CS 664 Slides #8
Regularization and MRF’s

Prof. Dan Huttenlocher
Fall 2003

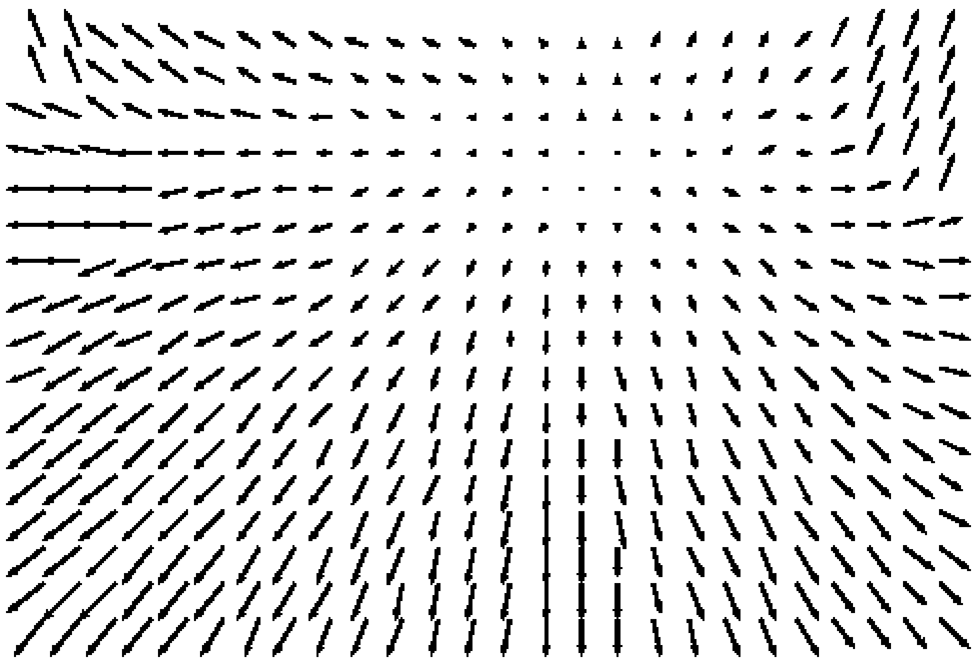
Regularization in Low Level Vision

- Low level vision problems concerned with estimating some quantity at each pixel
  - Visual motion (u(x,y),v(x,y))
  - Stereo disparity d(x,y)
  - Restoration of true intensity b(x,y)
- Problem under constrained
  - Only able to observe noisy values at each pixel
  - Sometimes single pixel not enough to estimate value
- Need to apply other constraints

Smooth but with discontinuities

First image

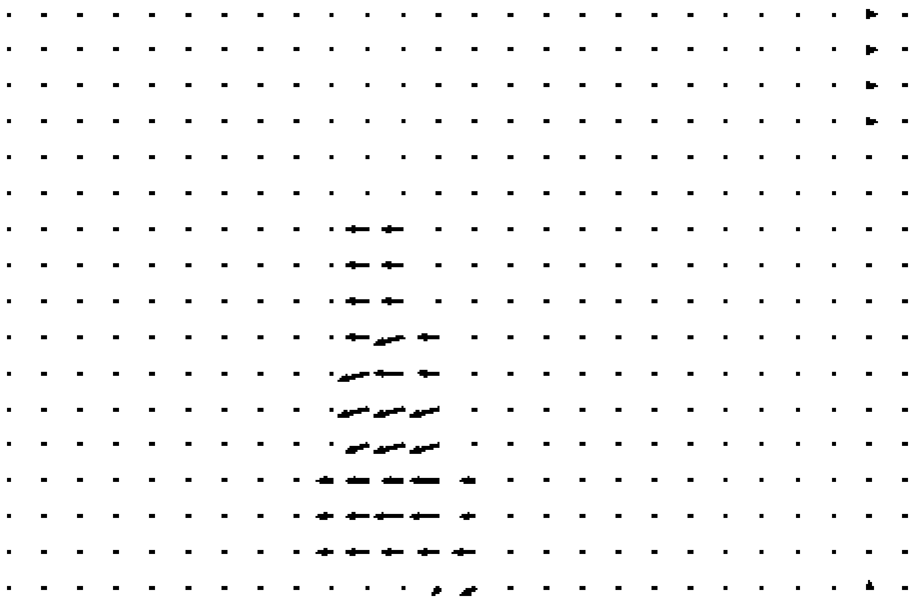
Second image



Small discontinuities important

First image

Second image



Smoothness Constraints

- Estimated values should change slowly as function of (x,y)
  - Except "boundaries" which are relatively rare
- Minimize an error function

$$E(r(x,y)) = V(r(x,y)) + \lambda\, D_{\mathbf{I}}(r(x,y))$$

  - For r being estimated at each x,y location
  - V penalizes change in r in local neighborhood
  - $D_{\mathbf{I}}$ penalizes r disagreeing with image data
  - λ controls tradeoff of these smoothness and data terms
    - Can itself be parameterized by x,y

Regularization for Visual Motion

- Use quadratic error function
- Smoothness term

  $$V(u(x,y),v(x,y)) = \sum\sum u_{\mathbf{x}}^{\mathbf{2}} + u_{\mathbf{y}}^{\mathbf{2}} + v_{\mathbf{x}}^{\mathbf{2}} + v_{\mathbf{y}}^{\mathbf{2}}$$

  – Where subscripts denote partials $u_{\mathbf{x}}=\partial u(x,y)/\partial x$, etc.
- Data term

  $$D_{\mathbf{I}}(u(x,y),v(x,y)) = \sum\sum (I_{\mathbf{x}}\cdot u + I_{\mathbf{y}}\cdot v + I_{\mathbf{t}})^{\mathbf{2}}$$
- Only for smoothly changing motion fields
  – No discontinuity boundaries
  – Does not work well in practice

