CPSC 540: Machine Learning
More DAGs, Undirected Graphical Models

Mark Schmidt

University of British Columbia

Winter 2017
Admin

- **Assignment 3:**
  - 2 late days to hand in today.

- **Assignment 4:**
  - Due March 20.

- For **graduate students planning to graduate in May:**
  - Send me a private message on Piazza ASAP.
D-Separation and Plate Notation Learning and Inference in DAGs

Last Time: Directed Acyclic Graphical (DAG) Models

- **DAG** models use a factorization of the joint distribution,

\[ p(x_1, x_2, \ldots, x_d) = \prod_{j=1}^{d} p(x_j | x_{\text{pa}(j)}) , \]

where \( \text{pa}(j) \) are the **parents** of node \( j \).

- This assumes a **Markov property**,

\[ p(x_j | x_{1:j-1}) = p(x_j | x_{\text{pa}(j)}) , \]

which generalizes the Markov property in Markov chains,

\[ p(x_j | x_{1:j-1}) = p(x_j | x_{j-1}) . \]
DAG models use a factorization of the joint distribution,

\[ p(x_1, x_2, \ldots, x_d) = \prod_{j=1}^{d} p(x_j | x_{\text{pa}(j)}) , \]

where \( \text{pa}(j) \) are the parents of node \( j \).

We visualize the assumptions made by the model as a graph:

- Structure determines conditional independences and computational tractability.
Outline

1. D-Separation and Plate Notation
2. Learning and Inference in DAGs
3. Undirected Graphical Models
D-Separation

We say that \( A \) and \( B \) are d-separated (conditionally independent) if all paths \( P \) from \( A \) to \( B \) are “blocked” because at least one of the following holds:

1. \( P \) includes a “chain” with an observed middle node (e.g., Markov chain):

   ![Diagram 1]

2. \( P \) includes a “fork” with an observed parent node (e.g., mixture model):

   ![Diagram 2]

3. \( P \) includes a “v-structure” or “collider” (e.g., factor analysis):

   ![Diagram 3]

where “child” and all its descendants are unobserved.
Alarm Example

- Earthquake $\not\perp$ Call.
- Earthquake $\perp$ Call | Alarm.
- Alarm $\not\perp$ Stuff Missing.
- Alarm $\perp$ Stuff Missing | Burglary.
Alarm Example

- Earthquake $\perp$ Burglary.
- Earthquake $\not\perp$ Burglary $|$ Alarm.
  - **Explaining away:** Knowing Earthquake would make Burglary is less likely.
- Call $\not\perp$ Stuff Missing.
- Earthquake $\perp$ Stuff Missing.
- Earthquake $\not\perp$ Stuff Missing $|$ Call.
Discussion of D-Separation

- D-separation lets you say if **conditional independence is implied** by assumptions:

\[(A \text{ and } B \text{ are d-separated given } E) \Rightarrow A \perp B \mid E.\]

- However, there **might be extra conditional independences** in the distribution:
  - These would depend on specific choices of the \(p(x_j \mid x_{pa(j)})\).
  - Or some orderings may reveal extra independences....

- Instead of restricting to \(\{1, 2, \ldots, j - 1\}\), consider **general parent choices**.
  - \(x_2\) could be a parent of \(x_1\).

- As long the **graph is acyclic**, there exists a valid ordering.
  
  \text{(all DAGs have a “topological order” of variables where parents are before children)}
Note that some graphs imply *same conditional independences*:

- **Equivalent** graphs: same \(v\)-structures and other (undirected) edges are the same.
- Examples of 3 *equivalent* graphs (left) and 3 non-equivalent graphs (right):
Discussion of D-Separation

- So the graph is not necessarily unique and is not the whole story.

- But, we can do a lot with d-separation:
  - Implies every independence/conditional-independence we've used in 340/540.

- Here we start blurring distinction between data/parameters/hyper-parameters...
On Day 2, our first independence assumption was the **IID assumption**:

Training/test examples come independently from data-generating process $D$.

If we knew $D$, then there would be no need to learn.

But $D$ is unobserved, so knowing about some $x^i$ tells us about the others.

