Structured Prediction for Scene Understanding I

Raquel Urtasun

University of Toronto

June 20, 2014

Goal of this lecture

- Understand what structured prediction is
- Learn how to formulate a problem to be successful in practice

Contents

What is structured prediction?

Structured Prediction

- In "typical" machine learning

$$f : \mathcal{X} \rightarrow \Re$$

the input $\mathcal{X}$ can be anything, and the output is a real number (e.g., classification, regression)

- In Structured Prediction

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

the input $\mathcal{X}$ can be anything, and the output is a **complex** object (e.g., image segmentation, parse tree)

Structured Prediction

- In "typical" machine learning

$$f : \mathcal{X} \rightarrow \Re$$

  the input $\mathcal{X}$ can be anything, and the output is a real number (e.g., classification, regression)

- In Structured Prediction

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

  the input $\mathcal{X}$ can be anything, and the output is a **complex** object (e.g., image segmentation, parse tree)

- In this lecture $\mathcal{Y}$ is a discrete space, ask me later if you are interested in continuous variables.

Structured Prediction

- In "typical" machine learning

$$f : \mathcal{X} \rightarrow \Re$$

  the input $\mathcal{X}$ can be anything, and the output is a real number (e.g., classification, regression)

- In Structured Prediction

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

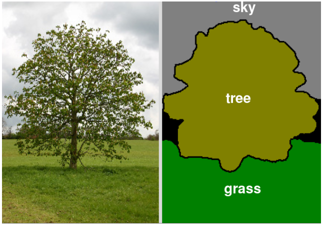
  the input $\mathcal{X}$ can be anything, and the output is a **complex** object (e.g., image segmentation, parse tree)

- In this lecture $\mathcal{Y}$ is a discrete space, ask me later if you are interested in continuous variables.

Structured Prediction and its Applications

We want to predict multiple random variables which are related

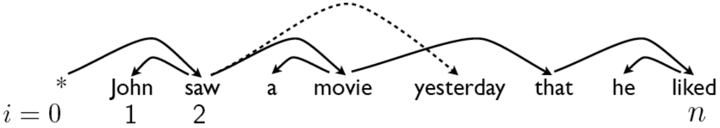
- Computer Vision:
    - Semantic Segmentation (output: pixel-wise labeling)
    - Object detection (output: 2D or 3D bounding boxes)
    - Stereo Reconstruction (output: 3D map)
    - Scene Understanding (output: 3D bounding box reprinting the layout)



Structured Prediction and its Applications

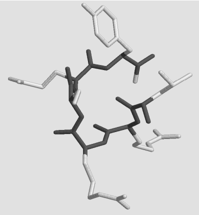
We want to predict multiple random variables which are related

- Natural Language processing
  - Machine Translation (output: sentence in another language)
  - Parsing (output: parse tree)



- Computational Biology
  - Protein Folding (output: 3D protein)

MRLLILALLGICSLTAYIVEGVGSEVSDKR
TCVSLTTQRLPVSRIKTYTITEGSLRAVIF
ITKRGLKVCADPQATWVRDVVRSMDRKSNT
RNNMIQTKPTGTQQSTNTAVTLTG



Why structured?

- Independent prediction is good but...

- Neighboring pixels should have same labels (if they look similar).

Why structured?

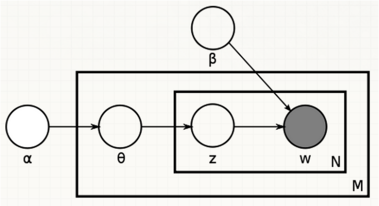
- Independent prediction is good but...

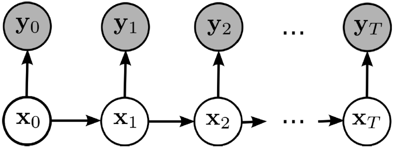
- Neighboring pixels should have same labels (if they look similar).

Graphical Model

A graphical model defines

- A family of probability distributions over a set of random variables
- This is expressed via a graph, which encodes the conditional independences of the distribution





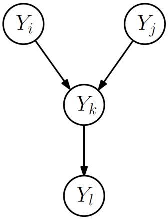
- Two types of graphical models: Directed and undirected

Bayesian Networks

- The graph $G = (V, \mathcal{E})$ is acyclic and directed
- Factorization over distributions by conditioning on parent nodes

$$p(\mathbf{y}) = \prod_{i \in V} p(y_i | y_{pa}(i))$$

- Example



$$p(\mathbf{y}) = p(y_l | y_k) p(y_k | y_i, y_j) p(y_i) p(y_j)$$

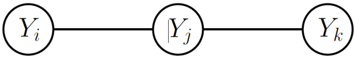
Undirected Graphical Model

- Also called Markov Random Field, or Markov Network
- Graph $G = (V, \mathcal{E})$ is undirected and has no self-edges
- Factorization over cliques

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{r \in \mathbb{R}} \psi_r(\mathbf{y}_r)$$

with $Z = \sum_{\mathbf{y} \in \mathcal{Y}} \prod_{r \in \mathbb{R}} \psi_r(\mathbf{y}_r)$ the partition function

- Example



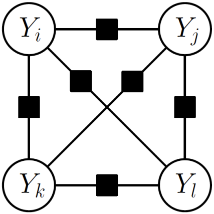
$$p(\mathbf{y}) = \frac{1}{Z} \psi(y_i, y_j)\psi(y_j, y_k)\psi(y_i)\psi(y_j)\psi(y_k)$$

- **Difficulty**: Exponentially many configurations
- Undirected models will be the focus of this lecture

Factor Graph Representation

- Graph $G = (V, \mathcal{F}, \mathcal{E})$, with variable nodes $\mathcal{V}$, factor nodes $\mathcal{F}$ and edges $\mathcal{E}$
- **Scope** of a factor $N(F) = \{i \in V : (i, F) \in \mathcal{E}\}$
- Factorization over factors

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(\mathbf{y}_{N(F)})$$



Factor Graph vs Graphical Model

- Factor graphs are explicit about the factorization

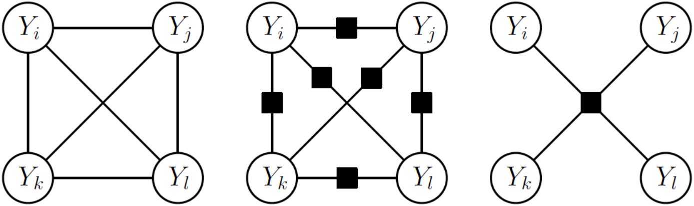


Figure : from [Nowozin et al]

Capacity

- They define the family of distributions and thus the *capacity*

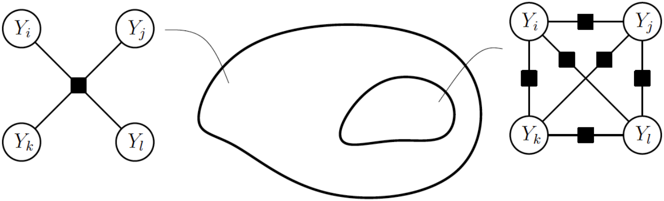


Figure : from [Nowozin et al]

Markov Random Fields vs Conditional Random Fields

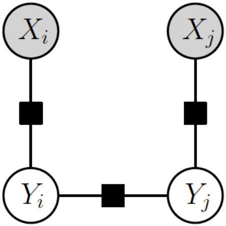
- Markov Random Fields (MRFs) define

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(\mathbf{y}_{N(F)})$$

- Conditional Random Fields (CRFs) define

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{F \in \mathcal{F}} \psi_F(\mathbf{y}_{N(F)}; \mathbf{x})$$

- **x** is not a random variable (i.e., not part of the probability distribution)



Energy vs Probabilities

- The probability is completely determined by the energy

$$
\begin{aligned}
p(\mathbf{y}) &= \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(\mathbf{y}_{N(F)}) \\
&= \frac{1}{Z} \exp\left(\log(\psi_F(\mathbf{y}_{N(F)}))\right) \\
&= \frac{1}{Z} \exp\left(-\sum_{F \in \mathcal{F}} E_F(y_F)\right)
\end{aligned}
$$

where $E_F(y_F) = -\log(\psi_F(\mathbf{y}_{N(F)}))$

Parameterization: log linear model

- Factor graphs define a family of distributions
- We are interestested in identifying individual members by parameters

$$E_F(\mathbf{y}_F) = -\mathbf{w}^T \phi_F(\mathbf{y}_F)$$

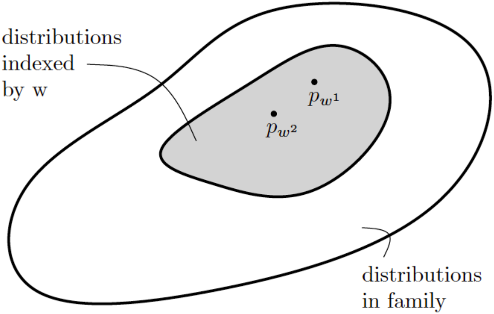


Figure : from [Nowozin et al]

Learning Tasks

- Estimation of the parameters $\mathbf{w}$

$$E_F(\mathbf{y}_F) = -\mathbf{w}^T \phi_F(\mathbf{y}_F)$$

- Learn the structure of the model
- Learn with hidden variables

Inference Tasks

Given an input $x \in \mathcal{X}$ we want to compute

- **MAP estimate** or minimum energy configuration

$$\begin{aligned}
\operatorname*{argmax}_{y \in \mathcal{Y}} p(\mathbf{y}|\mathbf{x}) &= \operatorname*{argmax}_{y \in \mathcal{Y}} \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(\mathbf{y}_{N(F)}; \mathbf{x}, \mathbf{w}) \\
&= \operatorname*{argmax}_{y \in \mathcal{Y}} \exp(-\sum_{F \in \mathcal{F}} E_F(\mathbf{y}_F, \mathbf{x}, \mathbf{w})) \\
&= \operatorname*{argmin}_{y \in \mathcal{Y}} \sum_{F \in \mathcal{F}} E_F(\mathbf{y}_F, \mathbf{x}, \mathbf{w})
\end{aligned}$$

- **Marginals** $p(y_i)$ or max marginals $\max_{y_i \in \mathcal{Y}_i} p(y_i)$, which requires computing the partition function Z, i.e.,

$$\begin{aligned}
\log(Z(\mathbf{x}, \mathbf{w})) &= \log \sum_{\mathbf{y} \in \mathcal{Y}} \exp(-E(\mathbf{y}; \mathbf{x}, \mathbf{w})) \\
\mu_F(\mathbf{y}_F) &= p(\mathbf{y}_F | \mathbf{x}, \mathbf{w})
\end{aligned}$$

Inference Tasks

Given an input $x \in \mathcal{X}$ we want to compute

- **MAP estimate** or minimum energy configuration

$$\begin{aligned}
\operatorname*{argmax}_{y \in \mathcal{Y}} p(\mathbf{y}|\mathbf{x}) &= \operatorname*{argmax}_{y \in \mathcal{Y}} \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(\mathbf{y}_{N(F)}; \mathbf{x}, \mathbf{w}) \\
&= \operatorname*{argmax}_{y \in \mathcal{Y}} \exp(-\sum_{F \in \mathcal{F}} E_F(\mathbf{y}_F, \mathbf{x}, \mathbf{w})) \\
&= \operatorname*{argmin}_{y \in \mathcal{Y}} \sum_{F \in \mathcal{F}} E_F(\mathbf{y}_F, \mathbf{x}, \mathbf{w})
\end{aligned}$$

- **Marginals** $p(y_i)$ or max marginals $\max_{y_i \in \mathcal{Y}_i} p(y_i)$, which requires computing the partition function Z, i.e.,

$$\begin{aligned}
\log(Z(\mathbf{x}, \mathbf{w})) &= \log \sum_{\mathbf{y} \in \mathcal{Y}} \exp(-E(\mathbf{y}; \mathbf{x}, \mathbf{w})) \\
\mu_F(\mathbf{y}_F) &= p(\mathbf{y}_F | \mathbf{x}, \mathbf{w})
\end{aligned}$$

Inference in Markov Random Fields

MAP Inference

Compute the MAP estimate is typically NP-hard

$$\max_{y \in \mathcal{Y}} p(\mathbf{y}|x) = \max_{y \in \mathcal{Y}} \sum_{r \in \mathcal{R}} \mathbf{w}^T \phi_r(\mathbf{y}_r)$$

Notable exceptions are:

- Belief propagation for tree-structure models

MAP Inference

Compute the MAP estimate is typically NP-hard

$$\max_{y \in \mathcal{Y}} p(\mathbf{y}|x) \quad = \quad \max_{y \in \mathcal{Y}} \sum_{r \in \mathcal{R}} \mathbf{w}^T \phi_r(\mathbf{y}_r)$$

Notable exceptions are:

- Belief propagation for tree-structure models
- Graph cuts for binary energies with sub modular potentials

MAP Inference

Compute the MAP estimate is typically NP-hard

$$\max_{y \in \mathcal{Y}} p(\mathbf{y}|x) \quad = \quad \max_{y \in \mathcal{Y}} \sum_{r \in \mathcal{R}} \mathbf{w}^T \phi_r(\mathbf{y}_r)$$

Notable exceptions are:

- Belief propagation for tree-structure models
- Graph cuts for binary energies with sub modular potentials
- Branch and bound: exponential in worst case, but works much faster in practice

MAP Inference

Compute the MAP estimate is typically NP-hard

$$\max_{y \in \mathcal{Y}} p(\mathbf{y}|x) = \max_{y \in \mathcal{Y}} \sum_{r \in \mathcal{R}} \mathbf{w}^T \phi_r(\mathbf{y}_r)$$

Notable exceptions are:

- Belief propagation for tree-structure models
- Graph cuts for binary energies with sub modular potentials
- Branch and bound: exponential in worst case, but works much faster in practice

Difficulties

- Deal with the exponentially many states in $\mathbf{y}$

MAP Inference

Compute the MAP estimate is typically NP-hard

$$\max_{y \in \mathcal{Y}} p(\mathbf{y}|x) = \max_{y \in \mathcal{Y}} \sum_{r \in \mathcal{R}} \mathbf{w}^T \phi_r(\mathbf{y}_r)$$

Notable exceptions are:

- Belief propagation for tree-structure models
- Graph cuts for binary energies with sub modular potentials
- Branch and bound: exponential in worst case, but works much faster in practice

Difficulties

- Deal with the exponentially many states in **y**

We are going to see examples of the three techniques

MAP Inference

Compute the MAP estimate is typically NP-hard

$$\max_{y \in \mathcal{Y}} p(\mathbf{y}|x) \quad = \quad \max_{y \in \mathcal{Y}} \sum_{r \in \mathcal{R}} \mathbf{w}^T \phi_r(\mathbf{y}_r)$$

Notable exceptions are:

- Belief propagation for tree-structure models
- Graph cuts for binary energies with sub modular potentials
- Branch and bound: exponential in worst case, but works much faster in practice

Difficulties

- Deal with the exponentially many states in **y**

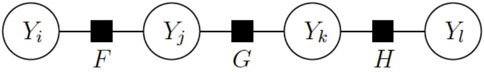
We are going to see examples of the three techniques

- Compact notation

$$\theta_r(\mathbf{y}_r) = \mathbf{w}^T \phi_r(\mathbf{y}_r)$$

- Inference can be written as

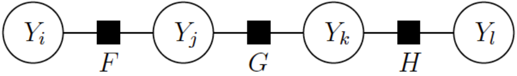
$$\max_{\mathbf{y} \in \mathcal{Y}} \sum_{r \in \mathcal{R}} \theta_r(\mathbf{y}_r)$$



- For the example

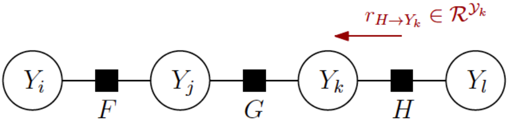
$$\max_{y_i, y_j, y_k, y_l} \quad \{\theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_G(y_k, y_l)\}$$

