Undirected Graphical Models

Prof. Nicholas Zabaras
Center for Informatics and Computational Science
https://cics.nd.edu/
University of Notre Dame
Notre Dame, IN, USA

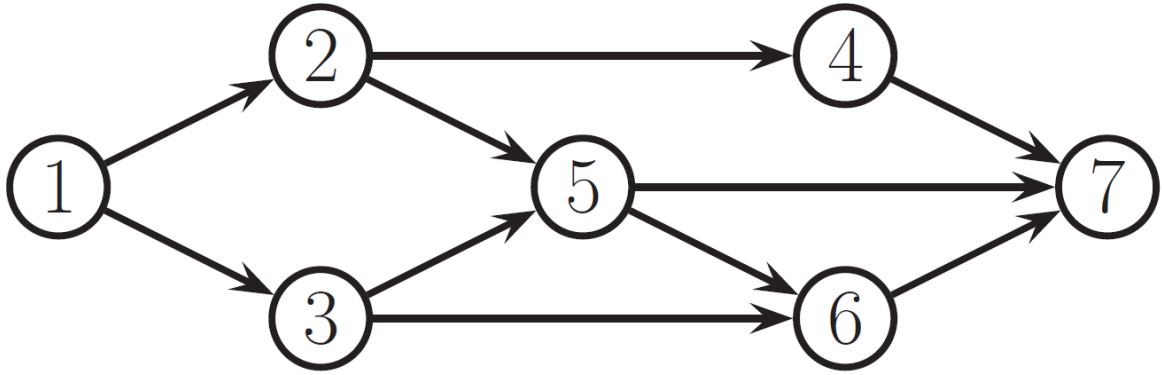
Email: nzabaras@gmail.com
URL: https://www.zabaras.com/

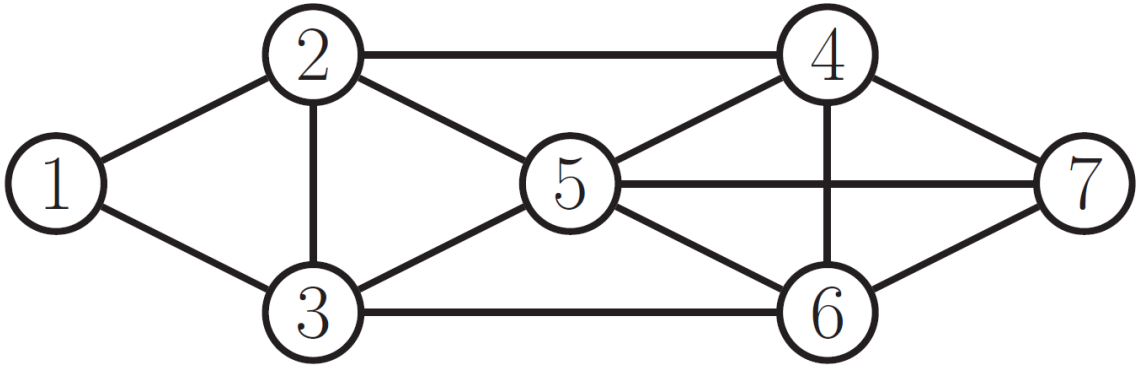
April 9, 2018

Contents

- Kevin Murphy, Machine Learning: A probabilistic Perspective, Chapter 19
- D. Koller and N. Friedman, Probabilistic Graphical Models Principles and Techniques, Chapters 4 and 7.
- Chris Bishop, Pattern Recognition and Machine Learning, Chapter 8

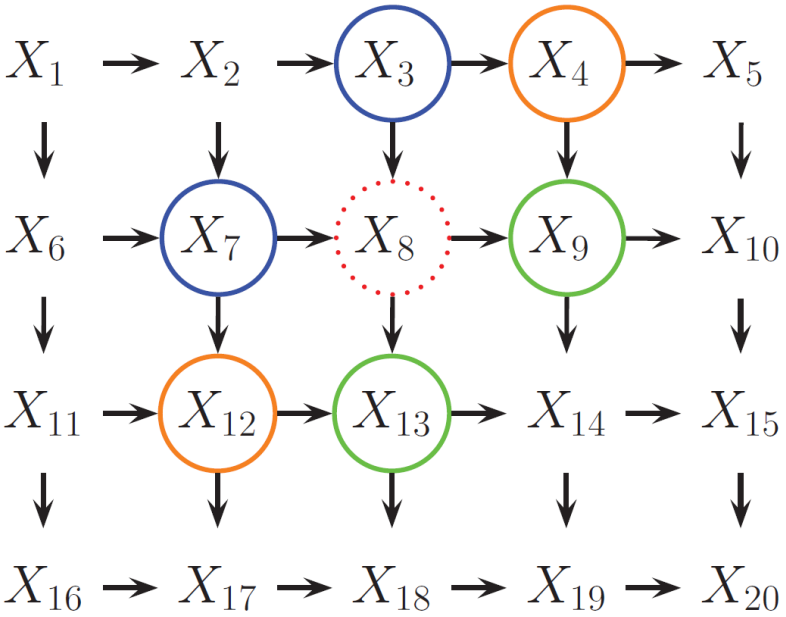
Introduction

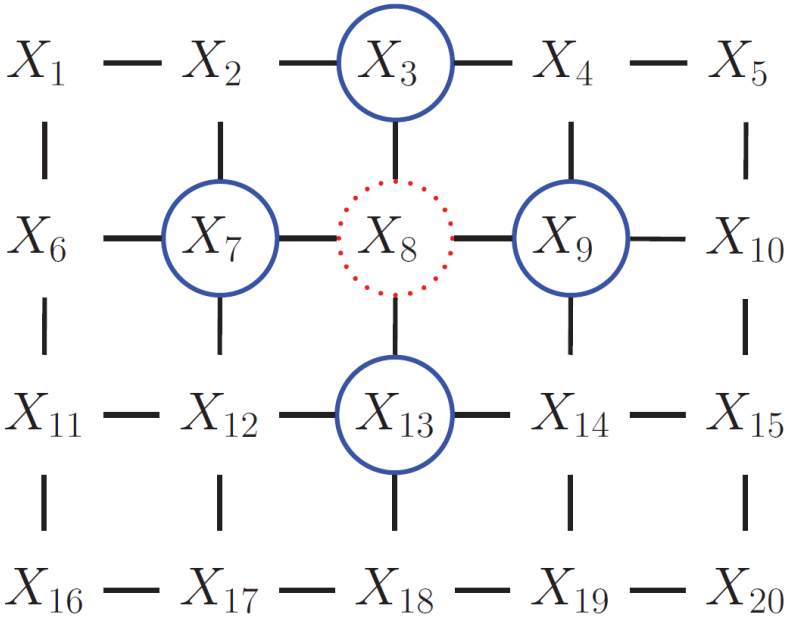




Why UGM?

- For some domains, being forced to choose a direction for the edges, as required by a DGM, is rather awkward.
  - image analysis
  - spatial statistics





- Abend, K., T. J. Harley, and L. N. Kanal (1965). Classification of Binary Random Patterns. *IEEE Transactions on Information Theory 11(4)*, 538–544.

Advantages and Disadvantages of UGM

Advantages:

- They are symmetric and therefore more "natural" for certain domains, such as spatial or relational data.
- Discriminative UGMs (aka conditional random fields, or CRFs), which define conditional densities of the form $p(\boldsymbol{y}|\boldsymbol{x})$, work better than discriminative DGMs.

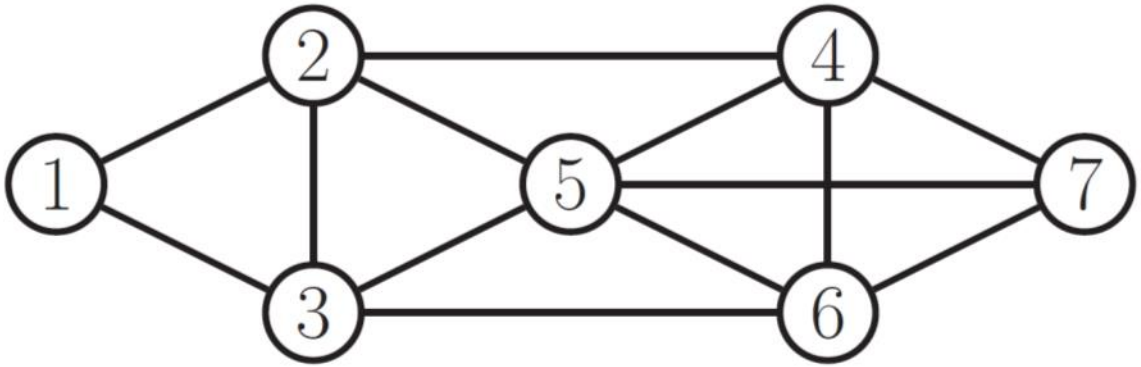
Disadvantages:

- The parameters are less interpretable and less modular.
- Parameter estimation is computationally more expensive

- Domke, J., A. Karapurkar, and Y. Aloimonos (2008). Who killed the directed model? In *CVPR*.

Conditional Independence Properties of UGMs

- UGMs define CI relationships via simple graph separation.
- For sets of nodes A, B and C, we say $X_A \perp_G X_B | X_C$, if C separates A from B in the graph G.
- This means that, when we remove all the nodes in C, if there are no paths connecting any node in A to any node in B, then the conditional Independence property holds.
- This is called the **global Markov property** for UGMs

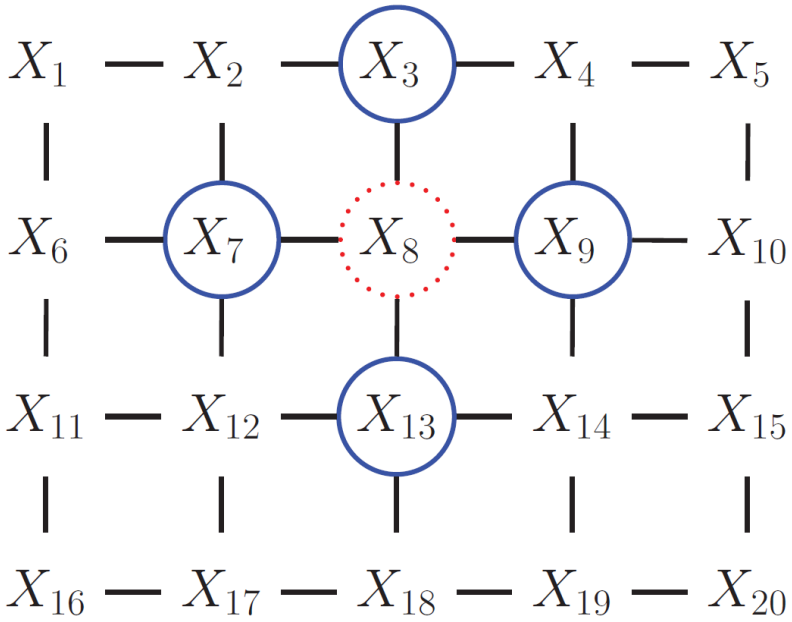


$1,2 \perp 6,7 | 3,4,5$

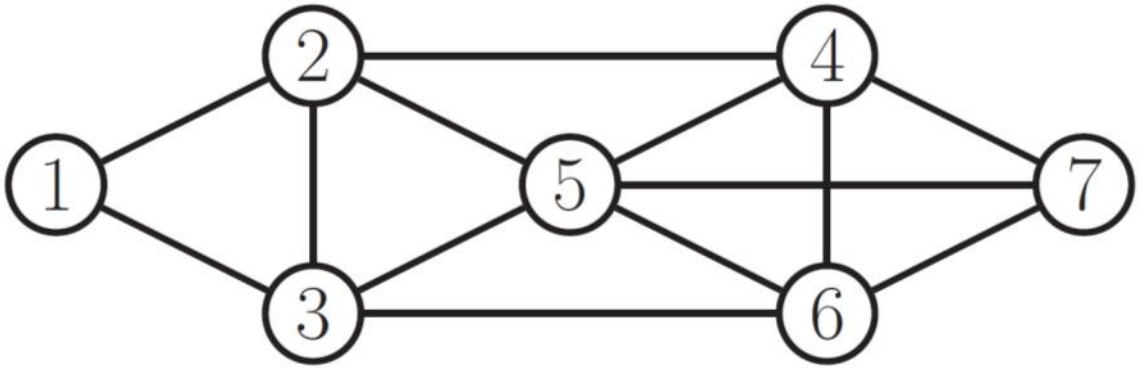
Local Markov Property

- The set of nodes that renders a node t conditionally independent of all the other nodes in the graph is called t's **Markov blanket.**

- One can show that, in a UGM, a node's Markov blanket is its set of immediate neighbors. This is called the **undirected local Markov property**.

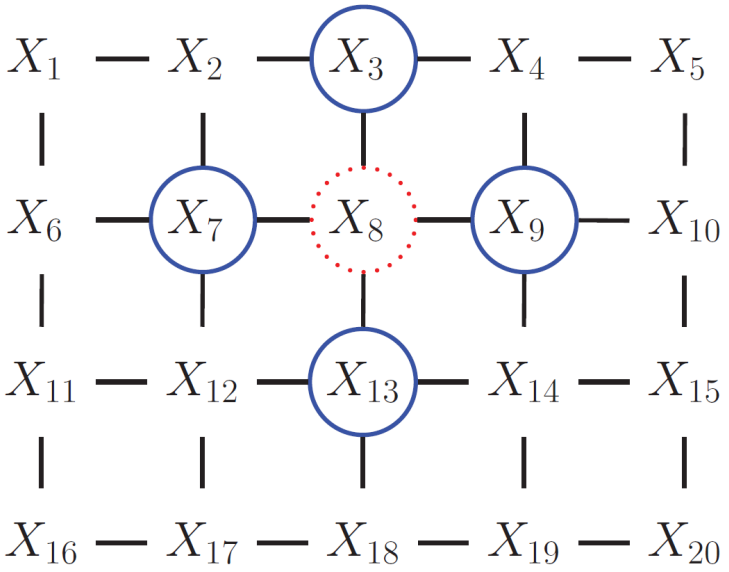


Local Markov Property



$1 \perp rest | 2,3$

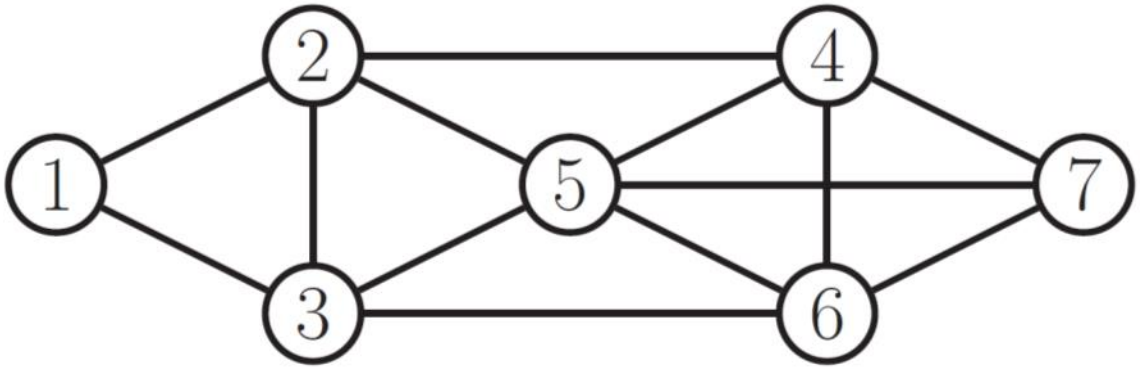
Pairwise Markov Property



- From the local Markov property, we can easily see that two nodes are **conditionally independent given the rest** if there is no direct edge between them. This is called the **pairwise Markov property**

- Global Markov implies local Markov which implies pairwise Markov and vice versa.

Pairwise Markov Property



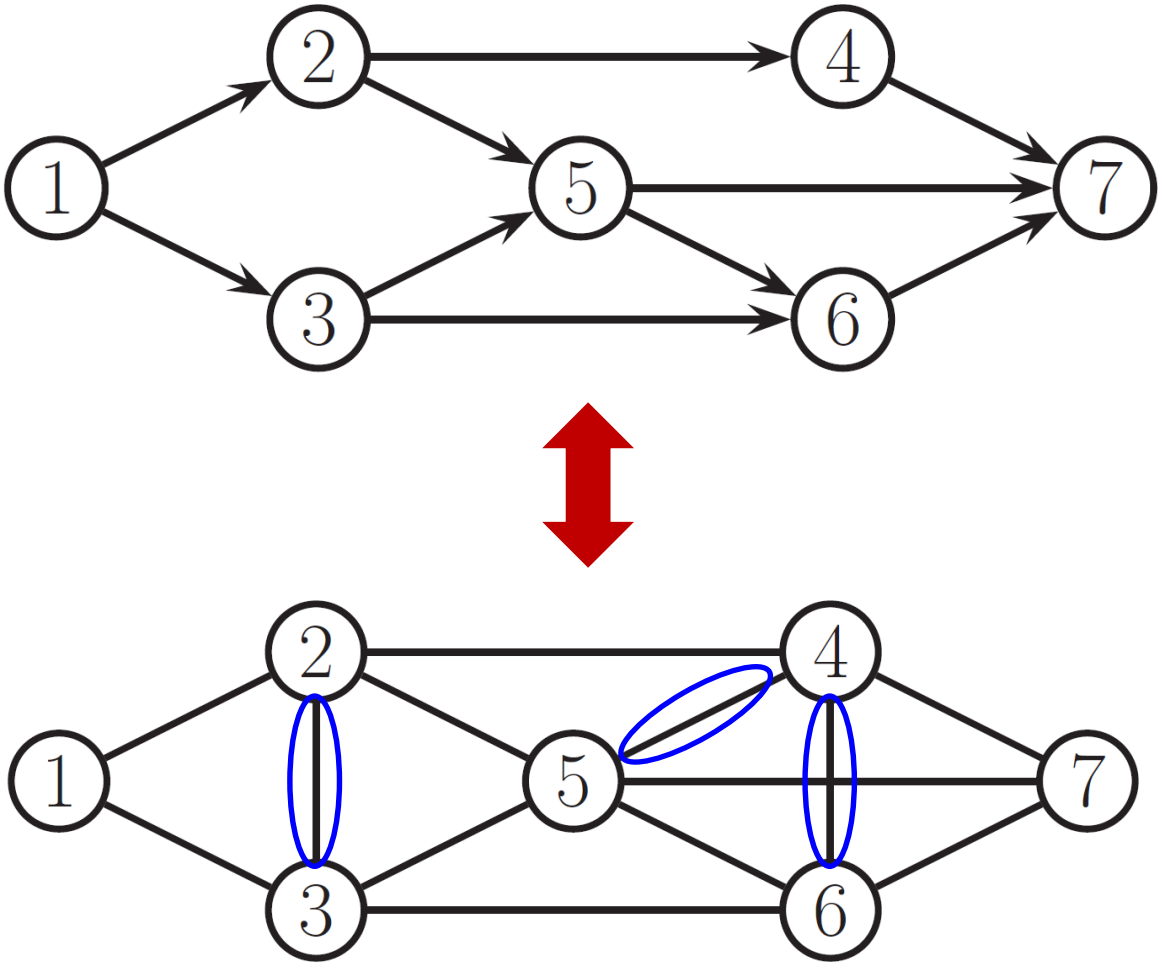
$$1 \perp 7 | rest$$

An Undirected Alternative to d-Separation

- We have seen that determining CI relationships in UGMs is much easier than in DGMs.
- We show how to determine CI relationships for a DGM using a UGM.
- It is tempting to simply convert the DGM to a UGM by dropping the orientation of the edges, but this is clearly incorrect
- $A \rightarrow B \leftarrow C$ is quite different from $A - B - C$.
- $A - B - C$ indicates $A \perp C | B$, which clearly is not true for $A \rightarrow B \leftarrow C$.
- To avoid such incorrect CI statements, we add edges between the "unmarried" parents *A* and *C*, and then drop the arrows from the edges

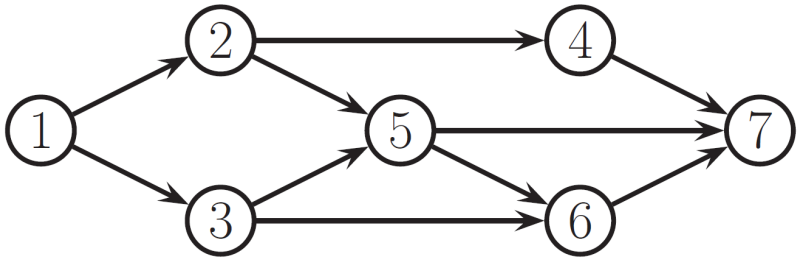
An Undirected Alternative to d-Separation

- We add edges between the "unmarried" parents *A* and *C*, and then drop the arrows from the edges. This is known as **moralization**.

