Happy New Year!

Structured Prediction for Computer Vision

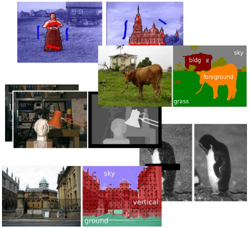
MLSS, Sydney 2015

Stephen Gould

19 February 2015

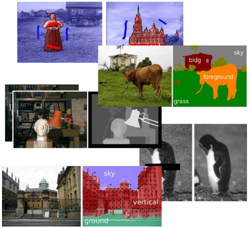
Structured Models are Pervasive in Computer Vision

Structured Models are Pervasive in Computer Vision

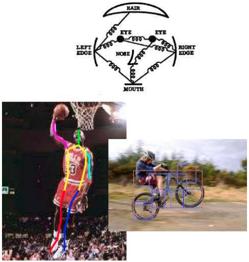


pixel labeling

Structured Models are Pervasive in Computer Vision

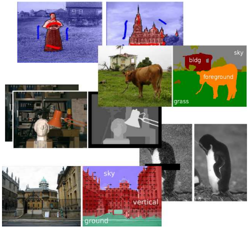


pixel labeling

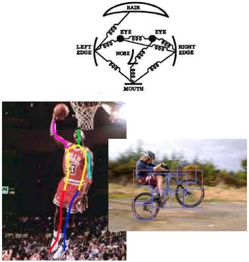


object detection, pose estimation

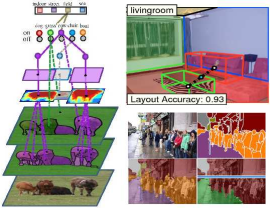
Structured Models are Pervasive in Computer Vision



pixel labeling

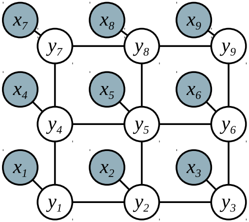


object detection, pose estimation

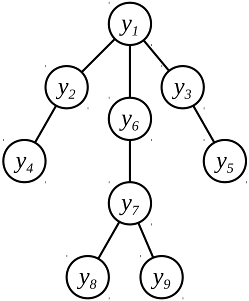


scene understanding

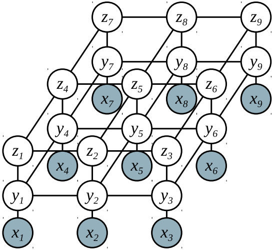
Structured Models are Pervasive in Computer Vision



pixel labeling



object detection, pose estimation



scene understanding

Demonstration: Pixel Labeling

[Agarwala et al., 2004]

- 640 × 480 image ≈ 300k pixels
- 4 possible labels per pixel
- $4^{300,000}$ label configurations
- inference in under 30 seconds (unoptimized code)

Demonstration: Pixel Labeling

[Agarwala et al., 2004]

- 640×480 image $\approx$ 300k pixels
- 4 possible labels per pixel
- $4^{300,000}$ label configurations
- inference in under 30 seconds (unoptimized code)

Conditional Markov Random Fields

- Also known as:
  - Markov Networks, Undirected Graphical Models, MRFs, Structured Prediction models
  - I make no distinction between these (in this tutorial)
- $\mathbf{X} \in \mathcal{X}$ are the observed random variables (always)
- $\mathbf{Y} = (Y_1, \ldots, Y_n) \in \mathcal{Y}$ are the output random variables
- $\mathbf{Y}_c$ are a subset of variables for clique $c \subseteq \{1, \ldots, n\}$
- Define a factored probability distribution

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$$

  where $Z(\mathbf{X}) = \sum_{\mathbf{Y} \in \mathcal{Y}} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$ is the partition function
- Main difficulty is the exponential number of configurations

Conditional Markov Random Fields

- Also known as:
  - Markov Networks, Undirected Graphical Models, MRFs, Structured Prediction models
  - I make no distinction between these (in this tutorial)
- $\mathbf{X} \in \mathcal{X}$ are the observed random variables (always)
- $\mathbf{Y} = (Y_1, \ldots, Y_n) \in \mathcal{Y}$ are the output random variables
- $\mathbf{Y}_c$ are a subset of variables for clique $c \subseteq \{1, \ldots, n\}$
- Define a factored probability distribution

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$$

  where $Z(\mathbf{X}) = \sum_{\mathbf{Y} \in \mathcal{Y}} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$ is the partition function
- Main difficulty is the exponential number of configurations

Conditional Markov Random Fields

- Also known as:
  - Markov Networks, Undirected Graphical Models, MRFs, Structured Prediction models
  - I make no distinction between these (in this tutorial)
- $\mathbf{X} \in \mathcal{X}$ are the observed random variables (always)
- $\mathbf{Y} = (Y_1, \ldots, Y_n) \in \mathcal{Y}$ are the output random variables
- $\mathbf{Y}_c$ are a subset of variables for clique $c \subseteq \{1, \ldots, n\}$
- Define a factored probability distribution

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$$

  where $Z(\mathbf{X}) = \sum_{\mathbf{Y} \in \mathcal{Y}} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$ is the partition function
- Main difficulty is the exponential number of configurations

Conditional Markov Random Fields

- Also known as:
  - Markov Networks, Undirected Graphical Models, MRFs, Structured Prediction models
  - I make no distinction between these (in this tutorial)
- $\mathbf{X} \in \mathcal{X}$ are the observed random variables (always)
- $\mathbf{Y} = (Y_1, \ldots, Y_n) \in \mathcal{Y}$ are the output random variables
- $\mathbf{Y}_c$ are a subset of variables for clique $c \subseteq \{1, \ldots, n\}$
- Define a factored probability distribution

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$$

  where $Z(\mathbf{X}) = \sum_{\mathbf{Y} \in \mathcal{Y}} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$ is the partition function
- Main difficulty is the exponential number of configurations

Conditional Markov Random Fields

- Also known as:
  - Markov Networks, Undirected Graphical Models, MRFs, Structured Prediction models
  - I make no distinction between these (in this tutorial)
- $\mathbf{X} \in \mathcal{X}$ are the observed random variables (always)
- $\mathbf{Y} = (Y_1, \ldots, Y_n) \in \mathcal{Y}$ are the output random variables
- $\mathbf{Y}_c$ are a subset of variables for clique $c \subseteq \{1, \ldots, n\}$
- Define a factored probability distribution

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$$

where $Z(\mathbf{X}) = \sum_{\mathbf{Y} \in \mathcal{Y}} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$ is the partition function

- Main difficulty is the exponential number of configurations

Conditional Markov Random Fields

- Also known as:
  - Markov Networks, Undirected Graphical Models, MRFs, Structured Prediction models
  - I make no distinction between these (in this tutorial)
- $\mathbf{X} \in \mathcal{X}$ are the observed random variables (always)
- $\mathbf{Y} = (Y_1, \ldots, Y_n) \in \mathcal{Y}$ are the output random variables
- $\mathbf{Y}_c$ are a subset of variables for clique $c \subseteq \{1, \ldots, n\}$
- Define a factored probability distribution

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$$

where $Z(\mathbf{X}) = \sum_{\mathbf{Y} \in \mathcal{Y}} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$ is the partition function

- Main difficulty is the exponential number of configurations

Conditional Markov Random Fields

- Also known as:
  - Markov Networks, Undirected Graphical Models, MRFs, Structured Prediction models
  - I make no distinction between these (in this tutorial)
- $\mathbf{X} \in \mathcal{X}$ are the observed random variables (always)
- $\mathbf{Y} = (Y_1, \ldots, Y_n) \in \mathcal{Y}$ are the output random variables
- $\mathbf{Y}_c$ are a subset of variables for clique $c \subseteq \{1, \ldots, n\}$
- Define a factored probability distribution

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$$

where $Z(\mathbf{X}) = \sum_{\mathbf{Y} \in \mathcal{Y}} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$ is the partition function

- Main difficulty is the exponential number of configurations

Conditional Markov Random Fields

- Also known as:
  - Markov Networks, Undirected Graphical Models, MRFs, Structured Prediction models
  - I make no distinction between these (in this tutorial)
- $\mathbf{X} \in \mathcal{X}$ are the observed random variables (always)
- $\mathbf{Y} = (Y_1, \ldots, Y_n) \in \mathcal{Y}$ are the output random variables
- $\mathbf{Y}_c$ are a subset of variables for clique $c \subseteq \{1, \ldots, n\}$
- Define a factored probability distribution

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$$

  where $Z(\mathbf{X}) = \sum_{\mathbf{Y} \in \mathcal{Y}} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X})$ is the partition function
- Main difficulty is the exponential number of configurations

Machine Learning Tasks

There are two main tasks that we are interested in when talking about conditional Markov random fields (machine learning, more generally):

- **Learning:** Given data (and a problem specification), how do we choose the structure and set the parameters of our model?
- **Inference:** Given our model, how do we answer queries about instances of our problem?

MAP Inference

We will mainly be interested in maximum a posteriori (MAP) inference

$$\begin{aligned}
\mathbf{y}^\star &= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y} \mid \mathbf{x}) \\
&= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X}) \\
&= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \log \left(\frac{1}{Z(\mathbf{X})} \prod_c \Psi_c(\mathbf{Y}_c; \mathbf{X}) \right) \\
&= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \sum_c \log \Psi_c(\mathbf{Y}_c; \mathbf{X}) - \log Z(\mathbf{X}) \\
&= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \sum_c \log \Psi_c(\mathbf{Y}_c; \mathbf{X})
\end{aligned}$$

Energy Functions

- Define an energy function

$$E(\mathbf{Y}; \mathbf{X}) = \sum_{c} \psi_c(\mathbf{Y}_c; \mathbf{X})$$

where $\psi_c(\cdot) = -\log \Psi_c(\cdot)$

- Then

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \exp\left\{-E(\mathbf{Y}; \mathbf{X})\right\}$$

- And

$$\underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} P(\mathbf{y} \mid \mathbf{x}) = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmin}} E(\mathbf{y}; \mathbf{x})$$

Energy Functions

- Define an energy function

$$E(\mathbf{Y}; \mathbf{X}) = \sum_c \psi_c(\mathbf{Y}_c; \mathbf{X})$$

where $\psi_c(\cdot) = -\log \Psi_c(\cdot)$

- Then

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \exp\{-E(\mathbf{Y}; \mathbf{X})\}$$

- And

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y} \mid \mathbf{x}) = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} E(\mathbf{y}; \mathbf{x})$$

Energy Functions

- Define an energy function

$$E(\mathbf{Y}; \mathbf{X}) = \sum_c \psi_c(\mathbf{Y}_c; \mathbf{X})$$

where $\psi_c(\cdot) = -\log \Psi_c(\cdot)$

- Then

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \exp\left\{-E(\mathbf{Y}; \mathbf{X})\right\}$$

- And

$$\underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} P(\mathbf{y} \mid \mathbf{x}) = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmin}} E(\mathbf{y}; \mathbf{x})$$

Energy Functions

- Define an energy function

$$E(\mathbf{Y}; \mathbf{X}) = \sum_c \psi_c(\mathbf{Y}_c; \mathbf{X})$$

where $\psi_c(\cdot) = -\log \Psi_c(\cdot)$

- Then

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \exp\left\{-E(\mathbf{Y}; \mathbf{X})\right\}$$

- And

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y} \mid \mathbf{x}) = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} E(\mathbf{y}; \mathbf{x})$$

energy minimization ‘equals’ MAP inference

Clique Potentials

- A clique potential $\psi_c(\mathbf{y}_c; \mathbf{x})$ defines a mapping from an assignment of the random variables to a real number

$$\psi_c : \mathcal{Y}_c \times \mathcal{X} \to \mathbb{R}$$

- The clique potential encodes a preference for assignments to the random variables (lower value is more preferred)
- Often parameterized as

$$\psi_c(\mathbf{y}_c; \mathbf{x}) = \mathbf{w}_c^T \phi_c(\mathbf{y}_c; \mathbf{x})$$

- In this tutorial is suffices to think of the clique potentials as big lookup tables
- We will also ignore the explicit conditioning on $\mathbf{X}$

Clique Potentials

- A clique potential $\psi_c(\mathbf{y}_c; \mathbf{x})$ defines a mapping from an assignment of the random variables to a real number

$$\psi_c : \mathcal{Y}_c \times \mathcal{X} \to \mathbb{R}$$

- The clique potential encodes a preference for assignments to the random variables (lower value is more preferred)
- Often parameterized as

$$\psi_c(\mathbf{y}_c; \mathbf{x}) = \mathbf{w}_c^T \phi_c(\mathbf{y}_c; \mathbf{x})$$

- In this tutorial is suffices to think of the clique potentials as big lookup tables
- We will also ignore the explicit conditioning on $\mathbf{X}$

Clique Potentials

- A clique potential $\psi_c(\mathbf{y}_c; \mathbf{x})$ defines a mapping from an assignment of the random variables to a real number

$$\psi_c : \mathcal{Y}_c \times \mathcal{X} \to \mathbb{R}$$

- The clique potential encodes a preference for assignments to the random variables (lower value is more preferred)
- Often parameterized as

$$\psi_c(\mathbf{y}_c; \mathbf{x}) = \mathbf{w}_c^T \phi_c(\mathbf{y}_c; \mathbf{x})$$

- In this tutorial is suffices to think of the clique potentials as big lookup tables
- We will also ignore the explicit conditioning on $\mathbf{X}$

Clique Potentials

- A clique potential $\psi_c(\mathbf{y}_c; \mathbf{x})$ defines a mapping from an assignment of the random variables to a real number

$$\psi_c : \mathcal{Y}_c \times \mathcal{X} \to \mathbb{R}$$

- The clique potential encodes a preference for assignments to the random variables (lower value is more preferred)
- Often parameterized as

$$\psi_c(\mathbf{y}_c; \mathbf{x}) = \mathbf{w}_c^T \phi_c(\mathbf{y}_c; \mathbf{x})$$

- In this tutorial is suffices to think of the clique potentials as big lookup tables
- We will also ignore the explicit conditioning on $\mathbf{X}$

Clique Potentials

- A clique potential $\psi_c(\mathbf{y}_c; \mathbf{x})$ defines a mapping from an assignment of the random variables to a real number

$$\psi_c : \mathcal{Y}_c \times \mathcal{X} \to \mathbb{R}$$

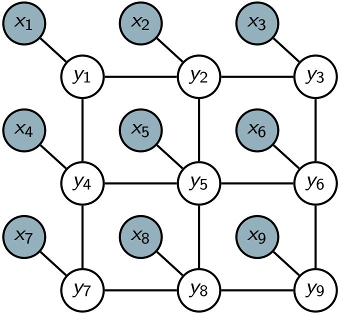
- The clique potential encodes a preference for assignments to the random variables (lower value is more preferred)
- Often parameterized as

$$\psi_c(\mathbf{y}_c; \mathbf{x}) = \mathbf{w}_c^T \phi_c(\mathbf{y}_c; \mathbf{x})$$

- In this tutorial is suffices to think of the clique potentials as big lookup tables
- We will also ignore the explicit conditioning on $\mathbf{X}$

Clique Potential Arity

$$
\begin{aligned}
E(\mathbf{y}; \mathbf{x}) &= \sum_c \psi_c(\mathbf{y}_c; \mathbf{x}) \\
&= \underbrace{\sum_{i \in \mathcal{V}} \psi_i^U(y_i; \mathbf{x})}_{\text{unary}} + \underbrace{\sum_{ij \in \mathcal{E}} \psi_{ij}^P(y_i, y_j; \mathbf{x})}_{\text{pairwise}} + \underbrace{\sum_{c \in \mathcal{C}} \psi_c^H(\mathbf{y}_c; \mathbf{x})}_{\text{higher-order}}.
\end{aligned}
$$



Example Energy Functions

Semantic Segm.

