# Administrivia,
# Introduction to Structured Prediction

4/4/2017

CS159: Advanced Topics in Machine Learning

# Course Details

- Instructors: Taehwan Kim and Yisong Yue
- TAs:

Hoang Le

Jialin Song

Stephan Zhang

- Course Website: https://taehwanptl.github.io/

# Style of Course

- Graduate level course
- Give students an overview of topics
- Dig deep into one topic for final project
- Assume students are mathematically mature
  - ➡ Goal is to understand basic concepts
  - ➡ Understand specific mathematical details depending on your interest

# Grading Breakdown

- Participation (20%)
- Mini-quizzes (10%)
- Final Project (70%)

# Paper Reading & Discussion

- Paper Reading Course
  - ➡ Reading assignments for each lecture
  - ➡ Lectures more like discussion
- Student presentations
  - ➡ Presentation schedule signup soon
  - ➡ Present in groups
  - ➡ Can choose which paper(s) to present

# Mini-quizzes

- Evening after every lecture

➡ Very short

➡ Easy if you read material & attended lecture

- Released via Piazza

➡ Also use Piazza for Q&A

# Final Project

- Can be on any topic related to the course
- Work in groups
- Will release timeline of progress reports soon

# Topics

- Graphical Models
- Inference Methods
  - Message Passing, Integer Programs, Dynamic Programming, Variational Methods
- Classical Discriminative Learning
  - Structured SVM, Structured Perceptron, Conditional Random Fields
- Non-Linear Approaches
  - Structured Random Forests, Deep Structured Prediction
- More Complex Structures
  - Hierarchical Classification, Sequence Prediction/Generation
- Applications: Computer Vision, Speech Recognition, NLP, etc.

# Focus of Course

- Rigorous algorithm design

➡ Math intensive, but nothing too hard

➡ Will walk through relevant math in class

- Apply to interesting applications

➡ What are the right ways to model a problem?

# What Does Rigorous Mean?

- Formal model

➡ Explicitly state your assumptions

- Rigorously reason about how your algorithm solves the model

➡ Sometimes with provable guarantees

- Argue that your model is a reasonable one

# What Makes a Good Final Project?

- Pure Theory

➡ Study proof techniques, try to extend proof, or apply to new setting

- Algorithms

➡ Extend algorithms, design new ones, for new settings

- Modeling

➡ Model new setting, what are the right assumptions?

# Outline

- First two lectures

➡ Review basic methods

- Topics/readings chosen by students

➡ With curating from instructors & TAs

➡ List of papers will be on website

- But is negotiable

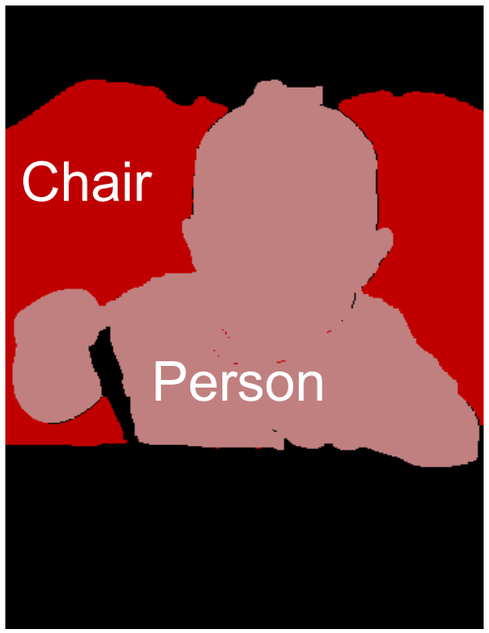
# Introduction to Structured Prediction

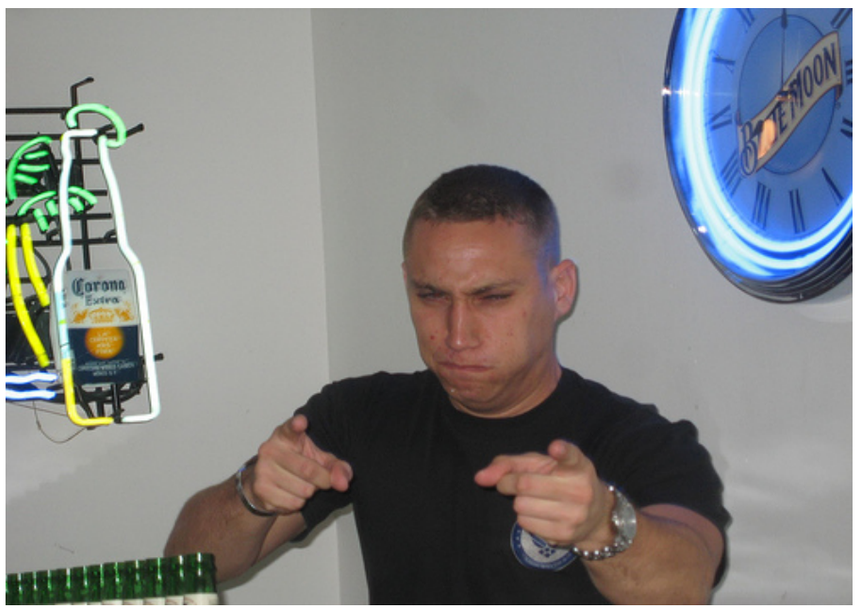
# Basic Formulation

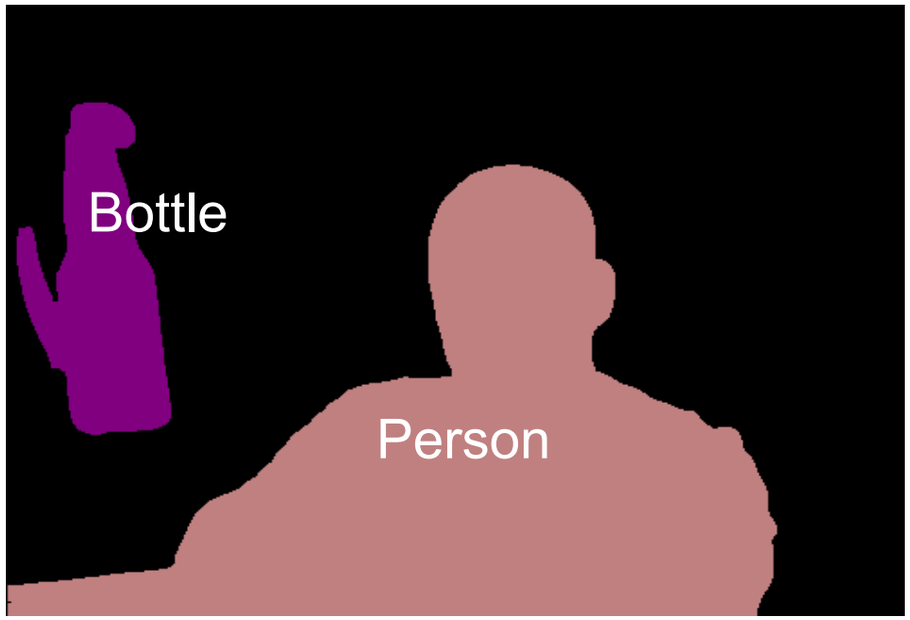
- "Normal" Machine Learning: $f : X \rightarrow \mathbb{R}$
  - ➡ Inputs *X* can be any kinds of objects
  - ➡ Output $y$ is a *real number*
    - classification, regression, …

- Structured Output Learning: $f : X \rightarrow Y$
  - ➡ Inputs *X* can be any kinds of objects
  - ➡ Outputs $y \in Y$ are *complex (structured) objects*
    - images, text, audio, folds of a protein, …

# Example: Image Segmentation


Li et al, 2010

# Example: Part-of-Speech Tagging

*[Example taken from Michael Collins]*

**INPUT**

Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

$x$

**OUTPUT**

Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N ./.

$y = (\mathrm{N,V,P,N,N,...})$

N = noun
V = verb
P = preposition
ADV = adverb
...

**Challenge**:

Predict a tag sequence

# Example: Image Caption Generation

**Our Approach (Generated Tuples)**

<act=puppy, pre=sit, loc=house>
<act=dog, pre=sit, loc=bed>

**Top 5 sentences generated using CNNS**

a man and woman laying down to the couch with a bed.
a man laying on top of a sofa with his dog.
a man laying on the couch with his dog.
a man is laying on the bed of a sofa.
a man and dog bed in the back of a sofa.

