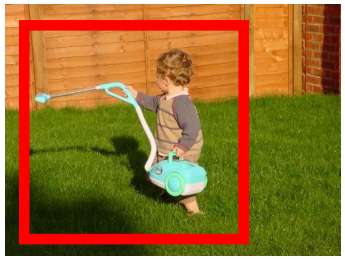
Markov Random Fields and Segmentation with Graph Cuts






Computer Vision

Jia-Bin Huang, Virginia Tech

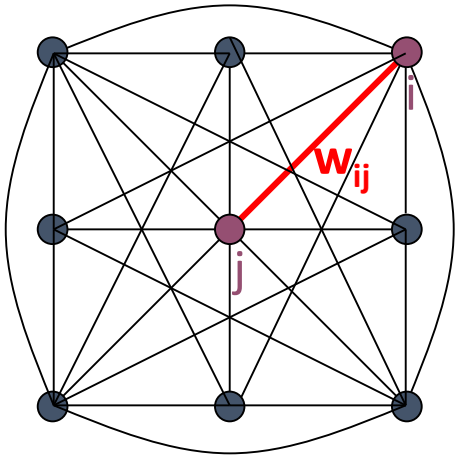
Many slides from D. Hoiem

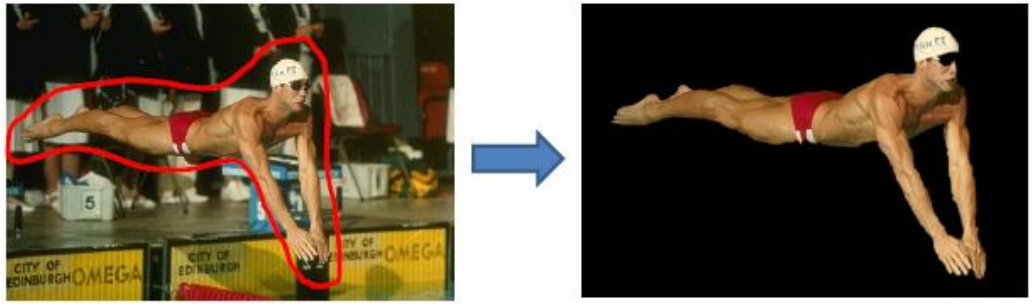
Administrative stuffs

- Final project
  - Proposal due Oct 27 (Thursday)
- HW 4 is out
  - Due 11:59pm on Wed, November 2nd, 2016

Today's class

- Review EM and GMM
- Markov Random Fields
- Segmentation with Graph Cuts
- HW 4



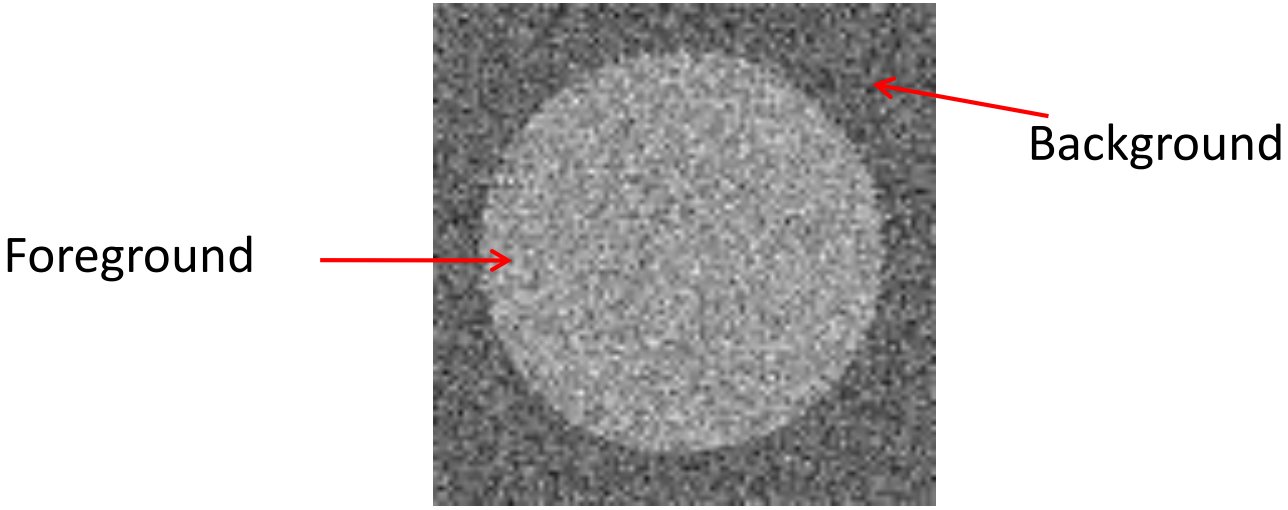


Missing Data Problems: Segmentation

Challenge: Segment the image into figure and ground without knowing what the foreground looks like in advance.

Three sub-problems:

1. If we had labels, how could we model the appearance of foreground and background?
   MLE: maximum likelihood estimation
2. Once we have modeled the fg/bg appearance, how do we compute the likelihood that a pixel is foreground? **Probabilistic inference**
3. How can we get both labels and appearance models at once?
   Hidden data problem: Expectation Maximization



EM: Mixture of Gaussians

1. Initialize parameters

2. Compute likelihood of hidden variables for current parameters

Given by normal distribution

$$\alpha_{nm} = p(z_n = m \mid x_n, \boldsymbol{\mu}^{(t)}, \boldsymbol{\sigma}^{2(t)}, \boldsymbol{\pi}^{(t)}) = \frac{p(x_n \mid z_n = m, \theta_m) p(z_n = m \mid \theta_m)}{\sum_k p(x_n \mid z_n = k, \theta_k) p(z_n = k \mid \theta_k)}$$

3. Estimate new parameters for each component, weighted by likelihood

$$\hat{\mu}_m^{(t+1)} = \frac{1}{\sum_n \alpha_{nm}} \sum_n \alpha_{nm} x_n \qquad \hat{\sigma}_m^{2(t+1)} = \frac{1}{\sum_n \alpha_{nm}} \sum_n \alpha_{nm} (x_n - \hat{\mu}_m)^2 \qquad \hat{\pi}_m^{(t+1)} = \frac{\sum_n \alpha_{nm}}{N}$$