We’ll use this understanding later to relax the IID assumption.
Plate Notation

- Graphical representation of the IID assumption:

- We can concisely represent repeated parts of graphs using plate notation:
Tilde Notation as a DAG

- When we write
  \[ y^i \sim \mathcal{N}(w^T x^i, 1), \]
  we can interpret it as the DAG model:

- If the \( x^i \) are IID then we can represent supervised learning as

- From \( d \)-separation on this graph we have
  \[ p(y|X, w) = \prod_{i=1}^{n} p(y^i|x^i, w). \]
Tilde Notation as a DAG

- When we do MAP estimation under the assumptions
  \[ y^i \sim \mathcal{N}(w^T x^i, 1), \quad w_j \sim \mathcal{N}(0, 1/\lambda), \]
  we can interpret it as the DAG model:

- Or introducing a second plate using:
Other Models in DAG/Plate Notation

- For naive Bayes or Gaussian discriminant analysis with diagonal $\Sigma_c$ we have

$$y^i \sim \text{Cat}(\theta), \quad x^i | y^i = c \sim D(\theta_c).$$

- Or in plate notation as
Other Models in DAG/Plate Notation

- In a full Gaussian model for a single $x$ we have
  \[ x^i \sim \mathcal{N}(\mu, \Sigma). \]

- For mixture of Gaussians we have
  \[ z^i \sim \text{Cat}(\theta), \quad x^i | z^i = c \sim \mathcal{N}(\mu_c, \Sigma_c). \]
Outline

1. D-Separation and Plate Notation
2. Learning and Inference in DAGs
3. Undirected Graphical Models
Parameter Learning in General DAG Models

- The log-likelihood in DAG models is **separable** in the conditionals,

\[
\log p(x|\Theta) = \log \prod_{j=1}^{d} p(x_j|x_{pa}(j), \Theta_j)
\]

\[
= \sum_{j=1}^{d} \log p(x_j|x_{pa}(j), \Theta_j)
\]

- If each \(p(x_j|x_{pa}(j))\) has its own parameters \(\Theta_j\), we can **fit them independently**.
  - We’ve done this before: naive Bayes, Gaussian discriminant analysis, etc.

- Sometimes you want to have **tied parameters** \((\Theta_j = \Theta_{j'})\)
  - Homogeneous Markov chains, Gaussian discriminant analysis with shared covariance.
  - Still easy, but need to fit \(p(x_j|x_{pa}(j), \Theta_j)\) and \(p(x_{j'}|x_{pa}(j'), \Theta_j)\) together.
Tabular Parameterization in DAG Models

- To specify distribution, we need to decide on the form of \( p(x_j|x_{pa(j)}, \Theta_j) \).

- For discrete data a default choice is the tabular parameterization:
  \[
  p(x_j|x_{pa(j)}, \Theta_j) = \theta_{x_j,x_{pa(j)}},
  \]
  as we did for Markov chains (but now with multiple parents).

- Intuitive: just need conditional probabilities of children given parents like
  \[
  p(\text{“wet grass”} = 1 \mid \text{“sprinkler”} = 1, \text{“rain”} = 0),
  \]
  and MLE is just counting.
Tabular Parameterization Example

Some quantities can be directly read from the tables:

\[ p(R = 1) = 0.2. \]

\[ p(G = 1|S = 0, R = 1) = 0.8. \]
Can calculate any probabilities using marginalization/product-rule/Bayes-rule.

\[
p(G = 1|R = 1) = p(G = 1, S = 0|R = 1) + p(G = 1, S = 1|R = 1) = \sum_b p(a|c) = \sum_b p(a, b|c)
\]

\[
= p(G = 1|S = 0, R = 1)p(S = 0|R = 1) + p(G = 1|S = 1, R = 1)p(S = 1|R = 1) = 0.8(0.99) + 0.99(0.01) = 0.81.
\]
Tabular Parameterization Example

Some companies sell software to help companies reason using tabular DAGs:

http://www.hugin.com/index.php/technology
Fitting DAGs using Supervised Learning

- But tabular parameterization requires **too many parameters**:
  - With binary states and \( k \) parents, need \( 2^{k+1} \) parameters.