**Our Approach (Generated Tuples)**

<act=boy, pre=sit, loc=street>
<act=boy, pre=sit, loc=soccer>

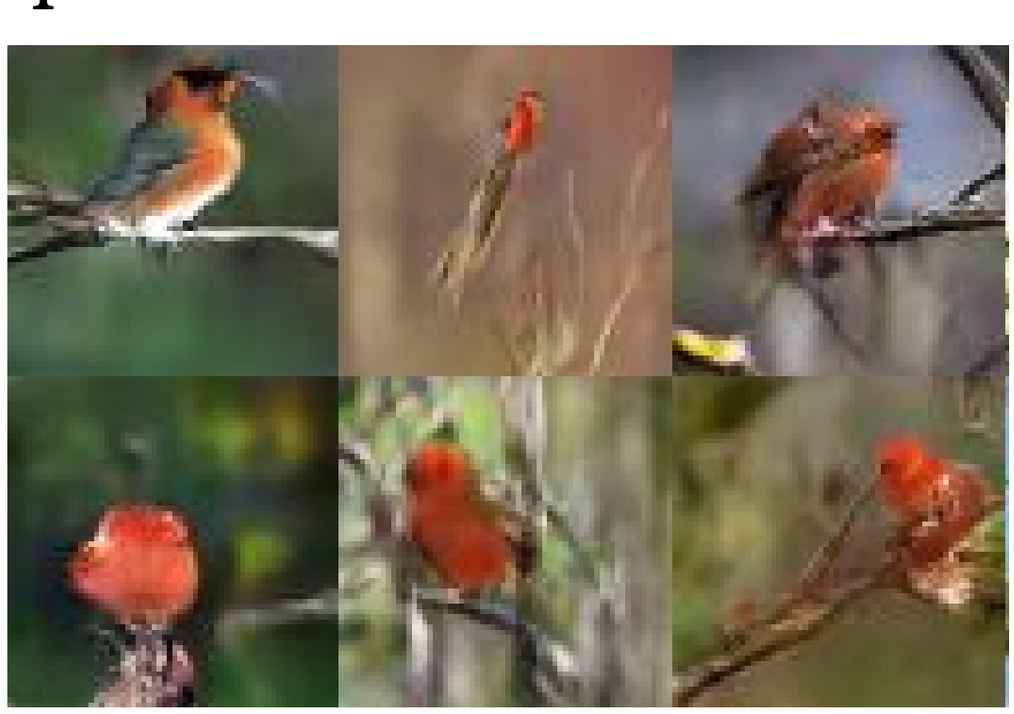
**Top 5 sentences generated using CNNS**

a young woman wearing a pink and standing in front of a metal shoe.
a boy in blue shirt and shorts is holding a sword to pick up two children in a batting position.
a young boy holding something in a batting cage.
a boy standing in a gym with a toy gun.
a little boy is holding a basketball, looking at the ground.

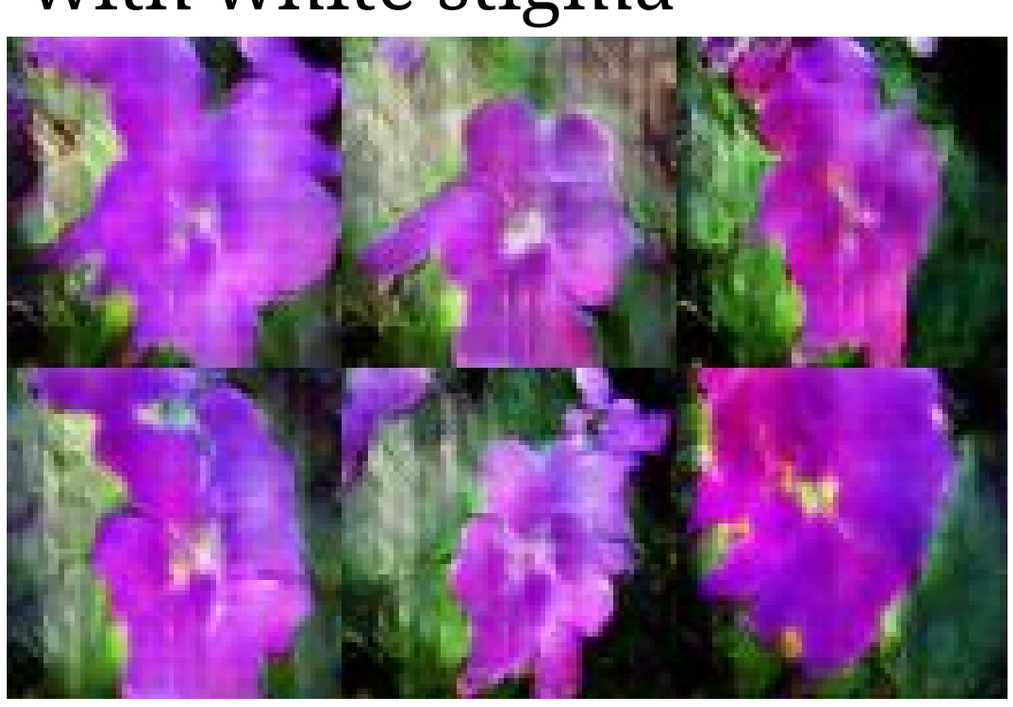
Quattoni et al, 2015

# Example: Image Generation From Text

this small bird has a pink breast and crown, and black primaries and secondaries.