Gaussian Mixture Models: Practical Tips

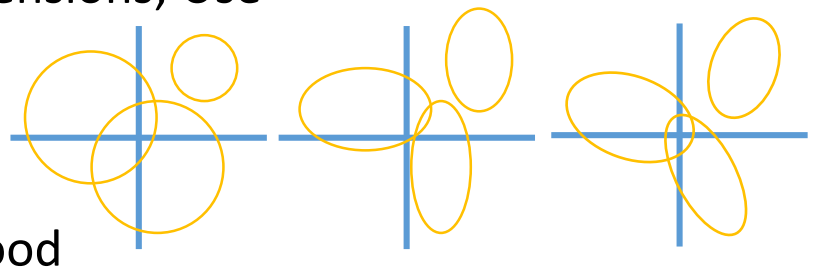
- Number of components
  - Select by hand based on knowledge of problem
  - Select using cross-validation or sample data
  - Usually, not too sensitive and safer to use more components
- Covariance matrix
  - Spherical covariance: dimensions within each component are independent with equal variance (1 parameter but usually too restrictive)
  - Diagonal covariance: dimensions within each component are not independent with difference variances (N parameters for N-D data)
  - Full covariance: no assumptions (N*(N+1)/2 parameters); for high N might be expensive to do EM, to evaluate, and may overfit
  - Typically use "Full" if lots of data, few dimensions; Use "Diagonal" otherwise
- Can get stuck in local minima
  - Use multiple restarts
  - Choose solution with greatest data likelihood

Spherical

$$\begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

$$\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

$$\begin{bmatrix} a & c \\ c & b \end{bmatrix}$$



Spherical Diagonal Full

“Hard EM”

- Same as EM except compute z* as most likely values for hidden variables
- K-means is an example
- Advantages
  - Simpler: can be applied when cannot derive EM
  - Sometimes works better if you want to make hard predictions at the end
- But
  - Generally, pdf parameters are not as accurate as EM

EM Demo

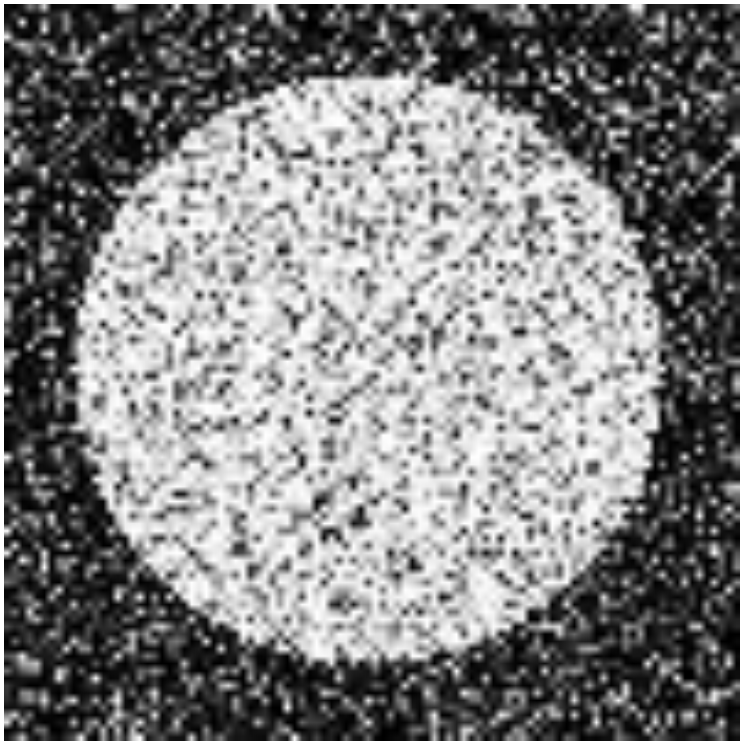
- GMM with images demos

```
function EM_segmentation(im, K)
x = im(:);
N = numel(x);
minsigma = std(x)/numel(x);  % prevent component from getting 0 variance

% Initialize GMM parameters
prior = zeros(K, 1);
mu = zeros(K, 1);
sigma = zeros(K, 1);
prior(:) = 1/K;
minx = min(x);
maxx = max(x);
for k = 1:K
    mu(k) = (0.1+0.8*rand(1))*(maxx-minx) + minx;
    sigma(k) = (1/K)*std(x);
end
% Initialize P(component_i | x_i) (initial values not important)
pm = ones(N, K);
oldpm = zeros(N, K);
```

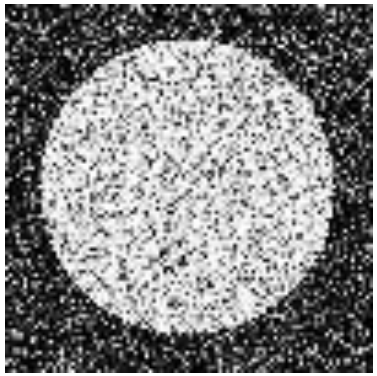
```
maxiter = 200;
niter = 0;
% EM algorithm: loop until convergence
while (mean(abs(pm(:)-oldpm(:)))>0.001) && (niter < maxiter)
  niter = niter+1;
  oldpm = pm;
% estimate probability that each data point belongs to each component
  for k = 1:K
     pm(:, k) = prior(k)*normpdf(x, mu(k), sigma(k));
  end
  pm = pm ./ repmat(sum(pm, 2), [1 K]);
  % compute maximum likelihood parameters for expected components
  for k = 1:K
     prior(k) = sum(pm(:, k))/N;
     mu(k) = sum(pm(:, k).*x) / sum(pm(:, k));
     sigma(k) = sqrt( sum(pm(:, k).*(x - mu(k)).^2) / sum(pm(:, k)));
     sigma(k) = max(sigma(k), minsigma); % prevent variance from going to 0
  end
end
```

What’s wrong with this prediction?



P(foreground | image)

Solution:
Encode dependencies between pixels



P(foreground | image)

Normalizing constant called “partition function”

$$P(\mathbf{y};\theta,data) = \frac{1}{Z}\prod_{i=1..N} f_1(y_i;\theta,data)\prod_{i,j\in edges} f_2(y_i,y_j;\theta,data)$$

Labels to be predicted

Individual predictions

Pairwise predictions

Writing Likelihood as an "Energy"

$$P(\mathbf{y};\theta,data) = \frac{1}{Z}\prod_{i=1..N} p_1(y_i;\theta,data)\prod_{i,j\in edges} p_2(y_i,y_j;\theta,data)$$

