Machine Learning – Lecture 18

Inference & Applications

12.07.2016

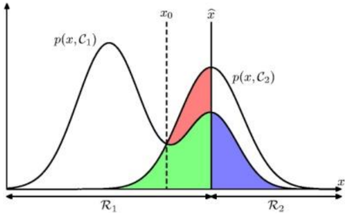
Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de

leibe@vision.rwth-aachen.de

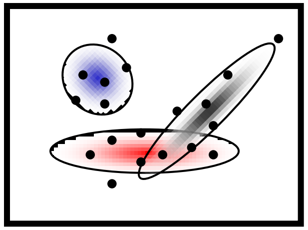
Announcements

- **Lecture evaluation**
  - Please fill out the evaluation forms…

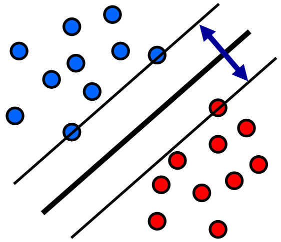
Course Outline

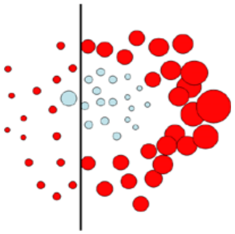


- **Fundamentals (2 weeks)**
  - Bayes Decision Theory
  - Probability Density Estimation

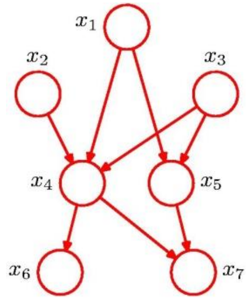


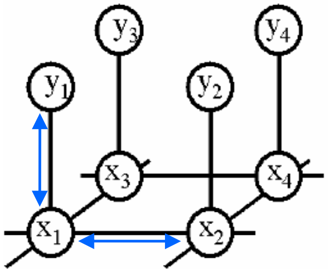
- **Discriminative Approaches (5 weeks)**
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees





- **Generative Models (4 weeks)**
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference
  - Applications

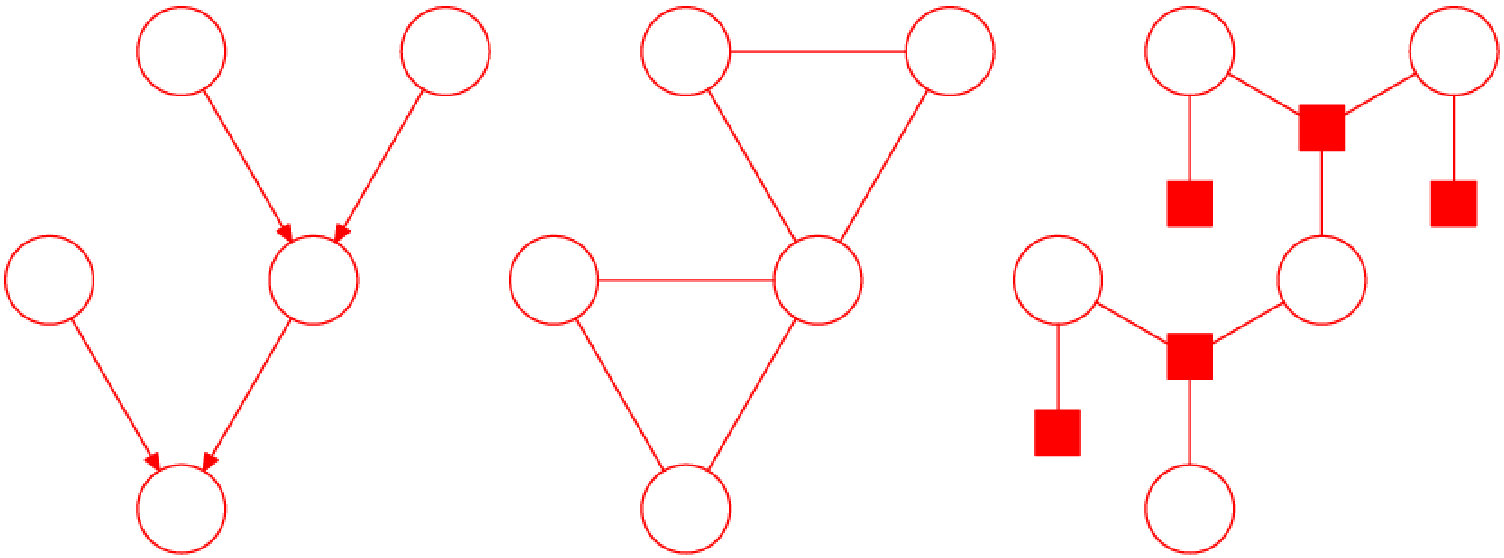




Topics of This Lecture

- Recap: Exact inference
  - Sum-Product algorithm
  - Max-Sum algorithm
  - Junction Tree algorithm
- Applications of Markov Random Fields
  - Application examples from computer vision
  - Interpretation of clique potentials
  - Unary potentials
  - Pairwise potentials
- Solving MRFs with Graph Cuts
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications

Recap: Factor Graphs



- Joint probability
  - Can be expressed as product of factors: $p(\mathbf{x}) = \frac{1}{Z} \prod_s f_s(\mathbf{x}_s)$
  - Factor graphs make this explicit through separate factor nodes.
- Converting a directed polytree
  - Conversion to undirected tree creates loops due to moralization!
  - Conversion to a factor graph again results in a tree!

Image source: C. Bishop, 2006

Recap: Sum-Product Algorithm

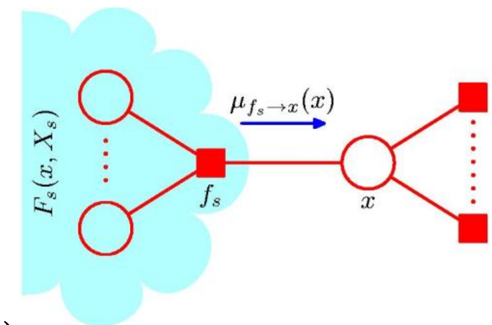
- Objectives
  - Efficient, exact inference algorithm for finding marginals.
- Procedure:
  - Pick an arbitrary node as root.
  - Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
  - Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
  - Compute the product of received messages at each node for which the marginal is required, and normalize if necessary.

$$p(x) \propto \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x)$$

- Computational effort
  - Total number of messages = 2 · number of graph edges.

Recap: Sum-Product Algorithm

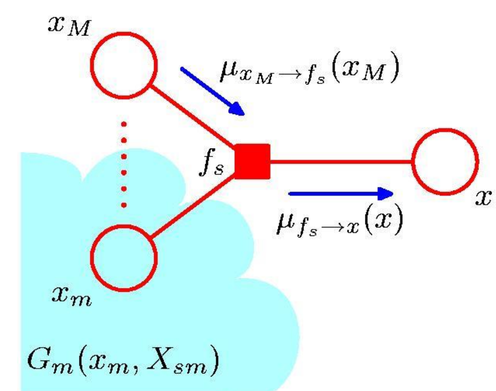
- Two kinds of messages
  - Message from factor node to variable nodes:
    - Sum of factor contributions



$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s)$$

$$= \sum_{X_s} f_s(\mathbf{x}_s) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

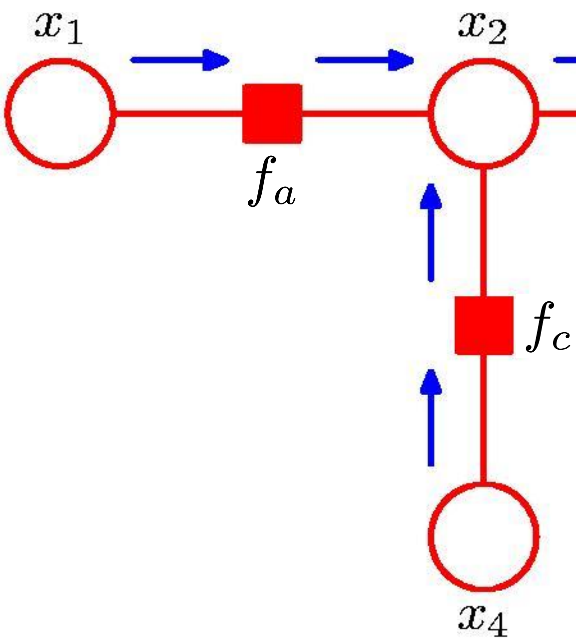
  - Message from variable node to factor node:
    - Product of incoming messages



$$\mu_{x_m \to f_s}(x_m) \equiv \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

$\Rightarrow$ Simple propagation scheme.

Recap: Sum-Product from Leaves to Root

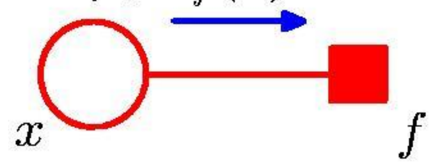


Message definitions:

$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} f_s(\mathbf{x}_s) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

$$\mu_{x_m \to f_s}(x_m) \equiv \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

$$\mu_{x \to f}(x) = 1$$



$$\mu_{f \to x}(x) = f(x)$$

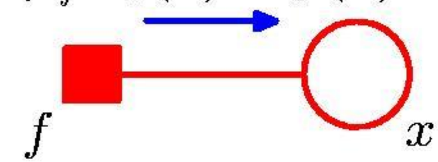
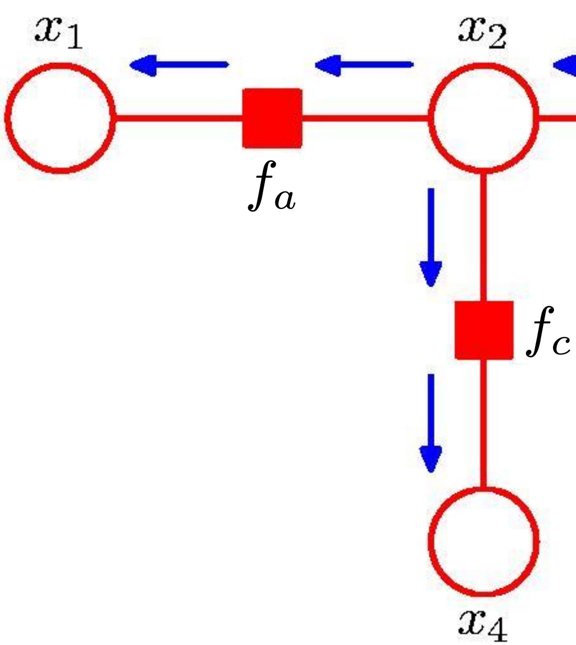


Image source: C. Bishop, 2006

Recap: Sum-Product from Root to Leaves

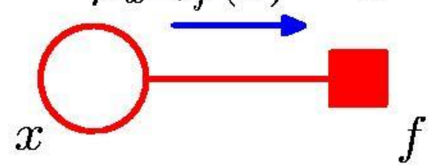


Message definitions:

$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} f_s(\mathbf{x}_s) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

$$\mu_{x_m \to f_s}(x_m) \equiv \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

$$\mu_{x \to f}(x) = 1$$



$$\mu_{f \to x}(x) = f(x)$$

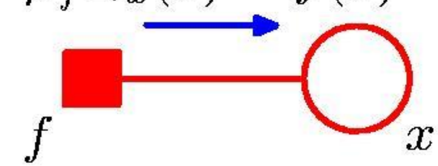


Image source: C. Bishop, 2006

Max-Sum Algorithm

- Objective: an efficient algorithm for finding
  - Value $\mathbf{x}^{\max}$ that maximises $p(\mathbf{x})$;
  - Value of $p(\mathbf{x}^{\max})$.
  - $\Rightarrow$ Application of dynamic programming in graphical models.