Belief Propagation



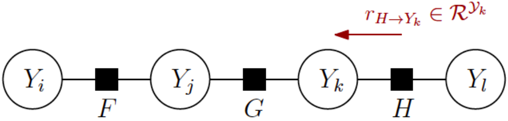
$$
\begin{aligned}
\theta^*(\mathbf{y}) &= \max_{y_i,y_j,y_k,y_l} \{\theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_H(y_k, y_l)\} \\
&= \max_{y_i,y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_l} \theta_H(y_k, y_l)
\end{aligned}
$$

Belief Propagation



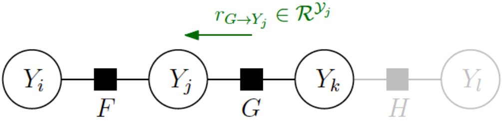
$$
\begin{aligned}
\theta^*(\mathbf{y}) &= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \underbrace{\max_{y_l} \theta_H(y_k, y_l)}_{r_{H \to y_k}(y_k)} \\
&= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + r_{H \to y_k}(y_k)
\end{aligned}
$$

Belief Propagation



$$\begin{aligned}\theta^*(\mathbf{y}) &= \max_{y_i,y_j} \theta_F(y_i,y_j) + \max_{y_k} \theta_G(y_j,y_k) + \underbrace{\max_{y_l} \theta_H(y_k,y_l)}_{r_{H\to y_k}(y_k)} \\ &= \max_{y_i,y_j} \theta_F(y_i,y_j) + \max_{y_k} \theta_G(y_j,y_k) + r_{H\to y_k}(y_k)\end{aligned}$$

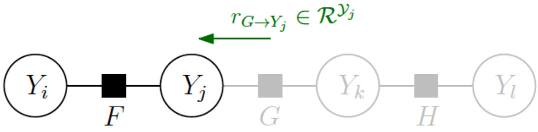
Belief Propagation



$$\theta^*(\mathbf{y}) = \max_{y_i, y_j} \theta_F(y_i, y_j) + \underbrace{\max_{y_k} \theta_G(y_j, y_k) + r_{H \to y_k}(y_k)}_{r_{G \to y_j}(y_j)}$$

$$= \max_{y_i, y_j} \theta_F(y_i, y_j) + r_{G \to y_j}(y_j)$$

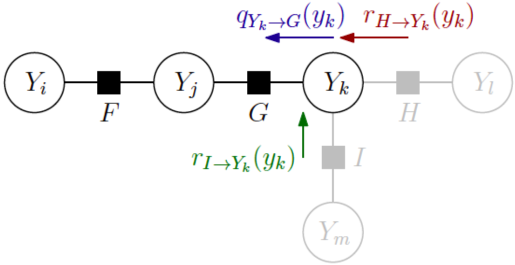
Belief Propagation



$$\theta^*(\mathbf{y}) = \max_{y_i,y_j} \theta_F(y_i,y_j) + \underbrace{\max_{y_k} \theta_G(y_j,y_k) + r_{H\to y_k}(y_k)}_{r_{G\to y_j}(y_j)}$$

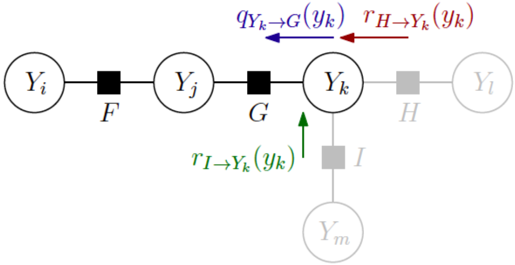
$$= \max_{y_i,y_j} \theta_F(y_i,y_j) + r_{G\to y_j}(y_j)$$

Tree Generalization



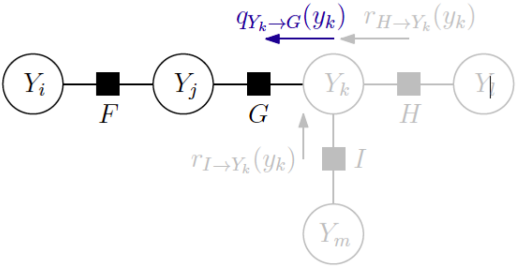
$$\begin{aligned}\theta^*(\mathbf{y}) &= \max_{y_i, y_k, y_k, y_l, y_m} \theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_I(y_m, y_k) + \theta_H(y_l, y_k) \\ &= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_m} \theta_I(y_m, y_k) + \max_{y_l} \theta_H(y_l, y_k)\end{aligned}$$

Tree Generalization



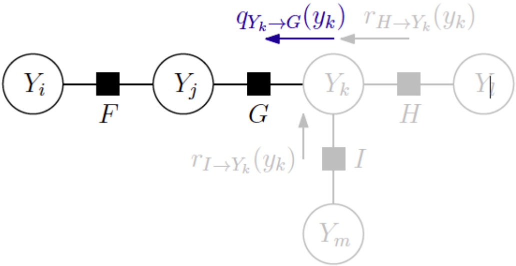
$$
\begin{aligned}
\theta^*(\mathbf{y}) &= \max_{y_i, y_k, y_k, y_l, y_m} \theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_I(y_m, y_k) + \theta_H(y_l, y_k) \\
&= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_m} \theta_I(y_m, y_k) + \max_{y_l} \theta_H(y_l, y_k) \\
&= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + r_{H \to y_k}(y_k) + r_{I \to y_k}(y_k)
\end{aligned}
$$

Tree Generalization



$$\begin{aligned}
\theta^*(\mathbf{y}) &= \max_{y_i, y_k, y_k, y_l, y_m} \theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_I(y_m, y_k) + \theta_H(y_l, y_k) \\
&= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_m} \theta_I(y_m, y_k) + \max_{y_l} \theta_H(y_l, y_k) \\
&= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + r_{H \to y_k}(y_k) + r_{I \to y_k}(y_k) \\
&= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + q_{y_k \to G}(y_k)
\end{aligned}$$

Tree Generalization



$$
\begin{aligned}
\theta^*(\mathbf{y}) &= \max_{y_i, y_k, y_k, y_l, y_m} \theta_F(y_i, y_j) + \theta_G(y_j, y_k) + \theta_I(y_m, y_k) + \theta_H(y_l, y_k) \\
&= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + \max_{y_m} \theta_I(y_m, y_k) + \max_{y_l} \theta_H(y_l, y_k) \\
&= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + r_{H \to y_k}(y_k) + r_{I \to y_k}(y_k) \\
&= \max_{y_i, y_j} \theta_F(y_i, y_j) + \max_{y_k} \theta_G(y_j, y_k) + q_{y_k \to G}(y_k)
\end{aligned}
$$

Factor Graph Max Product

Iteratively updates and passes messages:

- $r_{F \to y_i} \in \Re^{\mathcal{Y}_i}$: factor to variable message
- $q_{y_i \to F} \in \Re^{\mathcal{Y}_i}$: variable to factor message

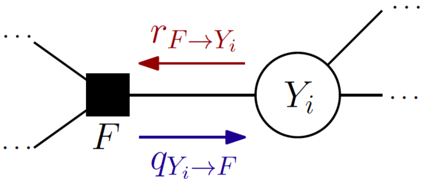


Figure : from [Nowozin et al]

Variable to factor

- Let $M(i)$ be the factors adjacent to variable i, $M(i) = \{F \in \mathcal{F} : (i, F) \in \mathcal{E}\}$
- Variable-to-factor message

$$q_{y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to y_i}(y_i)$$

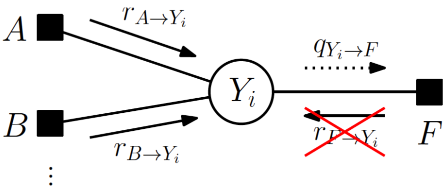


Figure : from [Nowozin et al]

Factor to variable

- Factor-to-variable message

$$r_{F\to y_i}(y_i) = \max_{y'_F \in \mathcal{Y}_F, y'_i = y_i} \left(\theta(y'_F) + \sum_{j \in N(F)\setminus\{i\}} q_{y_j\to F}(y'_j) \right)$$

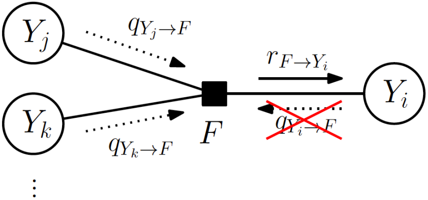


Figure : from [Nowozin et al]

1. Select one variable as tree root
2. Compute leaf-to-root messages
3. Compute root-to-leaf messages

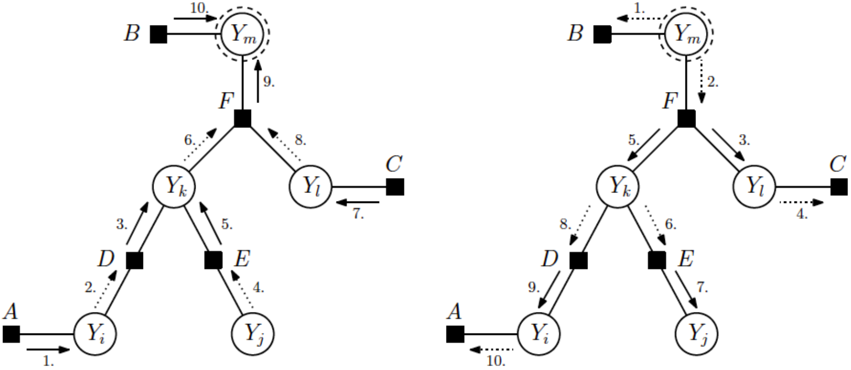


Figure : from [Nowozin et al]

Max Product v Sum Product

Max sum version of max-product

1. Compute leaf-to-root messages

$$q_{y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to y_i}(y_i)$$

2. Compute root-to-leaf messages

$$r_{F \to y_i}(y_i) = \max_{y'_F \in \mathcal{Y}_F, y'_i = y_i} \left(\theta(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{y_j \to F(y'_j)} \right)$$

Sum-product

1. Compute leaf-to-root messages

$$q_{y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to y_i}(y_i)$$

2. Compute root-to-leaf messages

$$r_{F \to y_i}(y_i) = \log \sum_{y'_F \in \mathcal{Y}_F, y'_i = y_i} \exp \left(\theta(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{y'_j \to F}(y'_j) \right)$$

Max Product v Sum Product

Max sum version of max-product

1. Compute leaf-to-root messages

$$q_{y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to y_i}(y_i)$$

2. Compute root-to-leaf messages

$$r_{F \to y_i}(y_i) = \max_{y'_F \in \mathcal{Y}_F, y'_i = y_i} \left(\theta(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{y_j \to F(y'_j)} \right)$$

Sum-product

1. Compute leaf-to-root messages

$$q_{y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to y_i}(y_i)$$

2. Compute root-to-leaf messages

$$r_{F \to y_i}(y_i) = \log \sum_{y'_F \in \mathcal{Y}_F, y'_i = y_i} \exp \left(\theta(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{y'_j \to F}(y'_j) \right)$$

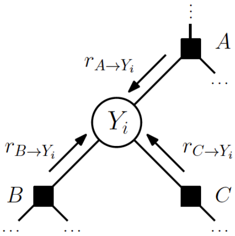
Computing marginals

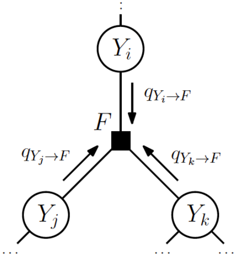
- Partition function can be evaluated at the root

$$\log Z = \log \sum_{y_r} \exp \left(\sum_{F \in M(r)} r_{F \to y_r}(y_r) \right)$$

- Marginal distributions, for each factor

$$\mu_F(y_F) = p(y_F) = \frac{1}{Z} \exp \left(\theta_F(y_F) + \sum_{i \in N(F)} q_{y_i \to F}(y_i) \right)$$





Computing marginals

- Partition function can be evaluated at the root

$$\log Z = \log \sum_{y_r} \exp \left(\sum_{F \in M(r)} r_{F \to y_r}(y_r) \right)$$

- Marginal distributions, for each factor

$$\mu_F(y_F) = p(y_F) = \frac{1}{Z} \exp \left(\theta_F(y_F) + \sum_{i \in N(F)} q_{y_i \to F}(y_i) \right)$$

- Marginals at every node

$$\mu_{y_i}(y_i) = p(y_i) = \frac{1}{Z} \exp \left(\sum_{F \in M(i)} r_{F \to y_i}(y_i) \right)$$

Generalizations to loops

- It is call **loopy belief propagation** (Perl, 1988)
- no schedule that removes dependencies
- Different messaging schedules (synchronous/asynchronous, static/dynamic)
- Slight changes in the algorithm

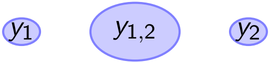
MAP LP Relaxation Task

Integer Linear Program (LP) equivalence [Werner 2007]:

- Inference task:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \sum_r \theta_r(\mathbf{y}_r)$$

- Variables $b_r(\mathbf{y}_r)$:



$$\max_{b_1,b_2,b_{12}} \begin{bmatrix} b_1(0) \\ b_1(1) \\ b_2(0) \\ b_2(1) \\ b_{12}(0,0) \\ b_{12}(1,0) \\ b_{12}(0,1) \\ b_{12}(1,1) \end{bmatrix}^\top \begin{bmatrix} \theta_1(0) \\ \theta_1(1) \\ \theta_2(0) \\ \theta_2(1) \\ \theta_{12}(0,0) \\ \theta_{12}(1,0) \\ \theta_{12}(0,1) \\ \theta_{12}(1,1) \end{bmatrix} \quad \text{s.t.} \quad b_r(\mathbf{y}_r) \in \{0,1\}$$

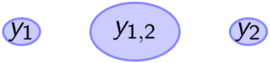
MAP LP Relaxation Task

Integer Linear Program (LP) equivalence [Werner 2007]:

- Inference task:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \sum_r \theta_r(\mathbf{y}_r)$$

- Variables $b_r(\mathbf{y}_r)$:



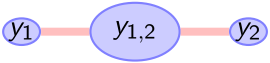
$$\max_{b_1,b_2,b_{12}} \begin{bmatrix} b_1(0) \\ b_1(1) \\ b_2(0) \\ b_2(1) \\ b_{12}(0,0) \\ b_{12}(1,0) \\ b_{12}(0,1) \\ b_{12}(1,1) \end{bmatrix}^\top \begin{bmatrix} \theta_1(0) \\ \theta_1(1) \\ \theta_2(0) \\ \theta_2(1) \\ \theta_{12}(0,0) \\ \theta_{12}(1,0) \\ \theta_{12}(0,1) \\ \theta_{12}(1,1) \end{bmatrix} \quad \text{s.t.} \quad \begin{array}{l} b_r(\mathbf{y}_r) \in \{0,1\} \\ \sum_{\mathbf{y}_r} b_r(\mathbf{y}_r) = 1 \end{array}$$

MAP LP Relaxation Task

Integer Linear Program (LP) equivalence [Werner 2007]:

- Inference task:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \sum_r \theta_r(\mathbf{y}_r)$$



- Variables $b_r(\mathbf{y}_r)$:

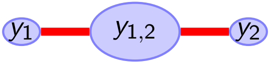
$$\max_{b_1,b_2,b_{12}} \begin{bmatrix} b_1(0) \\ b_1(1) \\ b_2(0) \\ b_2(1) \\ b_{12}(0,0) \\ b_{12}(1,0) \\ b_{12}(0,1) \\ b_{12}(1,1) \end{bmatrix}^\top \begin{bmatrix} \theta_1(0) \\ \theta_1(1) \\ \theta_2(0) \\ \theta_2(1) \\ \theta_{12}(0,0) \\ \theta_{12}(1,0) \\ \theta_{12}(0,1) \\ \theta_{12}(1,1) \end{bmatrix} \quad \text{s.t.} \quad \begin{array}{l} b_r(\mathbf{y}_r) \in \{0,1\} \\ \sum_{\mathbf{y}_r} b_r(\mathbf{y}_r) = 1 \\ \sum_{\mathbf{y}_p \setminus \mathbf{y}_r} b_p(\mathbf{y}_p) = b_r(\mathbf{y}_r) \end{array}$$