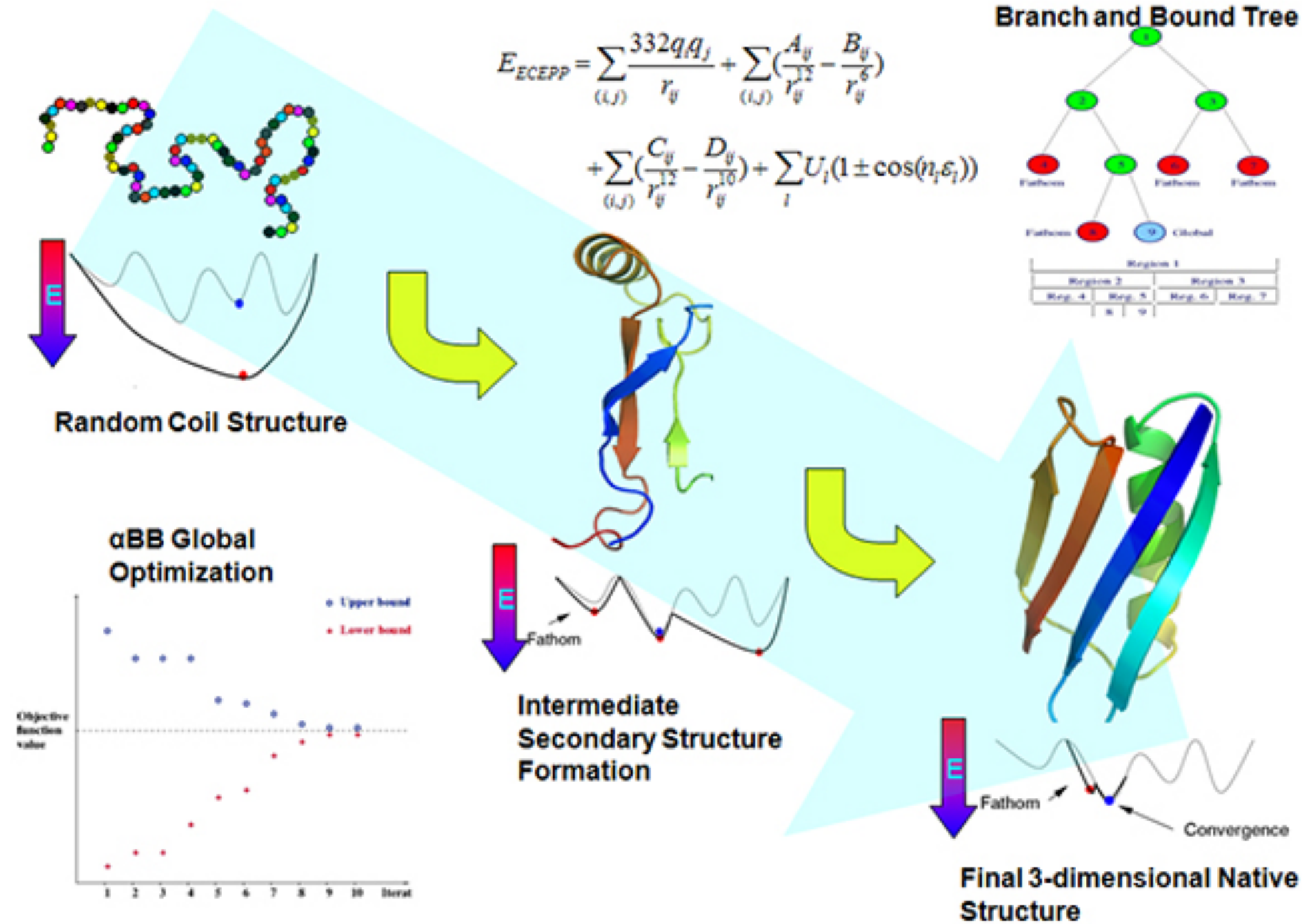


the flower has petals that are bright pinkish purple with white stigma



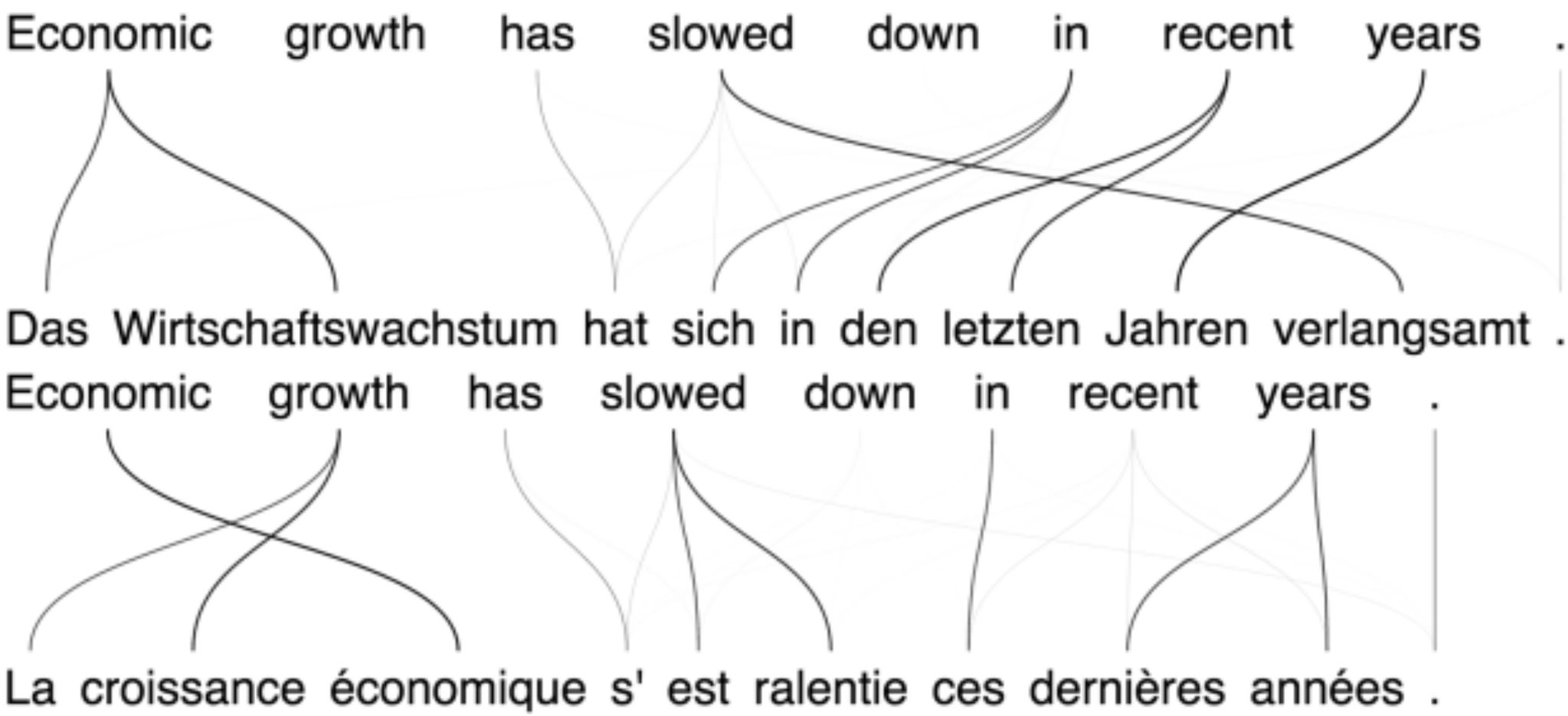
Reed et al, 2016

# Example: Protein Folding

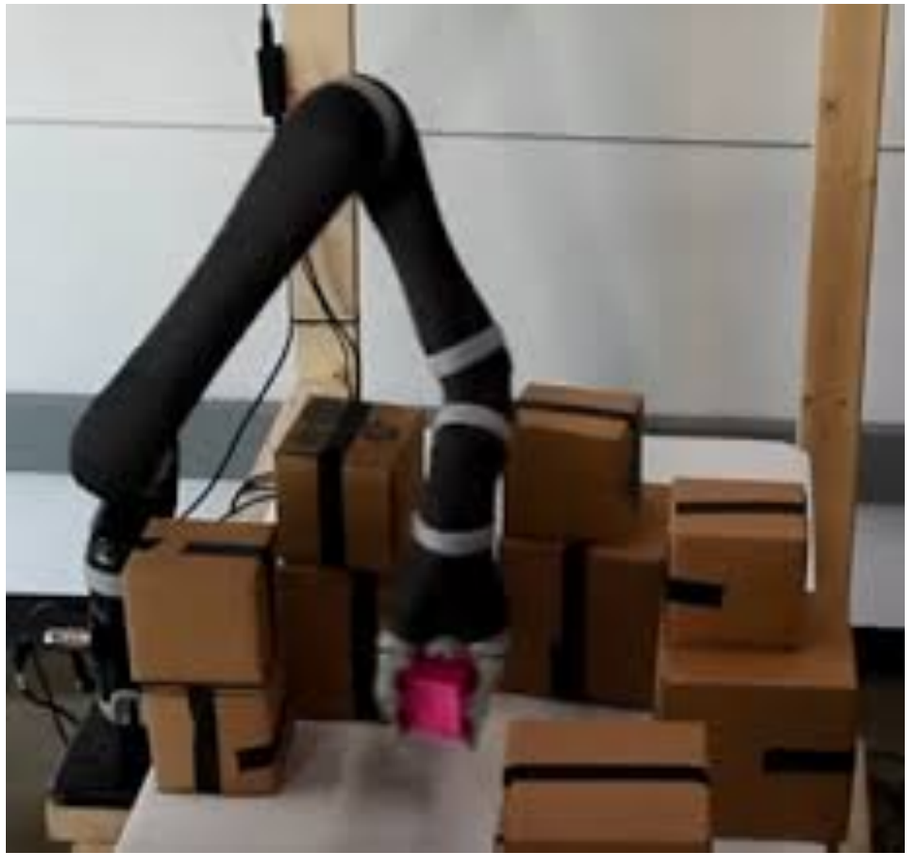



http://titan.princeton.edu/research/highlights/?n=11

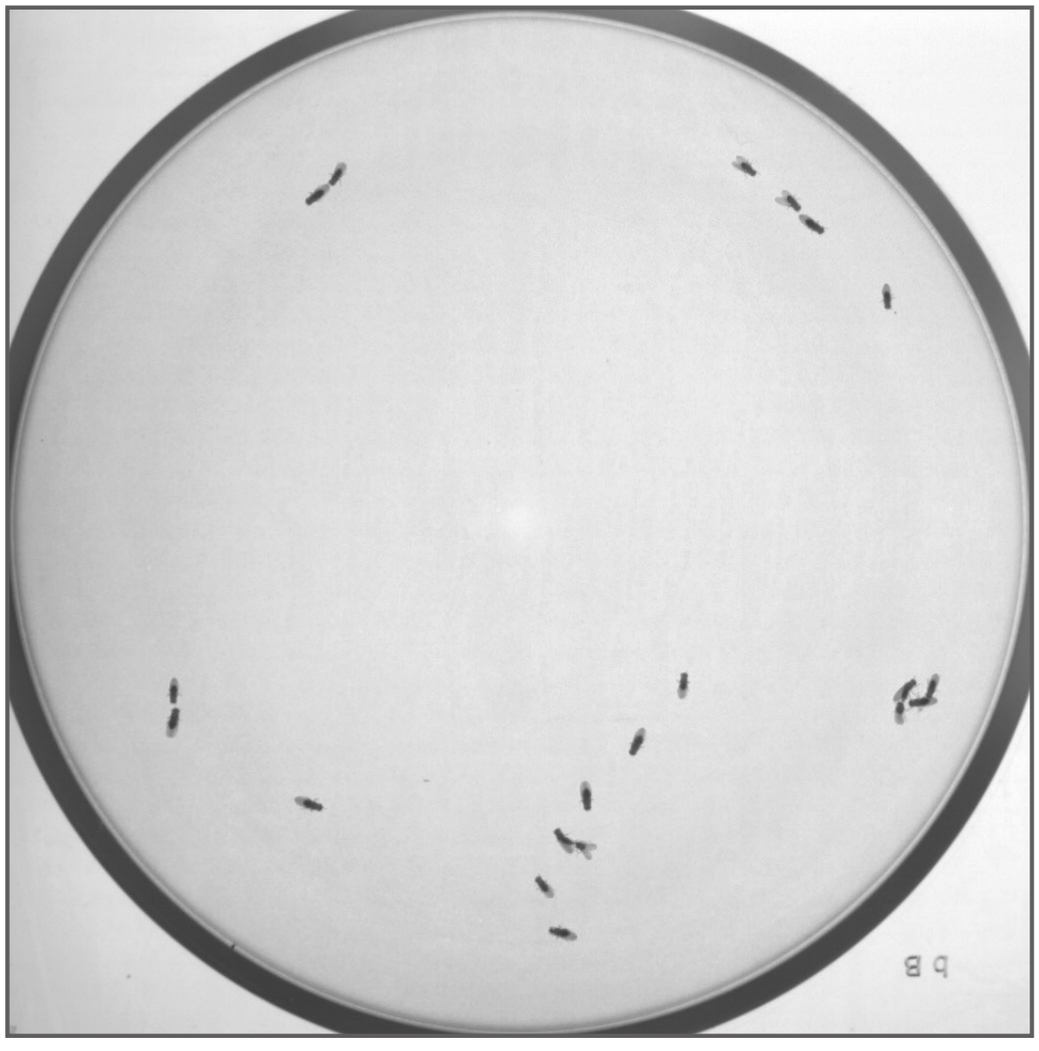
# Example: Machine Translation

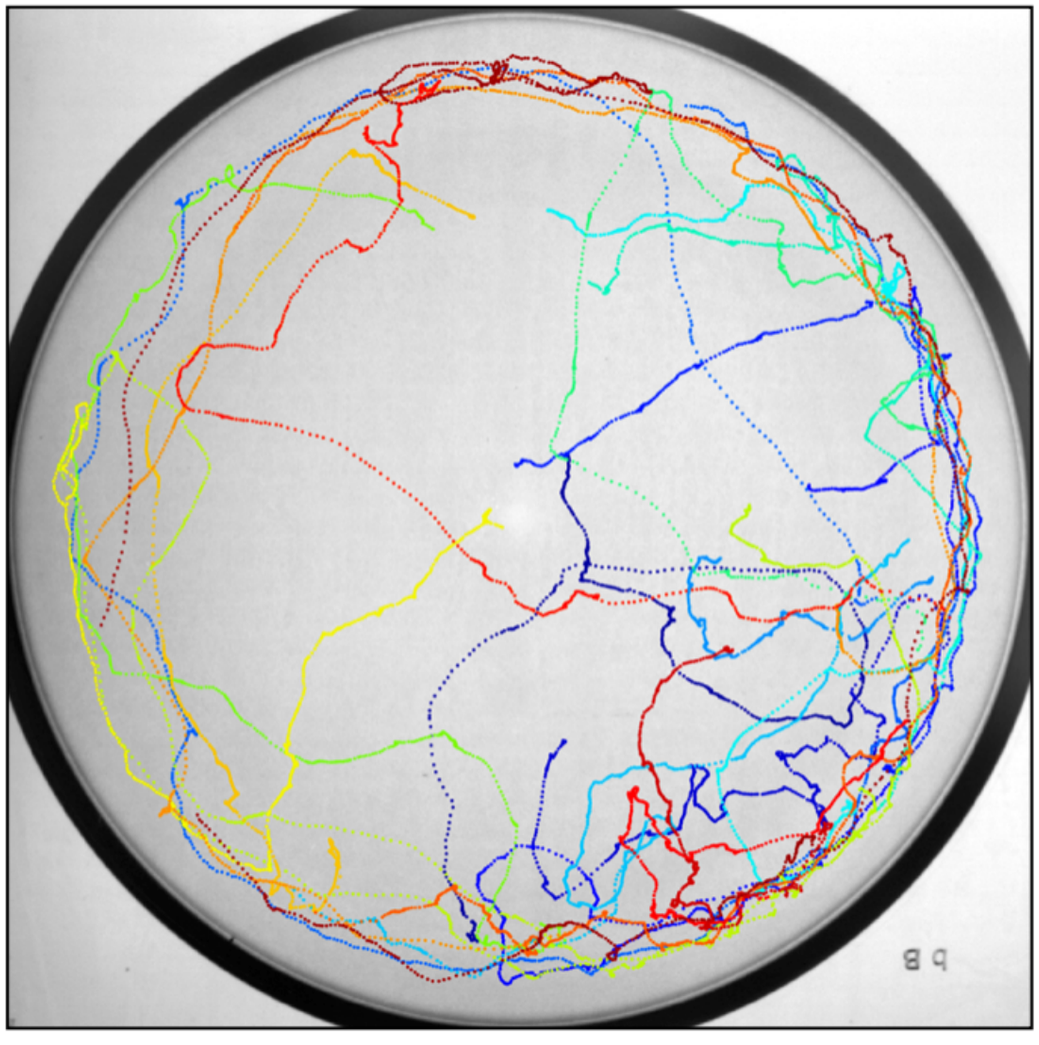


# Example: Robot Planning



# Example: Behavior Modeling





Eyjolfsdottir et al, 2017

# Structured Prediction Problems

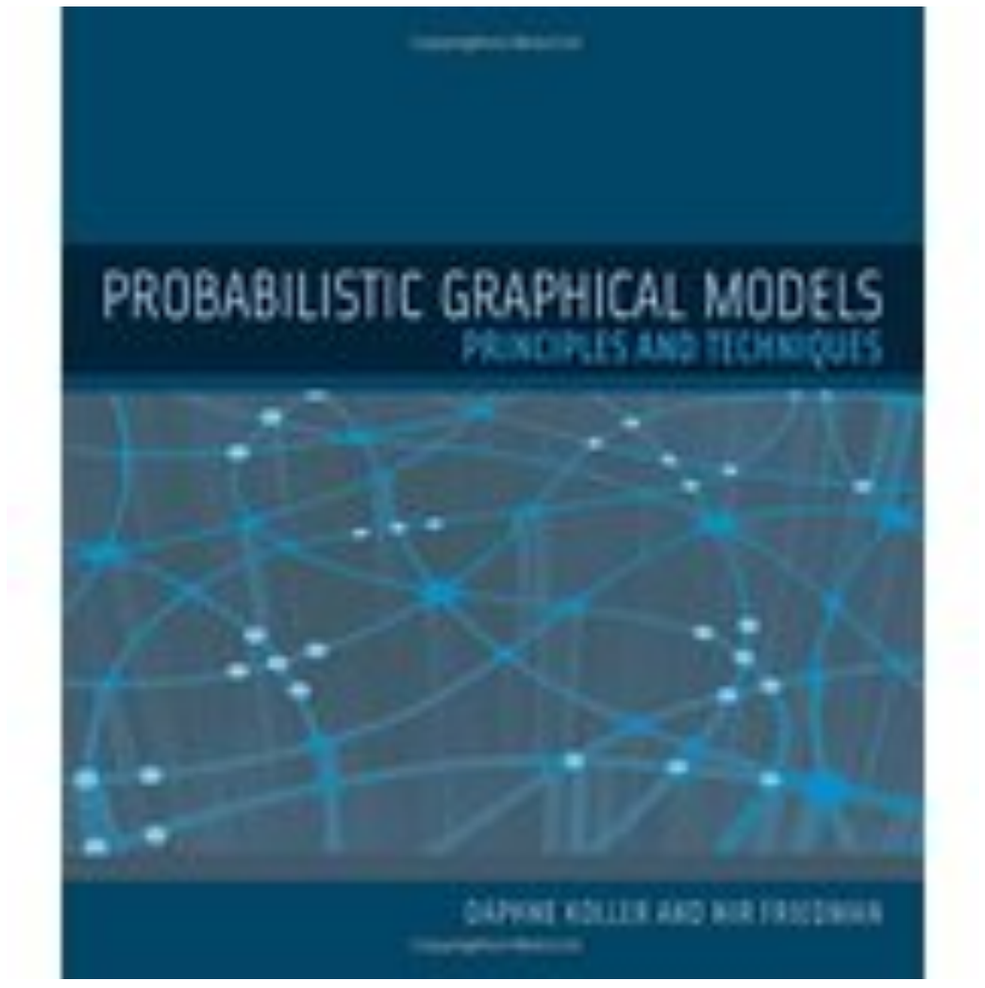