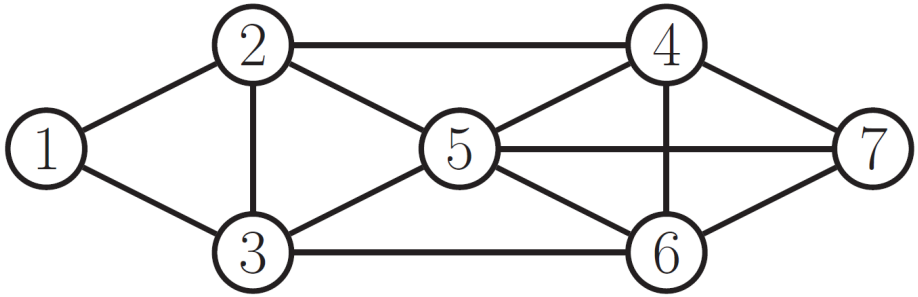


Ancestral Graph

- Moralization loses some CI information, and therefore we cannot use the moralized UGM to determine CI properties of the DGM.



$4 \perp 5|2$

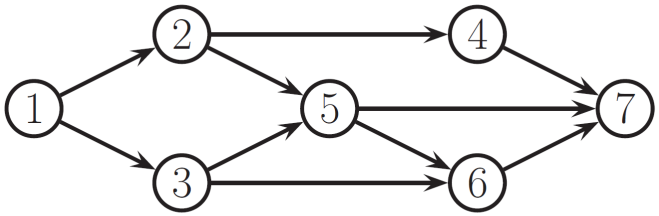


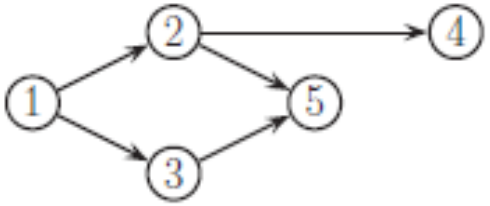
$4 \not\perp 5|2$

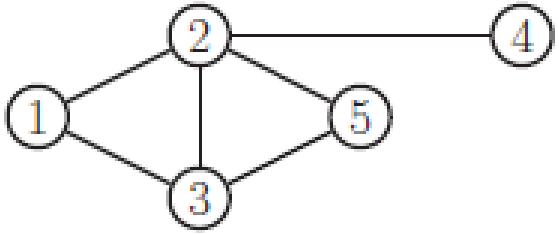
- However, notice that the 4-5 moralization edge, due to the common child 7, is not needed if we do not observe 7 or any of its descendants.

Ancestral Graph

- For determining if A ⊥ B|C, form the **ancestral graph** of DAG *G* with respect to U = A ∪ B ∪ C.
- This means we remove all nodes from *G* that are not in *U* or are not ancestors of U.
- We then moralize this ancestral graph, and apply the simple graph separation rules for UGMs.







- Consider *U* = *{*2, 4, 5*}*. It is now clear that we correctly conclude that 4 ⊥ 5|2.

DGM vs UGM

- We say that G is an I-map of a distribution p if

$$I(G) \subseteq I(p)$$

- Now define G to be perfect map of p if

$$I(G) = I(p)$$

- DGMs and UGMs are perfect maps for different sets of distributions
- In this sense, neither is more powerful than the other as a representation language

DGM vs UGM

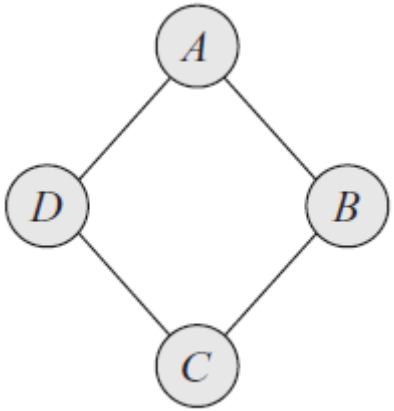
- As an example of some CI relationships that can be perfectly modeled by a DGM but not a UGM, consider a v-structure $A \rightarrow B \leftarrow C$.
- This asserts that $A \perp C$ and $A \not\perp C|B$.
- No UGM can represent both the independence stated above.
- In general, CI properties in UGMs are monotonic in the sense:
  - If $A \perp B|C$, then $A \perp B|(C \cup D)$
- DGMs are non-monotonic (conditioning on extra variables can eliminate independencies due to explaining away)

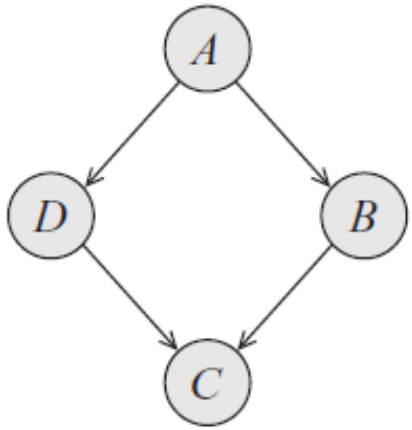
DGM Vs. UGM

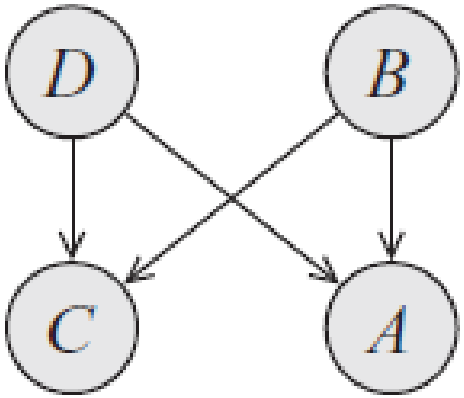
- Example of CI relationships that can be perfectly modeled by a UGM but not a DGM.

- There is no DGM that can precisely represent all and only the CI statements encoded by this UGM.

Incorrect Statements $\Rightarrow$ $B \perp D|A$ $A \perp C|B,D$ $B \perp D$ $A \perp C|B,D$



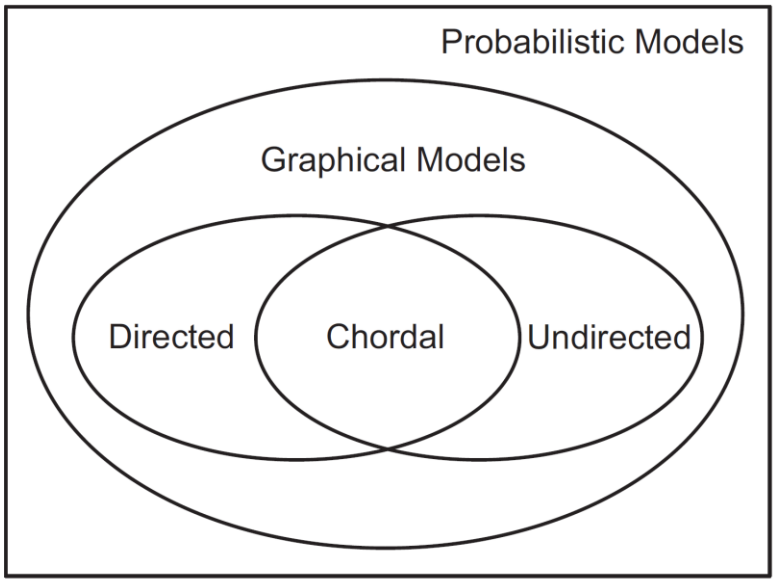




DGM vs UGM

- Some distributions can be perfectly modeled by either a DGM or a UGM; the resulting graphs are called decomposable or chordal.

- In chordal graph, if we collapse together all the variables in each maximal clique, to make "mega-variables", the resulting graph will be a tree.



Parameterization of MRFs

- Although the CI properties of UGM are simpler and more natural than for DGMs, representing the joint distribution for a UGM is less natural than for a DGM.
- Since there is no topological ordering associated with an undirected graph, we can't use the chain rule to represent $p(\boldsymbol{y})$.
- So instead of associating CPDs with each node, we associate potential function(s) or factor(s) $\psi_c(\boldsymbol{y}_c|\boldsymbol{\theta}_c)$ with each maximal clique in the graph.
- A potential function can be any non-negative function of its arguments
- The joint distribution is then defined to be proportional to the product of clique potentials

The Hammersley-Clifford theorem

- A positive distribution $p(\boldsymbol{y}) > 0$ satisfies the CI properties of an undirected graph G if p can be represented as a product of factors, one per maximal clique, i.e.,

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{c \in C} \psi_c(\boldsymbol{y}_c|\boldsymbol{\theta}_c)$$

Partition function

Set of all maximal clique

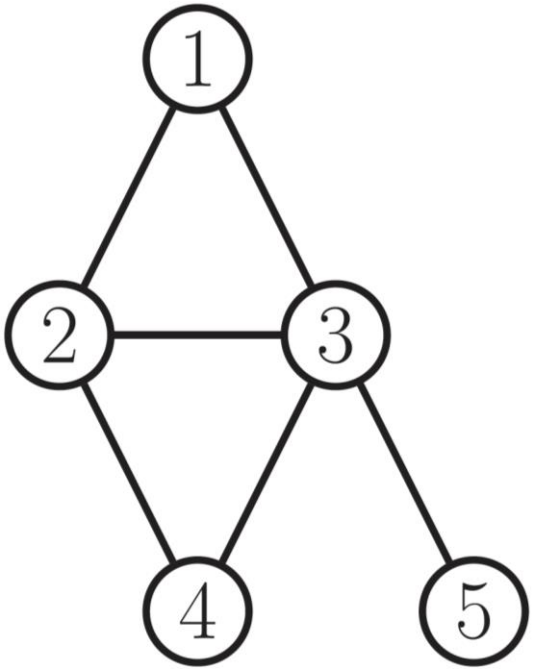
- Koller, D. and N. Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press

The Hammersley-Clifford theorem

- A positive distribution $p(\boldsymbol{y}) > 0$ satisfies the CI properties of an undirected graph G if p can be represented as a product of factors, one per maximal clique, i.e.,

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{c \in C} \psi_c(\boldsymbol{y}_c|\boldsymbol{\theta}_c)$$

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \psi_{123}(y_1, y_2, y_3) \psi_{234}(y_2, y_3, y_4) \psi_{35}(y_3, y_5)$$



Parameterization of MRFs

- There is a deep connection between UGMs and statistical physics.
- There is a model known as the Gibbs distribution:

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-\sum_{c} E(\boldsymbol{y}_c|\boldsymbol{\theta}_c)\right)$$

Energy associated with the variables in clique *c*

Parameterization of MRFs

- There is a deep connection between UGMs and statistical physics.
- There is a model known as the Gibbs distribution:

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-\sum_c E(\boldsymbol{y}_c|\boldsymbol{\theta}_c)\right)$$

- We can convert this to a UGM by defining

$$\psi_c(\boldsymbol{y}_c|\boldsymbol{\theta}_c) = \exp\left(-E(\boldsymbol{y}_c|\boldsymbol{\theta}_c)\right)$$

- Models of this form are known as energy based models, and are commonly used in physics and biochemistry, as well as some branches of machine learning (LeCun et al. 2006).

- LeCun, Y., S. Chopra, R. Hadsell, F.-J. Huang, and M.-A. Ranzato (2006). A tutorial on energy-based learning. In B. et al. (Ed.), *Predicting Structured Outputs*. MIT press.

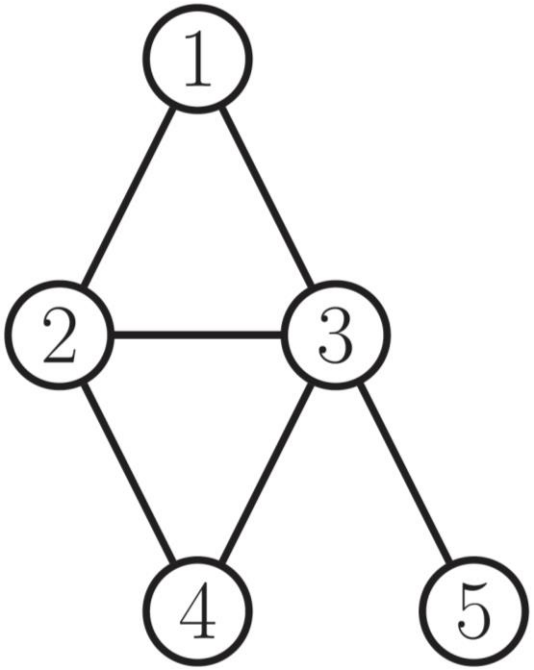
Pairwise MRF

- We are free to restrict the parameterization to the edges of the graph, rather than the maximal cliques. This is called a pairwise MRF.

$$p(\boldsymbol{y}|\boldsymbol{\theta}) \propto \psi_{12}(y_1, y_2)\psi_{13}(y_1, y_3)\psi_{23}(y_2, y_3)$$
$$\psi_{24}(y_2, y_4)\psi_{34}(y_3, y_4)\psi_{35}(y_3, y_5)$$

- This form is widely used due to its simplicity.

- Although, the representation based on maximal clique is more general.



Representing Potential Functions

- If the variables are discrete, we can represent the potential or energy functions as tables of (non-negative) numbers.
- The potential are not probabilities and represents relative "compatibility" between the different assignments to the potential.
- A more general approach is to define the log potentials as a linear function of the parameters

$$\log \psi_c(\boldsymbol{y}_c) \triangleq \phi_c(\boldsymbol{y}_c)^T \boldsymbol{\theta}_c$$

Feature vector

Maximum Entropy or Log-Linear Model

- If the variables are discrete, we can represent the potential or energy functions as tables of (non-negative) numbers.
- The potential are not probabilities and represents relative "compatibility" between the different assignments to the potential.
- A more general approach is to define the log potentials as a linear function of the parameters

$$\log \psi_c(\boldsymbol{y}_c) \triangleq \phi_c(\boldsymbol{y}_c)^T \boldsymbol{\theta}_c$$

- The resulting log-probability is expressed as:

$$\log p(\boldsymbol{y}|\boldsymbol{\theta}) = \sum_c \phi_c(\boldsymbol{y}_c)^T \boldsymbol{\theta}_c - Z(\boldsymbol{\theta})$$

Maximum entropy or log-linear model

Pairwise MRF: Potential Functions

- Consider a pairwise MRF, where for each edge, we associate a feature vector of length K^2:

$$\phi_{st}(y_s, y_t) = [\ldots, \mathbb{I}(y_s = j, y_t = k), \ldots]$$

Tabular representation

Representing Potential Functions

- Consider a pairwise MRF, where for each edge, we associate a feature vector of length K^2:

$$\phi_{st}(y_s, y_t) = [\dots, \mathbb{I}(y_s = j, y_t = k), \dots]$$

- If we have a weight for each feature, we can convert this into a $K \times K$ potential function as follows:

$$\psi_{st}(y_s = j, y_t = k) = \exp\left([\boldsymbol{\theta}_{st}^T \phi_{st}]_{jk}\right) = \exp\left(\theta_{st}(j,k)\right)$$

- This indicates that we can easily represent tabular potentials using a log-linear form.
- However, the log-linear form is more general.

Model of English Spelling

- Suppose we are interested in making a probabilistic model of English spelling
- Since certain letter combinations occur together quite frequently (e.g., “ing”), we will need higher order factors to capture this.
- Suppose we limit ourselves to letter trigrams.
- A tabular potential still has $26^3 = 17{,}576$ parameters in it.
- However, most of these triples will never occur.

Model of English Spelling

- An alternative approach is to define indicator functions that look for certain “special” triples, such as “ing”, “qu-”, etc.

- With this setup, we can define the potential on each trigram as:

$$\psi(y_{t-1}, y_t, y_{t+1}) = \exp\left(\sum_k \theta_k \phi_k(y_{t-1}, y_t, y_{t+1})\right)$$

indexes the different features, corresponding to “ing”, “qu-”, etc.

binary feature function

Representing Potential Functions

- An alternative approach is to define indicator functions that look for certain “special” triples, such as “ing”, “qu-”, etc.

- With this setup, we can define the potential on each trigram as:

$$\psi(y_{t-1}, y_t, y_{t+1}) = \exp\left(\sum_k \theta_k \phi_k(y_{t-1}, y_t, y_{t+1})\right)$$

- By tying the parameters across locations, we can define the probability of a word

$$p(\boldsymbol{y}|\boldsymbol{\theta}) \propto \exp\left(\sum_t \sum_k \theta_k \phi_k(y_{t-1}, y_t, y_{t+1})\right)$$

- Where these feature functions come from? Often from domain expertise but also can be learned from data!

Ising Model

- The Ising model is an example of an MRF that arose from statistical physics
- It was originally used for modeling the behavior of magnets.
- In particular, let $y_s \in \{-1,1\}$ represent the spin of an atom, which can either be spin down or up.
- In some magnets, called **ferro-magnets**, neighboring spins tend to line up in the same direction.
- In other kinds of magnets, called **anti-ferromagnets**, the spins "want" to be different from their neighbors.
- This can be modelled using MRF.

Ising model

- We create a graph in the form of a 2D or 3D lattice, and connect neighboring variables.

- We then define the following pairwise clique potential

$$\psi_{st}(y_s, y_t) = \begin{pmatrix} e^{w_{st}} & e^{-w_{st}} \\ e^{-w_{st}} & e^{w_{st}} \end{pmatrix}$$

Coupling strength between nodes

Ising Model: Pairwise Clique Potential

- We create a graph in the form of a 2D or 3D lattice, and connect neighboring variables.
- We then define the following pairwise clique potential

$$\psi_{st}(y_s, y_t) = \begin{pmatrix} e^{w_{st}} & e^{-w_{st}} \\ e^{-w_{st}} & e^{w_{st}} \end{pmatrix}$$

- If two nodes are not connected in the graph, we set $w_{st} = 0$.
- Assume that the weight matrix **W** is symmetric, so $w_{st} = w_{ts}$.
- Often it is assumed that all the edges have same strength, $w_{st} = J$.

Ising Model

- If all the weights are positive, $J > 0$, then neighboring spins are likely to be in the same state.
- This can be used to model **ferromagnets**, and is an example of an associative Markov network
- If the weights are sufficiently strong, the corresponding probability distribution will have two modes (ground states): one corresponding to +1s and one corresponding to -1s.
- If all of the weights are negative, $J < 0$, then the spins want to be different from their neighbors.
- This can be used to model an **anti-ferromagnet**, and results in a frustrated system.
- Computing the partition function $Z(J)$ can be done in polynomial time for associative Markov networks, but is NP-hard in general.

- Cipra, B. (2000). The Ising Model Is NP-Complete. *SIAM News 33*(6).

Ising Model & Gaussian Graphical Model

- There is an interesting analogy between Ising models and Gaussian graphical models
- Assuming, $y_t \in \{-1, +1\}$, we can write the unnormalized log probability of an Ising model as

$$\log \tilde{p}_y = -\sum_{s \sim t} y_s w_{st} y_t = -\frac{1}{2} \boldsymbol{y}^T \mathbf{W} \boldsymbol{y}$$

- The factor of $\frac{1}{2}$ arises because we sum each edge twice.
- If $w_{st} = J > 0$, we get a low energy (i.e., high probability) if neighboring states agree

Ising Model & Gaussian Graphical Model

- Sometimes there is an external field, which is an energy term which is added to each spin.

- This can be modelled using a local energy term of the form $-\mathbf{b}^T \boldsymbol{y}$. The modified distribution is given by

$$\log \tilde{p}_y = -\frac{1}{2}\boldsymbol{y}^T \mathbf{W} \boldsymbol{y} - \mathbf{b}^T \boldsymbol{y}$$

Bias term

Ising Model & Gaussian Graphical Model

- Sometimes there is an external field , which is an energy term which is added to each spin.
- This can be modelled using a local energy term of the form $-\mathbf{b}^T \boldsymbol{y}$. The modified distribution is given by

$$\log \tilde{p}_y = -\frac{1}{2}\boldsymbol{y}^T \mathbf{W} \boldsymbol{y} - \mathbf{b}^T \boldsymbol{y}$$

- If we define $\boldsymbol{\mu} \triangleq -\frac{1}{2}\boldsymbol{\Sigma}^{-1}\mathbf{b}$, $\boldsymbol{\Sigma}^{-1} = \mathbf{W}$ and $c \triangleq \frac{1}{2}\boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$, we can rewrite this in a form that looks similar to a Gaussian:

$$\tilde{p}_y \propto \exp\left(-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}) + c\right)$$

- One very important difference from the case of Gaussian (where $Z = |2\pi\boldsymbol{\Sigma}|$ at a cost of $\mathcal{O}(D^3)$) is that, for the Ising model, normalization constant requires summing over all 2^D bit vectors, which is NP-hard (Jerrum et al. 2004).