-log

$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

Cost of assignment y_i

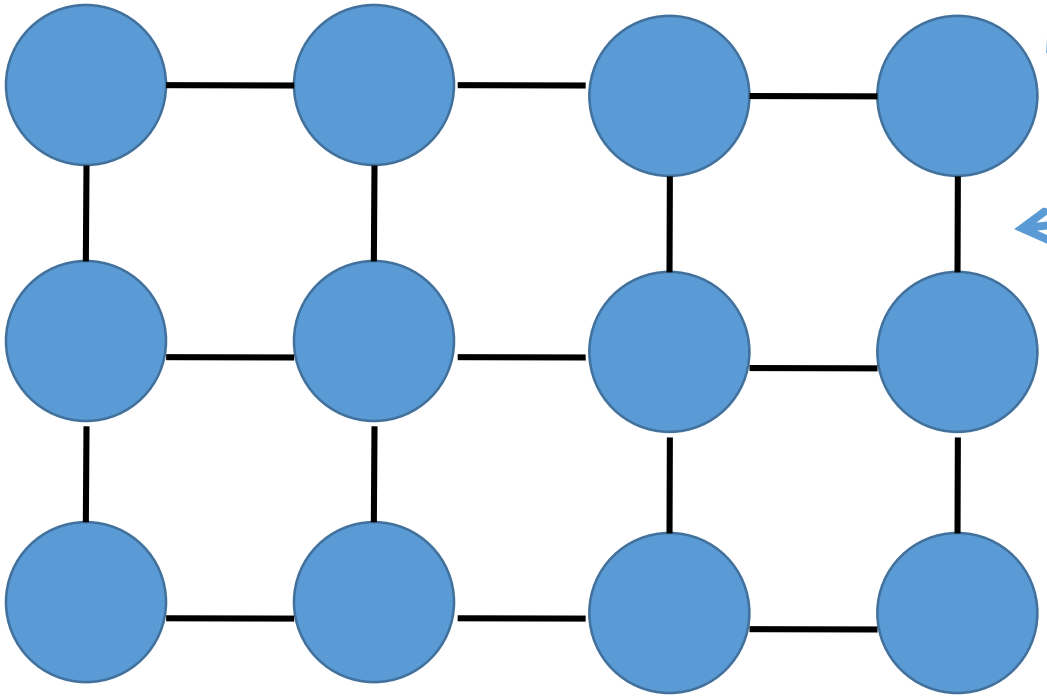
Cost of pairwise assignment y_i, y_j

Notes on energy-based formulation

$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

- Primarily used when you only care about the most likely solution (not the confidences)
- Can think of it as a general cost function
- Can have larger “cliques” than 2
  - Clique is the set of variables that go into a potential function

Markov Random Fields



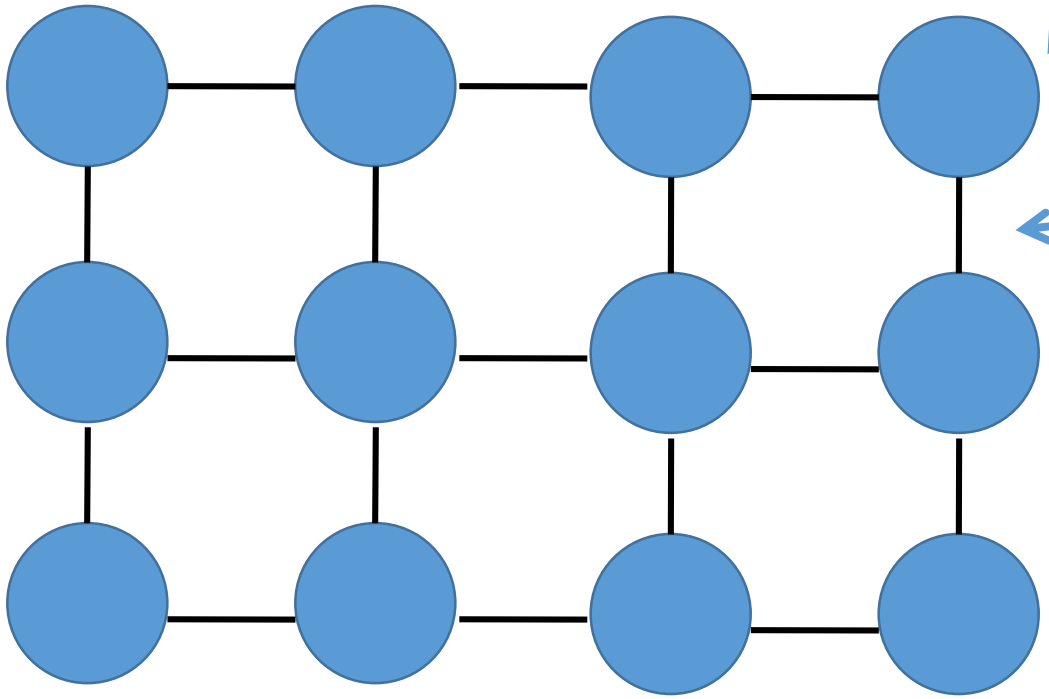
Cost to assign a label to each pixel

Cost to assign a pair of labels to connected pixels

$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

Markov Random Fields

- Example: "label smoothing" grid



Unary potential

0: -logP(y_i = 0 | data)
1: -logP(y_i = 1 | data)

