# CSCE 496/896 Lecture 11: Structured Prediction and Probabilistic Graphical Models

Stephen Scott and Vinod Variyam

(Adapted from Sebastian Nowozin and Christoph H. Lampert)

sscott@cse.unl.edu

Introduction
Definitions
Applications
Graphical Models
Training

We've long known how to answer the question:
**Does this picture contain a cat?**

E.g., convolutional layers feeding connected layers feeding softmax

# Introduction
## ... and in with the new.

What we want to know now is: **Where are the cats?**


No longer a classification problem; need more sophisticated (**structured**) output

# Outline

- Definitions
- Applications
- Graphical modeling of probability distributions
- Training models
- Inference

- Most machine learning approaches learn function $f : \mathcal{X} \rightarrow \mathbb{R}$
    - Inputs $\mathcal{X}$ are **any kind of objects**
    - Output $y$ is a **real number** (classification, regression, density estimation, etc.)
- Structured output learning approaches learn function $f : \mathcal{X} \rightarrow \mathcal{Y}$
    - Inputs $\mathcal{X}$ are **any kind of objects**
    - Outputs $y \in \mathcal{Y}$ are **complex (structured) objects** (images, text, audio, etc.)

# Definitions
## Structured Outputs (2)



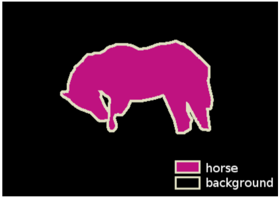
Can think of structured data as consisting of parts, where each part contains information, as well as how they fit together

- **Text:** Word sequence matters
- **Hypertext:** Links between documents matter
- **Chemical structures:** Relative positions of molecules matter
- **Images:** Relative positions of pixels matter

# Applications
## Image Processing

Semantic image segmentation: $f : \overbrace{\{\text{images}\}}^{\{0,\ldots,255\}^{3(m\times n)}} \rightarrow \overbrace{\{\text{masks}\}}^{\{0,1\}^{m\times n}}$

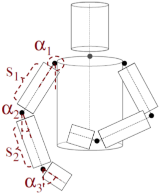
$\mapsto$



# Applications

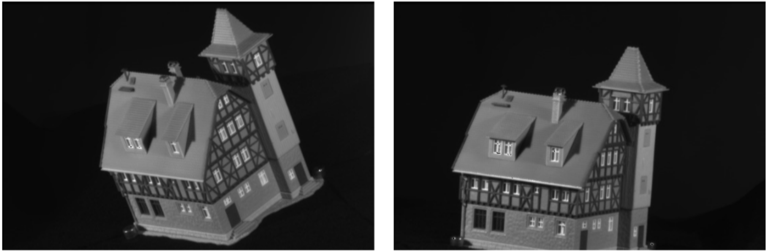
## Image Processing (2)

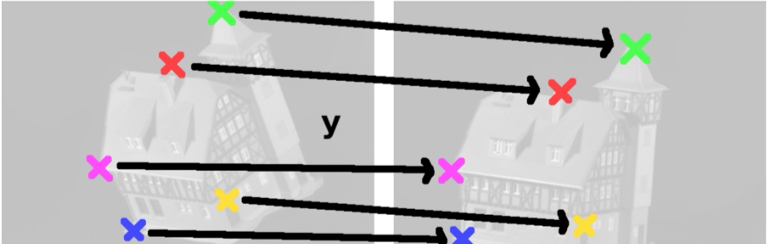
Pose estimation: $f : \overbrace{\{\text{images}\}}^{\{0,\ldots,255\}^{3(m\times n)}} \rightarrow \overbrace{\{K \text{ positions \& angles}\}}^{\mathbb{R}^{3K}}$




Point matching:

$f$ : {image pairs} $\rightarrow$ {mappings between images}





# Applications
## Image Processing (4)



Object localization

$f$ : {images} $\rightarrow$ {bounding box coordinates}

input: image


output: object position $(left, top$ $right, bottom)$

# Applications

## Others

- Natural language processing (e.g., translation; output is sentences)
- Bioinformatics (e.g., structure prediction; output is graphs)
- Speech processing (e.g., recognition; output is sentences)
- Robotics (e.g., planning; output is action plan)
- Image denoising (output is "clean" version of image)

Introduction
Definitions
Applications
Graphical Models
Directed
Undirected
Energy
Separation
Training

- To represent structured outputs, we will often employ probabilistic modeling
  - Joint distributions (e.g., $P(A, B, C)$)
  - Conditional distributions (e.g., $P(A \mid B, C)$)
- Can estimate joint and conditional probabilities by counting and normalizing, but have to be careful about representation

# Graphical Models
## Probabilistic Modeling (2)

- E.g., I have a coin with unknown probability $p$ of heads
- I want to estimate the probability of flipping it ten times and getting the sequence `HHTTHHTTTT`
- One way of representing this joint distribution is a single, big lookup table:

- Each experiment consists of ten coin flips
- For each outcome, increment its counter
- After $n$ experiments, divide `HHTTHHTTTT`'s counter by $n$ to get the estimate

| Outcome | Count |
|---|---|
| `TTHHTTHHTH` | 1 |
| `HHHTHTTTHH` | 0 |
| `HTTTTTHHHT` | 0 |
| `TTHTHTHHTT` | 1 |
| ⋮ | ⋮ |

- Will this work?

CSCE 496/896 Lecture 11: Structured Prediction and Probabilistic Graphical Models

Stephen Scott and Vinod Variyam

Introduction
Definitions
Applications
Graphical Models
Directed
Undirected
Energy
Separation
Training

- **Problem:** Number of possible outcomes grows exponentially with number of variables (flips)
  - $\Rightarrow$ Most outcomes will have count $= 0$, a few with 1, probably none with more
  - $\Rightarrow$ Lousy probability estimates
- Ten flips is bad enough, but consider 100 $\ddot{\frown}$
- How would **you** solve this problem?

CSCE 496/896 Lecture 11: Structured Prediction and Probabilistic Graphical Models

Stephen Scott and Vinod Variyam

Introduction
Definitions
Applications
Graphical Models
Directed
Undirected
Energy
Separation
Training

- Of course, we recognize that all flips are independent, so

$$\Pr[\mathtt{HHTTHHTTTT}] = p^4 (1-p)^6$$

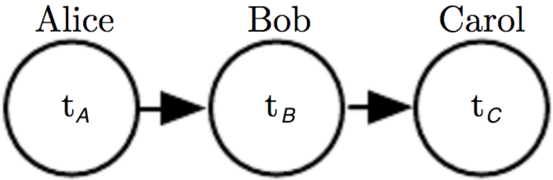
- So we can count $n$ coin flips to estimate $p$ and use the formula above
- I.e., we **factor** the joint distribution into independent components and multiply the results:

$$\Pr[\mathtt{HHTTHHTTTT}] = \Pr[f_1 = \mathtt{H}] \Pr[f_2 = \mathtt{H}] \Pr[f_3 = \mathtt{T}] \cdots \Pr[f_{10} = \mathtt{T}]$$

- We greatly reduce the number of parameters to estimate

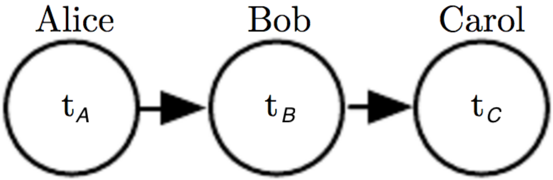
- Another example: **Relay racing team**
- Alice, then Bob, then Carol
- Let $t_A =$ Alice's finish time (in seconds), $t_B =$ Bob's, $t_C =$ Carol's
- Want to model the joint distribution $\Pr[t_A, t_B, t_C]$
- Let $t_C, t_B, t_A \in \{1, \ldots, 1000\}$
- How large would the table be for $\Pr[t_A, t_B, t_C]$?
- How many races must they run to populate the table?