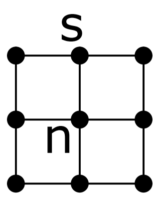
Problems With Regularization

- Computational difficulty
  - Extremely high dimensional minimization problem
    - 2mn dimensional space for $m \times n$ image and motion estimation
    - If k motion values, d^{2mn} possible solutions
    - Can solve with gradient descent methods
- Smoothness too strong a model
  - Can in principle estimate variable smoothness penalty $\lambda_{\mathbf{I}}(x,y)$
    - More difficult computation
    - Need to relate $\lambda_{\mathbf{I}}$ to V, $D_{\mathbf{I}}$

Regularization With Discontinuities

- Line process
  - Estimate binary value representing when discontinuity between neighboring pixels
- Pixels as sites $s \in \mathcal{S}$ (vertices in graph)
  - Neighborhood $\mathcal{N}_{\mathbf{s}}$ sites connected to s by edges
    - Grid graph 4-connected or 8-connected
  - Write smoothness term analogously as

$$\sum_{\mathbf{s} \in \mathcal{S}} \sum_{\mathbf{n} \in \mathcal{N}\mathbf{s}} (u_{\mathbf{s}}-u_{\mathbf{n}})^2 + (v_{\mathbf{s}}-v_{\mathbf{n}})^2$$

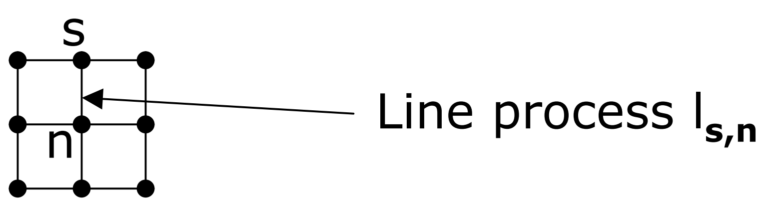


Line Process

- Variable smoothness penalty depending on binary estimate of discontinuity $l_{s,n}$

$$\sum_{s \in \mathcal{S}} \sum_{n \in \mathcal{N}s} [\alpha_s (1 - l_{s,n})((u_s - u_n)^2 + (v_s - v_n)^2) + \beta_s l_{s,n}]$$

  - With α_s, β_s constants controlling smoothness
- Minimization problem no longer as simple
  - Graduated non-convexity (GNC)



Robust Regularization

- Both smoothness and data constraints can be violated
  - Result not smooth at certain locations
    - Addressed by line process
  - Data values bad at certain locations
    - E.g., specularities, occlusions
    - Not addressed by line process
- Unified view: model both smoothness and data terms using robust error measures
  - Replace quadratic error which is sensitive to outliers

Robust Formulation

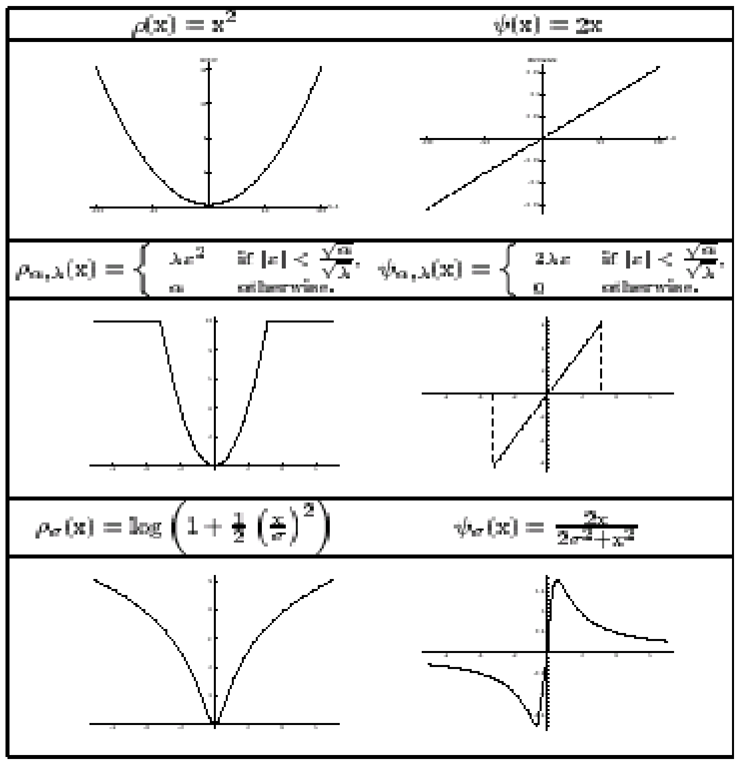
- Simply replace quadratic terms with robust error function ρ

$$\sum_{s \in \mathcal{S}} [\rho_1(I_x \cdot u_s + I_y \cdot v_s + I_t)$$
$$+ \lambda \sum_{n \in \mathcal{N}s} [\rho_2(u_s - u_n) + \rho_2(v_s - v_n)]]$$

  - In practice often estimate first term over small region around s
- Some robust error functions
  - Truncated linear: $\rho_\tau(x) = \min(\tau, x)$
  - Truncated quadratic: $\rho_\tau(x) = \min(\tau, x^2)$
  - Lorentzian: $\rho_\sigma(x) = \log(1 + ½(x/\sigma)^2)$

Influence Functions

- Useful to think of error functions in terms of degree to which a given value affects the result