Labels: $\mathcal{L} = \{\text{sky}, \text{tree}, \text{grass}, \ldots\}$
Unary: classifier, $\psi_i^U(y_i = \ell; \mathbf{x}) = \log \mathrm{P}\left(\phi_i(\mathbf{x}) \mid \ell\right)$
Pairwise: contrast-dependent smoothness prior,

$$\psi_{ij}^P(y_i, y_j; \mathbf{x}) = \begin{cases} \lambda_0 + \lambda_1 \exp\left(-\frac{\|x_i - x_j\|^2}{2\beta}\right), & \text{if } y_i \neq y_j \\ 0, & \text{otherwise} \end{cases}$$

Example Energy Functions

Semantic Segm.

Labels: $\mathcal{L} = \{\text{sky}, \text{tree}, \text{grass}, \ldots\}$
Unary: classifier, $\psi_i^U(y_i = \ell; \mathbf{x}) = \log \mathrm{P}\left(\phi_i(\mathbf{x}) \mid \ell\right)$
Pairwise: contrast-dependent smoothness prior,

$$\psi_{ij}^P(y_i, y_j; \mathbf{x}) = \begin{cases} \lambda_0 + \lambda_1 \exp\left(-\frac{\|x_i - x_j\|^2}{2\beta}\right), & \text{if } y_i \neq y_j \\ 0, & \text{otherwise} \end{cases}$$

Object Detection

Labels: $\mathcal{L} = [0, W] \times [0, H] \times \mathbb{R}_+$
Unary: part detector/filter response, $\psi_i^U = \phi_i(\mathbf{x}) * w_i(\ell)$
Pairwise: deformation cost,

$$\psi_{ij}^P(y_i, y_j; \mathbf{x}) = \begin{cases} \lambda \|y_i - y_j\|_2^2, & \text{same scale} \\ \infty, & \text{otherwise} \end{cases}$$

Example Energy Functions

Semantic Segm.

Labels: $\mathcal{L} = \{\text{sky}, \text{tree}, \text{grass}, \ldots\}$
Unary: classifier, $\psi_i^U(y_i = \ell; \mathbf{x}) = \log \mathrm{P}\left(\phi_i(\mathbf{x}) \mid \ell\right)$
Pairwise: contrast-dependent smoothness prior,

$$\psi_{ij}^P(y_i, y_j; \mathbf{x}) = \begin{cases} \lambda_0 + \lambda_1 \exp\left(-\frac{\|x_i - x_j\|^2}{2\beta}\right), & \text{if } y_i \neq y_j \\ 0, & \text{otherwise} \end{cases}$$

Object Detection

Labels: $\mathcal{L} = [0, W] \times [0, H] \times \mathbb{R}_+$
Unary: part detector/filter response, $\psi_i^U = \phi_i(\mathbf{x}) * w_i(\ell)$
Pairwise: deformation cost,

$$\psi_{ij}^P(y_i, y_j; \mathbf{x}) = \begin{cases} \lambda \|y_i - y_j\|_2^2, & \text{same scale} \\ \infty, & \text{otherwise} \end{cases}$$

Photo Montage

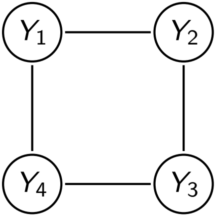
Labels: $\mathcal{L} = \{1, 2, \ldots, K\}$
Unary: none!
Pairwise: seam penalty

$$\psi_{ij}^P(y_i, y_j; \mathbf{x}) = \|\mathbf{x}_{y_i}(i) - \mathbf{x}_{y_j}(i)\| + \|\mathbf{x}_{y_i}(j) - \mathbf{x}_{y_j}(j)\|$$

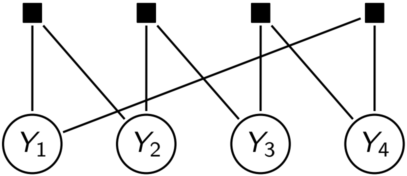
(or edge-normalized variant)

Graphical Representation

$$E(\mathbf{y}) = \psi(y_1, y_2) + \psi(y_2, y_3) + \psi(y_3, y_4) + \psi(y_4, y_1)$$



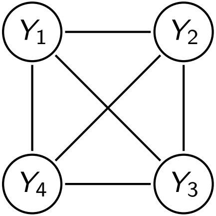
graphical model

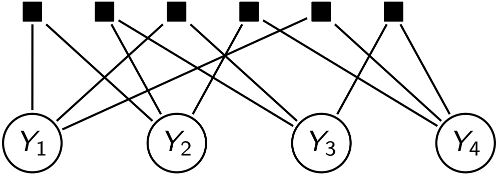


factor graph

Graphical Representation

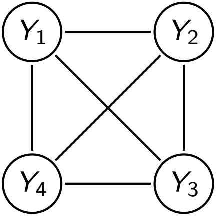
$$E(\mathbf{y}) = \sum_{i,j} \psi(y_i, y_j)$$

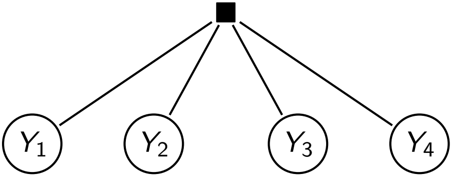




Graphical Representation

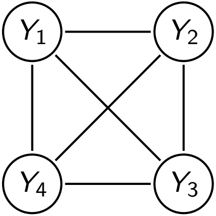
$$E(\mathbf{y}) = \psi(y_1, y_2, y_3, y_4)$$

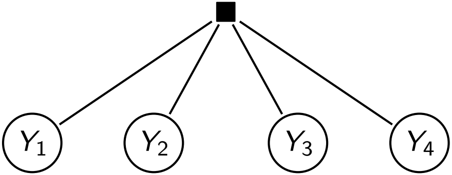




Graphical Representation

$$E(\mathbf{y}) = \psi(y_1, y_2, y_3, y_4)$$





don't worry too much about the graphical representation, look at the form of the energy function

MAP Inference / Energy Minimization

- Computing the energy minimizing assignment is NP-hard

$$\operatorname*{argmin}_{\mathbf{y}\in\mathcal{Y}} E(\mathbf{y};\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y} \mid \mathbf{x})$$

- Some structures admit tractable exact inference algorithms
  - low treewidth graphs → message passing
  - submodular potentials → graph-cuts
- Moreover, efficent approximate inference algorithms exist
  - message passing on general graphs
  - move making inference (submodular moves)
  - linear programming relaxations

MAP Inference / Energy Minimization

- Computing the energy minimizing assignment is NP-hard

$$\underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmin}}\, E(\mathbf{y};\mathbf{x}) = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\, P(\mathbf{y} \mid \mathbf{x})$$

- Some structures admit tractable exact inference algorithms
  - low treewidth graphs $\rightarrow$ message passing
  - submodular potentials $\rightarrow$ graph-cuts
- Moreover, efficent approximate inference algorithms exist
  - message passing on general graphs
  - move making inference (submodular moves)
  - linear programming relaxations

MAP Inference / Energy Minimization

- Computing the energy minimizing assignment is NP-hard

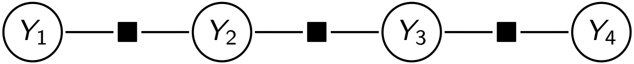
$$\operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} E(\mathbf{y}; \mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y} \mid \mathbf{x})$$

- Some structures admit tractable exact inference algorithms
  - low treewidth graphs $\rightarrow$ message passing
  - submodular potentials $\rightarrow$ graph-cuts
- Moreover, efficent approximate inference algorithms exist
  - message passing on general graphs
  - move making inference (submodular moves)
  - linear programming relaxations

exact inference

An Example: Chain Graph

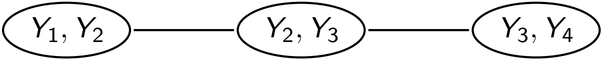
$$E(\mathbf{y}) = \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4)$$



$$\begin{aligned}
\min_{\mathbf{y}} E(\mathbf{y}) &= \min_{y_1, y_2, y_3, y_4} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4) \\
&= \min_{y_1, y_2, y_3} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \underbrace{\min_{y_4} \psi_C(y_3, y_4)}_{m_{C \to B}(y_3)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + \underbrace{\min_{y_3} \psi_B(y_2, y_3) + m_{C \to B}(y_3)}_{m_{B \to A}(y_2)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + m_{B \to A}(y_2)
\end{aligned}$$

An Example: Chain Graph

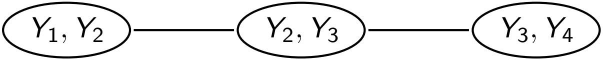
$$E(\mathbf{y}) = \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4)$$



$$\begin{aligned}
\min_{\mathbf{y}} E(\mathbf{y}) &= \min_{y_1, y_2, y_3, y_4} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4) \\
&= \min_{y_1, y_2, y_3} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \underbrace{\min_{y_4} \psi_C(y_3, y_4)}_{m_{C \to B}(y_3)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + \underbrace{\min_{y_3} \psi_B(y_2, y_3) + m_{C \to B}(y_3)}_{m_{B \to A}(y_2)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + m_{B \to A}(y_2)
\end{aligned}$$

An Example: Chain Graph

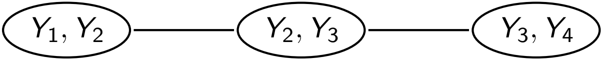
$$E(\mathbf{y}) = \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4)$$



$$\begin{aligned}
\min_{\mathbf{y}} E(\mathbf{y}) &= \min_{y_1, y_2, y_3, y_4} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4) \\
&= \min_{y_1, y_2, y_3} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \underbrace{\min_{y_4} \psi_C(y_3, y_4)}_{m_{C \to B}(y_3)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + \underbrace{\min_{y_3} \psi_B(y_2, y_3) + m_{C \to B}(y_3)}_{m_{B \to A}(y_2)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + m_{B \to A}(y_2)
\end{aligned}$$

An Example: Chain Graph

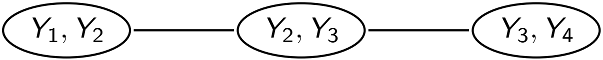
$$E(\mathbf{y}) = \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4)$$



$$\begin{aligned}
\min_{\mathbf{y}} E(\mathbf{y}) &= \min_{y_1, y_2, y_3, y_4} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4) \\
&= \min_{y_1, y_2, y_3} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \underbrace{\min_{y_4} \psi_C(y_3, y_4)}_{m_{C \to B}(y_3)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + \underbrace{\min_{y_3} \psi_B(y_2, y_3) + m_{C \to B}(y_3)}_{m_{B \to A}(y_2)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + m_{B \to A}(y_2)
\end{aligned}$$

An Example: Chain Graph

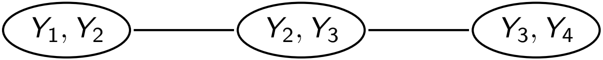
$$E(\mathbf{y}) = \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4)$$



$$\begin{aligned}
\min_{\mathbf{y}} E(\mathbf{y}) &= \min_{y_1, y_2, y_3, y_4} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4) \\
&= \min_{y_1, y_2, y_3} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \underbrace{\min_{y_4} \psi_C(y_3, y_4)}_{m_{C \to B}(y_3)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + \underbrace{\min_{y_3} \psi_B(y_2, y_3) + m_{C \to B}(y_3)}_{m_{B \to A}(y_2)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + m_{B \to A}(y_2)
\end{aligned}$$

An Example: Chain Graph

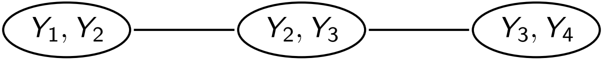
$$E(\mathbf{y}) = \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4)$$



$$\begin{aligned}
\min_{\mathbf{y}} E(\mathbf{y}) &= \min_{y_1, y_2, y_3, y_4} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4) \\
&= \min_{y_1, y_2, y_3} \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \underbrace{\min_{y_4} \psi_C(y_3, y_4)}_{m_{C \to B}(y_3)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + \underbrace{\min_{y_3} \psi_B(y_2, y_3) + m_{C \to B}(y_3)}_{m_{B \to A}(y_2)} \\
&= \min_{y_1, y_2} \psi_A(y_1, y_2) + m_{B \to A}(y_2)
\end{aligned}$$

Viterbi Decoding

$$E(\mathbf{y}) = \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4)$$



The energy minimizing assignment can be decoded as

$$y_1^\star = \operatorname*{argmin}_{y_1} \min_{y_2} \psi_A(y_1, y_2) + m_{B \to A}(y_2)$$