- But we can factor this distribution by observing that $t_A$ is independent of $t_B$ and $t_C$
  $\Rightarrow$ Can estimate $t_A$ on its own
- Also, $t_B$ directly depends on $t_A$, but is independent of $t_C$
- $t_C$ directly depends on $t_B$, and indirectly on $t_A$
- Can display this graphically:



# Graphical Models
## Factoring a Distribution (4)






- This **directed graphical model** (often called a **Bayesian network** or **Bayes net**) represents conditional dependencies among variables
- Makes factoring easy:

$$\Pr[t_A, t_B, t_C] = \Pr[t_A] \Pr[t_B \mid t_A] \Pr[t_C \mid t_B]$$

$$\Pr[t_A, t_B, t_C] = \Pr[t_A] \Pr[t_B \mid t_A] \Pr[t_C \mid t_B]$$

- Table for $\Pr[t_A]$ requires[1] 1000 entries, while $\Pr[t_B \mid t_A]$ requires $10^6$, as does $\Pr[t_C \mid t_B]$
  - $\Rightarrow$ Total $2.001 \times 10^6$, versus $10^9$
- Idea easily extends to continuous distributions by changing discrete probability $\Pr[\cdot]$ to pdf $p(\cdot)$

[1] Technically, we only need 999 entries, since the value of the last one is implied since probabilities must sum to one. However, then the analysis requires the use of a lot of “9”s, and that’s not something I’m willing to take on at this point in my life.

**Definition**: $X$ is **conditionally independent** of $Y$ given $Z$ if the probability distribution governing $X$ is independent of the value of $Y$ given the value of $Z$; that is, if

$$(\forall x_i, y_j, z_k) \;\; \Pr[X = x_i \mid Y = y_j, Z = z_k] = \Pr[X = x_i \mid Z = z_k]$$

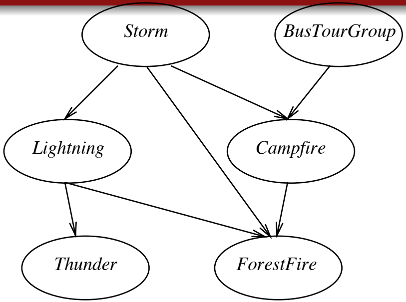
more compactly, we write

$$\Pr[X \mid Y, Z] = \Pr[X \mid Z]$$

**Example:** *Thunder* is conditionally independent of *Rain*, given *Lightning*

$$\Pr[Thunder \mid Rain, Lightning] = \Pr[Thunder \mid Lightning]$$

# Directed Models
## Definition




| | $S,B$ | $S,\neg B$ | $\neg S,B$ | $\neg S,\neg B$ |
|---|---|---|---|---|
| $C$ | 0.4 | 0.1 | 0.8 | 0.2 |
| $\neg C$ | 0.6 | 0.9 | 0.2 | 0.8 |

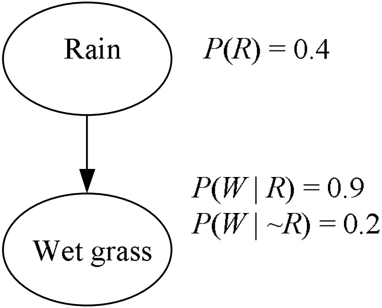

Network (directed acyclic graph) represents a set of conditional independence assertions:

- **Each node is asserted to be conditionally independent of its nondescendants, given its immediate predecessors**
- E.g., Given *Storm* and *BusTourGroup*, *Campfire* is CI of *Lightning* and *Thunder*

# Directed Models
## Causality



Can think of edges in a Bayes net as representing a **causal relationship** between nodes



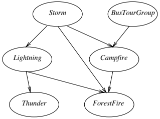

E.g., rain causes wet grass

Probability of wet grass depends on whether there is rain

# Directed Models

## Generative Models

Represents joint probability distribution over $\langle Y_1, \ldots, Y_n \rangle$, e.g., $\Pr[Storm, BusTourGroup, \ldots, ForestFire]$




| | S,B | S,¬B | ¬S,B | ¬S,¬B |
|---|---|---|---|---|
| C | 0.4 | 0.1 | 0.8 | 0.2 |
| ¬C | 0.6 | 0.9 | 0.2 | 0.8 |

Campfire

- In general, for $y_i$ = value of $Y_i$

$$\Pr[y_1, \ldots, y_n] = \prod_{i=1}^{n} \Pr[y_i \mid Parents(Y_i)]$$

  ($Parents(Y_i)$ denotes immediate predecessors of $Y_i$)
- E.g., $\Pr[S, B, C, \neg L, \neg T, \neg F] =$

$$\Pr[S] \cdot \Pr[B] \cdot \underbrace{\Pr[C \mid B, S]}_{0.4} \cdot \Pr[\neg L \mid S] \cdot \Pr[\neg T \mid \neg L] \cdot \Pr[\neg F \mid S, \neg L, \neg C]$$

- If variables continuous, use pdf $p(\cdot)$ instead of $\Pr[\cdot]$

# Directed Models
## Predicting Most Likely Label



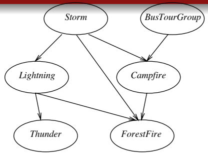
We sometimes call graphical models **generative** (vs **discriminative**) models since they can be used to generate instances $\langle Y_1, \ldots, Y_n \rangle$ according to joint distribution

Can use for classification

- Label $r$ to predict is one of the variables, represented by a node
- If we can determine the most likely value of $r$ given the rest of the nodes, can predict label
- One idea: Go through all possible values of $r$, and compute joint distribution (previous slide) with that value and other attribute values, then return one that maximizes

# Directed Models

## Predicting Most Likely Label (cont'd)




| | S,B | S,¬B | ¬S,B | ¬S,¬B |
|---|---|---|---|---|
| C | 0.4 | 0.1 | 0.8 | 0.2 |
| ¬C | 0.6 | 0.9 | 0.2 | 0.8 |


E.g., if *Storm* ($S$) is the label to predict, and we are given values of $B$, $C$, $\neg L$, $\neg T$, and $\neg F$, can use formula to compute $\Pr[S, B, C, \neg L, \neg T, \neg F]$ and $\Pr[\neg S, B, C, \neg L, \neg T, \neg F]$, then predict more likely one

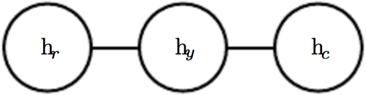
Easily handles unspecified attribute values

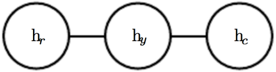
**Issue:** Takes time exponential in number of values of unspecified attributes

