CPSC 540: Machine Learning

Undirected Graphical Models

Mark Schmidt

University of British Columbia

Winter 2016

Admin

- Assignment 3:
  - 2 late days to hand it in today, Thursday is final day.
- Assignment 4:
  - Due in 1 week.
  - You can only use 1 late day on this assignment.
- Midterm;
  - March 17 in class.
  - Closed-book, two-page double-sided 'cheat sheet'.
  - Only covers topics from assignments 1-4.
  - No requirement to pass.
  - Midterm from last year posted on Piazza.

Last Time: Directed Acyclic Graphical Models

- DAG models use a factorization of the joint distribution,

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

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

Last Time: Directed Acyclic Graphical Models

- DAG models use a factorization of the joint distribution,

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

  where $\mathsf{pa}(j)$ are the parents of node j.
- We represent this factorization using a directed graph on d nodes:
  - We have an edge $i \to j$ if i is parent of j.
- D-separation says whether conditional independence is implied by graph:
  1. Path $i \to j \to k$ is blocked if j is observed.
  2. Path $i \leftarrow j \to k$ is blocked if j is observed.
  3. Path $i \to j \leftarrow k$ is blocked if j and none of its descendants observed.
- D-separation is also known as "Bayes ball".

Non-Uniquness of Graph and Equivalent Graphs

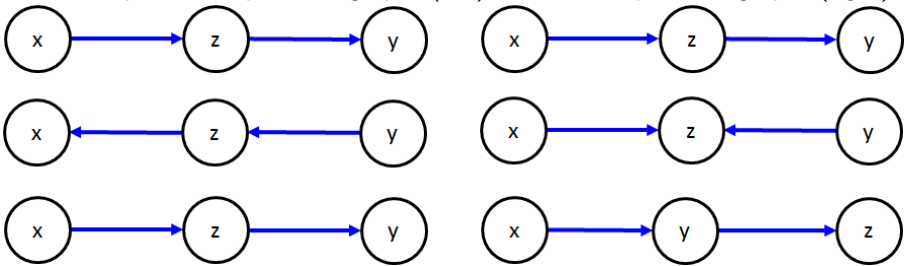
- Last time we assumed that parents of j are selected from $\{1, 2, \ldots, j-1\}$.
  - But this means the factorization depends on variable order.

Non-Uniquness of Graph and Equivalent Graphs

- Last time we assumed that parents of j are selected from $\{1, 2, \ldots, j-1\}$.
  - But this means the factorization depends on variable order.
- We could instead select parents of j from $\{1, 2, \ldots, j-1, j+1, j+2, \ldots, d\}$,
  - Gives valid factorization as long as graph is acyclic.
  - Acyclicity implies that a "topological order" order exists.
    - We could reorder so that parents come before children.

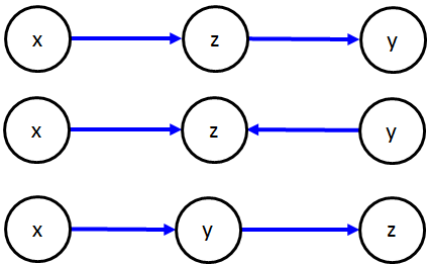
Non-Uniquness of Graph and Equivalent Graphs

- Last time we assumed that parents of j are selected from $\{1, 2, \ldots, j-1\}$.
  - But this means the factorization depends on variable order.
- We could instead select parents of j from $\{1, 2, \ldots, j-1, j+1, j+2, \ldots, d\}$,
  - Gives valid factorization as long as graph is acyclic.
  - Acyclicity implies that a "topological order" order exists.
    - We could reorder so that parents come 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):



Non-Uniquness of Graph and Equivalent Graphs

- Last time we assumed that parents of j are selected from $\{1, 2, \ldots, j-1\}$.
  - But this means the factorization depends on variable order.
- We could instead select parents of j from $\{1, 2, \ldots, j-1, j+1, j+2, \ldots, d\}$,
  - Gives valid factorization as long as graph is acyclic.
  - Acyclicity implies that a "topological order" order exists.
    - We could reorder so that parents come before children.
- Note that some graphs imply same conditional independences:
  - Equivalent graphs: same v-structures and other (undirected) edges are the same.
  - Examples of *non-equivalent* graphs:



Parameter Learning in General DAG Models

- The log-likelihood in DAG models has a nice form,

$$\begin{aligned}\log p(x) &= \log \prod_{j=1}^{d} p(x_j | x_{\mathsf{pa}(j)}) \\ &= \sum_{j=1}^{d} \log p(x_j | x_{\mathsf{pa}(j)})\end{aligned}$$

- If each $x_j | x_{\mathsf{pa}(j)}$ has its own parameters, we can fit them independently.

Parameter Learning in General DAG Models

- The log-likelihood in DAG models has a nice form,

$$\log p(x) = \log \prod_{j=1}^{d} p(x_j | x_{\mathsf{pa}(j)})$$
$$= \sum_{j=1}^{d} \log p(x_j | x_{\mathsf{pa}(j)})$$

- If each $x_j | x_{\mathsf{pa}(j)}$ has its own parameters, we can fit them independently.
  - E.g., for discrete x_j use logistic regression with x_j as target and $x_{\mathsf{pa}(j)}$ as features.
  - We've done this before: naive Bayes, Gaussian discriminant analysis, etc.

Parameter Learning in General DAG Models

- The log-likelihood in DAG models has a nice form,

$$\log p(x) = \log \prod_{j=1}^{d} p(x_j | x_{\mathsf{pa}(j)})$$
$$= \sum_{j=1}^{d} \log p(x_j | x_{\mathsf{pa}(j)})$$

- If each $x_j | x_{\mathsf{pa}(j)}$ has its own parameters, we can fit them independently.
  - E.g., for discrete x_j use logistic regression with x_j as target and $x_{\mathsf{pa}(j)}$ as features.
  - We've done this before: naive Bayes, Gaussian discriminant analysis, etc.
- Sometimes you want to have tied parameters:
  - E.g., Gaussian discriminant analysis with shared covariance.
  - E.g., homogenous Markov chains.

Parameter Learning in General DAG Models

- The log-likelihood in DAG models has a nice form,

$$\log p(x) = \log \prod_{j=1}^{d} p(x_j | x_{\mathsf{pa}(j)})$$
$$= \sum_{j=1}^{d} \log p(x_j | x_{\mathsf{pa}(j)})$$

- If each $x_j | x_{\mathsf{pa}(j)}$ has its own parameters, we can fit them independently.
  - E.g., for discrete x_j use logistic regression with x_j as target and $x_{\mathsf{pa}(j)}$ as features.
  - We've done this before: naive Bayes, Gaussian discriminant analysis, etc.
- Sometimes you want to have tied parameters:
  - E.g., Gaussian discriminant analysis with shared covariance.
  - E.g., homogenous Markov chains.
  - Still easy, but you need to fit $x_j | x_{\mathsf{pa}(j)}$ and $x_k | x_{\mathsf{pa}(k)}$ together if share parameters.