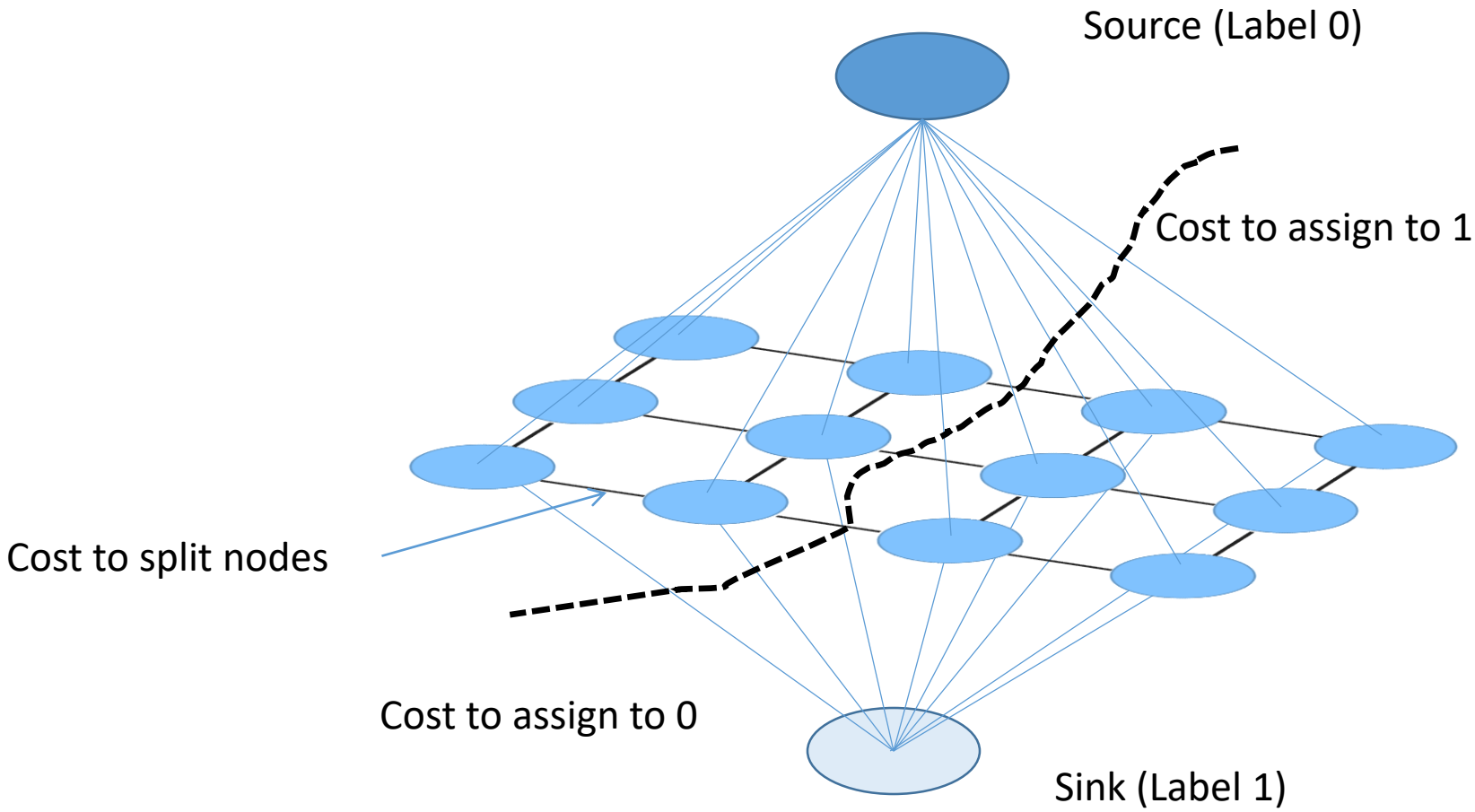
Pairwise Potential

	0	1
0	0	K
1	K	0

K>0

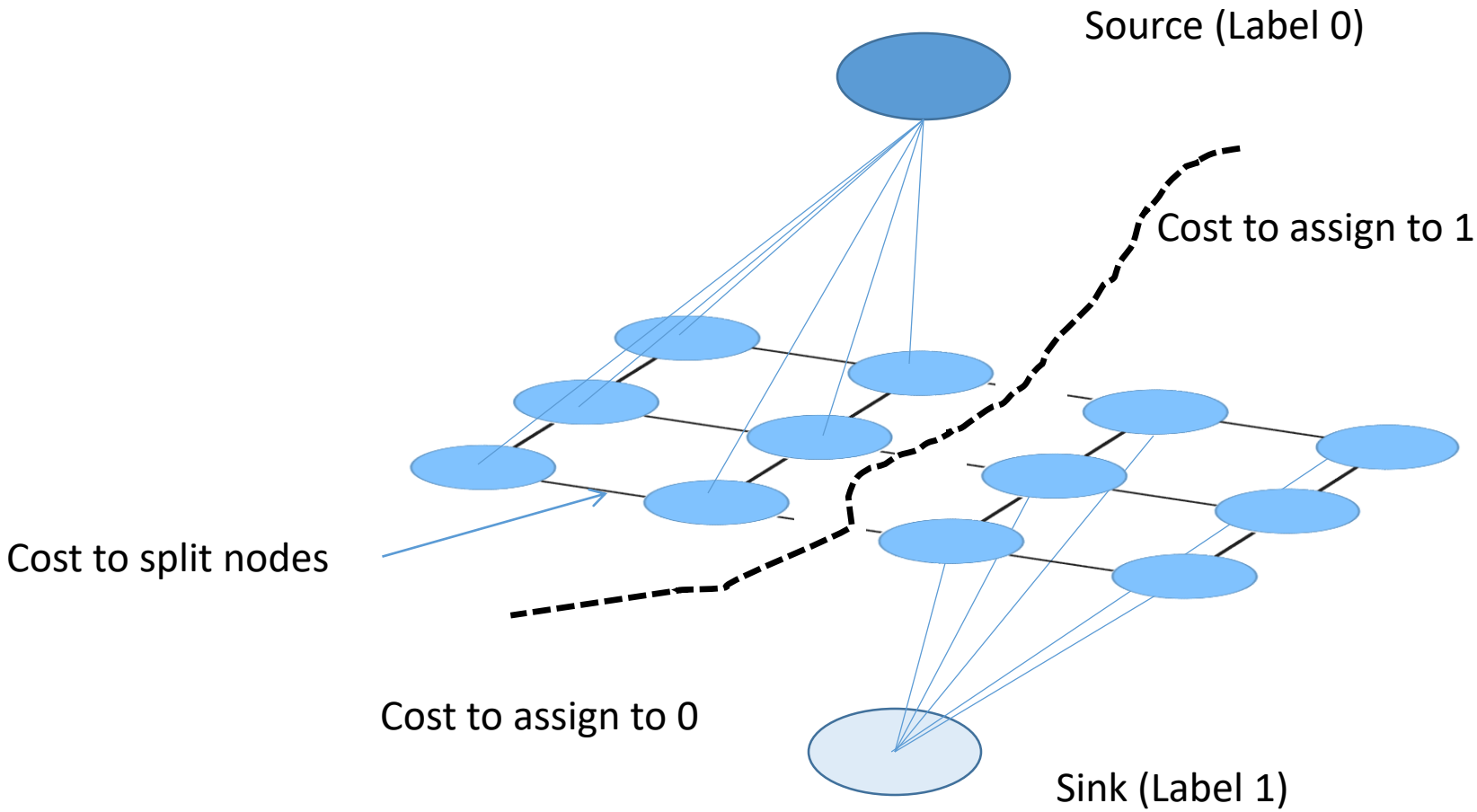
$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

Solving MRFs with graph cuts



$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

Solving MRFs with graph cuts



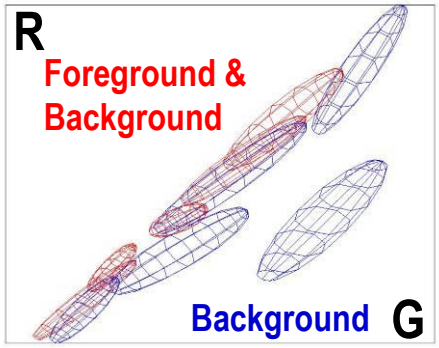
$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

GrabCut segmentation

User provides rough indication of foreground region.

Goal: Automatically provide a pixel-level segmentation.