More efficient approach: **Pearl's message passing algorithm** for chains and trees and polytrees (at most one path between any pair of nodes)

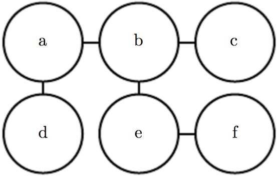
# Undirected Models

- Since directed edges imply causal relationships, might want to use **undirected** edges if causality not modeled
- E.g., let $h_y = 1$ if you are healthy, 0 if sick
  - $h_r$ same but for your roommate, $h_c$ for coworker
- $h_y$ and $h_r$ directly influence each other, but causality unknown and irrelevant
- $h_y$ and $h_c$ also directly influence each other
- $h_r$ and $h_c$ only indirect influence, via $h_y$
- Can model $\Pr[h_r, h_y, h_c]$ with **undirected model**, aka **Markov random field** (MRF), aka **Markov network**






- In directed models, factors defined by a node's parents: *conditionally indep. of nondescendants given parents*
- In undirected models, factors defined by maximal **cliques** (complete subgraphs): *conditionally indep. of all other variables given neighbors*
- In graph above, cliques are $\{\{h_r, h_y\}, \{h_y, h_c\}\}$
- In graph below, cliques are $\{\{a,d\},\{a,b\},\{b,c\},\{b,e\},\{e,f\}\}$



- Given clique $\mathcal{C} \in \mathcal{G}$ and $\boldsymbol{y}_{\mathcal{C}} =$ values on nodes in $\mathcal{C}$, **factor** $\phi_{\mathcal{C}}(\boldsymbol{y}_{\mathcal{C}})$ describes how likely they will co-exist
- Not quite a probability; need to **normalize** it first
- First go through all cliques $\mathcal{C}$, compute factor on $\mathcal{C}$ using values from $\boldsymbol{y}$:

$$\tilde{P}(\boldsymbol{y}) = \prod_{\mathcal{C} \in \mathcal{G}} \phi_{\mathcal{C}}(\boldsymbol{y}_{\mathcal{C}})$$

- Can convert this to a probability of $\boldsymbol{y}$ by normalizing:

$$\Pr[\boldsymbol{y}] = \tilde{P}(\boldsymbol{y})/Z \ ,$$

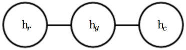
where $Z = \sum_{\boldsymbol{y} \in \mathcal{Y}} \tilde{P}(\boldsymbol{y})$ comes from summing (or integrating) over all possible values across all nodes
- $Z$ doesn't change if model doesn't

# Undirected Models

## Factors (3)

**Model:**

| $\phi(\mathcal{C}_{ry})$ | $h_y = 0$ | $h_y = 1$ |
|---|---|---|
| $h_r = 0$ | 2 | 1 |
| $h_r = 1$ | 1 | 10 |




| $\phi(\mathcal{C}_{yc})$ | $h_y = 0$ | $h_y = 1$ |
|---|---|---|
| $h_c = 0$ | 5 | 1 |
| $h_c = 1$ | 2 | 15 |

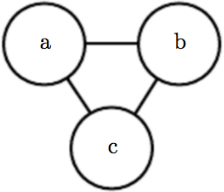
**Distribution:**

| $h_r$ | $h_y$ | $h_c$ | $\phi(\mathcal{C}_{ry})$ | $\phi(\mathcal{C}_{yc})$ | $\tilde{P}(\mathbf{y})$ | $\Pr[\mathbf{y}]$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 2 | 5 | 10 | 0.051 |
| 0 | 0 | 1 | 2 | 2 | 4 | 0.020 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0.005 |
| 0 | 1 | 1 | 1 | 15 | 15 | 0.076 |
| 1 | 0 | 0 | 1 | 5 | 5 | 0.025 |
| 1 | 0 | 1 | 1 | 2 | 2 | 0.010 |
| 1 | 1 | 0 | 10 | 1 | 10 | 0.051 |
| 1 | 1 | 1 | 10 | 15 | 150 | 0.762 |
| | | | | | $Z = 197$ | 1.0 |

What is time complexity of brute-force approach?

# Undirected Models

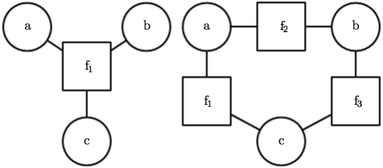
## Factor Graphs




- How do we interpret this MRF?
- Could be one factor: $\phi(\{a, b, c\})$
- Or, is it three: $\phi(\{a, b\}), \phi(\{a, c\}), \phi(\{b, c\})$

A **factor graph** makes explicit the scope of each factor $\phi$

$\phi(\{a, b, c\})$ $\phi(\{a, b\}), \phi(\{a, c\}), \phi(\{b, c\})$




Bipartite graph, so no circles or squares connected

Introduction
Definitions
Applications
Graphical Models
Directed
Undirected
Energy
Separation
Training

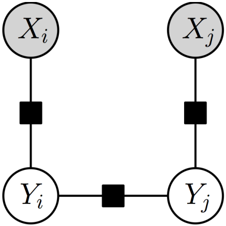
- Formally, a factor graph is a bipartite graph $(V, \mathcal{F}, \mathcal{E})$, where $V =$ **variable nodes**, $\mathcal{F} =$ **factor nodes** and edges $\mathcal{E} \subseteq V \times \mathcal{F}$ with one endpoint $V$ and one in $\mathcal{F}$
- The **scope** $N : \mathcal{F} \rightarrow 2^V$ of factor $f \in \mathcal{F}$ is the set of neighboring variables:

$$N(f) = \{i \in V : (i, f) \in \mathcal{E}\}$$

- Now compute distribution similar to before:

$$\Pr[\mathbf{y}] = \frac{1}{Z} \prod_{f \in \mathcal{F}} \phi_f(\mathbf{y}_{N(f)})$$

- A **conditional random field** (CRF) is a factor graph used to directly model a conditional distribution $\Pr[Y = \boldsymbol{y} \mid X = \boldsymbol{x}]$
- E.g., probability that a specific pixel $y$ is part of a cat given the **observation** (input image) $\boldsymbol{x}$




$$\Pr[Y_i = y_i, Y_j = y_j \mid X_i = x_i, X_j = x_j] =$$
$$\frac{1}{Z(x_i, x_j)} \phi_i(y_i; x_i)\phi_j(y_j; x_j)\phi_{i,j}(y_i, y_j)$$

$$\Pr[Y = \boldsymbol{y} \mid X = \boldsymbol{x}] = \frac{1}{Z(\boldsymbol{x})} \prod_{f \in \mathcal{F}} \phi_f(\boldsymbol{y}_f; \boldsymbol{x}_f)$$