Parameterization in DAG Models

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

Parameterization in DAG Models

- To specify distribution, we need to decide on form of $p(x_j|x_{\mathsf{pa}(j)})$.
- Common "parsimonious" choices:
  - Gaussian: solve d least squares problems ("Gaussian belief network").
  - Logistic: solve d logistic regression problems ("sigmoid belief networks").

Parameterization in DAG Models

- To specify distribution, we need to decide on form of $p(x_j|x_{\mathsf{pa}(j)})$.
- Common "parsimonious" choices:
  - Gaussian: solve d least squares problems ("Gaussian belief network").
  - Logistic: solve d logistic regression problems ("sigmoid belief networks").
  - Other linear models: student t, softmax, Poisson, etc.
  - Noisy-or: $p(x_j x_{\mathsf{pa}(j)}) = 1 - \prod_{k \in \mathsf{pa}(j)} q_j$.

Parameterization in DAG Models

- To specify distribution, we need to decide on form of $p(x_j|x_{\text{pa}(j)})$.
- Common "parsimonious" choices:
  - Gaussian: solve d least squares problems ("Gaussian belief network").
  - Logistic: solve d logistic regression problems ("sigmoid belief networks").
  - Other linear models: student t, softmax, Poisson, etc.
  - Noisy-or: $p(x_j x_{\text{pa}(j)}) = 1 - \prod_{k \in \text{pa}(j)} q_j$.
- Less-Parsimonious choices:
  - Change of basis on parent variables.
  - Kernel trick or kernel density estimation.
  - Mixture models.

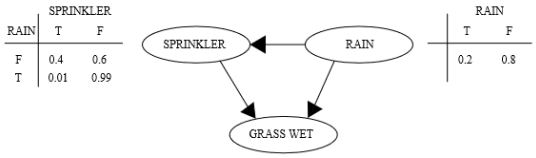
Parameterization in DAG Models

- To specify distribution, we need to decide on form of $p(x_j|x_{\text{pa}(j)})$.
- Common "parsimonious" choices:
  - Gaussian: solve d least squares problems ("Gaussian belief network").
  - Logistic: solve d logistic regression problems ("sigmoid belief networks").
  - Other linear models: student t, softmax, Poisson, etc.
  - Noisy-or: $p(x_j x_{\text{pa}(j)}) = 1 - \prod_{k \in \text{pa}(j)} q_j$.
- Less-Parsimonious choices:
  - Change of basis on parent variables.
  - Kernel trick or kernel density estimation.
  - Mixture models.
- If number of parents and states of x_j is small, can use tabular parameterization:

$$p(x_j|x_{\text{pa}(j)}) = \theta_{x_j, x_{\text{pa}(j)}}.$$

  - Intuitive: just specify $p(\text{wet grass}|\text{sprinkler}, \text{rain})$.
  - With binary states and k parents, need 2^{k+1} parameters.

Tabular Parameterization Examples



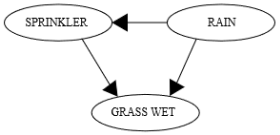
https://en.wikipedia.org/wiki/Bayesian_network

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

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

Tabular Parameterization Examples

RAIN	SPRINKLER T	SPRINKLER F
F	0.4	0.6
T	0.01	0.99



RAIN T	RAIN F
0.2	0.8

SPRINKLER	RAIN	GRASS WET T	GRASS WET F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

https://en.wikipedia.org/wiki/Bayesian_network

$$p(G=1|R=1) = p(G=1,S=0|R=1) + p(G=1,S=1|R=1) \quad \left(p(a) = \sum_b p(a,b)\right)$$
$$= p(G=1|S=0,R=1)p(S=0|R=1) + p(G=1|S=1,R=1)p(S=1|R=1)$$
$$= 0.8(0.99) + 0.99(0.01)$$

Outline

1 Learning DAG Parameters

2 Structured Prediction

3 Undirected Graphical Models

4 Decoding, Inference, and Sampling

Classic Machine Learning vs. Structured Prediction

- Classical supervised learning: Output is a single label.

Input: 

Output: "P"

Classic Machine Learning vs. Structured Prediction

- Classical supervised learning: Output is a single label.

Input: 

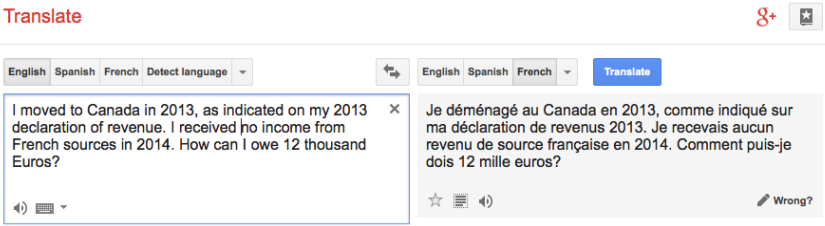
Output: "P"

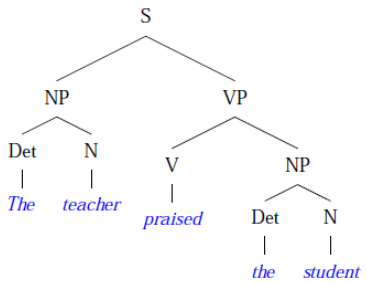
- Structured prediction: Output can be a general object.

Input: 

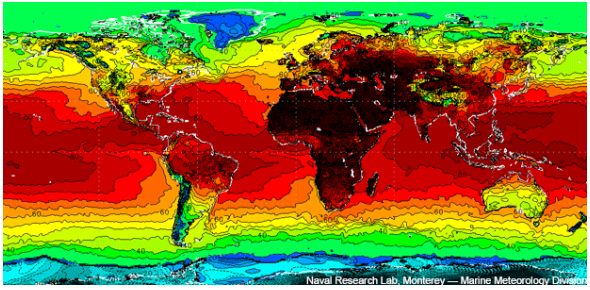
Output: "Paris"