MAP LP Relaxation Task

Integer Linear Program (LP) equivalence [Werner 2007]:

- Inference task:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \sum_r \theta_r(\mathbf{y}_r)$$



- Variables $b_r(\mathbf{y}_r)$:

$$\max_{b_1,b_2,b_{12}} \begin{bmatrix} b_1(0) \\ b_1(1) \\ b_2(0) \\ b_2(1) \\ b_{12}(0,0) \\ b_{12}(1,0) \\ b_{12}(0,1) \\ b_{12}(1,1) \end{bmatrix}^\top \begin{bmatrix} \theta_1(0) \\ \theta_1(1) \\ \theta_2(0) \\ \theta_2(1) \\ \theta_{12}(0,0) \\ \theta_{12}(1,0) \\ \theta_{12}(0,1) \\ \theta_{12}(1,1) \end{bmatrix} \quad \text{s.t.} \quad \begin{array}{l} b_r(\mathbf{y}_r) \in \{0,1\} \\ \sum_{\mathbf{y}_r} b_r(\mathbf{y}_r) = 1 \\ \sum_{\mathbf{y}_p \setminus \mathbf{y}_r} b_p(\mathbf{y}_p) = b_r(\mathbf{y}_r) \end{array}$$

MAP LP Relaxation Task

$$\max_{b_1,b_2,b_{12}} \begin{bmatrix} b_1(1) \\ b_1(2) \\ b_2(1) \\ b_2(2) \\ b_{12}(1,1) \\ b_{12}(2,1) \\ b_{12}(1,2) \\ b_{12}(2,2) \end{bmatrix}^\top \begin{bmatrix} \theta_1(1) \\ \theta_1(2) \\ \theta_2(1) \\ \theta_2(2) \\ \theta_{12}(1,1) \\ \theta_{12}(2,1) \\ \theta_{12}(1,2) \\ \theta_{12}(2,2) \end{bmatrix} \quad \text{s.t.} \quad \begin{array}{l} b_r(\mathbf{y}_r) \in \{0,1\} \\ \sum_{y_r} b_r(\mathbf{y}_r) = 1 \\ \sum_{\mathbf{y}_p \setminus \mathbf{y}_r} b_p(\mathbf{y}_p) = b_r(\mathbf{y}_r) \end{array}$$

MAP LP Relaxation Task

$$\max_{b_r} \quad \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r)\theta_r(\mathbf{y}_r) \qquad \text{s.t.} \quad \begin{aligned} & b_r(\mathbf{y}_r) \in \{0,1\} \\ & \sum_{y_r} b_r(\mathbf{y}_r) = 1 \\ & \sum_{\mathbf{y}_p \setminus \mathbf{y}_r} b_p(\mathbf{y}_p) = b_r(\mathbf{y}_r) \end{aligned}$$

MAP LP Relaxation Task

$$\max_{b_r} \quad \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r)\theta_r(\mathbf{y}_r) \quad \text{s.t.} \quad \begin{array}{l} b_r(\mathbf{y}_r) \in \{0,1\} \\ \sum_{y_r} b_r(\mathbf{y}_r) = 1 \\ \text{Marginalization} \end{array}$$

MAP LP Relaxation Task

LP relaxation:

$$\max_{b_r} \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r)\theta_r(\mathbf{y}_r) \quad \text{s.t.} \quad \begin{array}{l} b_r(\mathbf{y}_r) \in \{0,1\} \\ \text{Local probability } b_r \\ \\ \text{Marginalization} \end{array}$$

MAP LP Relaxation Task

LP relaxation:

$$\max_{b_r} \quad \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r)\theta_r(\mathbf{y}_r) \quad \text{s.t.} \quad \begin{array}{l} \cancel{b_r(\mathbf{y}_r) \in \{0,1\}} \\ \text{Local probability } b_r \\ \\ \text{Marginalization} \end{array}$$

Can be solved by any standard LP solver but **slow** because of typically many variables and constraints. Can we do better?

MAP LP Relaxation Task

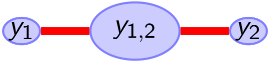
LP relaxation:

$$\max_{b_r} \quad \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r)\theta_r(\mathbf{y}_r) \quad \text{s.t.} \quad \begin{array}{l} \cancel{b_r(\mathbf{y}_r) \in \{0,1\}} \\ \text{Local probability } b_r \\ \\ \text{Marginalization} \end{array}$$

Can be solved by any standard LP solver but **slow** because of typically many variables and constraints. Can we do better?

MAP LP Relaxation Task

Observation: Graph structure in marginalization constraints.



Use dual to take advantage of structure in constraint set

- Set of parents of region r: $P(r)$
- Set of children of region r: $C(r)$

$$\forall r, \mathbf{y}_r, p \in P(r) \qquad \sum_{\mathbf{y}_p \setminus \mathbf{y}_r} b_p(\mathbf{y}_p) = b_r(\mathbf{y}_r)$$

- Lagrange multipliers for every constraint:

$$\forall r, \mathbf{y}_r, p \in P(r) \qquad \lambda_{r \to p}(\mathbf{y}_r)$$

MAP LP Relaxation Task

Re-parameterization of score $\theta_r(\mathbf{y}_r)$:

$$\hat{\theta}_r(\mathbf{y}_r) = \theta_r(\mathbf{y}_r) + \sum_{p \in P(r)} \lambda_{r \to p}(\mathbf{y}_r) - \sum_{c \in C(r)} \lambda_{c \to r}(\mathbf{y}_c)$$

Properties of dual program:

$$\min_{\lambda} q(\lambda) = \min_{\lambda} \sum_{r} \max_{\mathbf{y}_r} \hat{\theta}_r(\mathbf{y}_r)$$

- **Dual upper-bounds primal** $\forall \lambda$
- Convex problem
- Unconstrained task
- Doing block coordinate descent in the dual results on message passing (Lagrange multipliers are your messages)

MAP LP Relaxation Task

Block-coordinate descent solvers iterate the following steps:

- Take a block of Lagrange multipliers
- Optimize sub-problem of dual function w.r.t. this block while keeping all other variables fixed

Advantage: fast due to analytically computable sub-problems

Same type of algorithms also exist to compute approximate marginals

Graph-Cuts for MRF Inference

Theorem [Kolmogorov and Zabih, 2004]: If the energy function is a function of binary variables containing only unary and pairwise factors, the discrete energy minimization problem

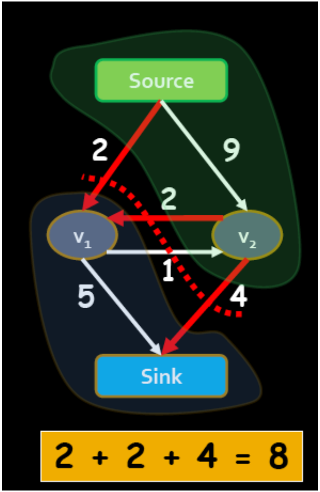
$$\min_{\mathbf{y}} \sum_{r \in \mathcal{R}} E(\mathbf{y}_r, x)$$

can be formulated as a graph cut problem if an only off all pairwise energies are sub modular

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

The ST-mincut problem

- The st-mincut is the st-cut with the minimum cost

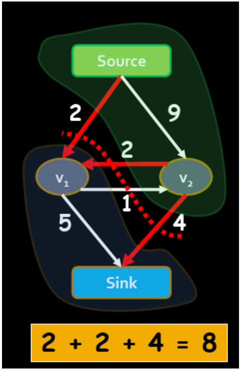


[Source: P. Kohli]

Back to our energy minimization

Construct a graph such that

1. Any st-cut corresponds to an assignment of x
2. The cost of the cut is equal to the energy of x : E(x)



[Source: P. Kohli]

St-mincut and Energy Minimization

$$E(x) = \sum_i \theta_i (x_i) + \sum_{i,j} \theta_{ij} (x_i, x_j)$$

For all ij $\theta_{ij}(0,1) + \theta_{ij}(1,0) \geq \theta_{ij}(0,0) + \theta_{ij}(1,1)$

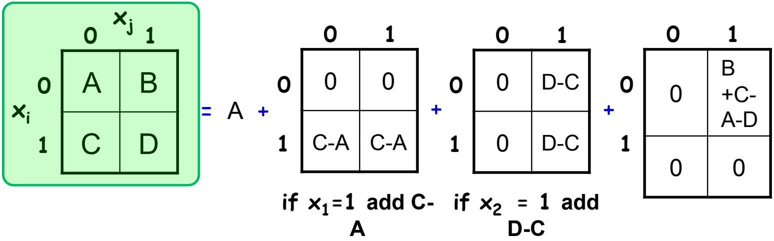
Equivalent (transformable)

$$E(x) = \sum_i c_i x_i + \sum_{i,j} c_{ij} x_i (1 - x_j)$$ $c_{ij} \geq 0$

[Source: P. Kohli]

How are they equivalent?

$A = \theta_{ij}(0,0)$ $B = \theta_{ij}(0,1)$ $C = \theta_{ij}(1,0)$ $D = \theta_{ij}(1,1)$

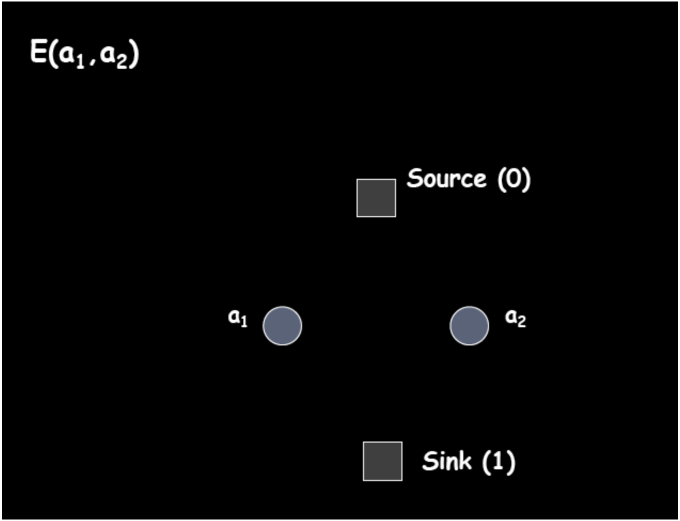


$$\theta_{ij}(x_i,x_j) = \theta_{ij}(0,0) + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, x_i + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, x_j + (\theta_{ij}(1,0) + \theta_{ij}(0,1) - \theta_{ij}(0,0) - \theta_{ij}(1,1))\,(1-x_i)\, x_j$$

$B+C-A-D \geq 0$ is true from the submodularity of θ_{ij}

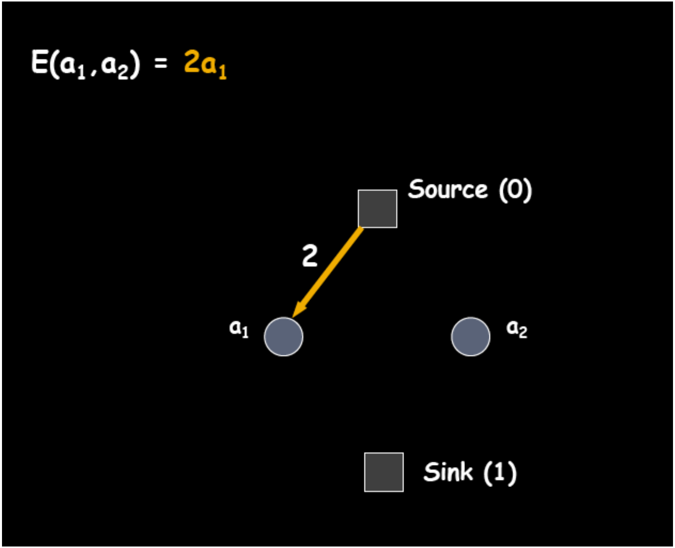
[Source: P. Kohli]

Graph Construction



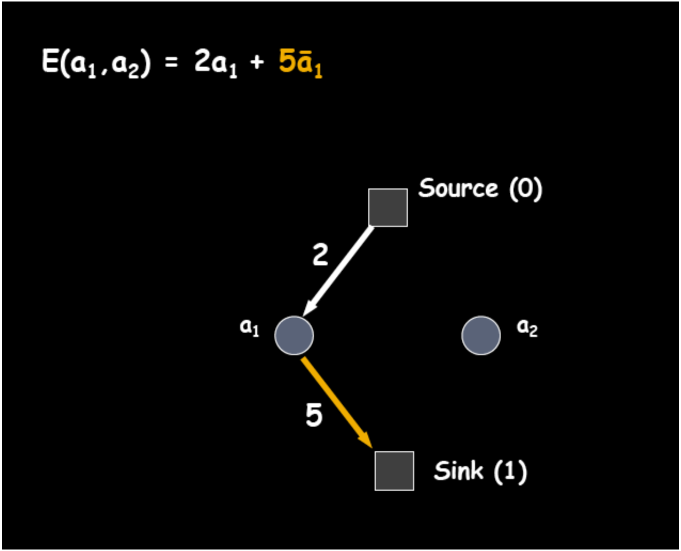
[Source: P. Kohli]

Graph Construction



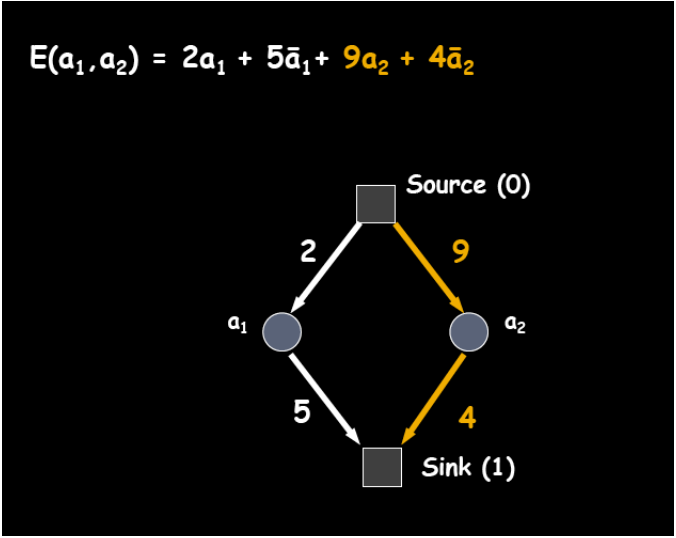
[Source: P. Kohli]

Graph Construction



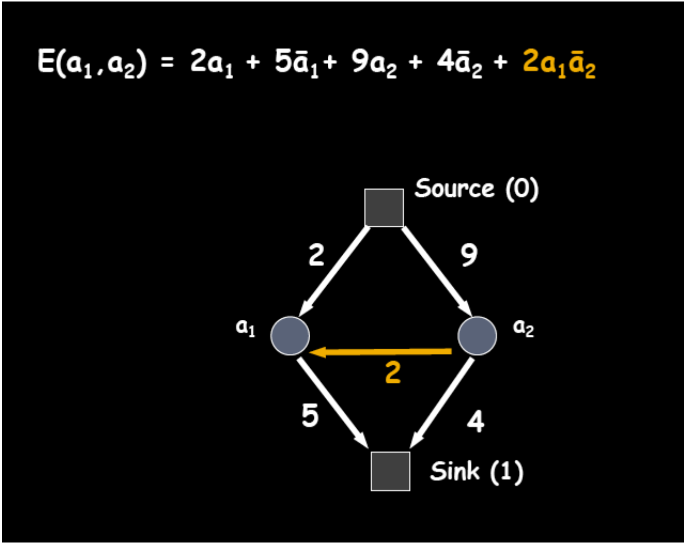
[Source: P. Kohli]

