 = 0, x_2 = 0) \log P(x_1 = 0) + P(x_1 = 0, x_2 = 1) \log P(x_1 = 0) + P(x_1 = 1, x_2 = 0) \log P(x_1 = 1) + P(x_1 = 1, x_2 = 1) \log P(x_1 = 1)] \]

\[ = - [P(x_i = 0) \log P(x_i = 0) + P(x_i = 1) \log P(x_i = 1)] \]

\[ = - \sum_{x_i} P(x_i) \log P(x_i) = - \sum_{x_i} P(x_i) \log P(x_i) \]
Proof (continued)

In the same way:

\[
\sum_x P(x) \log \frac{P(x, x_{\pi(l)})}{P(x_l)P(x_{\pi(l)})} = \sum_{x_l, x_{\pi(l)}} P(x_l, x_{\pi(l)}) \log \frac{P(x_l, x_{\pi(l)})}{P(x_l)P(x_{\pi(l)})} = I(x_l, x_{\pi(l)})
\]

Proof (continued)

One more piece of notation:

\[
H(\mathbf{x}) = - \sum_x P(\mathbf{x}) \log P(\mathbf{x})
\]

\[
H(x_l) = - \sum_{x_l} P(x_l) \log P(x_l)
\]

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

\[
KL(P, P_t) = - \sum_{l=1}^{n} I(x_{l}, x_{\pi(l)}) + \sum_{l=1}^{n} H(x_l) - H(\mathbf{x})
\]
Proof

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

Mutual information is always \( \geq 0 \)

Independent of the dependence tree

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

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

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
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 $x^k = (x^k_1, x^k_2, \ldots, x^k_n)$.

Define:

$n_{uv}(i, j) = \# \text{ of samples with } x_i = u \text{ and } x_j = v$

$f_{uv}(i, j) = \frac{n_{uv}(i, j)}{\sum_{u,v} n_{uv}(i, j)}$  \hspace{1cm} \text{Maximum Likelihood Estimator for } P(x_i = u, x_j = v)$

$f_u(i) = \sum_v f_{uv}(i, j)$  \hspace{1cm} \text{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 \( x_j \)
3. Compute MWST using Kruskal’s algorithm.
   Pick a root, orient edges away from the root.
4. Set the parameters in the CPTs for each node to be their maximum likelihood estimates:
   \[ P(x_i \mid x_{\pi(i)}) = \frac{f_{uv}(i, \pi(j))}{f_u(i)} \]
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 \( x_j \)

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

References