Examples of Structured Prediction

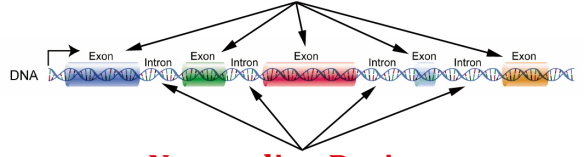




Examples of Structured Prediction



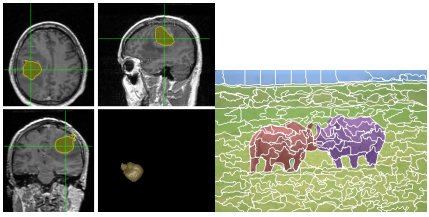
Coding Regions



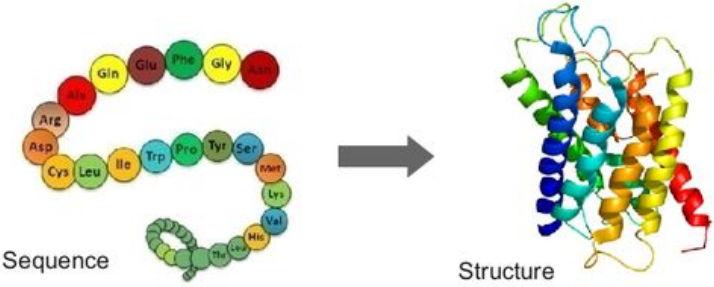
Non-coding Regions

(Containing large TE content)

Examples of Structured Prediction



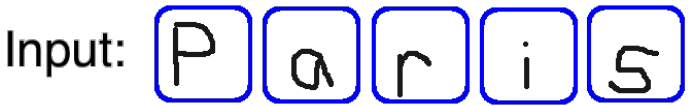
Examples of Structured Prediction



In 1917, Einstein applied the general theory of relativity to model the large-scale structure of the universe. He was visiting the United States when Adolf Hitler came to power in 1933 and did not go back to Germany, where he had been a professor at the Berlin Academy of Sciences. He settled in the U.S., becoming an American citizen in 1940. On the eve of World War II, he endorsed a letter to President Franklin D. Roosevelt alerting him to the potential development of "extremely powerful bombs of a new type" and recommending that the U.S. begin similar research. This eventually led to what would become the Manhattan Project. Einstein supported defending the Allied forces, but largely denounced using the new discovery of nuclear fission as a weapon. Later, with the British philosopher Bertrand Russell, Einstein signed the Russell–Einstein Manifesto, which highlighted the danger of nuclear weapons. Einstein was affiliated with the Institute for Advanced Study in Princeton, New Jersey, until his death in 1955.

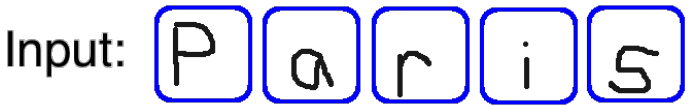
Tag colours:
LOCATION TIME PERSON ORGANIZATION MONEY PERCENT DATE

"Classic" ML for Structured Prediction



Two ways to formulate as "classic" machine learning:

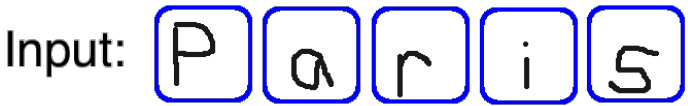
"Classic" ML for Structured Prediction



Two ways to formulate as "classic" machine learning:

1. Treat each word as a different class label.
   - Problem: there are too many possible words.

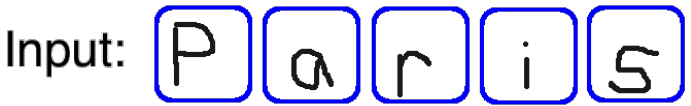
"Classic" ML for Structured Prediction



Two ways to formulate as "classic" machine learning:

1. Treat each word as a different class label.
   - Problem: there are too many possible words.
2. Predict each letter individually:
   - Works if you are really good at predicting individual letters.

"Classic" ML for Structured Prediction



Two ways to formulate as "classic" machine learning:

1. Treat each word as a different class label.
   - Problem: there are too many possible words.
2. Predict each letter individually:
   - Works if you are really good at predicting individual letters.
   - But Some tasks don't have a natural decomposition.
   - Ignores dependencies between letters.

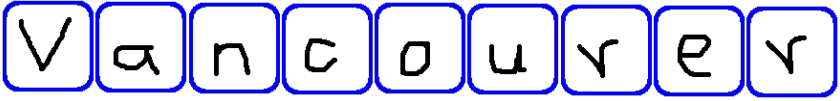
Motivation: Structured Prediction

- What letter is this?

Motivation: Structured Prediction

- What letter is this?

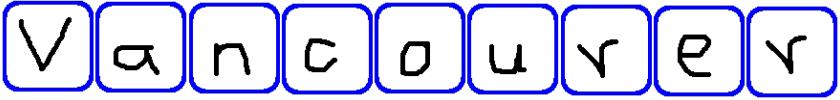
- What are these letters?



Motivation: Structured Prediction

- What letter is this?

- What are these letters?

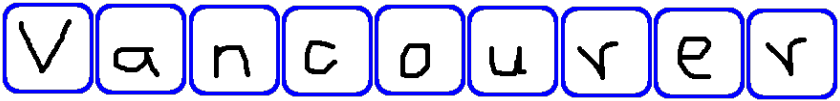


- Predict each letter using "classic" ML and neighbouring images?
  - Turn this into a standard supervised learning problem?

Motivation: Structured Prediction

- What letter is this?

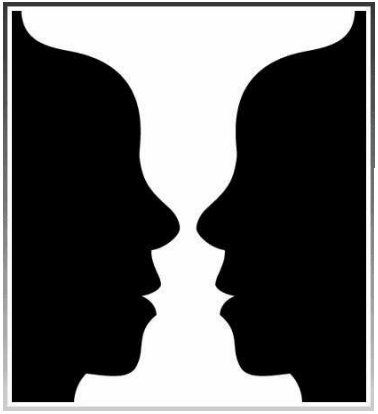
- What are these letters?

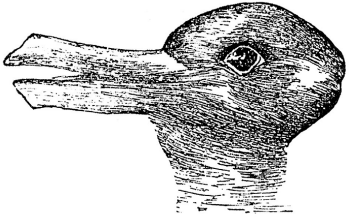


- Predict each letter using "classic" ML and neighbouring images?
  - Turn this into a standard supervised learning problem?
- Good or bad depending on goal:
  - Good if you want to predict individual letters.
  - Bad if goal is to predict entire word.

Does the brain do structured prediction?

Gestalt effect: "whole is other than the sum of the parts".





What do you see?
By shifting perspective you might see an old woman or a young woman.

Structured Prediction as Conditional Density Estimation

- Most structured prediction models formulate as conditional density estimation:
  - Model $p(Y|X)$ for possible output objects Y.