- Inference/Prediction

➡ Given input *X* and learned model, predict output *Y*

- Learning/Training

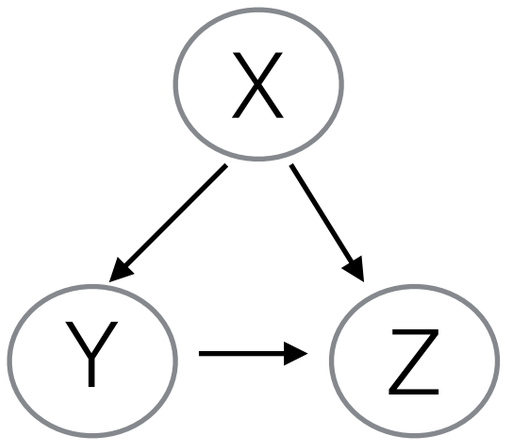
➡ Learn model parameters *w* from training data

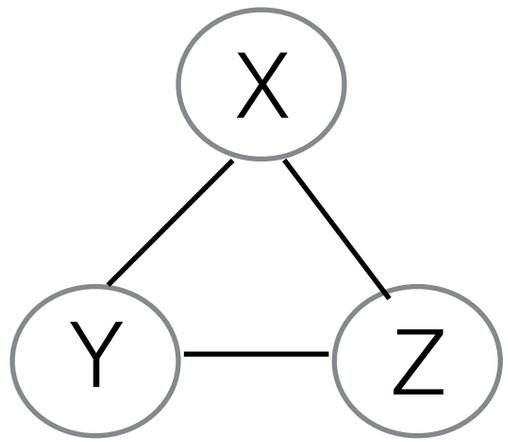
# Probabilistic Graphical Models




Recommended textbook: Probabilistic Graphical Models: Principles and Techniques by Koller and Friedman

# Probabilistic Graphical Models




- Variables X, Y, and Z
- Encode relationships between variables.
- Graph represents a set of independences and factorizes a distribution.