Relation to Line Process

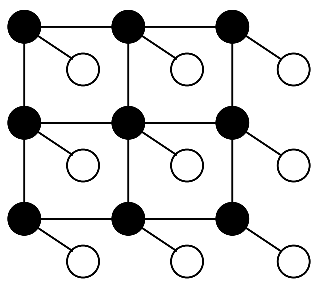
- Can think of robust error function as performing "outlier rejection"
  - Influence (near) zero for outliers but non-zero for inliers
- Line process makes a binary inlier/outlier decision
  - Based on external process or on degree of difference between estimated values
- Both robust estimation and line process formulations local characterizations

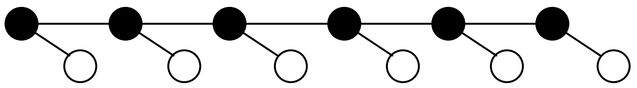
Relationship to MRF Models

- Markov random field (MRF)
  - Collection of random variables
  - Graph structure models spatial relations with local neighborhoods (Markov property)
    - Explicit dependencies among pixels
- Widely used in low-level vision problems
  - Stereo, motion, segmentation
- Seek best label for each pixel
  - Bayesian model, e.g., MAP estimation
- Common to consider corresponding energy minimization problems

Markov Random Fields in Vision

- Graph G=(V,E)
  - Assume vertices indexed 1, …, n
  - Observable variables $y=\{y_1, \ldots, y_n\}$
  - Unobservable variables $x=\{x_1, \ldots, x_n\}$
  - Edges connect each x_i to certain neighbors $\mathcal{N}_{xi}$
  - Edges connect each x_i to y_i
  - Consider cliques of size 2
    - Recall clique is fully connected sub-graph
    - 4-connected grid or 2-connected chain





MRF Models in Vision

- Prior P(x) factors into product of functions over cliques
  - Due to Hammersly-Clifford Theorem

$$P(x) = \prod_{c} \Psi_{c}(x_{c})$$

    - Ψ_c termed clique potential, of form $\exp(-V_c)$
  - For clique size 2 (cliques correspond to edges)

$$P(x) = \prod_{i,j} \Psi_{ij}(x_i,x_j)$$

- Probability of hidden and observed values

$$P(x,y) = \prod_{i,j} \Psi_{ij}(x_i,x_j) \prod_{i} \Psi_{ii}(x_i,y_i)$$

  - Given particular clique energy V_{ij} and observed y, seek values of x maximizing P(x,y)

Markov Property

- Neighborhoods completely characterize conditional distributions
  - Solving a global problem with local relationships
- Probability of values over subset S given remainder same as for that subset given its neighborhood
  - Given $S \subset V$ and $S^c = V - S$

    $P(x_{\mathbf{S}} \mid x_{\mathbf{S^c}}) = P(x_{\mathbf{S}} \mid \mathcal{N}_{\mathbf{x_S}})$
- Conceptually and computationally useful

MRF Estimation

- Various ways of maximizing probability
  - Common to use MAP estimate $\text{argmax}_{\mathbf{x}}\ P(x|y)$

    $$\text{argmax}_{\mathbf{x}} \prod_{\mathbf{i,j}} \Psi_{\mathbf{ij}}(x_{\mathbf{i}},x_{\mathbf{j}}) \prod_{\mathbf{i}} \Phi_{\mathbf{i}}(x_{\mathbf{i}},y_{\mathbf{i}})$$

- Probabilities hard to compute with
  - Use logs (or often negative log)

    $$\text{argmin}_{\mathbf{x}} \sum_{\mathbf{i,j}} V_{\mathbf{ij}}(x_{\mathbf{i}},x_{\mathbf{j}}) + \sum_{\mathbf{i}} D_{\mathbf{i}}(x_{\mathbf{i}},y_{\mathbf{i}})$$

- In energy function formulation often think of assigning best label $f_i \in \mathcal{L}$ to each node v_i given data y_i

  $$\text{argmin}_{\mathbf{f}} \left[\sum_{\mathbf{i}} D(y_{\mathbf{i}},f_{\mathbf{i}}) + \sum_{\mathbf{i,j}} V(f_{\mathbf{i}},f_{\mathbf{j}})\right]$$

Almost Same as Regularization

- Summation of data and smoothness terms

$$\mathrm{argmin}_{\mathbf{f}}\ [\sum_{\mathbf{i}} D(y_{\mathbf{i}},f_{\mathbf{i}}) + \sum_{\mathbf{i,j}} V(f_{\mathbf{i}},f_{\mathbf{j}})]$$

$$\mathrm{argmin}_{\mathbf{f}}\ \sum_{\mathbf{s}\in\mathcal{S}} [\rho_{\mathbf{1}}(d_{\mathbf{s}},f_{\mathbf{s}}) + \lambda \sum_{\mathbf{n}\in\mathcal{N}\mathbf{s}} [\rho_{\mathbf{2}}(f_{\mathbf{s}}-f_{\mathbf{n}})]]$$

  – Data term D vs. robust data function ρ_1
  – Clique term V vs. robust smoothness function ρ_2
    - Over cliques rather than neighbors of each site
    - Nearly same definitions on four connected grid
- Probabilistic formulation particularly helpful for learning problems
  – Parameters of D, V or even form of D,V

Common Clique Energies

- Enforce "smoothness", robust to outliers
  - Potts model
    - Same or outlier (based on label identity)

      $V_\tau(f_i,f_j) = 0$ when $f_i=f_j$, τ otherwise
  - Truncated linear model
    - Small linear change or outlier (label difference)

      $V_{\sigma,\tau}(f_i,f_j) = \min(\tau, \sigma|f_i-f_j|)$
  - Truncated quadratic model
    - Small quadratic change or outlier (label difference)

