Introduction to Probabilistic Graphical Models

Christoph Lampert

IST Austria (Institute of Science and Technology Austria)

IST AUSTRIA
Institute of Science and Technology

Schedule

Refresher of Probabilities
Introduction to Probabilistic Graphical Models
Probabilistic Inference
Learning Conditional Random Fields
MAP Prediction / Energy Minimization
Learning Structured Support Vector Machines

Links to slide download: `http://pub.ist.ac.at/~chl/courses/PGM_W16/`

Password for ZIP files (if any): `pgm2016`

Email for questions, suggestions or typos that you found: `chl@ist.ac.at`

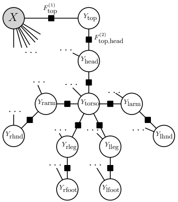
Supervised Learning Problem

- Given training examples $(x^1, y^1), \ldots, (x^N, y^N) \in \mathcal{X} \times \mathcal{Y}$
  $x \in \mathcal{X}$: input, *e.g.* image
  $y \in \mathcal{Y}$: structured output, *e.g.* human pose, sentence

Images: HumanEva dataset

Goal: be able to make predictions for new inputs, i.e. learn a function $f : \mathcal{X} \to \mathcal{Y}$.

Supervised Learning Problem



Step 1: define a proper graph structure of X and Y

Step 2: define a proper parameterization of $p(y|x;\theta)$

$$p(y|x;\theta) = \frac{1}{Z} e^{\sum_{i=1}^{d} \theta_i \phi_i(x,y)}$$

Step 3: learn parameters θ^* from training data

e.g. maximum likelihood

Step 4: for new $x \in \mathcal{X}$, make prediction

e.g. $y^* = \operatorname{argmax}_{y \in \mathcal{Y}} p(y|x;\theta^*)$

($\rightarrow$ today)

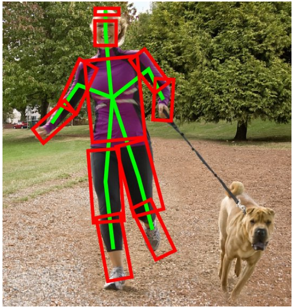
MAP Prediction / Energy Minimization

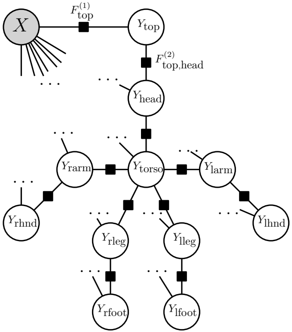
$\operatorname{argmax}_y p(y|x)$ / $\operatorname{argmin}_y E(y, x)$

MAP Prediction / Energy Minimization

- Exact Energy Minimization
  - Belief Propagation on chains/trees
  - Graph-Cuts for submodular energies
  - Integer Linear Programming
- Approximate Energy Minimization
  - Linear Programming Relaxations
  - Local Search Methods
    - Iterative Conditional Modes
    - Multi-label Graph Cuts
  - Simulated Annealing

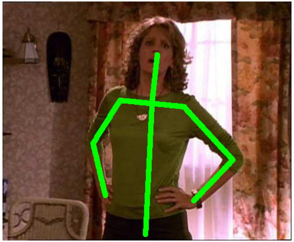
Example: Pictorial Structures / Deformable Parts Model





- **Tree-structured model** for articulated pose (Felzenszwalb and Huttenlocher, 2000), (Fischler and Elschlager, 1973), (Yang and Ramanan, 2013), (Pishchulin *et al.*, 2012)

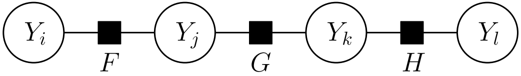
Example: Pictorial Structures / Deformable Parts Model



- most likely configuration $y^* = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, p(y|x) = \underset{y}{\operatorname{argmin}}\, E(y, x)$

Energy Minimization – Belief Propagation

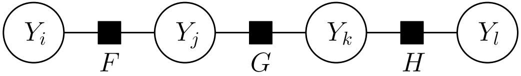
Chain model: same trick as for *inference*: **belief propagation**



$$\min_y E(y) = \min_{y_i,y_j,y_k,y_l} \quad E_F(y_i, y_j) + E_G(y_j, y_k) + E_H(y_k, y_l)$$

Energy Minimization – Belief Propagation

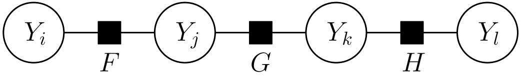
Chain model: same trick as for *inference*: **belief propagation**



$$\begin{aligned}\min_{y} E(y) &= \min_{y_i, y_j, y_k, y_l} \quad E_F(y_i, y_j) + E_G(y_j, y_k) + E_H(y_k, y_l) \\ &= \min_{y_i, y_j} \left[E_F(y_i, y_j) + \min_{y_k} \left[E_G(y_j, y_k) + \min_{y_l} E_H(y_k, y_l) \right] \right]\end{aligned}$$

Energy Minimization – Belief Propagation

Chain model: same trick as for *inference*: **belief propagation**

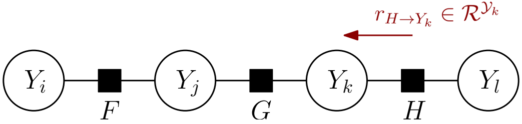


$$\min_y E(y) = \min_{y_i,y_j,y_k,y_l} \quad E_F(y_i,y_j) + E_G(y_j,y_k) + E_H(y_k,y_l)$$

$$= \min_{y_i,y_j} \Big[E_F(y_i,y_j) + \min_{y_k} \big[E_G(y_j,y_k) + \underbrace{\min_{y_l} E_H(y_k,y_l)}_{r_{H\to Y_k}(y_k)} \big]\Big]$$