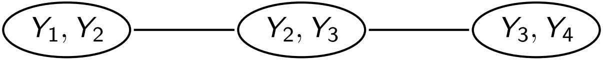
$$y_2^\star = \operatorname*{argmin}_{y_2} \psi_A(y_1^\star, y_2) + m_{B \to A}(y_2)$$

$$y_3^\star = \operatorname*{argmin}_{y_3} \psi_B(y_2^\star, y_3) + m_{C \to B}(y_3)$$

$$y_4^\star = \operatorname*{argmin}_{y_4} \psi_C(y_3^\star, y_4)$$

Viterbi Decoding

$$E(\mathbf{y}) = \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4)$$



The energy minimizing assignment can be decoded as

$$y_1^\star = \operatorname*{argmin}_{y_1} \min_{y_2} \psi_A(y_1, y_2) + m_{B \to A}(y_2)$$

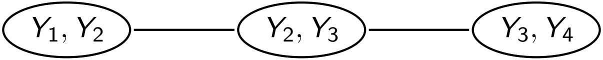
$$y_2^\star = \operatorname*{argmin}_{y_2} \psi_A(y_1^\star, y_2) + m_{B \to A}(y_2)$$

$$y_3^\star = \operatorname*{argmin}_{y_3} \psi_B(y_2^\star, y_3) + m_{C \to B}(y_3)$$

$$y_4^\star = \operatorname*{argmin}_{y_4} \psi_C(y_3^\star, y_4)$$

Viterbi Decoding

$$E(\mathbf{y}) = \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4)$$



The energy minimizing assignment can be decoded as

$$y_1^\star = \operatorname*{argmin}_{y_1} \min_{y_2} \psi_A(y_1, y_2) + m_{B \to A}(y_2)$$

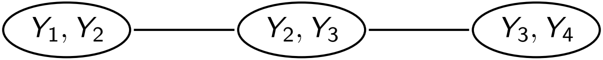
$$y_2^\star = \operatorname*{argmin}_{y_2} \psi_A(y_1^\star, y_2) + m_{B \to A}(y_2)$$

$$y_3^\star = \operatorname*{argmin}_{y_3} \psi_B(y_2^\star, y_3) + m_{C \to B}(y_3)$$

$$y_4^\star = \operatorname*{argmin}_{y_4} \psi_C(y_3^\star, y_4)$$

Viterbi Decoding

$$E(\mathbf{y}) = \psi_A(y_1, y_2) + \psi_B(y_2, y_3) + \psi_C(y_3, y_4)$$



The energy minimizing assignment can be decoded as

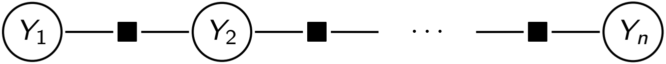
$$y_1^\star = \operatorname*{argmin}_{y_1} \min_{y_2} \psi_A(y_1, y_2) + m_{B \to A}(y_2)$$

$$y_2^\star = \operatorname*{argmin}_{y_2} \psi_A(y_1^\star, y_2) + m_{B \to A}(y_2)$$

$$y_3^\star = \operatorname*{argmin}_{y_3} \psi_B(y_2^\star, y_3) + m_{C \to B}(y_3)$$

$$y_4^\star = \operatorname*{argmin}_{y_4} \psi_C(y_3^\star, y_4)$$

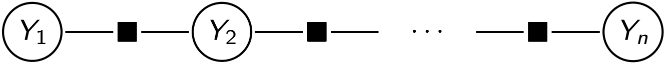
What did this cost us?



For a chain of length n with L labels per variable:

- Brute force enumeration would cost $|\mathcal{Y}| = L^n$
- Viterbi decoding (message passing) costs $O(nL^2)$
- The operation $\min \psi(\cdot, \cdot) + m(\cdot)$ can be sped up for potentials with certain structure (e.g., so called convex priors)

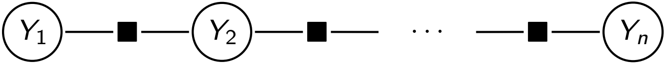
What did this cost us?



For a chain of length n with L labels per variable:

- Brute force enumeration would cost $|\mathcal{Y}| = L^n$
- Viterbi decoding (message passing) costs $O(nL^2)$
- The operation $\min \psi(\cdot, \cdot) + m(\cdot)$ can be sped up for potentials with certain structure (e.g., so called convex priors)

What did this cost us?



For a chain of length n with L labels per variable:

- Brute force enumeration would cost $|\mathcal{Y}| = L^n$
- Viterbi decoding (message passing) costs $O(nL^2)$
- The operation $\min \psi(\cdot,\cdot) + m(\cdot)$ can be sped up for potentials with certain structure (e.g., so called convex priors)

Factor Operations

The preceeding inference algorithm was based on two important operations defined on factors (clique potentials).

- **Factor addition** creates an outut whose scope is the union of the scope of its inputs. Each element of the output is the sum of the corresponding (projected) elements of the inputs.

$$\mathbf{Y}_c = \mathbf{Y}_a \cup \mathbf{Y}_b \quad : \quad \psi_c(\mathbf{y}_c) = \psi_a([\mathbf{y}_c]_a) + \psi_b([\mathbf{y}_c]_b)$$

- **Factor minimization** creates an output where one or more input variables are removed. Each element of the output is the result of minimizing over values of the removed variables.

$$\mathbf{Y}_c \subset \mathbf{Y}_a \quad : \quad \psi_c(\mathbf{y}_c) = \min_{\mathbf{y}_{a \setminus c} \in \mathcal{Y}_a \setminus \mathcal{Y}_c} \psi_a(\{\mathbf{y}_{a \setminus c}, \mathbf{y}_c\})$$

Factor Operations

The preceeding inference algorithm was based on two important operations defined on factors (clique potentials).

- **Factor addition** creates an outut whose scope is the union of the scope of its inputs. Each element of the output is the sum of the corresponding (projected) elements of the inputs.

$$\mathbf{Y}_c = \mathbf{Y}_a \cup \mathbf{Y}_b \quad : \quad \psi_c(\mathbf{y}_c) = \psi_a([\mathbf{y}_c]_a) + \psi_b([\mathbf{y}_c]_b)$$

- **Factor minimization** creates an output where one or more input variables are removed. Each element of the output is the result of minimizing over values of the removed variables.

$$\mathbf{Y}_c \subset \mathbf{Y}_a \quad : \quad \psi_c(\mathbf{y}_c) = \min_{\mathbf{y}_{a \setminus c} \in \mathcal{Y}_a \setminus \mathcal{Y}_c} \psi_a(\{\mathbf{y}_{a \setminus c}, \mathbf{y}_c\})$$

Factor Operations Worked Example

y_1	y_2	ψ_a
0	0	1
0	1	4
1	0	7
1	1	2

plus

y_2	y_3	ψ_b
0	0	5
0	1	-3
1	0	1
1	1	8

=

y_1	y_2	y_3	$\psi_c = \psi_a + \psi_b$
0	0	0	$1 + 5 = 6$
0	0	1	1 - 3 = -2
0	1	0	$4 + 1 = 5$
0	1	1	$4 + 8 = 12$
1	0	0	$7 + 5 = 12$
1	0	1	7 - 3 = 4
1	1	0	$2 + 1 = 3$
1	1	1	$2 + 8 = 10$

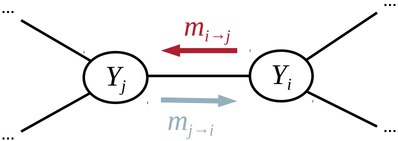
Clique Trees

A *clique tree* (or *tree decomposition*) for an energy function $E(\mathbf{y})$ is a pair $(\mathcal{C}, \mathcal{T})$, where $\mathcal{C} = \{C_1, \ldots, C_M\}$ is a family of subsets of $\{1, \ldots, n\}$ and $\mathcal{T}$ is a tree with nodes C_m satisfying:

- **Family Preserving:** if $\mathbf{Y}_c$ is a clique in $E(\mathbf{y})$ then there must exist a subset $C_m \in \mathcal{C}$ with $\mathbf{Y}_c \in C_m$;
- **Running Intersection Property:** if C_m and $C_{m'}$ both contain Y_i then there is a unique path through $\mathcal{T}$ between C_m and $C_{m'}$ such that Y_i is in every node along the path.

These properties are sufficient to ensure the message passing correctness of message passing.

Min-Sum Message Passing on Clique Trees



- messages sent in reverse then forward topological ordering
- message from clique i to clique j calculated as

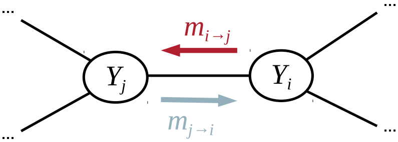
$$m_{i\to j}(\mathbf{Y}_j \cap \mathbf{Y}_i) = \min_{\mathbf{Y}_i \setminus \mathbf{Y}_j} \left(\psi_i(\mathbf{Y}_i) + \sum_{k \in \mathcal{N}(i)\setminus\{j\}} m_{k\to i}(\mathbf{Y}_i \cap \mathbf{Y}_k) \right)$$

- energy minimizing assignment decoded as

$$\mathbf{y}_i^\star = \operatorname*{argmin}_{\mathbf{Y}_i} \left(\overbrace{\psi_i(\mathbf{Y}_i) + \sum_{k \in \mathcal{N}(i)} m_{k\to i}(\mathbf{Y}_i \cap \mathbf{Y}_k)}^{\text{min marginal}} \right)$$

- ties must be decoded consistently

Min-Sum Message Passing on Clique Trees



- messages sent in reverse then forward topological ordering
- message from clique i to clique j calculated as

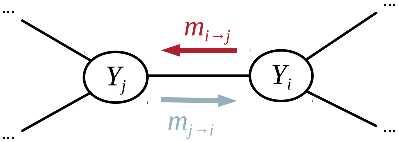
$$m_{i \to j}(\mathbf{Y}_j \cap \mathbf{Y}_i) = \min_{\mathbf{Y}_i \setminus \mathbf{Y}_j} \left(\psi_i(\mathbf{Y}_i) + \sum_{k \in \mathcal{N}(i) \setminus \{j\}} m_{k \to i}(\mathbf{Y}_i \cap \mathbf{Y}_k) \right)$$

- energy minimizing assignment decoded as

$$\mathbf{y}_i^\star = \operatorname*{argmin}_{\mathbf{Y}_i} \left(\overbrace{\psi_i(\mathbf{Y}_i) + \sum_{k \in \mathcal{N}(i)} m_{k \to i}(\mathbf{Y}_i \cap \mathbf{Y}_k)}^{\text{min marginal}} \right)$$

- ties must be decoded consistently

Min-Sum Message Passing on Clique Trees



- messages sent in reverse then forward topological ordering
- message from clique i to clique j calculated as

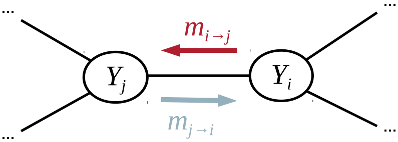
$$m_{i \to j}(\mathbf{Y}_j \cap \mathbf{Y}_i) = \min_{\mathbf{Y}_i \setminus \mathbf{Y}_j} \left(\psi_i(\mathbf{Y}_i) + \sum_{k \in \mathcal{N}(i) \setminus \{j\}} m_{k \to i}(\mathbf{Y}_i \cap \mathbf{Y}_k) \right)$$

- energy minimizing assignment decoded as

$$\mathbf{y}_i^\star = \operatorname*{argmin}_{\mathbf{Y}_i} \left(\overbrace{\psi_i(\mathbf{Y}_i) + \sum_{k \in \mathcal{N}(i)} m_{k \to i}(\mathbf{Y}_i \cap \mathbf{Y}_k)}^{\text{min marginal}} \right)$$

- ties must be decoded consistently

Min-Sum Message Passing on Clique Trees



- messages sent in reverse then forward topological ordering
- message from clique i to clique j calculated as

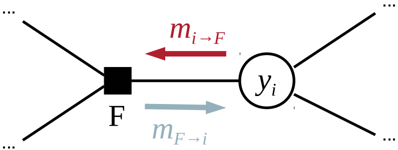
$$m_{i \to j}(\mathbf{Y}_j \cap \mathbf{Y}_i) = \min_{\mathbf{Y}_i \setminus \mathbf{Y}_j} \left(\psi_i(\mathbf{Y}_i) + \sum_{k \in \mathcal{N}(i) \setminus \{j\}} m_{k \to i}(\mathbf{Y}_i \cap \mathbf{Y}_k) \right)$$

- energy minimizing assignment decoded as

$$\mathbf{y}_i^\star = \operatorname*{argmin}_{\mathbf{Y}_i} \left(\overbrace{\psi_i(\mathbf{Y}_i) + \sum_{k \in \mathcal{N}(i)} m_{k \to i}(\mathbf{Y}_i \cap \mathbf{Y}_k)}^{\text{min marginal}} \right)$$

- ties must be decoded consistently

Min-Sum Message Passing on Factor Graphs (Trees)



- messages from variables to factors

$$m_{i \to F}(y_i) = \sum_{G \in \mathcal{N}(i) \setminus \{F\}} m_{G \to i}(y_i)$$

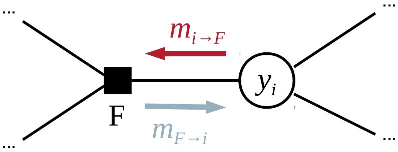
- messages from factors to variables

$$m_{F \to i}(y_i) = \min_{\mathbf{y}'_F, y'_i = y_i} \left(\psi_F(\mathbf{y}'_F) + \sum_{j \in \mathcal{N}(F) \setminus \{i\}} m_{j \to F}(y'_j) \right)$$

- energy minimizing assignment decoded as

$$y_i^\star = \operatorname{argmin}_{y_i} \sum_{F \in \mathcal{N}(i)} m_{F \to i}(y_i)$$

Min-Sum Message Passing on Factor Graphs (Trees)



- messages from variables to factors

$$m_{i\to F}(y_i) = \sum_{G\in\mathcal{N}(i)\setminus\{F\}} m_{G\to i}(y_i)$$

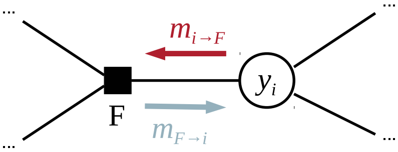
- messages from factors to variables

$$m_{F\to i}(y_i) = \min_{\mathbf{y}'_F, y'_i = y_i} \left(\psi_F(\mathbf{y}'_F) + \sum_{j\in\mathcal{N}(F)\setminus\{i\}} m_{j\to F}(y'_j) \right)$$

- energy minimizing assignment decoded as

$$y_i^\star = \operatorname*{argmin}_{y_i} \sum_{F\in\mathcal{N}(i)} m_{F\to i}(y_i)$$