Structured Prediction as Conditional Density Estimation

- Most structured prediction models formulate as conditional density estimation:
  - Model $p(Y|X)$ for possible output objects Y.
- Three paradigms:
  1. Generative models consider joint probability of X and Y

$$p(Y|X) \propto p(Y, X),$$

and generalize things like naive Bayes and Gaussian discriminant analysis.

Structured Prediction as Conditional Density Estimation

- Most structured prediction models formulate as conditional density estimation:
  - Model $p(Y|X)$ for possible output objects Y.
- Three paradigms:
  1. Generative models consider joint probability of X and Y

$$p(Y|X) \propto p(Y, X),$$

     and generalize things like naive Bayes and Gaussian discriminant analysis.
  2. Conditional models directly model conditional $P(Y|X)$,
     and generalize things like least squares and logistic regression.

Structured Prediction as Conditional Density Estimation

- Most structured prediction models formulate as conditional density estimation:
  - Model $p(Y|X)$ for possible output objects Y.
- Three paradigms:
  1. Generative models consider joint probability of X and Y

     $$p(Y|X) \propto p(Y, X),$$

     and generalize things like naive Bayes and Gaussian discriminant analysis.
  2. Conditional models directly model conditional $P(Y|X)$,
     and generalize things like least squares and logistic regression.
  3. Structured SVMs try to make $P(Y^i|X^i)$ larger than $P(Y|X^i)$ for all $Y \neq Y$,
     and generallze SVMs.
- Typically define graphical model on "parts" of Y (and X in generative case).
- But usually no natural order, so often use undirected graphical models.

Outline

1 Learning DAG Parameters

2 Structured Prediction

3 Undirected Graphical Models

4 Decoding, Inference, and Sampling

Undirected Graphical Models

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

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

  from among a set of subsets of $\mathcal{C}$.
- The ϕ_c are called potential functions: can be any non-negative function.

Undirected Graphical Models

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

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

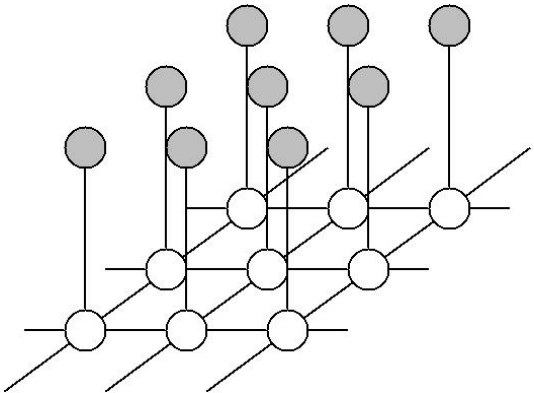
  from among a set of subsets of $\mathcal{C}$.
- The ϕ_c are called potential functions: can be any non-negative function.
- Ordering doesn't matter: more natural for things like pixels of an image.
- Important special case is pairwise undirected graphical model:

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

  where E are a set of undirected edges.
- Theoretically, only need ϕ_c for maximal subsets in C.

Undirected Graphical Models

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



http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0809/ORCHARD

- **Observed** nodes are the pixels, and we have lattice-structured label dependency.
  - Takes into account that neighbouring pixels are likely to have same labels.

From Probability Factorization to Graphs

- For a pairwise UGM,

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

we visualize independence assumptions as an undirected graph:

- We have edge i to j if $(i, j) \in E$.

From Probability Factorization to Graphs

- For a pairwise UGM,

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

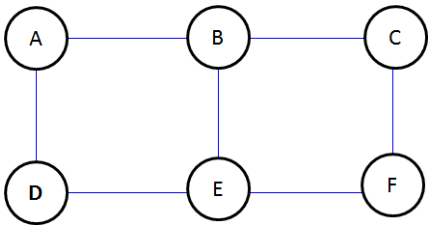
  we visualize independence assumptions as an undirected graph:
  - We have edge i to j if $(i, j) \in E$.
- For general UGMs,

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

  we have the edge (i, j) if i and j are together in at least one c.
  - Maximal subsets correspond to maximal cliques in the graphs.

Conditional Independence in Undirected Graphical Models

- It's easy to check conditional independence in UGMs:
  - $A \perp B\ C$ if C blocks all paths from any A to any B.
- Example:



  - $A \not\perp C$.
  - $A \not\perp C|B$.
  - $A \perp C|B, E$.
  - $A, B \not\perp F|C$
  - $A, B \perp F|C, E$.

Dealing with the Huge Number of Labels

- Why are UGMs useful for structured prediction?
  - Number of possible objects Y is huge.

Dealing with the Huge Number of Labels

- Why are UGMs useful for structured prediction?
  - Number of possible objects Y is huge.
  - We want to share information across Y.
  - We typically do this by writing Y as a set of "parts".

Dealing with the Huge Number of Labels

- Why are UGMs useful for structured prediction?
  - Number of possible objects Y is huge.
  - We want to share information across Y.
  - We typically do this by writing Y as a set of "parts".
- Specifically, write $p(Y|X)$ as product of potentials:
  - $\phi(Y_j, X_j)$: potential of individual letter given image.

Dealing with the Huge Number of Labels

- Why are UGMs useful for structured prediction?
  - Number of possible objects Y is huge.
  - We want to share information across Y.
  - We typically do this by writing Y as a set of "parts".
- Specifically, write $p(Y|X)$ as product of potentials:
  - $\phi(Y_j, X_j)$: potential of individual letter given image.
  - $\phi(Y_{j-1}, Y_j)$: dependency between adjacent letters ('q-u').

Dealing with the Huge Number of Labels

- Why are UGMs useful for structured prediction?
  - Number of possible objects Y is huge.
  - We want to share information across Y.
  - We typically do this by writing Y as a set of "parts".
- Specifically, write $p(Y|X)$ as product of potentials:
  - $\phi(Y_j, X_j)$: potential of individual letter given image.
  - $\phi(Y_{j-1}, Y_j)$: dependency between adjacent letters ('q-u').
  - $\phi(Y_{j-1}, Y_j, X_{j-1}, X_j)$: adjacent letters and image dependency.

Dealing with the Huge Number of Labels

- Why are UGMs useful for structured prediction?
  - Number of possible objects Y is huge.
  - We want to share information across Y.
  - We typically do this by writing Y as a set of "parts".
- Specifically, write $p(Y|X)$ as product of potentials:
  - $\phi(Y_j, X_j)$: potential of individual letter given image.
  - $\phi(Y_{j-1}, Y_j)$: dependency between adjacent letters ('q-u').
  - $\phi(Y_{j-1}, Y_j, X_{j-1}, X_j)$: adjacent letters and image dependency.
  - $\phi_j(Y_{j-1}, Y_j)$: position-based dependency (French: 'e-r' ending).