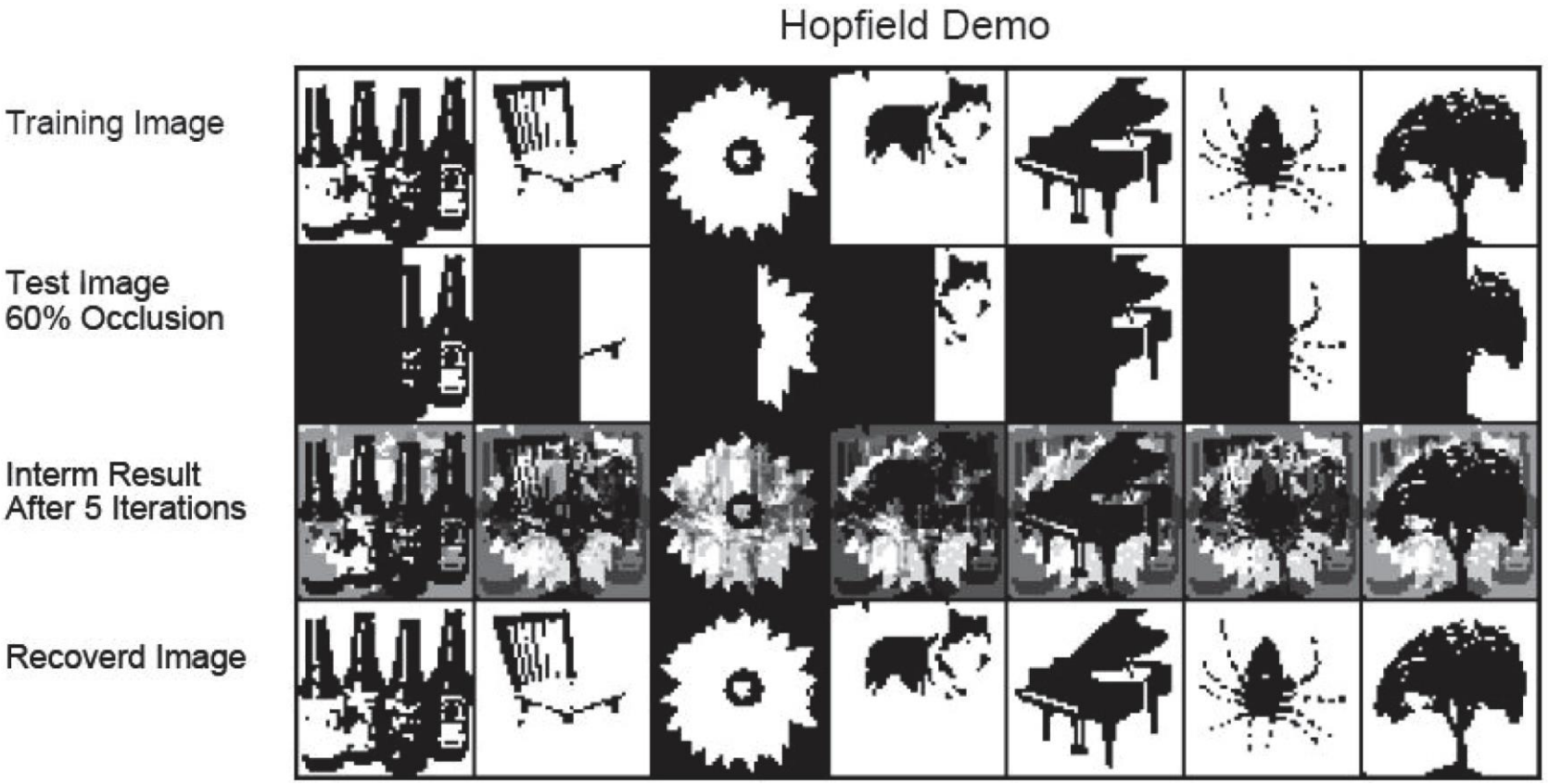
- Jerrum, M., A. Sinclair, and E. Vigoda (2004). A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. *Journal of the ACM*, 671– 697.

Hopfield Networks

- A Hopfield network (Hopfield 1982) is a fully connected Ising model with a symmetric weight matrix $\mathbf{W} = \mathbf{W}^T$.
- These weights, plus the bias terms $\mathbf{b}$, can be learned from training data using (approximate) maximum likelihood.
- The main application of Hopfield networks is as an associative memory or content addressable memory
  - suppose we train on a set of fully observed bit vectors, corresponding to patterns we want to memorize
  - at test time, we present a partial pattern to the network
  - we would like to estimate the missing variables
  - this is called pattern completion

- Hopfield, J. J. (1982, April). Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Science, USA 79*(8), 2554--2558.
- Hillar, C., J. Sohl-Dickstein, and K. Koepsell (2012, April). Efficient and optimal binary hopfield associative memory storage using minimum probability flow. Technical report.

Hopfield Networks



This can be thought of as retrieving an example from memory based on a piece of the example itself, hence the term “associative memory”

Hopfield Networks

- Exact inference is intractable in this model
- It is standard to use a coordinate descent algorithm known as iterative conditional modes (ICM).
- ICM sets each node to its most likely (lowest energy) state, given all its neighbors.

$$p(y_s = 1 | \boldsymbol{y}_{-s}, \boldsymbol{\theta}) = sigm\left(\boldsymbol{w}_{s,:}^T \boldsymbol{y}_{-s} + b_s\right)$$

- Most probable solution:

$$y_s^* = 1 \; if \sum_t w_{st} y_t > -b_s$$

$$y_s^* = 0 \; otherwise$$

Hopfield Networks

- Since inference is deterministic, it is also possible to interpret this model as a recurrent neural network.
- A Boltzmann machine generalizes the Hopfield/Ising model by including some hidden nodes, which makes the model representationally more powerful
- Inference in such models often uses Gibbs sampling, which is a stochastic version of the Iterative Conditional Modes (ICM).
- However, we could equally well apply Gibbs to a Hopfield net and ICM to a Boltzmann machine

- Hertz, J., A. Krogh, and R. G. Palmer (1991). *An Introduction to the Theory of Neural Comptuation*. Addison-Wesley.

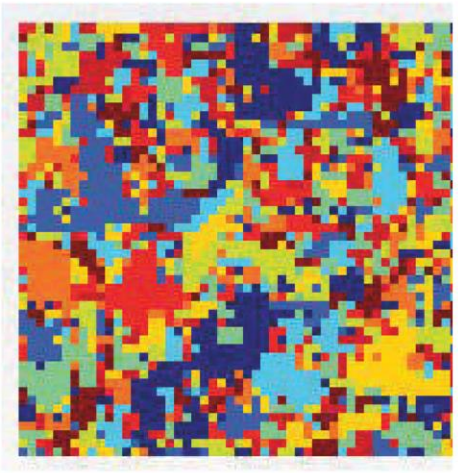
Potts Model

- It is easy to generalize the Ising model to multiple discrete states, $y_t \in \{1,2,\ldots,K\}$.
- The common potential function used is of the form ($K = 3$)

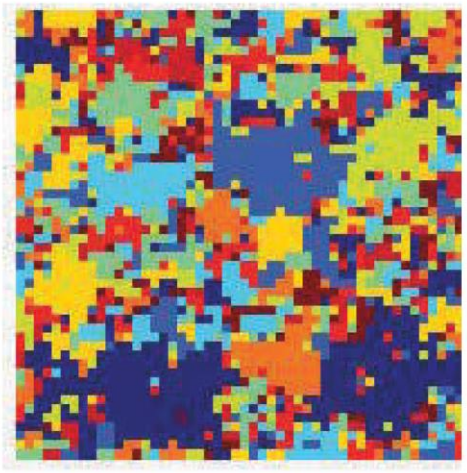
$$\psi_{st}(y_s, y_t) = \begin{pmatrix} e^J & 0 & 0 \\ 0 & e^J & 0 \\ 0 & 0 & e^J \end{pmatrix}$$

- This is called Potts Model.
- If $J > 0$, then neighboring nodes are encouraged to have the same label.

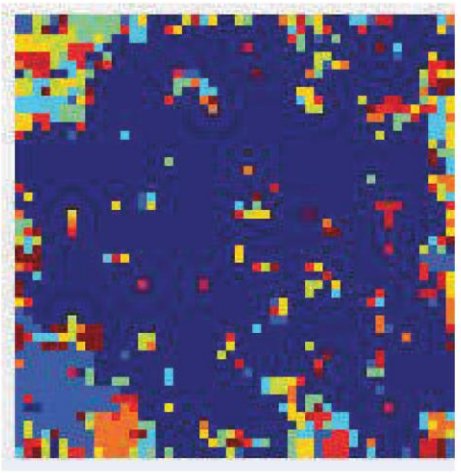
Potts Model



$J = 1.42$



$J = 1.44$



$J = 1.46$

gibbsDemoIsing from PMTK3

- We see that for $J > 1.44$, large clusters occur.
- For $J < 1.44$, many small clusters occur
- At the critical value of $J = 1.44$, there is a mix of small and large clusters.
- This rapid change in behavior as we vary a parameter of the system is called a phase transition

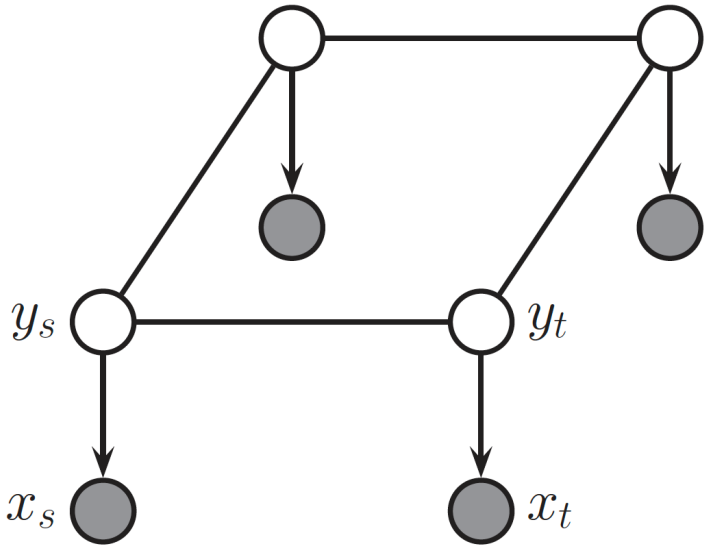
- MacKay, D. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Potts Model

- The Potts model can be used as a prior for image segmentation.
  - it says that neighboring pixels are likely to have the same discrete label and hence belong to the same segment

- We can combine this prior with a likelihood term as follows:

$$p(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta}) = p(\boldsymbol{y}|J) \prod_t p(x_t|y_t, \boldsymbol{\theta}) = \left[\frac{1}{Z(J)} \prod_{s \sim t} \psi(y_s, y_t; J)\right] \prod_t p(x_t|y_t, \boldsymbol{\theta})$$



Probability of observing a pixel given it belongs to class k

$$p(x_t|y_t = k, \boldsymbol{\theta})$$

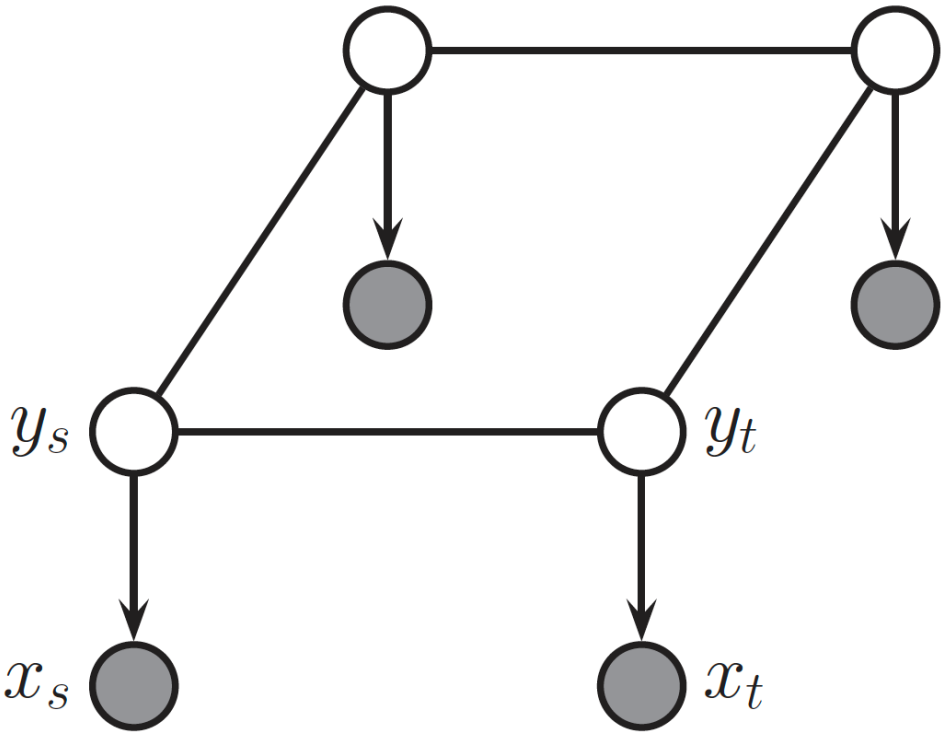
Potts Model

- The Potts model can be used as a prior for image segmentation.
  - it says that neighboring pixels are likely to have the same discrete label and hence belong to the same segment
- We can combine this prior with a likelihood term as follows:

$$p(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta}) = p(\boldsymbol{y}|J) \prod_t p(x_t|y_t, \boldsymbol{\theta}) = \left[\frac{1}{Z(J)} \prod_{s \sim t} \psi(y_s, y_t; J)\right] \prod_t p(x_t|y_t, \boldsymbol{\theta})$$

- This observation model can be modeled using a Gaussian or a non-parametric density

Potts Model



- The undirected 2d lattice represents the prior $p(y)$; in addition, there are directed edge from each y_t to its corresponding x_t, representing the local evidence

Potts Model

- This combination of an undirected and directed graph is called a chain graph.
- However, since the x_t nodes are observed, they can be "absorbed" into the model, thus leaving behind an undirected "backbone".
- This model is a 2d analog of an HMM, and could be called a partially observed MRF.
- Unfortunately, inference on this model is difficult and we have to use approximate inference.
- Potts prior is adequate for regularizing supervised learning problems
- It is not sufficiently accurate to perform image segmentation in an unsupervised manner.

- Morris, R. D., X. Descombes, and J. Zerubia (1996). The Ising/Potts model is not well suited to segmentation tasks. In *IEEE DSP Workshop*.
- Sudderth, E. and M. Jordan (2008). Shared Segmentation of Natural Scenes Using Dependent Pitman-Yor Processes. In *NIPS*

Gaussian MRFs

- An undirected GGM, also called a Gaussian MRF (see e.g., (Rue and Held 2005)), is a pairwise MRF of the following form

$$p(\boldsymbol{y}|\boldsymbol{\theta}) \propto \prod_{s \sim t} \psi_{st}(y_s, y_t) \prod_t \psi_t(y_t)$$

$$\psi_{st}(y_s, y_t) = \exp\left(-\frac{1}{2} y_s \Lambda_{st} y_t\right)$$

$$\psi_t(y_t) = \exp\left(-\frac{1}{2}\Lambda_{tt} y_t^2 + \eta_t y_t\right)$$

- Note that we could easily absorb the node potentials ψ_t into the edge potentials, but we have kept them separate for clarity

- Rue, H. and L. Held (2005). *Gaussian Markov Random Fields: Theory and Applications*, Volume 104 of *Monographs on Statistics and Applied Probability*. London: Chapman & Hall.

Gaussian MRFs

- The joint distribution can be written as

$$p(\boldsymbol{y}|\boldsymbol{\theta}) \propto \exp\left(\eta^T y - \frac{1}{2} y^T \Lambda y\right)$$

- We recognize this as a multivariate Gaussian written in **information form** where

$$\Lambda = \Sigma^{-1}$$
$$\eta = \Lambda\mu$$

- If $\Lambda_{st} = 0$, then there is no pairwise term connecting s and t.
- The zero entries in Λ are called structural zeros, since they represent the absent edges in the graph.
- Thus undirected GGMs correspond to sparse precision matrices.
  - This is used to learn the structure of graphical model.

Gaussian DGM vs. Gaussian UGM

- Directed GGMs correspond to sparse regression matrices and hence, sparse Cholesky factorizations of covariance matrices.
- Undirected GGMs correspond to sparse precision matrices.
- The advantage of the DAG formulation is that we can make the regression weights $\mathbf{W}$, and hence $\boldsymbol{\Sigma}$, be conditional on covariate information (Pourahmadi 2004), without worrying about positive definite constraints
- The disadvantage of the DAG formulation is its dependence on the order

- Pourahmadi, M. (2004). Simultaneous Modelling of Covariance Matrices: GLM, Bayesian and Nonparametric Perspectives. Technical report, Northern Illinois University.

Gaussian DGM vs. Gaussian UGM

- Can combine both representations, resulting in a Gaussian chain graph
- Consider a discrete-time, second-order Markov chain in which the states are continuous, $\boldsymbol{y}_t \in \mathbb{R}^D$.
- The transition function can be represented as a (vector-valued) linear-Gaussian CPD

$$p(\boldsymbol{y}_t|\boldsymbol{y}_{t-1}, \boldsymbol{y}_{t-2}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t|\mathbf{A}_1\boldsymbol{y}_{t-1} + \mathbf{A}_2\boldsymbol{y}_{t-2}, \boldsymbol{\Sigma})$$

- This is called vector auto-regressive or VAR(2)
- The time series aspect is most naturally modeled using a DGM
- The correlation amongst the components within a time slice is most naturally modeled using a UGM

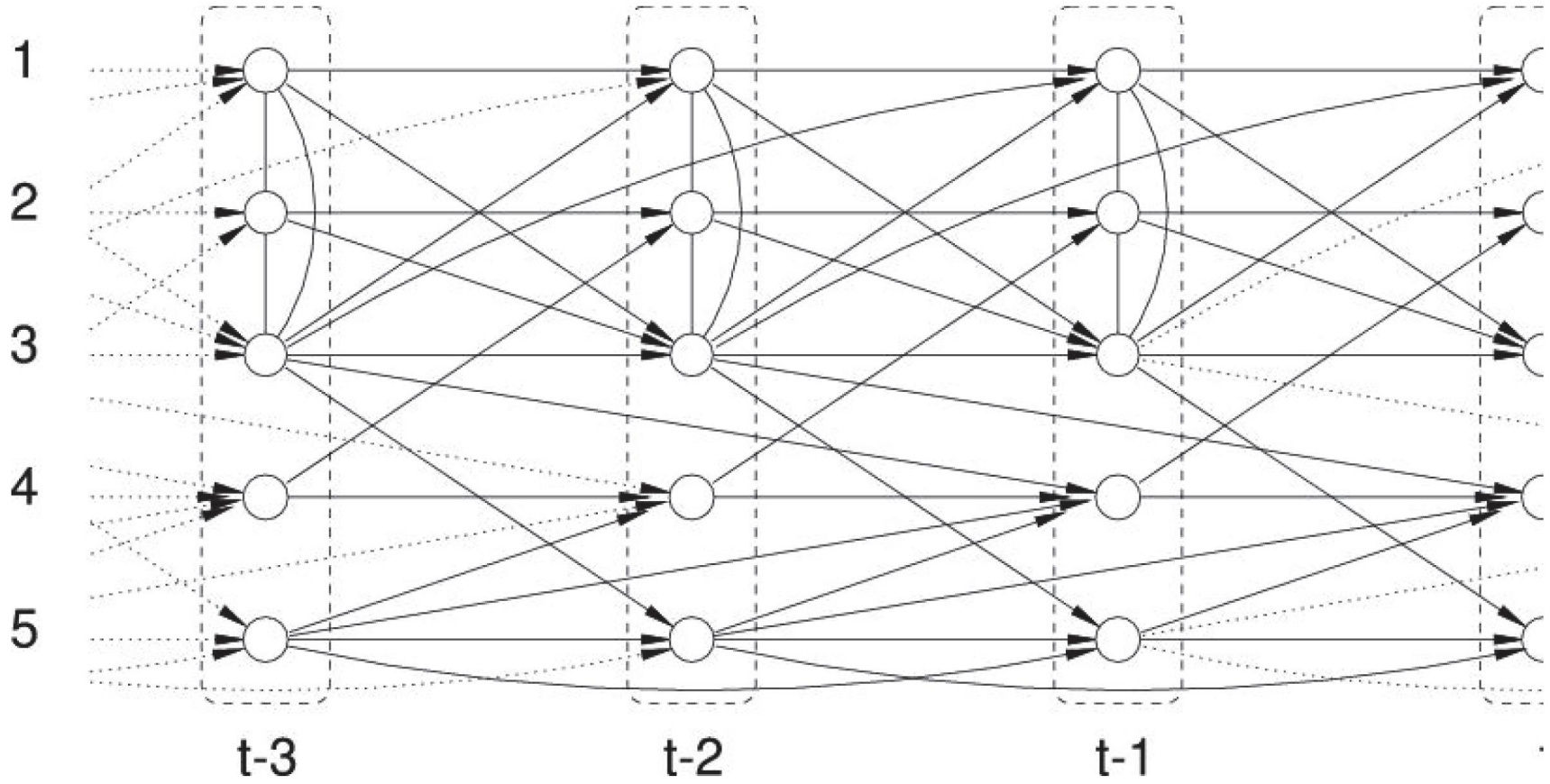
Dynamic Chain Graph

$$\mathbf{A}_1 = \begin{pmatrix} \frac{3}{5} & 0 & \frac{1}{5} & 0 & 0 \\ 0 & \frac{3}{5} & 0 & -\frac{1}{5} & 0 \\ \frac{2}{5} & \frac{1}{3} & \frac{3}{5} & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{5} \\ 0 & 0 & \frac{1}{5} & 0 & \frac{2}{5} \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 0 & 0 & -\frac{1}{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{5} & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & -\frac{1}{5} \end{pmatrix}$$

$$\mathbf{\Sigma} = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & 0 & 0 \\ \frac{1}{2} & 1 & -\frac{1}{3} & 0 & 0 \\ \frac{1}{3} & -\frac{1}{3} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{\Sigma}^{-1} = \begin{pmatrix} 2.13 & -1.47 & -1.2 & 0 & 0 \\ -1.47 & 2.13 & 1.2 & 0 & 0 \\ -1.2 & 1.2 & 1.8 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- Zeros in the transition matrices $\boldsymbol{A}_1, \boldsymbol{A}_2$ correspond to absent directed arcs from y_{t-1}, y_{t-2} into y_t.
- Zeros in the precision matrix $\boldsymbol{\Sigma}^{-1}$ correspond to absent undirected arcs between nodes in y_t.