- One solution is letting users specify a “parsimonious” parameterization:
  - Typically have a linear number of parameters.
  - For example, the “noisy-or” model: 
    
    \[
    p(x_j | x_{pa(j)}) = 1 - \prod_{k \in pa(j)} q_k.
    \]

- But if we have data, we can use **supervised learning**.
  - Write fitting \( p(x_j | x_{pa(j)}) \) as our usual \( p(y | x) \).
  - We’re predicting one column of \( X \) given the values of other columns.
Fitting DAGs using Supervised Learning

- Fitting DAGs using **supervised learning**:
  - For $j = 1 : d$:
    1. Set $\tilde{y}^i = x^i_j$ and $\tilde{x}^i = x^i_{pa(j)}$.
    2. Solve a supervised learning problem using $\{\tilde{X}, \tilde{y}\}$.
  - Use the $d$ regression/classification models as the density estimator.

- We can use our usual tricks:
  - Linear models, non-linear bases, regularization, kernel trick, random forests, etc.
  - With least squares it's called a **Gaussian belief network**.
  - With logistic regression it's called a **sigmoid belief networks**.
  - Don’t need Markov assumptions to tractably fit these models.
Recall our latest MNIST model using a **tabular DAG**:

- This model is pretty bad because you only see 8 parents.
MNIST Digits with Sigmoid Belief Network

- Samples from sigmoid belief network:

  (DAG with logistic regression for each variable)

where we use all previous pixels as parents (from 0 to 783 parents).
- Models long-range dependencies but has a linear assumption.
Sampling in DAGs

- We can use **ancestral sampling** to generate samples from a DAG:
  1. Sample $x_1$ from $p(x_1)$.
  2. If $x_1$ is a parent of $x_2$, sample $x_2$ from $p(x_2|x_1)$.
     - Otherwise, sample $x_2$ from $p(x_2)$.
  3. Go through the subsequent $j$ in order sampling $x_j$ from $p(x_j|x_{pa(j)})$.

- We can use these samples within **Monte Carlo** methods.

- How do **sample from a multivariate Gaussian**?
  - Write it as a Gaussian belief network, apply ancestral sampling.
Inference in Forest DAGs

- If we try to generalize the CK equations to DAGs we obtain

\[ p(x_j = s) = \sum_{x_{pa}(j)} p(x_j = s, x_{pa}(j)) = \sum_{x_{pa}(j)} p(x_j = s | x_{pa}(j)) p(x_{pa}(j)). \]

which works if each node has at most one parent.
- Such graphs are called trees (connected), or forests (disconnected).
  - Also called “singly-connected”.
- Forests allow efficient message-passing methods as in Markov chains.
  - In particular, decoding and univariate marginals/conditionals in \( O(dk^2) \).
  - Message passing applied to tree-structured graphs is called belief propagation.
Inference in General DAGs

- If we try to generalize the CK equations to DAGs we obtain

\[ p(x_j = s) = \sum_{x_{pa(j)}} p(x_j = s, x_{pa(j)}) = \sum_{x_{pa(j)}} p(x_j = s | x_{pa(j)}) p(x_{pa(j)}). \]

- What goes wrong if nodes have multiple parents?
  - The expression \( p(x_{pa(j)}) \) is a joint distribution and is not given recursively.

- Consider the non-tree graph:
**Inference in General DAGs**

- We can compute $p(x_4)$ in this non-tree using:

$$p(x_4) = \sum_{x_3} \sum_{x_2} \sum_{x_1} p(x_1, x_2, x_3, x_4)$$

$$= \sum_{x_3} \sum_{x_2} \sum_{x_1} p(x_4|x_2, x_3)p(x_3|x_1)p(x_2|x_1)p(x_1)$$

$$= \sum_{x_3} \sum_{x_2} p(x_4|x_2, x_3) \sum_{x_1} p(x_3|x_1)p(x_2|x_1)p(x_1) \underbrace{M_{23}(x_2, x_3)}_{x_1}$$

- Dependencies between $\{x_1, x_2, x_3\}$ mean our message depends on two variables.

$$p(x_4) = \sum_{x_3} \sum_{x_2} p(x_4|x_2, x_3)M_{23}(x_2, x_3)$$

$$= \sum_{x_3} M_{34}(x_3, x_4),$$
Inference in General DAGs

- With 2-variable messages, our cost increases to $O(dk^3)$.

- If we add the edge $x_1 \rightarrow x_4$, then the cost is $O(dk^4)$.
  (the same cost as enumerating all possible assignments)

- Unfortunately, cost is not as simple as counting number of parents.
  - Even if each node has 2 parents, we may need huge messages.
  - Decoding is NP-hard and marginals are #P-hard in general.

- We’ll see later that maximum message is given by treewidth of a particular graph.