Min-Sum Message Passing on Factor Graphs (Trees)



- messages from variables to factors

$$m_{i\to F}(y_i) = \sum_{G \in \mathcal{N}(i) \setminus \{F\}} m_{G \to i}(y_i)$$

- messages from factors to variables

$$m_{F\to i}(y_i) = \min_{\mathbf{y}'_F, y'_i = y_i} \left(\psi_F(\mathbf{y}'_F) + \sum_{j \in \mathcal{N}(F) \setminus \{i\}} m_{j \to F}(y'_j) \right)$$

- energy minimizing assignment decoded as

$$y_i^\star = \operatorname*{argmin}_{y_i} \sum_{F \in \mathcal{N}(i)} m_{F \to i}(y_i)$$

Message Passing on General Graphs

- Message passing can be generalized to graphs with loops
- If the treewidth is small we can still perform exact inference
  - **junction tree algorithm**: triangulate the graph and run message passing on the resulting tree
- Otherwise run message passing anyway
  - **loopy belief propagtaion**
  - different message schedules (synchronous/asynchronous, static/dynamic)
  - no convergence or approximation guarantees, in general

Message Passing on General Graphs

- Message passing can be generalized to graphs with loops
- If the treewidth is small we can still perform exact inference
  - **junction tree algorithm**: triangulate the graph and run message passing on the resulting tree
- Otherwise run message passing anyway
  - **loopy belief propagtaion**
  - different message schedules (synchronous/asynchronous, static/dynamic)
  - no convergence or approximation guarantees, in general

Message Passing on General Graphs

- Message passing can be generalized to graphs with loops
- If the treewidth is small we can still perform exact inference
  - **junction tree algorithm**: triangulate the graph and run message passing on the resulting tree
- Otherwise run message passing anyway
  - **loopy belief propagtaion**
  - different message schedules (synchronous/asynchronous, static/dynamic)
  - no convergence or approximation guarantees, in general

Message Passing on General Graphs

- Message passing can be generalized to graphs with loops
- If the treewidth is small we can still perform exact inference
    - **junction tree algorithm**: triangulate the graph and run message passing on the resulting tree
- Otherwise run message passing anyway
    - **loopy belief propagtaion**
    - different message schedules (synchronous/asynchronous, static/dynamic)
    - no convergence or approximation guarantees, in general

Message Passing on General Graphs

- Message passing can be generalized to graphs with loops
- If the treewidth is small we can still perform exact inference
  - **junction tree algorithm**: triangulate the graph and run message passing on the resulting tree
- Otherwise run message passing anyway
  - **loopy belief propagtaion**
  - different message schedules (synchronous/asynchronous, static/dynamic)
  - no convergence or approximation guarantees, in general

Message Passing on General Graphs

- Message passing can be generalized to graphs with loops
- If the treewidth is small we can still perform exact inference
  - **junction tree algorithm**: triangulate the graph and run message passing on the resulting tree
- Otherwise run message passing anyway
  - **loopy belief propagtaion**
  - different message schedules (synchronous/asynchronous, static/dynamic)
  - no convergence or approximation guarantees, in general

Message Passing on General Graphs

- Message passing can be generalized to graphs with loops
- If the treewidth is small we can still perform exact inference
  - **junction tree algorithm**: triangulate the graph and run message passing on the resulting tree
- Otherwise run message passing anyway
  - **loopy belief propagtaion**
  - different message schedules (synchronous/asynchronous, static/dynamic)
  - no convergence or approximation guarantees, in general

graph-cut based methods

Binary MRF Example

Consider the following energy function for two binary random variables, y_1 and y_2.

0	5
1	2

0	1
1	3

	0	1
0	0	3
1	4	0

$$E(y_1, y_2) = \psi_1(y_1) + \psi_2(y_2) + \psi_{12}(y_1, y_2)$$
$$= \underbrace{5\bar{y}_1 + 2y_1}_{\psi_1} + \underbrace{\bar{y}_2 + 3y_2}_{\psi_2} + \underbrace{3\bar{y}_1 y_2 + 4y_1\bar{y}_2}_{\psi_{12}}$$

where $\bar{y}_1 = 1 - y_1$ and $\bar{y}_2 = 1 - y_2$.

Binary MRF Example

Consider the following energy function for two binary random variables, y_1 and y_2.

0	5
1	2

0	1
1	3

	0	1
0	0	3
1	4	0

$$\begin{aligned} E(y_1, y_2) &= \psi_1(y_1) + \psi_2(y_2) + \psi_{12}(y_1, y_2) \\ &= \underbrace{5\bar{y}_1 + 2y_1}_{\psi_1} \\ &\quad + \underbrace{\bar{y}_2 + 3y_2}_{\psi_2} \\ &\quad + \underbrace{3\bar{y}_1 y_2 + 4y_1\bar{y}_2}_{\psi_{12}} \end{aligned}$$

where $\bar{y}_1 = 1 - y_1$ and $\bar{y}_2 = 1 - y_2$.

Binary MRF Example

Consider the following energy function for two binary random variables, y_1 and y_2.

0	5
1	2

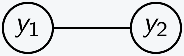
0	1
1	3

	0	1
0	0	3
1	4	0

$$
\begin{aligned}
E(y_1, y_2) &= \psi_1(y_1) + \psi_2(y_2) + \psi_{12}(y_1, y_2) \\
&= \underbrace{5\bar{y}_1 + 2y_1}_{\psi_1} \\
&\quad + \underbrace{\bar{y}_2 + 3y_2}_{\psi_2} \\
&\quad + \underbrace{3\bar{y}_1 y_2 + 4y_1\bar{y}_2}_{\psi_{12}}
\end{aligned}
$$

where $\bar{y}_1 = 1 - y_1$ and $\bar{y}_2 = 1 - y_2$.

Graphical Model



Probability Table

y_1	y_2	E	P
0	0	6	0.244
0	1	11	0.002
1	0	7	0.090
1	1	5	0.664

Pseudo-boolean Functions [Boros and Hammer, 2001]

Pseudo-boolean Function

A mapping $f : \{0,1\}^n \rightarrow \mathbb{R}$ is called a *pseudo-Boolean function*.

- Pseudo-boolean functions can be uniquely represented as *multi-linear polynomials*, e.g., $f(y_1, y_2) = 6 + y_1 + 5y_2 - 7y_1y_2$.
- Pseudo-boolean functions can also be represented in *posiform*, e.g., $f(y_1, y_2) = 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1 y_2 + 4y_1\bar{y}_2$. This representation is not unique.
- **A binary pairwise Markov random field (MRF) is just a quadratic pseudo-Boolean function.**

Pseudo-boolean Functions [Boros and Hammer, 2001]

Pseudo-boolean Function

A mapping $f : \{0,1\}^n \rightarrow \mathbb{R}$ is called a *pseudo-Boolean function*.

- Pseudo-boolean functions can be uniquely represented as *multi-linear polynomials*, e.g., $f(y_1, y_2) = 6 + y_1 + 5y_2 - 7y_1 y_2$.
- Pseudo-boolean functions can also be represented in *posiform*, e.g., $f(y_1, y_2) = 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1 y_2 + 4y_1 \bar{y}_2$. This representation is not unique.
- **A binary pairwise Markov random field (MRF) is just a quadratic pseudo-Boolean function.**

Pseudo-boolean Functions [Boros and Hammer, 2001]

Pseudo-boolean Function

A mapping $f : \{0,1\}^n \to \mathbb{R}$ is called a *pseudo-Boolean function*.

- Pseudo-boolean functions can be uniquely represented as *multi-linear polynomials*, e.g., $f(y_1, y_2) = 6 + y_1 + 5y_2 - 7y_1 y_2$.
- Pseudo-boolean functions can also be represented in *posiform*, e.g., $f(y_1, y_2) = 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1 y_2 + 4y_1 \bar{y}_2$. This representation is not unique.
- A binary pairwise Markov random field (MRF) is just a quadratic pseudo-Boolean function.

Pseudo-boolean Functions [Boros and Hammer, 2001]

Pseudo-boolean Function

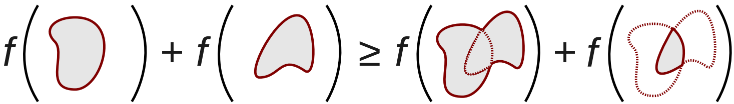
A mapping $f : \{0,1\}^n \to \mathbb{R}$ is called a *pseudo-Boolean function*.

- Pseudo-boolean functions can be uniquely represented as *multi-linear polynomials*, e.g., $f(y_1, y_2) = 6 + y_1 + 5y_2 - 7y_1y_2$.
- Pseudo-boolean functions can also be represented in *posiform*, e.g., $f(y_1, y_2) = 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1y_2 + 4y_1\bar{y}_2$. This representation is not unique.
- **A binary pairwise Markov random field (MRF) is just a quadratic pseudo-Boolean function.**

Submodular Functions

Submodularity

Let $\mathcal{V}$ be a set. A set function $f : 2^{\mathcal{V}} \to \mathbb{R}$ is called *submodular* if $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$ for all subsets $X, Y \subseteq \mathcal{V}$.



Submodular Binary Pairwise MRFs

Submodularity

A pseudo-Boolean function $f : \{0, 1\}^n \to \mathbb{R}$ is called *submodular* if $f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y})$ for all vectors $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$.

Submodularity checks for pairwise binary MRFs:

- polynomial form (of pseudo-boolean function) has negative coefficients on all bi-linear terms;
- posiform has pairwise terms of the form $u\bar{v}$;
- all pairwise potentials satisfy

$$\psi^P_{ij}(0, 1) + \psi^P_{ij}(1, 0) \geq \psi^P_{ij}(1, 1) + \psi^P_{ij}(0, 0)$$

Submodular Binary Pairwise MRFs

Submodularity

A pseudo-Boolean function $f : \{0, 1\}^n \to \mathbb{R}$ is called *submodular* if $f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y})$ for all vectors $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$.

Submodularity checks for pairwise binary MRFs:

- polynomial form (of pseudo-boolean function) has negative coefficients on all bi-linear terms;
- posiform has pairwise terms of the form $u\bar{v}$;
- all pairwise potentials satisfy

$$\psi_{ij}^P(0,1) + \psi_{ij}^P(1,0) \geq \psi_{ij}^P(1,1) + \psi_{ij}^P(0,0)$$

Submodularity of Binary Pairwise Terms

To see the equivalence of the last two conditions consider the following pairwise potential

	0	1
0	α	β
1	γ	δ

$$\alpha + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline \gamma-\alpha & \gamma-\alpha \\ \hline\end{array} + \begin{array}{|c|c|}\hline 0 & \delta-\gamma \\ \hline 0 & \delta-\gamma \\ \hline\end{array} + \begin{array}{|c|c|}\hline 0 & \beta+\gamma-\alpha-\delta \\ \hline 0 & 0 \\ \hline\end{array}$$

$$E(y_1, y_2) = \alpha + (\gamma - \alpha)y_1 + (\delta - \gamma)y_2 + (\beta + \gamma - \alpha - \delta)\bar{y}_1 y_2$$

[Kolmogorov and Zabih, 2004]

Submodularity of Binary Pairwise Terms

To see the equivalence of the last two conditions consider the following pairwise potential

	0	1
0	α	β
1	γ	δ

$$\alpha + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline \gamma - \alpha & \gamma - \alpha \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & \delta - \gamma \\ \hline 0 & \delta - \gamma \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & \beta + \gamma - \alpha - \delta \\ \hline 0 & 0 \\ \hline \end{array}$$

$$E(y_1, y_2) = \alpha + (\gamma - \alpha) y_1 + (\delta - \gamma) y_2 + (\beta + \gamma - \alpha - \delta) \bar{y}_1 y_2$$

[Kolmogorov and Zabih, 2004]

Submodularity of Binary Pairwise Terms

To see the equivalence of the last two conditions consider the following pairwise potential

	0	1
0	α	β
1	γ	δ

$$\alpha + \begin{array}{|c|c|}\hline 0 & 0 \\ \hline \gamma - \alpha & \gamma - \alpha \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & \delta - \gamma \\ \hline 0 & \delta - \gamma \\ \hline \end{array} + \begin{array}{|c|c|}\hline 0 & \beta + \gamma - \alpha - \delta \\ \hline 0 & 0 \\ \hline \end{array}$$

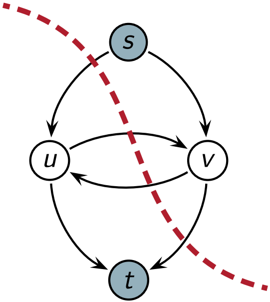
$$E(y_1, y_2) = \alpha + (\gamma - \alpha) y_1 + (\delta - \gamma) y_2 + (\beta + \gamma - \alpha - \delta) \bar{y}_1 y_2$$

[Kolmogorov and Zabih, 2004]

Minimum-cut Problem

Graph Cut

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a capacitated digraph with two distinguished vertices s and t. An st-cut is a partitioning of $\mathcal{V}$ into two disjoint sets $\mathcal{S}$ and $\mathcal{T}$ such that $s \in \mathcal{S}$ and $t \in \mathcal{T}$. The cost of the cut is the sum of edge capacities for all edges going from $\mathcal{S}$ to $\mathcal{T}$.



Quadratic Pseudo-boolean Optimization

Main idea:

- construct a graph such that every *st*-cut corresponds to a joint assignment to the variables $\mathbf{y}$
- the cost of the cut should be equal to the energy of the assignment, $E(\mathbf{y}; \mathbf{x})$.*
- the minimum-cut then corresponds to the the minimum energy assignment, $\mathbf{y}^\star = \operatorname{argmin}_{\mathbf{y}} E(\mathbf{y}; \mathbf{x})$.

*Requires non-negative edge weights.

Quadratic Pseudo-boolean Optimization

Main idea:

- construct a graph such that every *st*-cut corresponds to a joint assignment to the variables $\mathbf{y}$
- the cost of the cut should be equal to the energy of the assignment, $E(\mathbf{y}; \mathbf{x})$.*
- the minimum-cut then corresponds to the the minimum energy assignment, $\mathbf{y}^\star = \operatorname{argmin}_{\mathbf{y}} E(\mathbf{y}; \mathbf{x})$.

*Requires non-negative edge weights.