Energy Minimization – Belief Propagation

Chain model: same trick as for *inference*: **belief propagation**



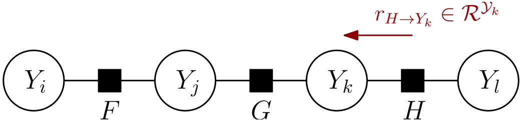
$$\min_{y} E(y) = \min_{y_i,y_j,y_k,y_l} \quad E_F(y_i,y_j) + E_G(y_j,y_k) + E_H(y_k,y_l)$$

$$= \min_{y_i,y_j} \left[E_F(y_i,y_j) + \min_{y_k} \left[E_G(y_j,y_k) + \underbrace{\min_{y_l} E_H(y_k,y_l)}_{r_{H\to Y_k}(y_k)} \right]\right]$$

$$= \min_{y_i,y_j} \left[E_F(y_i,y_j) + \min_{y_k} E_G(y_j,y_k) + r_{H\to Y_k}(y_k) \right]$$

Energy Minimization – Belief Propagation

Chain model: same trick as for *inference*: **belief propagation**



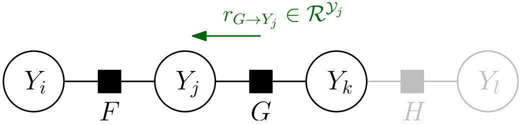
$$\min_y E(y) = \min_{y_i,y_j,y_k,y_l} \quad E_F(y_i,y_j) + E_G(y_j,y_k) + E_H(y_k,y_l)$$

$$= \min_{y_i,y_j} \left[E_F(y_i,y_j) + \min_{y_k} \left[E_G(y_j,y_k) + \underbrace{\min_{y_l} E_H(y_k,y_l)}_{r_{H\to Y_k}(y_k)} \right]\right]$$

$$= \min_{y_i,y_j} \left[E_F(y_i,y_j) + \underbrace{\min_{y_k} E_G(y_j,y_k) + r_{H\to Y_k}(y_k)}_{r_{G\to Y_j}(y_j)} \right]$$

Energy Minimization – Belief Propagation

Chain model: same trick as for *inference*: **belief propagation**



$$\min_y E(y) = \min_{y_i, y_j, y_k, y_l} \quad E_F(y_i, y_j) + E_G(y_j, y_k) + E_H(y_k, y_l)$$

$$= \min_{y_i, y_j} \left[E_F(y_i, y_j) + \min_{y_k} \left[E_G(y_j, y_k) + \underbrace{\min_{y_l} E_H(y_k, y_l)}_{r_{H \to Y_k}(y_k)} \right] \right]$$

$$= \min_{y_i, y_j} \left[E_F(y_i, y_j) + \underbrace{\min_{y_k} E_G(y_j, y_k) + r_{H \to Y_k}(y_k)}_{r_{G \to Y_j}(y_j)} \right]$$

$$= \min_{y_i, y_j} \left[E_F(y_i, y_j) + r_{G \to Y_j}(y_j) \right] \quad \ldots$$

Energy Minimization – Belief Propagation

Chain model: same trick as for *inference*: **belief propagation**

$$\min_y E(y) = \min_{y_i,y_j,y_k,y_l} \quad E_F(y_i,y_j) + E_G(y_j,y_k) + E_H(y_k,y_l)$$

$$= \min_{y_i,y_j}\Big[E_F(y_i,y_j) + \min_{y_k}\Big[E_G(y_j,y_k) + \underbrace{\min_{y_l} E_H(y_k,y_l)}_{r_{H\to Y_k}(y_k)}\Big]\Big]$$

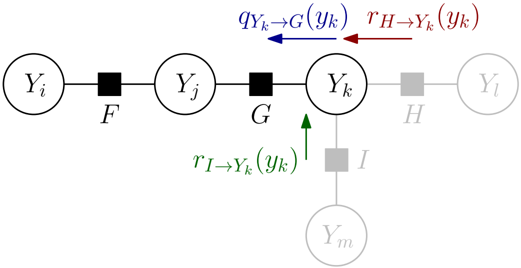
$$= \min_{y_i,y_j}\Big[E_F(y_i,y_j) + \underbrace{\min_{y_k} E_G(y_j,y_k) + r_{H\to Y_k}(y_k)}_{r_{G\to Y_j}(y_j)}\Big]$$

$$= \min_{y_i,y_j}\Big[E_F(y_i,y_j) + r_{G\to Y_j}(y_j)\Big] \quad \ldots$$

- actual argmax by backtracking which choices were maximal

Energy Minimization – Belief Propagation

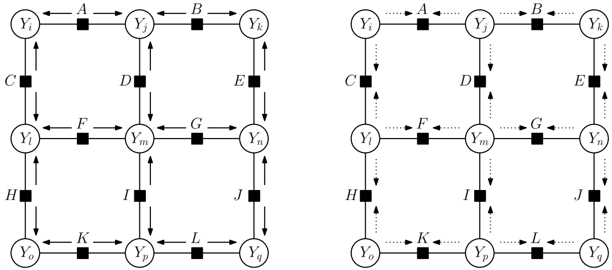
Tree models:



- $q_{H \to Y_k}(y_k) = \min_{y_l} E_H(y_k, y_l)$
- $q_{I \to Y_k}(y_k) = \min_{y_m} E_I(y_k, y_m)$
- $q_{Y_k \to G}(y_k) = q_{H \to Y_k}(y_k) + q_{I \to Y_k}(y_k)$

min-sum (more commonly **max-sum**) **belief propagation**

Belief Propagation in Cyclic Graphs



Loopy Max-Sum Belief Propagation

Same problem as in probabilistic inference:

- no guarantee of convergence
- no guarantee of optimality

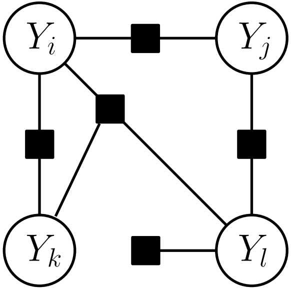
Some convergent variants exist, *e.g.* TRW-S [Kolmogorov, PAMI 2006]

Cyclic Graphs

In general, MAP prediction/energy minimization in models with cycles or higher-order terms is **intractable** (NP-hard).