- In general, maximum marginals $\neq$ joint maximum.
  - Example:

	$x = 0$	$x = 1$
$y = 0$	0.3	0.4
$y = 1$	0.3	0.0

$$\arg\max_x p(x, y) = 1 \qquad \arg\max_x p(x) = 0$$

Max-Sum Algorithm - Key Ideas

- **Key idea 1: Distributive Law (again)**

$$\max(ab, ac) = a\max(b, c)$$
$$\max(a+b, a+c) = a+\max(b, c)$$

$\Rightarrow$ **Exchange products/summations and max operations exploiting the tree structure of the factor graph.**

- **Key idea 2: Max-Product $\rightarrow$ Max-Sum**
  - **We are interested in the maximum value of the joint distribution**

$$p(\mathbf{x}^{\max}) = \max_{\mathbf{x}} p(\mathbf{x})$$

$\Rightarrow$ **Maximize the product $p(\mathbf{x})$.**

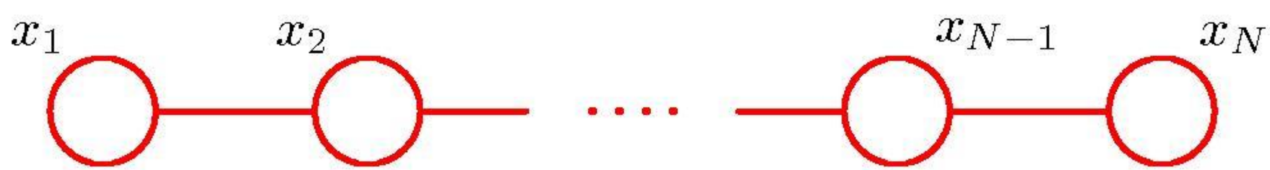
  - **For numerical reasons, use the logarithm.**

$$\ln\left(\max_{\mathbf{x}} p(\mathbf{x})\right) = \max_{\mathbf{x}} \ln p(\mathbf{x}).$$

$\Rightarrow$ **Maximize the sum (of log-probabilities).**

Max-Sum Algorithm

- **Maximizing over a chain (max-product)**



- **Exchange max and product operators**

$$p(\mathbf{x}^{\max}) = \max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1} \dots \max_{x_M} p(\mathbf{x})$$

$$= \frac{1}{Z} \max_{x_1} \cdots \max_{x_N} [\psi_{1,2}(x_1, x_2) \cdots \psi_{N-1,N}(x_{N-1}, x_N)]$$

$$= \frac{1}{Z} \max_{x_1} \left[\max_{x_2} \left[\psi_{1,2}(x_1, x_2) \left[\cdots \max_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right] \right]$$

- **Generalizes to tree-structured factor graph**

$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_n} \prod_{f_s \in \mathrm{ne}(x_n)} \max_{X_s} f_s(x_n, X_s)$$

Max-Sum Algorithm

- **Initialization (leaf nodes)**

$$\mu_{x\to f}(x) = 0 \qquad \mu_{f\to x}(x) = \ln f(x)$$

- **Recursion**
  - Messages

$$\mu_{f\to x}(x) = \max_{x_1,\ldots,x_M}\left[\ln f(x, x_1, \ldots, x_M) + \sum_{m\in\mathrm{ne}(f_s)\setminus x} \mu_{x_m\to f}(x_m)\right]$$

$$\mu_{x\to f}(x) = \sum_{l\in\mathrm{ne}(x)\setminus f} \mu_{f_l\to x}(x)$$

  - For each node, keep a record of which values of the variables gave rise to the maximum state:

$$\phi(x) = \arg\max_{x_1,\ldots,x_M}\left[\ln f(x, x_1, \ldots, x_M) + \sum_{m\in\mathrm{ne}(f_s)\setminus x} \mu_{x_m\to f}(x_m)\right]$$

Max-Sum Algorithm

- **Termination (root node)**
  - Score of maximal configuration

$$p^{\max} = \max_x \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right]$$

  - Value of root node variable giving rise to that maximum

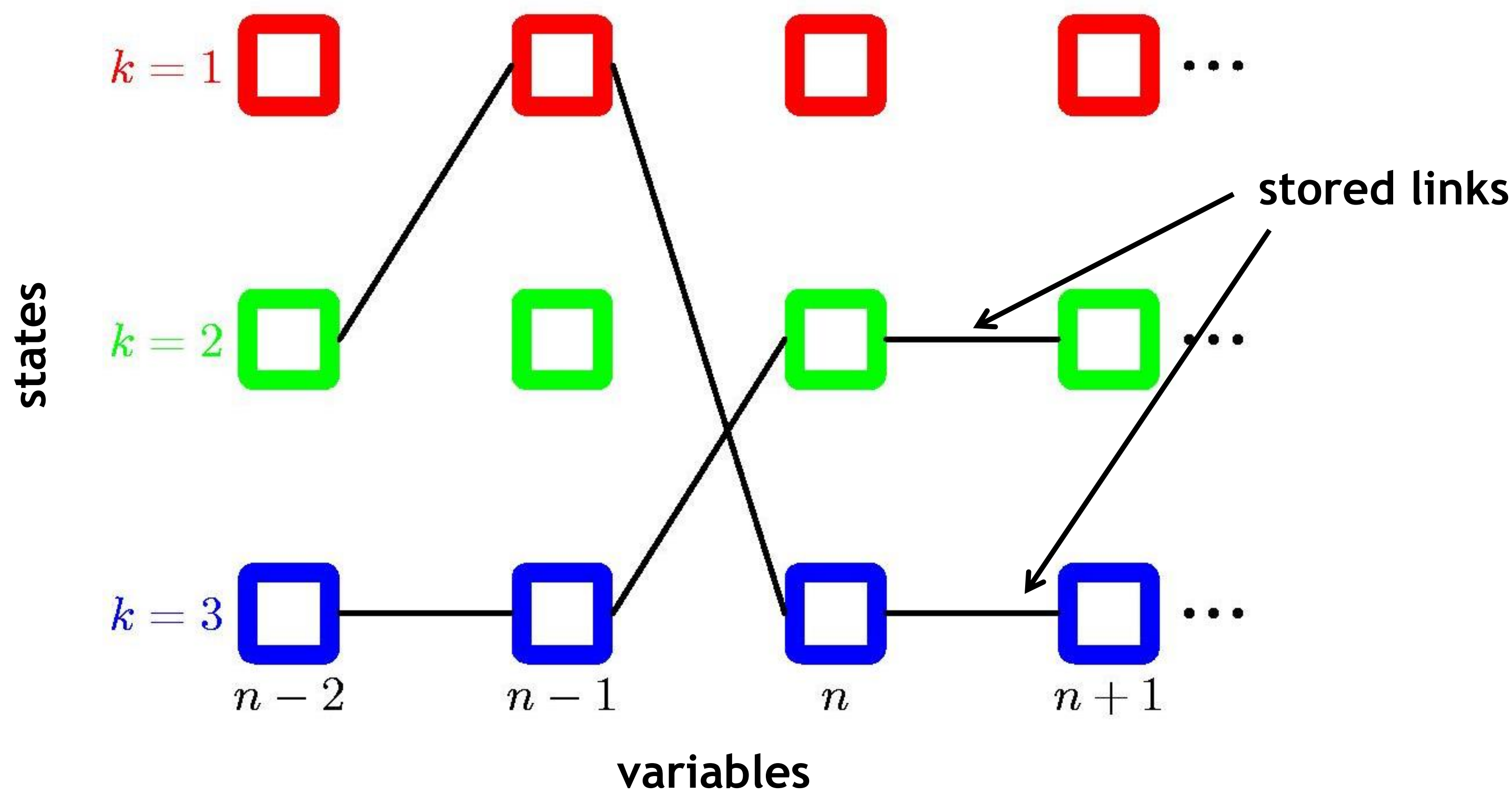
$$x^{\max} = \arg\max_x \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right]$$

  - Back-track to get the remaining variable values

$$x_{n-1}^{\max} = \phi(x_n^{\max})$$

Visualization of the Back-Tracking Procedure

- **Example: Markov chain**



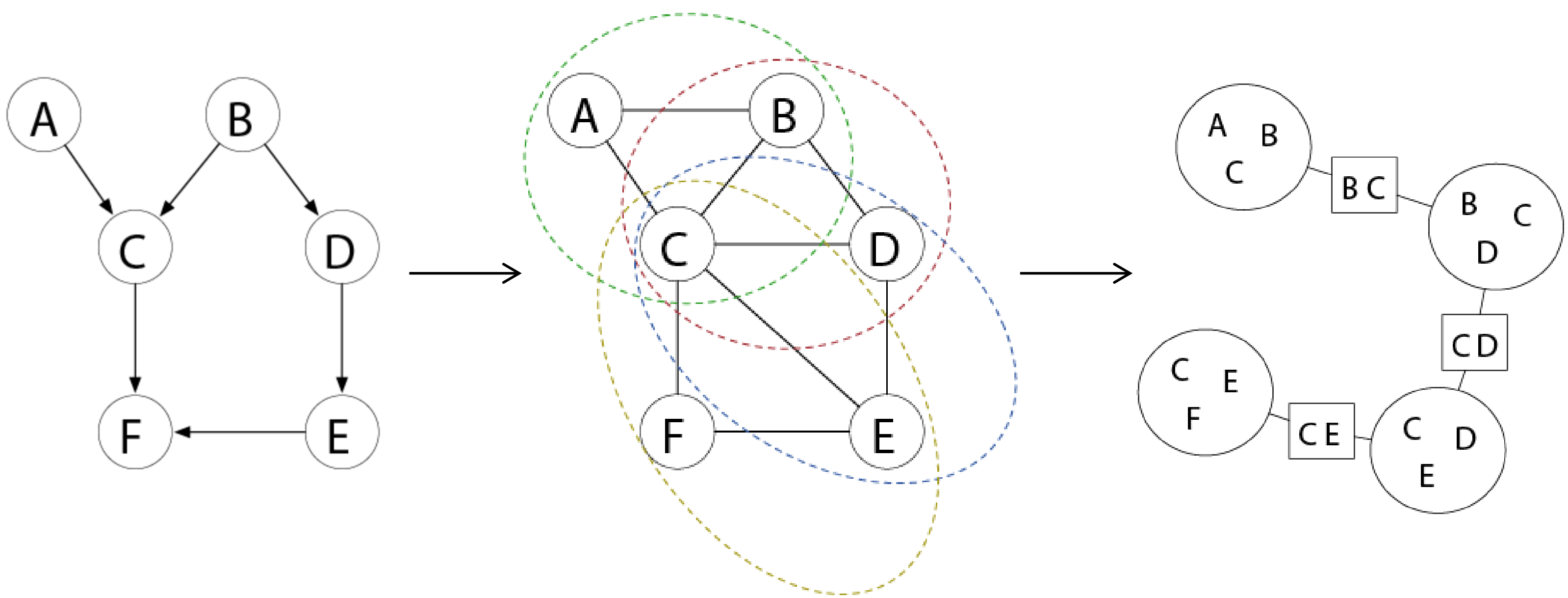
$\Rightarrow$ **Same idea as in Viterbi algorithm for HMMs…**

Topics of This Lecture

- Factor graphs
  - Construction
  - Properties
- Sum-Product Algorithm for computing marginals
  - Key ideas
  - Derivation
  - Example
- Max-Sum Algorithm for finding most probable value
  - Key ideas
  - Derivation
  - Example
- **Algorithms for loopy graphs**
  - **Junction Tree algorithm**
  - **Loopy Belief Propagation**

Junction Tree Algorithm

- Motivation
  - Exact inference on general graphs.
  - Works by turning the initial graph into a junction tree with one node per clique and then running a sum-product-like algorithm.
  - Intractable on graphs with large cliques.

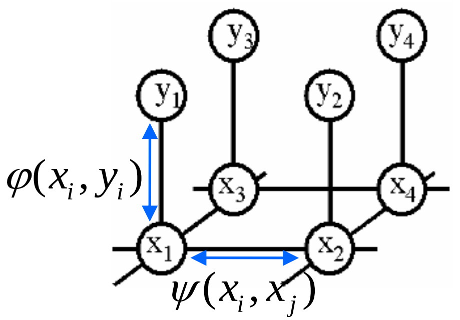


Loopy Belief Propagation

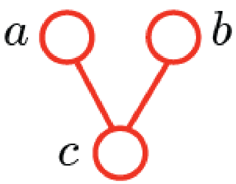
- Alternative algorithm for loopy graphs
  - Sum-Product on general graphs.
  - Strategy: simply ignore the problem.
  - Initial unit messages passed across all links, after which messages are passed around until convergence
    - Convergence is not guaranteed!
    - Typically break off after fixed number of iterations.
  - Approximate but tractable for large graphs.
  - Sometime works well, sometimes not at all.

Topics of This Lecture

- Recap: Exact inference
  - Sum-Product algorithm
  - Max-Sum algorithm
  - Junction Tree algorithm
- **Applications of Markov Random Fields**
  - **Application examples from computer vision**
  - **Interpretation of clique potentials**
  - **Unary potentials**
  - **Pairwise potentials**
- Solving MRFs with Graph Cuts
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications



Markov Random Fields (MRFs)



- What we've learned so far…
  - We know they are undirected graphical models.
  - Their joint probability factorizes into clique potentials,

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C)$$

which are conveniently expressed as energy functions.