Dealing with the Huge Number of Labels

- Why are UGMs useful for structured prediction?
  - Number of possible objects Y is huge.
  - We want to share information across Y.
  - We typically do this by writing Y as a set of "parts".
- Specifically, write $p(Y|X)$ as product of potentials:
  - $\phi(Y_j, X_j)$: potential of individual letter given image.
  - $\phi(Y_{j-1}, Y_j)$: dependency between adjacent letters ('q-u').
  - $\phi(Y_{j-1}, Y_j, X_{j-1}, X_j)$: adjacent letters and image dependency.
  - $\phi_j(Y_{j-1}, Y_j)$: position-based dependency (French: 'e-r' ending).
  - $\phi_j(Y_{j-2}, Y_{j-1}, Y_j)$: third-order and position (English: 'i-n-g' end).

Dealing with the Huge Number of Labels

- Why are UGMs useful for structured prediction?
  - Number of possible objects Y is huge.
  - We want to share information across Y.
  - We typically do this by writing Y as a set of "parts".
- Specifically, write $p(Y|X)$ as product of potentials:
  - $\phi(Y_j, X_j)$: potential of individual letter given image.
  - $\phi(Y_{j-1}, Y_j)$: dependency between adjacent letters ('q-u').
  - $\phi(Y_{j-1}, Y_j, X_{j-1}, X_j)$: adjacent letters and image dependency.
  - $\phi_j(Y_{j-1}, Y_j)$: position-based dependency (French: 'e-r' ending).
  - $\phi_j(Y_{j-2}, Y_{j-1}, Y_j)$: third-order and position (English: 'i-n-g' end).
  - $\phi(Y \in \mathcal{D})$: is y in dictionary $\mathcal{D}$?

Dealing with the Huge Number of Labels

- Why are UGMs useful for structured prediction?
  - Number of possible objects Y is huge.
  - We want to share information across Y.
  - We typically do this by writing Y as a set of "parts".
- Specifically, write $p(Y|X)$ as product of potentials:
  - $\phi(Y_j, X_j)$: potential of individual letter given image.
  - $\phi(Y_{j-1}, Y_j)$: dependency between adjacent letters ('q-u').
  - $\phi(Y_{j-1}, Y_j, X_{j-1}, X_j)$: adjacent letters and image dependency.
  - $\phi_j(Y_{j-1}, Y_j)$: position-based dependency (French: 'e-r' ending).
  - $\phi_j(Y_{j-2}, Y_{j-1}, Y_j)$: third-order and position (English: 'i-n-g' end).
  - $\phi(Y \in \mathcal{D})$: is y in dictionary $\mathcal{D}$?
- Learn the parameters of $p(Y|X)$ from data:
  - Learn parameters so that "correct" labels gets high probability
  - Potentials let us transfer knowledge to completely new objects Y.

(E.g., predict a word you've never seen before.)

Digression: Gaussian Graphical Models

- Multivariate Gaussian can be written as

$$p(x) \propto \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

$$\propto \exp\left(-\frac{1}{2}x^T\Sigma^{-1}x + x^T\underbrace{\Sigma^{-1}\mu}_{v}\right),$$

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:

$$p(x) \propto \exp(\left(-\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}x_i x_j \Sigma_{ij}^{-1} + \sum_{i=1}^{d} x_i v_i\right)$$

$$= \left(\prod_{i=1}^{d}\prod_{j=1}\underbrace{\exp\left(-\frac{1}{2}x_i x_j \Sigma_{ij}^{-1}\right)}_{\phi_{ij}(x_i,x_j)}\right)\left(\prod_{i=1}^{d}\underbrace{\exp\left(x_i v_i\right)}_{\phi_i(x_i)}\right)$$

- We also call multivariate Gaussian Gaussian graphical models (GGMS)

Independence in GGMs and Graphical Lasso

- We just showed that GGMs are pairwise UGMs with $\phi_{ij}(x_i, x_j) = x_i x_j \Theta_{ij}$,
  - Where Θ_{ij} is element (i, j) of Σ^{-1}.
  - Setting $\Theta_{ij} = 0$ is equivalent to removing direct dependency between i and j:

$$\Theta_{ij} = 0 \Rightarrow x_i \perp x_{-i} | x_j.$$

- GGMs conditional independencies corresponds to inverse covariance sparsity.

Independence in GGMs and Graphical Lasso

- We just showed that GGMs are pairwise UGMs with $\phi_{ij}(x_i, x_j) = x_i x_j \Theta_{ij}$,
  - Where Θ_{ij} is element (i, j) of Σ^{-1}.
  - Setting $\Theta_{ij} = 0$ is equivalent to removing direct dependency between i and j:

$$\Theta_{ij} = 0 \Rightarrow x_i \perp x_{-i} | x_j.$$

- GGMs conditional independencies corresponds to inverse covariance sparsity.
  - Diagonal Θ gives disconnected graph: all variables are indpendent.
  - Full Θ gives fully-connected graph: all variables in completely dependent.

Independence in GGMs and Graphical Lasso

- We just showed that GGMs are pairwise UGMs with $\phi_{ij}(x_i, x_j) = x_i x_j \Theta_{ij}$,
  - Where Θ_{ij} is element (i, j) of Σ^{-1}.
  - Setting $\Theta_{ij} = 0$ is equivalent to removing direct dependency between i and j:

$$\Theta_{ij} = 0 \Rightarrow x_i \perp x_{-i} | x_j.$$

- GGMs conditional independencies corresponds to inverse covariance sparsity.
  - Diagonal Θ gives disconnected graph: all variables are indpendent.
  - Full Θ gives fully-connected graph: all variables in completely dependent.
  - Tri-diagonal Θ gives chain-structured graph:
    - All variables are dependent, but conditionally independent given neighbours.

Independence in GGMs and Graphical Lasso

- Consider Gaussian with tri-diagonal precision Θ:

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

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

- $\Sigma_{ij} \neq 0$ so all variables are dependent: $x_1 \not\perp x_2$, $x_1 \not\perp x_5$, and so on.
- But conditional independence is described by a chain-structure:

$$x_1 \perp x_3, x_4, x_5 | x_2,$$

so $p(x_1|x_2, x_3, x_4, x_5) = p(x_1|x_2)$.

Independence in GGMs and Graphical Lasso

- GGMs conditional independencies corresponds to inverse covariance sparsity.

Independence in GGMs and Graphical Lasso

- GGMs conditional independencies corresponds to inverse covariance sparsity.
- Recall fitting multivariate Gaussian with L1-regularization,

$$\underset{\Theta \succ 0}{\text{argmin}}\, \text{Tr}(S\Theta) - \log|\Theta| + \lambda\|\Theta\|_1,$$

called the graphical Lasso because it encourages a sparse graph.