Some important exceptions:

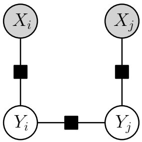
- low tree-width [Lauritzen, Spiegelhalter, 1988]
- **binary states, pairwise submodular interactions [Boykov, Jolly, 2001]**
- binary states, only pairwise interactions, planar graph [Globerson, Jaakkola, 2006]
- special (Potts $\mathcal{P}^n$) higher order factors [Kohli, Kumar, 2007]
- perfect graph structure [Jebara, 2009]



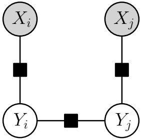
Submodular Energy Functions

- Binary variables: $\mathcal{Y}_i = \{0, 1\}$ for all $i \in \mathcal{V}$
- Energy function: unary and pairwise factors

$$E(y; x, w) = \sum_{i \in \mathcal{V}} E_i(y_i) + \sum_{(i,j) \in \mathcal{E}} E_{ij}(y_i, y_j)$$



Submodular Energy Functions



- Binary variables: $\mathcal{Y}_i = \{0, 1\}$ for all $i \in \mathcal{V}$
- Energy function: unary and pairwise factors

$$E(y; x, w) = \sum_{i \in \mathcal{V}} E_i(y_i) + \sum_{(i,j) \in \mathcal{E}} E_{ij}(y_i, y_j)$$

- Restriction 1 (without loss of generality):

$$E_i(y_i) \geq 0$$

(always achievable by adding a constant to E)

- Restriction 2 (**submodularity**):

$$\begin{aligned} E_{ij}(y_i, y_j) &= 0, && \text{if } y_i = y_j, \\ E_{ij}(y_i, y_j) &= E_{ij}(y_j, y_i) \geq 0, && \text{otherwise.} \end{aligned}$$

"neighbors prefer to have the same labels"

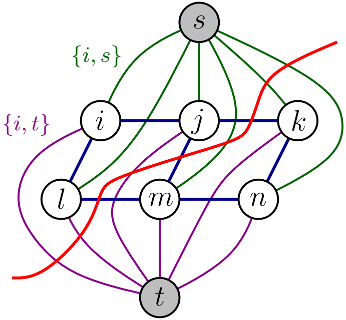
Graph-Cuts Algorithm for Submodular Energy Minimization [Greig *et al.*, 1989]

If conditions are fulfilled, energy minimization can be performed by a solving s-t-**mincut**:

- construct auxiliary undirected graph
- one node $\{i\}_{i \in V}$ per variable
- two extra nodes: source s, sink t
- weighted edges

Edge	weight
$\{i, j\}$	$E_{ij}(y_i = 0, y_j = 1)$
$\{i, s\}$	$E_i(y_i = 1)$
$\{i, t\}$	$E_i(y_i = 0)$

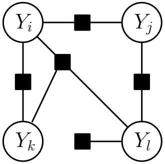
- find s-t-cut of minimal weight (polynomial time using max-flow theorem)



From minimal weight cut we recover labeling of minimal energy:

- $y_i^* = 1$ if edge $\{i, s\}$ is cut. Otherwise $y_i^* = 0$

Integer Linear Programming (ILP)



General energy $E(y) = \sum_F E_F(y_F)$

- variables with more than 2 states
- higher-order factors (more than 2 variables)
- non-submodular factors

Formulate as integer linear program (ILP)

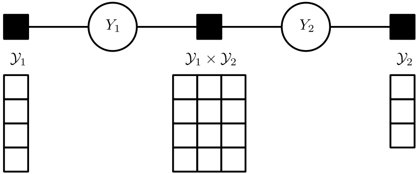
- linear objective function
- linear constraints
- variables to optimize over are integer-valued

ILPs are in general NP-hard, but some individual instances can be solved

- standard optimization toolboxes: *e.g.* CPLEX, Gurobi, COIN-OR, ...

Integer Linear Programming (ILP)

Example:

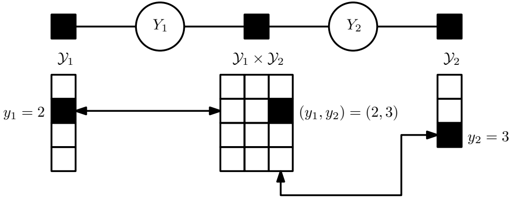


Encode assignment in indicator variables:

- $z_1 \in \{0,1\}^{\mathcal{Y}_1}$ $\qquad z_{1;k} = 1 \;\Leftrightarrow\; [\![y_1 = k]\!]$
- $z_2 \in \{0,1\}^{\mathcal{Y}_2}$ $\qquad z_{2;l} = 1 \;\Leftrightarrow\; [\![y_2 = l]\!]$
- $z_{12} \in \{0,1\}^{\mathcal{Y}_1 \times \mathcal{Y}_2}$ $\qquad z_{12;kl} = 1 \;\Leftrightarrow\; [\![y_1 = k \wedge y_2 = l]\!]$

Integer Linear Programming (ILP)

Example:



Encode assignment in indicator variables:

- $z_1 \in \{0,1\}^{\mathcal{Y}_1}, \qquad z_2 \in \{0,1\}^{\mathcal{Y}_2}, \qquad z_{12} \in \{0,1\}^{\mathcal{Y}_1 \times \mathcal{Y}_2}$

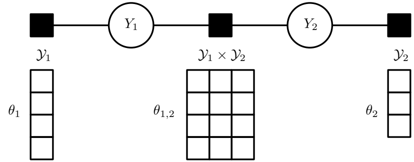
Consistency Constraints:

$$\sum_{k \in \mathcal{Y}_1} z_{1;k} = 1, \quad \sum_{l \in \mathcal{Y}_2} z_{2;l} = 1, \quad \sum_{k,l \in \mathcal{Y}_1 \times \mathcal{Y}_2} z_{12;kl} = 1 \quad \text{(indicator property)}$$

$$\sum_{k \in \mathcal{Y}_1} z_{12;kl} = z_{2;l} \qquad \sum_{l \in \mathcal{Y}_2} z_{12;kl} = z_{1;k} \qquad \text{(consistency)}$$