Quadratic Pseudo-boolean Optimization

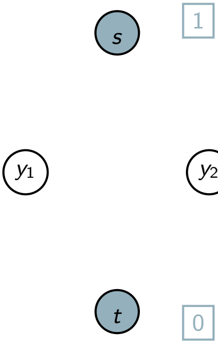
Main idea:

- construct a graph such that every *st*-cut corresponds to a joint assignment to the variables $\mathbf{y}$
- the cost of the cut should be equal to the energy of the assignment, $E(\mathbf{y}; \mathbf{x})$.*
- the minimum-cut then corresponds to the the minimum energy assignment, $\mathbf{y}^\star = \text{argmin}_{\mathbf{y}} E(\mathbf{y}; \mathbf{x})$.

*Requires non-negative edge weights.

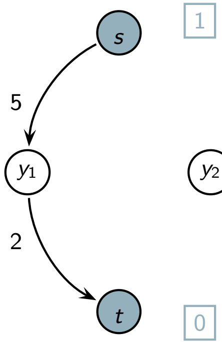
Example *st*-Graph Construction for Binary MRF

$$E(y_1, y_2) = \psi_1(y_1) + \psi_2(y_2) + \psi_{ij}(y_1, y_2)$$
$$= 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1 y_2 + 4y_1\bar{y}_2$$



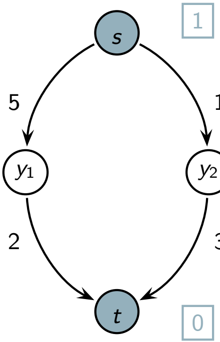
Example *st*-Graph Construction for Binary MRF

$$E(y_1, y_2) = \psi_1(y_1) + \psi_2(y_2) + \psi_{ij}(y_1, y_2)$$
$$= 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1 y_2 + 4y_1\bar{y}_2$$



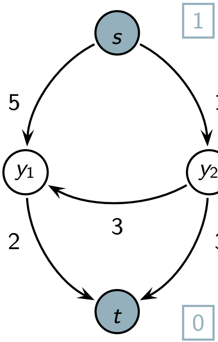
Example *st*-Graph Construction for Binary MRF

$$E(y_1, y_2) = \psi_1(y_1) + \psi_2(y_2) + \psi_{ij}(y_1, y_2)$$
$$= 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1 y_2 + 4y_1\bar{y}_2$$



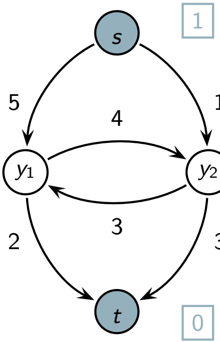
Example *st*-Graph Construction for Binary MRF

$$
\begin{aligned}
E(y_1, y_2) &= \psi_1(y_1) + \psi_2(y_2) + \psi_{ij}(y_1, y_2) \\
&= 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1 y_2 + 4y_1\bar{y}_2
\end{aligned}
$$



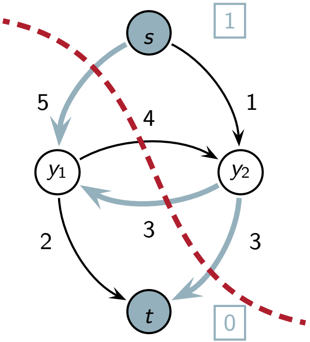
Example *st*-Graph Construction for Binary MRF

$$
\begin{aligned}
E(y_1, y_2) &= \psi_1(y_1) + \psi_2(y_2) + \psi_{ij}(y_1, y_2) \\
&= 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1 y_2 + 4y_1\bar{y}_2
\end{aligned}
$$



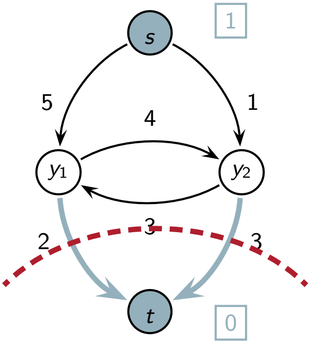
An Example *st*-Cut

$$\begin{aligned} E(0,1) &= \psi_1(0) + \psi_2(1) + \psi_{ij}(0,1) \\ &= 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1 y_2 + 4y_1\bar{y}_2 \end{aligned}$$



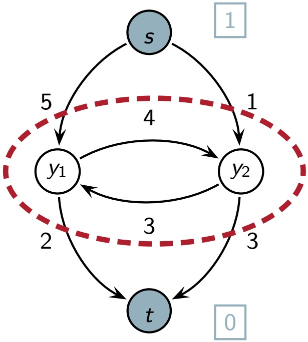
Another *st*-Cut

$$E(1,1) = \psi_1(1) + \psi_2(1) + \psi_{ij}(1,1)$$
$$= 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1 y_2 + 4 y_1 \bar{y}_2$$



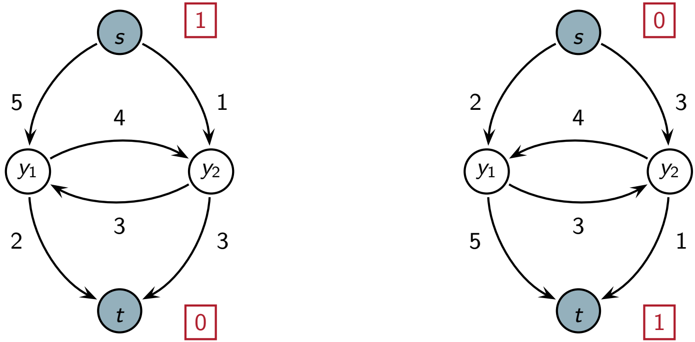
Invalid *st*-Cut

This is not a valid cut, since it does not correspond to a partitioning of the nodes into two sets—one containing s and one containing t.



Alternative *st*-Graph Construction

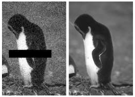
Sometimes you will see the roles of s and t switched.

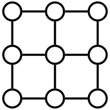


These graphs represent the same energy function.

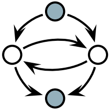
Big Picture: Where are we?

We can now formulate inference in a submodular binary pairwise MRF as a minimum-cut problem.





$\{0,1\}^n \to \mathbb{R}$

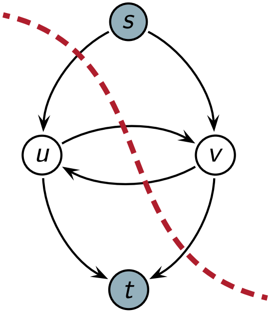


How do we solve the minimum-cut problem?

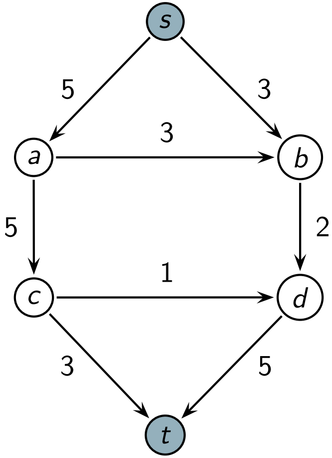
Max-flow/Min-cut Theorem

Max-flow/Min-cut Theorem [Fulkerson, 1956]

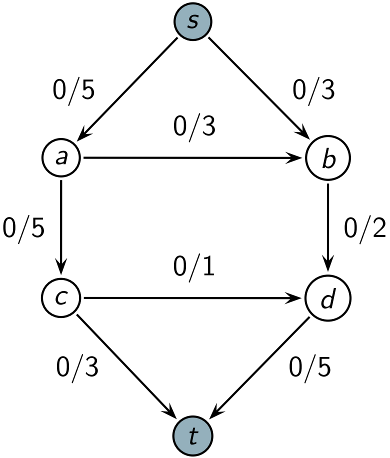
The maximum flow f from vertex s to vertex t is equal to the minimum cost st-cut.



Maximum Flow Example



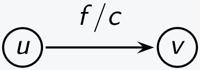
Maximum Flow Example (Augmenting Path)



flow

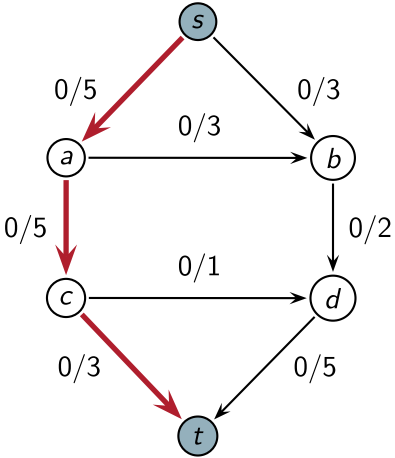
0

notation



edge with capacity c, and current flow f.

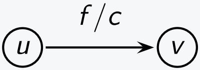
Maximum Flow Example (Augmenting Path)



flow

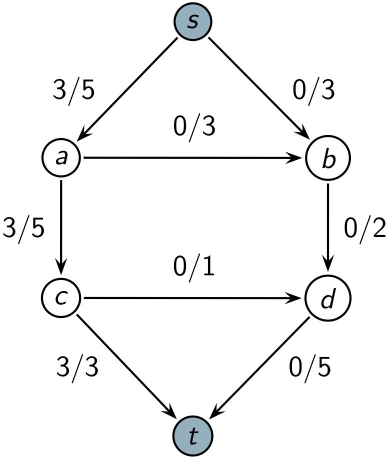
0

notation



edge with capacity c, and current flow f.

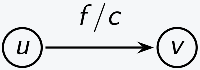
Maximum Flow Example (Augmenting Path)



flow

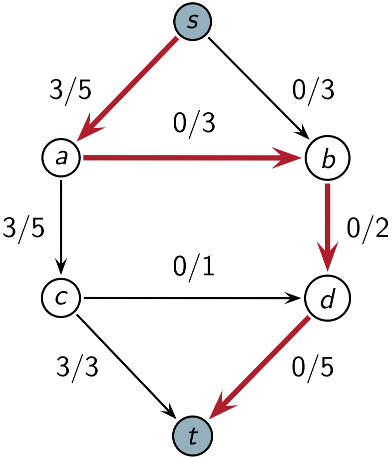
3

notation



edge with capacity c, and current flow f.

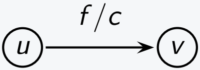
Maximum Flow Example (Augmenting Path)



flow

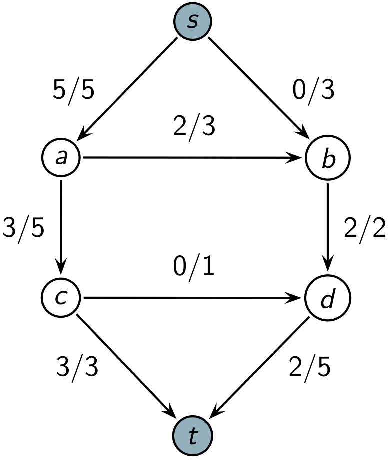
3

notation



edge with capacity c, and current flow f.

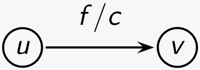
Maximum Flow Example (Augmenting Path)



flow

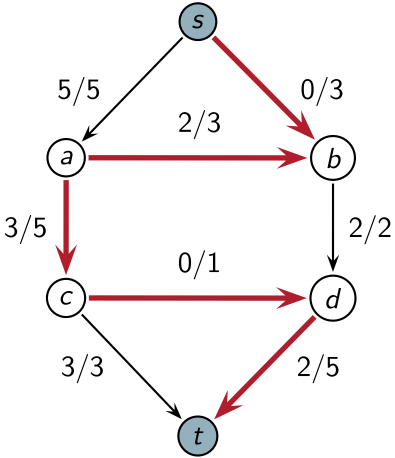
5

notation



edge with capacity c, and current flow f.

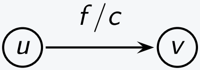
Maximum Flow Example (Augmenting Path)



flow

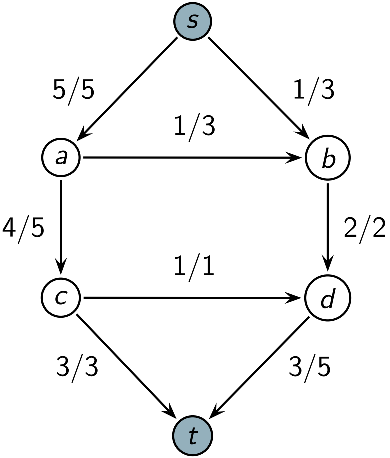
5

notation



edge with capacity c, and current flow f.

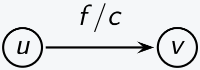
Maximum Flow Example (Augmenting Path)



flow

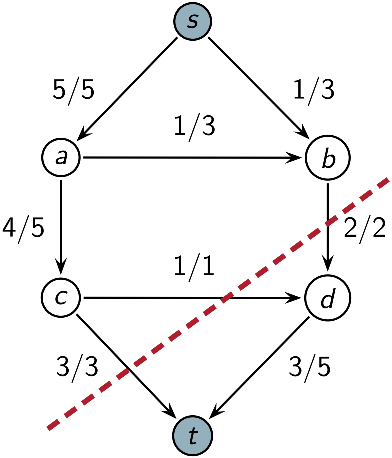
6

notation



edge with capacity c, and current flow f.

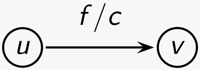
Maximum Flow Example (Augmenting Path)



flow

6

notation

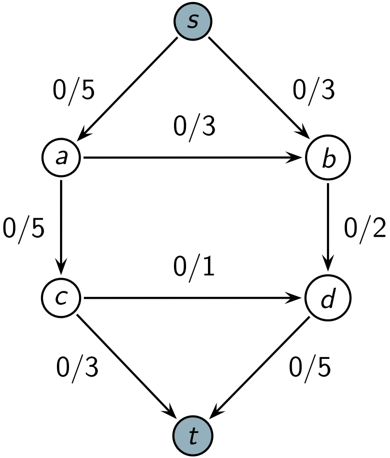


edge with capacity c, and current flow f.