- In general, we’ll need approximate inference methods to use general DAGs.
Conditional Sampling in DAGs

- What about **conditional sampling** in DAGs?
  - Could be easy or hard depending on what we condition on.
  - For example, still **easy if we condition on the first variables in the order**:
    - Just fix these and run ancestral sampling.

- **Hard to condition on the last variables in the order**:
  - Conditioning on descendent makes ancestors dependent.
Outline

1. D-Separation and Plate Notation
2. Learning and Inference in DAGs
3. Undirected Graphical Models
Directed vs. Undirected Models

- In some applications we have a **natural ordering** of the $x_j$.
  - In the “rain” data, the past affects the future.

- In some applications we **don’t have a natural order**.
  - E.g., pixels in an image.

- In these settings we often use **undirected graphical models**.
  - Also known as **Markov random fields** and originally from statistical physics.
Undirected Graphical Models

- Undirected graphical models (UGMs) assume \( p(x) \) factorizes over subsets \( c \),

\[
p(x_1, x_2, \ldots, x_d) \propto \prod_{c \in C} \phi_c(x_c),
\]

from among a set of subsets of \( C \).

- The \( \phi_c \) are called potential functions: can be any non-negative function.
  - Ordering doesn’t matter: more natural for things like pixels of an image.
  - Theoretically, only need \( \phi_c \) for maximal subsets in \( C \).

- Important special case is pairwise undirected graphical model:

\[
p(x) \propto \left( \prod_{j=1}^{d} \phi_j(x_j) \right) \left( \prod_{(i,j) \in E} \phi_{ij}(x_i, x_j) \right),
\]

where \( E \) are a set of undirected edges.
Undirected Graphical Models

- Pairwise UGMs are a classic way to model dependencies in images:

- Can model dependency between neighbouring pixels, without imposing ordering.
From Probability Factorization to Graphs

For a pairwise UGM,

\[ p(x) \propto \left( \prod_{j=1}^{d} \phi_j(x_j) \right) \left( \prod_{(i,j) \in E} \phi_{ij}(x_i, x_j) \right), \]

we visualize independence assumptions as an undirected graph:
- We have edge \( i \) to \( j \) if \( (i, j) \in E \).

For general UGMs,

\[ p(x_1, x_2, \ldots, x_d) \propto \prod_{c \in C} \phi_c(x_c), \]

we have the edge \( (i, j) \) if \( i \) and \( j \) are together in at least one \( c \).
Conditional Independence in Undirected Graphical Models

- It’s easy to check **conditional independence** in UGMs:
  - \( A \perp B \mid C \) if \( C \) blocks all paths from any \( A \) to any \( B \).

- Example:

\[
\begin{align*}
A & \not\perp C, \\
A & \not\perp C \mid B, \\
A & \perp C \mid B, E, \\
A, B & \not\perp F \mid C, \\
A, B & \perp F \mid C, E.
\end{align*}
\]
Multivariate Gaussian and Pairwise Graphical Models

- **Multivariate Gaussian** is a special case of a pairwise UGM.

- Edges of the graph are \((i, j)\) values where \(\Sigma_{ij}^{-1} \neq 0\).

- Unconditional independence of \((i, j)\) corresponds to having \(\Sigma_{ij} = 0\).
  - Can be seen from block Gaussian formula.
  - Corresponds to reachability in the graph.

- We use the term **Gaussian graphical model** (GGM) in this context.
  - Or Gaussian Markov random field.
Digression: Gaussian Graphical Models

- Multivariate Gaussian can be written as

\[
p(x) \propto \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right) \propto \exp \left( -\frac{1}{2} x^T \Sigma^{-1} x + x^T \underbrace{\Sigma^{-1} \mu}_v \right),
\]

and from here we can see that it’s a pairwise UGM:

\[
p(x) \propto \exp \left( -\frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j \Sigma_{ij}^{-1} + \sum_{i=1}^{d} x_i v_i \right)
= \prod_{i=1}^{d} \prod_{j=1}^{d} \exp \left( -\frac{1}{2} x_i x_j \Sigma_{ij}^{-1} \right) \phi_{ij}(x_i, x_j) \prod_{i=1}^{d} \exp \left( x_i v_i \right) \phi_i(x_i)
\]
Independence in GGMs

- So Gaussians are pairwise UGMs with \( \phi_{ij}(x_i, x_j) = \exp \left( -\frac{1}{2} x_i x_j \Theta_{ij} \right) \),
  - Where \( \Theta_{ij} \) is element \((i, j)\) of \( \Sigma^{-1} \).