Integer Linear Programming (ILP)

Example: $E(y_1, y_2) = E_1(y_1) + E_{12}(y_1, y_2) + E_2(y_2)$



Define coefficient vectors:

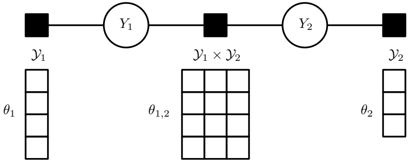
- $\theta_1 \in \mathbb{R}^{\mathcal{Y}_1}$ $\theta_{1;k} = E_1(k)$
- $\theta_2 \in \mathbb{R}^{\mathcal{Y}_2}$ $\theta_{2;l} = E_2(l)$
- $\theta_{12} \in \mathbb{R}^{\mathcal{Y}_1 \times \mathcal{Y}_2}$ $\theta_{12;kl} = E_{1,2}(k, l)$

Energy is a linear function of unknown z:

$$E(y_1, y_2) = \sum_{i \in V} \sum_{k \in \mathcal{Y}_i} \theta_{i;k} [\![y_i = k]\!] + \sum_{i,j \in \mathcal{E}} \sum_{k,l \in \mathcal{Y}_i \times \mathcal{Y}_j} \theta_{ij;kl} [\![y_i = k \wedge y_j = l]\!]$$

Integer Linear Programming (ILP)

Example: $E(y_1, y_2) = E_1(y_1) + E_{12}(y_1, y_2) + E_2(y_2)$



Define coefficient vectors:

- $\theta_1 \in \mathbb{R}^{\mathcal{Y}_1}$ $\quad \theta_{1;k} = E_1(k)$
- $\theta_2 \in \mathbb{R}^{\mathcal{Y}_2}$ $\quad \theta_{2;l} = E_2(l)$
- $\theta_{12} \in \mathbb{R}^{\mathcal{Y}_1 \times \mathcal{Y}_2}$ $\quad \theta_{12;kl} = E_{1,2}(k, l)$

Energy is a linear function of unknown z:

$$E(y_1, y_2) = \sum_{i \in V} \sum_{k \in \mathcal{Y}_i} \theta_{i;k} z_{i;k} + \sum_{(i,j) \in \mathcal{E}} \sum_{(k,l) \in \mathcal{Y}_i \times \mathcal{Y}_j} \theta_{ij;kl} z_{ij;kl}$$

Integer Linear Programming (ILP)

$$\min_{\mathbf{z}} \sum_{i \in V} \sum_{k \in \mathcal{Y}_i} \theta_{i;k} z_{i;k} + \sum_{(i,j) \in \mathcal{E}} \sum_{(k,l) \in \mathcal{Y}_i \times \mathcal{Y}_j} \theta_{ij;kl} z_{ij;kl}$$

subject to

$$z_{i;k} \in \{0,1\} \quad \text{for all } i \in V, \ \forall k \in \mathcal{Y}_i,$$

$$z_{ij;kl} \in \{0,1\} \quad \text{for all } (i,j) \in \mathcal{E}, (k,l) \in \mathcal{Y}_i \times \mathcal{Y}_j,$$

$$\sum_{k \in \mathcal{Y}_i} z_{i;k} = 1, \quad \text{for all } i \in V,$$

$$\sum_{k,l \in \mathcal{Y}_i \times \mathcal{Y}_j} z_{ij;kl} = 1, \quad \text{for all } (i,j) \in \mathcal{E},$$

$$\sum_{k \in \mathcal{Y}_i} z_{ij;kl} = z_{j;l} \quad \text{for all } (i,j) \in \mathcal{E}, \ l \in \mathcal{Y}_j,$$

$$\sum_{l \in \mathcal{Y}_j} z_{ij;kl} = z_{i;k} \quad \text{for all } (i,j) \in \mathcal{E}, \ k \in \mathcal{Y}_i,$$

Integer Linear Programming (ILP)

$$\min_{\mathbf{z}} \quad \sum_{i \in V} \sum_{k \in \mathcal{Y}_i} \theta_{i;k} z_{i;k} + \sum_{(i,j) \in \mathcal{E}} \sum_{(k,l) \in \mathcal{Y}_i \times \mathcal{Y}_j} \theta_{ij;kl} z_{ij;kl}$$

subject to

$$z_{i;k} \in \{0,1\} \qquad \text{for all } i \in V,\ \forall k \in \mathcal{Y}_i,$$

$$z_{ij;kl} \in \{0,1\} \qquad \text{for all } (i,j) \in \mathcal{E}, (k,l) \in \mathcal{Y}_i \times \mathcal{Y}_j,$$

$$\sum_{k \in \mathcal{Y}_i} z_{i;k} = 1, \quad \text{for all } i \in V,$$

$$\sum_{k,l \in \mathcal{Y}_i \times \mathcal{Y}_j} z_{ij;kl} = 1, \quad \text{for all } (i,j) \in \mathcal{E},$$

$$\sum_{k \in \mathcal{Y}_i} z_{ij;kl} = z_{j;l} \quad \text{for all } (i,j) \in \mathcal{E},\ l \in \mathcal{Y}_j,$$

$$\sum_{l \in \mathcal{Y}_j} z_{ij;kl} = z_{i;k} \quad \text{for all } (i,j) \in \mathcal{E},\ k \in \mathcal{Y}_i,$$

NP-hard to solve because of integrality constraints.