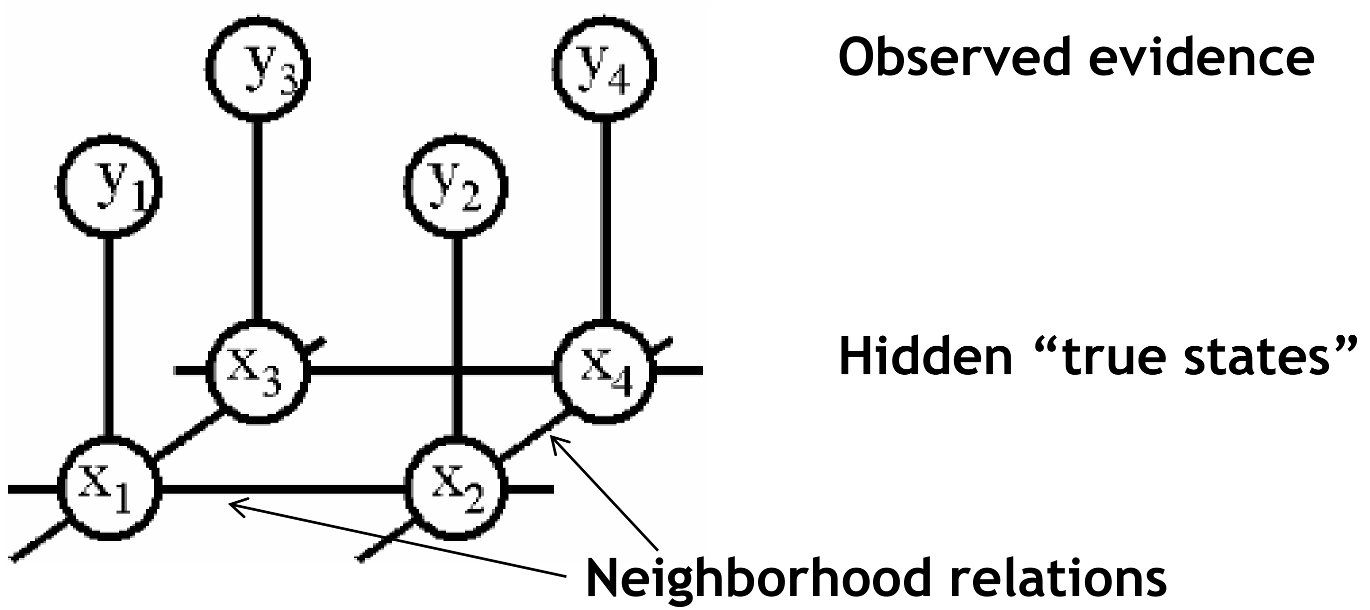
$$\psi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C)\}$$

  - We know how to perform inference for them.
    - Sum/Max-Product BP for exact inference in tree-shaped MRFs.
    - Loopy BP for approximate inference in arbitrary MRFs.
    - Junction Tree algorithm for converting arbitrary MRFs into trees.
- But what are they actually good for?
  - And how do we apply them in practice?

Image source: C. Bishop, 2006

Markov Random Fields

- Allow rich probabilistic models.
  - But built in a local, modular way.
  - Learn local effects, get global effects out.
- Very powerful when applied to regular structures.
  - Such as images...



Applications of MRFs

- **Movie “No Way Out” (1987)**

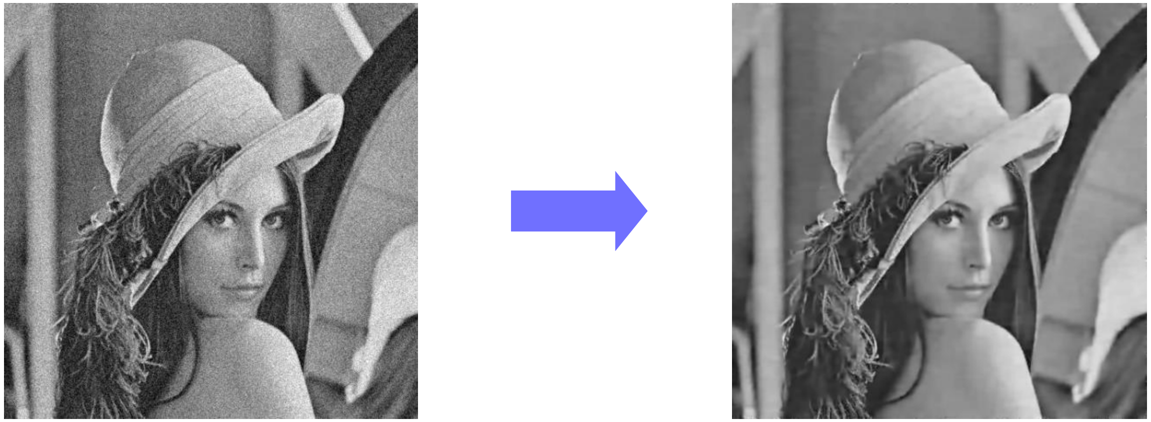
Applications of MRFs

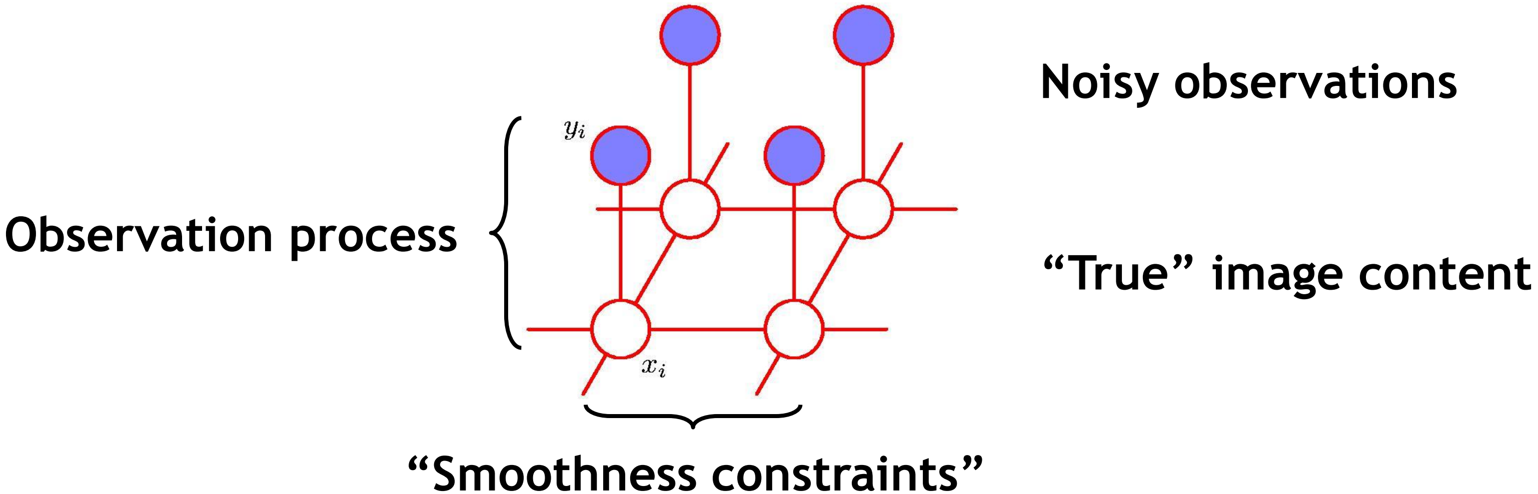
- Many applications for low-level vision tasks
  - Image denoising

Results by [Roth & Black, CVPR'05]

Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising

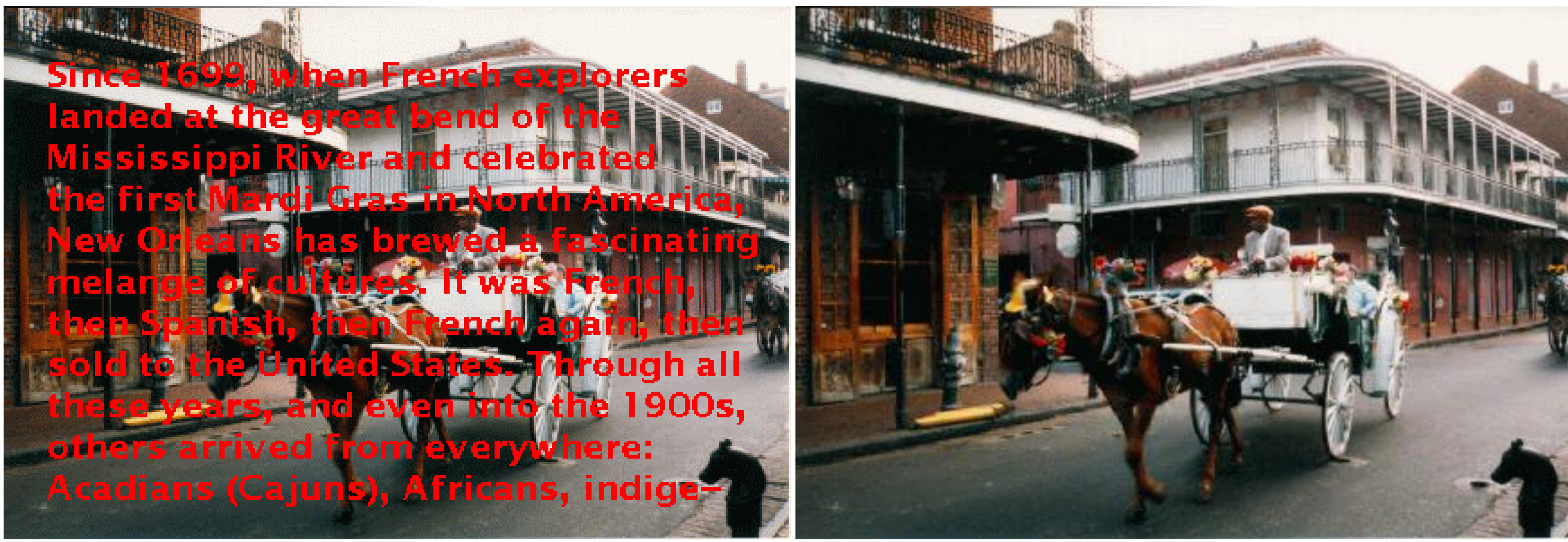




Results by [Roth & Black, CVPR'05]

Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting



Results by [Roth & Black, CVPR'05]

Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration

Results by [Roth & Black, CVPR'05]

Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation

Image source: Pawan M. Kumar

Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution

Convert a low-res image into a high-res image!

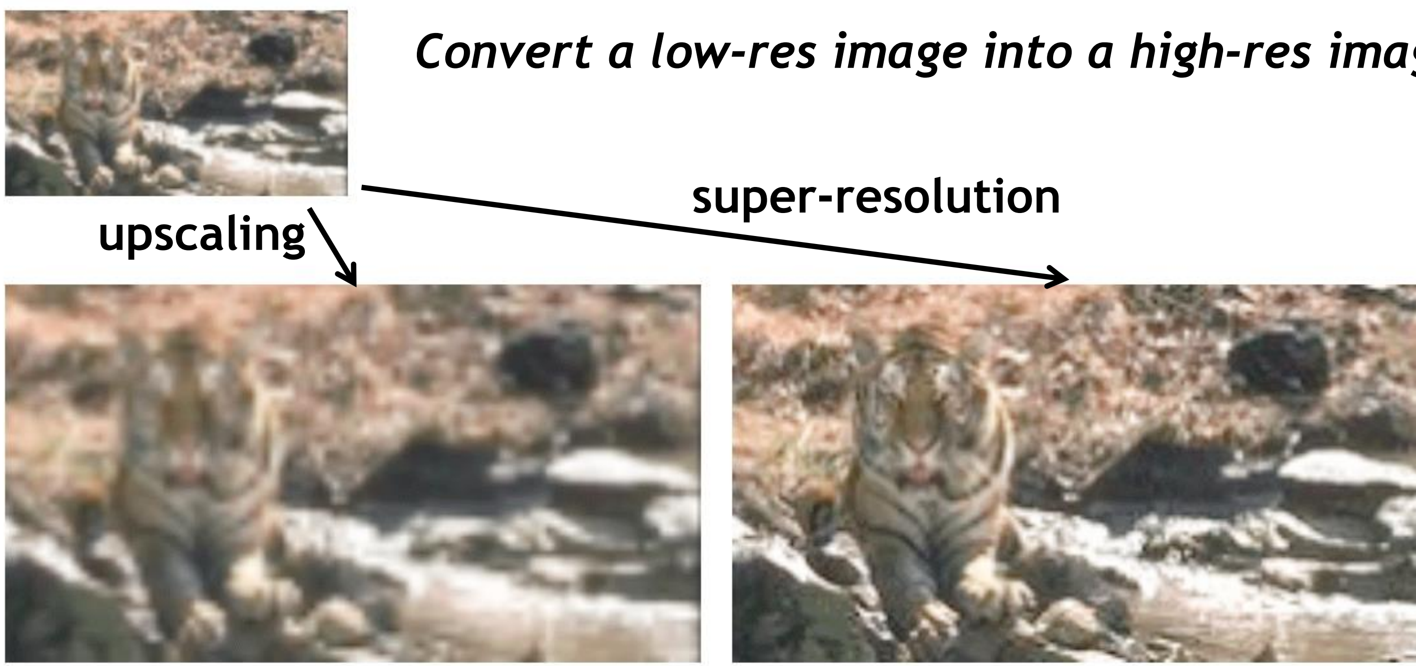


Image source: [Freeman et al., CG&A’03]

Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution

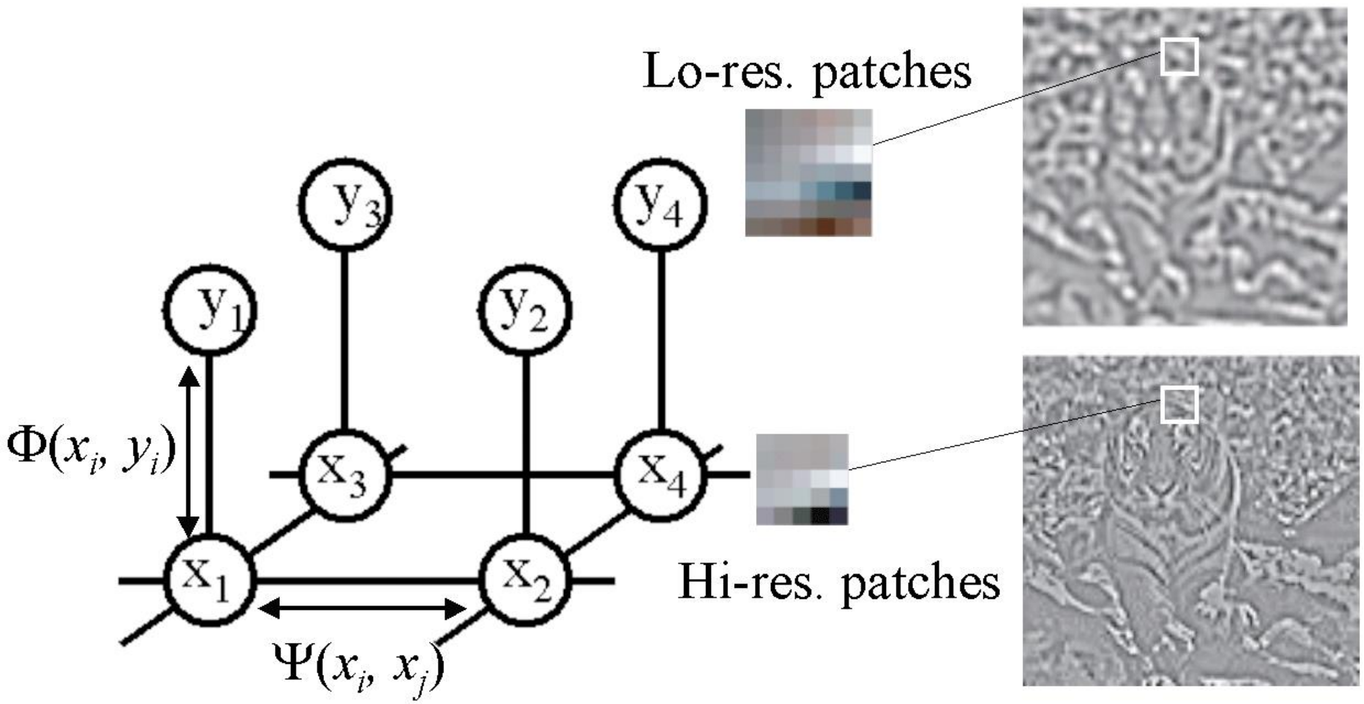


Image source: [Freeman et al., CG&A'03]

Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution
  - Optical flow

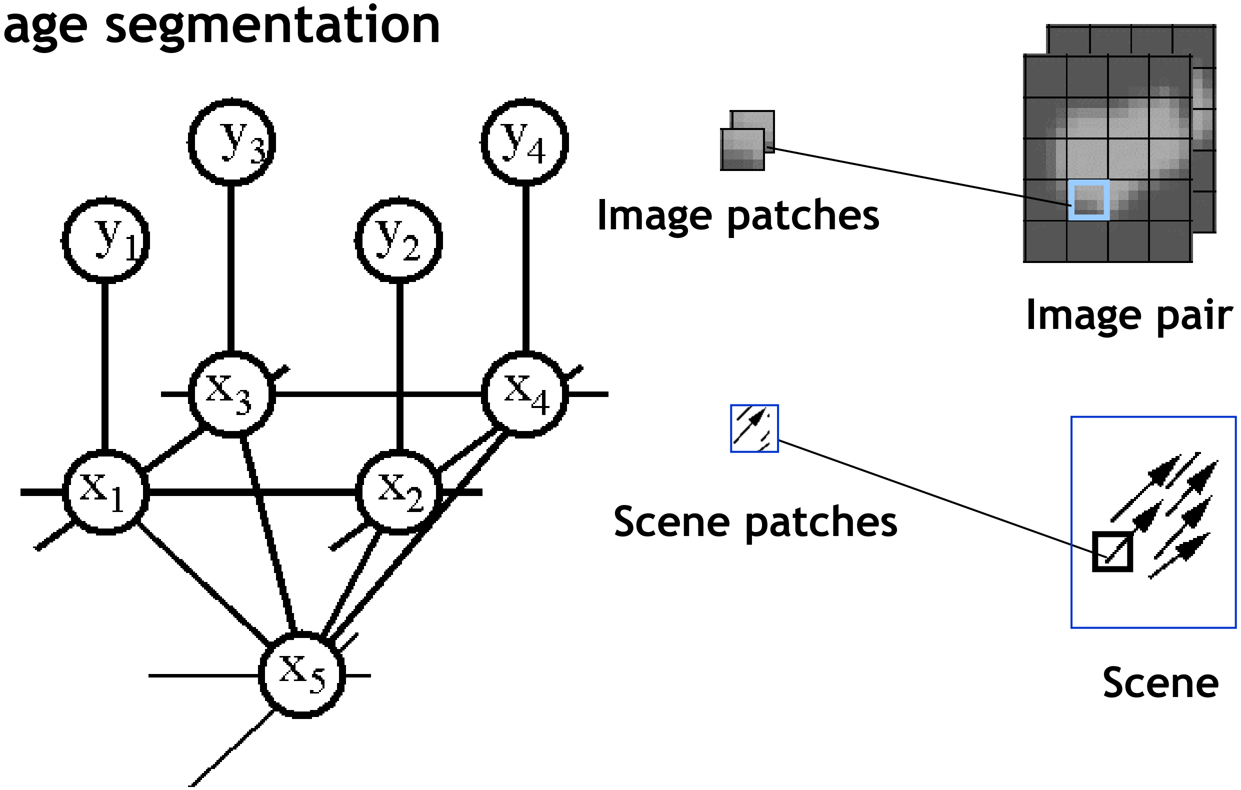
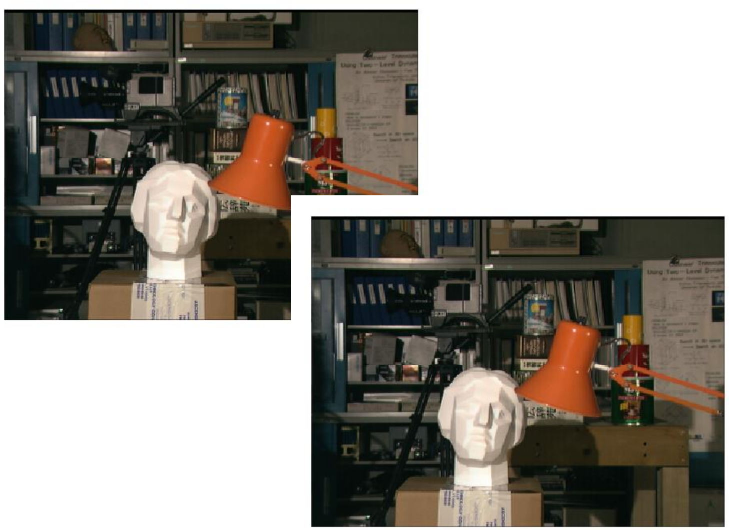


Image source: William Freeman

Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution
  - Optical flow
  - Stereo depth estimation



Stereo image pair

Disparity map

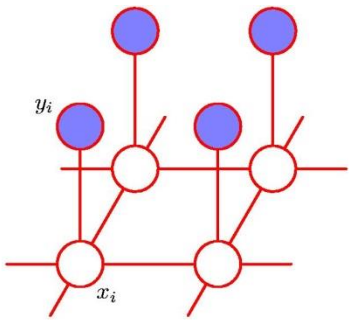
Applications of MRFs

- Many applications for low-level vision tasks
  - Image denoising
  - Inpainting
  - Image restoration
  - Image segmentation
  - Super-resolution
  - Optical flow
  - Stereo depth estimation

- MRFs have become a standard tool for such tasks.
  - Let's look at how they are applied in detail...

MRF Structure for Images

- Basic structure

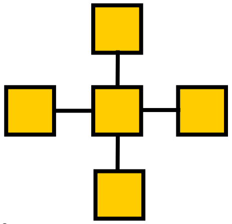


Noisy observations

"True" image content

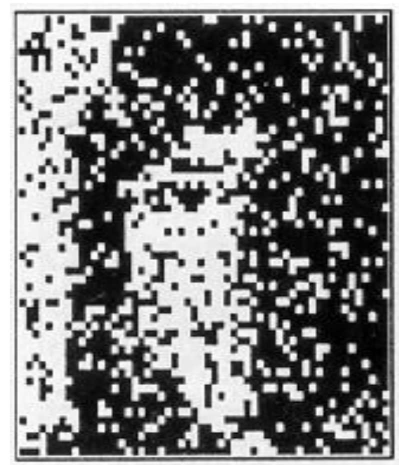
- Two components
  - Observation model
    - How likely is it that node x_i has label L_i given observation y_i?
    - This relationship is usually learned from training data.
  - Neighborhood relations
    - Simplest case: 4-neighborhood
    - Serve as smoothing terms.

    ⇒ Discourage neighboring pixels to have different labels.
    - This can either be learned or be set to fixed "penalties".