$Z$ now depends on $\boldsymbol{x}$

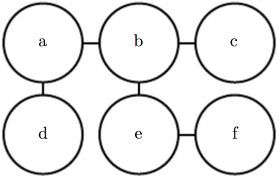
# Undirected Models

## Energy-Based Functions

- We now know how to factor the distribution graphically, but what form will $\phi(\cdot)$ take?
- Want to learn them to infer a distribution
- Need $\tilde{p}(\boldsymbol{x}) > 0$ for all $\boldsymbol{x}$ in order to get a distribution
- Define an **energy function** $E_f : \mathcal{Y}_{N(f)} \to \mathbb{R}$ for factor $f$
- Then define $\phi_f = \exp(-E_f(y_f)) > 0$ and get

$$
\begin{aligned}
p(Y = \boldsymbol{y}) &= \frac{1}{Z} \prod_{f \in \mathcal{F}} \phi_f(y_f) = \frac{1}{Z} \prod_{f \in \mathcal{F}} \exp\left(-E_f(y_f)\right) \\
&= \frac{1}{Z} \exp\left(-\sum_{f \in \mathcal{F}} E_f(y_f)\right)
\end{aligned}
$$

Using this form of $\phi$ allows us to factor our energy function as well!




$$E(a,b,c,d,e,f) = E_{a,b}(a,b) + E_{b,c}(b,c) + E_{a,d}(a,d) + E_{b,e}(b,e) + E_{e,f}(e,f)$$

# Undirected Models
## Energy-Based Functions (3)



- Still need a form for $E(\cdot)$ to parameterize and learn
- Define $E_f(\boldsymbol{y}_f; \boldsymbol{w})$ to depend on weight vector $\boldsymbol{w} \in \mathbb{R}^d$:

$$E_f : \mathcal{Y}_{N(f)} \times \mathbb{R}^d \to \mathbb{R}$$

- E.g., say we are doing binary image segmentation
  - Want adjacent pixes to try to take same value, so define $E_f : \{0,1\} \times \{0,1\} \times \mathbb{R}^2 \to \mathbb{R}$ as

$$E_f(0,0;\boldsymbol{w}) = E_f(1,1;\boldsymbol{w}) = w_1$$
$$E_f(0,1;\boldsymbol{w}) = E_f(0,1;\boldsymbol{w}) = w_2$$

  - We learn $w_1$ and $w_2$ from training data, expecting $w_1 > w_2$
  - More on this later

# Separation and D-Separation



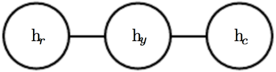
Introduction
Definitions
Applications
Graphical Models
Directed
Undirected
Energy
Separation
Training

- An edge between two nodes indicates a direct interaction between the variables
- Paths between nodes indicate **indirect** interactions
- Observing (instantiating) some variables change the interactions between others
- Useful to know which subsets of variables are conditionally independent from each other, given values of other variables
- Say that set of variables $\mathbb{A}$ is **separated** (if undirected model) or **d-separated** (if directed) from set $\mathbb{B}$ given set $\mathbb{S}$ if the graph implies that $\mathbb{A}$ and $\mathbb{B}$ are conditionally independent given $\mathbb{S}$

# Separation and D-Separation

## Example

Recall example on health of you, roommate, and coworker

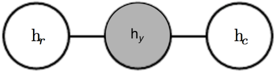



| $h_r$ | $\Pr[h_c = 0 \mid h_r]$ |
|---|---|
| 0 | $(10+1)/(10+4+1+15) = 11/30$ |
| 1 | $(5+10)/(5+2+10+150) = 15/167$ |

$\Rightarrow \Pr[h_c = 0]$ influenced by $h_r$

| $h_r$ | $h_y$ | $h_c$ | $\tilde{P}(\mathbf{y})$ |
|---|---|---|---|
| 0 | 0 | 0 | 10 |
| 0 | 0 | 1 | 4 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 15 |
| 1 | 0 | 0 | 5 |
| 1 | 0 | 1 | 2 |
| 1 | 1 | 0 | 10 |
| 1 | 1 | 1 | 150 |

What if we **know** that you are healthy ($h_y = 1$)?




| $h_r$ | $\Pr[h_c = 0 \mid h_y = 1, h_r]$ |
|---|---|
| 0 | $1/(1+15) = 1/16$ |
| 1 | $10/(10+150) = 10/160 = 1/16$ |

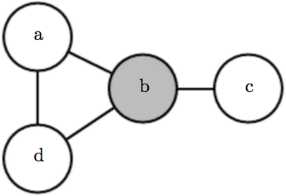
$\Rightarrow$ Given $h_y$, $h_c$ is CI from $h_r$

| $h_r$ | $h_y$ | $h_c$ | $\tilde{P}(\mathbf{y})$ |
|---|---|---|---|
| 0 | 0 | 0 | 10 |
| 0 | 0 | 1 | 4 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 15 |
| 1 | 0 | 0 | 5 |
| 1 | 0 | 1 | 2 |
| 1 | 1 | 0 | 10 |
| 1 | 1 | 1 | 150 |

# Separation and D-Separation
## Separation in Undirected Models



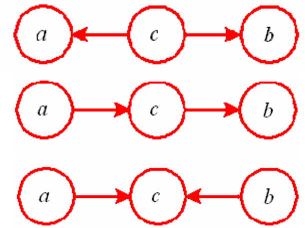
- If a variable is observed, it **blocks** all paths through it
- In an undirected model, two nodes are separated if all paths between them are blocked




- E.g., $a$ and $c$ are blocked, as are $d$ and $c$, but not $a$ and $d$ (even though one of their paths is blocked)

- In directed models, d-separation is more complicated
- Depends on the direction of the edges involved

- When considering nodes $a$ and $b$ connected via $c$, can classify connection as **tail-to-tail**, **head-to-tail**, and **head-to-head**

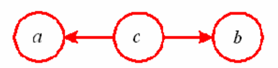



- For each case, assuming no other path exists (ignoring edge direction) between $a$ and $b$, we will determine if $a$ and $b$ are independent, or conditionally independent given $c$

# Separation and D-Separation

## D-Separation in Directed Models: Tail-to-Tail






E.g., $a$ = car won't start, $b$ = lights work, $c$ = battery low

$\Pr[c = 1] = 1/2$

| $c$ | $\Pr[a = 1 \mid c]$ |
|---|---|
| 0 | 1/3 |
| 1 | 1/2 |

| $c$ | $\Pr[b = 1 \mid c]$ |
|---|---|
| 0 | 4/5 |
| 1 | 1/10 |

- Factorization:

$$\Pr[a, b, c] = \Pr[a \mid c] \Pr[b \mid c] \Pr[c]$$

- When $c$ unknown, get $\Pr[a, b]$ by marginalizing:

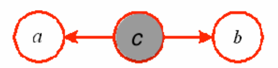
$$\Pr[a, b] = \sum_c \Pr[a \mid c] \Pr[b \mid c] \Pr[c] \quad ,$$

which generally does not equal $\Pr[a] \Pr[b]$
$\Rightarrow$ $a$ and $b$ not independent

# Separation and D-Separation

## D-Separation in Directed Models: Tail-to-Tail (2)





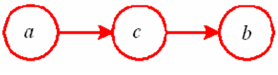

E.g., $c = 1$ (battery low)

- When conditioning on $c$:

$$\Pr[a, b \mid c] = \frac{\Pr[a, b, c]}{\Pr[c]} = \frac{\Pr[c] \Pr[a \mid c] \Pr[b \mid c]}{\Pr[c]} = \Pr[a \mid c] \Pr[b \mid$$