Linear Programming (LP) Relaxation

$$\min_{\mathbf{z}} \sum_{i \in V} \sum_{k \in \mathcal{Y}_i} \theta_{i;k} z_{i;k} + \sum_{(i,j) \in \mathcal{E}} \sum_{(k,l) \in \mathcal{Y}_i \times \mathcal{Y}_j} \theta_{ij;kl} z_{ij;kl}$$

subject to

$$\cancel{z_{i;k} \in \{0,1\}} \quad z_{i;k} \in [0,1] \quad \text{for all } i \in V,\ \forall k \in \mathcal{Y}_i,$$

$$\cancel{z_{ij;kl} \in \{0,1\}} \quad z_{ij;kl} \in [0,1] \quad \text{for all } (i,j) \in \mathcal{E}, (k,l) \in \mathcal{Y}_i \times \mathcal{Y}_j,$$

$$\sum_{k \in \mathcal{Y}_i} z_{i;k} = 1, \quad \text{for all } i \in V,$$

$$\sum_{k,l \in \mathcal{Y}_i \times \mathcal{Y}_j} z_{ij;kl} = 1, \quad \text{for all } (i,j) \in \mathcal{E},$$

$$\sum_{k \in \mathcal{Y}_i} z_{ij;kl} = z_{j;l} \quad \text{for all } (i,j) \in \mathcal{E},\ l \in \mathcal{Y}_j,$$

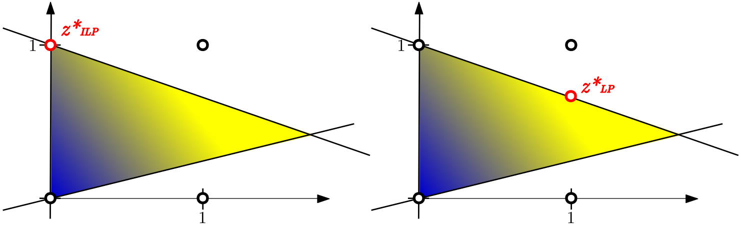
$$\sum_{l \in \mathcal{Y}_j} z_{ij;kl} = z_{i;k} \quad \text{for all } (i,j) \in \mathcal{E},\ k \in \mathcal{Y}_i,$$

Relax constraints $\rightarrow$ optimization problem becomes tractable

Linear Programming (LP) Relaxation

Solution $\mathbf{z}^*_{LP}$ might have fractional values

- $\rightarrow$ no corresponding labeling $y \in \mathcal{Y}$
- $\rightarrow$ round LP solution to $\{0, 1\}$ values



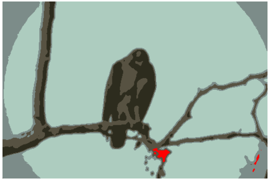
Problem:

- rounded solution usually not optimal, *i.e.* not identical to ILP solution

LP relaxations perform approximate energy minimization

Linear Programming (LP) Relaxation

Example: color quantization



Example: stereo reconstruction

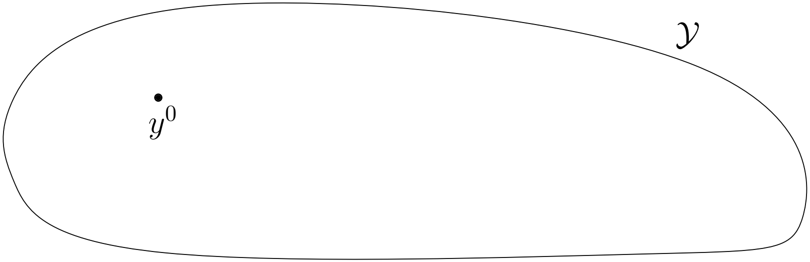
Images: Berkeley Segmentation Dataset

Local Search

Avoid getting fractional solutions: energy minimization by **local search**

Local Search

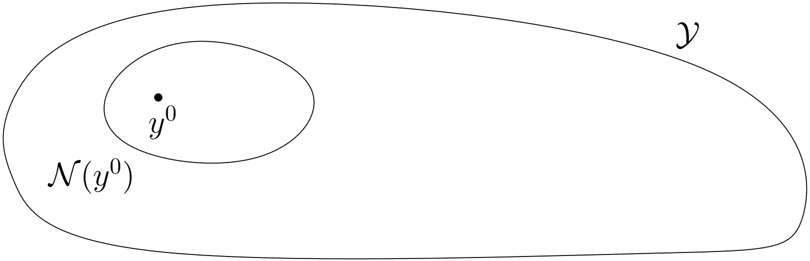
Avoid getting fractional solutions: energy minimization by **local search**



- choose starting labeling y^0

Local Search

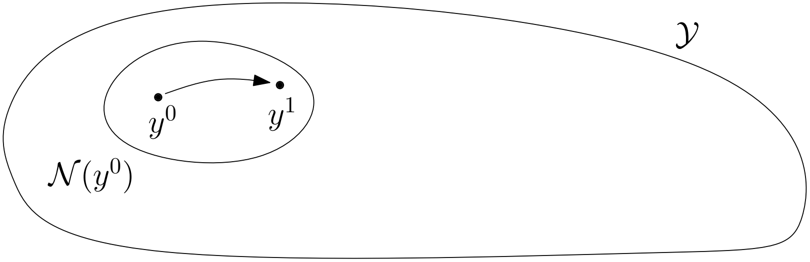
Avoid getting fractional solutions: energy minimization by **local search**



- choose starting labeling y^0
- construct neighborhood $\mathcal{N}(y^0) \subset \mathcal{Y}$ of labelings

Local Search

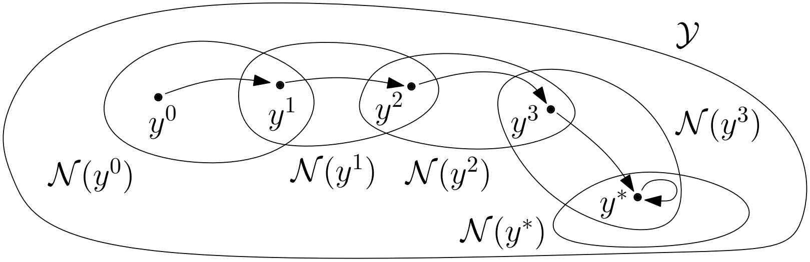
Avoid getting fractional solutions: energy minimization by **local search**