Graph Construction



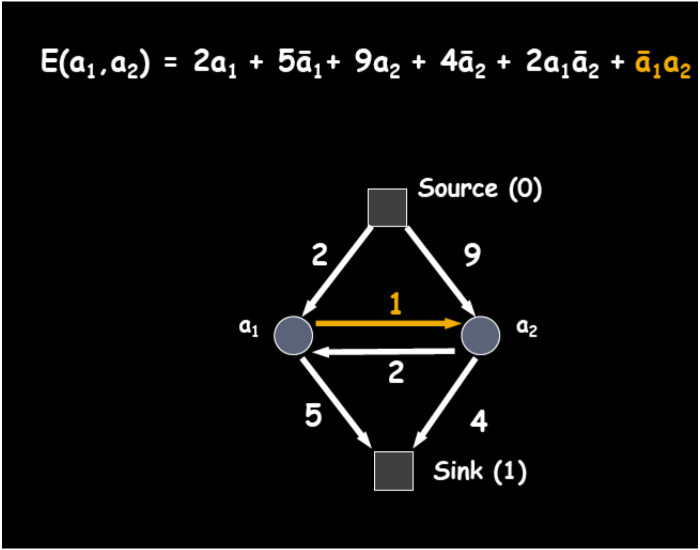
[Source: P. Kohli]

Graph Construction



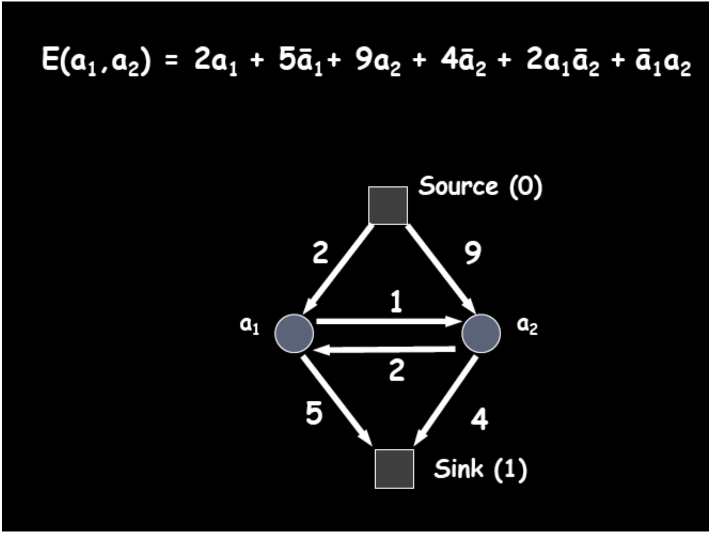
[Source: P. Kohli]

Graph Construction



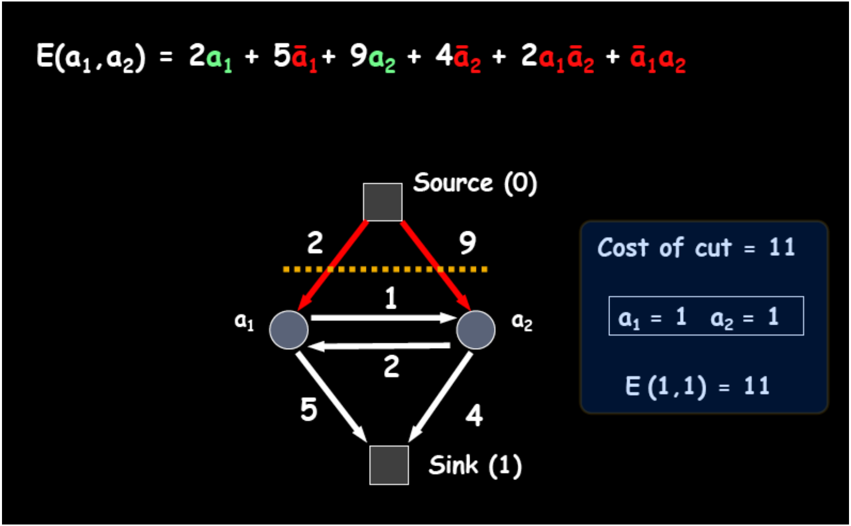
[Source: P. Kohli]

Graph Construction



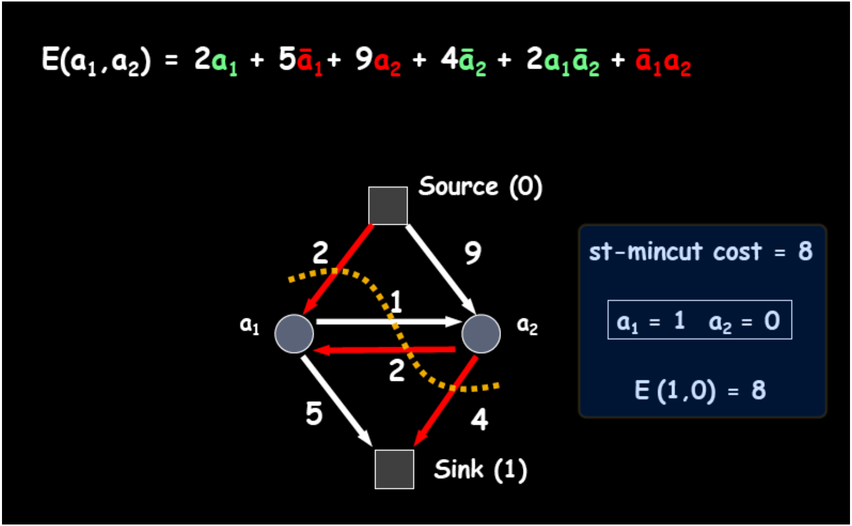
[Source: P. Kohli]

Graph Construction



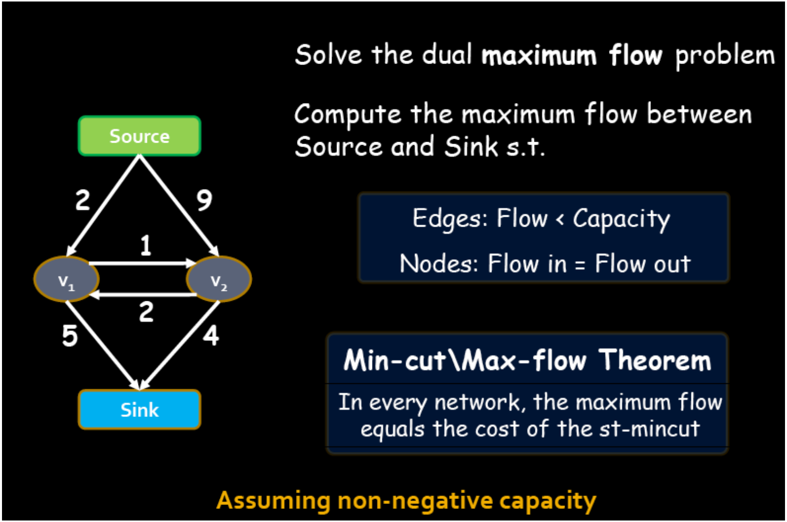
[Source: P. Kohli]

Graph Construction



[Source: P. Kohli]

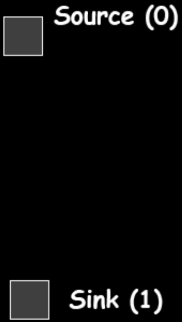
How to compute the St-mincut?



[Source: P. Kohli]

How does the code look like

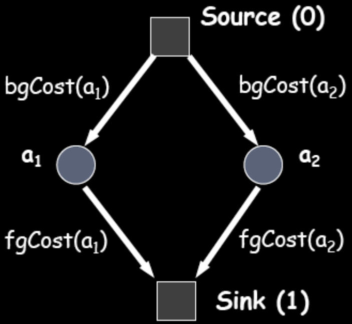
```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

How does the code look like

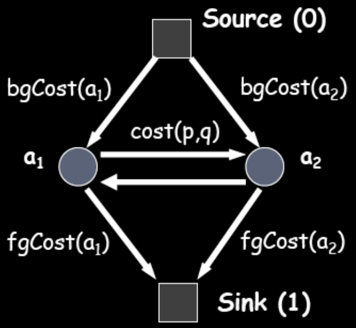
```
Graph *g;

For all pixels p

    /* Add a node to the graph */
    nodeID(p) = g->add_node();

    /* Set cost of terminal edges */
    set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
    add_weights(nodeID(p), nodeID(q), cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

How does the code look like

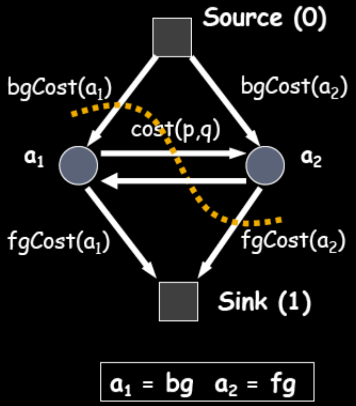
```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q), cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

How does the code look like

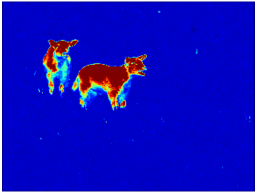
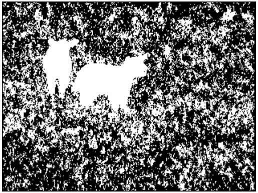
```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

Example: Figure-Ground Segmentation

Binary labeling problem

(Original) (Color model) (Indep. Prediction)

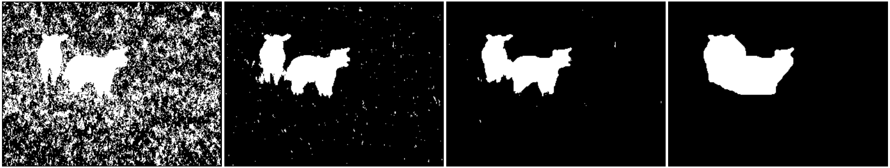
Figure : from [Nowozin et al]

Example: Figure-Ground Segmentation

- Markov Random Field

$$E(\mathbf{y}, \mathbf{x}, \mathbf{w}) = \sum_i \log p(y_i|x_i) + w \sum_{(i,j)\in\mathcal{E}} C(x_i, x_j) I(y_i \neq y_j)$$

with $C(x_i, x_j) = \exp(\gamma||x_i - x_j||^2)$, and $w \geq 0$.



(w=0) (w small) (w medium) (large w)

Figure : from [Nowozin et al]

- Why do we need the condition $w \geq 0$?

Generalization to Multi-label Problems

- Optimal solution is not possible anymore
- Solve to optimality subproblems that include current iterate
- This guarantees decrease in the objective

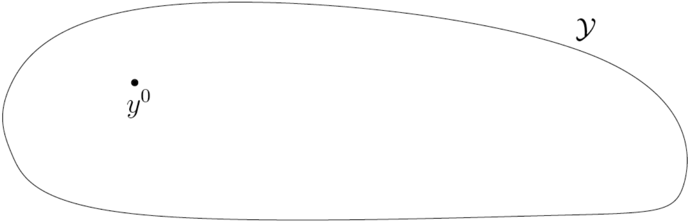


Figure : from [Nowozin et al]

Generalization to Multi-label Problems

- Optimal solution is not possible anymore
- Solve to optimality subproblems that include current iterate
- This guarantees decrease in the objective

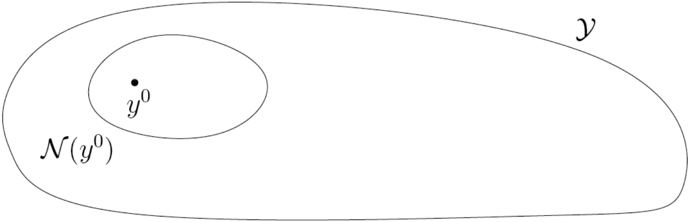


Figure : from [Nowozin et al]

Generalization to Multi-label Problems

- Optimal solution is not possible anymore
- Solve to optimality subproblems that include current iterate
- This guarantees decrease in the objective

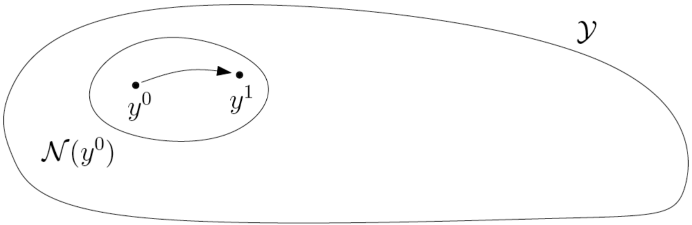


Figure : from [Nowozin et al]

Generalization to Multi-label Problems

- Optimal solution is not possible anymore
- Solve to optimality subproblems that include current iterate
- This guarantees decrease in the objective

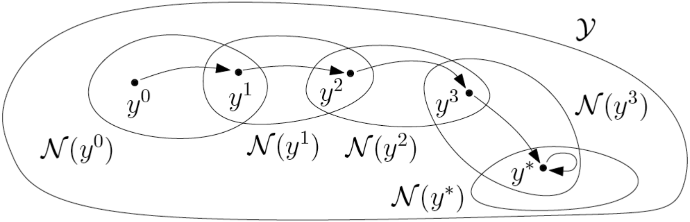


Figure : from [Nowozin et al]

Metric vs Semimetric

Two general classes of pairwise interactions

- **Metric** if it satisfies for any set of labels α, β, γ

$$
\begin{aligned}
V(\alpha,\beta) = 0 &\leftrightarrow \alpha = \beta \\
V(\alpha,\beta) &= V(\beta,\alpha) \geq 0 \\
V(\alpha,\beta) &\leq V(\alpha,\gamma) + V(\gamma,\beta)
\end{aligned}
$$

- **Semi-metric** if it satisfies for any set of labels α, β, γ

$$
\begin{aligned}
V(\alpha,\beta) = 0 &\leftrightarrow \alpha = \beta \\
V(\alpha,\beta) &= V(\beta,\alpha) \geq 0
\end{aligned}
$$

Metric vs Semimetric

Two general classes of pairwise interactions

- **Metric** if it satisfies for any set of labels α, β, γ

$$\begin{aligned} V(\alpha,\beta) = 0 &\leftrightarrow \alpha = \beta \\ V(\alpha,\beta) &= V(\beta,\alpha) \geq 0 \\ V(\alpha,\beta) &\leq V(\alpha,\gamma) + V(\gamma,\beta) \end{aligned}$$

- **Semi-metric** if it satisfies for any set of labels α, β, γ

$$\begin{aligned} V(\alpha,\beta) = 0 &\leftrightarrow \alpha = \beta \\ V(\alpha,\beta) &= V(\beta,\alpha) \geq 0 \end{aligned}$$

Examples for 1D label set

- Truncated quadratic is a semi-metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|^2)$$

with K a constant.

- Truncated absolute distance is a metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|)$$

with K a constant.

Examples for 1D label set

- Truncated quadratic is a semi-metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|^2)$$

with K a constant.

- Truncated absolute distance is a metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|)$$

with K a constant.

- Potts model is a metric

$$V(\alpha, \beta) = K \cdot T(\alpha \neq \beta)$$

with $T(\cdot) = 1$ if the argument is true and 0 otherwise.

Examples for 1D label set

- Truncated quadratic is a semi-metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|^2)$$

  with K a constant.

- Truncated absolute distance is a metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|)$$

  with K a constant.

- Potts model is a metric

$$V(\alpha, \beta) = K \cdot T(\alpha \neq \beta)$$

  with $T(\cdot) = 1$ if the argument is true and 0 otherwise.