- Thus $a$ and $b$ conditionally independent given $c$ (car not starting independent of lights working)
- Say that connection between $a$ and $b$ is **blocked** by $c$ when it is observed and **unblocked** when unobserved
- Always true for uncoupled tail-to-tail connections (where there's no edge between $a$ and $b$)

Introduction
Definitions
Applications
Graphical Models
Directed
Undirected
Energy
Separation
Training




E.g., $a =$ leave on time, $b =$ on time for work, $c =$ catch the ferry

$\Pr[a = 1] = 1/2$

| $a$ | $\Pr[c = 1 \mid a]$ |
|---|---|
| 0 | 1/3 |
| 1 | 1/2 |

| $c$ | $\Pr[b = 1 \mid c]$ |
|---|---|
| 0 | 1/5 |
| 1 | 9/10 |

- Factorization:

$$\Pr[a, b, c] = \Pr[a] \Pr[c \mid a] \Pr[b \mid c]$$

- When $c$ unknown, get $\Pr[a, b]$ by marginalizing:

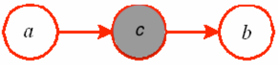
$$\Pr[a, b] = \Pr[a] \sum_c \Pr[c \mid a] \Pr[b \mid c] = \Pr[a] \Pr[b \mid a] \ ,$$

which generally does not equal $\Pr[a] \Pr[b]$
$\Rightarrow$ $a$ and $b$ not independent

# Separation and D-Separation

## D-Separation in Directed Models: Head-to-Tail (2)






E.g., $c = 1$ (catch ferry)

- When conditioning on $c$:

$$\Pr[a, b \mid c] = \frac{\Pr[a, b, c]}{\Pr[c]} = \frac{\Pr[a]\Pr[c \mid a]\Pr[b \mid c]}{\Pr[c]} = \Pr[a \mid c]\Pr[b \mid$$

- Thus $a$ and $b$ conditionally independent given $c$ (on time for work independent of leaving on time)
- Say that connection between $a$ and $b$ is blocked by $c$ when it is observed and unblocked when unobserved
- Always true for uncoupled head-to-tail connections

CSCE 496/896 Lecture 11: Structured Prediction and Probabilistic Graphical Models

Stephen Scott and Vinod Variyam

Introduction

Definitions

Applications

Graphical Models

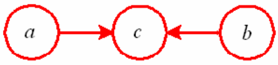
Directed

Undirected

Energy

Separation

Training


E.g., $a$ = rain, $b$ = sprinkler, $c$ = wet grass

$\Pr[a = 1] = 1/4$, $\Pr[b = 1] = 1/3$

| $a$ | $b$ | $\Pr[c = 1 \mid a, b]$ |
|---|---|---|
| 0 | 0 | 1/10 |
| 0 | 1 | 6/10 |
| 1 | 0 | 4/5 |
| 1 | 1 | 10/11 |

- Factorization:

$$P(a, b, c) = P(a)P(b)P(c \mid a, b)$$

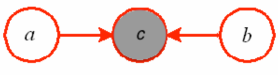
- When $c$ unknown, get $P(a, b)$ by marginalizing:

$$P(a, b) = P(a)P(b) \sum_c P(c \mid a, b) = P(a)P(b)$$

$\Rightarrow$ $a$ and $b$ are independent

# Separation and D-Separation

## D-Separation in Directed Models: Head-to-Head (2)



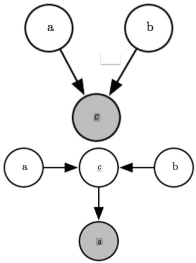

E.g., $c = 1$ (grass wet)

- When conditioning on $c$:

$$\Pr[a, b \mid c] = \frac{\Pr[a, b, c]}{\Pr[c]} = \frac{\Pr[a]\Pr[b]\Pr[c \mid a, b]}{\Pr[c]} ,$$

which generally does not equal $\Pr[a \mid c]\Pr[b \mid c]$

- $a$-$b$ connection blocked by $c$ when $c$ **unobserved** and unblocked when observed (also unblocks if one of $c$'s **descendants** observed)
- E.g., if grass wet and not raining, $\Pr[b = 1]$ increases
- Always true for uncoupled head-to-head connections



CSCE 496/896 Lecture 11: Structured Prediction and Probabilistic Graphical Models

Stephen Scott and Vinod Variyam

Introduction

Definitions

Applications

Graphical Models

Directed

Undirected

Energy

Separation

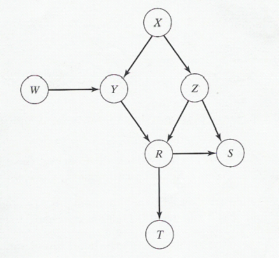
Training

$W$ and $T$:

- $[W, Y, R, T]$ blocked by $Y$ or $R$
- $[W, Y, X, Z, R, T]$ blocked by $X$ or $Z$ or $R$
- $[W, Y, X, Z, S, R, T]$ blocked by $X$ or $Z$ or $R$ but **not** by $S$ since observing $S$ unblocks the chain

$Y$ and $T$:

- $[Y, R, T]$ blocked by $R$
- $[Y, X, Z, R, T]$ blocked by $X$ or $Z$ or $R$
- $[Y, X, Z, S, R, T]$ blocked by $X$ or $Z$ or $R$

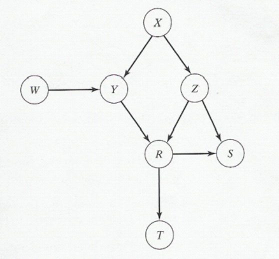


$W$ and $S$:

- $[W, Y, R, S]$ blocked by $Y$ or $R$
- $[W, Y, X, Z, R, S]$ blocked by $X$ or $Z$ or $R$
- $[W, Y, X, Z, S]$ blocked by $X$ or $Z$
- $[W, Y, R, Z, S]$ blocked by $Y$ or $Z$

$Y$ and $S$:

- $[Y, R, S]$ blocked by $R$
- $[Y, R, Z, S]$ blocked by $Z$
- $[Y, X, Z, R, S]$ blocked by $X$ or $Z$ or $R$
- $[Y, X, Z, S]$ blocked by $X$ or $Z$

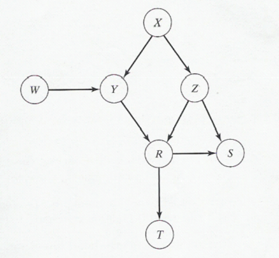



Thus $\{W, Y\}$ and $\{S, T\}$ are CI given $\{R, Z\}$

$W$ and $X$:

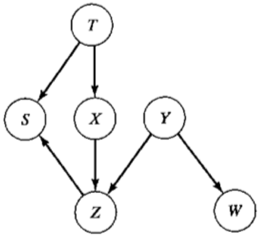
- Chain $[W, Y, X]$ blocked by $Y$ when not observed
- Chain $[W, Y, R, Z, X]$ blocked by $R$ when not observed
- Chain $[W, Y, R, S, Z, X]$ blocked by $S$ when not observed

Thus $W$ and $X$ are independent



# Markov Blankets






- Let $\mathcal{V}$ be a set of random variables (nodes), and $X \in \mathcal{V}$. A **Markov blanket** $\mathcal{M}_X$ of $X$ is any set of variables such that $X$ is CI of all other variables given $\mathcal{M}_X$
- If no proper subset of $\mathcal{M}_X$ is a Markov blanket, then $\mathcal{M}_X$ is a **Markov boundary**