- choose starting labeling y^0
- construct neighborhood $\mathcal{N}(y^0) \subset \mathcal{Y}$ of labelings
- find minimizer within neighborhood, $y^1 = \operatorname{argmin}_{y \in \mathcal{N}(y^0)} E(y)$

Local Search

Avoid getting fractional solutions: energy minimization by **local search**



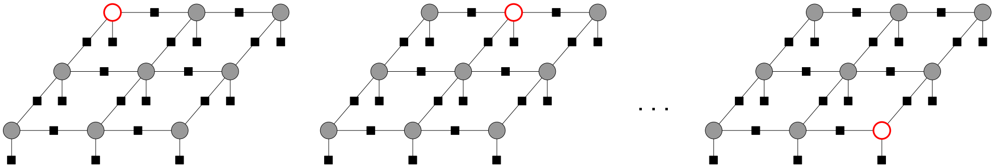
- choose starting labeling y^0
- construct neighborhood $\mathcal{N}(y^0) \subset \mathcal{Y}$ of labelings
- find minimizer within neighborhood, $y^1 = \operatorname{argmin}_{y \in \mathcal{N}(y^0)} E(y)$
- iterate until no more changes

Iterated Conditional Modes (ICM) [Besag, 1986]

Define local neighborhoods:

- $\mathcal{N}_i(y) = \{(y_1, \dots, y_{i-1}, \bar{y}, y_{i+1}, \dots, y_n) | \bar{y} \in \mathcal{Y}_i\}$ for $i \in V$.

  all labeling reachable from y by changing value of y_i



Iterated Conditional Modes (ICM) [Besag, 1986]

Define local neighborhoods:

- $\mathcal{N}_i(y) = \{(y_1, \ldots, y_{i-1}, \bar{y}, y_{i+1}, \ldots, y_n) | \bar{y} \in \mathcal{Y}_i\}$ for $i \in V$.

  all labeling reachable from y by changing value of y_i

ICM procedure:

- neighborhood $\mathcal{N}(y) = \bigcup_{i \in V} \mathcal{N}_i(y)$

  all states reachable from y by changing a single variable

- $y^{t+1} = \operatorname{argmin}_{y \in \mathcal{N}(y^t)} E(y)$ by exhaustive search $\quad (\sum_i |\mathcal{Y}_i|$ evaluations)

Iterated Conditional Modes (ICM) [Besag, 1986]

Define local neighborhoods:

- $\mathcal{N}_i(y) = \{(y_1, \dots, y_{i-1}, \bar{y}, y_{i+1}, \dots, y_n) | \bar{y} \in \mathcal{Y}_i\}$ for $i \in V$.

  all labeling reachable from y by changing value of y_i

ICM procedure:

- neighborhood $\mathcal{N}(y) = \bigcup_{i \in V} \mathcal{N}_i(y)$

  all states reachable from y by changing a single variable

- $y^{t+1} = \operatorname*{argmin}_{y \in \mathcal{N}(y^t)} E(y)$ by exhaustive search $\quad (\sum_i |\mathcal{Y}_i|$ evaluations)

Observation: larger neighborhood sizes are better

- ICM: $|\mathcal{N}(y)|$ linear in $|V|$

  $\rightarrow$ many iterations to explore exponentially large $\mathcal{Y}$
- ideal: $|\mathcal{N}(y)|$ exponential in $|V|$,

  $\rightarrow$ but: we must ensure that $\operatorname{argmin}_{y \in \mathcal{N}(y)} E(y)$ remains tractable

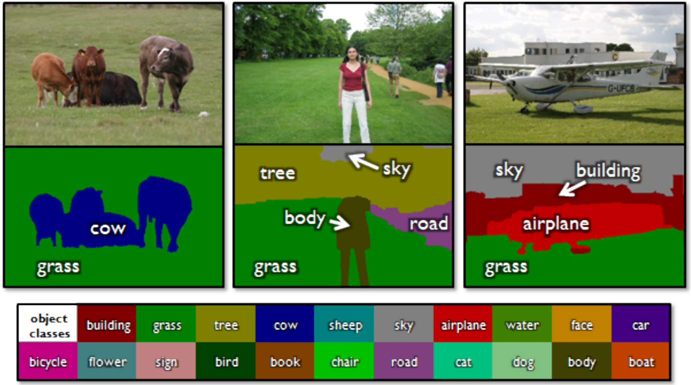
Multilabel Graph-Cut: α-expansion

- $E(y)$ with unary and pairwise terms
- $\mathcal{Y}_i = \mathcal{L} = \{1, \ldots, K\}$ for $i \in V$ (multi-class)

Multilabel Graph-Cut: α-expansion

- $E(y)$ with unary and pairwise terms
- $\mathcal{Y}_i = \mathcal{L} = \{1, \ldots, K\}$ for $i \in V$ (multi-class)

Example: semantic segmentation



Multilabel Graph-Cut: α-expansion

- $E(y)$ with unary and pairwise terms
- $\mathcal{Y}_i = \mathcal{L} = \{1, \dots, K\}$ for $i \in V$ (multi-class)

Algorithm

- initialize y^0 arbitrarily (*e.g.* everything label 0)
- repeat
  - for any $\alpha \in \mathcal{L}$
    - construct neighborhood:

$$\mathcal{N}(y) = \big\{(\bar{y}_1, \dots, \bar{y}_{|V|}) \ : \ \bar{y}_i \in \{y_i, \alpha\}\big\}$$

"each variable can keep its value or switch to α"

    - solve $y \leftarrow \operatorname{argmin}_{y \in \mathcal{N}(y)} E(y)$
- until y has not changed for a whole iteration

Multilabel Graph-Cut: α-expansion

Theorem **[Boykov et al. 2001]**
If all pairwise terms are *metric*, *i.e.* for all $(i,j) \in \mathcal{E}$

$$E_{ij}(k,l) \geq 0 \quad \text{with} \quad E_{ij}(k,l) = 0 \Leftrightarrow k = l$$

$$E_{ij}(k,l) = E_{ij}(l,k)$$

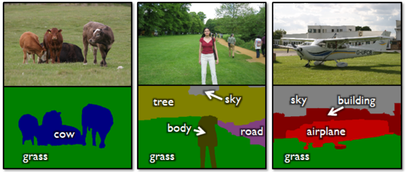
$$E_{ij}(k,l) \leq E_{ij}(k,m) + E_{ij}(m,l) \quad \text{for all } k,l,m$$

Then $\operatorname{argmin}_{y \in \mathcal{N}(y)} E(y)$ can be solved optimally using GraphCut.