- Directed graphical model and undirected graphical model

# Conditional Independence

- Recall that two events A and B are independent if

$$P(A \cap B) = P(A)P(B), \text{ or equivalently,} P(A|B) = P(A)$$

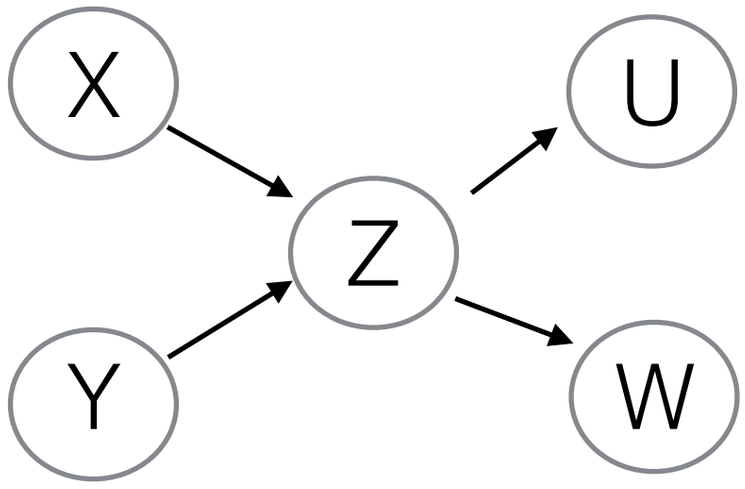
and we denote it as $A \perp B$

- Two events A and B are conditionally independent given an event C if

$$P(A \cap B|C) = P(A|C)P(B|C), \text{ or equivalently,} P(A|B, C) = P(A|C)$$

and we denote it as $A \perp B|C$

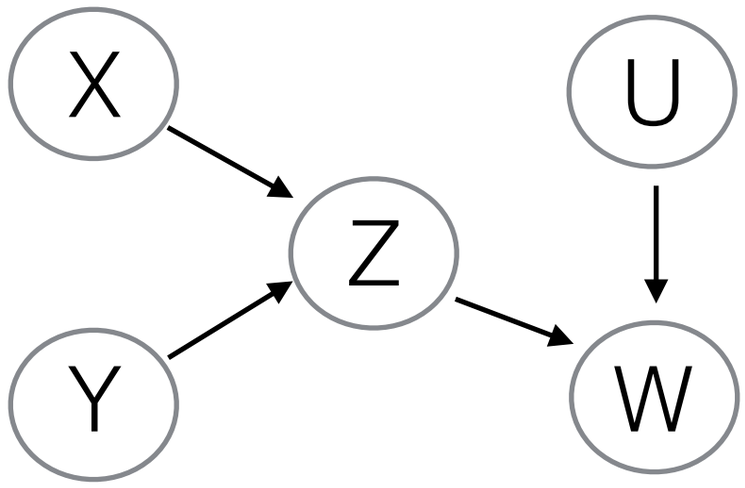
# Directed Graphical Models




- Also called Bayesian Networks.
- Include: Naive Bayes, HMM, …
- Recall $P(x_1, x_2, ..., x_n) = p(x_1)p(x_2|x_1) \cdots p(x_n|x_{n-1}, ..., x_2, x_1).$
- We have $p(x_i|x_{i-1}, ..., x_1) = p(x_i|x_{A_i})$

  where $x_{A_i}$ are variables of parents.
- Formally, a directed graph G=(V,E) with random variables for each node in V and conditional probability distribution per node: *p*(node | its parent(s))

# Directed Graphical Models



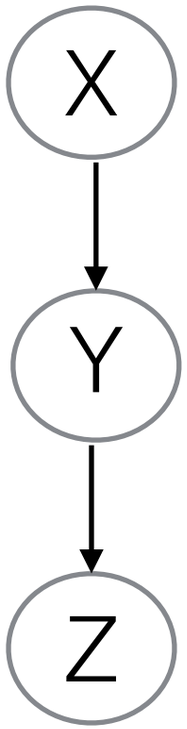

- Variables: U,W,X,Y, and Z

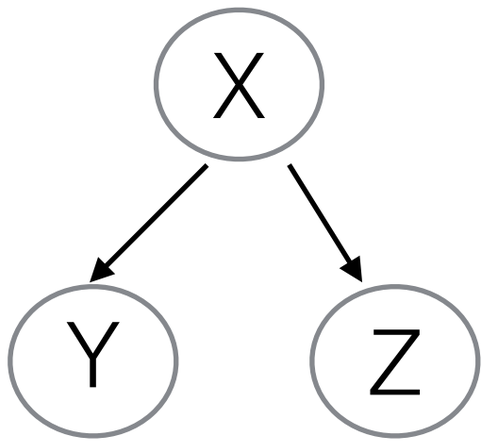
- Example: how to factorize the joint probability distribution according to this graph?

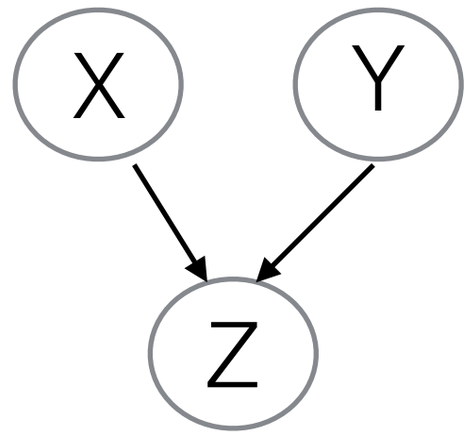
$$P(U, W, X, Y, Z) = p(X)p(Y)p(Z|X, Y)p(U)p(W|Z, U)$$

- Question: which independence assumptions with given structure by G?

# Conditional Independence

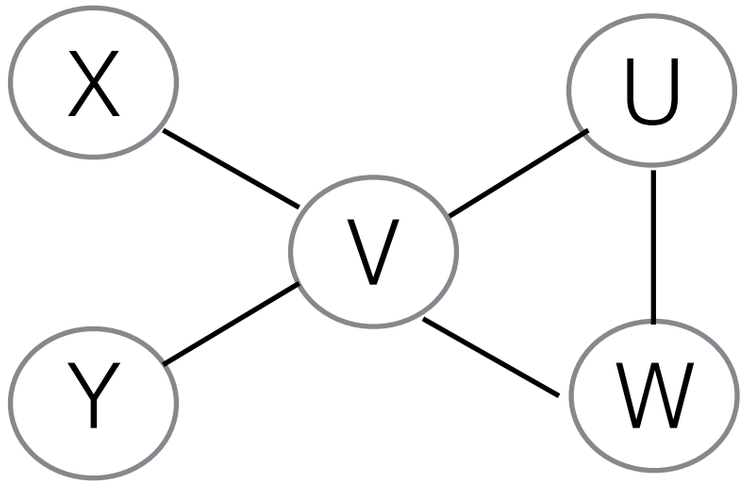



- Assume three variables: A, B, and C
- Three scenarios: 1) cascade $Z \perp X | Y$
  2) common parent $Y \perp Z | X$
  3) V-structure $X \not\perp Y | Z$
- More general graphs: d-separation with *active* paths

# Undirected Graphical Models



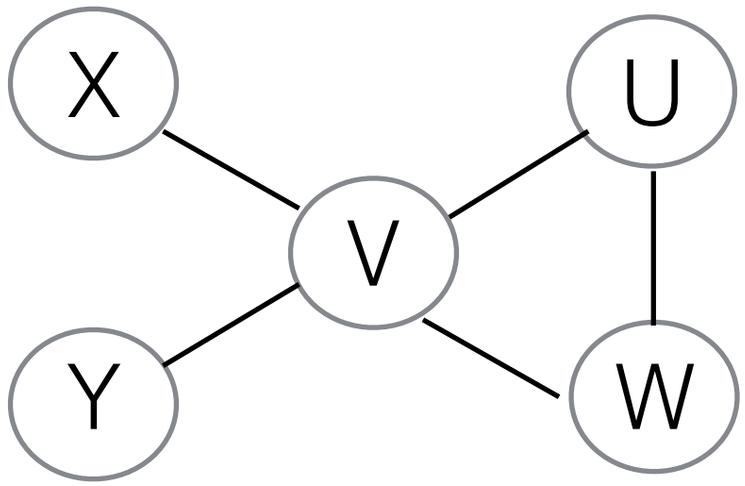

- Some distribution cannot be perfectly represented by Bayesian network.

- Represented by undirected graph and also called Markov Random Fields (MRF).

- Variables: U,W,X,Y, and V

- We define a *factor* (or potential) $\phi(X, V)$ rather than $p(V|X)$.