- **Theorem:** The set of $X$'s parents, children, and co-parents (other parents of $X$'s children) form a Markov blanket of $X$
- Node $X$ has Markov blanket $\{T, Y, Z\}$

- Learning a CRF with input $\boldsymbol{x}$, parameterized by weight vector $\boldsymbol{w}$:

$$\Pr[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}] = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{w})} \exp\left(-E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w})\right)$$

  where $Z(\boldsymbol{x}, \boldsymbol{w}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp\left(-E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w})\right)$
- Let energy function $E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) = \langle \boldsymbol{w}, \varphi(\boldsymbol{x}, \boldsymbol{y}) \rangle$
  - I.e., a weighted sum of features produced by **feature function** $\varphi(\boldsymbol{x}, \boldsymbol{y})$
  - $\varphi(\boldsymbol{x}, \boldsymbol{y})$ could be a deep network, possibly trained earlier
  - $\boldsymbol{w}$ is trained to get $\Pr_P[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}]$ "close" to the true distribution $\Pr_D[\boldsymbol{y} \mid \boldsymbol{x}]$

# Learning Graphical Models

## Conditional Random Fields (2)

- Want $\boldsymbol{w}$ such that $\Pr_P[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}]$ is close to the true distribution $\Pr_D[\boldsymbol{y} \mid \boldsymbol{x}]$
- Measure distance via **Kullback-Leibler (KL) divergence**: for any $\boldsymbol{x} \in \mathcal{X}$ we have

$$\mathrm{KL}(P \| D) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \Pr_D[\boldsymbol{y} \mid \boldsymbol{x}] \log \frac{\Pr_D[\boldsymbol{y} \mid \boldsymbol{x}]}{\Pr_P[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}]}$$

- By marginalizing over all $\boldsymbol{x} \in \mathcal{X}$ we get

$$\mathrm{KL}_{tot}(P \| D) = \sum_{\boldsymbol{x} \in \mathcal{X}} \Pr_D[\boldsymbol{x}] \sum_{\boldsymbol{y} \in \mathcal{Y}} \Pr_D[\boldsymbol{y} \mid \boldsymbol{x}] \log \frac{\Pr_D[\boldsymbol{y} \mid \boldsymbol{x}]}{\Pr_P[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}]}$$

# Learning Graphical Models

## Conditional Random Fields (3)



- Goal is to find weights yielding close distribution, so

$$
\begin{aligned}
\boldsymbol{w}^* &= \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^d} \mathsf{KL}_{tot}(P\|D) \\
&= \operatorname*{argmax}_{\boldsymbol{w}\in\mathbb{R}^d} \sum_{\boldsymbol{x}\in\mathcal{X}} \Pr_D[\boldsymbol{x}] \sum_{\boldsymbol{y}\in\mathcal{Y}} \Pr_D[\boldsymbol{y} \mid \boldsymbol{x}] \log \Pr_P[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}] \\
&= \operatorname*{argmax}_{\boldsymbol{w}\in\mathbb{R}^d} \sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{\boldsymbol{y}\in\mathcal{Y}} \Pr_D[\boldsymbol{x}] \Pr_D[\boldsymbol{y} \mid \boldsymbol{x}] \log \Pr_P[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}] \\
&= \operatorname*{argmax}_{\boldsymbol{w}\in\mathbb{R}^d} \sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{\boldsymbol{y}\in\mathcal{Y}} \Pr_D[\boldsymbol{x}, \boldsymbol{y}] \log \Pr_P[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}] \\
&= \operatorname*{argmax}_{\boldsymbol{w}\in\mathbb{R}^d} \mathsf{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D} \left[\log \Pr_P[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}]\right] \\
&\approx \operatorname*{argmax}_{\boldsymbol{w}\in\mathbb{R}^d} \sum_{(\boldsymbol{x}^n,\boldsymbol{y}^n)\in\mathcal{D}} \log \Pr_P[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}]
\end{aligned}
$$

for training data $\mathcal{D}$

- I.e., we choose a model ($\boldsymbol{w}^*$) that **maximizes the conditional log likelihood** of the data
  - If all $(\boldsymbol{x}, \boldsymbol{y})$ instances are drawn iid, then $\boldsymbol{w}^*$ maximizes the probability of seeing all the $\boldsymbol{y}$s given all the $\boldsymbol{x}$s
- Throw in a regularizer for good measure
- **Definition:** Let $\Pr[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}] = \frac{1}{Z(\boldsymbol{x},\boldsymbol{w})} \exp\left(-\langle \boldsymbol{w}, \varphi(\boldsymbol{x}, \boldsymbol{y})\rangle\right)$ be a probability distribution parameterized by $\boldsymbol{w} \in \mathbb{R}^d$ and let $\mathcal{D} = \{(\boldsymbol{x}^n, \boldsymbol{y}^n)\}_{n=1,\ldots,N}$ be a set of training examples. For any $\lambda > 0$, **regularized maximum conditional likelihood (RMCL)** training chooses

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^d} \lambda \|\boldsymbol{w}\|^2 + \sum_{n=1}^{N} \langle \boldsymbol{w}, \varphi(\boldsymbol{x}^n, \boldsymbol{y}^n)\rangle + \sum_{n=1}^{N} \log Z(\boldsymbol{x}^n, \boldsymbol{w})$$

**Goal:** find $\boldsymbol{w}$ minimizing

$$\mathcal{L}(\boldsymbol{w}) = \lambda\|\boldsymbol{w}\|^2 + \sum_{n=1}^{N} \langle \boldsymbol{w}, \varphi(\boldsymbol{x}^n, \boldsymbol{y}^n)\rangle + \sum_{n=1}^{N} \log Z(\boldsymbol{x}^n, \boldsymbol{w})$$

Compute the gradient:

$$\begin{aligned}
\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}) &= 2\lambda\boldsymbol{w} + \sum_{n=1}^{N}\left[\varphi(\boldsymbol{x}^n, \boldsymbol{y}^n) - \sum_{\boldsymbol{y}\in\mathcal{Y}}\left(\frac{\exp(-\langle \boldsymbol{w}, \varphi(\boldsymbol{x}^n, \boldsymbol{y})\rangle)}{\sum_{\boldsymbol{y}'\in\mathcal{Y}}\exp(-\langle \boldsymbol{w}, \varphi(\boldsymbol{x}^n, \boldsymbol{y}')\rangle)}\right)\varphi(\boldsymbol{x}^n, \boldsymbol{y})\right] \\
&= 2\lambda\boldsymbol{w} + \sum_{n=1}^{N}\left[\varphi(\boldsymbol{x}^n, \boldsymbol{y}^n) - \sum_{\boldsymbol{y}\in\mathcal{Y}} \Pr_P[\boldsymbol{y} \mid \boldsymbol{x}^n, \boldsymbol{w}]\varphi(\boldsymbol{x}^n, \boldsymbol{y})\right] \\
&= \boxed{2\lambda\boldsymbol{w} + \sum_{n=1}^{N}\left[\varphi(\boldsymbol{x}^n, \boldsymbol{y}^n) - \mathsf{E}_{\boldsymbol{y}\sim P(\boldsymbol{y}|\boldsymbol{x}^n, \boldsymbol{w})}\left[\varphi(\boldsymbol{x}^n, \boldsymbol{y})\right]\right]}
\end{aligned}$$