Dynamic Chain Graph

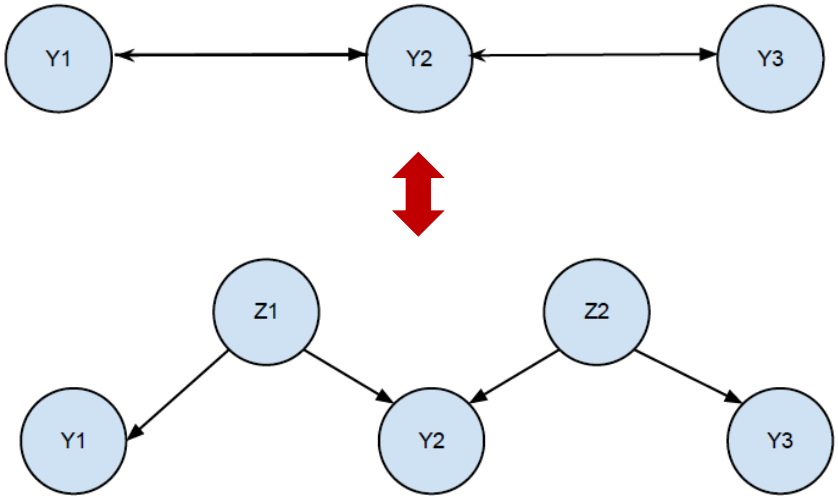


A VAR(2) process represented as a dynamic chain graph (Dahlhaus and Eichler 2000)

- Dahlhaus, R. and M. Eichler (2000). Causality and graphical models for time series. In P. Green, N. Hjort, and S. Richardson (Eds.), *Highly structured stochastic systems*. Oxford University Press.

Bidirected Graphs and DAGs

- Sometimes we have a sparse covariance matrix rather than a sparse precision matrix
- This can be represented using a bi-directed graph , where each edge has arrows in both directions.
- Here nodes that are not connected are unconditionally independent



$$Y_1 \perp Y_3$$

$$\Sigma_{1,3} = \Sigma_{31} = 0$$

- We can combine bidirected and directed edges to get a directed mixed graphical model. These are useful in ARMA models.

- Choi, M. J. (2011). *Trees and Beyond: Exploiting and Improving Tree-Structured Graphical Models*. Ph.D. thesis, MIT.

Markov Logic Networks

- Creating models for complex domains having variable number of objects and relationships is often done using first-order logic
- For example, consider the sentences “Smoking causes cancer” and “If two people are friends, and one smokes, then so does the other”.
- We can write these sentences in first-order logic as

$$\forall x, Sm(x) \Longrightarrow Ca(x)$$

$$\forall x. \forall y. Fr(x, y) \wedge Sm(x) \Longrightarrow Sm(y)$$

- Logical approaches to AI are no longer widely used.
- There have been a variety of attempts to combine first order logic with probability theory, an area known as **statistical relational AI.**

Markov Logic Networks

- One simple approach is to take logical rules and attach weights (known as certainty factors) to them, and then to interpret them as conditional probability distributions
- For example, we might say $p(Ca(x) = 1|Sm(x) = 1) = 0.9$
- Unfortunately, the rule does not say what to predict if $Sm(x) = 0$.
- Furthermore, combining CPDs in this way is not guaranteed to define a consistent joint distribution, because the resulting graph may not be a DAG

Markov Logic Networks

- An alternative approach is to treat these rules as a way of defining potential functions in an unrolled UGM.
- The result is known as a Markov logic network (Domingos and Lowd 2009).
- To specify the network, we first rewrite all the rules in conjunctive normal form (CNF), also known as clausal form

$$\neg Sm(x) \vee Ca(x)$$

$$\neg Fr(x, y) \vee \neg Sm(x) \vee Sm(y)$$

- Either x does not smoke or he has cancer

Markov Logic Networks

- Inference in first-order logic is only semi-decidable, so it is common to use a restricted subset
- A common approach (as used in Prolog) is to restrict the language to Horn clauses , which are clauses that contain at most one positive literal.
- Essentially this means the model is a series of if-then rules, where the right hand side of the rules (the “then” part, or consequence) has only a single term
- Once we have encoded our knowledge base as a set of clauses, we can attach weights to each one; these weights are the parameter of the model, and they define the clique potentials

$$\psi_x(\boldsymbol{x}_c) = \exp\left(w_c \phi_c(\boldsymbol{x}_c)\right)$$

Logical expression

Markov Logic Networks

- Inference in first-order logic is only semi-decidable, so it is common to use a restricted subset
- A common approach (as used in Prolog) is to restrict the language to Horn clauses , which are clauses that contain at most one positive literal.
- Essentially this means the model is a series of if-then rules, where the right hand side of the rules (the “then” part, or consequence) has only a single term
- Once we have encoded our knowledge base as a set of clauses, we can attach weights to each one; these weights are the parameter of the model, and they define the clique potentials

$$\psi_x(\boldsymbol{x}_c) = \exp\left(w_c \phi_c(\boldsymbol{x}_c)\right)$$

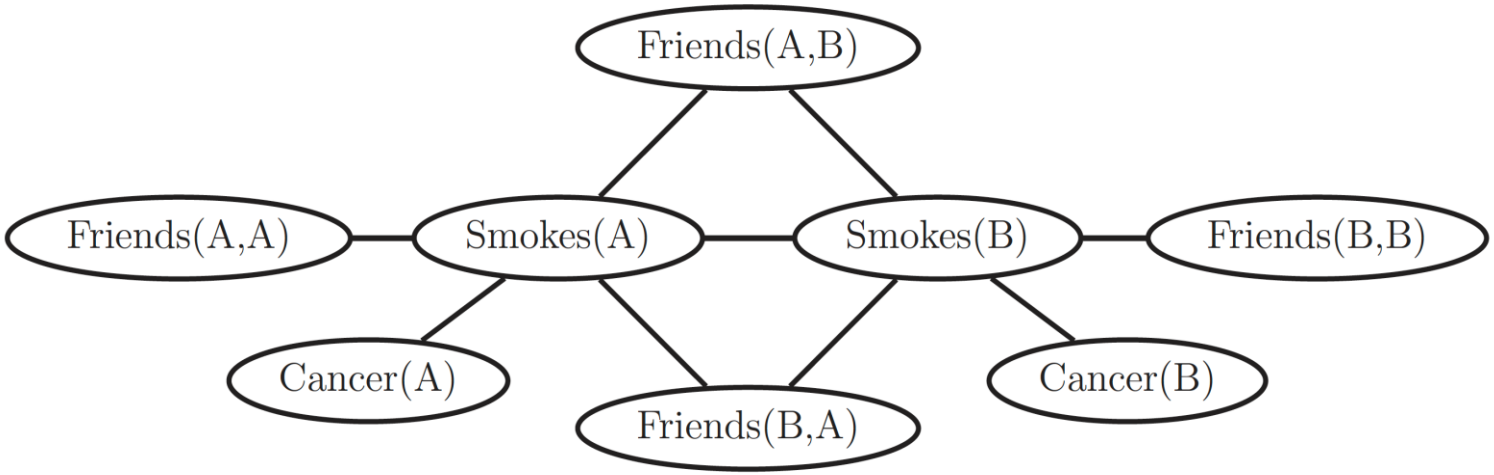
weight

Markov Logic Networks

$$\psi_x(\boldsymbol{x}_c) = \exp\left(w_c \phi_c(\boldsymbol{x}_c)\right)$$

- The weight of a clause specifies the probability of a world in which this clause is satisfied relative to a world in which it is not satisfied.
- Suppose there are two objects (people) in the world, Anna and Bob, which we will denote by constant symbols A and B.
- We can make a ground network from the above clauses by creating binary random variables S_x, C_x and $F_{x,y}$, for $x, y \in \{A, B\}$.

Markov Logic Networks



- Note that we have not encoded the fact that Fr is a symmetric relation and hence $Fr(A, B)$ and $Fr(B, A)$ may have different values.
- Similarly, we have the "degenerate" nodes $Fr(A, A)$ and $Fr(B, B)$ because we did not specify $x \neq y$.

Markov Logic Networks

- We can think of MLNs as a convenient way of specifying a UGM template, that can get unrolled to handle data of arbitrary size.
- There are several other ways to define relational probabilistic models; see e.g., (Koller and Friedman 2009; Kersting et al. 2011) for details
- In some cases, there is uncertainty about the number or existence of objects or relations (the so-called open universe problem)

- Koller, D. and N. Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press
- Kersting, K., S. Natarajan, and D. Poole (2011). Statistical Relational AI: Logic, Probability and Computation. Technical report, UBC.

Learning: Gradient Method

- Consider an MRF in log-linear form

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_c \boldsymbol{\theta}_c^T \phi_c(\boldsymbol{y})\right)$$

- The scaled log-likelihood is given by

$$l(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_i \left[\sum_c \boldsymbol{\theta}_c^T \phi_c(\boldsymbol{y}_i) - \log Z(\boldsymbol{\theta})\right]$$

- Since MRFs are in the exponential family, we know that this function is convex in $\boldsymbol{\theta}$.

- So it has a unique global maximum which we can find using gradient-based optimizers.

Learning: Gradient Method

- In particular, the derivative for the weights of a particular clique

$$\frac{\partial l}{\partial \boldsymbol{\theta}_c} = \frac{1}{N}\sum_i \left[\phi_c(\boldsymbol{y}_i) - \frac{\partial}{\partial \boldsymbol{\theta}_c}\log Z(\boldsymbol{\theta})\right]$$

- The derivative of the log partition function w.r.t. $\boldsymbol{\theta}_c$ is the expectation of the *c*'th feature under the model:

$$\frac{\partial}{\partial \boldsymbol{\theta}_c}\log Z(\boldsymbol{\theta}) = \mathbb{E}(\phi_c(\boldsymbol{y})|\boldsymbol{\theta}) = \sum_{\boldsymbol{y}} \phi_c(\boldsymbol{y}) p(\boldsymbol{y}|\boldsymbol{\theta})$$

$$\frac{\partial l}{\partial \boldsymbol{\theta}_c} = \left[\frac{1}{N}\sum_i \phi_c(\boldsymbol{y}_i) - \mathbb{E}(\phi_c(\boldsymbol{y})|\boldsymbol{\theta})\right]$$

Learning: Gradient Method

$$\frac{\partial l}{\partial \boldsymbol{\theta}_c} = \left[\frac{1}{N} \sum_i \phi_c(\boldsymbol{y}_i) - \mathbb{E}(\phi_c(\boldsymbol{y}) | \boldsymbol{\theta}) \right]$$

- In the first term, we fix $\boldsymbol{y}$ to its observed values; this is sometimes called the clamped term.
- In the second term, $\boldsymbol{y}$ is free; this is sometimes called the unclamped term.
- Note that computing the unclamped term requires inference in the model, and this must be done once per gradient step.
- This makes UGM training much slower.

Learning: Gradient Method

- The gradient of the log likelihood can be rewritten as

$$\frac{\partial l}{\partial \boldsymbol{\theta}_c} = \mathbb{E}_{p_{emp}}[\phi_c(\boldsymbol{y})] - \mathbb{E}_{p(\cdot|\boldsymbol{\theta})}[\phi_c(\boldsymbol{y})]$$

- At the optimum, the gradient will be zero

$$\mathbb{E}_{p_{emp}}[\phi_c(\boldsymbol{y})] = \mathbb{E}_{p(\cdot|\boldsymbol{\theta})}[\phi_c(\boldsymbol{y})]$$

- This is called moment matching

- This observation motivates a different optimization algorithm to be discussed later on.

Learning: Partially Observed Models

- Suppose we have missing data and/or hidden variables in our model.
- In general, we can represent such models as follows

$$p(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_c \boldsymbol{\theta}_c^T \phi_c(\boldsymbol{h}, \boldsymbol{y})\right)$$

- The log likelihood has the form

$$l(\boldsymbol{\theta}) = \frac{1}{N}\sum_i \log\left(\sum_{h_i} p(\boldsymbol{y}_i, \boldsymbol{h}_i|\boldsymbol{\theta})\right) = \frac{1}{N}\sum_i \log\left(\frac{1}{Z(\boldsymbol{\theta})}\sum_{h_i} \tilde{p}(\boldsymbol{y}_i, \boldsymbol{h}_i|\boldsymbol{\theta})\right)$$

$$\tilde{p}(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{\theta}) \triangleq \exp\left(\sum_c \boldsymbol{\theta}_c^T \phi_c(\boldsymbol{h}, \boldsymbol{y})\right)$$

Learning: Partially Observed Models

- The log likelihood has the form

$$l(\boldsymbol{\theta}) = \frac{1}{N}\sum_i \log\left(\sum_{h_i} p(\boldsymbol{y}_i, \boldsymbol{h}_i|\boldsymbol{\theta})\right)$$

- This term is the same as the partition function for the whole model, except that $\boldsymbol{y}$ is fixed at y_i.

- Hence the gradient is just the expected features where we clamp y_i, but averages over $\mathbf{h}$:

$$\frac{\partial}{\partial \boldsymbol{\theta}_c}\log\left(\sum_{h_i} p(\boldsymbol{y}_i, \boldsymbol{h}_i|\boldsymbol{\theta})\right) = \mathbb{E}(\phi_c(\boldsymbol{h}, \boldsymbol{y}_i)|\boldsymbol{\theta})$$

- So the overall gradient is given by

$$\frac{\partial l}{\partial \boldsymbol{\theta}_c} = \frac{1}{N}\sum_i \{\mathbb{E}(\phi_c(\boldsymbol{h}, y_i)|\boldsymbol{\theta}) - \mathbb{E}(\phi_c(\boldsymbol{h}, \boldsymbol{y})|\boldsymbol{\theta})\}$$

Learning: Partially Observed Models

- So the overall gradient is given by

$$\frac{\partial l}{\partial \boldsymbol{\theta}_c} = \frac{1}{N}\sum_i \{\mathbb{E}(\phi_c(\boldsymbol{h}, y_i)|\boldsymbol{\theta}) - \mathbb{E}(\phi_c(\boldsymbol{h}, \boldsymbol{y})|\boldsymbol{\theta})\}$$

- The first set of expectations are computed by "clamping" the visible nodes to their observed values.
- The second set are computed by letting the visible nodes be free.
- In both cases we marginalize the hidden variables.
- An alternative approach is to use generalized EM, where we use gradient methods in the M-step

  - Koller, D. and N. Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press (pp. 956)

Approximate Methods for Computing the MLEs of MRFs

- When fitting a UGM there is (in general) no closed form solution for the ML or the MAP estimate
- So we need to use gradient-based optimizers
- This gradient requires inference
- In models where inference is intractable, learning also becomes intractable
- This has motivated various computationally faster alternatives to ML/MAP estimation

Approximate Methods for Computing the MLEs of MRFs

Method	Restriction	Exact MLE?
Closed form	Only Chordal MRF	Exact
IPF	Only Tabular / Gaussian MRF	Exact
Gradient-based optimization	Low tree width	Exact
Max-margin training	Only CRFs	N/A
Pseudo-likelihood	No hidden variables	Approximate
Stochastic ML	-	Exact (up to MC error)
Contrastive divergence	-	Approximate
Minimum probability flow	Can integrate out the hiddens	Approximate

Pseudo Likelihood

- One alternative to MLE is to maximize the pseudo likelihood

$$l_{PL}(\boldsymbol{\theta}) \triangleq \sum_{y} \sum_{d=1}^{D} p_{emp}(\boldsymbol{y}) \log p(y_d | \boldsymbol{y}_{-\boldsymbol{d}}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{d=1}^{D} \log p\left(y_{id} \middle| \boldsymbol{y}_{i,-d}, \boldsymbol{\theta}\right)$$

- That is, we optimize the product of the full conditionals, also known as the composite likelihood.

- Lindsay, B. (1988). Composite likelihood methods. *Contemporary Mathematics 80*(1), 221–239.
- Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician 24*, 179–196.
- Liang, P. and M. I. Jordan (2008). An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *International Conference on Machine Learning (ICML)*.

Pseudo Likelihood

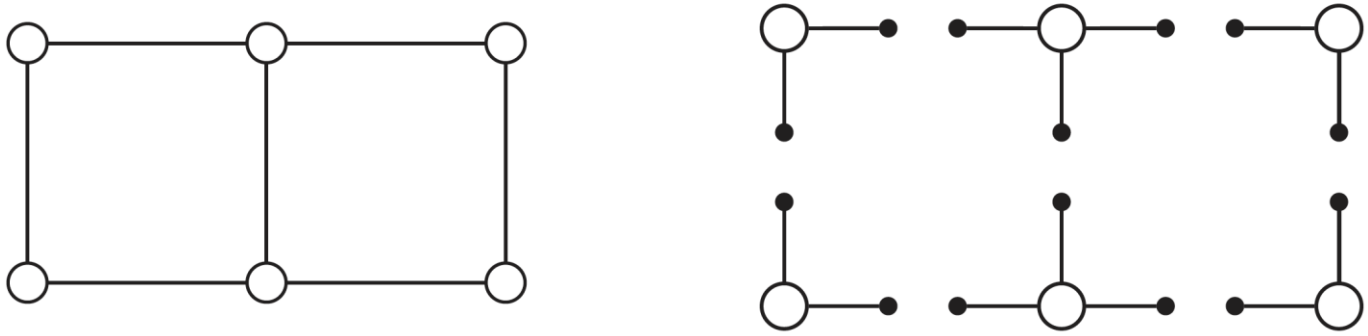
$$l_{PL}(\boldsymbol{\theta}) \triangleq \sum_{y} \sum_{d=1}^{D} p_{emp}(\boldsymbol{y}) \log p(y_d | \boldsymbol{y}_{-d}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{d=1}^{D} \log p\left(y_{id} \middle| \boldsymbol{y}_{i,-d}, \boldsymbol{\theta}\right)$$

$$l_{ML}(\boldsymbol{\theta}) = \sum_{y} p_{emp}\,(\boldsymbol{y}) \log p(\boldsymbol{y}|\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \log p(\boldsymbol{y}_i|\boldsymbol{\theta})$$

- In the case of Gaussian MRFs, PL is equivalent to ML (Besag 1975)
- But this is not true in general (Liang and Jordan 2008)

- Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician 24*, 179–196.
- Liang, P. and M. I. Jordan (2008). An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *International Conference on Machine Learning (ICML)*.