Theorem **[Veksler 2001].** The solution, y_α, returned by α-expansion fulfills

$$E(y_\alpha) \;\leq\; 2c \cdot \min_{y \in \mathcal{Y}} E(y) \quad \text{for } c = \max_{(i,j)\in\mathcal{E}} \frac{\max_{k \neq l} E_{ij}(k,l)}{\min_{k \neq l} E_{ij}(k,l)}$$

Example: Semantic Segmentation



$$E(y) = \sum_{i \in V} E_i(y_i) + \lambda \sum_{(i,j) \in \mathcal{E}} [\![y_i \neq y_j]\!]$$ "Potts model"

- $E_{ij}(k,l) \geq 0 \qquad E_{ij}(k,l) = 0 \Leftrightarrow k = l \qquad E_{ij}(k,l) = E_{ij}(l,k) \qquad \checkmark$
- $E_{ij}(k,l) \leq E_{ij}(k,m) + E_{ij}(m,l) \qquad \checkmark$
- $c = \max_{(i,j)\in\mathcal{E}} \frac{\max_{k\neq l} E_{ij}(k,l)}{\min_{k\neq l} E_{ij}(k,l)} = 1$
- factor-2 approximation guarantee: $E(y_\alpha) \leq 2 \min_{y \in \mathcal{Y}} E(y)$

Example: Stereo Estimation

$$E(y) = \sum_{i \in V} E_i(y_i) + \lambda \sum_{(i,j) \in \mathcal{E}} |y_i - y_j|$$

- $|y_i - y_j|$ is metric $\checkmark$
- $c = \max_{(i,j) \in \mathcal{E}} \frac{\max_{k \neq l} E_{ij}(k,l)}{\min_{k \neq l} E_{ij}(k,l)} = |\mathcal{L} - 1|$
- weak guarantees, but often close to optimal labelings in practice

Images: Middlebury stereo vision dataset

Sampling

Sampling was a general purpose probabilistic inference method. Can we use it for prediction?

MAP prediction from samples:

- $S = \{x^1, \dots, x^N\}$ samples from $p(x)$
- $\hat{x}^* \leftarrow \operatorname{argmax}_{x \in S} p(x)$
- **output** $\hat{x}^*$

Problem:

- will need many samples
- with $x^* = \operatorname{argmax}_{x \in \mathcal{X}} p(x)$: $\Pr(\hat{x}^* \neq x^*) = (1 - p(x^*))^N \approx 1 - Np(x^*)$
- for graphical model, probability values are tiny, e.g. $p(x^*) = 10^{-100}$ can easily happen
- $N \approx 5 \cdot 10^{99}$ required to have 50% chance

Sampling

Let's construct a better distribution:

Idea 2: Form a new distribution, p', that has all its probability mass at the location of maximum of p

$$p'(x) = [\![x = x^*]\!] \qquad \text{for } x^* = \operatorname*{argmax}_{x \in \mathcal{X}} p(x)$$

and sample from it

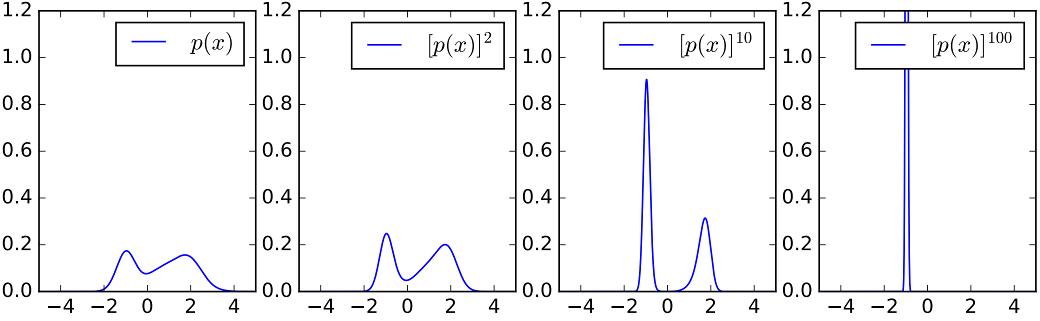
Advantage: we need only 1 sample

Problem: to define p' we need x^*, which is what we're after.

Sampling

Idea 3: do the same as idea 2, but more implicitly:

$$p_\beta(x) \propto [p(x)]^\beta \qquad \text{for very large } \beta$$



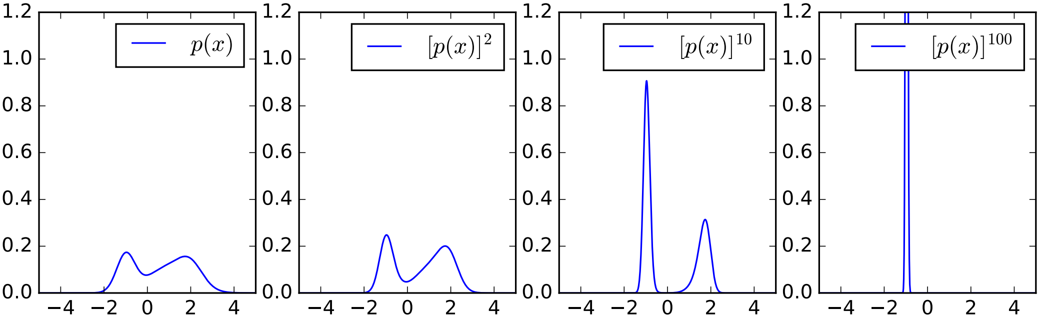
Particularly easy for distributions in exponential form: $p(x) \propto e^{-E(x)}$ becomes $p_\beta(x) \propto e^{-\beta E(x)}$

Simulated Annealing

Practical questions: How to choose β? How to sample from p_β?