# Undirected Graphical Models



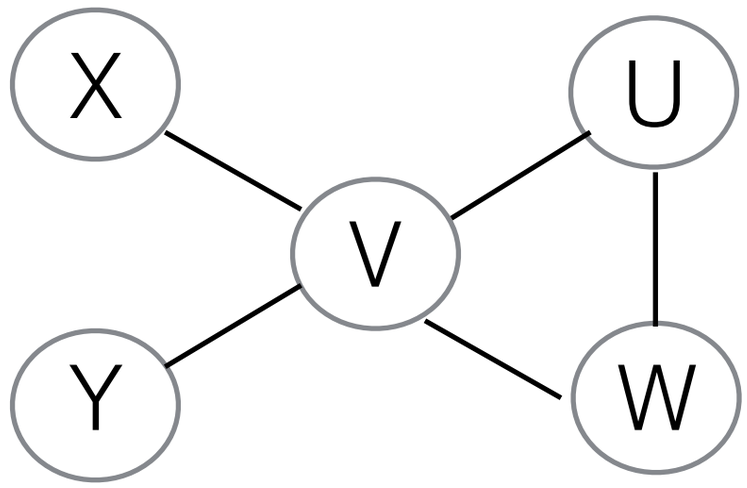

- We define a factor over *cliques* (i.e., fully connected subgraphs)

- A probability is a form of: $\tilde{p}(X, Y, V, U, W) = \phi(X, V)\phi(Y, V)\phi(V, U, W)$

- Normalized probability: $p(X, Y, V, U, W) = \frac{1}{Z}\tilde{p}(X, Y, V, U, W)$

  where $Z = \sum_{X,Y,V,U,W} \tilde{p}(X, Y, V, U, W)$

# Undirected Graphical Models

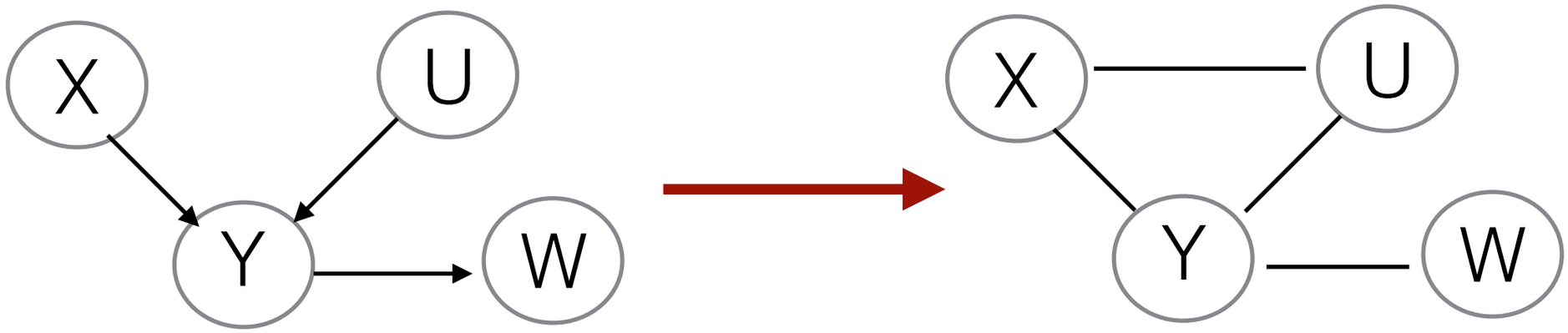



- Formally, a MRF is a probabilistic distribution over variables $x_i$ defined by an *undirected* graph G in which nodes correspond to variables $x_i$.

- It has the form $$p(x_1, ..., x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c)$$

  where $$Z = \sum_{x1,...,x_n} \prod_{c \in C} \phi_c(x_c)$$

# Relationship between Directed and Undirected Graphical Model



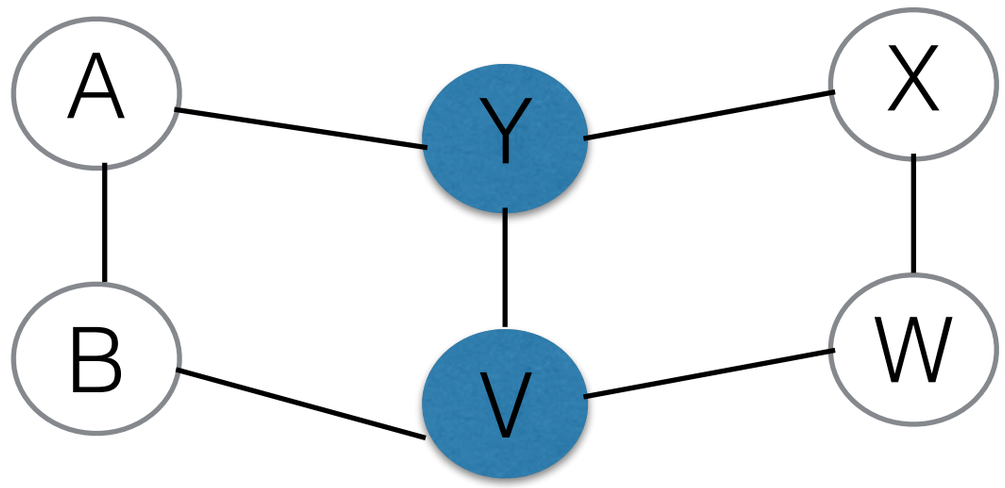

- Bayesian networks are a special case of MRFs.

- We can convert directed graph G into its corresponding undirected graph G' by adding edges to all parents of a given node, and its process is called *moralization*.

- MRF is more powerful than Bayesian networks but are more difficult to deal with computationally.

- Try to use Bayesian networks first and then switch to MRFs if it does not work.

# Conditional Independence




- What independencies are modeled by an undirected graphical model?

- Variables *x,y* are dependent if they are connected by a path of unobserved variables.

- *Markov blanket U* of a variable *X* is the minimal set of nodes such that *X* is independent from the rest of the graph if *U* is observed.

# Inference in Graphical Models

# Inference in Graphical Models

- *Marginal inference*: what is the probability of a given variables after sum everything else out (e.g., probability of spam vs non-spam)?

$$p(y=1) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_n} p(y=1, x_1, x_2, ..., x_n)$$

- *MAP (Maximum A Posteriori) inference*: what is the most likely assignment to the variables, possibly conditioned on evidence (e.g.,predicting characters from handwriting).

$$\max_{x_1,\ldots,x_n} p(y=1, x_1, ..., x_n)$$

- Inference is a challenging task, depending on the structure of the graph, and in many case, NP-hard.

- If tractable, we use exact inference algorithms and otherwise, we use approximate inference algorithms.

# Inference Methods

- Exact inferences

➡ Variable elimination, message passing, junction tree, and graph cuts*

- Approximate inferences

➡ Loopy belief propagation, linear programing relaxation, sampling methods, and variational methods

# Variable Elimination

- We use the *structure* of graph and *dynamic programming* for efficient inference.
- For simplicity, suppose we have a chain Bayesian network.

$$p(x_1, ..., x_n) = p(x_1) \prod_{i=1}^{n} p(x_i|x_{i-1})$$

- We are interested in computing marginal distribution

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1, ..., x_{n-1}, x_n)$$

- Naive approach would take $O(d^n)$.
- Observe that

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i|x_{i-1})$$

$$= \sum_{x_{n-1}} p(x_n|x_{n-1}) \sum_{x_{n-2}} p(x_{n-1}|x_{n-2}) \cdots \sum_{x_1} p(x_2|x_1)p(x_1).$$

# Variable Elimination

- We can start by computing factor $\tau(x_2) = \sum_{x_1} p(x_2|x_1)p(x_1)$ and it takes $O(d^2)$. Then we have

$$p(x_n) = \sum_{x_{n-1}} p(x_n|x_{n-1}) \sum_{x_{n-2}} p(x_{n-1}|x_{n-2}) \cdots \sum_{x_2} p(x_3|x_2)\tau(x_2).$$

- Then we repeat to compute factor $\tau(x_3) = \sum_{x_2} p(x_3|x_2)\tau(x_2)$ until we are only left with $x_n$.

- In total, it takes $O(nd^2)$ instead of $O(d^n)$.

- Variable elimination algorithm mainly performs two operations: *product* and *marginalization*.

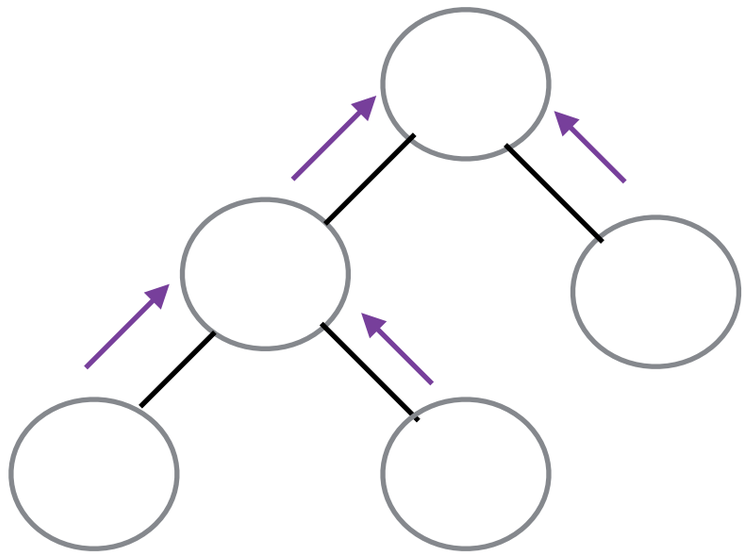
- Similarly, this can be applied to MRFs.