# Learning Graphical Models

## Conditional Random Fields: RMCL (3)

- The gradient has a nice, compact form, and is **convex**
  - $\Rightarrow$ Any local optimum is a global one
- **Problem:** Computing expectation requires summing over exponentially many combinations of values of $\boldsymbol{y}$
- We can factor energy function, and therefore its derivative, and therefore the expectation of its derivative
- Let's focus on an individual factor $f$:

$$\mathrm{E}_{\boldsymbol{y}_f \sim P(\boldsymbol{y}_f \mid \boldsymbol{x}^n, \boldsymbol{w})} \left[ \varphi_f(\boldsymbol{x}^n, \boldsymbol{y}_f) \right] = \sum_{\boldsymbol{y}_f \in \mathcal{Y}_f} \Pr_P(\boldsymbol{y}_f \mid \boldsymbol{x}, \boldsymbol{w}) \varphi_f(\boldsymbol{x}^n, \boldsymbol{y}_f)$$

- Summation still has exponentially many terms, but instead of $K^{|V|}$ now it's $K^{|N(f)|}$ (more manageable)
- Still need to compute each factor's marginal probability

- Efficient **inference** of marginal probabilities and $Z$ in a graphical model is itself a major research area
- Depends on the structural model we're using
- Start with **belief propagation** in acyclic models
- Then approximate **loopy belief propagation** for cyclic models

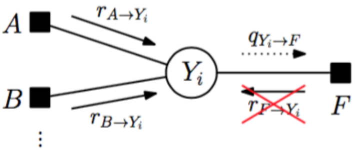
- **Belief propagation** is a general approach to inference in directed and undirected graphical models
- Generally, some node $i$ sends a message to another node $j$ regarding $i$'s belief about variable $y$
  - $i$ informs $j$ its belief about marginal probability $\Pr[y]$
  - E.g., message value high $\Rightarrow$ belief is $\Pr[y]$ also high
  - Each node messages each of its neighbors about its belief for each value of the random variable
- **Sum-Product Algorithm** uses belief propagation to find marginal probabilities and $Z$ in **tree-structured** factor graphs (connected and acyclic)
- Each edge $(i, f) \in \mathcal{E} \subseteq V \times \mathcal{F}$ has
  1. $q_{Y_i \to f} \in \mathbb{R}^{|\mathcal{Y}_i|}$ is a **variable-to-factor** message
  2. $r_{f \to Y_i} \in \mathbb{R}^{|\mathcal{Y}_i|}$ is a **factor-to-variable** message
- Note they are vector quantities, one component per value of $Y_i$

## Variable-to-Factor Message

- For variable $i \in V$, let

$$M(i) = \{f \in \mathcal{F} : (i,f) \in \mathcal{E}\}$$

be the set of factors adjacent to $i$




- For each value $y_i$ of variable $i$, variable-to-factor message is

$$q_{Y_i \to f}(y_i) = \sum_{f' \in M(i) \setminus \{f\}} r_{f' \to Y_i}(y_i)$$

- Variable node $i$ sums up all factor-to-variable messages from all factors except $f$ and transmits result to $f$

CSCE 496/896 Lecture 11: Structured Prediction and Probabilistic Graphical Models

Stephen Scott and Vinod Variyam

Introduction

Definitions

Applications

Graphical Models

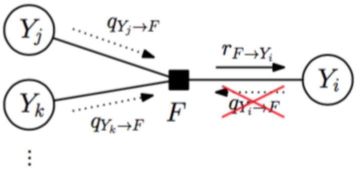
Training

## Factor-to-Variable Message

- For factor $f \in \mathcal{F}$, recall

$$N(f) = \{i \in V : (i,f) \in \mathcal{E}\}$$

is the set of variables adjacent to $f$




- For each value $y_i$ of variable $i$, factor-to-variable message is

$$r_{f \to Y_i}(y_i) = \log \sum_{\substack{y'_f \in \mathcal{Y}_f, \\ y'_i = y_i}} \exp \left( -E_f(y'_f) + \sum_{j \in N(f) \setminus \{i\}} q_{Y_j \to f'}(y'_i) \right)$$

- Factor node $f$ sums up all variable-to-factor messages from all variables except $i$ and transmits result to $i$

# Learning Graphical Models

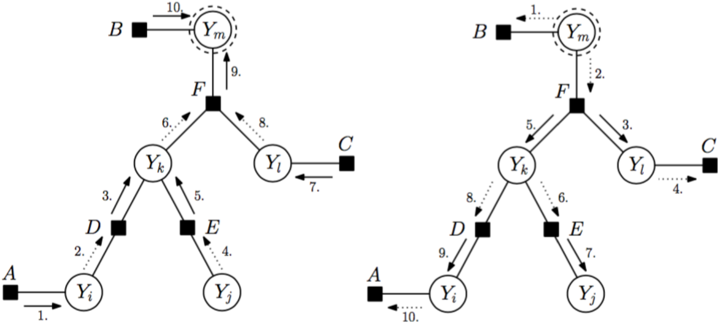
## Inference: Sum-Product Algorithm (4)



- Since we have a tree structure, there is always at least one variable adjacent to only one factor or one factor adjacent to one variable
- These messages depend on nothing, so start there
- Then order the other message computations via precedence graph
- Designate an arbitrary variable node to be the root
- Two phases of algorithm:
  1. **Leaf-to-root** phase: start at leaves and compute messages toward root
  2. **Root-to-leaf** phase: start at root and compute messages toward leaves

# Learning Graphical Models

## Inference: Sum-Product Algorithm (5)




After two phases, all messages computed

# Learning Graphical Models

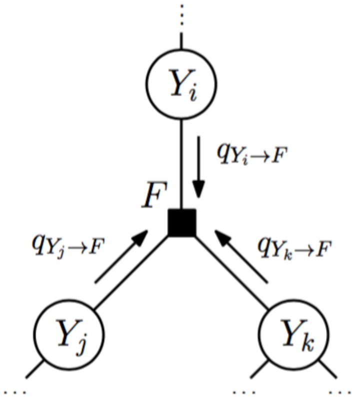
## Inference: Sum-Product Algorithm (6)

To **compute** $Z$, sum over factor-to-variable messages directed to root $Y_r$:

$$\log Z = \log \sum_{y_r \in \mathcal{Y}_r} \exp\left( \sum_{f \in M(r)} r_{f \to Y_r}(y_r) \right)$$

To **compute factor marginals:**

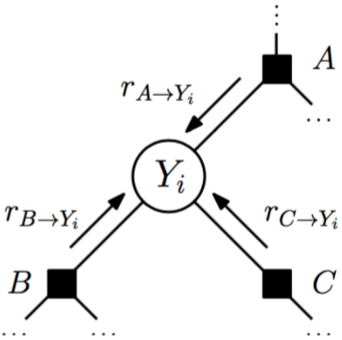
$$\mu_f(\mathbf{y}_f) = \Pr[Y_f = \mathbf{y}_f] = \exp\left(-E_f(\mathbf{y}_f) + \sum_{i \in N(f)} q_{Y_i \to f}(y_i) - \log Z\right)$$