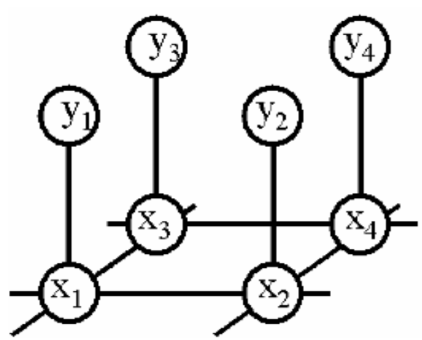
MRF Nodes as Pixels

Original image



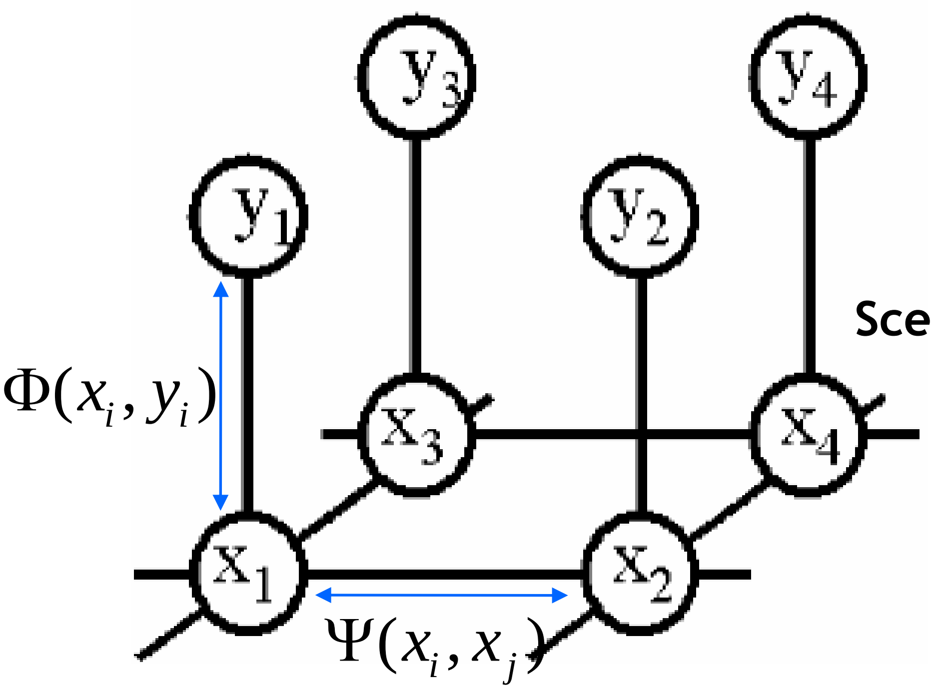
Degraded image

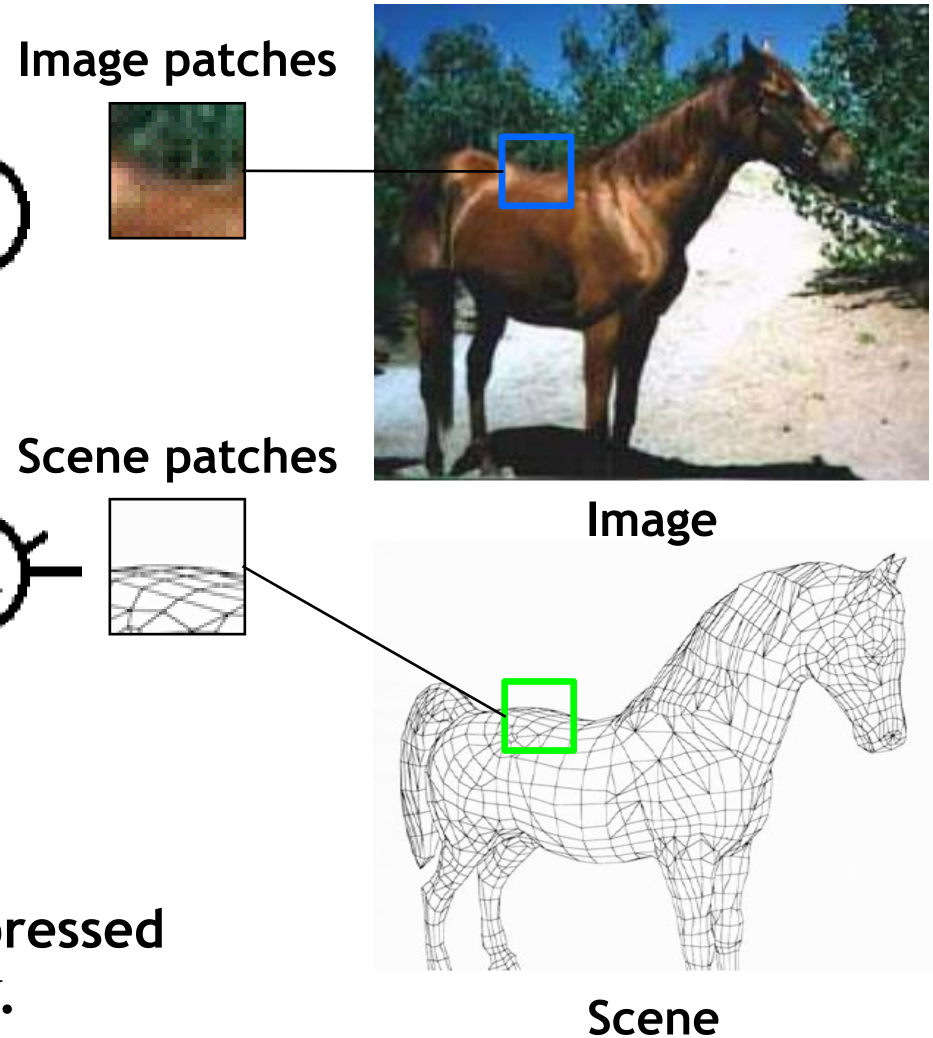
Reconstruction from MRF modeling pixel neighborhood statistics



These neighborhood statistics can be learned from training data!

MRF Nodes as Patches

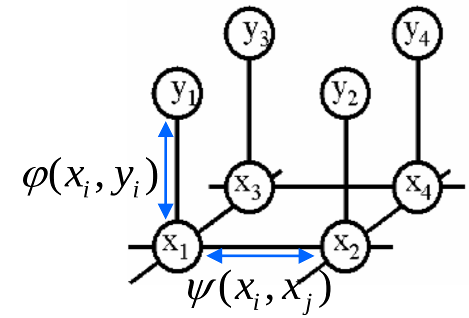




More general relationships expressed by potential functions Φ and Ψ.

Network Joint Probability

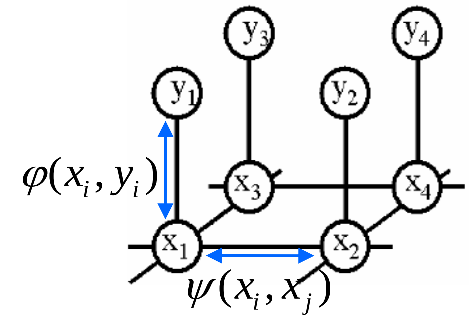
- **Interpretation of the factorized joint probability**



$$P(x, y) = \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j)$$

- x: **Scene**
- y: **Image**
- Φ: **Image-scene compatibility function**
- y_i: **Local observations**
- Ψ: **Scene-scene compatibility function**
- x_i, x_j: **Neighboring scene nodes**

Energy Formulation



- Energy function

$$E(x,y) = \underbrace{\sum_i \varphi(x_i, y_i)}_{\text{Single-node potentials}} + \underbrace{\sum_{i,j} \psi(x_i, x_j)}_{\text{Pairwise potentials}}$$

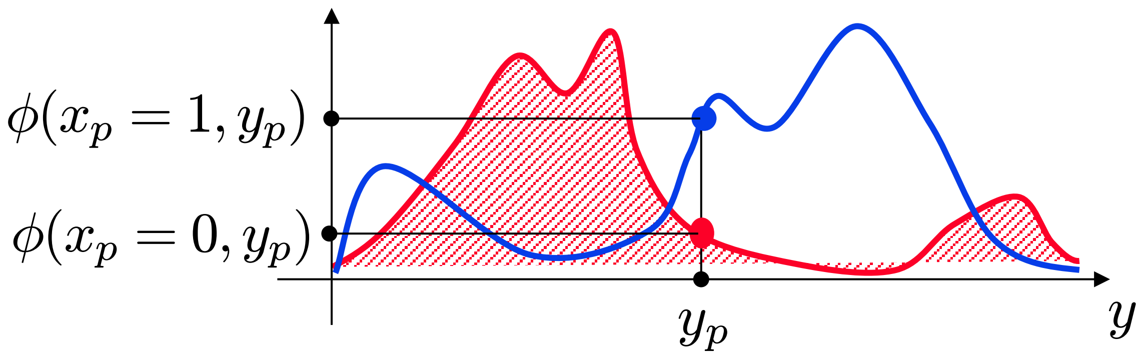
- Single-node (unary) potentials φ
  - Encode local information about the given pixel/patch.
  - How likely is a pixel/patch to belong to a certain class (e.g. foreground/background)?
- Pairwise potentials ψ
  - Encode neighborhood information.
  - How different is a pixel/patch's label from that of its neighbor? (e.g. based on intensity/color/texture difference, edges)

How to Set the Potentials? Some Examples

- **Unary potentials**
  - **E.g., color model, modeled with a Mixture of Gaussians**

$$\phi(x_i, y_i; \theta_\phi) = \log \sum_k \theta_\phi(x_i, k) p(k|x_i) \mathcal{N}(y_i; \bar{y}_k, \Sigma_k)$$

⇒ Learn color distributions for each label



How to Set the Potentials? Some Examples

- Pairwise potentials
  - Potts Model

$$\psi(x_i, x_j; \theta_\psi) = \theta_\psi \delta(x_i \neq x_j)$$

    - Simplest discontinuity preserving model.
    - Discontinuities between any pair of labels are penalized equally.
    - Useful when labels are unordered or number of labels is small.

  - Extension: "contrast sensitive Potts model"

$$\psi(x_i, x_j, g_{ij}(y); \theta_\psi) = \theta_\psi g_{ij}(y) \delta(x_i \neq x_j)$$

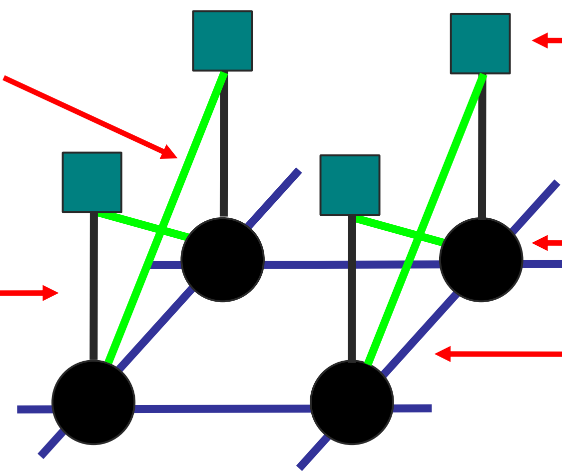
where

$$g_{ij}(y) = e^{-\beta \left\| y_i - y_j \right\|^2} \qquad \beta = 2 \cdot avg\left(\left\| y_i - y_j \right\|^2 \right)$$

    - Discourages label changes except in places where there is also a large change in the observations.

Extension: Conditional Random Fields (CRF)

- Idea: Model conditional instead of joint probability



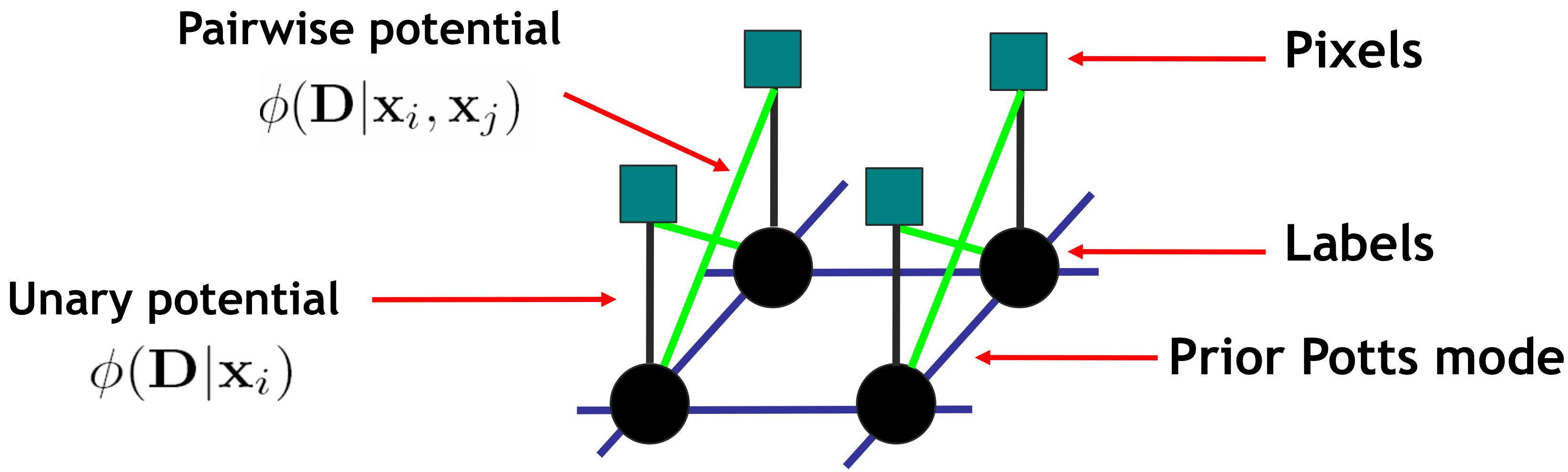
- Energy formulation

$$E(\mathbf{x}) = \sum_{i \in S} \left(\phi(\mathbf{D}|\mathbf{x}_i) + \sum_{j \in N_i} \left(\phi(\mathbf{D}|\mathbf{x}_i, \mathbf{x}_j) + \psi(\mathbf{x}_i, \mathbf{x}_j) \right) \right) + \text{const}$$

Unary likelihood | Contrast Term | Uniform Prior (Potts Model)

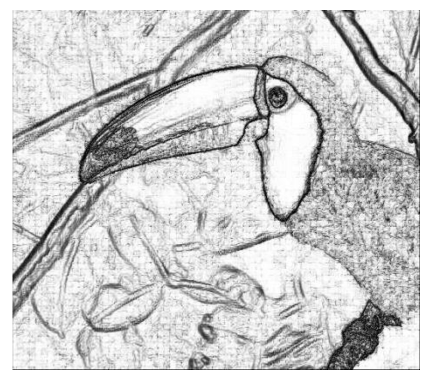
Example: MRF for Image Segmentation

- MRF structure



Data (D)

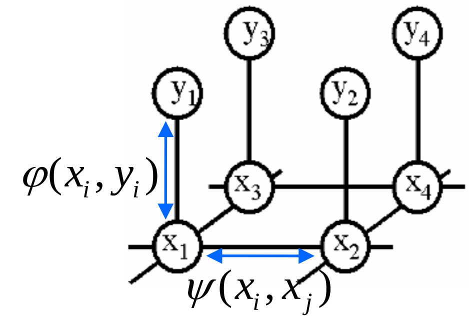
Unary likelihood



Pair-wise Terms

MAP Solution

Energy Minimization

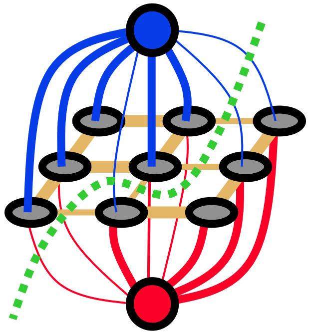


- **Goal:**
  - Infer the optimal labeling of the MRF.
- **Many inference algorithms are available, e.g.**
  - Simulated annealing ← *What you saw in the movie.*
  - Iterated conditional modes (ICM) ← *Too simple.*
  - Belief propagation ← *Last lecture*
  - Graph cuts ← *Use this one!*
  - Variational methods
  - Monte Carlo sampling

  (Variational methods, Monte Carlo sampling) ← *For more complex problems*
- **Recently, Graph Cuts have become a popular tool**
  - Only suitable for a certain class of energy functions.
  - But the solution can be obtained very fast for typical vision problems (~1MPixel/sec).

