CPSC 440: Advanced Machine Learning
Learning Graphical Models

Mark Schmidt

University of British Columbia

Winter 2022
Last Time: D-Separation

**D-separation** can be used to “read” conditional independence from graph.
- Can be derived by considering DAG as “inheritance of genes”.

3 Cases that can “block” a path between nodes:

- **Case 1:** Observing a variable in a “chain” blocks a path.
- **Case 2:** Observing a parent in a “fork” blocks a path.
- **Case 3:** Not observing a child in a “v-structure” blocks a path.

We say that variables are “d-separated” if every path between them is blocked.
Discussion of D-Separation

- Last time we discussed equivalent graphs (same independent assumptions).
- We also discussed how extra independence assumptions may not appear in graph.

- 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 340/440.
    - Except for multivariate Gaussians, which are undirected graphical models.

- Requires using DAGs to represent relationships between data and parameters.
  - This allows us to see and reason about their relationships.
Tilde Notation as a DAG

- When we write
  \[ y^i \sim \mathcal{N}(w^T x^i, 1), \]
  this can be interpreted as a DAG model:

  ![Diagram](image)

  - “The variables on the right of \( \sim \) are the parents of the variables on the left”.
  - We can see our standard \( X \perp w \) assumption in the graph.
    - Common child case: \( w \) only depends on \( X \) if we know \( y \).
During week 1, our first independence assumption was the IID assumption:

- Training/test examples come independently from data-generating process $D$.
- But $D$ is unobserved, so knowing about some $x^i$ tells us about the others.
  - This why the IID assumptions lets us learn.
Plate Notation

- Graphical representation of the IID assumption:

- It’s common to represent repeated parts of graphs using plate notation:
If the $x^i$ are IID then we can represent linear regression as

![Diagram showing plate notation]

or

From $d$-separation on this graph we have $p(y \mid X, w) = \prod_{i=1}^{n} p(y^i \mid x^i, w)$.

- Our standard assumption that data is independent given parameters.

- We often omit the data-generating distribution $D$.
  - But if you want to learn then you should remember that it’s there.

- Note that plate reflects parameter tieing: that we use same $w$ for all $i$. 
IID Bernoulli-Beta Model

- The Bernoulli-beta model as a DAG (with parameters and hyper-parameters):

- Notice data is independent of hyper-parameters given parameters.
  - This is another of our standard independence assumptions.
Non-IID Bernoulli-Beta Model

- The non-IID variant we considered with grouped data:

- DAG reflects that we do not tie parameters across all training examples.
- Notice that if you fix $\alpha$ and $\beta$ then you can’t learn across groups:
  - The $\theta_j$ are d-separated given $\alpha$ and $\beta$. 
Non-IID Bernoulli-Beta Model

- Variant of the previous model with a hyper-hyper-parameter:

- Which is needed to avoid degeneracy.
Bayesian Linear Regression as a DAG

- In Bayesian linear regression we assume
  \[ y^i \sim \mathcal{N}(w^T x^i, 1), \quad w_j \sim \mathcal{N}(0, 1/\lambda), \]
  which we can interpret it as the DAG model:

- Or introducing a second plate over parameters:
Outline

1. Plate Notation

2. Graphical Model Learning and Inference
Density Estimators vs. Relationship Visualizers

- Besides dependency visualization, we can use DAGs as density estimators.

- Recall that DAGs model joint distribution using

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

- We need to choose a parameterization for these conditional probabilities:
  - **Tabular** parameterization (discrete \( x_j \)): can model any joint probability.
    - Common choice, sometimes setting parameters using expert knowledge.
  - **Gaussian** (continuous \( x_j \)): \( x_j \sim \mathcal{N}(w^T x_{pa(j)}, \sigma^2) \).
    - Called a Gaussian belief net. Joint distribution becomes a multivariate Gaussian.
  - **Sigmoid** (binary \( x_j \in \{-1, +1\} \)): \( p(x_j \mid x_{j-1}, w) = \frac{1}{1 + \exp(-x_j w^T x_{pa(j)})} \).
    - Called a sigmoid belief net.
  - Could use softmax, probabilistic random forest, neural network, and so on.
    - Our tricks for probabilistic supervised learning can be used for unsupervised learning.
Tabular Parameterization Example

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

http://www.hugin.com/index.php/technology
DAG Learning and Sampling

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 $\{\tilde{X}, \tilde{y}\}$.
   - Gives you a model of $p(x_j | x_{pa(j)})$.