Colour Model

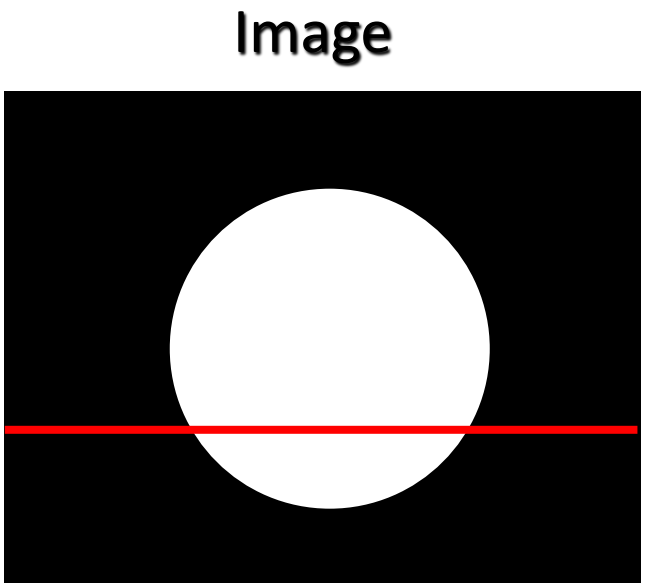


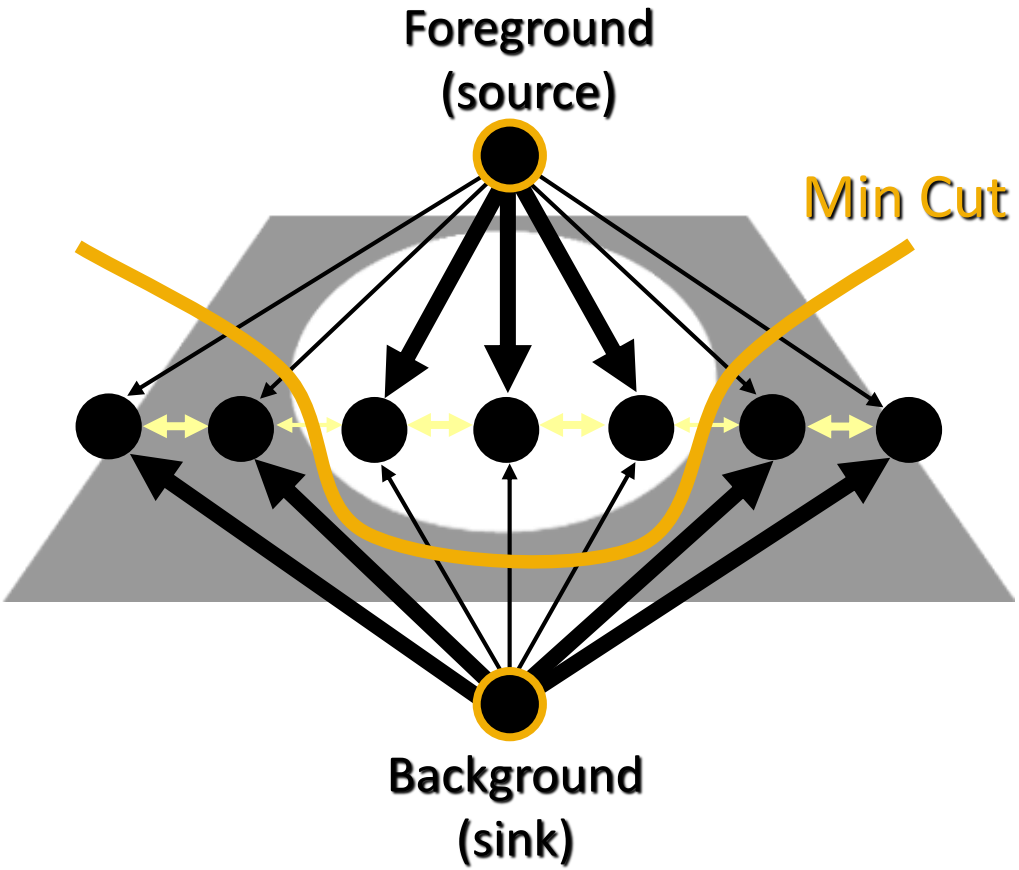
Gaussian Mixture Model (typically 5-8 components)

Source: Rother

Graph cuts

Boykov and Jolly (2001)



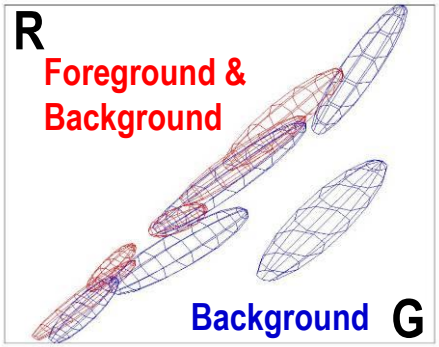


Cut: separating source and sink; Energy: collection of edges

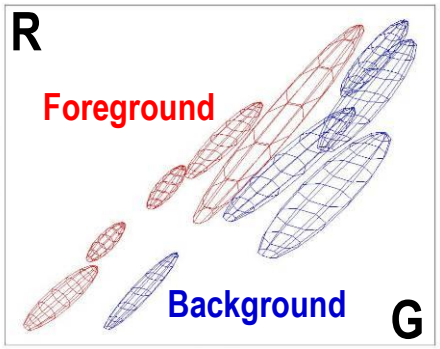
Min Cut: Global minimal energy in polynomial time

Source: Rother

Colour Model



Iterated
graph cut



Gaussian Mixture Model (typically 5-8 components)

Source: Rother

GrabCut segmentation

1. Define graph
   - usually 4-connected or 8-connected
     - Divide diagonal potentials by sqrt(2)
2. Define unary potentials
   - Color histogram or mixture of Gaussians for background and foreground

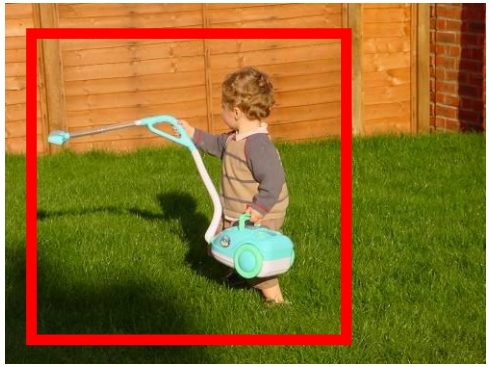
$$unary\_potential(x) = -\log\left(\frac{P(c(x);\theta_{foreground})}{P(c(x);\theta_{background})}\right)$$

3. Define pairwise potentials

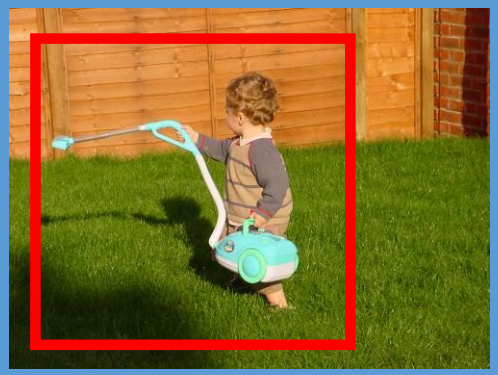
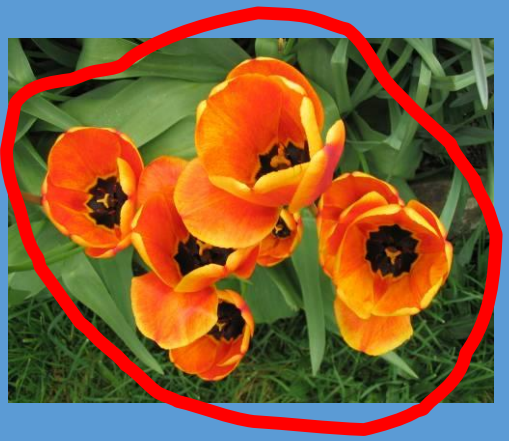
$$edge\_potential(x,y) = k_1 + k_2 \exp\left\{\frac{-\|c(x)-c(y)\|^2}{2\sigma^2}\right\}$$

4. Apply graph cuts
5. Return to 2, using current labels to compute foreground, background models

What is easy or hard about these cases for graphcut-based segmentation?