      $V_{\sigma,\tau}(f_i,f_j) = \min(\tau, \sigma|f_i-f_j|^2)$

1D Graphs (Chains)

- Simpler than 2D for illustration purposes
- Fast polynomial time algorithms
- Problem definition
  - Sequence of nodes V=(1, …, n)
  - Edges between adjacent pairs (i, i+1)
  - Observed value y_i at each node
  - Seek labeling $f=(f_1, \ldots, f_n)$, $f_i \in \mathcal{L}$, minimizing

$$\sum_i [D(y_i,f_i) + V(f_i,f_{i+1})] \quad \text{(note } V(f_n,f_{n+1})=0\text{)}$$

- Contrast with smoothing by convolution

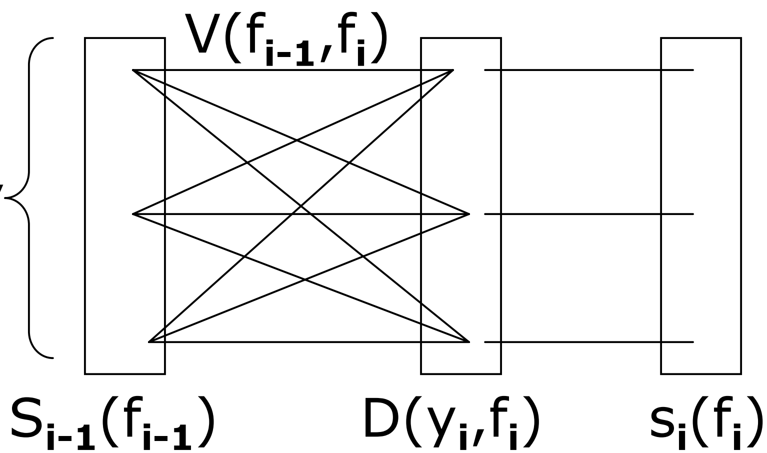
d_i	1	3	2	1	3	12	10	11	10	12
f_i	2	2	2	2	2	11	11	11	11	11

Viterbi Recurrence

- Don't need explicit min over $f=(f_1, \ldots, f_n)$
  - Instead recursively compute

    $s_i(f_i) = D(y_i,f_i) + \min_{f_{i-1}} (s_{i-1}(f_{i-1}) + V(f_{i-1},f_i))$
  - Note $s_i(f_i)$ for given i encodes a lowest cost label sequence ending in state f_i at that node

Possible labels, values of f_i



Viterbi Algorithm

- Find a lowest cost assignment f1, …, fn
- Initialize

  $s_1(f_1) = D(y_1,f_1)+\pi$, with π cost of f_1 if not uniform
- Recurse

  $s_i(f_i) = D(y_i,f_i) + \min_{f_{i-1}} (s_{i-1}(f_{i-1}) + V(f_{i-1},f_i))$

  $b_i(f_i) = \operatorname{argmin}_{f_{i-1}} (s_{i-1}(f_{i-1}) + V(f_{i-1},f_i))$
- Terminate

  $\min_{f_n} s_n(f_n)$, cost of cheapest path (neg log prob)

  $f_n{}^* = \operatorname{argmin}_{f_n} s_n(f_n)$
- Backtrack

  $f_{n-1}{}^* = b_n(f_n)$

Viterbi Algorithm

- For sequence of n data elements, with m possible labels per element
  - Compute $s_i(f_i)$ for each element using recurrence
    - $O(nm^2)$ time
  - For final node compute f_n minimizing $s_n(f_n)$
  - Trace back from node back to first node
    - Minimizers computed when computing costs on "forward" pass
- First step dominates running time
- Avoid searching exponentially many paths

Large Label Sets Problematic

- Viterbi slow with large number of labels
  - $O(m^2)$ term in calculating $s_i(f_i)$
- For our problems V usually has a special form so can compute in linear time
  - Consider linear clique energy

    $s_i(f_i) = D(y_i, f_i) + \min_{f_{i-1}} (s_{i-1}(f_{i-1}) + |f_{i-1} - f_i|)$
  - Minimization term is precisely the distance transform DTs_{i-1} of a function considered earlier
    - Which can compute in linear time
  - But linear model not robust
    - Can extend to truncated linear

Truncated Distance Cost

- Avoid explicit $\min_{f_{i-1}}$ for each f_i
  - Truncated linear model

    $$\min_{\mathbf{f_{i-1}}} (s_{\mathbf{i-1}}(f_{\mathbf{i-1}}) + \min(\tau,|f_{\mathbf{i-1}}-f_{\mathbf{i}}|))$$
  - Factor $f_{\mathbf{i}}$ out of minimizations over $f_{\mathbf{i-1}}$

    $$\min(\min_{\mathbf{f_{i-1}}}(s_{\mathbf{i-1}}(f_{\mathbf{i-1}})+\tau), \min_{\mathbf{f_{i-1}}}(s_{\mathbf{i-1}}(f_{\mathbf{i-1}})+|f_{\mathbf{i-1}}-f_{\mathbf{i}}|))$$

    $$\min(\min_{\mathbf{f_{i-1}}}(s_{\mathbf{i-1}}(f_{\mathbf{i-1}})+\tau), DT_{\mathbf{s_{i-1}}}(f_{\mathbf{i}}))$$
- Analogous for truncated quadratic model
- Similar for Potts model except no need for distance transform
- O(mn) algorithm for best label sequence

Belief Propagation

- Local message passing scheme in graph
  - Every node in parallel computes messages to send to neighbors
    - Iterate time-steps, t, until convergence
- Various message updating schemes
  - Here consider max product for undirected graph
    - Becomes min sum using costs (neg log probs)
  - Message $m_{i,j,t}$ sent from node i to j at time t

$$m_{i,j,t}(f_j) = \min_{f_i} [V(f_i,f_j)+D(y_i,f_i) + \sum_{k \in \mathcal{N}_i \backslash j} m_{k,i,t-1}(f_i)]$$