Can sample from DAGs using ancestral sampling:

- Sample $x_1$ from $p(x_1)$.
- Sample $x_2$ from $p(x_2 | x_{pa(2)})$.
- Sample $x_3$ from $p(x_3 | x_{pa(3)})$.
- Until we have sampled $x_d$.

This allows us to do inference with Monte Carlo methods.
  - Conditional sampling can be hard, may need rejection sampling for conditionals.
MNIST Digits with Tabular DAG Model

- Recall our latest MNIST model using a tabular DAG:

  ![MNIST Digits](image)

- 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.
Exact Inference in DAGs?

- Can we do exact inference in DAGs like in Markov chains?

- Continuous-state Gaussian DAGs:
  - Special case of multivariate Gaussian, so inference is tractable.
    - Most operations are $O(d)$ or $O(d^3)$.

- Continuous-state non-Gaussian DAGs:
  - Inference usually not closed-form, so need Monte Carlo or variational inference.
  - If parents are conjugate, then Gibbs sampling is easy to implement.

- Discrete-state DAGs (whether tabular or sigmoid or other):
  - Inference takes exponential-time in the “treewidth” of the graph.
  - Exact inference is cheap in trees and forests, which have a treewidth of 1.
    - Low-treewidth graphs allow efficient exact inference, otherwise need approximations.
Inference in Forest DAGs ("Belief Propagation")

- Graphs with at most one parent per node are called trees (connected).
  - Also called “singly-connected” or forests (if disconnected).
  - We can generalize the CK equations to trees/forests:
    \[
    p(x_j = s) = \sum_{x_{pa(j)}} p(x_j = s, x_{pa(j)}) = \sum_{x_{pa(j)}} p(x_j = s \mid x_{pa(j)}) p(x_{pa(j)}).
    \]

- Trees/forests allow efficient dynamic programming methods as in Markov chains.
  - In particular, decoding and univariate marginals/conditionals in $O(dk^2)$.
  - Forward-backward applied to tree-structured graphs is called belief propagation.
  - It also possible to find optimal tree given data ("structure learning").
Undirected Graphical Models (UGMs)

- Undirected graphical models (UGMs) are another popular graphical model class. UGMs are also known as Markov random fields.

- UGMs define joint distribution in terms of non-negative potential functions,

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

- Define potential \( \phi_c \) for each set \( c \) where we want to model a direct relationship.

- The most common choice is a pairwise UGM,

\[
p(x_1, x_2, \ldots, x_d) \propto \left( \prod_{j=1}^{d} \phi_j(x_j) \right) \left( \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j) \right).
\]

which only has potentials on single variables (\( \phi \)) and pairs of variables (\( \psi \)).

- The “edge potentials” \( \psi \) are defined on edge of an undirected graph \( \mathcal{E} \).
Pairwise Undirected Graphical Models

- Pairwise undirected graphical models factorize probability using

$$p(x_1, x_2, \ldots, x_d) \propto \left( \prod_{j=1}^{d} \phi_j(x_j) \right) \left( \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j) \right).$$

- Special cases:
  - Markov chains are the special case $E$ only has edges between adjacent nodes.
  - Multivariate Gaussian corresponds to specific choice of the $\phi$ and $\psi$ functions.
    - Gaussians AKA “Gaussian graphical models” or “Gaussian Markov random fields”.
  - Ising model for binary $x_j$ uses
    $$\phi_j(x_j) = \exp(x_i w_i), \quad \phi_{ij}(x_i, x_j) = \exp(x_i x_j w_{ij}),$$

where $w_i$ is the node weight and $w_{ij}$ is the edge weight.
  - If $w_{ij} > 0$ it encourages neighbours to have same value (“attractive”).
  - If $w_{ij} < 0$ it encourages neighbours to have different values (“repulsive”).
Conditional Independence in UGMs

- UGM independence properties described by an undirected graph.
  - For pairwise UGMs, the edges are given by the set of edges $\mathcal{E}$.