Easier examples



More difficult Examples

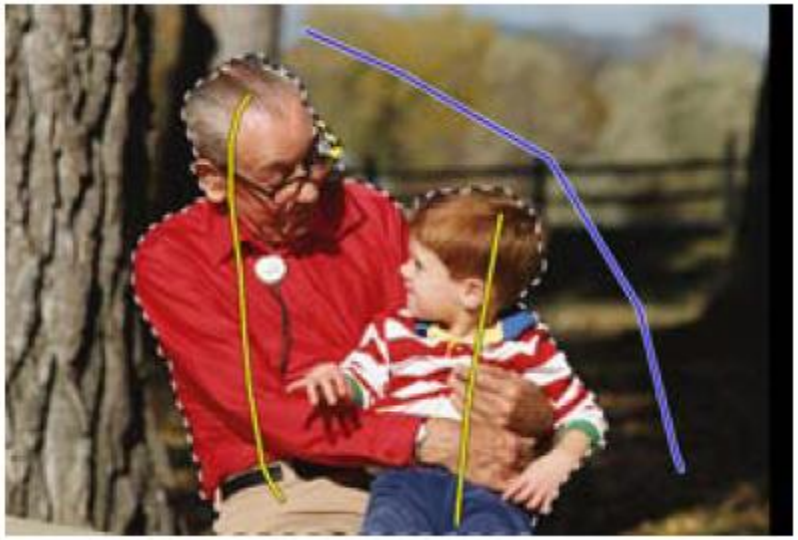
Fine structure

Harder Case

Initial Rectangle

Initial Result

Lazy Snapping (Li et al. SG 2004)



Graph cuts with multiple labels

- Alpha expansion

  Repeat until no change

  For $\alpha = 1..M$

  Assign each pixel to current label or α (2-class graphcut)

  - Achieves "strong" local minimum

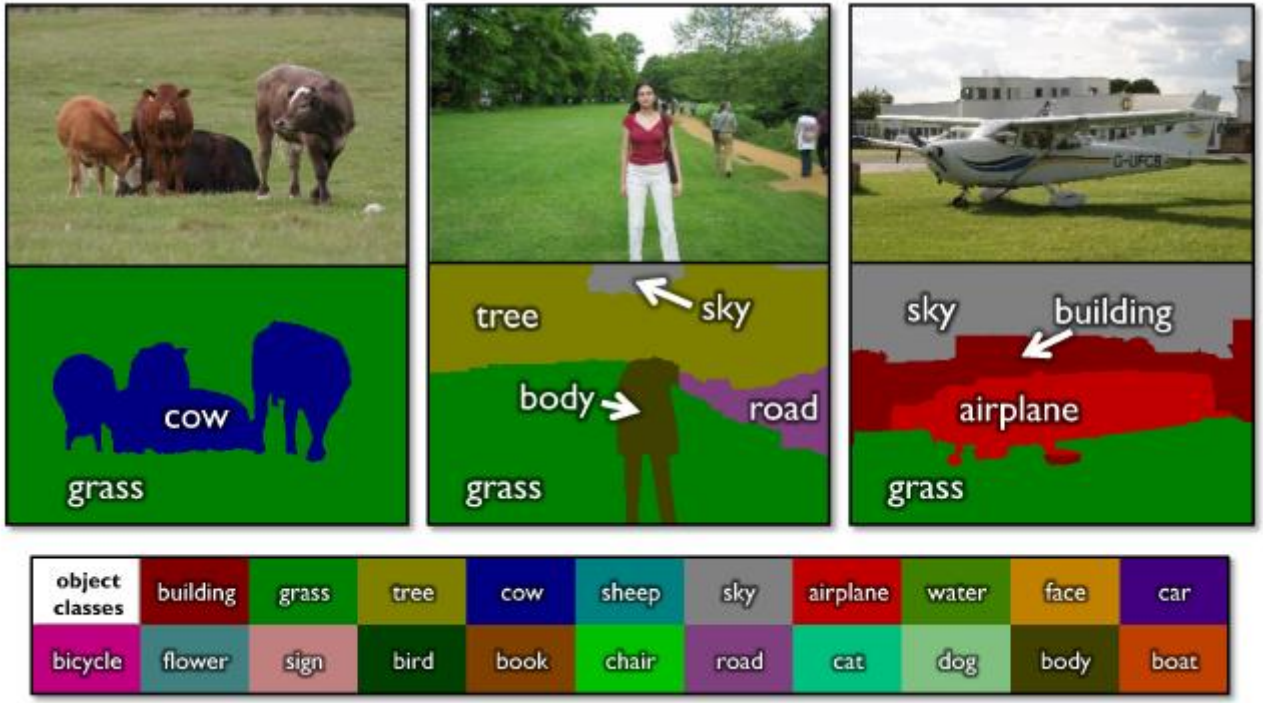
- Alpha-beta swap

  Repeat until no change

  For $\alpha = 1..M, \quad \beta = 1..M$ (except α)

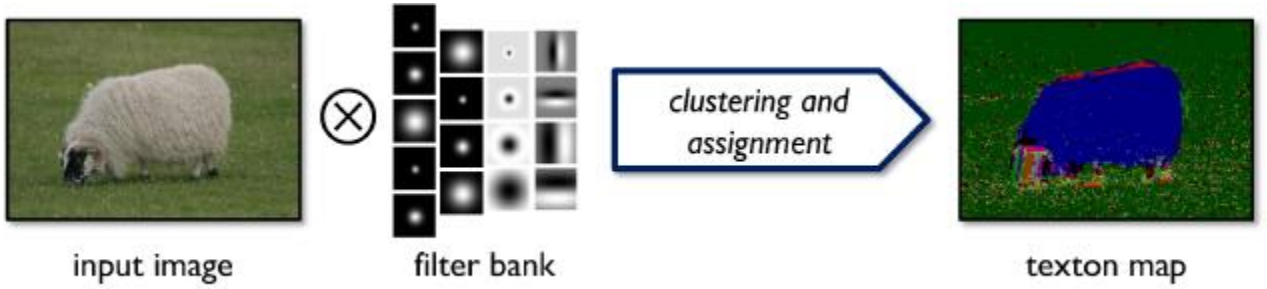
  Re-assign all pixels currently labeled as α or β to one of those two labels while keeping all other pixels fixed