Move Making Algorithms

- **Alpha Expansion**: Checks if current nodes want to switch to label α
- **Alpha - Beta Swaps**: Checks if a node with class α wants to switch to β.
- Binary problems that can be solve exactly for certain type of potentials

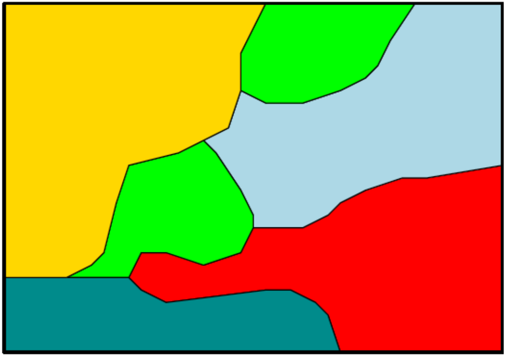


Figure : Alpha-beta Swaps. Figure from [Nowozin et al]

Move Making Algorithms

- **Alpha Expansion**: Checks if current nodes want to switch to label α
- **Alpha - Beta Swaps**: Checks if a node with class α wants to switch to β.
- Binary problems that can be solve exactly for certain type of potentials

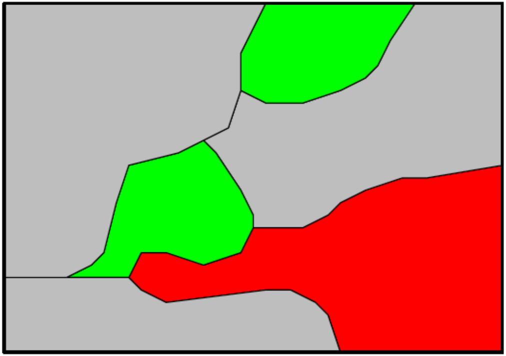


Figure : Alpha-beta Swaps. Figure from [Nowozin et al]

Move Making Algorithms

- **Alpha Expansion**: Checks if current nodes want to switch to label α
- **Alpha - Beta Swaps**: Checks if a node with class α wants to switch to β.
- Binary problems that can be solve exactly for certain type of potentials

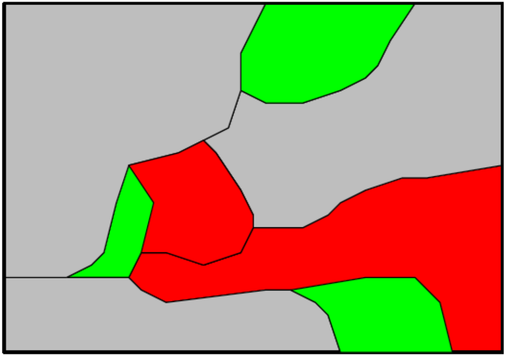


Figure : Alpha-beta Swaps. Figure from [Nowozin et al]

Move Making Algorithms

- **Alpha Expansion**: Checks if current nodes want to switch to label α
- **Alpha - Beta Swaps**: Checks if a node with class α wants to switch to β.
- Binary problems that can be solve exactly for certain type of potentials

Figure : Alpha-beta Swaps. Figure from [Nowozin et al]

Move Making Algorithms

- **Alpha Expansion**: Checks if current nodes want to switch to label α
- **Alpha - Beta Swaps**: Checks if a node with class α wants to switch to β.
- Binary problems that can be solve exactly for certain type of potentials

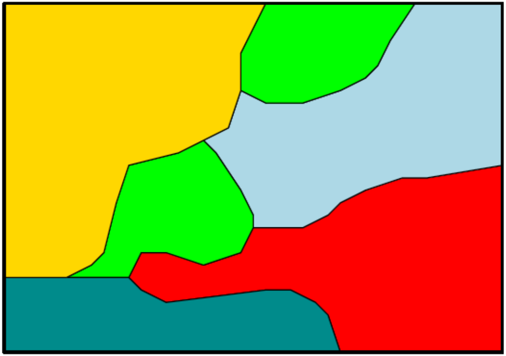


Figure : Alpha-beta Swaps. Figure from [Nowozin et al]

Binary Moves

- $\alpha - \beta$ moves works for semi-metrics
- α expansion works for V being a metric

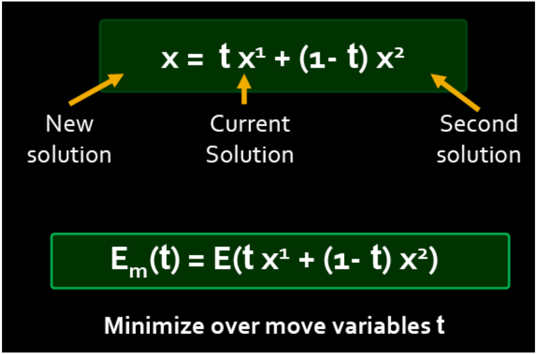
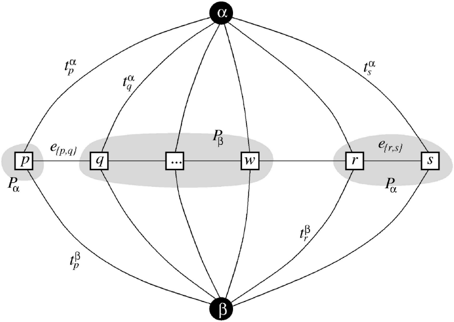


Figure : from P. Kohli tutorial on graph-cuts

- For certain x^1 and x^2, the move energy is sub-modular

Graph Construction

- The set of vertices includes the two terminals α and β, as well as image pixels p in the sets $\mathcal{P}_\alpha$ and $\mathcal{P}_\beta$ (i.e., $f_p \in \{\alpha, \beta\}$).
- Each pixel $p \in \mathcal{P}_{\alpha\beta}$ is connected to the terminals α and β, called *t*-links.
- Each set of pixels $p, q \in \mathcal{P}_{\alpha\beta}$ which are neighbors is connected by an edge $e_{p,q}$



edge	weight	for
t_p^α	$D_p(\alpha) + \sum_{q \in \mathcal{N}_p, q \notin \mathcal{P}_{\alpha\beta}} V(\alpha, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
t_p^β	$D_p(\beta) + \sum_{q \in \mathcal{N}_p, q \notin \mathcal{P}_{\alpha\beta}} V(\beta, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
$e_{\{p,q\}}$	$V(\alpha, \beta)$	$\{p,q\} \in \mathcal{N}$, $p, q \in \mathcal{P}_{\alpha\beta}$

Learning in graphical models

Learning Tasks

- Estimation of the parameters $\mathbf{w}$

$$E_F(\mathbf{y}_F) = -\mathbf{w}^T \phi_F(\mathbf{y}_F)$$

- Learn the structure of the model
- Learn with hidden variables

Learning the parameters

- Log-loss learning
- Max margin learning
- One parameter extensions
- Pseudolikelihood
- Perturb and MAP approaches
- Contrastive Divergence
- ...

Supervised Learning

- We are given a dataset of $\mathcal{S} = \{(\mathbf{x}^i, \mathbf{y}^i), \cdots, (\mathbf{x}^N, \mathbf{y}^N)\}$
- We also have the task loss that we want to minimize $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

Supervised Learning

- We are given a dataset of $\mathcal{S} = \{(\mathbf{x}^i, \mathbf{y}^i), \cdots, (\mathbf{x}^N, \mathbf{y}^N)\}$
- We also have the task loss that we want to minimize $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
- We want to find the weights by solving

$$\min_{\mathbf{w}} \mathbb{E}_{(x,y)\sim\mathcal{D}}\{\Delta(y, f(x))\}$$

with $f(x) = \operatorname{argmax}_{y\in\mathcal{Y}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$

Supervised Learning

- We are given a dataset of $\mathcal{S} = \{(\mathbf{x}^i, \mathbf{y}^i), \cdots, (\mathbf{x}^N, \mathbf{y}^N)\}$
- We also have the task loss that we want to minimize $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
- We want to find the weights by solving

$$\min_{\mathbf{w}} \mathbb{E}_{(x,y)\sim\mathcal{D}}\{\Delta(y, f(x))\}$$

with $f(x) = \underset{y\in\mathcal{Y}}{\text{argmax}}\, \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$

- This is difficult, so we can replace it by an empirical estimate, a surrogate loss and add regularizer to prevent overfitting

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{D}} \ell(\mathbf{w}, x, y) + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

Supervised Learning

- We are given a dataset of $\mathcal{S} = \{(\mathbf{x}^i, \mathbf{y}^i), \cdots, (\mathbf{x}^N, \mathbf{y}^N)\}$
- We also have the task loss that we want to minimize $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
- We want to find the weights by solving

$$\min_{\mathbf{w}} \mathbb{E}_{(x,y)\sim\mathcal{D}}\{\Delta(y, f(x))\}$$

  with $f(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$

- This is difficult, so we can replace it by an empirical estimate, a surrogate loss and add regularizer to prevent overfitting

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{D}} \ell(\mathbf{w}, x, y) + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- Typical supervised learning algorithms are convex.

Supervised Learning

- We are given a dataset of $\mathcal{S} = \{(\mathbf{x}^i, \mathbf{y}^i), \cdots, (\mathbf{x}^N, \mathbf{y}^N)\}$
- We also have the task loss that we want to minimize $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
- We want to find the weights by solving

$$\min_{\mathbf{w}} \mathbb{E}_{(x,y)\sim\mathcal{D}}\{\Delta(y, f(x))\}$$

with $f(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$

- This is difficult, so we can replace it by an empirical estimate, a surrogate loss and add regularizer to prevent overfitting

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{D}} \ell(\mathbf{w}, x, y) + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- Typical supervised learning algorithms are convex.
- Why is this problem difficult?

Supervised Learning

- We are given a dataset of $\mathcal{S} = \{(\mathbf{x}^i, \mathbf{y}^i), \cdots, (\mathbf{x}^N, \mathbf{y}^N)\}$
- We also have the task loss that we want to minimize $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
- We want to find the weights by solving

$$\min_{\mathbf{w}} \mathbb{E}_{(x,y)\sim\mathcal{D}}\{\Delta(y, f(x))\}$$

with $f(x) = \operatorname*{argmax}_{y\in\mathcal{Y}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$

- This is difficult, so we can replace it by an empirical estimate, a surrogate loss and add regularizer to prevent overfitting

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{D}} \ell(\mathbf{w}, x, y) + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- Typical supervised learning algorithms are convex.
- Why is this problem difficult?

Max-margin Learning

- Regularized Risk Minimization

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \ell(\mathbf{w}, x, y) + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- In structured SVMs

$$\ell_{hinge}(\mathbf{w}, x, y) = \max_{\hat{y}\in\mathcal{Y}} \left\{\Delta(y, \hat{y}) + \mathbf{w}^\top\Phi(x, \hat{y}) - \mathbf{w}^\top\Phi(x, y)\right\}$$

Max-margin Learning

- Regularized Risk Minimization

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \ell(\mathbf{w},x,y) + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- In structured SVMs

$$\ell_{hinge}(\mathbf{w},x,y) = \max_{\hat{y}\in\mathcal{Y}} \left\{\Delta(y,\hat{y}) + \mathbf{w}^\top\Phi(x,\hat{y}) - \mathbf{w}^\top\Phi(x,y)\right\}$$

- Optimize the unconstrained problem

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \max_{\hat{y}\in\mathcal{Y}} \left\{\Delta(y,\hat{y}) + \mathbf{w}^\top\Phi(x,\hat{y}) - \mathbf{w}^\top\Phi(x,y)\right\} + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- Convex but non-smooth.
- Use sub gradient methods

Max-margin Learning

- Regularized Risk Minimization

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \ell(\mathbf{w},x,y) + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- In structured SVMs

$$\ell_{hinge}(\mathbf{w},x,y) = \max_{\hat{y}\in\mathcal{Y}} \left\{\Delta(y,\hat{y}) + \mathbf{w}^\top\Phi(x,\hat{y}) - \mathbf{w}^\top\Phi(x,y)\right\}$$

- Optimize the unconstrained problem

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \max_{\hat{y}\in\mathcal{Y}} \left\{\Delta(y,\hat{y}) + \mathbf{w}^\top\Phi(x,\hat{y}) - \mathbf{w}^\top\Phi(x,y)\right\} + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- Convex but non-smooth.
- Use sub gradient methods

Equivalent Formulation

- Optimize the unconstrained problem

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \max_{\hat{y}\in\mathcal{Y}} \left\{ \Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y}) - \mathbf{w}^\top \Phi(x,y) \right\} + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- Write as constraints

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \xi_n^2 + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

$$s.t. \quad \max_{\hat{y}\in\mathcal{Y}} \left\{ \Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y}) - \mathbf{w}^\top \Phi(x,y) \right\} \leq \xi_n$$

Equivalent Formulation

- Optimize the unconstrained problem

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \max_{\hat{y}\in\mathcal{Y}} \left\{ \Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y}) - \mathbf{w}^\top \Phi(x,y) \right\} + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- Write as constraints

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \xi_n^2 + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

$$s.t. \quad \max_{\hat{y}\in\mathcal{Y}} \left\{ \Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y}) - \mathbf{w}^\top \Phi(x,y) \right\} \leq \xi_n$$

- Or equivalently

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \xi_n^2 + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

$$s.t. \quad \forall \hat{y} \quad \ell(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y}) - \mathbf{w}^\top \Phi(x,y) \leq \xi_n$$

- Use cutting plane methods as exp. many constraints

Equivalent Formulation

- Optimize the unconstrained problem

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \max_{\hat{y}\in\mathcal{Y}} \left\{\Delta(y,\hat{y}) + \mathbf{w}^\top\Phi(x,\hat{y}) - \mathbf{w}^\top\Phi(x,y)\right\} + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- Write as constraints

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \xi_n^2 + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

$$s.t. \quad \max_{\hat{y}\in\mathcal{Y}} \left\{\Delta(y,\hat{y}) + \mathbf{w}^\top\Phi(x,\hat{y}) - \mathbf{w}^\top\Phi(x,y)\right\} \leq \xi_n$$

- Or equivalently

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \xi_n^2 + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

$$s.t. \ \ \forall\hat{y} \ \ \ell(y,\hat{y}) + \mathbf{w}^\top\Phi(x,\hat{y}) - \mathbf{w}^\top\Phi(x,y) \leq \xi_n$$

- Use cutting plane methods as exp. many constraints

Log-loss Learning

- Regularized Risk Minimization

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \ell(\mathbf{w}, x, y) + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- CRF loss: The conditional distribution is

$$p_{x,y}(\hat{y};\mathbf{w}) = \frac{1}{Z(x,y)} \exp\left(\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

$$Z(x,y) = \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

where $\Delta(y,\hat{y})$ is a prior distribution and $Z(x,y)$ the partition function, and

$$\ell_{log}(\mathbf{w}, x, y) = \ln \frac{1}{p_{x,y}(y;\mathbf{w})}.$$

Log-loss Learning

- Regularized Risk Minimization

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \ell(\mathbf{w},x,y) + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- CRF loss: The conditional distribution is

$$p_{x,y}(\hat{y};\mathbf{w}) = \frac{1}{Z(x,y)} \exp\left(\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

$$Z(x,y) = \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

where $\Delta(y,\hat{y})$ is a prior distribution and $Z(x,y)$ the partition function, and

$$\ell_{log}(\mathbf{w},x,y) = \ln \frac{1}{p_{x,y}(y;\mathbf{w})}.$$

- Convex problem
- Problem: to do gradient descent I need to compute Z

Log-loss Learning

- Regularized Risk Minimization

$$\min_{\mathbf{w}} \sum_{(x,y)\in\mathcal{S}} \ell(\mathbf{w}, x, y) + \frac{C}{p}\|\mathbf{w}\|_p^p,$$

- CRF loss: The conditional distribution is

$$p_{x,y}(\hat{y};\mathbf{w}) = \frac{1}{Z(x,y)} \exp\left(\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

$$Z(x,y) = \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

where $\Delta(y,\hat{y})$ is a prior distribution and $Z(x,y)$ the partition function, and

$$\ell_{log}(\mathbf{w}, x, y) = \ln \frac{1}{p_{x,y}(y;\mathbf{w})}.$$

- Convex problem
- Problem: to do gradient descent I need to compute Z

Relation between loss functions

- The CRF program is

$$\text{(CRF)} \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p} \|\mathbf{w}\|_p^p \right\},$$

where $(x,y) \in \mathcal{S}$ ranges over training pairs and $\mathbf{d} = \sum_{(x,y)\in\mathcal{S}} \Phi(x,y)$ is the vector of empirical means, and

$$Z(x,y) = \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

- In structured SVMs

$$\text{(structured SVM)} \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \max_{\hat{y}\in\mathcal{Y}} \left\{\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right\} - \mathbf{d}^\top \mathbf{w} + \frac{C}{p} \|\mathbf{w}\|_p^p \right\},$$

Relation between loss functions

- The CRF program is

$$(\text{CRF}) \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

  where $(x,y) \in \mathcal{S}$ ranges over training pairs and $\mathbf{d} = \sum_{(x,y)\in\mathcal{S}} \Phi(x,y)$ is the vector of empirical means, and

$$Z(x,y) = \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right)$$

- In structured SVMs

$$(\text{structured SVM}) \qquad \min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \max_{\hat{y}\in\mathcal{Y}} \left\{\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})\right\} - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

A family of structure prediction problems

- One parameter extension of CRFs and structured SVMs [Hazan & Urtasun, NIPS 2010]

$$\min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z_\epsilon(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

  d is the empirical means, and

$$\ln Z_\epsilon(x,y) = \epsilon \ln \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\frac{\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})}{\epsilon}\right)$$

- CRF if $\epsilon = 1$, Structured SVM if $\epsilon = 0$ respectively.
- One can devise a single algorithm to solve both problems

A family of structure prediction problems

- One parameter extension of CRFs and structured SVMs [Hazan & Urtasun, NIPS 2010]

$$\min_{\mathbf{w}} \left\{ \sum_{(x,y)\in\mathcal{S}} \ln Z_\epsilon(x,y) - \mathbf{d}^\top \mathbf{w} + \frac{C}{p}\|\mathbf{w}\|_p^p \right\},$$

  d is the empirical means, and

$$\ln Z_\epsilon(x,y) = \epsilon \ln \sum_{\hat{y}\in\mathcal{Y}} \exp\left(\frac{\Delta(y,\hat{y}) + \mathbf{w}^\top \Phi(x,\hat{y})}{\epsilon}\right)$$

- CRF if $\epsilon = 1$, Structured SVM if $\epsilon = 0$ respectively.
- One can devise a single algorithm to solve both problems

Structure Prediction for Scene Understanding II

Raquel Urtasun

University of Toronto

June 20, 2014

Structured Prediction in Practice

Recipe for Success using Structure Prediction

- What are my random variables?
- How are they related? i.e., graph

Recipe for Success using Structure Prediction

- What are my random variables?
- How are they related? i.e., graph

Recipe for Success using Structure Prediction

- What are my random variables?
- How are they related? i.e., graph
- How do I encode my prior knowledge about the problem?

$$E(y_1, \cdots, y_n, \mathbf{x}) = \sum_{r \in \mathcal{R}} \mathbf{w}_r^T \phi_r(\mathbf{y}_r, \mathbf{x})$$

Recipe for Success using Structure Prediction

- What are my random variables?
- How are they related? i.e., graph
- How do I encode my prior knowledge about the problem?

$$E(y_1, \cdots, y_n, \mathbf{x}) = \sum_{r \in \mathcal{R}} \mathbf{w}_r^T \phi_r(\mathbf{y}_r, \mathbf{x})$$

- **Advise**: Forget about probabilities in your potentials, the partition function will take care of that!