Augmenting Path Algorithm Summary

- while an **augmenting path** exists (directed path with positive capacity between the source and sink)
    - send **flow** along the augmenting path updating **edge capacities** to produce a **residual graph**
- put all nodes reachable from the source in $\mathcal{S}$
- put all nodes that can reach the sink in $\mathcal{T}$

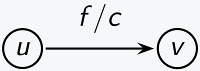
Maximum Flow Example (Push-Relabel)



state

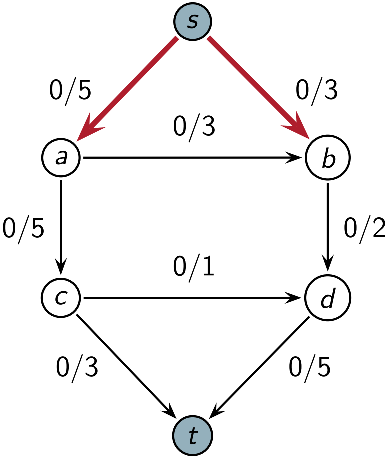
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	0	0
b	0	0
c	0	0
d	0	0
t	0	0

notation



edge with capacity c, current flow f.

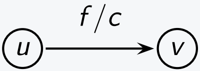
Maximum Flow Example (Push-Relabel)



state

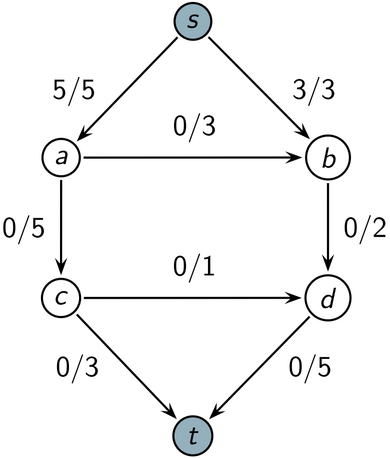
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	0	0
b	0	0
c	0	0
d	0	0
t	0	0

notation



edge with capacity c, current flow f.

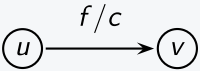
Maximum Flow Example (Push-Relabel)



state

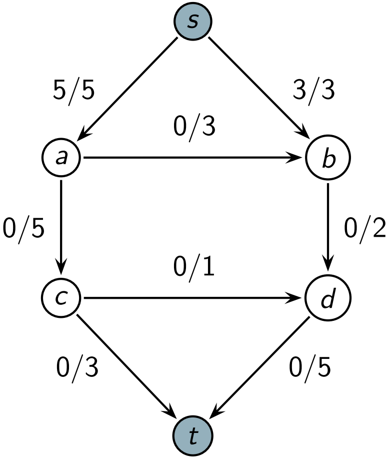
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	0	5
b	0	3
c	0	0
d	0	0
t	0	0

notation



edge with capacity c, current flow f.

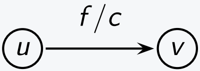
Maximum Flow Example (Push-Relabel)



state

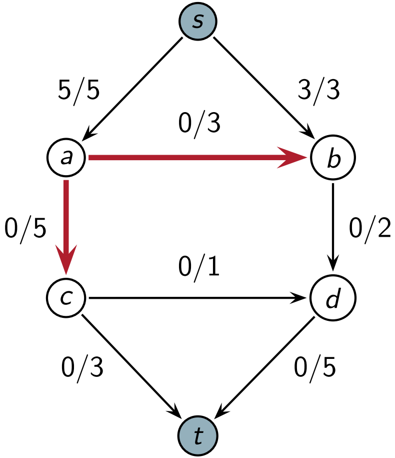
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	5
b	0	3
c	0	0
d	0	0
t	0	0

notation



edge with capacity c, current flow f.

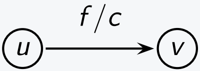
Maximum Flow Example (Push-Relabel)



state

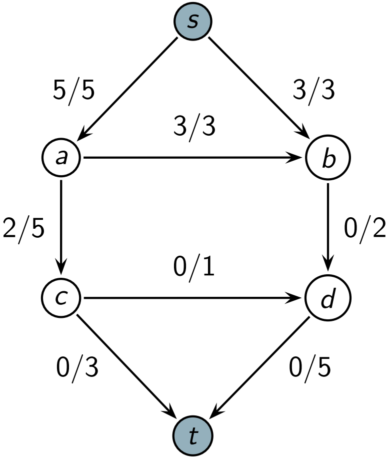
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	5
b	0	3
c	0	0
d	0	0
t	0	0

notation



edge with capacity c, current flow f.

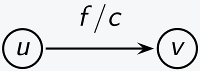
Maximum Flow Example (Push-Relabel)



state

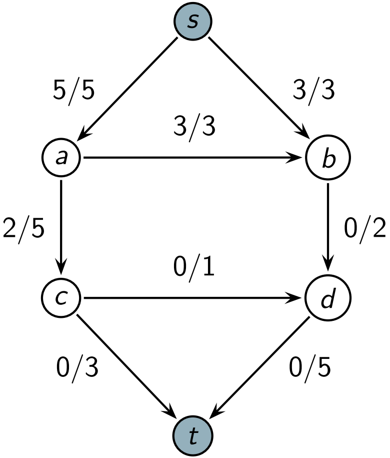
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	0	6
c	0	2
d	0	0
t	0	0

notation



edge with capacity c, current flow f.

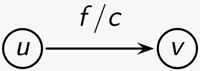
Maximum Flow Example (Push-Relabel)



state

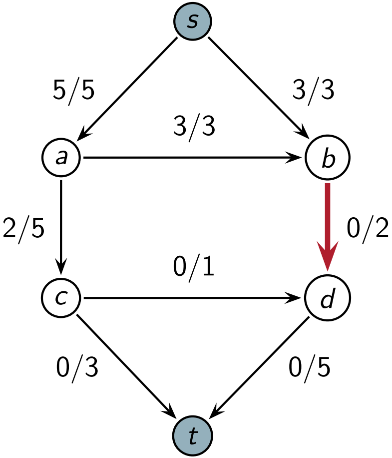
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	1	6
c	0	2
d	0	0
t	0	0

notation



edge with capacity c, current flow f.

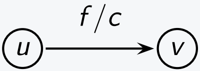
Maximum Flow Example (Push-Relabel)



state

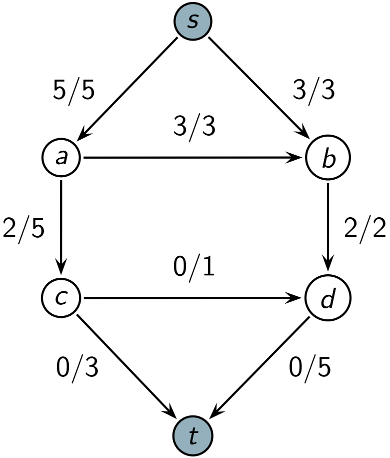
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	1	6
c	0	2
d	0	0
t	0	0

notation



edge with capacity c, current flow f.

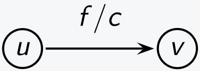
Maximum Flow Example (Push-Relabel)



state

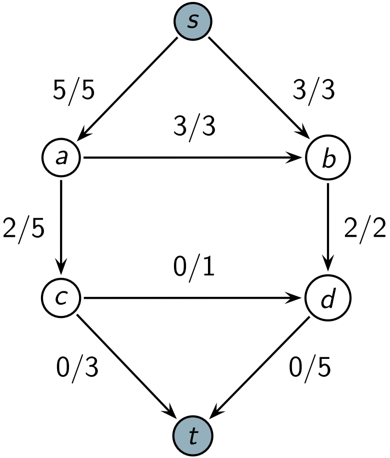
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	1	4
c	0	2
d	0	2
t	0	0

notation



edge with capacity c, current flow f.

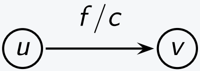
Maximum Flow Example (Push-Relabel)



state

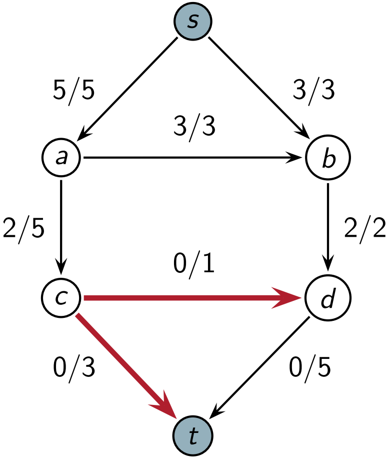
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	1	4
c	1	2
d	0	2
t	0	0

notation



edge with capacity c, current flow f.

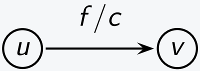
Maximum Flow Example (Push-Relabel)



state

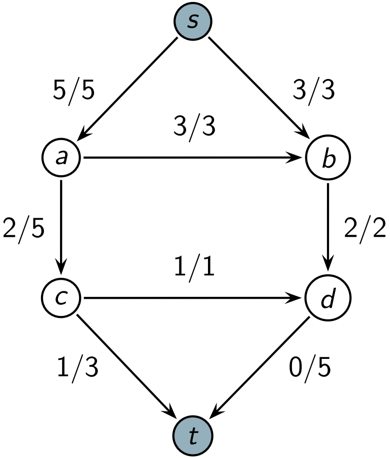
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	1	4
c	1	2
d	0	2
t	0	0

notation



edge with capacity c, current flow f.

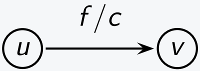
Maximum Flow Example (Push-Relabel)



state

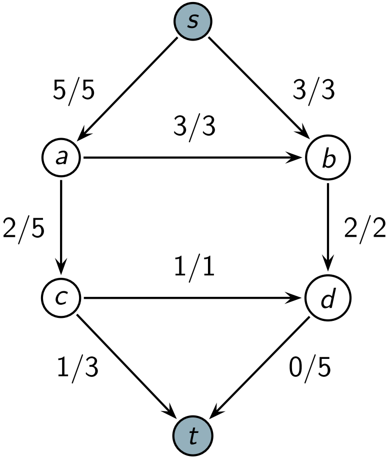
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	1	4
c	1	0
d	0	3
t	0	1

notation



edge with capacity c, current flow f.

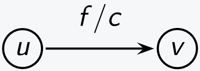
Maximum Flow Example (Push-Relabel)



state

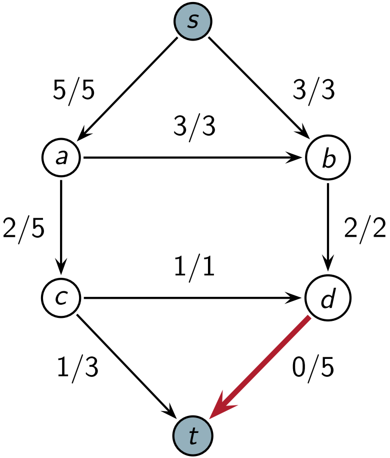
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	1	4
c	1	0
d	1	3
t	0	1

notation



edge with capacity c, current flow f.

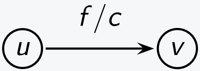
Maximum Flow Example (Push-Relabel)



state

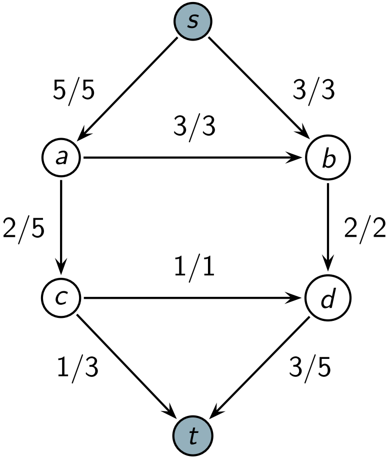
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	1	4
c	1	0
d	1	3
t	0	1

notation



edge with capacity c, current flow f.

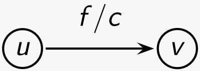
Maximum Flow Example (Push-Relabel)



state

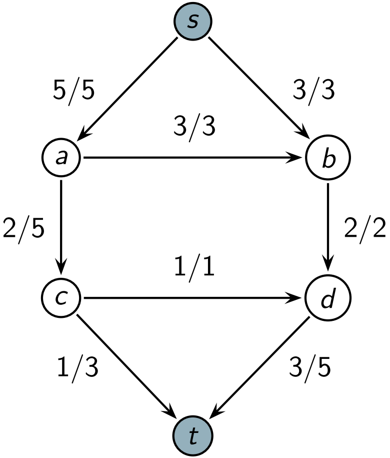
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	1	4
c	1	0
d	1	0
t	0	4

notation



edge with capacity c, current flow f.

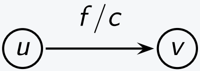
Maximum Flow Example (Push-Relabel)



state

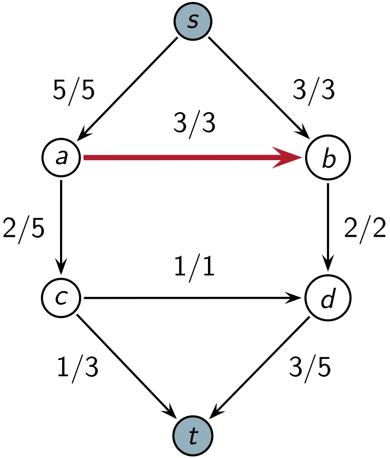
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	2	4
c	1	0
d	1	0
t	0	4

notation



edge with capacity c, current flow f.

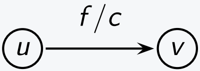
Maximum Flow Example (Push-Relabel)



state

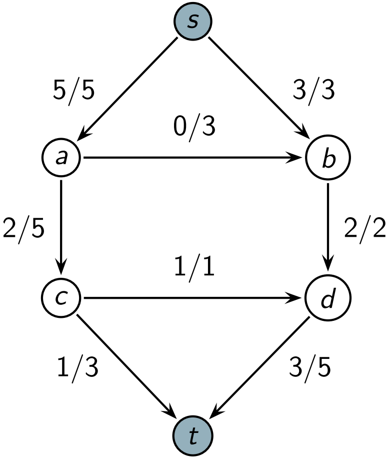
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	0
b	2	4
c	1	0
d	1	0
t	0	4

notation



edge with capacity c, current flow f.

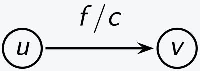
Maximum Flow Example (Push-Relabel)



state

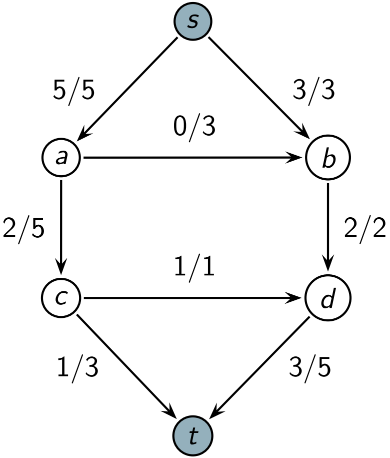
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	1	3
b	2	1
c	1	0
d	1	0
t	0	4

notation



edge with capacity c, current flow f.