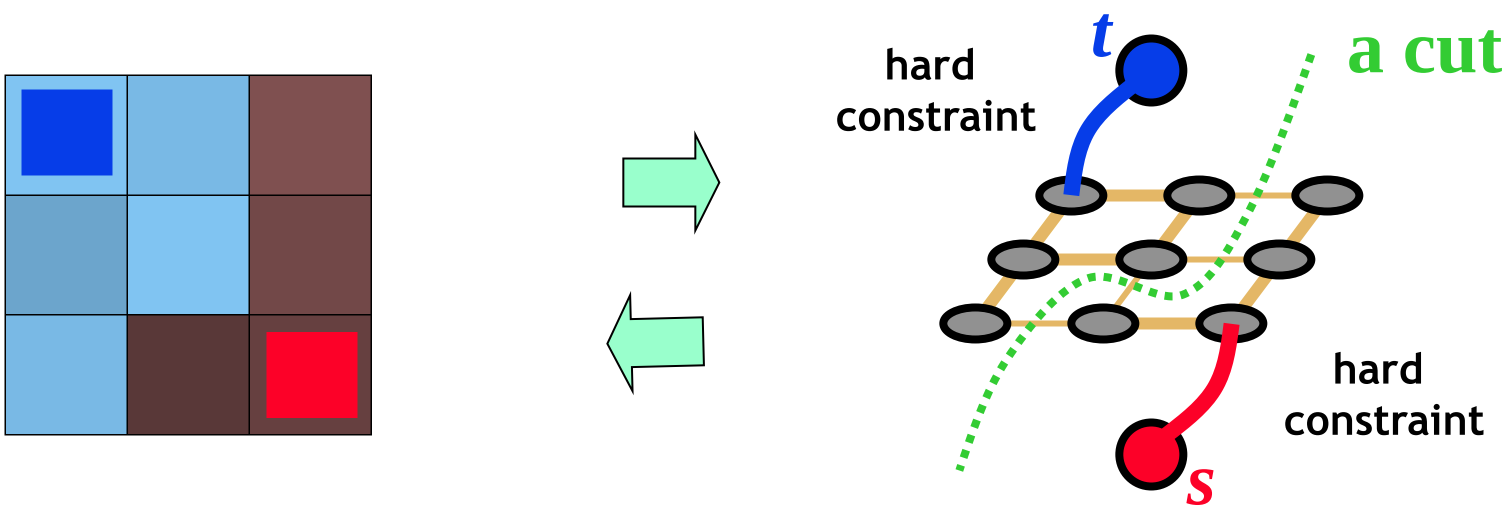
Topics of This Lecture

- Recap: Exact inference
  - Factor Graphs
  - Sum-Product/Max-Sum Belief Propagation
  - Junction Tree algorithm
- Applications of Markov Random Fields
  - Application examples from computer vision
  - Interpretation of clique potentials
  - Unary potentials
  - Pairwise potentials
- **Solving MRFs with Graph Cuts**
  - **Graph cuts for image segmentation**
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications



Graph Cuts for Binary Problems

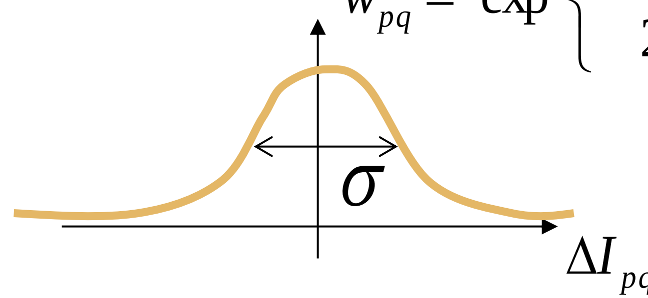
- Idea: convert MRF into source-sink graph



Minimum cost cut can be computed in polynomial time

(max-flow/min-cut algorithms)

$$w_{pq} = \exp\left\{-\frac{\Delta I_{pq}}{2\sigma^2}\right\}$$



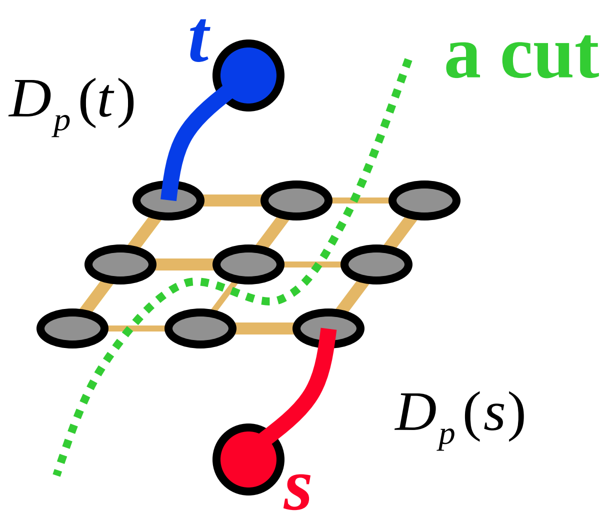
[Boykov & Jolly, ICCV'01]

Simple Example of Energy

unary potentials   pairwise potentials

$$E(L) = \sum_{p} D_p(L_p) + \sum_{pq \in N} w_{pq} \cdot \delta(L_p \neq L_q)$$

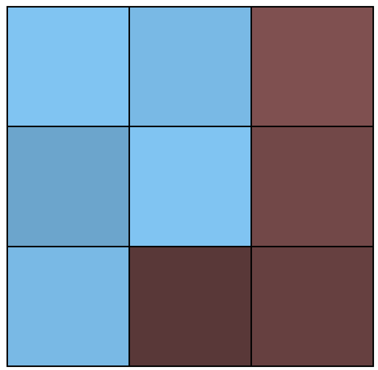
t-links   n-links

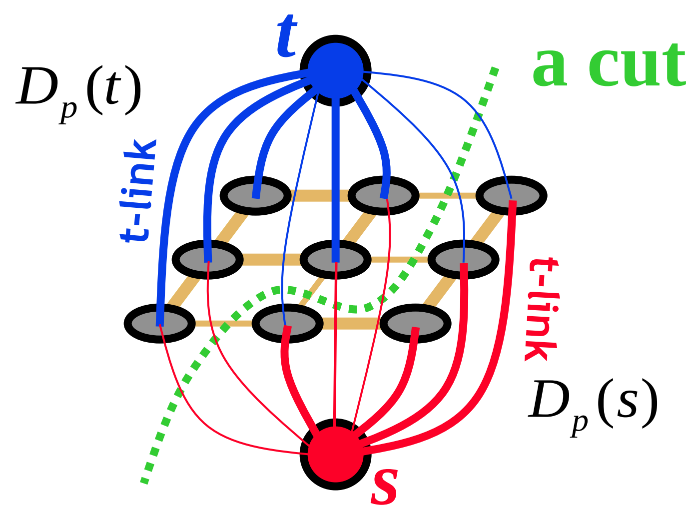


$$L_p \in \{s, t\}$$

(binary object segmentation)

Adding Regional Properties





Regional bias example

Suppose I^s and I^t are given "expected" intensities of object and background

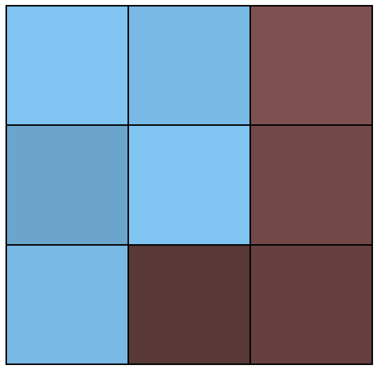
$$D_p(s) \propto \exp\left(-\| I_p - I^s \|^2 / 2\sigma^2\right)$$

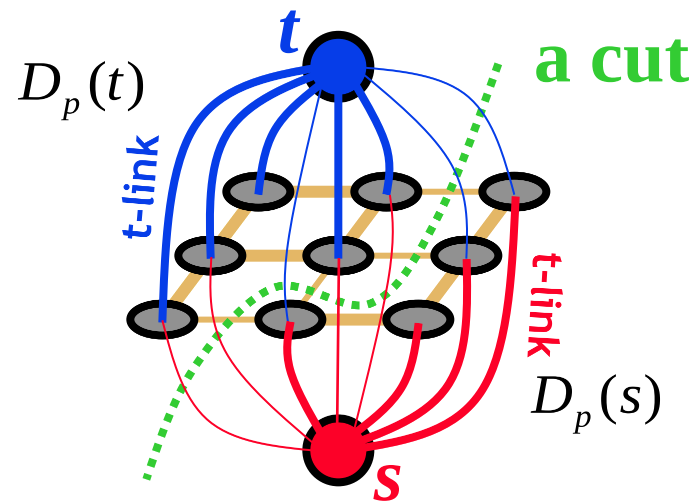
$$D_p(t) \propto \exp\left(-\| I_p - I^t \|^2 / 2\sigma^2\right)$$

NOTE: hard constrains are not required, in general.