![Diagram of UGM nodes: A, B, C, D, E, F]

- Constructing graph when you may have 3-variable or higher-order potentials:
  - Graph has edge $(i,j)$ if $i$ and $j$ are together in at least one $c$.

- So these two factorizations have the same graph:

  $$p(x_1, x_2, x_3) \propto \phi_{12}(x_1, x_2)\phi_{13}(x_1, x_3)\phi_{23}(x_2, x_3), \quad p(x_1, x_2, x_3) \propto \phi_{123}(x_1, x_3, x_3).$$

- UGM implies $A \perp B \mid C$ if $C$ separates all nodes in $A$ from all nodes in $B$.
  - Same rule as a Gaussians.
DAGs vs. UGMs

Neither DAGs or UGMs are “more powerful” than the other.

- Any distribution can be re-written as a DAG or UGM.
- But may need to use a highly-connected graph.

Set of independences in DAG cannot always be written as UGM (and vice versa).

- UGMs cannot reflect independences in common child graph: \((x) \rightarrow (y) \leftarrow (z)\).
- DAGs cannot reflect independences in 4-node loop: \((x) - (y) - (z) - (x)\).
- Independences representable as both DAGs and UGMs are called decomposable.
  - An example is Markov chains: independences are same in DAG and UGM graphs.

DAGs are often used when it makes sense to work with conditionals, or we have an idea of causal directions.

UGMs are often used when there are no obvious directions (like MNIST), and are more-often used when we want to add features to do supervised learning.
Tractability of UGMs

Without using $\propto$, a UGM probability would be

$$p(x) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c),$$

where $Z$ is the constant that makes the probabilities sum up to 1.

$$Z = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} \prod_{c \in C} \phi_c(x_c) \quad \text{or} \quad Z = \int_{x_1} \int_{x_2} \cdots \int_{x_d} \prod_{c \in C} \phi_c(x_c) dx_d dx_{d-1} \cdots dx_1.$$

Whether you can compute $Z$ (and do inference) depends on the choice of the $\phi_c$:

- Gaussian case: $O(d^3)$ in general, but $O(d)$ for forests (no loops).
- Continuous non-Gaussian: usually requires approximate inference.
- Discrete case: $\#P$-hard in general, but $O(dk^2)$ for forests (no loops).
Discrete DAGs vs. Discrete UGMs

Common inference tasks in graphical models:

1. Compute $p(x)$ for an assignment to the variables $x$.
2. Generate a sample $x$ from the distribution.
3. Compute univariate marginals $p(x_j)$.
4. Compute decoding $\text{argmax}_x p(x)$.
5. Compute univariate conditional $p(x_j \mid x_{j'})$.

With discrete $x_i$, all of the above are easy in tree-structured graphs.

- For DAGs, a tree-structured graph has at most one parent.
- For UGMs, a tree-structured graph has no cycles.

With discrete $x_i$, the above may be harder for general graphs:

- In DAGs the first two are easy, the others are NP-hard.
- In UGMs all of these are NP-hard.
Bonus Slides on Inference in UGMs

- I am not planning to go into details on inference in graphical models.
- I put all my slides on the topic available here:
- Covers topics such as:
  - Inference in non-tree DAGs/UGMs.
  - Learning the graph structure.
  - Writing Gaussians as pairwise UGMs.
  - Treewidth of graphs, and efficient inference with low treewidth.
  - Exact decoding for binary attractive models using graph cuts.
  - ICM and alpha-expansion algorithms for approximate decoding.
  - Block Gibbs sampling in UGMs (UGMs are what Gibbs sampling was invented for).
Summary

- Independence assumptions about data and parameters can be written as DAGs.
- **Plate Notation** lets us compactly draw graphs with repeated patterns.
  - There are fancier versions of plate notation called “probabilistic programming”.
- **Parameter learning in DAGs:**
  - Can fit each $p(x_j \mid x_{pa(j)})$ independently.
  - Tabular parameterization, or treat as supervised learning.
- **Sampling in DAGs** is easy (ancestral sampling).
- **Exact inference in discrete DAGs** is easy for trees.
  - But becomes exponential in “treewidth” of graph.
- **Undirected graphical models** factorize probability into non-negative potentials.
  - Gaussians are a special case, but can place potentials on any subset of variables.
  - Inference is again exponential in “treewidth” of graph.