Pseudo Likelihood



- In PL, we learn to predict each node given all of its neighbors.
- This objective is generally fast to compute since each full conditional $p(y_{id}|\boldsymbol{y}_{i,-d},\boldsymbol{\theta})$ only requires summing over the states of a single node, y_{id}, order to compute the local normalization constant
- The PL approach is similar to fitting each full conditional separately except that the parameters are tied between adjacent nodes

- Parise, S. and M. Welling (2005). Learning in Markov Random Fields: An Empirical Study. In *Joint Statistical Meeting*.
- Hoefling, H. and R. Tibshirani (2009). Estimation of Sparse Binary Pairwise Markov Networks using Pseudo-likelihoods. *J. of Machine Learning Research 10*.

Stochastic Maximum Likelihood

- The gradient of the log-likelihood for a fully observed MRF is given by

$$\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \left[\frac{1}{N} \sum_i \phi_c(\boldsymbol{y}_i) - \mathbb{E}(\phi_c(\boldsymbol{y}) | \boldsymbol{\theta}) \right]$$

- The gradient for a partially observed MRF is similar

- In both cases, we can approximate the model expectations using Monte Carlo sampling.

- We can combine this with stochastic gradient descent, which takes samples from the empirical distribution

Stochastic Maximum Likelihood

Algorithm 19.1: Stochastic maximum likelihood for fitting an MRF

Initialize weights $\boldsymbol{\theta}$ randomly;
$k = 0$, $\eta = 1$;
for *each epoch* **do**
  for *each minibatch of size* B **do**
    for *each sample* $s = 1 : S$ **do**
      Sample $\mathbf{y}^{s,k} \sim p(\mathbf{y}|\boldsymbol{\theta}_k)$;
    $\hat{E}(\boldsymbol{\phi}(\mathbf{y})) = \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{\phi}(\mathbf{y}^{s,k})$;
    for *each training case* i *in minibatch* **do**
      $\mathbf{g}_{ik} = \boldsymbol{\phi}(\mathbf{y}_i) - \hat{E}(\boldsymbol{\phi}(\mathbf{y}))$;
    $\mathbf{g}_k = \frac{1}{B}\sum_{i \in B} \mathbf{g}_{ik}$;
    $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \mathbf{g}_k$;
    $k = k + 1$;
    Decrease step size η;

Stochastic Maximum Likelihood

- Typically we use MCMC to generate the samples.
- Of course, running MCMC to convergence at each step of the inner loop would be extremely slow
- Fortunately, it was shown in (Younes 1989) that we can start the MCMC chain at its previous value, and just take a few steps
- This is valid because $p(\boldsymbol{y}|\boldsymbol{\theta}^k)$ is likely to be close to $p(\boldsymbol{y}|\boldsymbol{\theta}^{k-1})$, since we only change the parameters by a small amount.
- This algorithm is close to the "persistent contrastive divergence" algorithm used in Deep Learning.

- Younes, L. (1989). Parameter estimation for imperfectly observed Gibbsian fields. *Probab. Theory and Related Fields 82*, 625–645

Feature Induction in MRF

- MRFs require a good set of features
- One unsupervised way to learn such features is known as feature induction.
- The idea is to start with a base set of features, and then to continually create new feature combinations out of old ones, greedily adding the best ones to the model (Pietra et al. 1997; Zhu et al. 1997).
- This was later extended to the CRF case in (McCallum 2003).

- Pietra, S. D., V. D. Pietra, and J. Lafferty (1997). Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence 19*(4).
- Zhu, C. S., N. Y. Wu, and D. Mumford (1997, November). Minimax entropy principle and its application to texture modeling. *Neural Computation 9*(8).
- McCallum, A. (2003). Efficiently inducing features of conditional random fields. In *UAI*.

Feature Induction in MRF

- To illustrate the basic idea, we present an example from (Pietra et al. 1997), which described how to build unconditional probabilistic models to represent English spelling

- Initially the model has no features, which represents the uniform distribution

- The algorithm starts by choosing to add the feature

$$\phi_1(\boldsymbol{y}) = \sum_t \mathbb{I}(y_t \in \{a, \ldots, z\})$$

which checks if any letter is lower case or not

- Pietra, S. D., V. D. Pietra, and J. Lafferty (1997). Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence 19*(4).

Feature Induction in MRF

- The algorithm starts by choosing to add the feature

$$\phi_1(\boldsymbol{y}) = \sum_t \mathbb{I}(y_t \in \{a, \ldots, z\})$$

  which checks if any letter is lower case or not.

- After the feature is added, the parameters are (re)-fit by maximum likelihood.

- For this feature, it turns out that $\hat{\theta}_i = 1.944$.

- This means that a word with a lowercase letter in any position is about $e^{1.944} \approx 7$ times more likely than the same word without a lowercase letter in that position

Feature Induction in MRF

- The second feature (two adjacent characters being low case)

$$\phi_2(\boldsymbol{y}) = \sum_{s \sim t} \mathbb{I}(y_s \in \{a, \dots, z\}, y_t \in \{a, \dots, z\})$$

- Now the model has the form

$$p(\boldsymbol{y}) = \frac{1}{Z} \exp\big(\theta_1 \phi_1(\boldsymbol{y}) + \theta_2 \phi_2(\boldsymbol{y})\big)$$

- Continuing in this way, the algorithm adds features for the strings.

- This approach of feature learning can be thought of as a form of graphical model structure learning except it is more fine-grained.
  - we add features that are useful, regardless of the resulting graph structure
  - However, the resulting graphs can become densely connected.

Iterative Proportional Fitting (IPF)

- Consider a pairwise MRF where the potentials are represented as tables, with one parameter per variable setting.
- We can represent this in log-linear form using

$$\psi_{st} = \exp(\theta_{st}^T[\mathbb{I}(y_s = 1, y_t = 1), \ldots, \mathbb{I}(y_s = K, y_t = K)])$$

- Thus the feature vectors are just indicator functions
- We know that at the maximum of the likelihood

$$\mathbb{E}_{p_{emp}}[\mathbb{I}(y_s = j, y_t = k)] = \mathbb{E}_{p(\cdot|\boldsymbol{\theta})}[\mathbb{I}(y_s = j, y_t = k)]$$

$$p_{emp}(y_s = j, y_t = k) = p(y_s = j, y_t = k|\boldsymbol{\theta})$$

Iterative Proportional Fitting (IPF)

- We know that at the maximum of the likelihood

$$\mathbb{E}_{p_{emp}}[\mathbb{I}(y_s = j, y_t = k)] = \mathbb{E}_{p(\cdot|\boldsymbol{\theta})}[\mathbb{I}(y_s = j, y_t = k)]$$

$$p_{emp}(y_s = j, y_t = k) = p(y_s = j, y_t = k|\boldsymbol{\theta})$$

$$p_{emp}(y_s = j, y_t = k) = \frac{N_{st,jk}}{N} = \frac{\sum_{n=1}^{N} \mathbb{I}(y_{ns} = j, y_{nt} = k)}{N}$$

- For a general graph, the condition that must hold at the optimum is

$$p_{emp}(\boldsymbol{y}_c) = p(\boldsymbol{y}_c|\boldsymbol{\theta})$$

- For a special family of graphs known as decomposable graphs, one can show that $p(\boldsymbol{y}_c|\boldsymbol{\theta}) = \psi_c(\boldsymbol{y}_c)$.

Iterative Proportional Fitting (IPF)

- For a special family of graphs known as decomposable graphs, one can show that $p(\boldsymbol{y}_c|\boldsymbol{\theta}) = \psi_c(\boldsymbol{y}_c)$.

- However, even if the graph is not decomposable, we can imagine trying to enforce this condition.

- This suggests an iterative coordinate ascent scheme where at each step we compute

$$\psi_c^{t+1}(\mathbf{y}_c) = \psi_c^t(\mathbf{y}_c) \times \frac{p_{emp}(\boldsymbol{y}_c)}{p(\boldsymbol{y}_c|\psi_c^t)}$$

  where the multiplication is elementwise.

- This is known as iterative proportional fitting or IPF.

Iterative Proportional Fitting (IPF)

Algorithm 19.2: Iterative Proportional Fitting algorithm for tabular MRFs

Initialize $\psi_c = 1$ for $c = 1 : C$;
repeat
  for $c = 1 : C$ **do**
    $p_c = p(\mathbf{y}_c|\boldsymbol{\psi})$;
    $\hat{p}_c = p_{\text{emp}}(\mathbf{y}_c)$;
    $\psi_c = \psi_c * \frac{\hat{p}_c}{p_c}$;
until *converged*;

- Fienberg, S. (1970). An iterative procedure for estimation in contingency tables. *Annals of Mathematical Statistics 41*(3), 907-917.
- Bishop, Y., S. Fienberg, and P. Holland (1975). *Discrete Multivariate Analysis: Theory and Practice*. MIT Press

Iterative Proportional Fitting (IPF): Example

- Let us consider a simple example from http://en.wikipedia.org/wiki/Iterative_proportional_fitting
- We have two binary variables, Y_1 and Y_2, where $Y_{n1} = 1$ if man n is left handed, and $Y_{n1} = 0$ otherwise.
- Similarly $Y_{n2} = 1$ if women n is left handed, and $Y_{n2} = 0$ otherwise.
- We can summarize the data as:

	right-handed	left-handed	Total
male	43	9	52
female	44	4	48
Total	87	13	100

Iterative Proportional Fitting (IPF): Example

- Suppose we want to fit a disconnected graphical model containing nodes Y_1 and Y_2 but with no edge between them.

$$p(Y_1, Y_2) = \frac{1}{Z}\psi_1(Y_1)\psi_2(Y_2)$$

- That is, we want to find vectors ψ_1 and ψ_2 such that

$$\mathbf{M} \triangleq \psi_1 \psi_2^T \approx \mathbf{C}$$

Model's expected count

Model's empirical count

Iterative Proportional Fitting (IPF): Example

- Suppose we want to fit a disconnected graphical model containing nodes Y_1 and Y_2 but with no edge between them.
- That is, we want to find vectors ψ_1 and ψ_2 such that

$$\mathbf{M} \triangleq \psi_1 \psi_2^T \approx \mathbf{C}$$

- By moment matching, we find that the row and column sums of the model must exactly match the row and column sums of the data
- One possible solution is to use $\psi_1 = [52 \quad 48]$ and $\psi_2 = [87 \quad 13]$.

	right-handed	left-handed	Total
male	45.24	6.76	52
female	41.76	6.24	48
Total	87	13	100

IPFdemo2x2 from PMTK3

Speed of IPF

- IPF is a fixed point algorithm for enforcing the moment matching constraints and is guaranteed to converge to the global optimum.
- The number of iterations depends on the form of the model.
- If the graph is decomposable, then IPF converges in a single iteration, but in general, IPF may require many iterations.
- It is clear that the dominant cost of IPF is computing the required marginals under the model.
- Efficient methods, such as the junction tree algorithm can be used, resulting in something called efficient IPF (Jirousek and Preucil 1995).

- Jirousek, R. and S. Preucil (1995). On the effective implementation of the iterative proportional fitting procedure. *Computational Statistics & Data Analysis 19*, 177–189.

Speed of IPF

- Coordinate descent can be slow.
- An alternative method is to update all the parameters at once, by simply following the gradient of the likelihood
- This gradient approach has the further significant advantage that it works for models in which the clique potentials may not be fully parameterized.
- Although it is possible to adapt IPF to this setting of general features, resulting in a method known as iterative scaling
- In practice the gradient method is much faster (Malouf 2002; Minka 2003)

- Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proc. Sixth Conference on Natural Language Learning (CoNLL-2002)*, pp. 49–55.
- Minka, T. (2003). A comparison of numerical optimizers for logistic regression. Technical report, MSR.

Generalization of IPF

- We can use IPF to fit Gaussian graphical models
- Instead of working with empirical counts, we work with empirical means and covariances (Speed and Kiiveri 1986)
- It is also possible to create a Bayesian IPF algorithm for sampling from the posterior of the model's parameters (see e.g., (Dobra and Massam 2010))

- Speed, T. and H. Kiiveri (1986). Gaussian Markov distributions over finite graphs. *Annals of Statistics 14*(1), 138–150.
- Dobra, A. and H. Massam (2010). The mode oriented stochastic search (MOSS) algorithm for log-linear models with conjugate priors. *Statistical Methodology 7*, 240–253.

IPF for Decomposable Graphical Models

- There is a special family of undirected graphical models known as decomposable graphical models
- The basic idea is that it contains graphs which are "tree-like".
- Such graphs can be represented by UGMs or DGMs without any loss of information
- In the case of decomposable graphical models, IPF converges in one iteration
- In fact, the MLE has a closed form solution (Lauritzen 1996).

- Lauritzen, S. (1996). *Graphical Models*. OUP.

IPF for Decomposable Graphical Models

- In particular, for tabular potentials we have

$$\hat{\psi}_c(\boldsymbol{y}_c = k) = \frac{\sum_{i=1}^{N} \mathbb{I}(y_{i,c} = k)}{N}$$

- Similarly, for Gaussian potentials, we have

$$\hat{\mu}_c = \frac{\sum_{i=1}^{N} \boldsymbol{y}_{ic}}{N}$$

$$\hat{\boldsymbol{\Sigma}}_c = \frac{\sum_i (\boldsymbol{y}_{ic} - \hat{\mu}_c)(\boldsymbol{x}_{ic} - \hat{\mu}_c)^T}{N}$$

- By using conjugate priors, we can also easily compute the full posterior over the model parameters in the decomposable case, just as we did in the DGM case

Conditional Random Fields

- In CRF (Lafferty et al. 2001) or discriminative random field (Kumar and Hebert 2003), the clique potentials are conditioned on input features

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w}) = \frac{1}{Z(\boldsymbol{x},\boldsymbol{w})}\prod_c \psi_c(\boldsymbol{y}_c|\boldsymbol{x},\boldsymbol{w})$$

- A CRF can be thought of as a structured output extension of logistic regression

- We will usually assume a log-linear representation of the potentials:

$$\psi_c(\boldsymbol{y}_c|\boldsymbol{x},\boldsymbol{w}) = \exp\left(\boldsymbol{w}_c^T \phi(\boldsymbol{x},\boldsymbol{y}_c)\right)$$

Feature vector

Local set of levels

Global input

- Lafferty, J., A. McCallum, and F. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Intl. Conf. on Machine Learning*.
- Kumar, S. and M. Hebert (2003). Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Intl. Conf. on Computer Vision*.

Conditional Random Fields

- In CRF (Lafferty et al. 2001) or discriminative random field (Kumar and Hebert 2003), the clique potentials are conditioned on input features

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w}) = \frac{1}{Z(\boldsymbol{x},\boldsymbol{w})}\prod_c \psi_c(\boldsymbol{y}_c|\boldsymbol{x},\boldsymbol{w})$$

- A CRF can be thought of as a structured output extension of logistic regression
- We will usually assume a log-linear representation of the potentials:

$$\psi_c(\boldsymbol{y}_c|\boldsymbol{x},\boldsymbol{w}) = \exp\left(\boldsymbol{w}_c^T\phi(\boldsymbol{x},\boldsymbol{y}_c)\right)$$

- In CRF, we don't need to "waste resources" modeling things that we always observe
- Instead we can focus on modeling what we care about.

- Lafferty, J., A. McCallum, and F. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Intl. Conf. on Machine Learning*.
- Kumar, S. and M. Hebert (2003). Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Intl. Conf. on Computer Vision*.

Conditional Random Fields

- Another important advantage of CRFs is that we can make the potentials (or factors) of the model be data-dependent.
  - in image processing applications, we may "turn off" the label smoothing between two neighboring nodes *s* and *t* if there is an observed discontinuity in the image intensity between pixels *s* and *t*.
  - in natural language processing problems, we can make the latent labels depend on global properties of the sentence
- The disadvantage of CRFs over MRFs is that they require labeled training data.
- CRFs are slower to train.

Chain-structured CRFs, MEMMs and the Label-Bias Problem

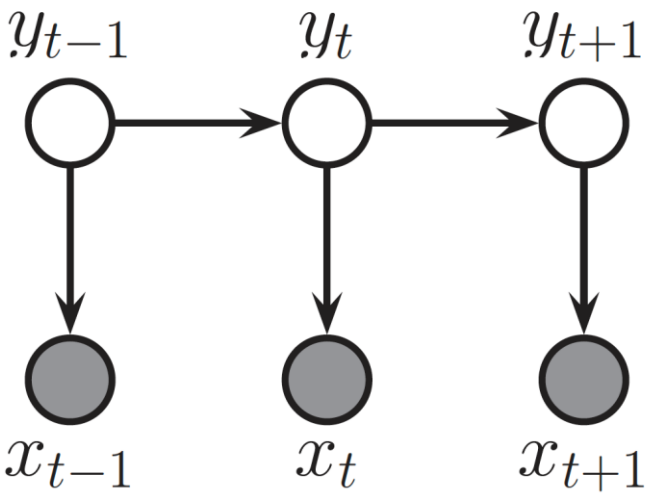
- The most widely used kind of CRF uses a chain-structured graph to model correlation amongst neighboring labels.
- Such models are useful for a variety of sequence labeling tasks.
- Traditionally, HMMs have been used for such tasks.
- An HMM requires specifying a generative observation model, $p(\boldsymbol{x_t}|\boldsymbol{y_t}, \boldsymbol{w})$ which can be difficult.
- Furthermore, each $\boldsymbol{x_t}$ is required to be local, since it is hard to define a generative model for the whole stream of observations.

Chain-structured CRFs, MEMMs and the Label-Bias Problem

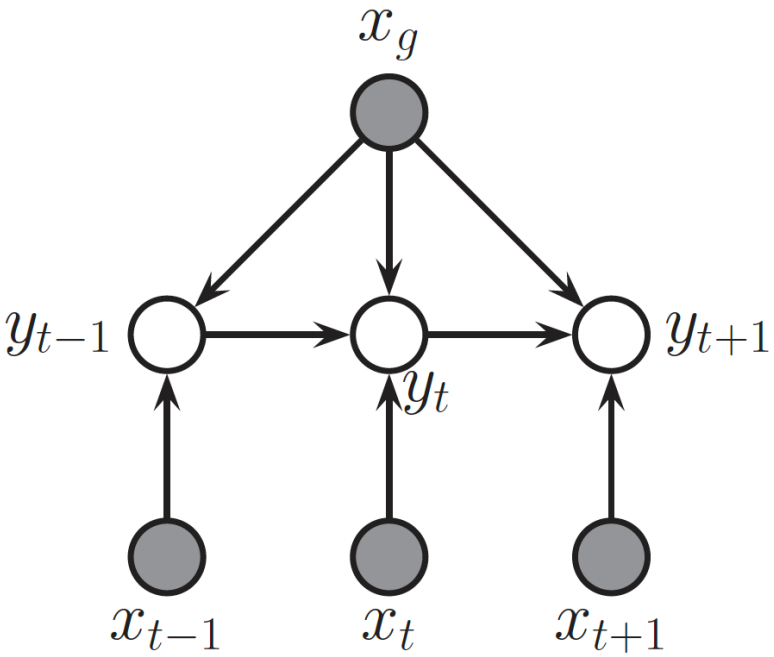
- An obvious way to make a discriminative version of an HMM is to "reverse the arrows" from $\boldsymbol{y}_t$ to $\boldsymbol{x}_t$.
- This is called a maximum entropy Markov model or MEMM (McCallum et al. 2000; Kakade et al. 2002)
- An MEMM is simply a Markov chain in which the state transition probabilities are conditioned on the input features.
- It is therefore a special case of an input-output HMM.
- Although this seems like the natural generalization of logistic regression to the structured output setting, it suffers from a subtle problem known (rather obscurely) as the label bias problem (Lafferty et al. 2001).

- Lafferty, J., A. McCallum, and F. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Intl. Conf. on Machine Learning*.
- McCallum, A., D. Freitag, and F. Pereira (2000). Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Intl. Conf. on Machine Learning*.
- Kakade, S., Y. W. Teh, and S. Roweis (2002). An alternate objective function for markovian fields In *Intl. Conf. on Machine Learning*.