Independence in GGMs and Graphical Lasso

- GGMs conditional independencies corresponds to inverse covariance sparsity.
- Recall fitting multivariate Gaussian with L1-regularization,

$$\underset{\Theta \succ 0}{\text{argmin}} \, \text{Tr}(S\Theta) - \log |\Theta| + \lambda \|\Theta\|_1,$$

  called the graphical Lasso because it encourages a sparse graph.
- Consider instead fitting DAG model with Gaussian probabilities:
  - DAG structure corresponds to Cholesky of covariance of multivariate Gaussian.

Outline

1. Learning DAG Parameters
2. Structured Prediction
3. Undirected Graphical Models
4. Decoding, Inference, and Sampling

Tractability of UGMs

- In UGMs we assume that

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

where Z is the constant such that

$$\sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} p(x) = 1 \text{ (discrete)}, \quad \int_{x_1} \int_{x_2} \cdots \int_{x_d} p(x) dx_d dx_{d-1} \ldots dx_1 = 1 \text{ (cont)}.$$

- So Z is

$$Z = \sum_{x} \prod_{c \in \mathcal{C}} \phi_c(x_c) \text{ (discrete)}, \quad \int_{x} \prod_{c \in \mathcal{C}} \phi_c(x_c) dx \text{ (cont)}$$

Tractability of UGMs

- In UGMs we assume that

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

where Z is the constant such that

$$\sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} p(x) = 1 \text{ (discrete)}, \quad \int_{x_1} \int_{x_2} \cdots \int_{x_d} p(x) dx_d dx_{d-1} \ldots dx_1 = 1 \text{ (cont)}.$$

- So Z is

$$Z = \sum_{x} \prod_{c \in \mathcal{C}} \phi_c(x_c) \text{ (discrete)}, \quad \int_{x} \prod_{c \in \mathcal{C}} \phi_c(x_c) dx \text{ (cont)}$$

- Whether you can compute Z depends on the choice of ϕ_c:
  - Gaussian case: $O(d^3)$ in general, but $O(d)$ for "nice" graphs.

Tractability of UGMs

- In UGMs we assume that

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

where Z is the constant such that

$$\sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} p(x) = 1 \text{ (discrete)}, \quad \int_{x_1} \int_{x_2} \cdots \int_{x_d} p(x) dx_d dx_{d-1} \ldots dx_1 = 1 \text{ (cont)}.$$

- So Z is

$$Z = \sum_{x} \prod_{c \in \mathcal{C}} \phi_c(x_c) \text{ (discrete)}, \quad \int_{x} \prod_{c \in \mathcal{C}} \phi_c(x_c) dx \text{ (cont)}$$

- Whether you can compute Z depends on the choice of ϕ_c:
  - Gaussian case: $O(d^3)$ in general, but $O(d)$ for "nice" graphs.
  - Continuous non-Gaussian: usually requires numerical integration.

Tractability of UGMs

- In UGMs we assume that

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

where Z is the constant such that

$$\sum_{x_1} \sum_{x_2} \cdots \sum_{x_d} p(x) = 1 \text{ (discrete)}, \quad \int_{x_1} \int_{x_2} \cdots \int_{x_d} p(x) dx_d dx_{d-1} \ldots dx_1 = 1 \text{ (cont)}.$$

- So Z is

$$Z = \sum_{x} \prod_{c \in \mathcal{C}} \phi_c(x_c) \text{ (discrete)}, \quad \int_{x} \prod_{c \in \mathcal{C}} \phi_c(x_c) dx \text{ (cont)}$$

- Whether you can compute Z depends on the choice of ϕ_c:
  - Gaussian case: $O(d^3)$ in general, but $O(d)$ for "nice" graphs.
  - Continuous non-Gaussian: usually requires numerical integration.
  - Discrete case: NP-hard in general, but $O(dk^2)$ for "nice" graphs.

Decoding, Inference, and Sampling

- We're going to study 3 operations given discrete UGM:

Decoding, Inference, and Sampling

- We're going to study 3 operations given discrete UGM:
  1. Decoding: Compute the optimal configuration,

$$\underset{x}{\text{argmax}}\{p(x_1, x_2, \ldots, x_n)\}.$$

Decoding, Inference, and Sampling

- We're going to study 3 operations given discrete UGM:
  1. Decoding: Compute the optimal configuration,

$$\underset{x}{\text{argmax}}\{p(x_1, x_2, \ldots, x_n)\}.$$

  2. Inference: Compute partition function and marginals,

$$Z = \sum_x p(x), \quad p(x_j = s) = \sum_{x|x_j=s} p(x).$$

Decoding, Inference, and Sampling

- We're going to study 3 operations given discrete UGM:
  1. Decoding: Compute the optimal configuration,

$$\underset{x}{\text{argmax}}\{p(x_1, x_2, \ldots, x_n)\}.$$

  2. Inference: Compute partition function and marginals,

$$Z = \sum_x p(x), \quad p(x_j = s) = \sum_{x|x_j=s} p(x).$$

  3. Sampling: Generate x according from the distribution:

$$x \sim p(x).$$

- In general discrete UGMs, all of these are NP-hard.
  - We'll study cases that can be efficiently solved/approximated.

Decoding, Inference, and Sampling

- Let's illustrate the three tasks with a Bernoulli example,

$$p(x = 1) = 0.75, \quad p(x = 0) = 0.25.$$

- In this setting all 3 operations are trivial:
  1. Decoding:

$$\underset{x}{\text{argmax}}\{p(x)\} = 1.$$

Decoding, Inference, and Sampling

- Let's illustrate the three tasks with a Bernoulli example,

$$p(x = 1) = 0.75, \quad p(x = 0) = 0.25.$$

- In this setting all 3 operations are trivial:
  1. Decoding:
  $$\underset{x}{\text{argmax}}\{p(x)\} = 1.$$
  2. Inference:
  $$Z = p(x = 1) + p(x = 0) = 1, \quad p(x = 1) = 0.75.$$

Decoding, Inference, and Sampling

- Let's illustrate the three tasks with a Bernoulli example,

$$p(x = 1) = 0.75, \quad p(x = 0) = 0.25.$$

- In this setting all 3 operations are trivial:
  1. Decoding:
  $$\underset{x}{\text{argmax}}\{p(x)\} = 1.$$
  2. Inference:
  $$Z = p(x = 1) + p(x = 0) = 1, \quad p(x = 1) = 0.75.$$
  3. Sampling: Let $u \sim \mathcal{U}(0, 1)$, then
  $$x = \begin{cases} 1 & u \leq 0.75 \\ 0 & \text{otherwise} \end{cases}.$$