- Next time: adding graphical models to neural networks.
Naive Bayes with DAGs/Plates

- For naive Bayes we have
  \[ y^i \sim \text{Cat}(\theta), \quad x^i \mid y^i = c \sim \text{Cat}(\theta_c). \]

- Or in plate notation as
Does Semi-Supervised Learning Make Sense?

- Should unlabeled examples always help supervised learning?
  - No!

- Consider choosing unlabeled features $\bar{x}^i$ uniformly at random.
  - Unlabeled examples collected in this way will not help.
  - By construction, distribution of $\bar{x}^i$ says nothing about $\bar{y}^i$.

- Example where SSL is not possible:
  - Try to detect food allergy by trying random combinations of food:
    - The actual random process isn’t important, as long as it isn’t affected by labels.
    - You can sample an infinite number of $\bar{x}^i$ values, but they says nothing about labels.

- Example where SSL is possible:
  - Trying to classify images as “cat” vs. “dog.:
    - Unlabeled data would need to be images of cats or dogs (not random images).
    - Unlabeled data contains information about what images of cats and dogs look like.
    - For example, there could be clusters or manifolds in the unlabeled images.
Does Semi-Supervised Learning Make Sense?

Let’s assume our semi-supervised learning model is represented by this DAG:

Assume we observe \( \{X, y, \bar{X}\} \) and are interested in test labels \( \tilde{y} \):

- There is a dependency between \( y \) and \( \tilde{y} \) because of path through \( w \).
  - Parameter \( w \) is tied between training and test distributions.
  - There is a dependency between \( X \) and \( \tilde{y} \) because of path through \( w \) (given \( y \)).
    - But note that there is also a second path through \( D \) and \( \tilde{X} \).
  - There is a dependency between \( \bar{X} \) and \( \tilde{y} \) because of path through \( D \) and \( \tilde{X} \).
- Unlabeled data helps because it tells us about data-generating distribution \( D \).
Does Semi-Supervised Learning Make Sense?

Now consider generating $\bar{X}$ independent of $D$:

Assume we observe $\{X, y, \bar{X}\}$ and are interested in test labels $\tilde{y}$:

- Knowing $X$ and $y$ are useful for the same reasons as before.
- But knowing $\bar{X}$ is not useful:
  - Without knowing $\tilde{y}$, $\bar{X}$ is $d$-separated from $\tilde{y}$ (no dependence).
Some quantities can be directly read from the tables:

$$p(R = 1) = 0.2.$$  

$$p(G = 1 \mid S = 0, R = 1) = 0.8.$$  

Can calculate any probabilities using marginalization/product-rule/Bayes-rule (bonus).
Can calculate any probabilities using marginalization/product-rule/Bayes-rule, for example:

\[
p(G = 1 \mid R = 1) = p(G = 1, S = 0 \mid R = 1) + p(G = 1, S = 1 \mid R = 1) = p(G = 1 \mid S = 0, R = 1)p(S = 0 \mid R = 1) + p(G = 1 \mid S = 1, R = 1)p(S = 1 \mid R = 1) = 0.8(0.99) + 0.99(0.01) = 0.81.
\]
**Dynamic Bayesian Networks**

- **Dynamic Bayesian networks** combine ideas from DAGs and Markov chains:
  - At each time, we have a set of variables $x^t$.
  - The initial $x^0$ comes from an “initial” DAG.
  - Given $x^{t-1}$, we generate $x^t$ from a “transition” DAG.

![Diagram of Dynamic Bayesian Networks](https://www.cs.ubc.ca/~murphyk/Papers/dbnsem_uai98.pdf)

- Can be used to model multiple variables over time.
  - Unconditional sampling is easy but inference may be hard.