Chain-structured CRFs, MEMMs and the Label-Bias Problem



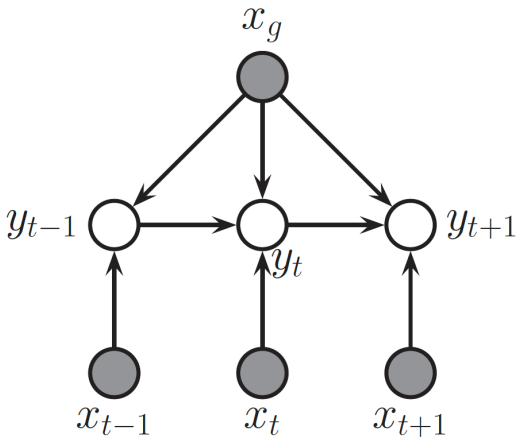
A generative directed HMM



A discriminative directed MEMM

Chain-structured CRFs, MEMMs and the Label-Bias Problem

- Although this seems like the natural generalization of logistic regression to the structured output setting, it suffers from a subtle problem known (rather obscurely) as the label bias problem (Lafferty et al. 2001).
- The problem is that local features at time *t* do not influence states prior to time t.
- This follows by examining the DAG, which shows that $\boldsymbol{x}_t$ is d-separated from $\boldsymbol{y}_{t-1}$ (and all earlier time points) by the v-structure at $\boldsymbol{y}_t$, which is a hidden child, thus blocking the information flow.



- Lafferty, J., A. McCallum, and F. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Intl. Conf. on Machine Learning*.

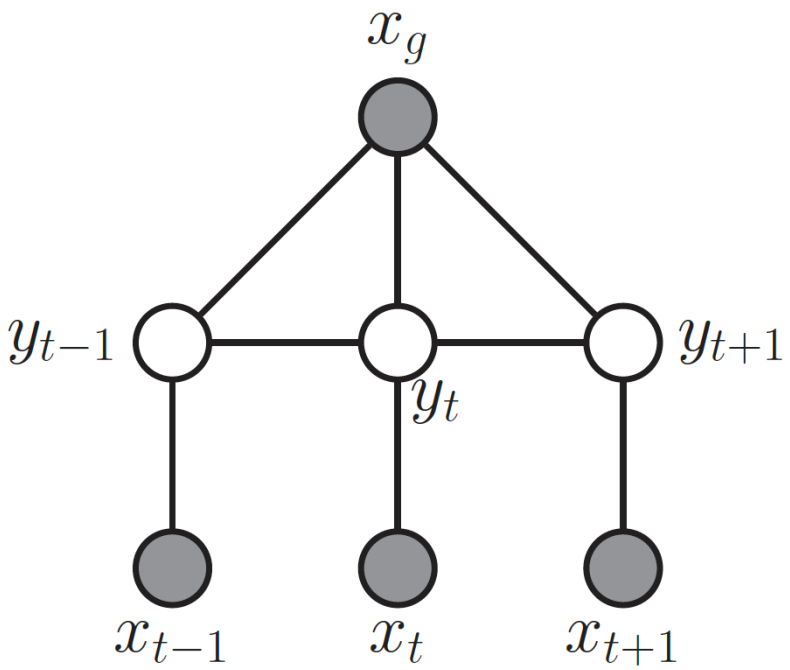
Chain-structured CRFs, MEMMs and the Label-Bias Problem

- Suppose we see the word “banks”; this could be a verb (as in “he banks at BoA”), or a noun (as in “the river banks were overflowing”).
- Locally the POS tag for the word is ambiguous.
- Suppose that later in the sentence, we see the word “fishing” - this gives us enough context to infer that the sense of “banks” is “river banks”.
- However, in an MEMM (unlike in an HMM and CRF), the “fishing” evidence will not flow backwards, so we will not be able to disambiguate “banks”

Chain-structured CRFs, MEMMs and the Label-Bias Problem

- Consider a chain-structured CRF:

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w}) = \frac{1}{Z(\boldsymbol{x},\boldsymbol{w})}\prod_{t=1}^{T}\psi(y_t|\boldsymbol{x},\boldsymbol{w})\prod_{t=1}^{T-1}\psi(y_t,y_{t-1}|\boldsymbol{x},\boldsymbol{w})$$



- We see that the label bias problem no longer exists

Chain-structured CRFs, MEMMs and the Label-Bias Problem

- The label bias problem in MEMMs occurs because directed models are locally normalized
- By contrast, MRFs and CRFs are globally normalized, which means that local factors do not need to sum to 1.
- However, this solution comes at a price: we do not get a valid probability distribution over $\mathbf{y}$ until we have seen the whole sentence, since only then can we normalize over all configurations.
- Consequently, CRFs are not as useful as DGMs (whether discriminative or generative) for online or real-time inference
- Furthermore, the fact that Z depends on all the nodes makes CRFs much slower to train than DGMs

Application of CRF: Handwriting Recognition

- A natural application of CRFs is to classify hand-written digit strings

- The key observation is that locally a letter may be ambiguous
- Depending on the (unknown) labels of one's neighbors, it is possible to use context to reduce the error rate
- Note that the node potential $\psi_t(y_t|x_t)$ is often taken to be a probabilistic discriminative classifier, such as a neural network or RVM, that is trained on isolated letters
- The edge potentials, $\psi_{st}(y_s, y_t)$ are often taken to be a language bigram model

Application of CRF: Noun Phrase Chunking

- One common NLP task is noun phrase chunking , which refers to the task of segmenting a sentence into its distinct noun phrases (NPs).
- This is a simple example of a technique known as shallow parsing.
- We tag each word in the sentence with B (meaning beginning of a new NP), I (meaning inside a NP), or O (meaning outside an NP). This is called BIO notation.

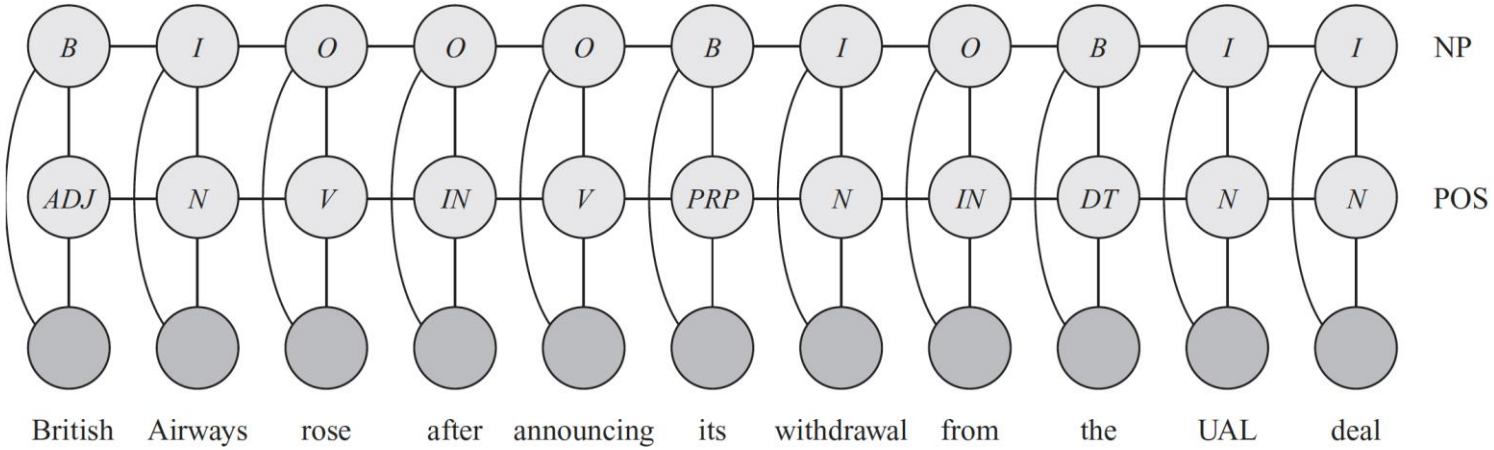
```
   B       I         O     O       O          B     I          O     B    I    I
(British Airways) rose after announcing (its withdrawl) from (the UAI deal)
```

Application of CRF: Noun Phrase Chunking

- One common NLP task is noun phrase chunking , which refers to the task of segmenting a sentence into its distinct noun phrases (NPs).
- This is a simple example of a technique known as shallow parsing.
- We tag each word in the sentence with B (meaning beginning of a new NP), I (meaning inside a NP), or O (meaning outside an NP). This is called BIO notation.
- A standard approach to this problem would first convert the string of words into a string of POS tags, and then convert the POS tags to a string of BIOs
- However, such a pipeline method can propagate errors

Application of CRF: Noun Phrase Chunking

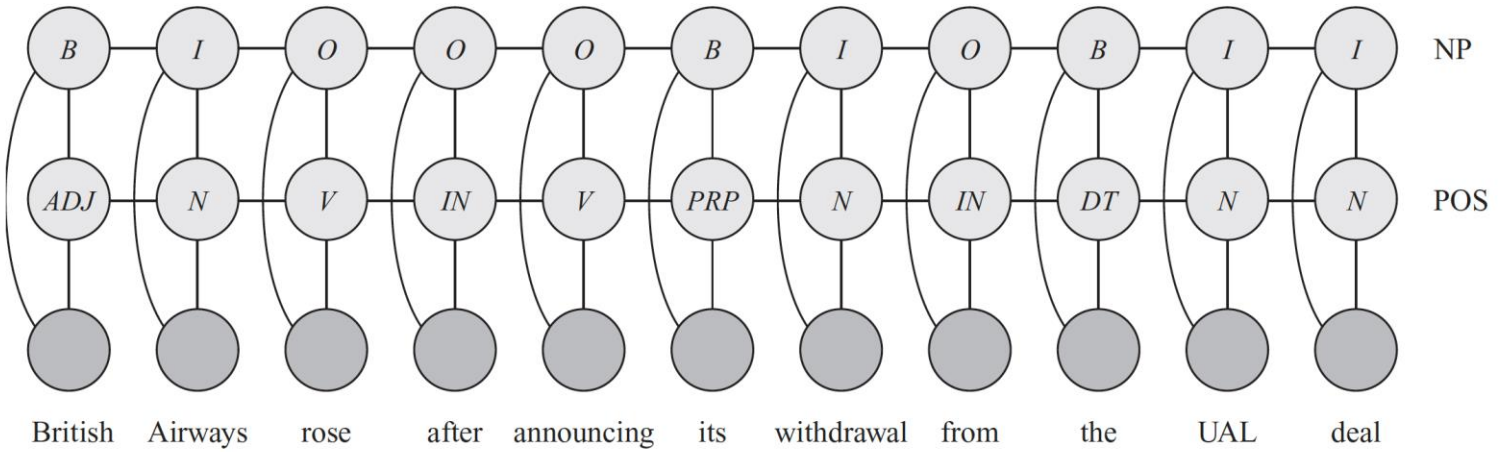
- A more robust approach is to build a joint probabilistic model of the form $p(\mathrm{NP}_{1:T}, \mathrm{POS}_{1:T} | \mathrm{words}_{1:T})$. One way to do this is to use the CRF.



KEY

B	Begin noun phrase	*V*	Verb
I	Within noun phrase	*IN*	Preposition
O	Not a noun phrase	*PRP*	Possesive pronoun
N	Noun	*DT*	Determiner (e.g., a, an, the)
ADJ	Adjective		

Application of CRF: Noun Phrase Chunking

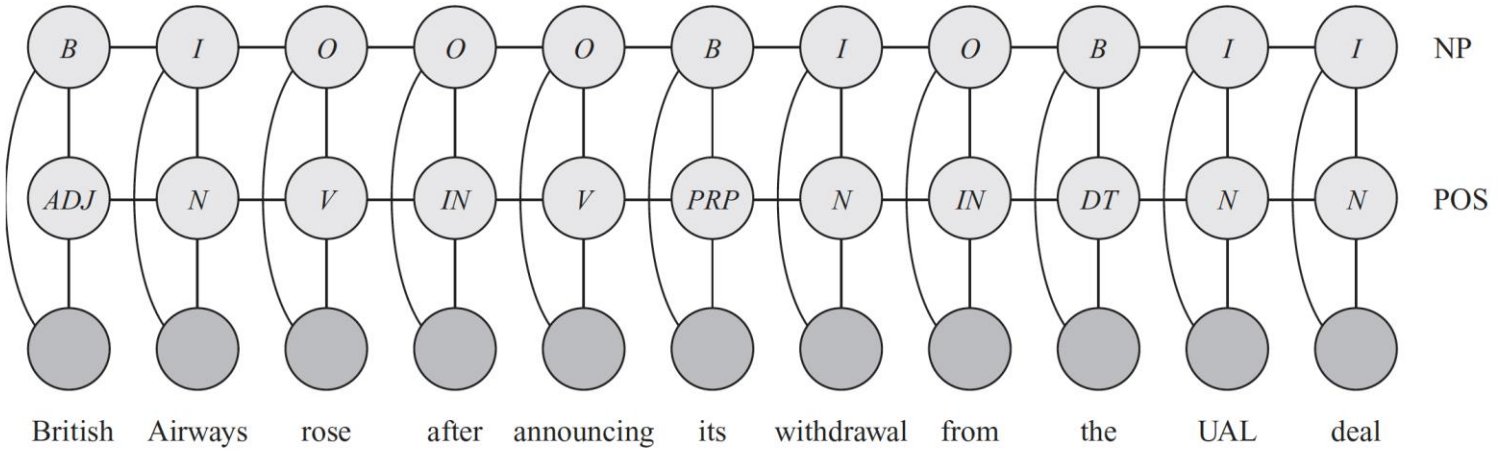


KEY

B	Begin noun phrase	*V*	Verb
I	Within noun phrase	*IN*	Preposition
O	Not a noun phrase	*PRP*	Possesive pronoun
N	Noun	*DT*	Determiner (e.g., a, an, the)
ADJ	Adjective		

- The connections between adjacent labels encode the probability of transitioning between the B, I and O states, and can enforce constraints such as the fact that B must preceed I

Application of CRF: Noun Phrase Chunking



KEY

B	Begin noun phrase	*V*	Verb
I	Within noun phrase	*IN*	Preposition
O	Not a noun phrase	*PRP*	Possesive pronoun
N	Noun	*DT*	Determiner (e.g., a, an, the)
ADJ	Adjective		

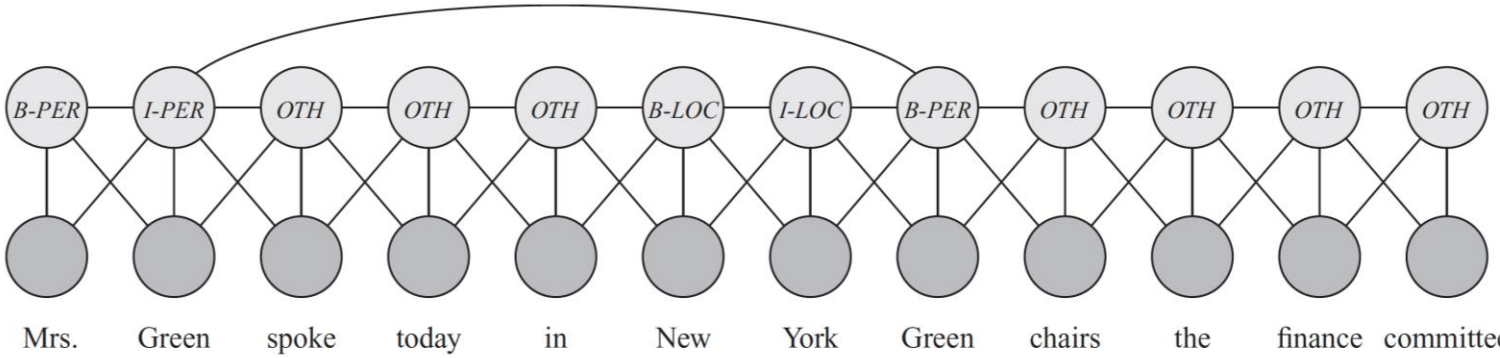
- The features are usually hand engineered and include things like: does this word begin with a capital letter, is this word followed by a full stop, is this word a noun, etc.
- Typically there are $\sim 1,000-10,000$ features per node

Application of CRF: Noun Phrase Chunking

- The number of features has minimal impact on the inference time
- This is because the features are observed and do not need to be summed over.
- There is a small increase in the cost of evaluating potential functions with many features, but this is usually negligible
- If not, one can use l_1 regularization to prune out irrelevant features.
- However, the graph structure can have a dramatic effect on inference time.

Application of CRF: Named Entity Recognition

- We can get better performance by considering long-range correlations between words
- For example, we might add a link between all occurrences of the same word, and force the word to have the same tag in each occurrence
- This is known as a skip-chain CRF.



KEY

B-PER	Begin person name	*I-LOC*	Within location name
I-PER	Within person name	*OTH*	Not an entitiy
B-LOC	Begin location name		

Application of CRF: Natural Language Parsing

- A generalization of chain-structured models for language is to use probabilistic grammar
- In particular, a probabilistic context free grammar or PCFG is a set of re-write or production rules of the form $\sigma \rightarrow \sigma'\sigma''$ or $\sigma \rightarrow x$, where $\sigma, \sigma', \sigma'' \in \Sigma$ are non-terminals (analogous to parts of speech)
- $x \rightarrow \mathcal{X}$ are terminals, i.e., words.
- Each such rule has an associated probability
- The resulting model defines a probability distribution over sequences of words
- We can compute the probability of observing a particular sequence by summing over all trees that generate it
- This can be done using the inside-outside algorithm.