Using graph cuts for recognition



TextonBoost (Shotton et al. 2009 IJCV)

Using graph cuts for recognition



Unary Potentials from classifier

(note: edge potentials are from input image also; this is illustration from paper)

Alpha Expansion Graph Cuts

TextonBoost (Shotton et al. 2009 IJCV)

Limitations of graph cuts

- Associative: edge potentials penalize different labels

Must satisfy

$$E^{i,j}(0,0) + E^{i,j}(1,1) \leq E^{i,j}(0,1) + E^{i,j}(1,0)$$

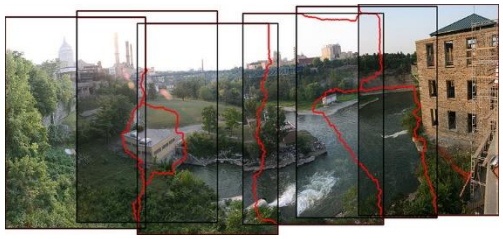
- If not associative, can sometimes clip potentials

- Graph cut algorithm applies to only 2-label problems
  - Multi-label extensions are not globally optimal (but still usually provide very good solutions)

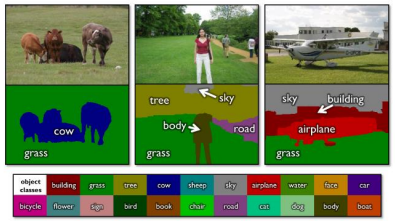
Graph cuts: Pros and Cons

- Pros
  - Very fast inference (stereo, image labeling, recognition)
  - Can incorporate data likelihoods and priors
  - Applies to a wide range of problems

stereo



stitching



recognition

- Cons
  - Not always applicable (associative only)
  - Need unary terms (not used for bottom-up segmentation, for example)
- Use whenever applicable

More about MRFs/CRFs

- Other common uses
  - Graph structure on regions
  - Encoding relations between multiple scene elements
- Inference methods
  - Loopy BP or BP-TRW
    - Exact for tree-shaped structures
    - Approximate solutions for general graphs
    - More widely applicable and can produce marginals but often slower

Further reading and resources

- Graph cuts
  - http://www.cs.cornell.edu/~rdz/graphcuts.html
  - Classic paper: What Energy Functions can be Minimized via Graph Cuts? (Kolmogorov and Zabih, ECCV '02/PAMI '04)

- Belief propagation

  Yedidia, J.S.; Freeman, W.T.; Weiss, Y., "Understanding Belief Propagation and Its Generalizations", Technical Report, 2001: http://www.merl.com/publications/TR2001-022/

- Comparative study
  - Szeliski et al. A comparative study of energy minimization methods for markov random fields with smoothness-based priors, PAMI 2008
  - Kappes et al. A comparative study of modern inference techniques for discrete energy minimization problems, CVPR 2013

HW 4 Part 1: SLIC (Achanta et al. PAMI 2012)

http://infoscience.epfl.ch/record/177415/files/Superpixel_PAMI2011-2.pdf

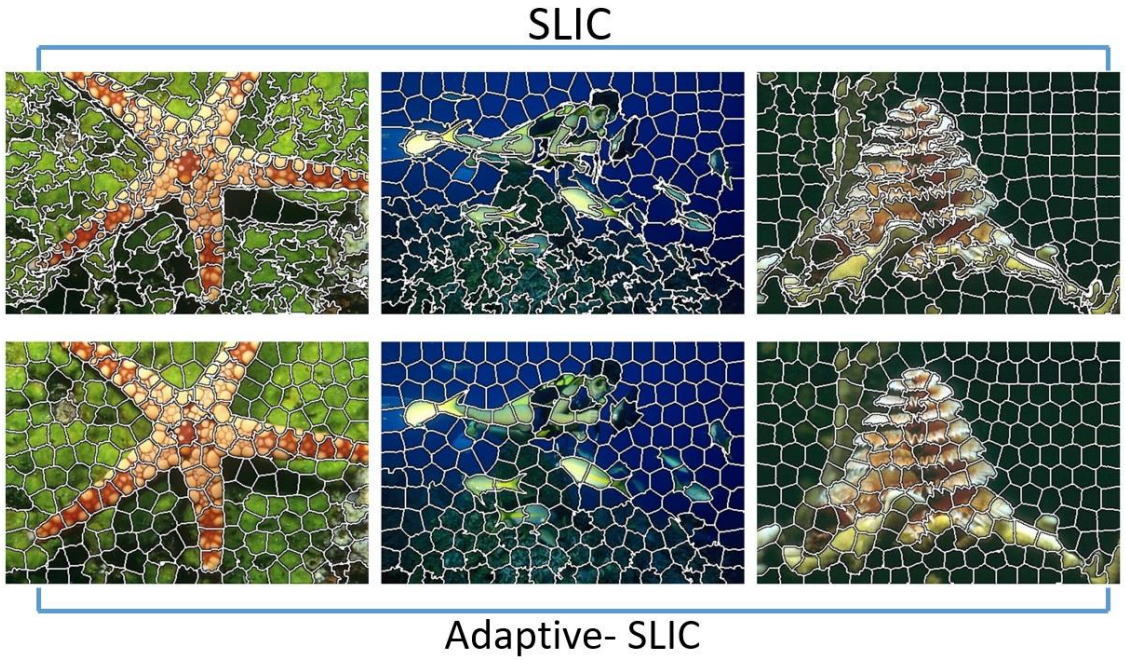
1. Initialize cluster centers on pixel grid in steps S
   - Features: Lab color, x-y position
2. Move centers to position in 3x3 window with smallest gradient
3. Compare each pixel to cluster center within 2S pixel distance and assign to nearest
4. Recompute cluster centers as mean color/position of pixels belonging to each cluster
5. Stop when residual error is small

HW 4 Part 1: SLIC – Graduate credits

- (**up to 15 points**) Improve your results on SLIC
  - Color space, gradient features, edges
- (**up to 15 points**) Implement Adaptive-SLIC

$$D = \sqrt{\left(\frac{d_c}{m_c}\right)^2 + \left(\frac{d_s}{m_s}\right)^2}$$

- d_c: Color difference
- d_s: Spatial difference
- maximum observed spatial and color distances (m_s, m_c)