Belief Propagation

- After message passing "converges" at iteration T
  - Each node computes final value based on neighbors

$$b_i(f_i) = D(y_i, f_i) + \sum_{k \in \mathcal{N}_i} m_{r,i,T}(f_i)$$

  - Select label f_i minimizing b_i for each node
    - Corresponds to maximizing belief (probability)
- For singly-connected chain node generally has two neighbors i-1 and i+1

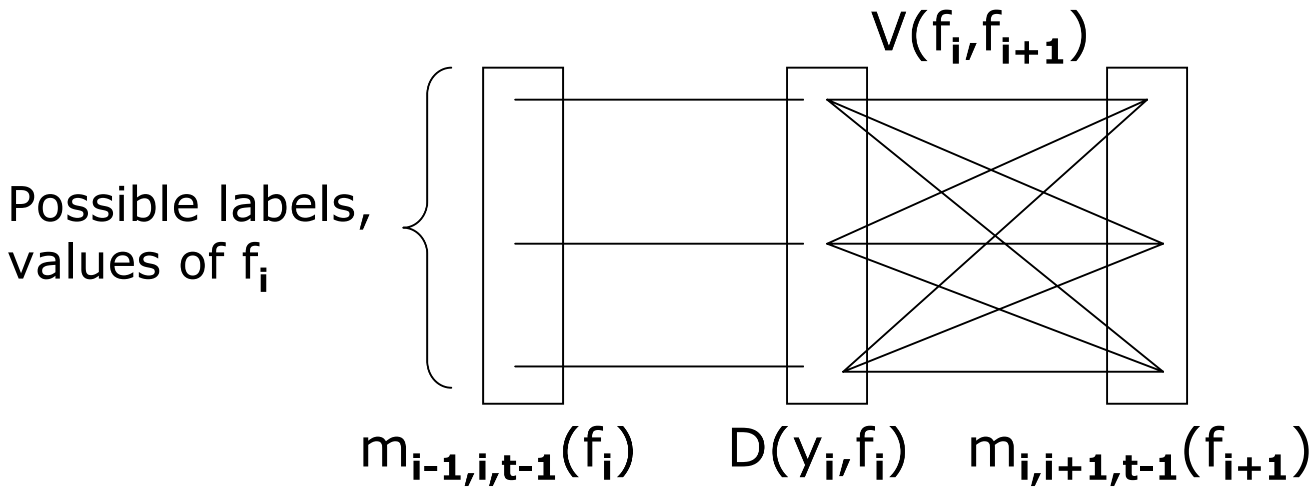
$$m_{i,i-1,t}(f_{i-1}) = \min_{f_i} [V(f_i, f_{i-1}) + D(y_i, f_i) + m_{i+1,i,t-1}(f_i)]$$

  - Analogous for i+1 neighbor

Belief Propagation on a Chain

- Message passed from i to i+1

  $m_{i,i+1,t}(f_{i+1}) = \min_{fi} [V(f_i,f_{i+1})+D(y_i,f_i)+m_{i-1,i,t-1}(f_i)]$
- Note relation to Viterbi recursion
- Can show BP converges to same minimum as Viterbi for chain (if unique min)



Min Sum Belief Prop Algorithm

- For chain, two messages per node
  - Node i sends messages $m_{i,l}$ to left $m_{i,r}$ to right
  - Initialize: $m_{i,l,0}=m_{i,r,0}=(0, \ldots, 0)$ for all nodes i
  - Update messages, for t from 1 to T

    $m_{i,l,t}(f_l) = \min_{f_i} [V(f_i,f_l)+D(y_i,f_i)+m_{r,i,t-1}(f_i)]$

    $m_{i,r,t}(f_r) = \min_{f_i} [V(f_i,f_r)+D(y_i,f_i)+m_{l,i,t-1}(f_i)]$
  - Compute belief at each node

    $b_i(f_i)=D(y_i,f_i)+m_{r,i,T}(f_i)+m_{l,i,T}(f_i)$
  - Select best at each node

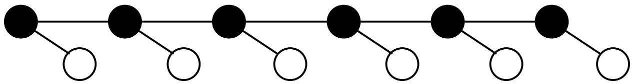
    $\mathrm{argmin}_{f_i}\, b_i(f_i)$
- For chain, global min of $\sum_i[D(y_i,f_i)+V(f_i,f_{i+1})]$

Relation to HMM

- Hidden Markov model
  - Set of unobservable (hidden) states
  - Sequence of observed values, y_i
  - Transitions between states are Markov
    - Depend only on previous state (or fixed number)
    - State transition matrix (costs or probabilities)
  - Distribution of possible observed values for each state
  - Given y_i determine best state sequence
- Widely used in speech recognition and temporal modeling

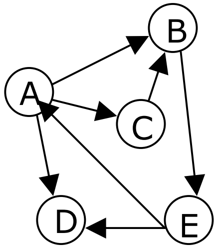
Hidden Markov Models

- Two different but equivalent views
  - Sequence of unobservable random variables and observable values
    - 1D MRF with label set
    - Penalties $V(f_i,f_j)$, data costs $D(y_i,f_i)$

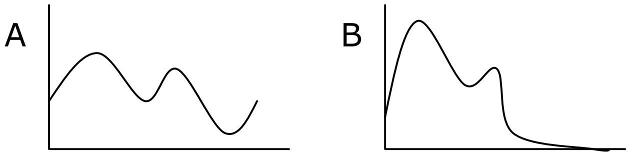


  - Hidden non-deterministic state machine
    - Distribution over observable values for each state