# Variable Elimination

- Let $p(x_1, ..., x_n) = \prod_{c \in C} \phi_c(x_c).$
- The factor product operation: $\phi_3(x_c) = \phi_1(x_c^{(1)}) \times \phi_2(x_c^{(2)}).$

  e.g., $\phi_3(a, b, c) = \phi_1(a, b) \times \phi_2(b, c).$
- The marginalization operation: $\tau(x) = \sum_y \phi(x, y)$
- We can also introduce evidence $P(Y|E = e) = \frac{P(Y, E = e)}{P(E = e)}.$
- It is NP-hard to find the best ordering, while there are some heuristic approaches.

# Belief Propagation



- Now we assume undirected graphical models. We can covert directed graphical models to undirected ones by moralization.

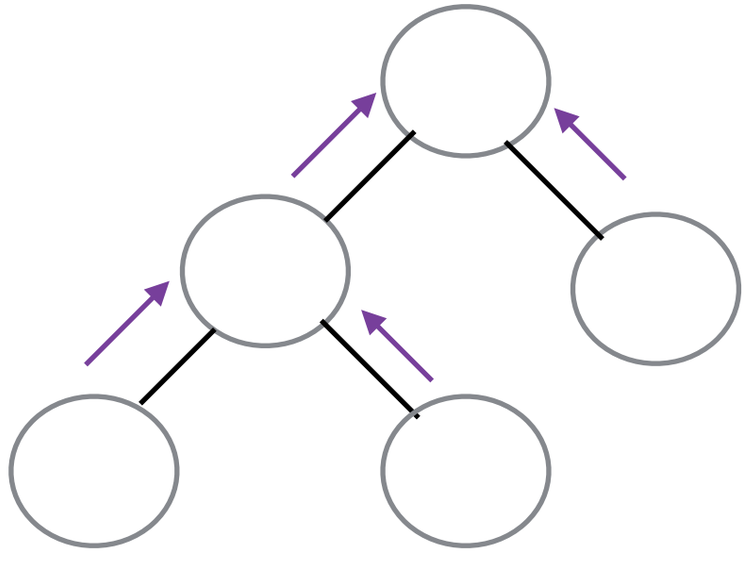
- In variable elimination, we need to recompute for a new query. Why not storing factors?

- First we assume a tree structure for the graph.

- Suppose that we choose $x_k$. We make it as root and pass all information (factors) to the root node to compute marginal probabilities.

- We store all passed messages (factors) and then we can answer queries in $O(1)$.

# Message Passing



- Each step, we compute the factor $\tau(x_k) = \sum_{x_j} \phi(x_k, x_j)\tau_j(x_j)$ where $x_k$ is the parent of $x_j$ in the tree.

- Then, $x_k$ will be eliminated, and $\tau(x_k)$ will be passed up the tree to the parent $x_l$ of $x_k$ in order to be multiplied by $\phi(x_l, x_k)$
- We can think of $\tau(x_k)$ as a message that $x_k$ sends to $x_l$.

- A node $x_i$ sends a message to a neighbor $x_j$ whenever it has received messages from all nodes besides $x_j$.

# Sum-product Message Passing

- When a node $x_i$ is ready to transmit to $x_j$ , send the message

$$m_{i\to j}(x_j) = \sum_{x_i} \phi(x_i)\phi(x_i, x_j) \prod_{l\in N(i)\setminus j} m_{l\to i}(x_i).$$

- After computing all messages, we have

$$p(x_i) = \prod_{l\in N(i)} m_{l\to i}(x_i).$$

# Max-product Message Passing

- MAP inference $\max_{x_1,\ldots,x_n} p(y=1,x_1,\ldots,x_n)$
- By replacing sum with *maxes*, we can decompose MAP inference problem in the same way as marginal inference problem.
- Suppose we compute the partition function of a chain MRF:

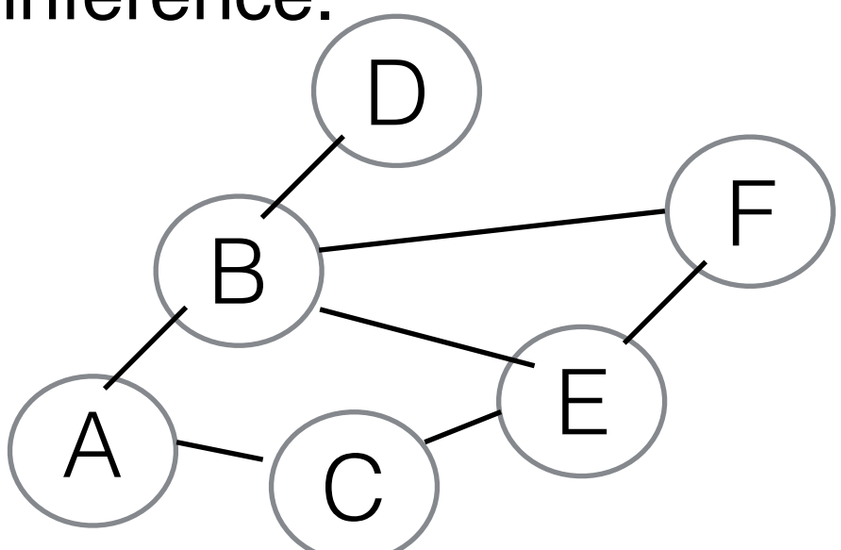
$$
\begin{aligned}
Z &= \sum_{x_1} \cdots \sum_{x_n} \phi(x_i) \prod_{i=2}^{n} \phi(x_i, x_{i-1}) \\
&= \sum_{x_n} \sum_{x_{n-1}} \phi(x_n, x_{n-1}) \sum_{x_{n-2}} \phi(x_{n-1}, x_{n-2}) \cdots \sum_{x_1} \phi(x_2, x_1)\phi(x_1).
\end{aligned}
$$

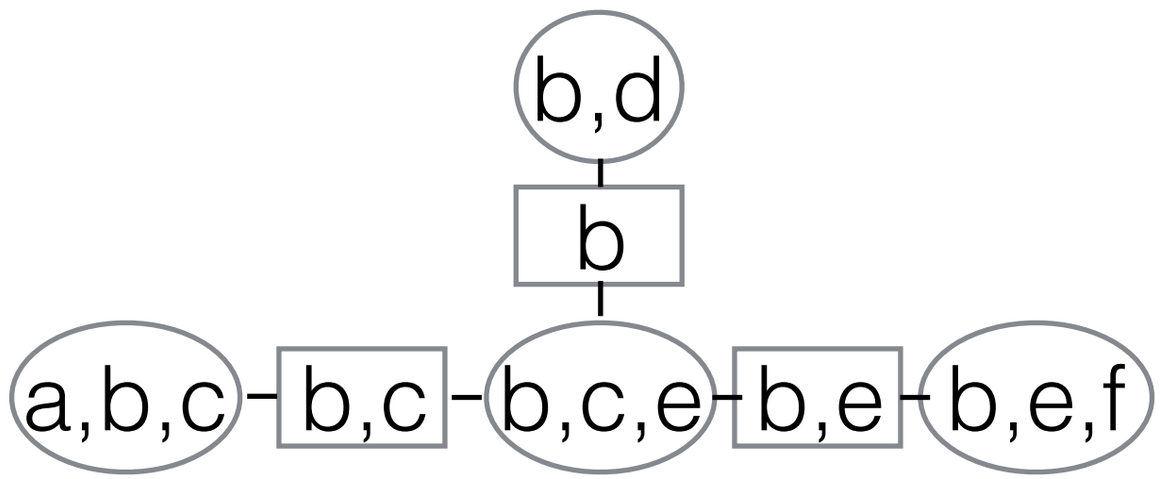
- Then we have:

$$
\begin{aligned}
\tilde{p}^* &= \max_{x_1} \cdots \max_{x_n} \phi(x_i) \prod_{i=2}^{n} \phi(x_i, x_{i-1}) \\
&= \max_{x_n} \max_{x_{n-1}} \phi(x_n, x_{n-1}) \max_{x_{n-2}} \phi(x_{n-1}, x_{n-2}) \cdots \max_{x_1} \phi(x_2, x_1)\phi(x_1).
\end{aligned}
$$

# Junction Tree Algorithm

- So far, we have focused on a tree. What if this is not the case?
- Inference is no longer tractable.
- Junction tree algorithm tries to partition the graph into clusters of variables and convert to a tree of clusters, and then run message passing on this graph.
- If we can *locally* solve for each cluster, we can do exact inference.