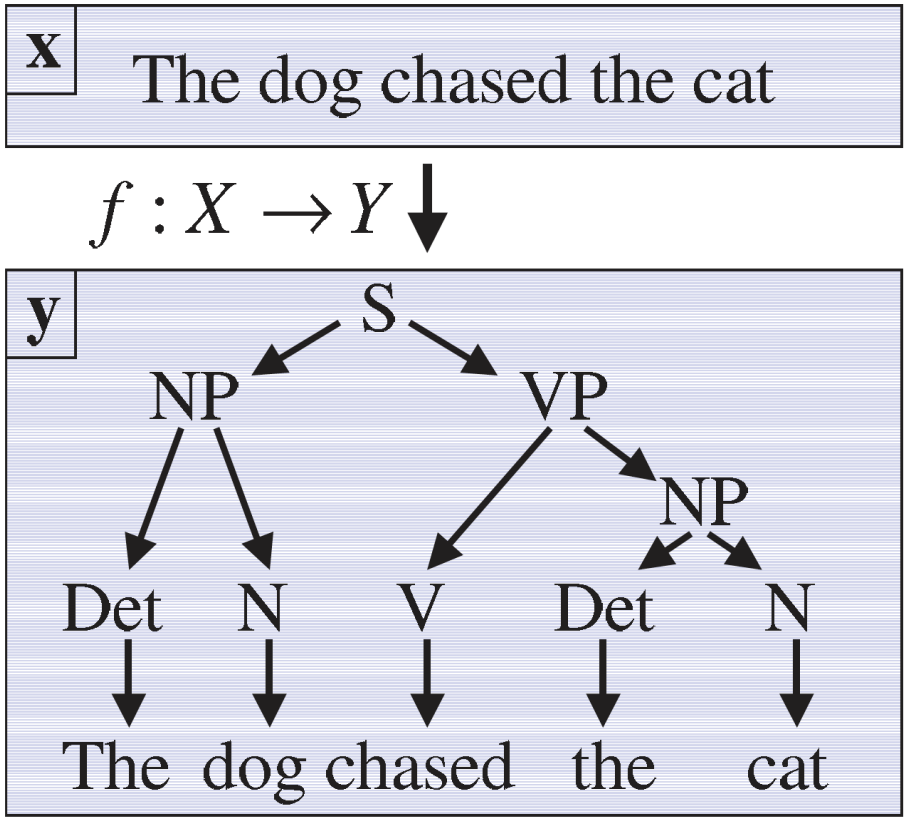
Application of CRF: Natural Language Parsing

- PCFGs are generative models
- It is possible to make discriminative versions which encode the probability of a labeled tree, $\mathbf{y}$, given a sequence of words $\mathbf{x}$, by using a CRF of the form

$$p(\boldsymbol{y}|\boldsymbol{x}) \propto \exp\left(\boldsymbol{w}^T \phi(\boldsymbol{x}, \boldsymbol{y})\right)$$

- For example, we might define $\phi(\boldsymbol{x}, \boldsymbol{y})$ to count the number of times each production rule was used (which is analogous to the number of state transitions in a chain-structured model)

Application of CRF: Natural Language Parsing



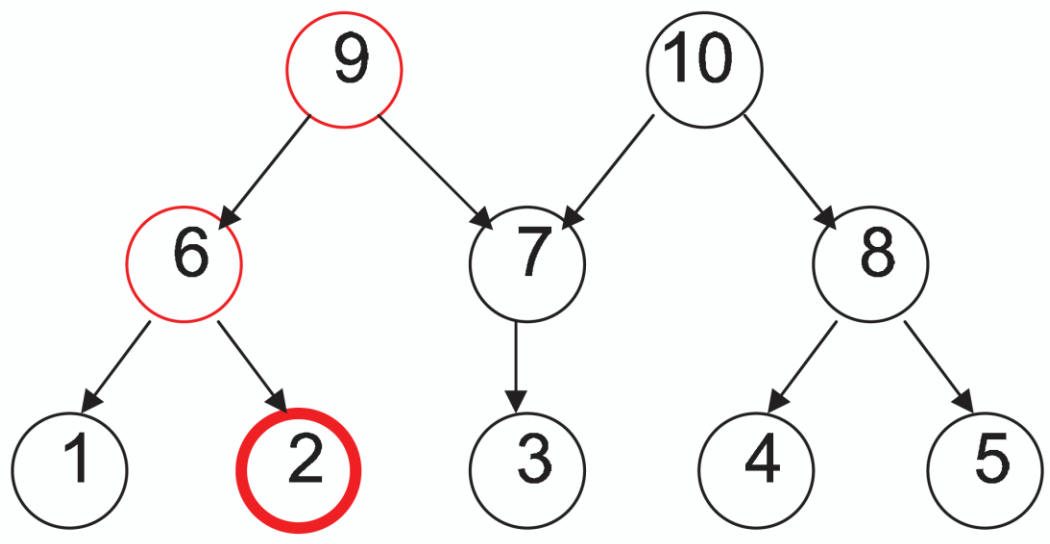
$$\Psi(\mathbf{x},\mathbf{y}) = \begin{pmatrix} 1 & S \rightarrow NP\ VP \\ 0 & S \rightarrow NP \\ 2 & NP \rightarrow Det\ N \\ 1 & VP \rightarrow V\ NP \\ \vdots & \\ 0 & Det \rightarrow dog \\ 2 & Det \rightarrow the \\ 1 & N \rightarrow dog \\ 1 & V \rightarrow chased \\ 1 & N \rightarrow cat \end{pmatrix}$$

Application of CRF: Hierarchical Classification

- Suppose we are performing multi-class classification, where we have a label taxonomy , which groups the classes into a hierarchy.
- We can encode the position of y within this hierarchy by defining a binary vector $\phi(y)$, where we turn on the bit for component y and for all its children.
- This can be combined with input features $\phi(\boldsymbol{x})$ using a tensor product

$$\phi(\boldsymbol{x}, y) = \phi(\boldsymbol{x}) \otimes \phi(y)$$

Application of CRF: Hierarchical Classification



$$\Lambda(2) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad \Psi(\mathbf{x}, 2) = \begin{pmatrix} 0 \\ \mathbf{x} \\ 0 \\ 0 \\ 0 \\ \mathbf{x} \\ 0 \\ 0 \\ \mathbf{x} \\ 0 \end{pmatrix}$$

$$\langle \mathbf{w}, \Psi(\mathbf{x}, 2) \rangle = \langle w_2, \mathbf{x} \rangle + \langle w_6, \mathbf{x} \rangle + \langle w_9, \mathbf{x} \rangle$$

Figure: Illustration of a simple label taxonomy, and how it can be used to compute a distributed representation for the label for class 2. In this figure, $\phi(\mathbf{x}) = \mathbf{x}$, $\phi(y = 2) = \Lambda(2)$, $\phi(\mathbf{x}, y)$ is denoted by $\Psi(\mathbf{x}, 2)$, and $\mathbf{w}^T \phi(\mathbf{x}, y)$ is denoted by $\langle \mathbf{w}, \Psi(\mathbf{x}, 2) \rangle$. Source: Figure 5.1 of (Altun et al. 2006) . Used with kind permission of Yasemin Altun.

Application of CRF: Hierarchical Classification

- This method is widely used for text classification, where manually constructed taxnomies (such as the Open Directory Project at www.dmoz.org) are quite common

- The benefit is that information can be shared between the parameters for nearby categories, enabling generalization across classes.

Application of CRF: Protein Side Chain Prediction

- An interesting analog to the skip-chain model arises in the problem of predicting the structure of protein side chains
- Each residue in the side chain has 4 dihedral angles, which are usually discretized into 3 values called rotamers.
- The goal is to predict this discrete sequence of angles, $\boldsymbol{y}$, from the discrete sequence of amino acids, $\boldsymbol{x}$.

Application of CRF: Stereo Vision

- Low-level vision problems are problems where the input is an image (or set of images), and the output is a processed version of the image.
- In such cases, it is common to use 2d lattice structured models.
- A classic low-level vision problem is dense stereo reconstruction.
- The goal is to estimate the depth of every pixel given two images taken from slightly different angles

Application of CRF: Stereo Vision

- By using some standard preprocessing techniques, one can convert depth estimation into a problem of estimating the disparity y_s between the pixel at location (i_s, j_s) in the left image and the corresponding pixel at location $(i_s + y_s, j_s)$ in the right image.

- We typically assume that corresponding pixels have similar intensity, so we define a local node potential of the form

$$\psi_s(y_s|\boldsymbol{x}) \propto \exp\left\{-\frac{1}{2\sigma^2}\left(x_L(i_s, j_s) - x_R(i_s + y_s, j_s)\right)^2\right\}$$

Left image Right image

Application of CRF: Stereo Vision

- By using some standard preprocessing techniques, one can convert depth estimation into a problem of estimating the disparity y_s between the pixel at location (i_s, j_s) in the left image and the corresponding pixel at location $(i_s + y_s, j_s)$ in the right image

- We typically assume that corresponding pixels have similar intensity, so we define a local node potential of the form

$$\psi_s(y_s|\boldsymbol{x}) \propto \exp\left\{-\frac{1}{2\sigma^2}\left(x_L(i_s, j_s) - x_R(i_s + y_s, j_s)\right)^2\right\}$$

- This equation can be generalized to model the intensity of small windows around each location.
  - In highly textured regions, it is usually possible to find the corresponding patch using cross correlation
  - in regions of low texture, there will be considerable ambiguity

Application of CRF: Stereo Vision

- We can easily add a Gaussian prior on the edges of the MRF that encodes the assumption that neighboring disparities as

$$\psi_{st}(y_s, y_t) \propto \exp\left(-\frac{1}{2\gamma^2}(y_s - y_t)^2\right)$$

- The resulting model is a Gaussian CRF

- However, using Gaussian edge-potentials will over-smooth the estimate.

  - this prior fails to account for the occasional large changes in disparity that occur between neighboring pixels.
  - One gets much better results using a truncated Gaussian potential of the form

$$\psi_{st}(y_s, y_t) \propto \exp\left(-\frac{1}{2\gamma^2}\min\left((y_s - y_t)^2), \delta_0^2\right)\right)$$

    - γ encodes the expected smoothness
    - δ_o encodes the maximum penalty

Application of CRF: Stereo Vision

- We can easily add a Gaussian prior on the edges of the MRF that encodes the assumption that neighboring disparities as

$$\psi_{st}(y_s, y_t) \propto \exp\left(-\frac{1}{2\gamma^2}(y_s - y_t)^2\right)$$

- The resulting model is a Gaussian CRF

- However, using Gaussian edge-potentials will over-smooth the estimate.

  - this prior fails to account for the occasional large changes in disparity that occur between neighboring pixels.
  - One gets much better results using a truncated Gaussian potential of the form

$$\psi_{st}(y_s, y_t) \propto \exp\left(-\frac{1}{2\gamma^2}\min\left((y_s - y_t)^2), \delta_0^2\right)\right)$$

    - This is called a discontinuity preserving potential
    - such penalties are not convex

Application of CRF: Stereo Vision

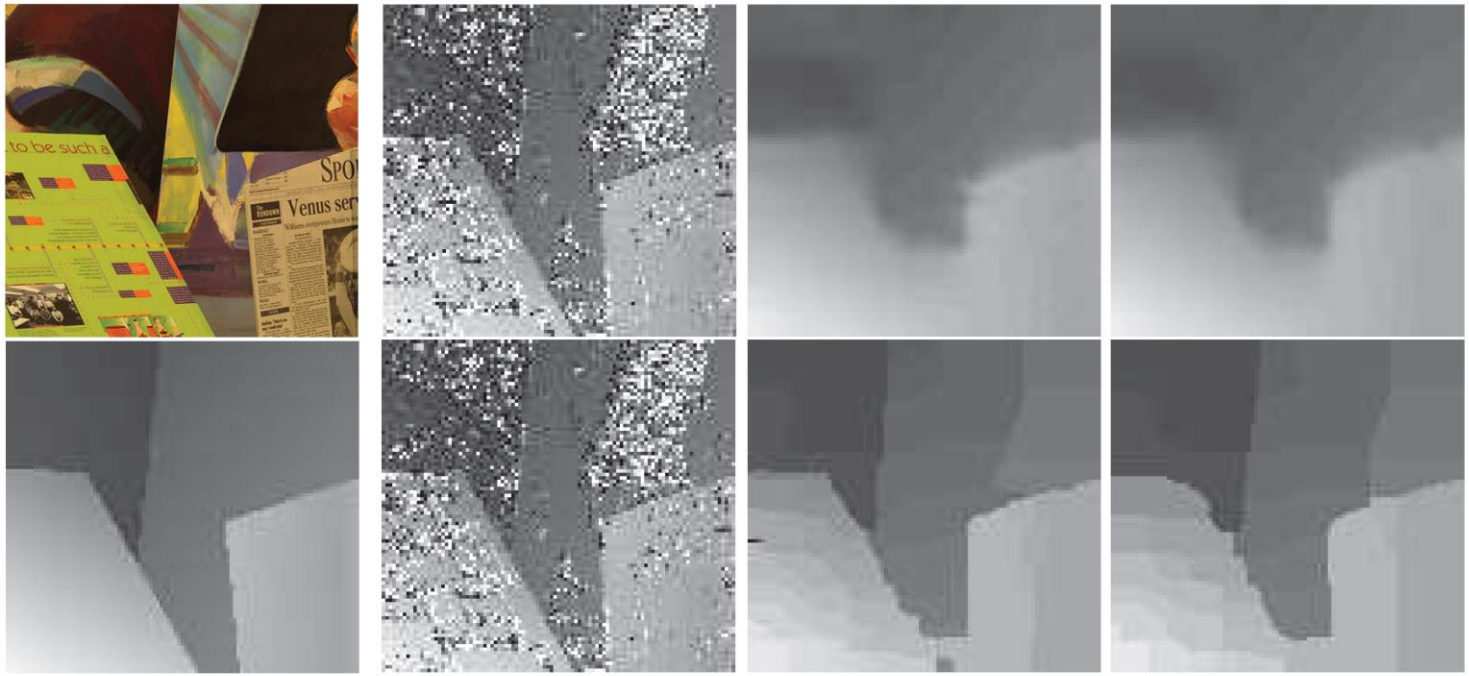


Illustration of belief propagation for stereo depth estimation. Left column: image and true disparities. Remaining columns: initial estimate, estimate after 1 iteration, and estimate at convergence. Top row: Gaussian edge potentials. Bottom row: robust edge potentials

Training CRF

- We can modify the gradient based optimization of MRFs to the CRF case in a straightforward way.

- The scaled log-likelihood becomes

$$l(\mathbf{w}) \triangleq \frac{1}{N}\sum_i \log p(\boldsymbol{y}_i|\mathbf{x}_i,\mathbf{w}) = \frac{1}{N}\sum_i \left[\sum_c \mathbf{w}_c^T \boldsymbol{\phi}_c(\boldsymbol{y}_i,\boldsymbol{x}_i) - \log Z(\mathbf{w},\boldsymbol{x}_i)\right]$$

- The gradient becomes

$$\frac{dl}{d\mathbf{w}_c} = \frac{1}{N}\sum_i \left[\boldsymbol{\phi}_c(\boldsymbol{y}_i,\boldsymbol{x}_i) - \frac{d}{d\mathbf{w}}\log Z(\mathbf{w},\boldsymbol{x}_i)\right]$$
$$= \frac{1}{N}\sum_i \left[\boldsymbol{\phi}_c(\boldsymbol{y}_i,\mathrm{x}_i) - \mathbb{E}[\boldsymbol{\phi}_c(\boldsymbol{y},\boldsymbol{x}_i)]\right]$$

Training CRF

$$\frac{dl}{d\mathbf{w}_c} = \frac{1}{N}\sum_i \left[\boldsymbol{\phi}_c(\boldsymbol{y}_i, \boldsymbol{x}_i) - \frac{d}{d\mathbf{w}} \log Z(\mathbf{w}, \boldsymbol{x}_i)\right]$$

$$= \frac{1}{N}\sum_i \left[\boldsymbol{\phi}_c(\boldsymbol{y}_i, \boldsymbol{x}_i) - \mathbb{E}[\boldsymbol{\phi}_c(\boldsymbol{y}, \boldsymbol{x}_i)]\right]$$

- Note that we now have to perform inference for every single training case inside each gradient step.
- As a consequence, training in CRF is $O(N)$ times slower than MRFs.
- This is because the partition function depends on the inputs x_i.

Training CRF

- In most applications of CRFs (and some applications of MRFs), the size of the graph structure can vary.
- Hence we need to use parameter tying to ensure we can define a distribution of arbitrary size.
- In the pairwise case, we can write the model as follows

$$p(\boldsymbol{y}|\mathbf{x},\mathbf{w}) = \frac{1}{Z(\mathbf{x},\mathbf{w})}\exp\left(\mathbf{w}^T\boldsymbol{\phi}(\boldsymbol{y},\mathbf{x})\right)$$

where $\mathbf{w} = [\mathbf{w}_n, \mathbf{w}_e]$ and

$$\phi(\boldsymbol{y},\mathbf{x}) \triangleq \left[\sum_t \phi_t(y_t,\mathbf{x}), \sum_{s\sim t} \phi_{st}(y_s,y_t,\mathbf{x})\right]$$

Summed node feature

Summed edge feature

Training CRF

- In most applications of CRFs (and some applications of MRFs), the size of the graph structure can vary.
- Hence we need to use parameter tying to ensure we can define a distribution of arbitrary size.
- In the pairwise case, we can write the model as follows

$$p(\boldsymbol{y}|\mathbf{x},\mathbf{w}) = \frac{1}{Z(\mathbf{x},\mathbf{w})}\exp\left(\mathbf{w}^T\boldsymbol{\phi}(\boldsymbol{y},\mathbf{x})\right)$$

where $\mathbf{w} = [\mathbf{w}_n, \mathbf{w}_e]$ and

$$\phi(\boldsymbol{y},\mathbf{x}) \triangleq \left[\sum_t \phi_t(y_t,\mathbf{x}), \sum_{s\sim t}\phi_{st}(y_s,y_t,\mathbf{x})\right]$$

- The gradient expression is modified to handle this case

Training CRF

- In practice, it is important to use a prior/ regularization to prevent overfitting
- If we use a Gaussian prior, the new objective becomes

$$l'(\mathbf{w}) \triangleq \frac{1}{N}\sum_i \log p(y_i|x_i,\mathbf{w}) - \lambda\|\mathbf{w}\|_2^2$$

- The gradient is modified accordingly.
- We can also use $\mathbb{l}_1/\mathbb{l}_2$ regularization
- For example, we could use $\mathbb{l}_1$ for the edge weights to learn a sparse graph structure and $\mathbb{l}_2$ for the node weights

$$l'(\mathbf{w}) \triangleq \frac{1}{N}\sum_i \log p(y_i|x_i,\mathbf{w}) - \lambda_1\|\mathbf{w}_e\| - \lambda_2\|\mathbf{w}_n\|_2^2$$

Edge weight

Node weight

Training CRF

$$l'(\mathbf{w}) \triangleq \frac{1}{N}\sum_i \log p(y_i|x_i,\mathbf{w}) - \lambda_1\|\mathbf{w}_e\| - \lambda_2\|\mathbf{w}_n\|_2^2$$

- Unfortunately, the optimization algorithms are more complicated when we use $\mathbb{l}_1$.
- However, the problem is still convex.
- For large dataset, it is preferable to use stochastic gradient descent (SGD).
- It is possible (and useful) to define CRFs with hidden variables
  - for example to allow for an unknown alignment between the visible features and the hidden labels
- In this case, the objective function is no longer convex
  - Nevertheless, we can find a locally optimal ML or MAP parameter estimate using EM and/ or gradient methods

Structural SVM

- Training a CRF requires inference, in order to compute the expected sufficient statistics needed to evaluate the gradient
- For certain models, computing a joint MAP estimate of the states is provably simpler than computing marginals.
- We discuss a way to train structured output classifiers that that leverages the existence of fast MAP solvers
- These methods are known as structural support vector machines or SSVMs (Tsochantaridis et al. 2005).
- There is also a very similar class of methods known as max margin Markov networks or M3nets (Taskar et al. 2003).

- Tsochantaridis, I., T. Joachims, T. Hofmann, and Y. Altun (2005, September). Large margin methods for structured and interdependent output variables. *J. of Machine Learning Research 6*, 1453–1484.
- Taskar, B., C. Guestrin, and D. Koller (2003). Max-margin markov networks. In *NIPS*.

SSVMs: A Probabilistic View

- In general, we mostly concentrate on fitting models using MAP parameter estimation

$$R_{MAP}(\mathbf{w}) = -\log p(\mathbf{w}) - \sum_{i=1}^{N} \log p(y_i|x_i, \mathbf{w})$$

- However, at test time, we pick the label so as to minimize the posterior expected loss

$$\hat{y}(\mathbf{x}|\mathbf{w}) = \underset{\hat{y}}{\operatorname{argmin}} \sum_{\boldsymbol{y}} L(\hat{\boldsymbol{y}}, \boldsymbol{y}) p(\boldsymbol{y}_i|x_i, \mathbf{w})$$

Loss

SSVMs: A Probabilistic View

- In general, we mostly concentrate on fitting models using MAP parameter estimation

$$R_{MAP}(\mathbf{w}) = -\log p(\mathbf{w}) - \sum_{i=1}^{N} \log p(y_i|x_i, \mathbf{w})$$

- However, at test time, we pick the label so as to minimize the posterior expected loss

$$\hat{y}(\mathbf{x}|\mathbf{w}) = \underset{\hat{\boldsymbol{y}}}{\operatorname{argmin}} \sum_{\boldsymbol{y}} L(\hat{\boldsymbol{y}}, \boldsymbol{y}) p(\boldsymbol{y}_i|x_i, \mathbf{w})$$

- It therefore seems reasonable to take the loss function into account when performing parameter estimation

SSVMs: A Probabilistic View

- However, this violates the fundamental Bayesian distinction between inference and decision making
- Performing these tasks separately will only result in an optimal decision if we can compute the exact posterior
- In most cases, this is intractable, so we need to perform loss-calibrated inference (Lacoste-Julien et al. 2011).
- However, loss-calibrated inference is computationally demanding.
- We just perform loss-calibrated MAP parameter estimation, which is computationally simpler. (Stoyanov et al. 2011).

- Lacoste-Julien, S., F. Sha, and M. I. Jordan (2009). DiscLDA: Discriminative learning for dimensionality reduction and classification. In *NIPS*.
- Stoyanov, V., A. Ropson, and J. Eisner (2011). Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *AI/Statistics*.

SSVMs: A Probabilistic View

$$R_{MAP}(\mathbf{w}) = -\log p(\mathbf{w}) - \sum_{i=1}^{N} \log p(y_i|x_i, \mathbf{w})$$

$$\hat{y}(\mathbf{x}|\mathbf{w}) = \underset{\hat{\boldsymbol{y}}}{\operatorname{argmin}} \sum_{\boldsymbol{y}} L(\hat{\boldsymbol{y}}, \boldsymbol{y}) p(\boldsymbol{y}|\mathbf{x}, \mathbf{w})$$

- Following (Yuille and He 2011), let us instead minimize the posterior expected loss on the training set

$$R_{EL}(\mathbf{w}) \triangleq -\log p(\mathbf{w}) + \sum_{i=1}^{N} \log \sum_{\boldsymbol{y}} L(y_i, \boldsymbol{y}) \boldsymbol{p}(\boldsymbol{y}|x_i, \mathbf{w})$$

- We can write our model in the following form:

$$p(\boldsymbol{y}|\mathbf{x}, \mathbf{w}) = \frac{\exp\left(\mathbf{w}^T \phi(\mathbf{x}, \boldsymbol{y})\right)}{Z(\mathbf{x}, \mathbf{w})}$$

$$p(\mathbf{w}) = \frac{\exp\left(-E(\mathbf{w})\right)}{Z}$$

  - Yuille, A. L. and X. He (2011). Probabilistic models of vision and maxmargin methods. *Frontiers of Electrical and Electronic Engineering 7* (1).