HW 4 Part 2: EM algorithm

Dealing with noisy annotations is a common problem in computer vision, especially when using crowdsourcing tools, like Amazon's Mechanical Turk. For this problem, you've collected photo aesthetic ratings for 150 images. Each image is labeled 5 times by a total of 25 annotators (each annotator provided 30 labels). Each label consists of a continuous score from 0 (unattractive) to 10 (attractive). The problem is that some users do not understand instructions or are trying to get paid without attending to the image. These "bad" annotators assign a label uniformly at random from 0 to 10. Other "good" annotators assign a label to the i^{th} image with mean μ_i and standard deviation σ (σ is the same for all images). Your goal is to solve for the most likely image scores and to figure out which annotators are trying to cheat you. In your write-up, use the following notation:

- $x_{ij} \in [0, 10]$: the score for i^{th} image from the j^{th} annotator
- $m_j \in \{0, 1\}$: whether each j^{th} annotator is "good" ($m_j = 1$) or "bad" ($m_j = 0$)
- $P(x_{ij}|m_j = 0) = \frac{1}{10}$: uniform distribution for bad annotators
- $P(x_{ij}|m_j = 1; \mu_i, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2}\frac{(x_{ij}-\mu_i)^2}{\sigma^2})$: normal distribution for good annotators
- $P(m_j = 1; \beta) = \beta$: prior probability for being a good annotator

2.1 Derivation of EM Algorithm (20 pts)

Derive the EM algorithm to solve for each μ_i, each m_j, σ, and β. Show the major steps of the derivation and make it clear how to compute each variable in the update step.

2.2 Application to Data (15 pts)

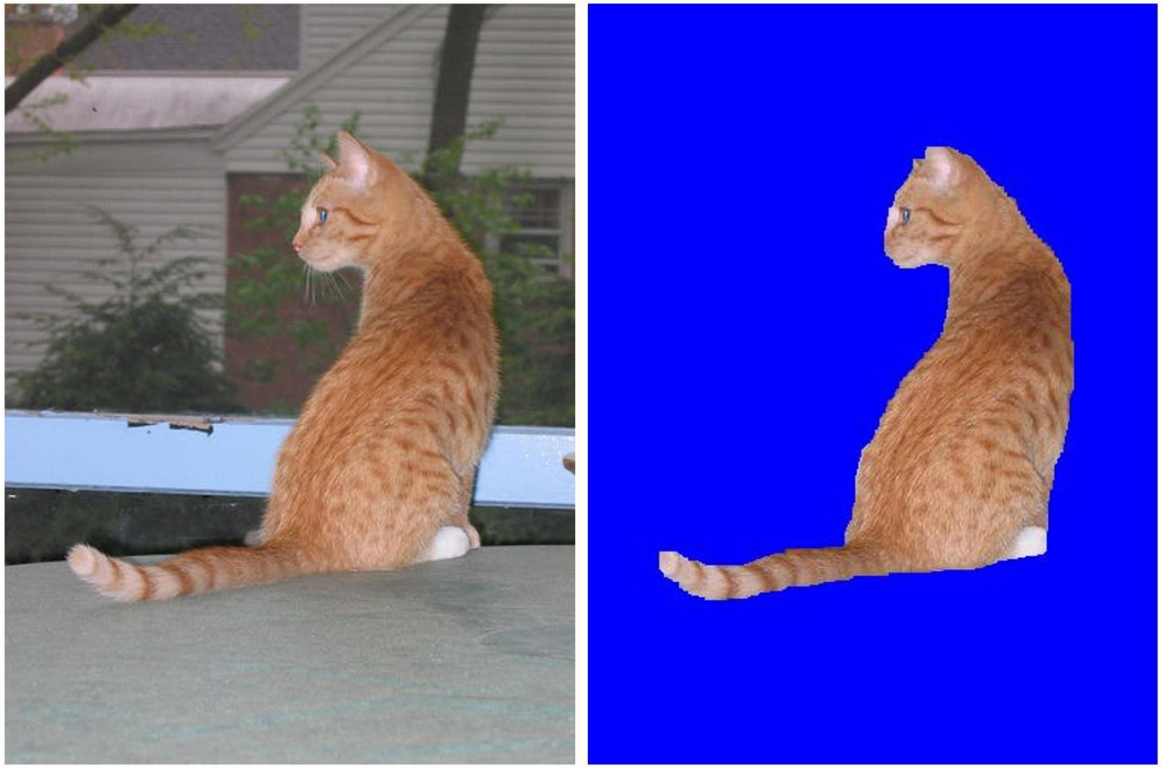
The false scores come from a uniform distribution

The true scores for each image have a Gaussian distribution

Annotators are always "bad" or always "good"

The "good/bad" label of each annotator is the missing data

HW 4 Part 3: GraphCut



- Define unary potential
- Define pairwise potential
  - Contrastive term
- Solve segementation using graph-cut
  - Read `GraphCut.m`

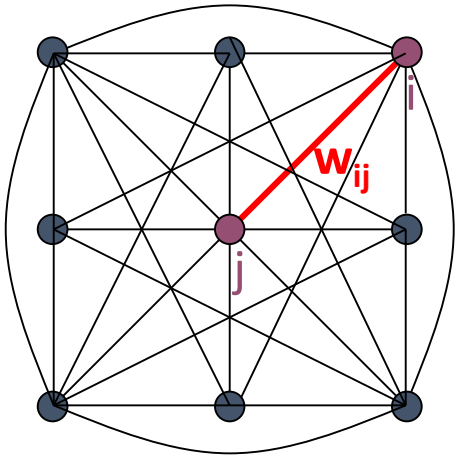
HW 4 Part 3: GraphCut – Graduate credits

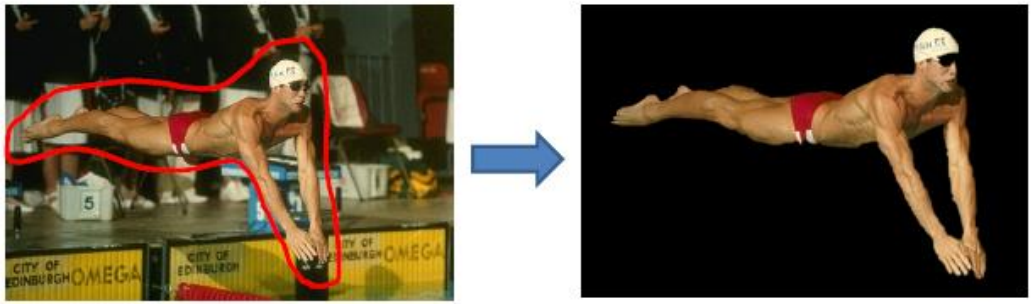
- (**up to 15 points**) try two more images

- (**up to 10 points**) image composition

Things to remember

- Markov Random Fields
  - Encode dependencies between pixels
- Likelihood as energy
- Segmentation with Graph Cuts





Next module: Object Recognition

- Face recognition
- Image categorization
- Machine learning
- Object category detection
- Tracking objects