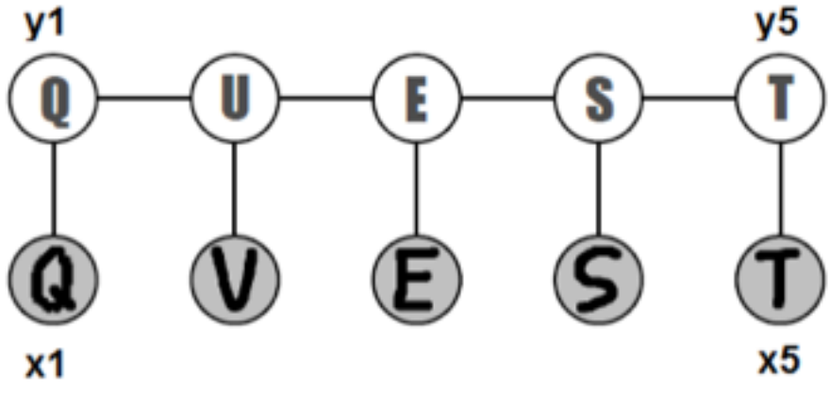
# Loopy Belief Propagation

- In many case, finding a good junction tree is difficult.
- We may satisfy with a quick *approximate* solution instead.
- The main idea of *loopy belief propagation* is disregarding loops in the graph and performing message passing anyway.
- At each time t, we perform update:

$$m_{i\to j}^{t+1}(x_j) = \sum_{x_i} \phi(x_i)\phi(x_i, x_j) \prod_{l\in N(i)\setminus j} m_{l\to i}^{t}(x_i).$$

- We keep updating for a fixed number of steps or until convergence.
- This heuristic approach often works well in practice.

# MAP Inference





Everingham et al, 2010

- MAP inference: e.g., handwriting recognition, image segmentation

$$\max_x \log p(x) = \max_x \sum_c \phi_c(x_c) - \log Z$$

- We can ignore $\log Z$ and thus it becomes:

$$\max_x \log p(x) = \max_x \sum_c \phi_c(x_c)$$

- Generally, MAP inference is easier than marginal distribution.

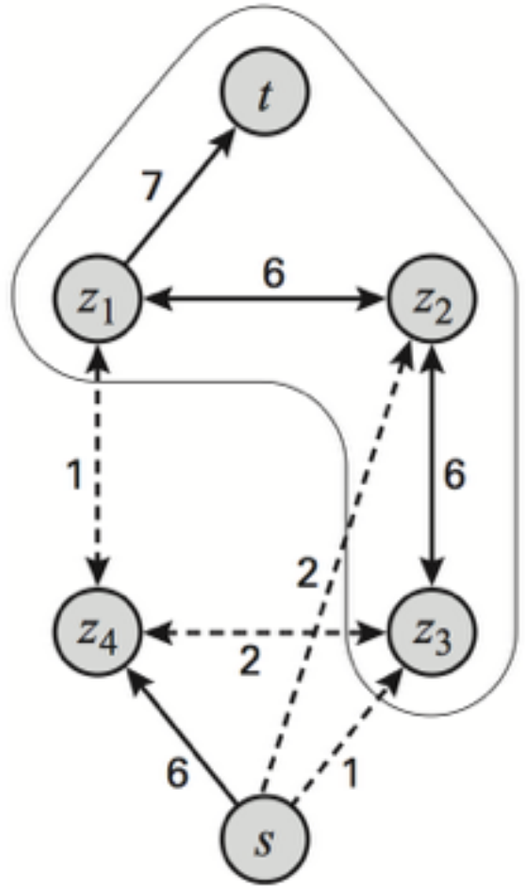
- For some cases we can solve MAP inference in polynomial time, while general inference is NP-hard.

# Graph Cuts

- Efficient exact MAP inference (for certain conditions)
- Use image segmentation as a motivating example.
- Suppose a MRF with binary potentials (or edge energies) of the form:

$$\phi(x_i, x_j) = 0 \quad \text{if} \quad x_i = x_j$$
$$\lambda \quad \text{if} \quad x_i \neq x_j$$

- We look for an assignment to minimize the energy.
- We can solve this as min-cut problem in an augmented graph G'.
- The cost of a min-cut equals the minimum energy in the model.




(from Stefano Ermon's class materials)

# Linear Programming

- Graph cuts are only applicable in certain restricted classes of MRFs.

- Linear programming

$$\begin{aligned} &\min c \cdot x \\ &\text{s.t} \quad Ax \leq b \end{aligned}$$

  where $x \in \mathbb{R}^n, c, b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$

- We can reduce MAP objective to Integer Linear Programming (ILP) by introducing indicator variables for each node and state, and each edge and pair of states

- Then objective becomes:

$$\max_{\mu} \sum_{i \in V} \sum_{x_i} \theta_i(x_i)\mu_i(x_i) + \sum_{i,j \in E} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j)\mu_{ij}(x_i, x_j)$$

- We also add consistency constraints: each indicator variables should be binary and some of them with states should be 1.

- ILP is still NP-hard (and approximate inference), but we can solve it with LP-relaxation (and round them to recover binary values). In practice it works well.

# Other MAP Inference Approaches

- Local search
- Branch and bound
- Simulated annealing

# Sampling Methods

- In practice, many interesting classes of models do not have exact polynomial-time solutions.

- We can use sampling methods to approximate expectations of functions: $\mathbb{E}_{x\sim p}[f(x)] = \sum_x f(x)p(x).$

- We draw samples $x^1, ..., x^T$ according to *p* and approximate a target expectation with: $\mathbb{E}_{x\sim p}[f(x)] \approx \frac{1}{T}\sum_{t=1}^{T} f(x^t)$

and this is called *Monte Carlo* approximation.

- Special cases: rejection sampling, importance sampling

# Markov Chain Monte Carlo

- *Markov chain* is a sequence of random variables

$$S_0, S1, ... \text{ with } S_i \in \{1, 2, ..., d\}, S_i \sim P(S_i|S_{i-1})$$

- Let *T* denote a matrix with $T_{ij} = P(S_t = i|S_{t-1} = j).$

- After t steps with initial vector probability $p_0$, $p_t = T^t p_0$

- If $\pi = \lim_{t \to \infty} p_t$ exists, we call it a *stationary distribution* of the Markov chain.

- We use Markov chain over assignments to a probability function *p;* by running the chain, we will thus sample from *p.*

- We may use generated samples to compute marginal probabilities. Also we may take samples with the highest probabilities and use it as an estimate of the mode (i.e., MAP inference).

# Metropolis-Hastings Algorithm

- To construct Markov chain within MCMC, we can use Metropolis-Hastings (MH) algorithm.

- Given (unnormalized) target distribution $\tilde{p}(x)$ and proposal distribution $q(x'|x)$

- At each step of the Markov chain, we choose a new point x' according to *q*. Then we accept this proposed change with probability $\min(1, \frac{\tilde{p}(x')q(x^{t-1}|x')}{\tilde{p}(x^{t-1})q(x'|x^{t-1})})$

- *q* can be chosen as something simple, like uniform or Gaussian

- Given any *q*, MH algorithm will ensure that $\tilde{p}$ will be a stationary distribution of the resulting Markov chain.

# Gibbs Sampling

- A widely-used special case of the Metropolis-Hastings methods is Gibbs sampling.

- Given $x_1, ..., x_n$ and starting configuration $x^0 = (x_1^0, ..., x_n^0)$, at each time step t,

$$\text{Sample} \quad x_i' \sim \tilde{p}(x_i | x_{-i}^t) \text{ for } 1 \leq i \leq n$$

$$\text{Set} \quad x^{t+1} = (x_1', ..., x_i', x_n').$$

- It can be considered as $q(x_i', x_{-i} | x_i, x_{-i}) = \tilde{p}(x_i' | x_{-i})$ and always accepting the proposal.

- MCMC might requires long time to converge.

# Variational Inference

- Main idea is casting inference as an optimization problem.

- Suppose we are given an intractable probability distribution *p*. Variational techniques try to solve an optimization problem over a class of tractable distributions $Q$ in order to find a $q \in Q$ that is most similar to *p.*

- We then query *q* (rather than *p*) in order to get an approximate solution.

- Variational approaches will almost *never* find the globally optimal solution. But we will always know if they have converged and, even in some cases, we have bounds on their accuracy.

- Also variational inference methods often scale better.

# Kullback-Leibler (KL) Divergence

- We need to choose approximating family *Q* and objective *J(q)*; latter needs similarity between q and p and we use *Kullback-Leibler (KL) divergence*:

$$KL(q||p) = \sum_x q(x) \log \frac{q(x)}{p(x)}$$

$$KL(q||p) \geq 0 \text{ for all } q, p$$

$$KL(q||p) = 0 \text{ if and only if } q = p$$

$$KL(q||p) \neq KL(p||q)$$

# The Variational Lower Bound

- Let $p(x_1, ..., x_n; \theta) = \frac{\tilde{p}(x_1, ..., x_n; \theta)}{Z(\theta)} = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$
- Optimizing $KL(q||p)$ directly is not possible due to normalization constant $Z(\theta)$.
- Instead, we use this objective: $J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$.
- Note that
$$\begin{aligned} J(q) &= \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} \\ &= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta) \\ &= KL(q||p) - \log Z(\theta) \end{aligned}$$
- Then, $\log Z(\theta) = KL(q||p) - J(q) \geq -J(q)$
- $-J(q)$ is called the variational lower bound or the evidence lower bound (ELBO).

# Mean-field Inference

- How do we choose approximating family *Q*?

e.g., exponential families, neural networks, …

- We consider the set of fully-factored: $q(x) = q_1(x_1)q_2(x_2)\cdots q_n(x_n)$

- Then we solve $\min_{q_1,\cdots,q_n} J(q)$

  by coordinate descent over $q_j$

# Next Lecture

- Learning in Structured Prediction.