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp(-E(\mathbf{y}, \mathbf{x}))$$

Recipe for Success using Structure Prediction

- What are my random variables?
- How are they related? i.e., graph
- How do I encode my prior knowledge about the problem?

$$E(y_1, \cdots, y_n, \mathbf{x}) = \sum_{r \in \mathcal{R}} \mathbf{w}_r^T \phi_r(\mathbf{y}_r, \mathbf{x})$$

- **Advise**: Forget about probabilities in your potentials, the partition function will take care of that!

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp(-E(\mathbf{y}, \mathbf{x}))$$

- How can I do inference? Why is this complicated?

$$\min_{y_1, \cdots, y_n} E(y_1, \cdots, y_n)$$

Recipe for Success using Structure Prediction

- What are my random variables?
- How are they related? i.e., graph
- How do I encode my prior knowledge about the problem?

$$E(y_1, \cdots, y_n, \mathbf{x}) = \sum_{r \in \mathcal{R}} \mathbf{w}_r^T \phi_r(\mathbf{y}_r, \mathbf{x})$$

- **Advise**: Forget about probabilities in your potentials, the partition function will take care of that!

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp(-E(\mathbf{y}, \mathbf{x}))$$

- How can I do inference? Why is this complicated?

$$\min_{y_1, \cdots, y_n} E(y_1, \cdots, y_n)$$

- If you know how to do inference you will know how to do learning! Where does the complication come from?

Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? $\rightarrow$ there is still room for hackers!
- It allows you to abstract and encode models to solve your problems

Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? $\rightarrow$ there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems

Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? $\rightarrow$ there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems
- Easy to reason jointly about multiple problems

Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? $\rightarrow$ there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems
- Easy to reason jointly about multiple problems
- Why do I care about holistic (i.e., joint) models?

Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? $\rightarrow$ there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems
- Easy to reason jointly about multiple problems
- Why do I care about holistic (i.e., joint) models?
- Well understood inference algorithms, some of them exact!

Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? $\rightarrow$ there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems
- Easy to reason jointly about multiple problems
- Why do I care about holistic (i.e., joint) models?
- Well understood inference algorithms, some of them exact!
- Good learning algorithms exist as well

Why Would I Use Structure Prediction?

- Why to worry about math if I can hack up something quickly? $\rightarrow$ there is still room for hackers!
- It allows you to abstract and encode models to solve your problems
- Captures well the combinatorial structure of some problems
- Easy to reason jointly about multiple problems
- Why do I care about holistic (i.e., joint) models?
- Well understood inference algorithms, some of them exact!
- Good learning algorithms exist as well

What's not so good?

- Use as a keyword, approaches that don't think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials

What's not so good?

- Use as a keyword, approaches that don't think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!

What's not so good?

- Use as a keyword, approaches that don't think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!
- Do I need to understand inference? Yes, yes and yes! I don't think this is a negative point though

What's not so good?

- Use as a keyword, approaches that don't think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!
- Do I need to understand inference? Yes, yes and yes! I don't think this is a negative point though
- Is a log-linear model expressive enough?

What's not so good?

- Use as a keyword, approaches that don't think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!
- Do I need to understand inference? Yes, yes and yes! I don't think this is a negative point though
- Is a log-linear model expressive enough?
- Where does the structure come from?

What's not so good?

- Use as a keyword, approaches that don't think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!
- Do I need to understand inference? Yes, yes and yes! I don't think this is a negative point though
- Is a log-linear model expressive enough?
- Where does the structure come from?
- Can I learn everything from unlabeled data? How deep are you?

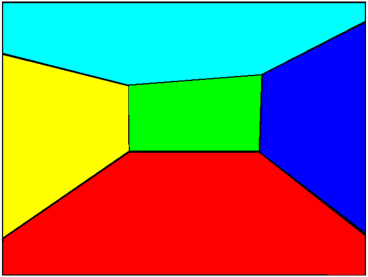
What's not so good?

- Use as a keyword, approaches that don't think about how the problem is represented, how the energy looks like, etc.
- Particularly overloaded terms, e.g., high-order potentials
- Problems with continuous variables: we need better algorithms!
- Do I need to understand inference? Yes, yes and yes! I don't think this is a negative point though
- Is a log-linear model expressive enough?
- Where does the structure come from?
- Can I learn everything from unlabeled data? How deep are you?

First task: 3D indoor scene understanding

3D layout for Indoors

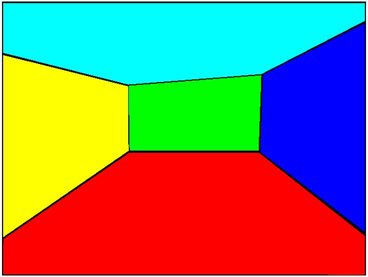
Task: Estimate the 3D layout from a single image



- What's the metric? how do I know if I did well?
- How would you parameterize this problem? (i.e., what are your random variables?)

3D layout for Indoors

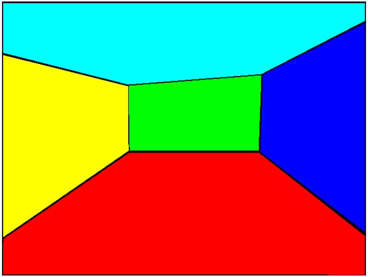
Task: Estimate the 3D layout from a single image



- What's the metric? how do I know if I did well?
- How would you parameterize this problem? (i.e., what are your random variables?)
- What prior knowledge would you like to encode?

3D layout for Indoors

Task: Estimate the 3D layout from a single image



- What's the metric? how do I know if I did well?
- How would you parameterize this problem? (i.e., what are your random variables?)
- What prior knowledge would you like to encode?

3D layout for Indoors

- Isn't this a segmentation task where each pixel can be labeled as a wall?

3D layout for Indoors

- Isn't this a segmentation task where each pixel can be labeled as a wall?
- Let's start with the most simple parameterization: split the image into super pixels, and for each define

$$y_i \in \{1, \cdots, 5\}$$

the label the super pixel is associated with

3D layout for Indoors

- Isn't this a segmentation task where each pixel can be labeled as a wall?
- Let's start with the most simple parameterization: split the image into super pixels, and for each define

$$y_i \in \{1, \cdots, 5\}$$

  the label the super pixel is associated with
- Define the energy as

$$E(y_1, \cdots, y_n, \mathbf{x}) = \sum_{r \in \mathcal{R}} \mathbf{w}_r^T \phi_r(\mathbf{y}_r, \mathbf{x})$$

- What are the $\phi_r(\mathbf{y}_r, \mathbf{x})$?

3D layout for Indoors

- Isn't this a segmentation task where each pixel can be labeled as a wall?
- Let's start with the most simple parameterization: split the image into super pixels, and for each define

$$y_i \in \{1, \cdots, 5\}$$

the label the super pixel is associated with

- Define the energy as

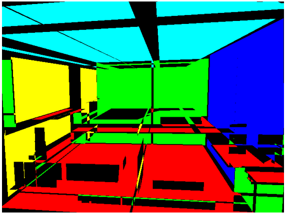
$$E(y_1, \cdots, y_n, \mathbf{x}) = \sum_{r \in \mathcal{R}} \mathbf{w}_r^T \phi_r(\mathbf{y}_r, \mathbf{x})$$

- What are the $\phi_r(\mathbf{y}_r, \mathbf{x})$?

Geometric Features as Unaries

- Orientation maps [Leet el al 09], geometric context [Hoiem et al. 05]

original image



orientation map

geometric context

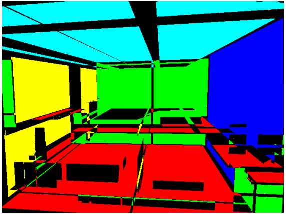
How do I construct my unaries $\phi_i(\mathbf{x}, y_i)$?

- What are my pairwise potentials $\phi_{ij}(\mathbf{x}, y_i, y_j)$?

Geometric Features as Unaries

- Orientation maps [Leet el al 09], geometric context [Hoiem et al. 05]

original image



orientation map

geometric context

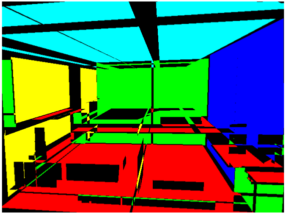
How do I construct my unaries $\phi_i(\mathbf{x}, y_i)$?

- What are my pairwise potentials $\phi_{ij}(\mathbf{x}, y_i, y_j)$?
- What's the problem with smoothness potentials?

Geometric Features as Unaries

- Orientation maps [Leet el al 09], geometric context [Hoiem et al. 05]

original image



orientation map

geometric context

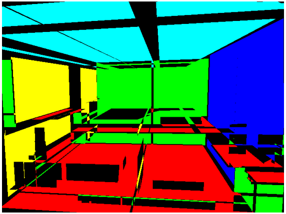
How do I construct my unaries $\phi_i(\mathbf{x}, y_i)$?

- What are my pairwise potentials $\phi_{ij}(\mathbf{x}, y_i, y_j)$?
- What's the problem with smoothness potentials?
- Are we missing something? What extra knowledge do we have?

Geometric Features as Unaries

- Orientation maps [Leet el al 09], geometric context [Hoiem et al. 05]

original image



orientation map

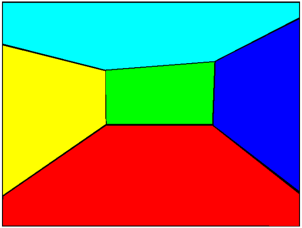
geometric context

How do I construct my unaries $\phi_i(\mathbf{x}, y_i)$?

- What are my pairwise potentials $\phi_{ij}(\mathbf{x}, y_i, y_j)$?
- What's the problem with smoothness potentials?
- Are we missing something? What extra knowledge do we have?

Manhattan World for Segmentation

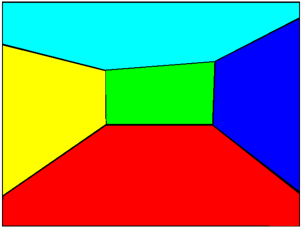
- Labels are not appearing at random in the image



- We can encode that the world is Manhattan by expressing **ordering** constraints

Manhattan World for Segmentation

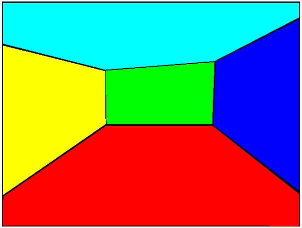
- Labels are not appearing at random in the image



- We can encode that the world is Manhattan by expressing **ordering** constraints
- What would that be?

Manhattan World for Segmentation

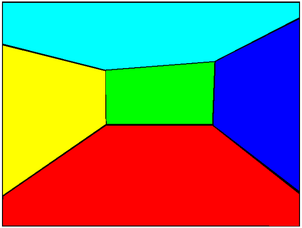
- Labels are not appearing at random in the image



- We can encode that the world is Manhattan by expressing **ordering** constraints
- What would that be?
- What's the order of the potentials?

Manhattan World for Segmentation

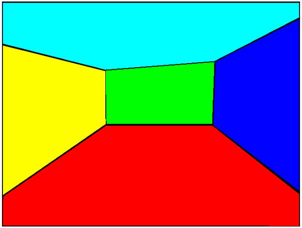
- Labels are not appearing at random in the image



- We can encode that the world is Manhattan by expressing **ordering** constraints
- What would that be?
- What's the order of the potentials?
- Can we do inference easily?

Manhattan World for Segmentation

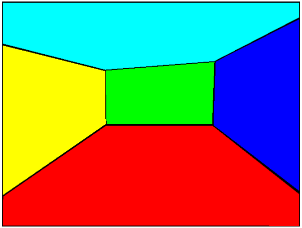
- Labels are not appearing at random in the image



- We can encode that the world is Manhattan by expressing **ordering** constraints
- What would that be?
- What's the order of the potentials?
- Can we do inference easily?
- Which algorithm will you use? would it take a long time? would it be optimal?

Manhattan World for Segmentation

- Labels are not appearing at random in the image



- We can encode that the world is Manhattan by expressing **ordering** constraints
- What would that be?
- What's the order of the potentials?
- Can we do inference easily?
- Which algorithm will you use? would it take a long time? would it be optimal?

Encoding Manhattan World Structure

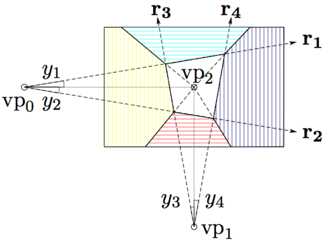
- Let's assume that I can compute vanishing points
- How should I express the problem? how many degrees of freedom do I have?

Encoding Manhattan World Structure

- Let's assume that I can compute vanishing points
- How should I express the problem? how many degrees of freedom do I have?

Encoding Manhattan World Structure

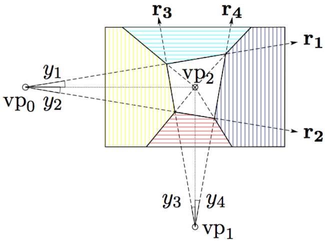
- Let's assume that I can compute vanishing points
- How should I express the problem? how many degrees of freedom do I have?
- We parameterize a layout with 4 variables $y_i \in \mathcal{Y}$, $i \in \{1, ..., 4\}$ [Lee et al. 09]



- What have I lost with respect to before?

Encoding Manhattan World Structure

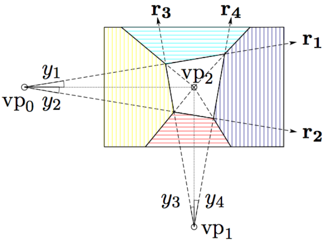
- Let's assume that I can compute vanishing points
- How should I express the problem? how many degrees of freedom do I have?
- We parameterize a layout with 4 variables $y_i \in \mathcal{Y}$, $i \in \{1, ..., 4\}$ [Lee et al. 09]



- What have I lost with respect to before?
- What have I won?