V



D



Using HMM's

- Three classical problems for HMM
  - Given observation sequence $Y=y_1, \ldots, y_n$ and HMM $\lambda=(D,V,\pi)$

1. Compute $P(Y|\lambda)$, probability of observing Y given the model
   - Alternatively cost (negative log prob)
2. Determine the best state sequence $x_1, \ldots, x_n$ given Y
   - Various definitions of best, one is MAP estimate $\text{argmax}_X P(X|Y,\lambda)$ or min cost
3. Adjust model $\lambda=(D,V,\pi)$ to maximize $P(Y|\lambda)$
   - Learning problem often solved by EM

HMM Inference or Decoding

- Determine the best state sequence X given observation sequence Y
  - MAP (maximum a posteriori) estimate

    $\arg\max_{\mathbf{X}} P(X|Y,\lambda)$

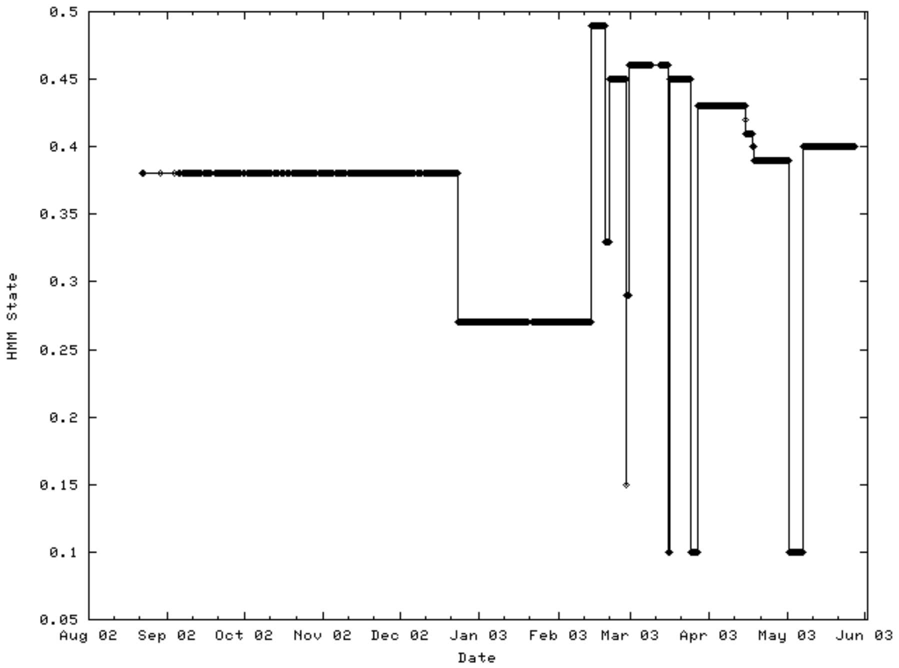
    - Equivalently minimize cost, negative log prob
    - Computed using Viterbi or max-product (min-sum) belief propagation
  - Most likely state at each time $P(X_{\mathbf{t}}|Y_{\mathbf{1}},\ldots,Y_{\mathbf{t}},\lambda)$
    - Maximize probability of states individually
    - Computed using forward-backward procedure or sum-product belief propagation

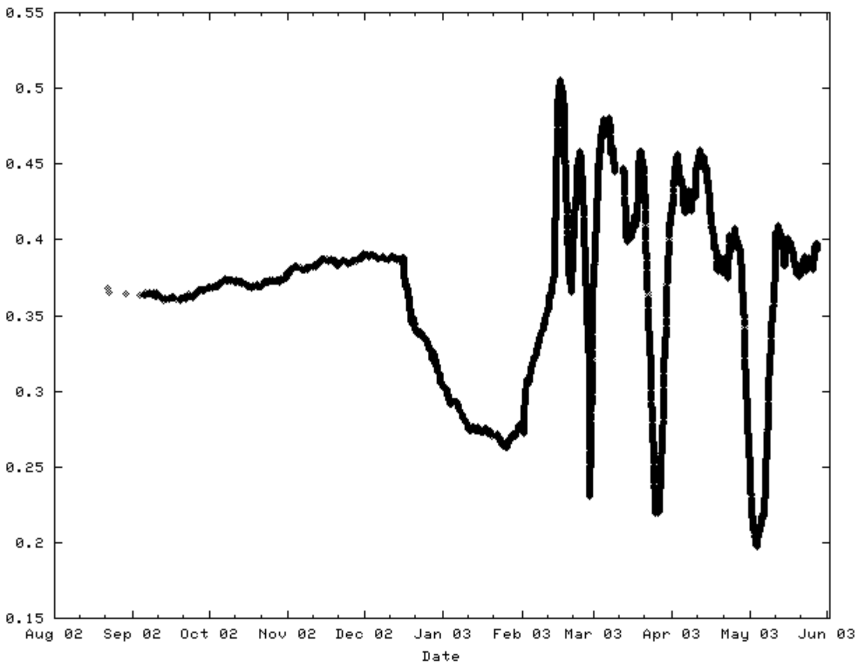
1D HMM Example

- Estimate bias of "changing coin" from sequence of observed {H,T} values
  - Use MAP formulation
    - Find lowest cost state sequence
- States correspond to possible bias values, e.g., .10, …, .90 (large state space)
  - Data costs $-\log P(H|x_i)$, $-\log P(T|x_i)$
- Used to analyze time varying popularity of item downloads at Internet Archive
  - Each visit results in download or not (H/T)