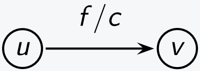
Maximum Flow Example (Push-Relabel)



state

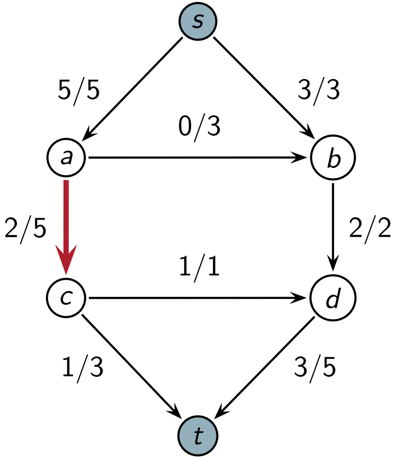
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	2	3
b	2	1
c	1	0
d	1	0
t	0	4

notation



edge with capacity c, current flow f.

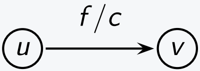
Maximum Flow Example (Push-Relabel)



state

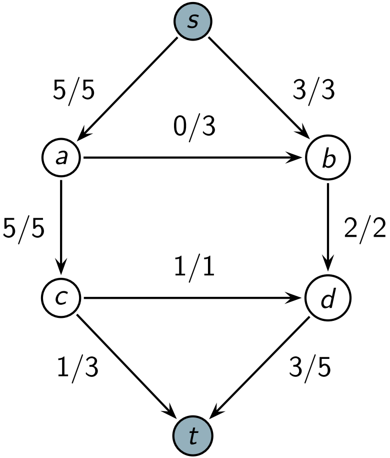
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	2	3
b	2	1
c	1	0
d	1	0
t	0	4

notation



edge with capacity c, current flow f.

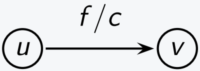
Maximum Flow Example (Push-Relabel)



state

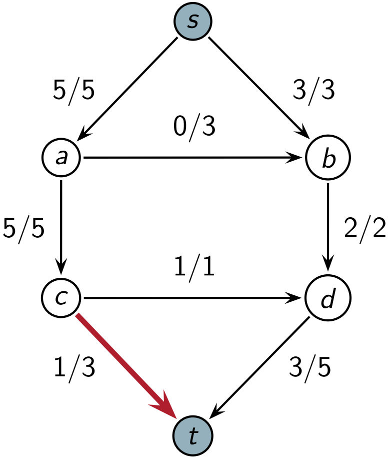
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	2	0
b	2	1
c	1	3
d	1	0
t	0	4

notation



edge with capacity c, current flow f.

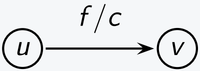
Maximum Flow Example (Push-Relabel)



state

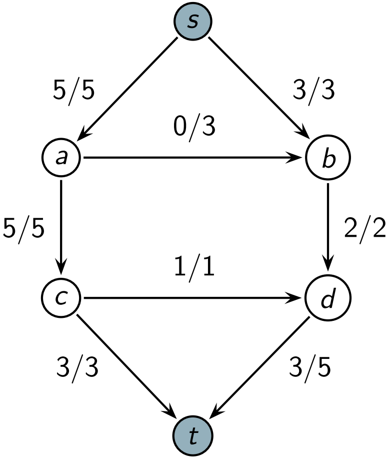
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	2	0
b	2	1
c	1	3
d	1	0
t	0	4

notation



edge with capacity c, current flow f.

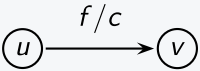
Maximum Flow Example (Push-Relabel)



state

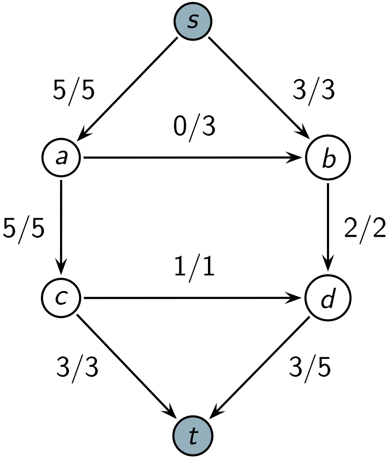
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	2	0
b	2	1
c	1	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

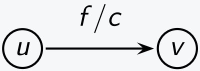
Maximum Flow Example (Push-Relabel)



state

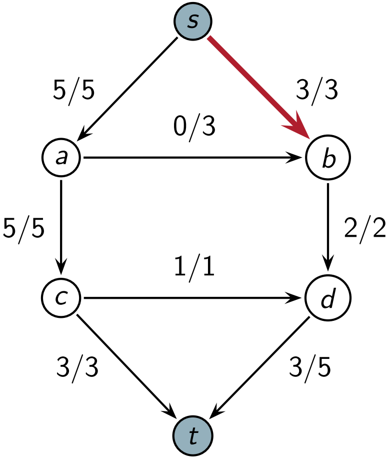
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	2	0
b	7	1
c	1	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

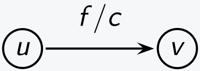
Maximum Flow Example (Push-Relabel)



state

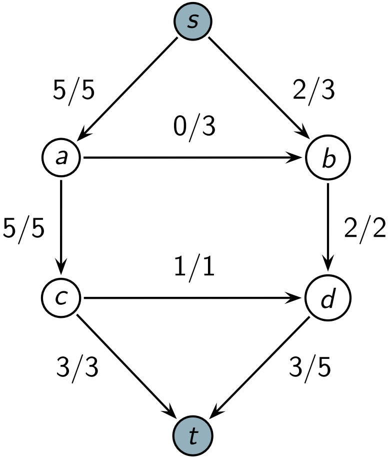
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	2	0
b	7	1
c	1	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

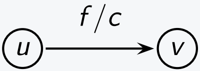
Maximum Flow Example (Push-Relabel)



state

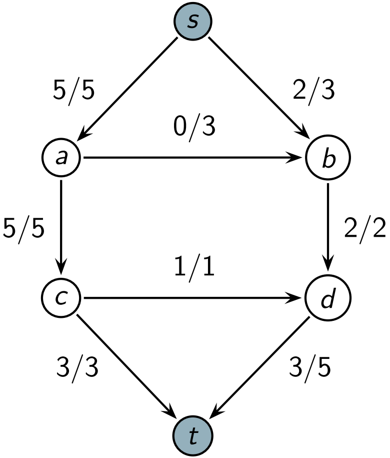
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	2	0
b	7	0
c	1	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

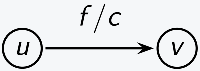
Maximum Flow Example (Push-Relabel)



state

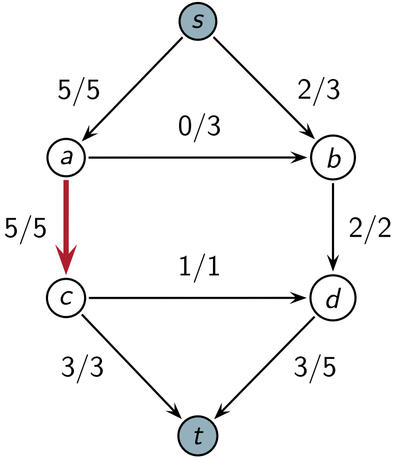
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	2	0
b	7	0
c	3	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

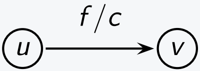
Maximum Flow Example (Push-Relabel)



state

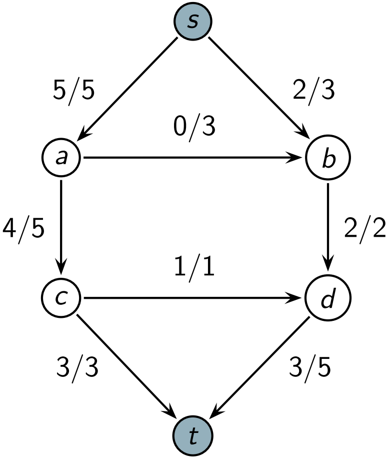
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	2	0
b	7	0
c	3	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

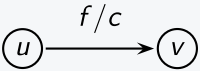
Maximum Flow Example (Push-Relabel)



state

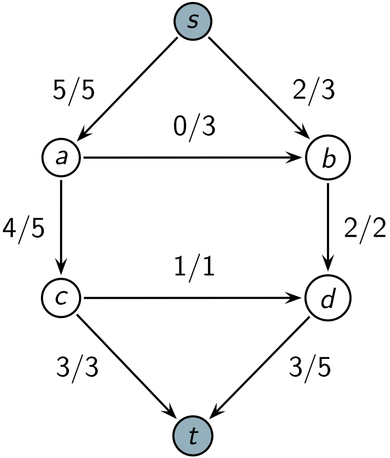
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	2	1
b	7	0
c	3	0
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

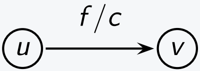
Maximum Flow Example (Push-Relabel)



state

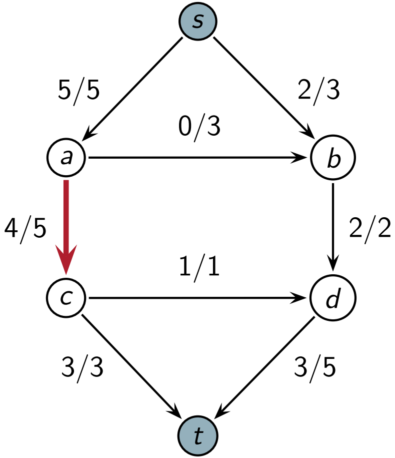
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	4	1
b	7	0
c	3	0
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

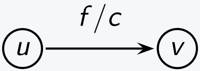
Maximum Flow Example (Push-Relabel)



state

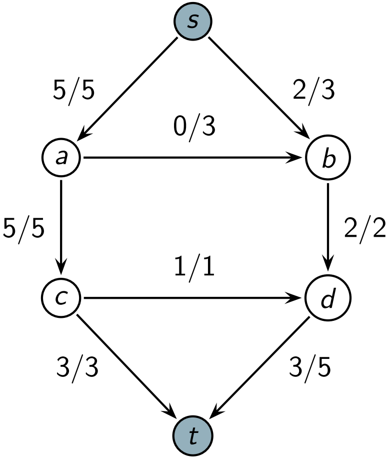
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	4	1
b	7	0
c	3	0
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

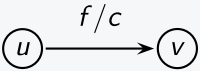
Maximum Flow Example (Push-Relabel)



state

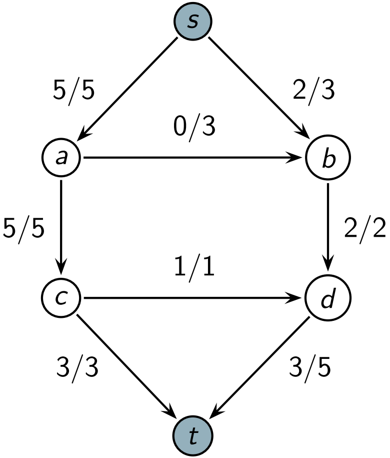
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	4	0
b	7	0
c	3	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

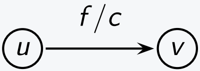
Maximum Flow Example (Push-Relabel)



state

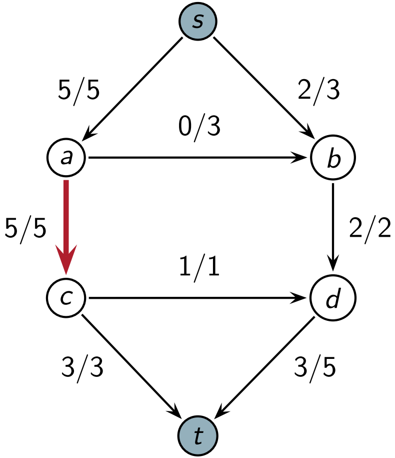
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	4	0
b	7	0
c	5	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

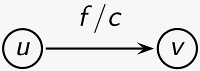
Maximum Flow Example (Push-Relabel)



state

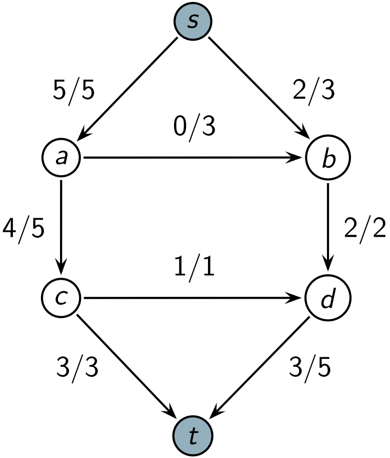
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	4	0
b	7	0
c	5	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

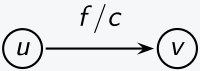
Maximum Flow Example (Push-Relabel)



state

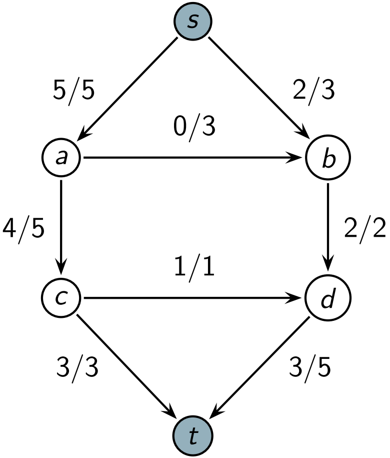
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	4	1
b	7	0
c	5	0
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

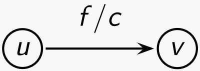
Maximum Flow Example (Push-Relabel)



state

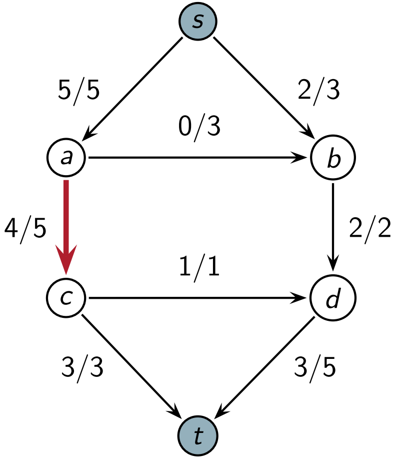
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	6	1
b	7	0
c	5	0
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