- Connection between precision matrix \( \Theta = \Sigma^{-1} \) and conditional independence:
  - Setting \( \Theta_{ij} = 0 \) is equivalent to removing \( \phi_{ij}(x_i, x_j) \) from the UGM.
  \[
  \Theta_{ij} \neq 0 \Rightarrow x_i \not\perp x_j | x_{-ij}.
  \]

- Gaussian conditional independencies corresponds to sparsity in precision matrix.
  - Diagonal \( \Theta \) gives disconnected graph: all variables are independent.
  - Full \( \Theta \) gives fully-connected graph: there are no independences.
Independence in GGMs

- Consider Gaussian with **tri-diagonal precision** $\Theta$:

  $\Sigma^{-1} = \begin{bmatrix}
  32.0897 & 13.1740 & 0 & 0 & 0 \\
  13.1740 & 18.3444 & -5.2602 & 0 & 0 \\
  0 & -5.2602 & 7.7173 & 2.1597 & 0 \\
  0 & 0 & 2.1597 & 20.1232 & 1.1670 \\
  0 & 0 & 0 & 1.1670 & 3.8644
  \end{bmatrix}$

  $\Sigma = \begin{bmatrix}
  0.0494 & -0.0444 & -0.0312 & 0.0034 & -0.0010 \\
  -0.0444 & 0.1083 & 0.0761 & -0.0083 & 0.0025 \\
  -0.0312 & 0.0761 & 0.1872 & -0.0204 & 0.0062 \\
  0.0034 & -0.0083 & -0.0204 & 0.0528 & -0.0159 \\
  -0.0010 & 0.0025 & 0.0062 & -0.0159 & 0.2636
  \end{bmatrix}$

  - $\Sigma_{ij} \neq 0$ so all variables are dependent: $x_1 \perp x_2$, $x_1 \perp x_5$, and so on.
  - But **conditional independence** is described by a Markov chain:

    \[ p(x_1|x_2, x_3, x_4, x_5) = p(x_1|x_2). \]
Graphical Lasso

- Conditional independence in GGMs is described by sparsity in $\Theta$.

- Recall fitting multivariate Gaussian with L1-regularization,

$$\arg\min_{\Theta > 0} \text{Tr}(S\Theta) - \log |\Theta| + \lambda \|\Theta\|_1,$$

which is called the graphical Lasso because it encourages a sparse graph.
  - Special case of graph structure learning.

- Consider instead fitting DAG model with Gaussian probabilities:
  - DAG structure corresponds to sparsity in Cholesky of covariance.
Tractability of UGMs

- In UGMs we assume that
  \[ p(x) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c), \]
  where \( Z \) is the constant such that
  \[ \sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} p(x) = 1 \text{ (discrete), } \int_{x_1} \int_{x_2} \cdots \int_{x_d} p(x) dx_d dx_{d-1} \cdots dx_1 = 1 \text{ (cont).} \]

- So \( Z \) is
  \[ Z = \sum_{x} \prod_{c \in C} \phi_c(x_c) \text{ (discrete), } \int x \prod_{c \in C} \phi_c(x_c) dx \text{ (cont)} \]

- Whether you can compute \( Z \) depends on the choice of \( \phi_c \):
  - Gaussian case: \( O(d^3) \) in general, but \( O(d) \) for forests (no loops).
  - Discrete case: \#P-hard in general, but \( O(dk^2) \) for forests (no loops).
  - Continuous non-Gaussian: usually requires numerical integration.
Summary

- **Plate Notation** lets compactly draw graphs with repeated patterns.
  - There are fancier versions of plate notation called “probabilistic programming”.

- **Parameter learning in DAGs:**
  - Can fit each $p(x_j|x_{pa(j)})$ independently.
  - Tabular parameterization, or treat as supervised learning.

- **Inference in DAGs:**
  - Ancestral sampling and Monte Carlo methods work as faster.
  - Message-passing message sizes depend on graph structure.

- **Undirected graphical models** factorize probability into non-negative potentials.
  - Simple conditional independence properties.
  - Include Gaussians as special case.

- Next time: our first visit to the wild world of approximate inference.