Encoding Manhattan World Structure

- Let's assume that I can compute vanishing points
- How should I express the problem? how many degrees of freedom do I have?
- We parameterize a layout with 4 variables $y_i \in \mathcal{Y}$, $i \in \{1, ..., 4\}$ [Lee et al. 09]



- What have I lost with respect to before?
- What have I won?

Energy of the problem

- Let's define the energy. Which potentials will you use?

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T \phi(\mathbf{y}_r, x)$$

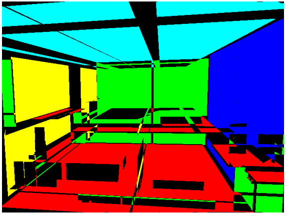
Energy of the problem

- Let's define the energy. Which potentials will you use?

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T \phi(\mathbf{y}_r, x)$$

- Let's start with the geometric features

original image



orientation map

geometric context

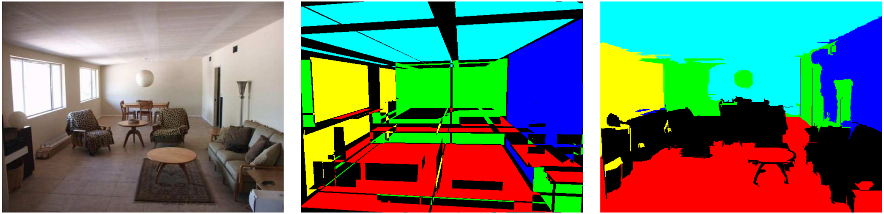
- We will like to maximize the yellow pixels in the left wall, green in the frontal wall, etc

Energy of the problem

- Let's define the energy. Which potentials will you use?

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T \phi(\mathbf{y}_r, x)$$

- Let's start with the geometric features



original image — orientation map — geometric context

- We will like to maximize the yellow pixels in the left wall, green in the frontal wall, etc
- We will also like to minimize the other colors in those walls, e.g., all but yellow in left wall

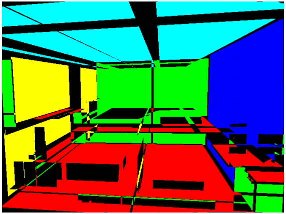
Energy of the problem

- Let's define the energy. Which potentials will you use?

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T \phi(\mathbf{y}_r, x)$$

- Let's start with the geometric features

original image



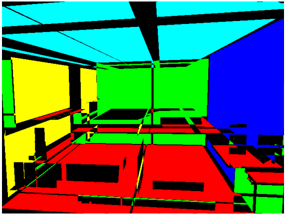
orientation map

geometric context

- We will like to maximize the yellow pixels in the left wall, green in the frontal wall, etc
- We will also like to minimize the other colors in those walls, e.g., all but yellow in left wall

More on energy

original image



orientation map

geometric context

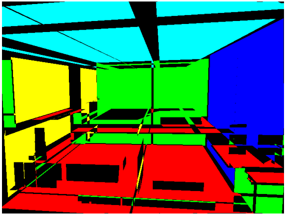
- How do I express this in my potentials?

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T \phi(\mathbf{y}_r, x)$$

- How many y_i's do I need to define them?

More on energy

original image



orientation map

geometric context

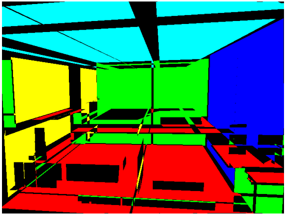
- How do I express this in my potentials?

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T \phi(\mathbf{y}_r, x)$$

- How many y_i's do I need to define them?
- Do I need other potentials?

More on energy

original image



orientation map

geometric context

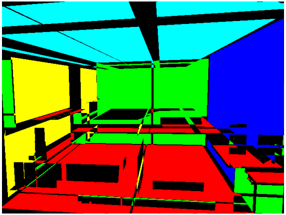
- How do I express this in my potentials?

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T \phi(\mathbf{y}_r, x)$$

- How many y_i's do I need to define them?
- Do I need other potentials?
- Why did I need more potentials than just geometric features before?

More on energy

original image



orientation map

geometric context

- How do I express this in my potentials?

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T \phi(\mathbf{y}_r, x)$$

- How many y_i's do I need to define them?
- Do I need other potentials?
- Why did I need more potentials than just geometric features before?

Inference

- Is inference easy in this model? Why?
- What can we do?

Inference

- Is inference easy in this model? Why?
- What can we do?
- Multi-label problem, message passing seems the best option

Inference

- Is inference easy in this model? Why?
- What can we do?
- Multi-label problem, message passing seems the best option
- Problem: High order potentials $\rightarrow$ very very slow !

Inference

- Is inference easy in this model? Why?
- What can we do?
- Multi-label problem, message passing seems the best option
- Problem: High order potentials $\rightarrow$ very very slow !
- Let's think about it for a second, maybe we can do something

Inference

- Is inference easy in this model? Why?
- What can we do?
- Multi-label problem, message passing seems the best option
- Problem: High order potentials $\rightarrow$ very very slow !
- Let's think about it for a second, maybe we can do something
- Remember we want to compute sum of features in faces, and search over all possible faces

Inference

- Is inference easy in this model? Why?
- What can we do?
- Multi-label problem, message passing seems the best option
- Problem: High order potentials $\rightarrow$ very very slow !
- Let's think about it for a second, maybe we can do something
- Remember we want to compute sum of features in faces, and search over all possible faces
- Let's first take a detour

Inference

- Is inference easy in this model? Why?
- What can we do?
- Multi-label problem, message passing seems the best option
- Problem: High order potentials $\rightarrow$ very very slow !
- Let's think about it for a second, maybe we can do something
- Remember we want to compute sum of features in faces, and search over all possible faces
- Let's first take a detour

Integral Images

- We are interested in computing the sum of some features inside a rectangle, and we want to vary the rectangle
- How can we do this efficiently?
- Compute the **sum area table**, also called **integral image**

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

$$s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l)$$

- This can be efficiently computed using a recursive (raster-scan) algorithm

$$s(i,j) = s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(i,j)$$

Integral Images

- We are interested in computing the sum of some features inside a rectangle, and we want to vary the rectangle
- How can we do this efficiently?
- Compute the **sum area table**, also called **integral image**

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

$$s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l)$$

- This can be efficiently computed using a recursive (raster-scan) algorithm

$$s(i,j) = s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(i,j)$$

- Then compute the sum on the rectangle by accessing 4 numbers

$$S([i_0,i_1] \times [j_0,j_1]) = s(i_1,j_1) - s(i_1,j_0-1) - s(i_0-1,j_1) + s(i_0-1,j_0-1)$$

Integral Images

- We are interested in computing the sum of some features inside a rectangle, and we want to vary the rectangle
- How can we do this efficiently?
- Compute the **sum area table**, also called **integral image**

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

3	5	12	*14*	17
4	11	19	24	31
9	17	28	38	46
13	24	37	**48**	62
15	30	44	59	81

$$s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l)$$

- This can be efficiently computed using a recursive (raster-scan) algorithm

$$s(i,j) = s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(i,j)$$

- Then compute the sum on the rectangle by accessing 4 numbers

$$S([i_0,i_1]\times[j_0,j_1]) = s(i_1,j_1) - s(i_1,j_0-1) - s(i_0-1,j_1) + s(i_0-1,j_0-1)$$

- Can we do something similar in our case?

Integral Images

- We are interested in computing the sum of some features inside a rectangle, and we want to vary the rectangle
- How can we do this efficiently?
- Compute the **sum area table**, also called **integral image**

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

3	5	12	*14*	17
4	11	19	24	31
9	17	28	38	46
13	24	37	**48**	62
15	30	44	59	81

$$s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l)$$

- This can be efficiently computed using a recursive (raster-scan) algorithm

$$s(i,j) = s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(i,j)$$

- Then compute the sum on the rectangle by accessing 4 numbers

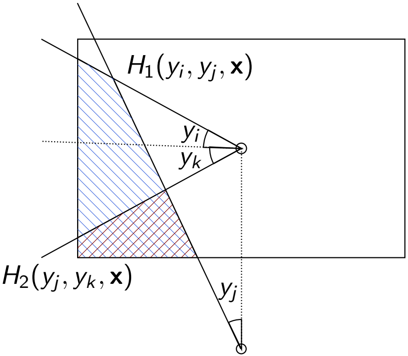
$$S([i_0,i_1] \times [j_0,j_1]) = s(i_1,j_1) - s(i_1,j_0-1) - s(i_0-1,j_1) + s(i_0-1,j_0-1)$$

- Can we do something similar in our case?

Generalization to 3D

- Faces are generalizations of rectangles
- We need to extend the concept of integral images to 3D
- This is called **integral geometry** [Schwing et al. 12a]
- How does this work?

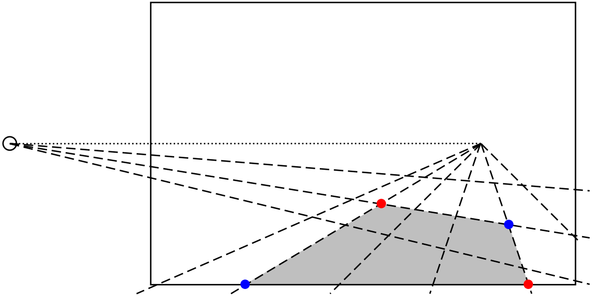
$$\phi_{\{left\_w\}}(y_i, y_j, y_k, \mathbf{x}) \quad = \quad H_1(y_i, y_j, \mathbf{x}) - H_2(y_j, y_k, \mathbf{x})$$



Generalization to 3D

- Faces are generalizations of rectangles
- We need to extend the concept of integral images to 3D
- This is called **integral geometry** [Schwing et al. 12a]
- How does this work?

$$\phi_{\{floor\}}(y_i, y_j, y_k, \mathbf{x}) = H_1(y_i, y_j, \mathbf{x}) - H_2(y_j, y_k, \mathbf{x})$$



What are the implications?

- We can now write the problem in terms of potentials of order at most 2

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T(\mathbf{y}_r, \mathbf{x})$$

and r only contains sets of 2 random variables

- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily

What are the implications?

- We can now write the problem in terms of potentials of order at most 2

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T (\mathbf{y}_r, \mathbf{x})$$

  and r only contains sets of 2 random variables

- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily

- Good news is that it still depends on pairwise potentials (which are accumulators) but there is quite a few more

What are the implications?

- We can now write the problem in terms of potentials of order at most 2

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T (\mathbf{y}_r, \mathbf{x})$$

  and r only contains sets of 2 random variables

- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily
- Good news is that it still depends on pairwise potentials (which are accumulators) but there is quite a few more
- Some of this r share the same weights, as they come from the integral geometry.

What are the implications?

- We can now write the problem in terms of potentials of order at most 2

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T (\mathbf{y}_r, \mathbf{x})$$

  and r only contains sets of 2 random variables

- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily
- Good news is that it still depends on pairwise potentials (which are accumulators) but there is quite a few more
- Some of this r share the same weights, as they come from the integral geometry.
- If they are not shared then they do not represent the same problem

What are the implications?

- We can now write the problem in terms of potentials of order at most 2

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T (\mathbf{y}_r, \mathbf{x})$$

  and r only contains sets of 2 random variables

- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily
- Good news is that it still depends on pairwise potentials (which are accumulators) but there is quite a few more
- Some of this r share the same weights, as they come from the integral geometry.
- If they are not shared then they do not represent the same problem
- This speed ups the message passing inference by a few orders of magnitude

What are the implications?

- We can now write the problem in terms of potentials of order at most 2

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T(\mathbf{y}_r, \mathbf{x})$$

  and r only contains sets of 2 random variables
- Life is a bit more complicated than what I showed you as I was varying the parameterization to make you understand easily
- Good news is that it still depends on pairwise potentials (which are accumulators) but there is quite a few more
- Some of this r share the same weights, as they come from the integral geometry.
- If they are not shared then they do not represent the same problem
- This speed ups the message passing inference by a few orders of magnitude

Exact Inference?

- Can we compute the optimal solution?
- The graph of the previous problem loops

Exact Inference?

- Can we compute the optimal solution?
- The graph of the previous problem loops
- Message passing will not give the optimal

Exact Inference?

- Can we compute the optimal solution?
- The graph of the previous problem loops
- Message passing will not give the optimal
- What other algorithms do you know that give the optimal solution?

Exact Inference?

- Can we compute the optimal solution?
- The graph of the previous problem loops
- Message passing will not give the optimal
- What other algorithms do you know that give the optimal solution?
- Let's look at branch and bound

Exact Inference?

- Can we compute the optimal solution?
- The graph of the previous problem loops
- Message passing will not give the optimal
- What other algorithms do you know that give the optimal solution?
- Let's look at branch and bound

Branch and Bound

Algorithm 1 branch and bound (BB) inference

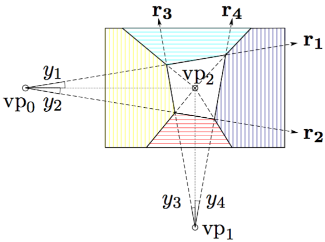
put pair $(\bar{f}(\mathcal{Y}), \mathcal{Y})$ into queue and set $\hat{\mathcal{Y}} = \mathcal{Y}$
repeat
  split $\hat{\mathcal{Y}} = \hat{\mathcal{Y}}_1 \times \hat{\mathcal{Y}}_2$ with $\hat{\mathcal{Y}}_1 \cap \hat{\mathcal{Y}}_2 = \emptyset$
  put pair $(\bar{f}(\hat{\mathcal{Y}}_1), \hat{\mathcal{Y}}_1)$ into queue
  put pair $(\bar{f}(\hat{\mathcal{Y}}_2), \hat{\mathcal{Y}}_2)$ into queue
  retrieve $\hat{\mathcal{Y}}$ having highest score
until $|\hat{\mathcal{Y}}| = 1$

We have to define:

1. A parameterization that defines **sets of hypothesis**.
2. A **scoring function** f
3. **Tight bounds** on the scoring function that can be computed very **efficiently**

Parameterization of the Problem

- Layout with 4 variables $y_i \in \mathcal{Y}$, $i \in \{1, ..., 4\}$ [Lee et al. 09]
- How do we define $\mathcal{Y}$?
- Is this problem continuous or discrete?



- We parameterize the sets by **intervals** of minimum and maximum angles

$$\{[y_1^{min}, y_1^{max}], \cdots, [y_4^{min}, y_4^{max}]\}$$

- Why intervals?
- We have defined already the scoring function. What about the bounds?

Properties of the Bounds

Derive bounds $\bar{f}$ for the original scoring function $\mathbf{w}^T \phi(\mathbf{y}, \mathbf{x})$ that satisfy:

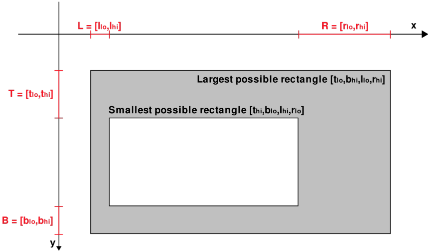
1. The bound of the interval $\hat{\mathcal{Y}}$ has to upper-bound the true cost of each hypothesis $y \in \hat{\mathcal{Y}}$,
$$\forall y \in \hat{\mathcal{Y}}, \quad \bar{f}(\hat{\mathcal{Y}}) \geq \mathbf{w}^T \phi(\mathbf{y}, \mathbf{x}).$$
2. The bound has to be exact for every single hypothesis,
$$\forall y \in \mathcal{Y}, \quad \bar{f}(y) = \mathbf{w}^T \phi(\mathbf{y}, \mathbf{x}).$$

Can we define this for our problem?