[Boykov & Jolly, ICCV'01]

Adding Regional Properties





"expected" intensities of
object and background
I^s and I^t
can be re-estimated

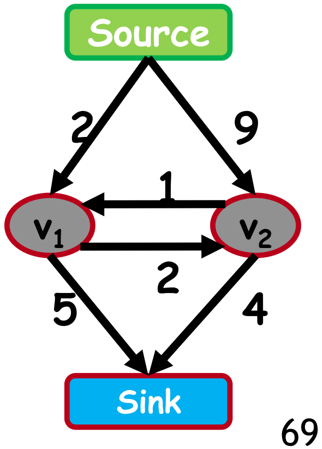
$$D_p(s) \propto \exp\left(-\| I_p - I^s \|^2 / 2\sigma^2\right)$$

$$D_p(t) \propto \exp\left(-\| I_p - I^t \|^2 / 2\sigma^2\right)$$

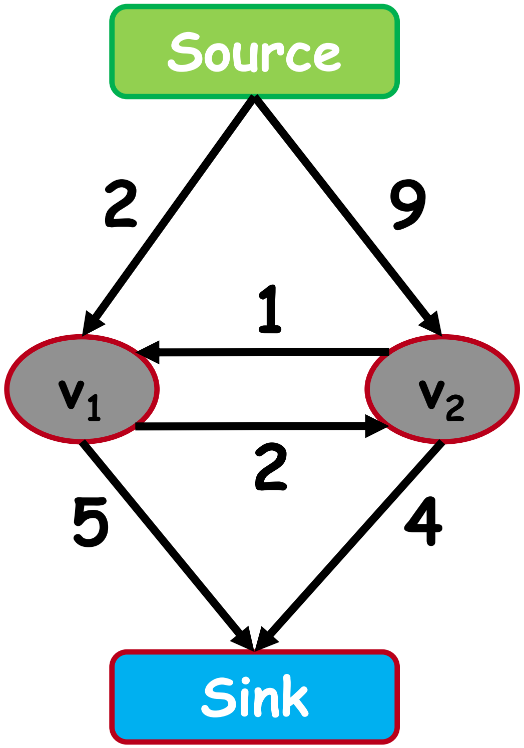
EM-style optimization

Topics of This Lecture

- Recap: Exact inference
  - Factor Graphs
  - Sum-Product/Max-Sum Belief Propagation
  - Junction Tree algorithm
- Applications of Markov Random Fields
  - Application examples from computer vision
  - Interpretation of clique potentials
  - Unary potentials
  - Pairwise potentials
- **Solving MRFs with Graph Cuts**
  - **Graph cuts for image segmentation**
  - **s-t mincut algorithm**
  - **Extension to non-binary case**
  - **Applications**



How Does it Work? The s-t-Mincut Problem



Graph (V, E, C)

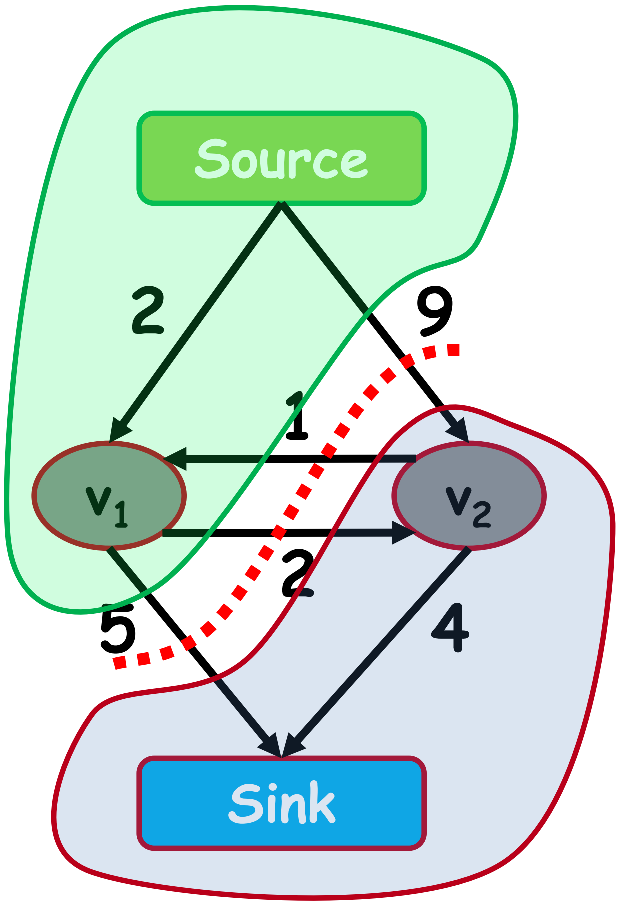
Vertices $V = \{v_1, v_2 \dots v_n\}$

Edges $E = \{(v_1, v_2) \dots\}$

Costs $C = \{c_{(1, 2)} \dots\}$

Slide credit: Pushmeet Kohli

The s-t-Mincut Problem



5 + 2 + 9 = 16

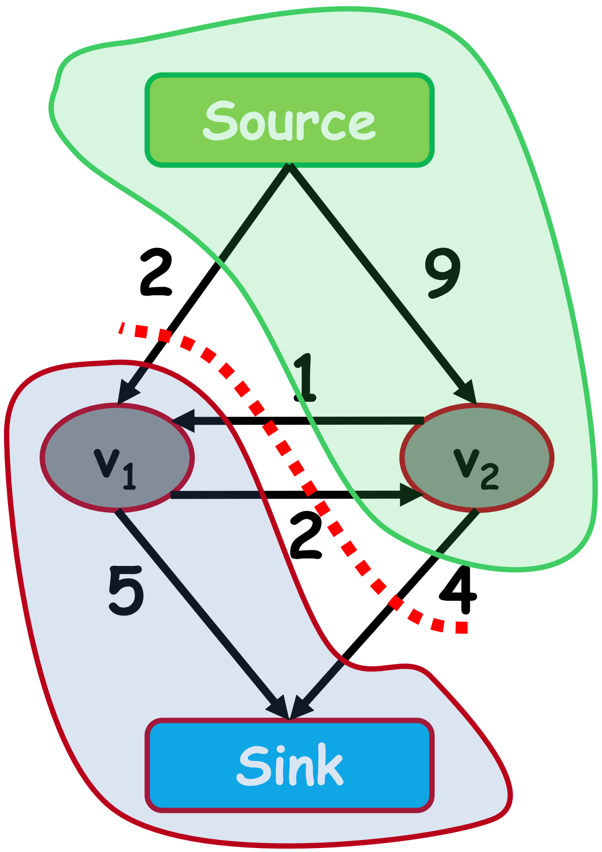
What is an st-cut?

An st-cut (S,T) divides the nodes between source and sink.

What is the cost of a st-cut?

Sum of cost of all edges going from S to T

The s-t-Mincut Problem



2 + 1 + 4 = 7

What is an st-cut?

An st-cut (S,T) divides the nodes between source and sink.

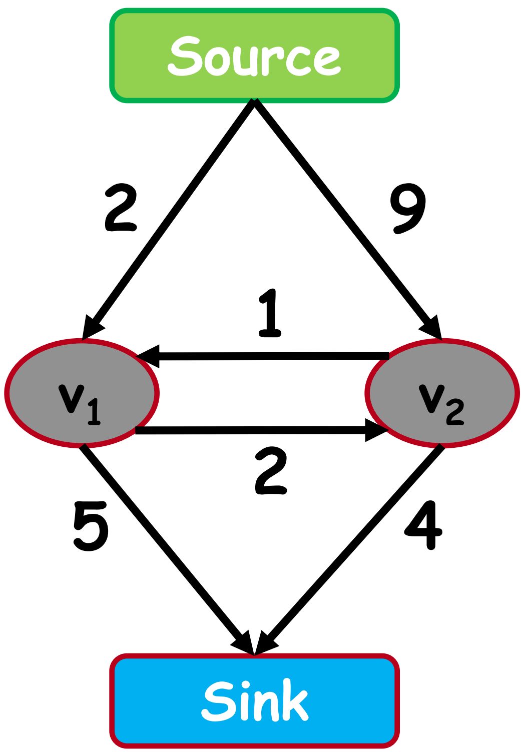
What is the cost of a st-cut?

Sum of cost of all edges going from S to T

What is the st-mincut?

st-cut with the minimum cost

How to Compute the s-t-Mincut?



Solve the dual maximum flow problem

Compute the maximum flow between Source and Sink

Constraints

Edges: Flow < Capacity

Nodes: Flow in = Flow out

Min-cut/Max-flow Theorem

In every network, the maximum flow equals the cost of the st-mincut

Slide credit: Pushmeet Kohli

History of Maxflow Algorithms

Augmenting Path and Push-Relabel

year	discoverer(s)	bound
1951	Dantzig	$O(n^2mU)$
1955	Ford & Fulkerson	$O(m^2U)$
1970	Dinitz	$O(n^2m)$
1972	Edmonds & Karp	$O(m^2 \log U)$
1973	Dinitz	$O(nm \log U)$
1974	Karzanov	$O(n^3)$
1977	Cherkassky	$O(n^2m^{1/2})$
1980	Galil & Naamad	$O(nm \log^2 n)$
1983	Sleator & Tarjan	$O(nm \log n)$
1986	Goldberg & Tarjan	$O(nm \log(n^2/m))$
1987	Ahuja & Orlin	$O(nm + n^2 \log U)$
1987	Ahuja et al.	$O(nm \log(n\sqrt{\log U}/m))$
1989	Cheriyan & Hagerup	$E(nm + n^2 \log^2 n)$
1990	Cheriyan et al.	$O(n^3/\log n)$
1990	Alon	$O(nm + n^{8/3} \log n)$
1992	King et al.	$O(nm + n^{2+\epsilon})$
1993	Phillips & Westbrook	$O(nm(\log_{m/n} n + \log^{2+\epsilon} n))$
1994	King et al.	$O(nm \log_{m/(n \log n)} n)$
1997	Goldberg & Rao	$O(m^{3/2} \log(n^2/m) \log U)$ $O(n^{2/3}m \log(n^2/m) \log U)$

n: #nodes

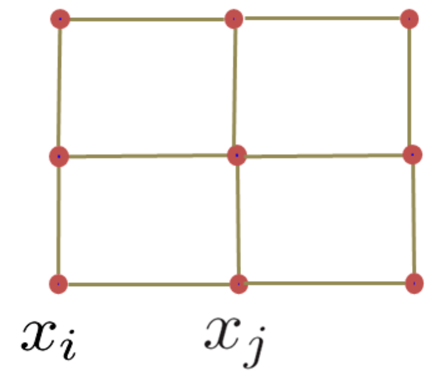
m: #edges

U: maximum edge weight

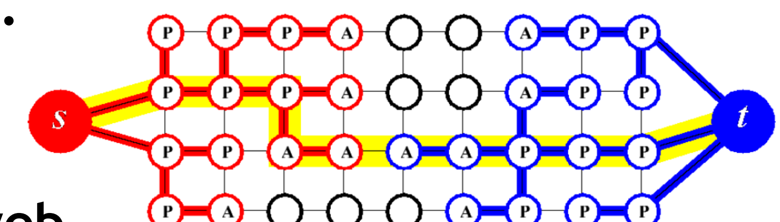
Algorithms assume non-negative edge weights

Applications: Maxflow in Computer Vision

- **Specialized algorithms for vision problems**
  - Grid graphs
  - Low connectivity (m ~ O(n))



- **Dual search tree augmenting path algorithm**
  [Boykov and Kolmogorov PAMI 2004]
  - Finds approximate shortest augmenting paths efficiently.
  - High worst-case time complexity.
  - Empirically outperforms other algorithms on vision problems.
  - Efficient code available on the web
    http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html



When Can s-t Graph Cuts Be Applied?

$$E(L) = \underbrace{\sum_{p} E_p(L_p)}_{\text{unary potentials}} + \underbrace{\sum_{pq \in N} E(L_p, L_q)}_{\text{pairwise potentials}}$$

t-links n-links $L_p \in \{s,t\}$

- s-t graph cuts can only globally minimize **binary energies** that are **submodular**. [Boros & Hummer, 2002, Kolmogorov & Zabih, 2004]

E(L) can be minimized by *s-t* graph cuts $\Leftrightarrow$ $E(s,s)+E(t,t) \le E(s,t)+E(t,s)$

Submodularity ("convexity")

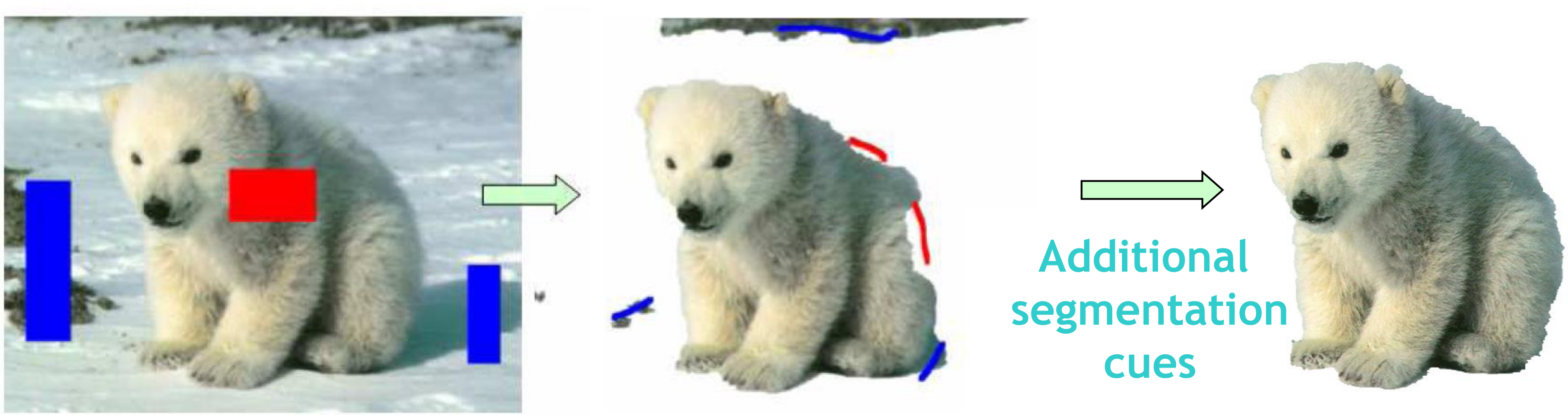
- Submodularity is the discrete equivalent to convexity.
  - Implies that every local energy minimum is a global minimum.
  - $\Rightarrow$ Solution will be globally optimal.