1D HMM Example

- Truncated linear penalty term $V(f_i,f_j)$
  - Contrast with smoothing
  - Particularly hard task for 0-1 valued data



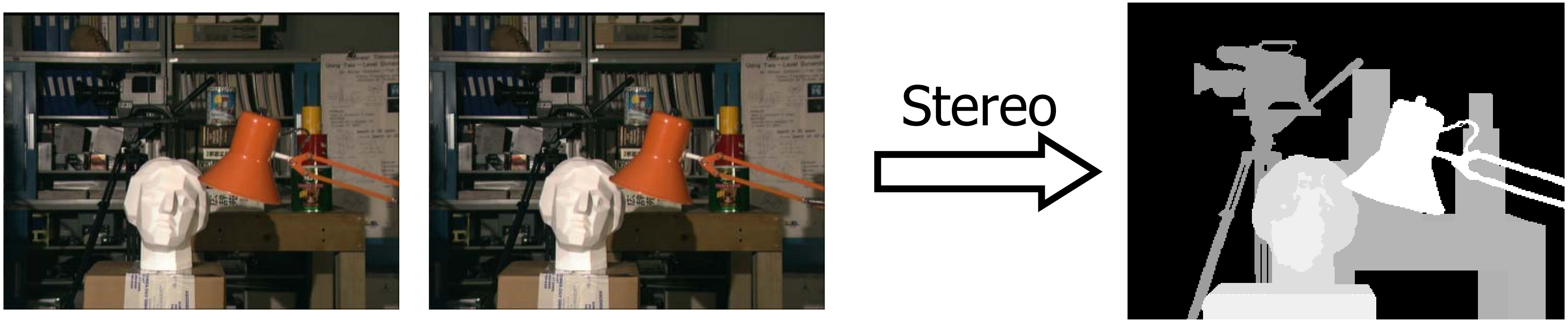


Algorithms for Grids (2D)

- Polynomial time for binary label set or for convex cost function $V(f_i,f_j)$
  - Compute minimum cut in certain graph
  - NP hard in general (reduction from multi-way cut)
- Approximation methods (not global min)
  - Graph cuts and "expansion moves"
  - Loopy belief propagation
  - Many other local minimization techniques
    - Monte Carlo sampling methods, annealing, etc.
  - Consider graph cuts and belief propagation
    - Reasonably fast
    - Can characterize the local minimum

Evaluating 2D Energy Min Methods

- Benchmark for stereo data
  - Estimate a disparity d at each pixel
    - Inversely proportional to depth, closer things move farther (larger disparity between images)
    - Data term $D(y_i,f_i)=(I(y_{i,u},y_{i,v})-I(y_{i,u}+f_i,y_{i,v}))^2$
  - Rectified images so disparities are horizontal (u)
  - Ground truth and error measure



Loopy Belief Propagation

- Apply belief propagation to graph with loops such as MRF
  - No longer globally optimal result
  - Results suggesting good local minimum in sense that no better minima "nearby"
- In practice works quite well
  - Some of best stereo results obtained using LBP
  - Relatively slow – minutes per image
    - Number of iterations proportional to image diameter – propagate messages across grid
  - Alternative approaches such as removing loops

Loopy Belief Prop on Grid

- Min sum algorithm, analogous to chain
  - Four message types (dirs) $m_{i,l}$, $m_{i,r}$, $m_{i,u}$, $m_{i,d}$
  - Initialize, $m_{i,l,0}=m_{i,r,0}= m_{i,u,0}= m_{i,d,0}=(0, \ldots, 0)$
  - Update messages, for t from 1 to T,

    $$m_{i,l,t}(f_l) = \min_{fi} [V(f_i,f_l)+D(y_i,f_i)+m_{r,i,t-1}(f_i) + m_{u,i,t-1}(f_i)+ m_{d,i,t-1}(f_i)]$$

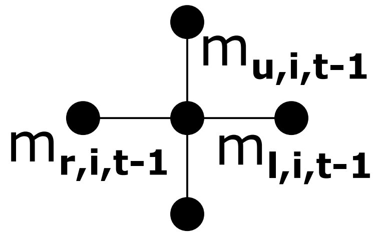
    - Analogous for other three directions r,u,d
  - Compute beliefs,

    $$b_i(f_i)=D(y_i,f_i)+m_{r,i,T}(f_i)+m_{l,i.,T}(f_i) +m_{u,i,T}(f_i)+ m_{d,i,T}(f_i)$$

  - Select best, $\text{argmin}_{fi}\ b_i(f_i)$

Loopy Belief Prop Schematic

- At time step t can think of each node
  - Using “incoming” messages from time t-1
  - Creating new “outgoing” messages
- Store 8 messages per node
  - Four directions at t-1 for others to use
  - Four directions at t for i to compute
  - Each message a cost for each label (distribution)