Intuitions from 2D

Let's look at the 2D case again

- We want to compute the bounding box that maximizes a scoring function
- Let's try to do this with branch and bound
- We define an interval as the max and min of the x and y axis of the rectangle



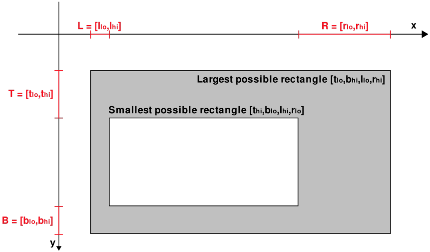
- The scoring function sums features in the rectangle defined by the BBox

$$E(y_1, \cdots, y_4) = \sum_{i \in BBox(\mathbf{y})} f_i(\mathbf{x})$$

Intuitions from 2D

Let's look at the 2D case again

- We want to compute the bounding box that maximizes a scoring function
- Let's try to do this with branch and bound
- We define an interval as the max and min of the x and y axis of the rectangle



- The scoring function sums features in the rectangle defined by the BBox

$$E(y_1, \cdots, y_4) = \sum_{i \in BBox(\mathbf{y})} f_i(\mathbf{x})$$

Branch and Bound for BBox prediction

- The scoring function sums features in the rectangle defined by the BBox

$$E(y_1, \cdots, y_4) = \sum_{i \in BBox(\mathbf{y})} f_i(\mathbf{x})$$

- Some features are positive and some are negative

Branch and Bound for BBox prediction

- The scoring function sums features in the rectangle defined by the BBox

$$E(y_1, \cdots, y_4) = \sum_{i \in BBox(\mathbf{y})} f_i(\mathbf{x})$$

- Some features are positive and some are negative
- **Trick**: Divide the space into negative and positive features

$$E(y_1, \cdots, y_4) = \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^+(\mathbf{x})}_{f^+(\mathbf{y},\mathbf{x})} + \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^-(\mathbf{x})}_{f^-(\mathbf{y},\mathbf{x})}$$

Branch and Bound for BBox prediction

- The scoring function sums features in the rectangle defined by the BBox

$$E(y_1, \cdots, y_4) = \sum_{i \in BBox(\mathbf{y})} f_i(\mathbf{x})$$

- Some features are positive and some are negative
- **Trick**: Divide the space into negative and positive features

$$E(y_1, \cdots, y_4) = \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^+(\mathbf{x})}_{f^+(\mathbf{y},\mathbf{x})} + \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^-(\mathbf{x})}_{f^-(\mathbf{y},\mathbf{x})}$$

- Bound the positive and negative independently

$$bound(E(\bar{\mathcal{Y}})) = \bar{f}^+(\bar{\mathcal{Y}}, \mathbf{x}) + \bar{f}^-(\bar{\mathcal{Y}}, \mathbf{x})$$

Branch and Bound for BBox prediction

- The scoring function sums features in the rectangle defined by the BBox

$$E(y_1, \cdots, y_4) = \sum_{i \in BBox(\mathbf{y})} f_i(\mathbf{x})$$

- Some features are positive and some are negative
- **Trick**: Divide the space into negative and positive features

$$E(y_1, \cdots, y_4) = \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^+(\mathbf{x})}_{f^+(\mathbf{y},\mathbf{x})} + \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^-(\mathbf{x})}_{f^-(\mathbf{y},\mathbf{x})}$$

- Bound the positive and negative independently

$$bound(E(\bar{\mathcal{Y}})) = \bar{f}^+(\bar{\mathcal{Y}}, \mathbf{x}) + \bar{f}^-(\bar{\mathcal{Y}}, \mathbf{x})$$

Bounding the functions

- Energy was defined as

$$E(y_1, \cdots, y_4) = \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^+(\mathbf{x})}_{f^+(\mathbf{y},\mathbf{x})} + \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^-(\mathbf{x})}_{f^-(\mathbf{y},\mathbf{x})}$$

- Bound the positive and negative independently

$$bound(E(\bar{\mathcal{Y}})) = \bar{f}^+(\bar{\mathcal{Y}}, \mathbf{x}) + \bar{f}^-(\bar{\mathcal{Y}}, \mathbf{x})$$

- These bounds are very simple? What are they?

Bounding the functions

- Energy was defined as

$$E(y_1, \cdots, y_4) = \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^+(\mathbf{x})}_{f^+(\mathbf{y},\mathbf{x})} + \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^-(\mathbf{x})}_{f^-(\mathbf{y},\mathbf{x})}$$

- Bound the positive and negative independently

$$bound(E(\bar{\mathcal{Y}})) = \bar{f}^+(\bar{\mathcal{Y}}, \mathbf{x}) + \bar{f}^-(\bar{\mathcal{Y}}, \mathbf{x})$$

- These bounds are very simple? What are they?
- How can we compute them very fast?

Bounding the functions

- Energy was defined as

$$E(y_1, \cdots, y_4) = \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^+(\mathbf{x})}_{f^+(\mathbf{y},\mathbf{x})} + \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^-(\mathbf{x})}_{f^-(\mathbf{y},\mathbf{x})}$$

- Bound the positive and negative independently

$$bound(E(\bar{\mathcal{Y}})) = \bar{f}^+(\bar{\mathcal{Y}}, \mathbf{x}) + \bar{f}^-(\bar{\mathcal{Y}}, \mathbf{x})$$

- These bounds are very simple? What are they?
- How can we compute them very fast?
- What's the complexity of computing them?

Bounding the functions

- Energy was defined as

$$E(y_1, \cdots, y_4) = \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^+(\mathbf{x})}_{f^+(\mathbf{y},\mathbf{x})} + \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^-(\mathbf{x})}_{f^-(\mathbf{y},\mathbf{x})}$$

- Bound the positive and negative independently

$$bound(E(\bar{\mathcal{Y}})) = \bar{f}^+(\bar{\mathcal{Y}}, \mathbf{x}) + \bar{f}^-(\bar{\mathcal{Y}}, \mathbf{x})$$

- These bounds are very simple? What are they?
- How can we compute them very fast?
- What's the complexity of computing them?
- How many integral images do we need?

Bounding the functions

- Energy was defined as

$$E(y_1, \cdots, y_4) = \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^+(\mathbf{x})}_{f^+(\mathbf{y},\mathbf{x})} + \underbrace{\sum_{i \in BBox(\mathbf{y})} f_i^-(\mathbf{x})}_{f^-(\mathbf{y},\mathbf{x})}$$

- Bound the positive and negative independently

$$bound(E(\bar{\mathcal{Y}})) = \bar{f}^+(\bar{\mathcal{Y}}, \mathbf{x}) + \bar{f}^-(\bar{\mathcal{Y}}, \mathbf{x})$$

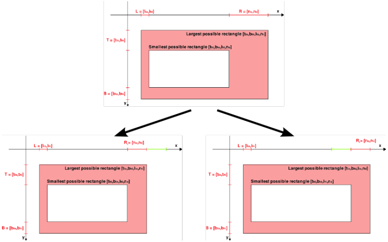
- These bounds are very simple? What are they?
- How can we compute them very fast?
- What's the complexity of computing them?
- How many integral images do we need?

Algorithm for 2D BBox [Lampert et al. 06]

Algorithm 1 Efficient Subwindow Search

Require: image x
Require: quality bounding function $\hat{f}$ (see Sect.III)
Ensure: $(t_{\text{opt}}, b_{\text{opt}}, l_{\text{opt}}, r_{\text{opt}}) = \operatorname{argmax}_{y \in \mathcal{Y}} f(y)$
  initialize P as empty priority queue
  set $[T, B, L, R] = [1, n] \times [1, n] \times [1, m] \times [1, m]$
  repeat
    split $[T, B, L, R] \to [T_1, B_1, L_1, R_1] \dot\cup [T_2, B_2, L_2, R_2]$
    push $([T_1, B_1, L_1, R_1]; \hat{f}([T_1, B_1, L_1, R_1])$ onto P
    push $([T_2, B_2, L_2, R_2]; \hat{f}([T_2, B_2, L_2, R_2])$ onto P
    retrieve top state $[T, B, L, R]$ from P
  until $[T, B, L, R]$ consists of only one rectangle
  set $(t_{\text{opt}}, b_{\text{opt}}, l_{\text{opt}}, r_{\text{opt}}) = [T, B, L, R]$

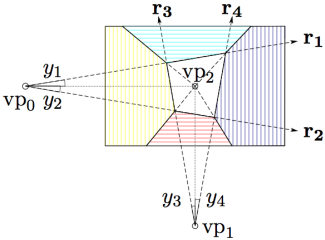
- How do we split?



- When do we terminate?

3D layout estimation

- Let's go back to our problem



- We parameterize the sets by **intervals** of minimum and maximum angles

$$\{[y_1^{min}, y_1^{max}], \cdots, [y_4^{min}, y_4^{max}]\}$$

- The scoring function sums features over the faces

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T \phi(\mathbf{y}_r, \mathbf{x}) = \sum_\alpha f_\alpha(\mathbf{y}, \mathbf{x})$$

with $\alpha = \{floor, left\_w, right\_w, ceiling, front\_w\}$

- What about the bounds?

Bounds for 3D layout

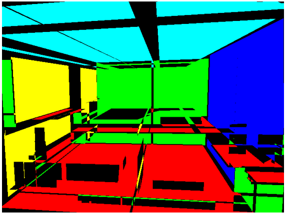
- The scoring function sums features over the faces

$$E(y_1, \cdots, y_4) = \sum_r \mathbf{w}_r^T \phi(\mathbf{y}_r, \mathbf{x}) = \sum_\alpha f_\alpha(\mathbf{y}, \mathbf{x})$$

  with $\alpha = \{floor, left\_w, right\_w, ceiling, front\_w\}$

- Let's bound each "face" α separately
- Recall where the features come from

original image



orientation map

geometric context

- Some features are positive, some are negative. Why? How do I know which ones are positive/negative?

Deriving bounds

- Inference can be then done by

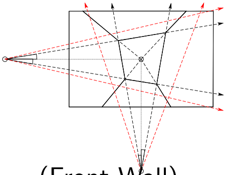
$$E(y_1, \cdots, y_4) = \sum_{\alpha} f_{\alpha}^{+}(x, y) + f_{\alpha}^{-}(x, y),$$

- We can bound each of this terms separately

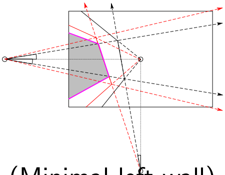
$$bound(E(\hat{\mathcal{Y}}, \mathbf{x})) = \sum_{\alpha \in \mathcal{F}} \overline{f}_{\alpha}^{+}(\hat{\mathcal{Y}}, \mathbf{x}) + \overline{f}_{\alpha}^{-}(\hat{\mathcal{Y}}, \mathbf{x})$$

- We construct bounds by computing the max positive and min negative contribution of the score within the set $\hat{\mathcal{Y}}$ for each face $\alpha \in \mathcal{F}$.

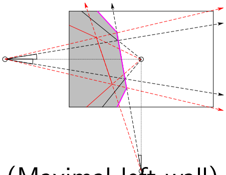
$$\overline{f}_{\textit{front-wall}}(\hat{\mathcal{Y}}) = f_{\textit{front-wall}}^{+}(x, y_{up}) + f_{\textit{front-wall}}^{-}(x, y_{low}),$$



(Front Wall)



(Minimal left wall)



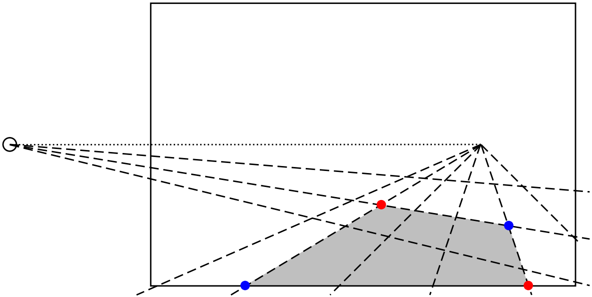
(Maximal left wall)

Efficient bounds

- How can we compute the bounds efficiently?

Efficient bounds

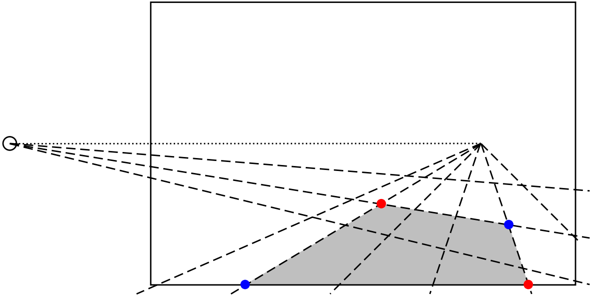
- How can we compute the bounds efficiently?



- What's the complexity?

Efficient bounds

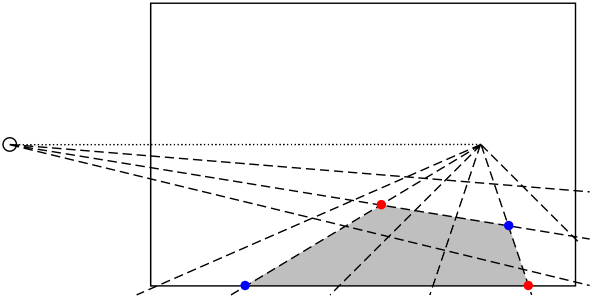
- How can we compute the bounds efficiently?



- What's the complexity?
- How many evaluations?

Efficient bounds

- How can we compute the bounds efficiently?



- What's the complexity?
- How many evaluations?

Results

[A. Schwing and R. Urtasun, ECCV12]

Table : Pixel classification error in the layout dataset of [Hedau et al. 09].

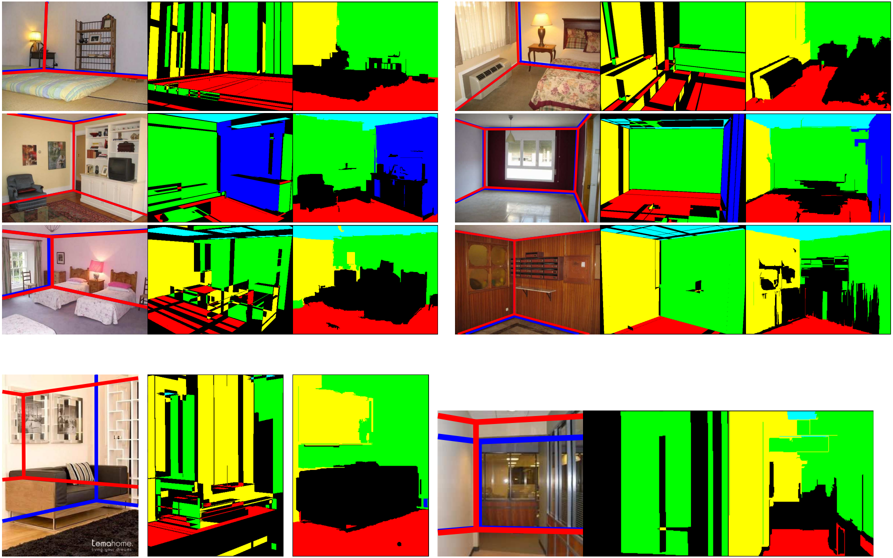
	OM	GC	OM + GC	Other	Time
[Hoiem07]	-	28.9	-	-	-
[Hedau09] (a)	-	26.5	-	-	-
[Hedau09] (b)	-	21.2	-	-	10-30 min
[Wang10]	22.2	-	-	-	
[Lee10]	24.7	22.7	18.6	-	-
[delPero11]	-	-	-	16.3	12 min
Ours	**18.6**	**15.4**	**13.6**	-	0.007s

Table : Pixel classification error in the bedroom data set [Hedau et al. 10].

	[delPero11]	[Hoiem07]	[Hedau09](a)	Ours
w/o box	29.59	23.04	22.94	**16.46**

- Takes on average 0.007s for exact solution over 50^4 possibilities !
- It's 6 orders of magnitude faster than the state-of-the-art!

Qualitative Results



Conclusion

Conclusion:

- We have studied structured prediction including learning and inference
- We have investigated how to think to solve a real-world problem

Relations to previous two talks:

- RBMs are graphical models
- Your potentials $\phi_r(y_r)$ can be "deep"

Open questions:

- Latent variable models: non-convex learning
- Learn the structure of the graph
- Go beyond log-linear models
- MAP inference: high order potentials
- Continuos Markov random fields

If you are interested in doing research at University of Toronto, talk to me!