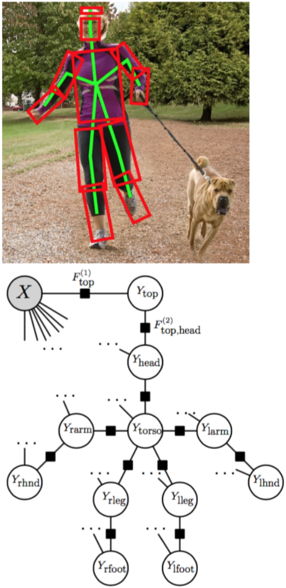


Introduction
Definitions
Applications
Graphical Models
Training

To **compute variable marginals:**

$$\Pr[Y_i = y_i] = \exp\left(\sum_{f \in M(i)} r_{f \to Y_i}(y_i) - \log Z\right)$$



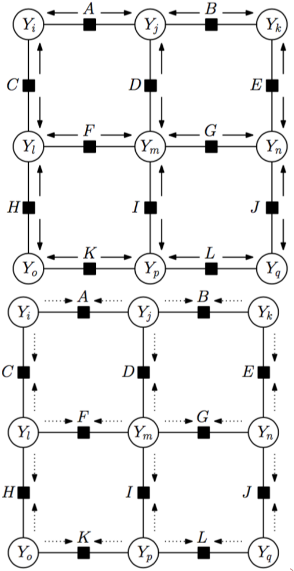



- E.g., $E_{f^{(1)}_{\text{top}}}(y_{\text{top}}; \boldsymbol{x})$ is energy function for factor $f_{\text{top}}$ representing top of person
- $\boldsymbol{x}$ is observed image and $Y_{\text{top}}$ is tuple $(a, b, s, \theta)$ where $(a, b)$ are coordinates, $s$ is scale, and $\theta$ is rotation
- $E_{f^{(2)}_{\text{top,head}}}(y_{\text{top}}, y_{\text{head}})$ relates adjecnt pairs of variables

# Learning Graphical Models

## Inference: Loopy Belief Propagation

- When graph has a cycle, can still perform message passing to **approximate** $Z$ and marginal probabilities
- Initialize messages to fixed value
- Perform updates in random order until convergence
- Factor-to-variable messages $r_{f \to Y_i}$ computed as before
- Variable-to-factor messages computed differently



CSCE 496/896 Lecture 11: Structured Prediction and Probabilistic Graphical Models

Stephen Scott and Vinod Variyam

Introduction

Definitions

Applications

Graphical Models

Training

**Variable-to-factor messages:**

$$\bar{q}_{Y_i \to f}(y_i) = \sum_{f' \in M(i) \setminus \{f\}} r_{f' \to Y_i}(y_i)$$

$$\delta = \log \sum_{y_i \in \mathcal{Y}_i} \exp\left(\bar{q}_{Y_i \to f}(y_i)\right)$$

$$q_{Y_i \to f}(y_i) = \bar{q}_{Y_i \to f}(y_i) - \delta$$

To **compute factor marginals:**

$$\begin{aligned}
\bar{\mu}_f(\boldsymbol{y}_f) &= -E_f(\boldsymbol{y}_f) + \sum_{j \in N(f)} q_{Y_j \to f}(y_j) \\
z_f &= \log \sum_{\boldsymbol{y}_f \in \mathcal{Y}_f} \exp(\bar{\mu}_f(\boldsymbol{y}_f)) \\
\mu_f(\boldsymbol{y}_f) &= \exp\left(\bar{\mu}_f(\boldsymbol{y}_f) - z_f\right)
\end{aligned}$$

To **compute variable marginals:**

$$\begin{aligned}
\bar{\mu}_i(y_i) &= \sum_{f' \in M(i)} r_{f' \to Y_i}(y_i) \\
z_i &= \log \sum_{y_i \in \mathcal{Y}_i} \exp(\bar{\mu}_i(y_i)) \\
\mu_i(y_i) &= \exp\left(\bar{\mu}_i(y_i) - z_i\right)
\end{aligned}$$

To **compute** $Z$**:**

$$\log Z = \sum_{i \in V} (|M(i) - 1|) \left[ \sum_{y_i \in \mathcal{Y}_i} \mu_i(y_i) \log \mu_i(y_i) \right] - \sum_{f \in \mathcal{F}} \sum_{\boldsymbol{y}_f \in \mathcal{Y}_f} \mu_f(\boldsymbol{y}_f)(E_f(\boldsymbol{y}_f) + \log \mu_f(\boldsymbol{y}_f))$$

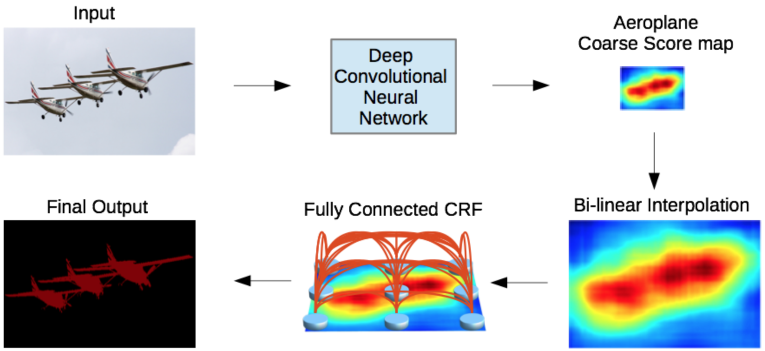
Chen et al. (2015): Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs

- Adapted DCNN ResNet-101 (trained for image classification) to the task of semantic segmentation
- Replaced connected layer with a “de-convolution” layer to upscale to original resolution for segmented image
- Result effective, but segment edges blurred
- Used CRF to sharpen

# Learning Graphical Models

## Conditional Random Fields: Case Study (2): Overview






- Score map generated as output of DCNN interpolated to input resolution
- Right area, but boundary of high-scoring region is fuzzy
- CRF sharpens to final output

- Energy function:

$$E(\mathbf{y}) = \sum_i \theta_i(y_i) + \sum_{i,j} \theta_{ij}(y_i, y_j)$$

  where $y_i \in \{0, 1\}$ is label assignment for pixel $i$
- Use $\theta_i(y_i) = -\log P(y_i)$ and

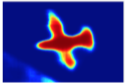
$$\theta_{ij}(y_i, y_j) = \mu(y_i, y_j) \left[ w_1 \exp\left( -\frac{\|p_i - p_j\|^2}{2\sigma_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\sigma_\beta^2} \right) + w_2 \exp\left( -\frac{\|p_i - p_j\|^2}{2\sigma_\gamma^2} \right) \right]$$

  where
  - $\mu(y_i, y_j) = 1$ iff $y_i \neq y_j$ (different labels)
  - $p_i$ = position of pixel $i$
  - $I_i$ = RGB color of pixel $i$
  - $\sigma$ = parameters
- Inference via specialized algorithms for Gaussian-based functions

# Learning Graphical Models

## Conditional Random Fields: Case Study (3): CRF Training Example

Image/G.T.



DCNN output

CRF Iteration 1

CRF Iteration 2

CRF Iteration 10