These are often coupled, especially sampling works by a Monte Carlo Markov Chain (MCMC):

- samples $x^1, x^2, \dots$ from MCMC are dependent,
- two consecutive samples are often similar to each other $\rightarrow$ random walk,
- for a distribution with multiple peaks that are separated by a low-probability region, MCMC sampling will jump around a peak, but very rarely switch peaks



Sampling

Image: By Kingpin13 - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=25010763

MAP Prediction / Energy Minimization – Summary

Task: compute $\operatorname{argmin}_{y \in \mathcal{Y}} E(y|x)$

Exact Energy Minimization

Only possible for certain models:

- trees/forests: max-sum belief propagation
- general graphs: junction chain algorithm (if tractable)
- submodular energies: GraphCut
- general graphs: integer linear programming (if tractable)

Approximate Energy Minimization

Many techniques with different properties and guarantees:

- linear programs relaxations, ICM, α-expansion

Best choices depends on model and requirements.

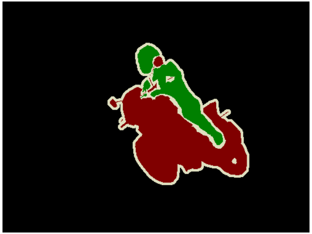
Loss functions

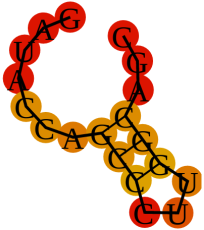
Loss functions

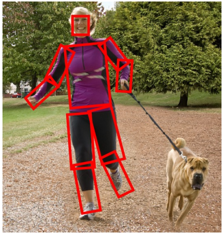
We model structured data, *e.g.* $y = (y_1, \ldots, y_m)$. What makes a good prediction?

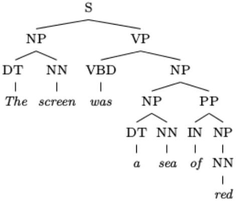
- The *loss function* is application dependent

$$\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+,$$









Example 1: 0/1 loss

Loss is 0 for perfect prediction, 1 otherwise:

$$\Delta_{0/1}(\bar{y}, y) = [\![\bar{y} \neq y]\!] = \begin{cases} 0 & \text{if } \bar{y} = y \\ 1 & \text{otherwise} \end{cases}$$

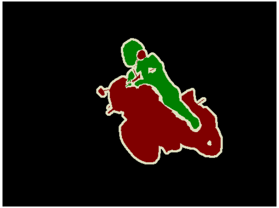
Every mistake is equally bad. Rarely very useful for *structured data*, *e.g.*

- handwriting recognition: one letter wrong is as bad as all letters wrong
- image segmentation: one pixel wrong is as bad as all pixels wrong
- automatic translation: a missing article is as bad as completely random output

Example 2: Hamming loss

Count the number of mislabeled variables:

$$\Delta_H(\bar{y}, y) = \frac{1}{m} \sum_{i=1}^{m} [\![\bar{y}_m \neq y_m]\!]$$



x:	I	need	a	coffee	break
$\bar{y}$:	subject	verb	article	object	object
y:	subject	verb	article	object	verb
$[\![\bar{y}_m \neq y_m]\!]$	0	0	0	0	1

$\rightarrow \Delta_H(\bar{y}, y) = 0.2$

Often used for graph labeling tasks, *e.g.* image segmentation, natural language processing, ...

Example 3: Squared error

If the individual variables y_i are numeric, *e.g.* pixel intensities, object locations, etc.

Sum of squared errors

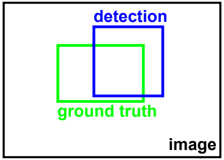
$$\Delta_Q(\bar{y}, y) = \frac{1}{m} \sum_{i=1}^{m} \|\bar{y}_i - y_i\|^2.$$

Used, *e.g.*, in stereo reconstruction, optical flow estimation, ...

Example 4: Task specific losses

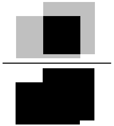
Object detection

- bounding boxes, or
- arbitrarily shaped regions



Intersection-over-union loss:

$$\Delta_{\text{IoU}}(y, \bar{y}) = 1 - \frac{\text{area}(y \cap \bar{y})}{\text{area}(y \cup \bar{y})} \qquad = 1 -$$



Used, *e.g.*, in PASCAL VOC challenges for object detection, because its scale-invariance.

Making Optimal Predictions

Given a structured distribution $p(x, y)$ or $p(y|x)$, what's the best y to predict?

Decision theory: pick y^* that causes minimal expected loss:

$$y^* = \operatorname*{argmin}_{\bar{y}\in\mathcal{Y}} \mathbb{E}_{y\sim p(y|x)}\{\Delta(y,\bar{y})\} = \operatorname*{argmin}_{\bar{y}\in\mathcal{Y}} \sum_{y\in\mathcal{Y}} \Delta(y,\bar{y})p(y|x)$$

For many loss functions not tractable to compute, but some exceptions:

- $\mathcal{R}_{\Delta_{0/1}}(y) = 1 - p(y)$, so $y^* = \operatorname{argmax}_y p(y)$ → use MAP prediction
- $\mathcal{R}_{\Delta_H}(y) = 1 - \sum_{i\in V} p(y_i)$, so $y^* = (y_1^*, \ldots, y_n^*)$ with $y_i^* = \operatorname*{argmax}_{k\in\mathcal{Y}_i} p(y_i = k)$
  → use marginal inference