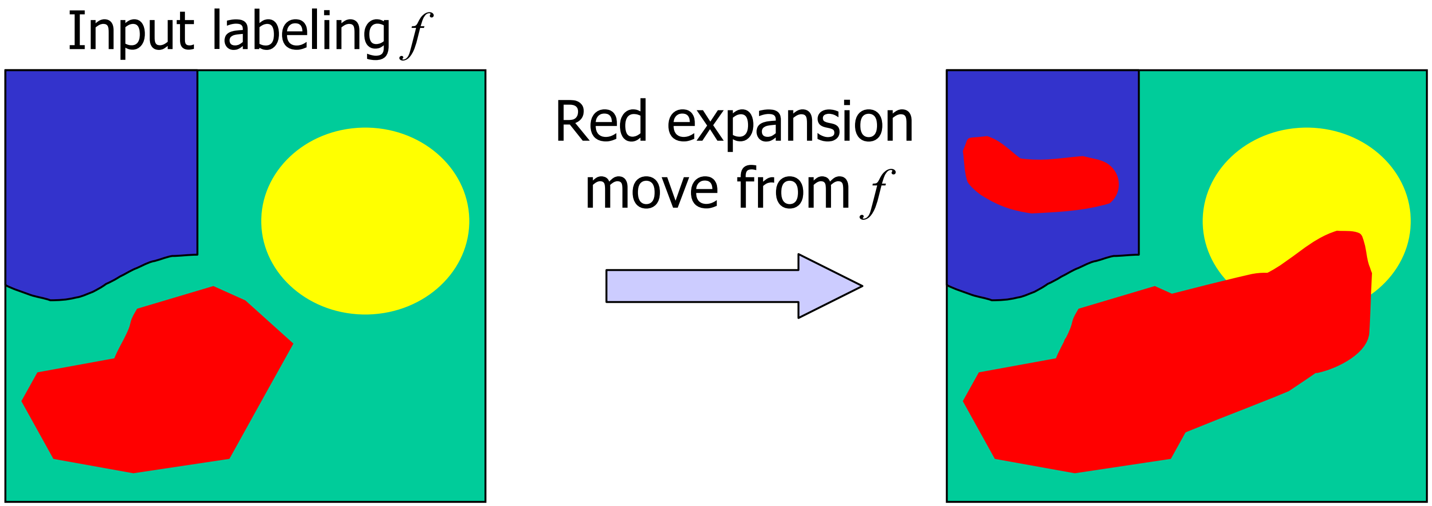


Compute $m_{i,d,t}$

Graph Cuts

- Two label problem can be solved using minimum cut in suitably defined graph
- For many labels use expansion move heuristic
  - Re-cast problem of finding labeling as sequence of binary problems
  - Given a labeling, seek best labeling that "expands" particular label α
    - Binary because just consider change to α and leave alone as options
  - Iterate repeatedly over labels

Expansion Moves



- Find red expansion move that most decreases energy
  - Move there, then find the best blue expansion move, etc
  - Done when no α-expansion move decreases the energy, for any label α
  - Rapidly computes a strong local minimum
- Nice theoretical properties

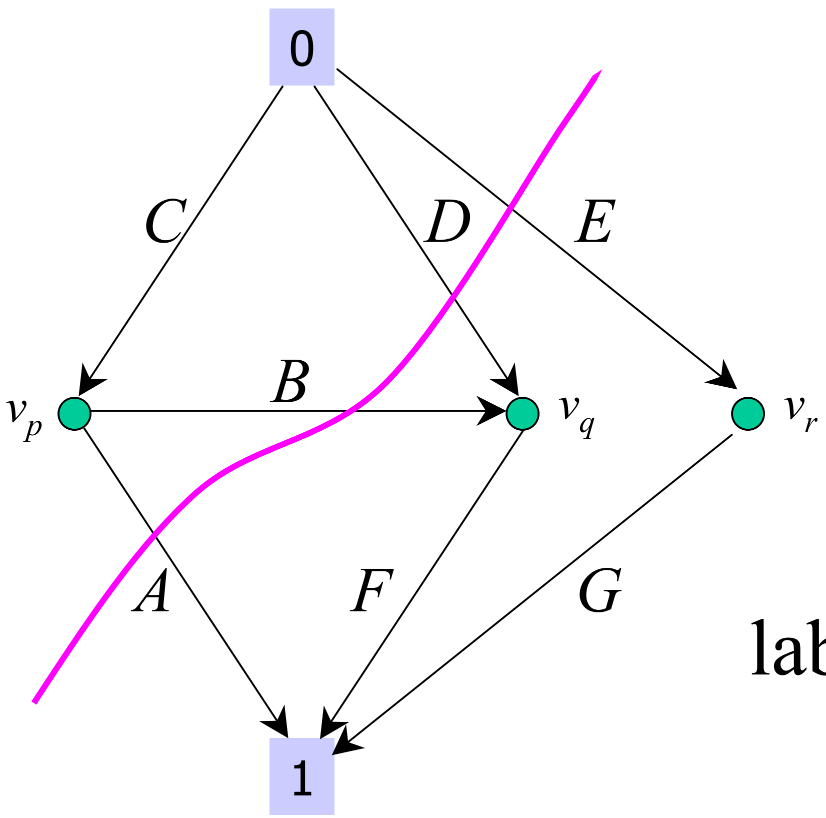
Expansion Move Algorithm

- The overall problem involves *k* labels, but the key sub-problem involves only 2
  - Minimize energy over all $O(2^n)$ labelings within a single α-expansion move from f
  - Each pixel p either keeps its old label f_p, or acquire the new label α
  - In practice few iterations over labels
- Approach: classical problem reduction
  - Reduction to computing the minimum cost s-t cut on the appropriate graph
    - Fast polynomial time algorithm

Graph Cuts for Expansion Moves

- Consider directed graph with non-negative edge weights and with two terminal nodes
  - Source (node 0) and sink (node 1)
- A *cut* is a partition of the nodes into two sets S, T such that $0 \in S$, $1 \in T$
  - Or, a set of edges that separate the terminals
  - Binary labeling of the non-terminal nodes!
- The cost of the cut is the sum of the weights of edges from S to T
  - There are fast ways to find the cut with the minimum cost, even on very large graphs

Graph Cut Illustration



$$S = \{0, v_p\}$$

$$T = \{1, v_q, v_r\}$$

$$\text{labeling} = \{v_p \leftarrow 0; v_q \leftarrow 1; v_r \leftarrow 1\}$$

$$\text{cost} = A + B + D + E$$

Graph Cut Example

Ground Truth Results

Graph Cut Results