Topics of This Lecture

- Recap: Exact inference
  - Factor Graphs
  - Sum-Product/Max-Sum Belief Propagation
  - Junction Tree algorithm
- Applications of Markov Random Fields
  - Application examples from computer vision
  - Interpretation of clique potentials
  - Unary potentials
  - Pairwise potentials
- Solving MRFs with Graph Cuts
  - Graph cuts for image segmentation
  - s-t mincut algorithm
  - Extension to non-binary case
  - Applications

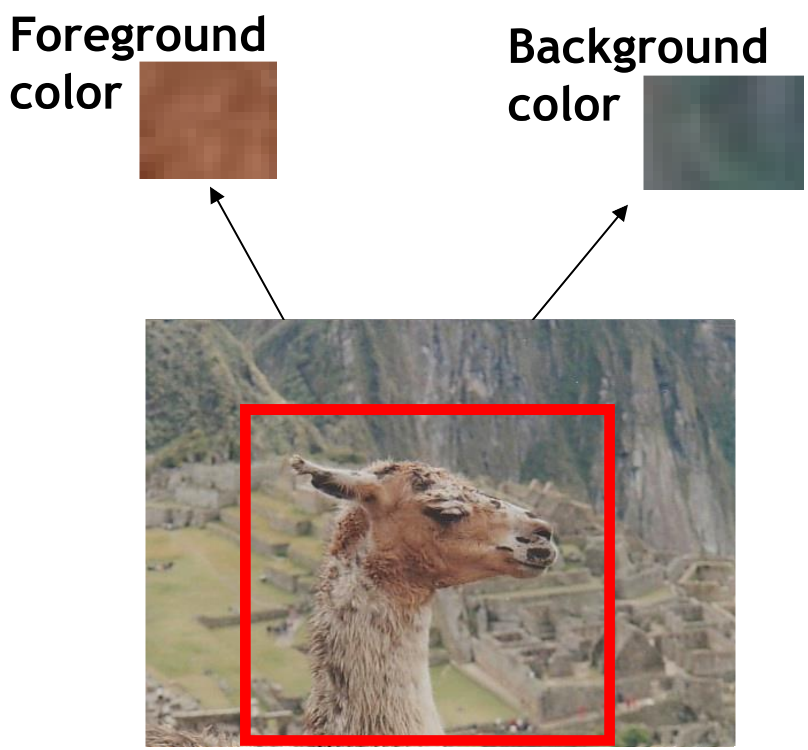
GraphCut Applications: “GrabCut”

- Interactive Image Segmentation [Boykov & Jolly, ICCV’01]
  - Rough region cues sufficient
  - Segmentation boundary can be extracted from edges
- Procedure
  - User marks foreground and background regions with a brush.
  - This is used to create an initial segmentation which can then be corrected by additional brush strokes.



User segmentation cues

GrabCut: Data Model

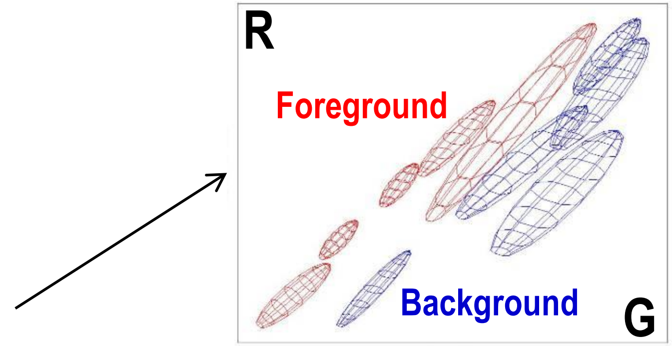


Global optimum of the energy

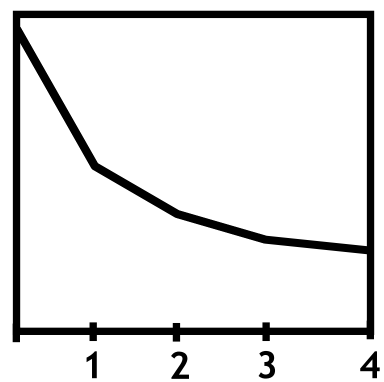
- **Obtained from interactive user input**
  - User marks foreground and background regions with a brush
  - Alternatively, user can specify a bounding box

Iterated Graph Cuts

Result

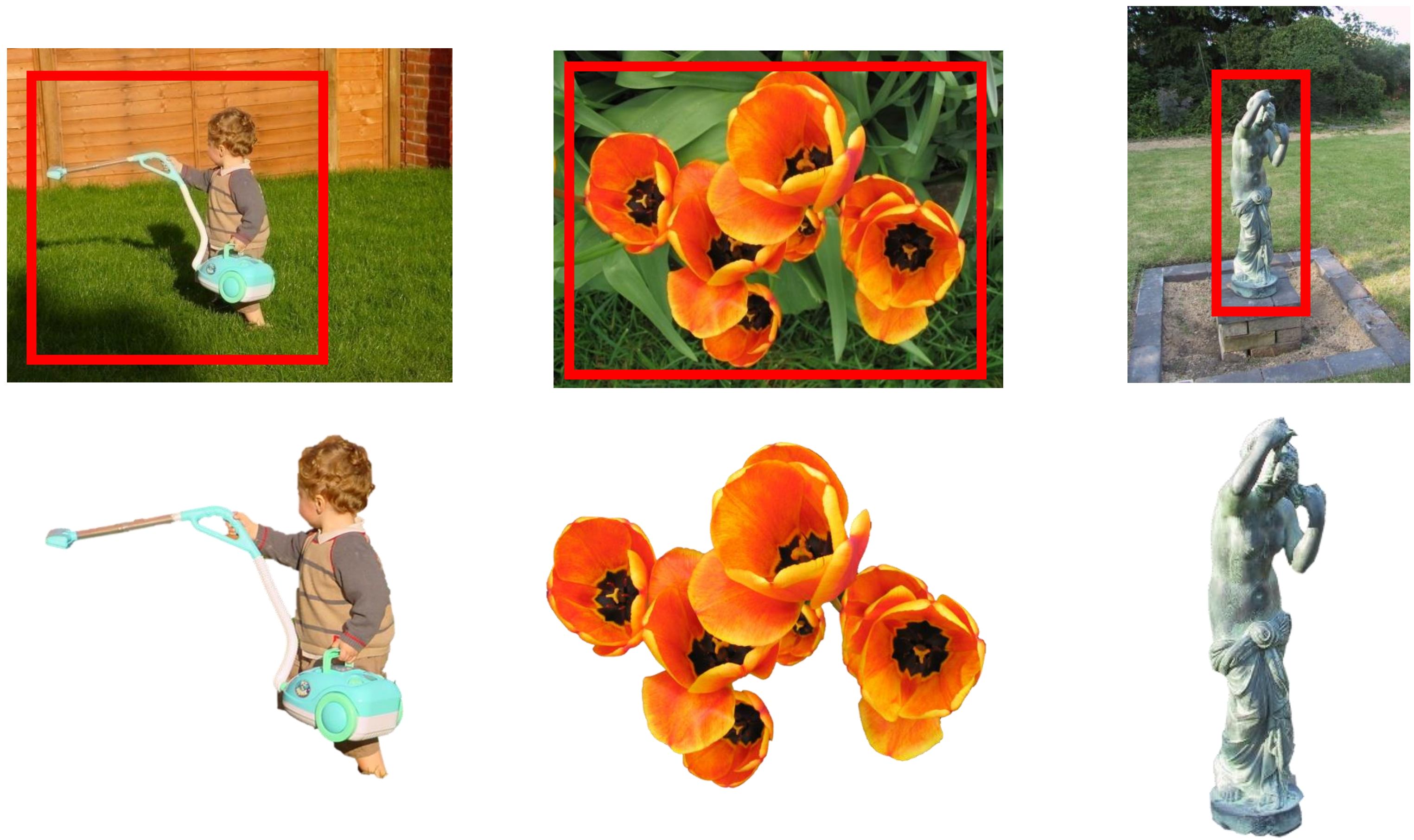


Color model (Mixture of Gaussians)



Energy after each iteration

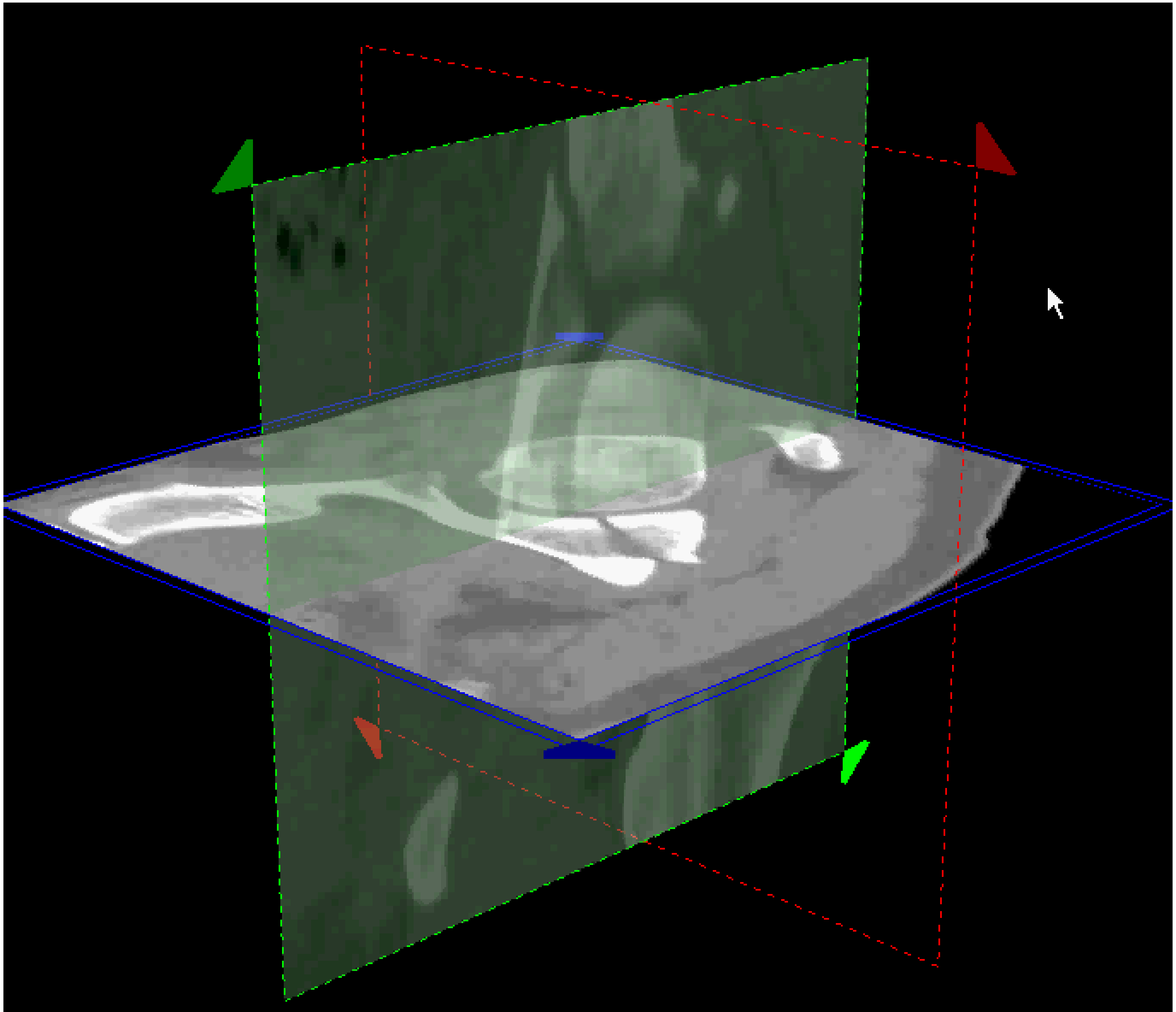
GrabCut: Example Results



- ***This is included in the newest versions of MS Office!***

Image source: Carsten Rother

Applications: Interactive 3D Segmentation



[Y. Boykov, V. Kolmogorov, ICCV'03]

References and Further Reading

- A gentle introduction to Graph Cuts can be found in the following paper:
  - Y. Boykov, O. Veksler, Graph Cuts in Vision and Graphics: Theories and Applications. In *Handbook of Mathematical Models in Computer Vision*, edited by N. Paragios, Y. Chen and O. Faugeras, Springer, 2006.

- Try the GraphCut implementation at

  http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html