3 Tasks by Hand on a Simple Example

- To illustrate the tasks, let's take a simple 2-variable example,

$$p(x_1, x_2) \propto \tilde{p}(x_1, x_2) = \phi_1(x_1)\phi_2(x_2)\phi_{12}(x_1, x_2).$$

where $\tilde{p}$ is the unnormalized probability (ignoring Z) and

$$\phi_1(x_1) = \begin{cases} 1 & x_1 = 0 \\ 2 & x_1 = 1 \end{cases}, \quad \phi_2(x_2) = \begin{cases} 1 & x_1 = 0 \\ 3 & x_2 = 1 \end{cases}, \quad \phi_{1,2}(x_1, x_2) = \begin{cases} 2 & x_1 = x_2 \\ 1 & x_1 \neq x_2 \end{cases}.$$

3 Tasks by Hand on a Simple Example

- To illustrate the tasks, let's take a simple 2-variable example,

$$p(x_1, x_2) \propto \tilde{p}(x_1, x_2) = \phi_1(x_1)\phi_2(x_2)\phi_{12}(x_1, x_2).$$

  where $\tilde{p}$ is the unnormalized probability (ignoring Z) and

$$\phi_1(x_1) = \begin{cases} 1 & x_1 = 0 \\ 2 & x_1 = 1 \end{cases}, \quad \phi_2(x_2) = \begin{cases} 1 & x_1 = 0 \\ 3 & x_2 = 1 \end{cases}, \quad \phi_{1,2}(x_1, x_2) = \begin{cases} 2 & x_1 = x_2 \\ 1 & x_1 \neq x_2 \end{cases}.$$

- x_1 wants to be 1, x_2 really wants to be 1, both want to be same.

3 Tasks by Hand on a Simple Example

- To illustrate the tasks, let's take a simple 2-variable example,

$$p(x_1, x_2) \propto \tilde{p}(x_1, x_2) = \phi_1(x_1)\phi_2(x_2)\phi_{12}(x_1, x_2).$$

  where $\tilde{p}$ is the unnormalized probability (ignoring Z) and

$$\phi_1(x_1) = \begin{cases} 1 & x_1 = 0 \\ 2 & x_1 = 1 \end{cases}, \quad \phi_2(x_2) = \begin{cases} 1 & x_1 = 0 \\ 3 & x_2 = 1 \end{cases}, \quad \phi_{1,2}(x_1, x_2) = \begin{cases} 2 & x_1 = x_2 \\ 1 & x_1 \neq x_2 \end{cases}.$$

- x_1 wants to be 1, x_2 really wants to be 1, both want to be same.
- We can think of the possible states/potentials in a big table:

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$

3 Tasks by Hand on a Simple Example

- To illustrate the tasks, let's take a simple 2-variable example,

$$p(x_1, x_2) \propto \tilde{p}(x_1, x_2) = \phi_1(x_1)\phi_2(x_2)\phi_{12}(x_1, x_2).$$

  where $\tilde{p}$ is the unnormalized probability (ignoring Z) and

$$\phi_1(x_1) = \begin{cases} 1 & x_1 = 0 \\ 2 & x_1 = 1 \end{cases}, \quad \phi_2(x_2) = \begin{cases} 1 & x_1 = 0 \\ 3 & x_2 = 1 \end{cases}, \quad \phi_{1,2}(x_1, x_2) = \begin{cases} 2 & x_1 = x_2 \\ 1 & x_1 \neq x_2 \end{cases}.$$

- x_1 wants to be 1, x_2 really wants to be 1, both want to be same.
- We can think of the possible states/potentials in a big table:

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$
0	0				

3 Tasks by Hand on a Simple Example

- To illustrate the tasks, let's take a simple 2-variable example,

$$p(x_1, x_2) \propto \tilde{p}(x_1, x_2) = \phi_1(x_1)\phi_2(x_2)\phi_{12}(x_1, x_2).$$

  where $\tilde{p}$ is the unnormalized probability (ignoring Z) and

$$\phi_1(x_1) = \begin{cases} 1 & x_1 = 0 \\ 2 & x_1 = 1 \end{cases}, \quad \phi_2(x_2) = \begin{cases} 1 & x_1 = 0 \\ 3 & x_2 = 1 \end{cases}, \quad \phi_{1,2}(x_1, x_2) = \begin{cases} 2 & x_1 = x_2 \\ 1 & x_1 \neq x_2 \end{cases}.$$

- x_1 wants to be 1, x_2 really wants to be 1, both want to be same.
- We can think of the possible states/potentials in a big table:

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$
0	0	1	1	2	

3 Tasks by Hand on a Simple Example

- To illustrate the tasks, let's take a simple 2-variable example,

$$p(x_1, x_2) \propto \tilde{p}(x_1, x_2) = \phi_1(x_1)\phi_2(x_2)\phi_{12}(x_1, x_2).$$

  where $\tilde{p}$ is the unnormalized probability (ignoring Z) and

$$\phi_1(x_1) = \begin{cases} 1 & x_1 = 0 \\ 2 & x_1 = 1 \end{cases}, \quad \phi_2(x_2) = \begin{cases} 1 & x_1 = 0 \\ 3 & x_2 = 1 \end{cases}, \quad \phi_{1,2}(x_1, x_2) = \begin{cases} 2 & x_1 = x_2 \\ 1 & x_1 \neq x_2 \end{cases}.$$

- x_1 wants to be 1, x_2 really wants to be 1, both want to be same.
- We can think of the possible states/potentials in a big table:

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$
0	0	1	1	2	2

3 Tasks by Hand on a Simple Example

- To illustrate the tasks, let's take a simple 2-variable example,

$$p(x_1, x_2) \propto \tilde{p}(x_1, x_2) = \phi_1(x_1)\phi_2(x_2)\phi_{12}(x_1, x_2).$$

  where $\tilde{p}$ is the unnormalized probability (ignoring Z) and

$$\phi_1(x_1) = \begin{cases} 1 & x_1 = 0 \\ 2 & x_1 = 1 \end{cases}, \quad \phi_2(x_2) = \begin{cases} 1 & x_1 = 0 \\ 3 & x_2 = 1 \end{cases}, \quad \phi_{1,2}(x_1, x_2) = \begin{cases} 2 & x_1 = x_2 \\ 1 & x_1 \neq x_2 \end{cases}.$$

- x_1 wants to be 1, x_2 really wants to be 1, both want to be same.
- We can think of the possible states/potentials in a big table:

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$
0	0	1	1	2	2
0	1				

3 Tasks by Hand on a Simple Example

- To illustrate the tasks, let's take a simple 2-variable example,

$$p(x_1, x_2) \propto \tilde{p}(x_1, x_2) = \phi_1(x_1)\phi_2(x_2)\phi_{12}(x_1, x_2).$$

  where $\tilde{p}$ is the unnormalized probability (ignoring Z) and

$$\phi_1(x_1) = \begin{cases} 1 & x_1 = 0 \\ 2 & x_1 = 1 \end{cases}, \quad \phi_2(x_2) = \begin{cases} 1 & x_1 = 0 \\ 3 & x_2 = 1 \end{cases}, \quad \phi_{1,2}(x_1, x_2) = \begin{cases} 2 & x_1 = x_2 \\ 1 & x_1 \neq x_2 \end{cases}.$$

- x_1 wants to be 1, x_2 really wants to be 1, both want to be same.
- We can think of the possible states/potentials in a big table:

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$
0	0	1	1	2	2
0	1	1	3	1	3

3 Tasks by Hand on a Simple Example

- To illustrate the tasks, let's take a simple 2-variable example,

$$p(x_1, x_2) \propto \tilde{p}(x_1, x_2) = \phi_1(x_1)\phi_2(x_2)\phi_{12}(x_1, x_2).$$

where $\tilde{p}$ is the unnormalized probability (ignoring Z) and

$$\phi_1(x_1) = \begin{cases} 1 & x_1 = 0 \\ 2 & x_1 = 1 \end{cases}, \quad \phi_2(x_2) = \begin{cases} 1 & x_1 = 0 \\ 3 & x_2 = 1 \end{cases}, \quad \phi_{1,2}(x_1, x_2) = \begin{cases} 2 & x_1 = x_2 \\ 1 & x_1 \neq x_2 \end{cases}.$$

- x_1 wants to be 1, x_2 really wants to be 1, both want to be same.
- We can think of the possible states/potentials in a big table:

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$
0	0	1	1	2	2
0	1	1	3	1	3
1	0	2	1	1	2
1	1	2	3	2	12

Decoding on Simple Example

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$
0	0	1	1	2	2
0	1	1	3	1	3
1	0	2	1	1	2
1	1	2	3	2	12

- Decoding is finding the maximizer of $p(x_1, x_2)$:

Decoding on Simple Example

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$
0	0	1	1	2	2
0	1	1	3	1	3
1	0	2	1	1	2
1	1	2	3	2	12

- Decoding is finding the maximizer of $p(x_1, x_2)$:
  - In this case it is $x_1 = 1$ and $x_2 = 1$.

Inference on Simple Example

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$
0	0	1	1	2	2
0	1	1	3	1	3
1	0	2	1	1	2
1	1	2	3	2	12

- One inference task is finding Z:

Inference on Simple Example

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$
0	0	1	1	2	2
0	1	1	3	1	3
1	0	2	1	1	2
1	1	2	3	2	12

- One inference task is finding Z:
  - In this case $Z = 2 + 3 + 2 + 12 = 19$.

Inference on Simple Example

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$	$p(x_1, x_2)$
0	0	1	1	2	2	0.11
0	1	1	3	1	3	0.16
1	0	2	1	1	2	0.11
1	1	2	3	2	12	0.63

- One inference task is finding Z:
  - In this case $Z = 2 + 3 + 2 + 12 = 19$.
- With Z you can find the probability of configurations:
  - E.g., $p(x_1 = 0, x_2 = 0) = 2/19 \approx 0.11$.

Inference on Simple Example

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$	$p(x_1, x_2)$
0	0	1	1	2	2	0.11
0	1	1	3	1	3	0.16
1	0	2	1	1	2	0.11
1	1	2	3	2	12	0.63

- One inference task is finding Z:
  - In this case $Z = 2 + 3 + 2 + 12 = 19$.
- With Z you can find the probability of configurations:
  - E.g., $p(x_1 = 0, x_2 = 0) = 2/19 \approx 0.11$.
- Inference also includes finding marginals like $p(x_1 = 1)$:

$$p(x_1 = 1) = \sum_{x_2} p(x_1 = 1, x_2) = p(x_1 = 1, x_2 = 0) + p(x_1 = 1, x_2 = 1)$$
$$= 2/19 + 12/19 = 0.737.$$

Sampling on Simple Example

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$	$p(x_1, x_2)$	cumsum
0	0	1	1	2	2	0.11	0.11
0	1	1	3	1	3	0.16	0.26
1	0	2	1	1	2	0.11	0.37
1	1	2	3	2	12	0.63	1.00

- Sampling is generating configurations according to $p(x_1, x_2)$:
  - E.g., 63% of the time we should return $x_1 = 1$ and $x_2 = 1$.

Sampling on Simple Example

x_1	x_2	ϕ_1	ϕ_2	$\phi_{1,2}$	$\tilde{p}(x_1, x_2)$	$p(x_1, x_2)$	cumsum
0	0	1	1	2	2	0.11	0.11
0	1	1	3	1	3	0.16	0.26
1	0	2	1	1	2	0.11	0.37
1	1	2	3	2	12	0.63	1.00

- Sampling is generating configurations according to $p(x_1, x_2)$:
  - E.g., 63% of the time we should return $x_1 = 1$ and $x_2 = 1$.
- To implement this:
  1. Generate a random number $u \in [0, 1]$.
  2. Find the smallest cumsum of the probabilities greater than u.
  - If $u = 0.59$ return $x_1 = 1$ and $x_2 = 1$
  - If $u = 0.12$ return $x_1 = 0$ and $x_2 = 1$.

Summary

- Parameter Learning in DAGs is easy: just fit each $p(x_j|x_{\mathsf{pa}(j)})$.

Summary

- Parameter Learning in DAGs is easy: just fit each $p(x_j|x_{\mathsf{pa}(j)})$.
- Tabular parameterization in DAGs for modelling discrete data.

Summary

- Parameter Learning in DAGs is easy: just fit each $p(x_j|x_{\mathsf{pa}(j)})$.
- Tabular parameterization in DAGs for modelling discrete data.
- Structured prediction is supervised learning with "objects" as labels.

Summary

- Parameter Learning in DAGs is easy: just fit each $p(x_j|x_{\mathsf{pa}(j)})$.
- Tabular parameterization in DAGs for modelling discrete data.
- Structured prediction is supervised learning with "objects" as labels.
- Undirected graphical models factorize probability into non-negative potentials.
  - Simple conditional independence properties.
  - Include Gaussians as special case.

Summary

- Parameter Learning in DAGs is easy: just fit each $p(x_j|x_{\text{pa}(j)})$.
- Tabular parameterization in DAGs for modelling discrete data.
- Structured prediction is supervised learning with "objects" as labels.
- Undirected graphical models factorize probability into non-negative potentials.
  - Simple conditional independence properties.
  - Include Gaussians as special case.
- Decoding, inference, and sampling tasks in UGMs.

- Next time: using graph structure for the 3 tasks, and learning potentials.