SSVMs: A Probabilistic View

$$p(\boldsymbol{y}|\mathbf{x},\mathbf{w}) = \frac{\exp\left(\mathbf{w}^T\phi(\mathbf{x},\boldsymbol{y})\right)}{Z(\mathbf{x},\mathbf{w})}$$

$$p(\mathbf{w}) = \frac{\exp\left(-E(\mathbf{w})\right)}{Z}$$

$$L(y_i, y) = \exp\left(\tilde{L}(y_i, y)\right)$$

$$R_{EL} = -\log p(\mathbf{w}) + \sum_i \log\left[\sum_y \exp\left(\tilde{L}(y_i, y)\right)\frac{\exp\left(\mathbf{w}^T\phi(\mathbf{x},\boldsymbol{y})\right)}{Z(\mathbf{x},\mathbf{w})}\right]$$

$$= E(\mathbf{w}) + \sum_i -\log Z(x_i,\mathbf{w}) + \log\sum_y \exp\left(\tilde{L}(y_i, y) + \mathbf{w}^T\phi(\mathrm{x}_i,\boldsymbol{y})\right)$$

- Next, we discuss various bounds to simplify the problem.

SSVMs: A Probabilistic View

- For any function $f(\boldsymbol{y})$,

$$\max_{y \in \mathcal{Y}} f(y) \leq \log \sum_{y \in \mathcal{Y}} \exp[f(y)] \leq \log \left[|\mathcal{Y}| \exp \left(\max_{y} f(\boldsymbol{y}) \right) \right]$$
$$= \log|\mathcal{Y}| + \max_{y \in \mathcal{Y}} f(y)$$

- We can ignore the $\log|\mathcal{Y}|$, which is independent of $\boldsymbol{y}$ and treat $\max_{y \in \mathcal{Y}} f(y)$ as both the lower and upper bounds:

$$R_{EL} \sim E(\mathbf{w}) + \sum_{i=1}^{N} \left[\max_{\boldsymbol{y}} \{ \tilde{L}(y_i, \boldsymbol{y}) + \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \} - \max_{\boldsymbol{y}} \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \right]$$

SSVMs: A Probabilistic View

$$\max_{y \in \mathcal{Y}} f(y) \leq \log \sum_{y \in \mathcal{Y}} \exp[f(y)] \leq \log \left[|\mathcal{Y}| \exp \left(\max_{y} f(\boldsymbol{y}) \right) \right]$$
$$= \log|\mathcal{Y}| + \max_{y \in \mathcal{Y}} f(y)$$

$$R_{EL} = E(\mathbf{w}) + \sum_{i} -\log Z(x_i, \mathbf{w}) + \log \sum_{y} \exp\left(\tilde{L}(y_i, y) + \mathbf{w}^T \phi(\mathrm{x}_i, \boldsymbol{y}) \right)$$

$$R_{EL} \sim E(\mathbf{w}) + \sum_{i=1}^{N} \left[\max_{y} \{ \tilde{L}(y_i, \boldsymbol{y}) + \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \} - \max_{y} \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \right]$$

SSVMs: A Probabilistic View

$$R_{EL} \sim E(\mathbf{w}) + \sum_{i=1}^{N} \left[\max_{\boldsymbol{y}} \{ \tilde{L}(y_i, \boldsymbol{y}) + \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \} - \max_{\boldsymbol{y}} \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \right]$$

- Unfortunately, this objective is not convex in $\mathbf{w}$.
- However, we can devise a convex upper bound by exploiting the following looser lower bound on the log-sum-exp function

$$f(y') \leq \log \sum_{y} \exp\big(f(y)\big), \qquad y' \in \mathcal{Y}$$

- Applying this to REL:

$$R_{EL} \leq E(\mathbf{w}) + \sum_{i=1}^{N} \left[\max_{\boldsymbol{y}} \{ \tilde{L}(y_i, \boldsymbol{y}) + \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \} - \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \right]$$

SSVMs: A Probabilistic View

$$R_{EL} \sim E(\mathbf{w}) + \sum_{i=1}^{N} \left[\max_{\boldsymbol{y}} \{ \tilde{L}(y_i, \boldsymbol{y}) + \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \} - \max_{\boldsymbol{y}} \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \right]$$

$$\max_{y \in \mathcal{Y}} f(y) \leq \log \sum_{y \in \mathcal{Y}} \exp[f(y)] \leq \log \left[|\mathcal{Y}| \exp\left(\max_{y} f(\boldsymbol{y}) \right) \right]$$

$$= \log|\mathcal{Y}| + \max_{y \in \mathcal{Y}} f(y)$$

$$R_{EL} \sim E(\mathbf{w}) + \sum_{i=1}^{N} \left[\max_{\boldsymbol{y}} \{ \tilde{L}(y_i, \boldsymbol{y}) + \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \} - \log \sum \exp\left(\mathbf{w}^T \phi(x_i, \boldsymbol{y}) \right) \right]$$

SSVMs: A Probabilistic View

$$R_{EL} \sim E(\mathbf{w}) + \sum_{i=1}^{N} \left[\max_{\boldsymbol{y}} \{ \tilde{L}(y_i, \boldsymbol{y}) + \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \} - \log \sum \exp\left(\mathbf{w}^T \phi(x_i, \boldsymbol{y}) \right) \right]$$

$$f(y') \leq \log \sum_{y} \exp\left(f(y) \right), \qquad y' \in \mathcal{Y}$$

$$R_{EL} \leq E(\mathbf{w}) + \sum_{i=1}^{N} \left[\max_{\boldsymbol{y}} \{ \tilde{L}(y_i, \boldsymbol{y}) + \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \} - \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \right]$$

SSVMs: A Probabilistic View

- Applying this to REL:

$$R_{EL} \leq E(\mathbf{w}) + \sum_{i=1}^{N} \left[\max_{\boldsymbol{y}} \{ \tilde{L}(y_i, \boldsymbol{y}) + \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \} - \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \right]$$

- We set $E(\mathbf{w}) = -\frac{1}{2C} \|\mathbf{w}\|_2^2$ (corresponding to a spherical Gaussian prior)

$$R_{SSVM}(\mathbf{w}) \triangleq -\frac{1}{2C} \|\mathbf{w}\|^2 + \sum_{i=1}^{N} \left[\max_{\boldsymbol{y}} \{ \tilde{L}(y_i, \boldsymbol{y}) + \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \} - \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \right]$$

- This is the same objective as used in the SSVM approach of (Tsochantaridis et al. 2005).

- Tsochantaridis, I., T. Joachims, T. Hofmann, and Y. Altun (2005, September). Large margin methods for structured and interdependent output variables. *J. of Machine Learning Research 6*, 1453–1484.

SSVMs: A Probabilistic View

- SSVM criterion can be seen as optimizing an upper bound on the Bayesian objective.
- This bound will be tight when $\|\mathbf{w}\|$ is large.
- Unfortunately, a large $\|\mathbf{w}\|$ corresponds to a model that is likely to overfit. So, it is unlikely that we will be working in this regime (because we will tune the strength of the regularizer to avoid this situation)
- An alternative justification for the SVM criterion is that it focusses effort on fitting parameters that affect the decision boundary
  - This is a better use of computational resources than fitting the full distribution, especially when the model is wrong

SSVMs: A Non-Probabilistic View

- We now present SSVMs in a more traditional (non-probabilistic) way, following (Tsochantaridis et al. 2005)
- Let $f(\mathbf{x}; \mathbf{w}) = \max_{\boldsymbol{y} \in \mathcal{Y}} \mathbf{w}^T \phi(\mathbf{x}, \boldsymbol{y})$ be the prediction function
- We can obtain zero loss on the training set using this predictor if

$$\forall i. \max_{\boldsymbol{y} \in \mathcal{Y} \setminus y_i} \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \leq \mathbf{w}^T \phi(x_i, y_i)$$

- Each one of these nonlinear inequalities can be equivalently replaced by $|\mathcal{Y}| - 1$ linear inequalities
- This results in total $N(|\mathcal{Y}| - 1)$ linear constraints:

$$\forall i. \forall y \in \mathcal{Y} \setminus \boldsymbol{y}_i, \mathbf{w}^T \phi(x_i, y_i) - \mathbf{w}^T \phi(x_i, \boldsymbol{y}) \geq 0$$

- Considering $\delta_i \triangleq \phi(x_i, y_i) - \phi(x_i, \boldsymbol{y})$:

$$\mathbf{w}^T \delta_i \geq 0$$

- Tsochantaridis, I., T. Joachims, T. Hofmann, and Y. Altun (2005, September). Large margin methods for structured and interdependent output variables. *J. of Machine Learning Research* 6, 1453–1484.

SSVMs: A Non-Probabilistic View

- If we can achieve zero loss, there will typically be multiple solution vectors $\mathbf{w}^T$.

- We pick the one that maximizes the margin, defined as

$$\gamma \triangleq \min_i f(\mathbf{x}, y_i; \mathbf{w}) - \max_i f(\mathbf{x}, y'; \mathbf{w})$$

- The margin can be made arbitrarily large by rescaling $\mathbf{w}$. So, we fix its norm to be 1,

$$\max_{\mathbf{w}:\|\mathbf{w}\|=1} \gamma \qquad \text{s.t. } \forall i. \forall \boldsymbol{y} \in \mathcal{Y} \backslash y_i. \;\; \mathbf{w}^T \delta_i \geq \gamma$$

- Equivalently, we can write

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{s.t. } \forall i. \forall \boldsymbol{y} \in \mathcal{Y} \backslash y_i. \;\; \mathbf{w}^T \delta_i \geq 1$$

SSVMs: A Non-Probabilistic View

- To allow for the case where zero loss cannot be achieved (equivalent to the data being inseparable in the case of binary classification), we introduce slack variables:

$$\min_{\mathbf{w},\xi} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i \qquad \text{s.t. } \forall i.\, \forall \boldsymbol{y} \in \mathcal{Y}\backslash y_i.\ \ \mathbf{w}^T\delta_i \geq 1 - \xi_i,$$

$$\xi_i \geq 0$$

- In the case of structured outputs, we don't want to treat all constraint violations equally.

- One way to achieve this is to divide the slack variable by the size of the loss (this is called slack re-scaling)

SSVMs: A Non-Probabilistic View

- One way to achieve this is to divide the slack variable by the size of the loss (this is called slack re-scaling)

$$\min_{\mathbf{w},\xi} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i \qquad \text{s.t. } \forall i.\, \forall \boldsymbol{y} \in \mathcal{Y}\backslash y_i.\ \ \mathbf{w}^T\delta_i \geq 1 - \frac{\xi_i}{L(y_i, \boldsymbol{y})},\ \ \xi_i \geq 0$$

- Alternatively, we can define the margin to be proportional to the loss (this is called margin re-rescaling)

$$\min_{\mathbf{w},\xi} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i \qquad \text{s.t. } \forall i.\, \forall \boldsymbol{y} \in \mathcal{Y}\backslash y_i.\ \ \mathbf{w}^T\delta_i \geq L(y_i, \boldsymbol{y}) - \xi_i,\ \ \xi_i \geq 0$$

- In fact, we can write $\forall \boldsymbol{y} \in \mathcal{Y}$ instead of $\forall \boldsymbol{y} \in \mathcal{Y}\backslash y_i$.
  - since if $y = y_i$, $\mathbf{w}^T\delta_i = 0$ and $\xi = 0$. This latter approach is used in M3nets

SSVMs: A Non-Probabilistic View

- We can solve for the ξ_i^* terms as follows

$$\xi_i^*(\mathbf{w}) = \max\left\{0, \max_y (L(y_i, \boldsymbol{y}) - \mathbf{w}^T \delta_i)\right\} = \max_y (L(y_i, \boldsymbol{y}) - \mathbf{w}^T \delta_i)$$

- Substituting in, and dropping the constraints, we get the following equivalent problem

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_i \max_y \left(L(y_i, \boldsymbol{y}) + \mathbf{w}^T \phi(x_i, y) - \mathbf{w}^T \phi(x_i, y_i)\right)$$

Empirical Risk Minimization

- In the frequentist approach to machine learning, the goal is to minimize the regularized empirical risk

$$R(\mathbf{w}) + \frac{C}{N}\sum_{i=1}^{N} L\big(\boldsymbol{y}_i, f(\mathbf{x}_i, \mathbf{w})\big)$$

where $R(\mathbf{w})$ is the regularizer and $f(\mathbf{x}_i, \mathbf{w}) = \max_y \mathbf{w}^T \phi(x_i, y) = \hat{y}_i$ is the prediction.

- However, this objective is hard to optimize, because the loss is not differentiable.

- Hence, we will construct a convex upper bound instead.

Empirical Risk Minimization

- Hence, we will construct a convex upper bound instead.

$$R(\mathbf{w}) + \frac{C}{N}\sum_i \max_y (L(y_i, y) - \mathbf{w}^T \delta_i)$$

- Although the above objectives are simple quadratic programs (QP), they have $O(N|\mathcal{Y}|)$ constraints.

- This is intractable, since $\mathcal{Y}$ is usually exponentially large

- In the case of the margin rescaling formulation, it is possible to reduce the exponential number of constraints to a polynomial number, provided the loss function and the feature vector decompose according to a graphical model.

- This is the approach used in M3nets (Taskar et al. 2003).

- Taskar, B., C. Guestrin, and D. Koller (2003). Max-margin markov networks. In *NIPS*.

Empirical Risk Minimization

- An alternative approach is to work directly with the exponentially sized QP
- This allows for the use of more general loss functions.
- There are several possible methods to make this feasible.
- One is to use cutting plane methods.
- Another is to use stochastic sub-gradient methods.

Cutting plane methods for fitting SSVMs

- This method can handle general loss functions, and is implemented in the popular SVMstruct package (http://svmlight.joachims.org/svm_struct.html).

- The method is based on the cutting plane method from convex optimization (Kelley 1960)

- The method proceeds as

  - We start with an initial guess $\mathbf{w}$ and no constraints.
  - At each iteration, we find the most violated constraint involving $\mathbf{x}_i$ and $\hat{y}_i$.
  - Compute loss-augmented margin violation and check if it exceeds the current value ξ_i by more than ϵ.
  - If yes, add $\hat{y}_i$ to the working set of constraints for this training case.
  - Solve the resulting new QP to find the new $\mathbf{w}$ and $\boldsymbol{\xi}$.

- Kelley, J. E. (1960). The cutting-plane method for solving convex programs. *J. of the Soc. for Industrial and Applied Math. 8*, 703–712.

Cutting Plane Methods for Fitting SSVMs

Algorithm 19.3: Cutting plane algorithm for SSVMs (margin rescaling, N-slack version)

Input $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_n)\}$, C, ϵ ;
$\mathcal{W}_i = \emptyset$, $\xi_i = 0$ for $i = 1 : N$;
repeat
  for $i = 1 : N$ **do**
    $\hat{\mathbf{y}}_i = \text{argmax}_{\hat{\mathbf{y}}_i \in \mathcal{Y}} \, L(\mathbf{y}_i, \mathbf{y}) - \mathbf{w}^T \boldsymbol{\delta}_i(\hat{\mathbf{y}}_i)$;
    if $L(\mathbf{y}_i, \mathbf{y}) - \mathbf{w}^T \boldsymbol{\delta}_i(\hat{\mathbf{y}}_i) > \xi_i + \epsilon$ **then**
      $\mathcal{W}_i = \mathcal{W}_i \cup \{\hat{\mathbf{y}}_i\}$;
      $(\mathbf{w}, \boldsymbol{\xi}) = \text{argmin}_{\mathbf{w}, \boldsymbol{\xi} \geq \mathbf{0}} \frac{1}{2}||\mathbf{w}||_2^2 + C \sum_{i=1}^N \xi_i$;
        s.t. $\forall i = 1 : N, \forall \mathbf{y}' \in \mathcal{W}_i : \mathbf{w}^T \boldsymbol{\delta}_i(\hat{\mathbf{y}}_i) \geq L(\mathbf{y}_i, \mathbf{y}') - \xi_i$;
until *no* $\mathcal{W}_i$ *has changed*;
Return $(\mathbf{w}, \boldsymbol{\xi})$

Cutting Plane Methods for Fitting SSVMs

- The key to efficiency is the ability to find the most violated constraint.
- We call this process loss-augmented decoding
- If the loss function has an additive decomposition of the same form as the features
  - we can fold the loss into the weight vector, i.e., we can find a new set of parameters $\mathbf{w}'$ such that

$$(\mathbf{w}')^T \delta_i = (\mathbf{w})^T \delta_i$$

  - We can then use a standard decoding algorithm, such as Viterbi, on the model $p(\boldsymbol{y}|\mathbf{x}, \mathbf{w}')$.

A Linear Time Algorithm

- Although the above algorithm takes polynomial time, we can do better, and devise an algorithm that runs in linear time, assuming we use a linear kernel

- The basic idea is to have a single slack variable, instead of N, but to use $|\mathcal{Y}|^N$ constraints, instead of $N|\mathcal{Y}|$.

- Specifically, we optimize

$$\max_{\mathbf{w},\xi \geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + C\xi$$

$$s.t. \; \forall (\bar{y}_1, \ldots, \bar{y}_N) \in \mathcal{Y}^N : \frac{1}{N}\mathbf{w}^T \sum_{i=1}^{N} \delta_i(\bar{y}_i) \geq \frac{1}{N}\sum_{i=1}^{N} L(y_i, \bar{y}_i) - \xi$$

- The cutting plane algorithm, with slight modification, can be used to solve this problem

Online Algorithms for Fitting SSVMS

- Although the cutting plane algorithm can be made to run in time linear in the number of data points, that can still be slow if we have a large dataset.
- In such cases, it is preferable to use online learning
- One popular algorithm is the structured perceptron algorithm
  - At each step, we compute $\hat{\boldsymbol{y}} = \max p(\boldsymbol{y}|\mathbf{x})$ for the current sample **x.**
  - If $\hat{\boldsymbol{y}} = \boldsymbol{y}$, we do nothing, otherwise we update the weight vector

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \phi(\boldsymbol{y}, \mathbf{x}) - \phi(\hat{\boldsymbol{y}}, \mathbf{x})$$

Online Algorithms for Fitting SSVMS

- The disadvantage of the structured perceptron algorithm is that it implicitly assumes 0-1 loss, and it does not enforce any kind of margin.
- An alternative approach is to perform stochastic sub-gradient descent
- We start by considering the objective function

$$f(\mathbf{w}) = \sum_i \max_{\hat{\boldsymbol{y}}_i} [L(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i) + \mathbf{w}^T \phi(\mathbf{x}_i, \hat{\boldsymbol{y}}_i)] - \phi(\mathbf{x}_i, \boldsymbol{y}_i) + \lambda \|\mathbf{w}\|^2$$

- Letting $\hat{\boldsymbol{y}}_i$ to be the argmax of this max, the subgradient is

$$g(\mathbf{w}) = \sum_i \phi(\mathbf{x}_i, \hat{\boldsymbol{y}}_i) - \phi(\mathbf{x}_i, \boldsymbol{y}_i) + 2\lambda \mathbf{w}$$

Online Algorithms for Fitting SSVMS

- Letting $\hat{\boldsymbol{y}}_i$ to be the argmax of this max, the subgradient is

$$g(\mathbf{w}) = \sum_i \phi(\mathbf{x}_i, \hat{\boldsymbol{y}}_i) - \phi(\mathbf{x}_i, \boldsymbol{y}_i) + 2\lambda \mathbf{w}$$

- In stochastic subgradient descent, we approximate this gradient with a single term and then perform an update

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_k g_i(\mathbf{w}_k) = \mathbf{w}_k - \eta_k \left[\phi(\mathbf{x}_i, \hat{\boldsymbol{y}}_i) - \phi(\mathbf{x}_i, \boldsymbol{y}_i) + \frac{2}{N} \lambda \mathbf{w} \right]$$

η_k is the step-size parameters and should satisfy the Robbins-Monro conditions