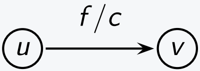
Maximum Flow Example (Push-Relabel)



state

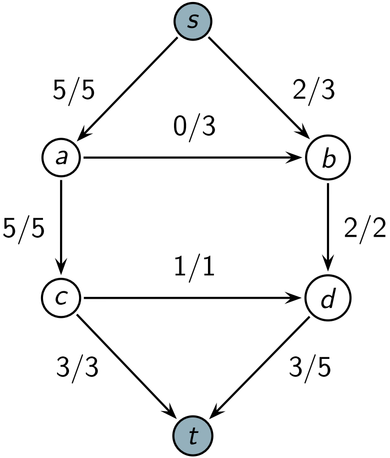
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	6	1
b	7	0
c	5	0
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

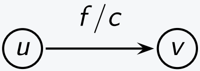
Maximum Flow Example (Push-Relabel)



state

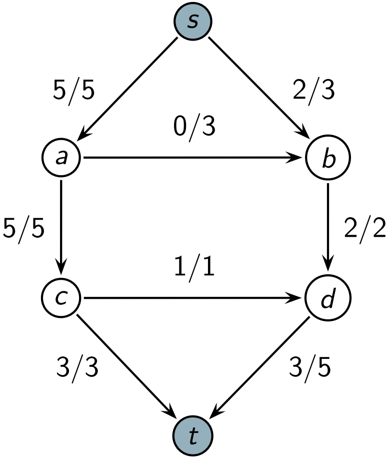
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	6	0
b	7	0
c	5	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

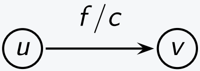
Maximum Flow Example (Push-Relabel)



state

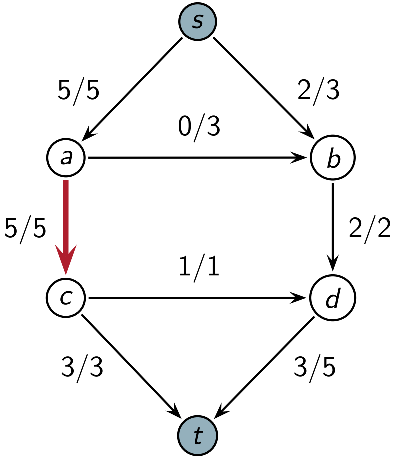
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	6	0
b	7	0
c	7	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

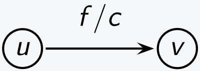
Maximum Flow Example (Push-Relabel)



state

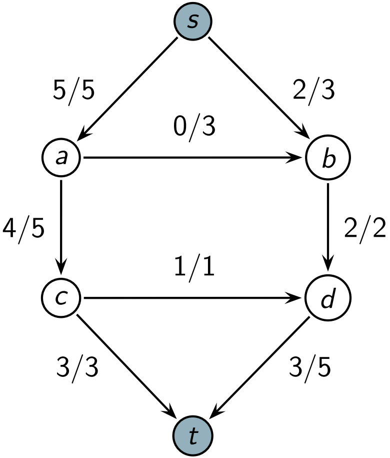
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	6	0
b	7	0
c	7	1
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

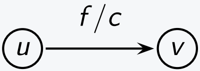
Maximum Flow Example (Push-Relabel)



state

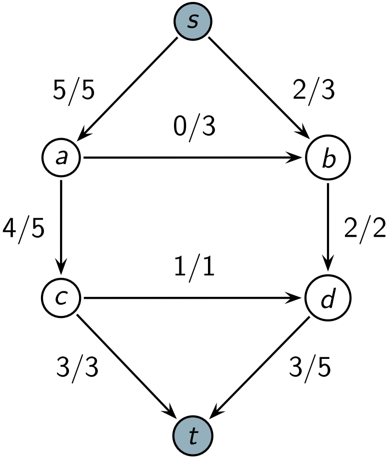
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	6	1
b	7	0
c	7	0
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

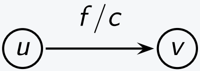
Maximum Flow Example (Push-Relabel)



state

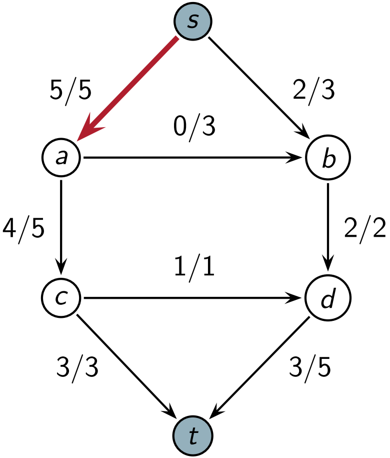
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	7	1
b	7	0
c	7	0
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

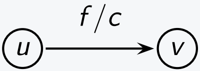
Maximum Flow Example (Push-Relabel)



state

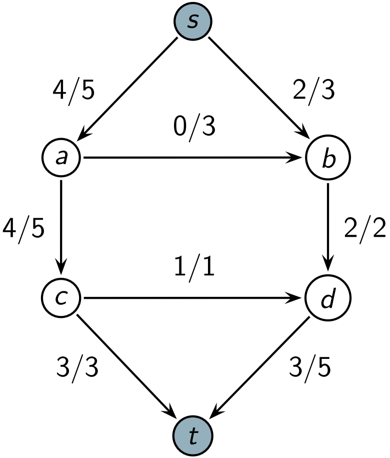
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	7	1
b	7	0
c	7	0
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

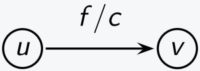
Maximum Flow Example (Push-Relabel)



state

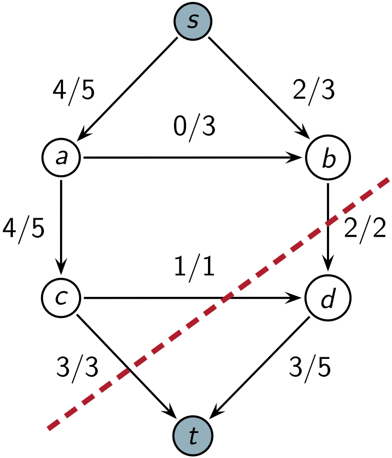
	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	7	0
b	7	0
c	7	0
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

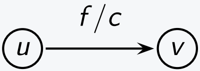
Maximum Flow Example (Push-Relabel)



state

	$h(\cdot)$	$e(\cdot)$
s	6	∞
a	7	0
b	7	0
c	7	0
d	1	0
t	0	6

notation



edge with capacity c, current flow f.

Push-Relabel Algorithm Summary

- **Initialize:** set height of s to number of nodes in the graph; set excess for all nodes to zero.
- **Push:** for a node with excess capacity, push as much flow as possible onto neighbours with lower height
- **Relabel:** for a node with excess capacity and no neighbours with lower height, increase its height to one more than its lowest neighbour (with residual capacity).

Comparison of Maximum Flow Algorithms

Current state-of-the-art algorithm for exact minimization of general submodular pseudo-Boolean functions is $O(n^5 T + n^6)$, where T is the time taken to evaluate the function [Orlin, 2009].

†assumes integer capacities

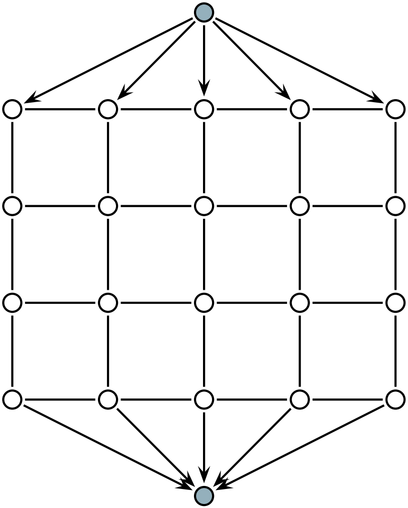
Comparison of Maximum Flow Algorithms

Current state-of-the-art algorithm for exact minimization of general submodular pseudo-Boolean functions is $O(n^5 T + n^6)$, where T is the time taken to evaluate the function [Orlin, 2009].

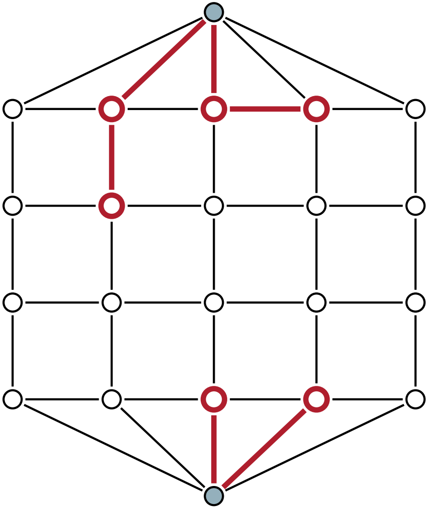
Algorithm	**Complexity**
Ford-Fulkerson	$O(E \max f)^\dagger$
Edmonds-Karp (BFS)	$O(VE^2)$
Push-relabel	$O(V^3)$
Boykov-Kolmogorov	$O(V^2 E \max f)$ ($\sim O(V)$ in practice)

† assumes integer capacities

Maximum Flow (Boykov-Kolmogorov, PAMI 2004)



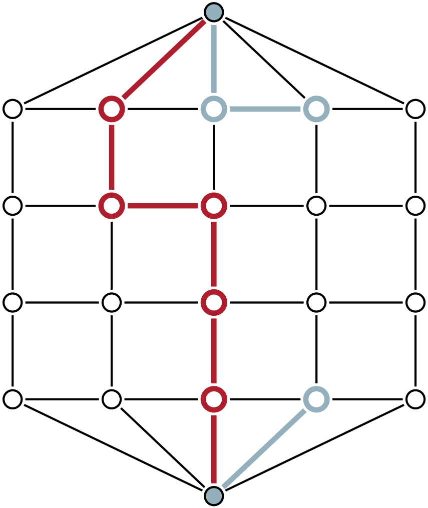
Maximum Flow (Boykov-Kolmogorov, PAMI 2004)



growth stage

search trees from s and t grow until they touch

Maximum Flow (Boykov-Kolmogorov, PAMI 2004)



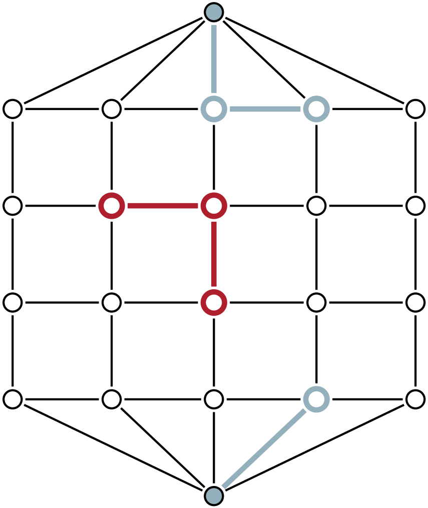
growth stage

search trees from s and t grow until they touch

augmentation stage

the path found is augmented

Maximum Flow (Boykov-Kolmogorov, PAMI 2004)



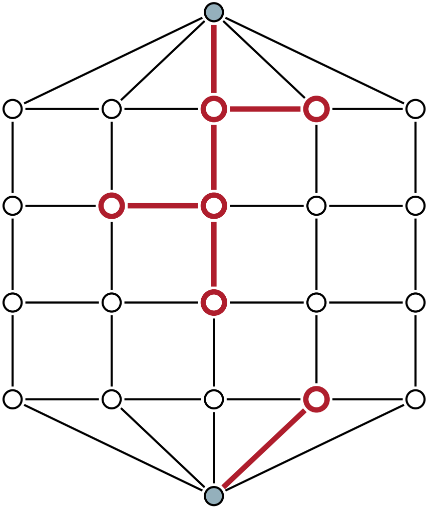
growth stage

search trees from s and t grow until they touch

augmentation stage

the path found is augmented; trees break into forests

Maximum Flow (Boykov-Kolmogorov, PAMI 2004)



growth stage

search trees from s and t grow until they touch

augmentation stage

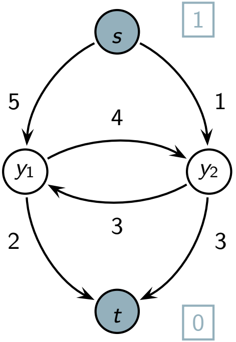
the path found is augmented; trees break into forests

adoption stage

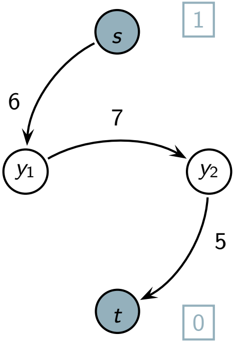
trees are restored

Reparameterization of Energy Functions

$$E(y_1, y_2) = 2y_1 + 5\bar{y}_1 + 3y_2 + \bar{y}_2 + 3\bar{y}_1 y_2 + 4y_1\bar{y}_2$$

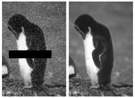


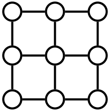
$$E(y_1, y_2) = 6\bar{y}_1 + 5y_2 + 7y_1\bar{y}_2$$



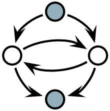
Big Picture: Where are we now?

We can perform inference in submodular binary pairwise Markov random fields exactly.





$\{0,1\}^n \to \mathbb{R}$

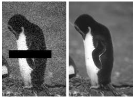


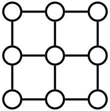
What about...

- non-submodular binary pairwise Markov random fields?
- multi-label Markov random fields?
- higher-order Markov random fields?

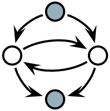
Big Picture: Where are we now?

We can perform inference in submodular binary pairwise Markov random fields exactly.





$\{0,1\}^n \to \mathbb{R}$



What about...

- non-submodular binary pairwise Markov random fields?
- multi-label Markov random fields?
- higher-order Markov random fields?

Non-submodular Binary Pairwise MRFs

Non-submodular binary pairwise MRFs have potentials that do not satisfy $\psi_{ij}^P(0,1) + \psi_{ij}^P(1,0) \geq \psi_{ij}^P(1,1) + \psi_{ij}^P(0,0)$.

They are often handled in one of the following ways:

- approximate the energy function by one that is submodular (i.e., project onto the space of submodular functions);
- solve a relaxation of the problem using QPBO (Rother et al., 2007) or dual-decomposition (Komodakis et al., 2007).

Non-submodular Binary Pairwise MRFs

Non-submodular binary pairwise MRFs have potentials that do not satisfy $\psi^P_{ij}(0,1) + \psi^P_{ij}(1,0) \geq \psi^P_{ij}(1,1) + \psi^P_{ij}(0,0)$.

They are often handled in one of the following ways:

- approximate the energy function by one that is submodular (i.e., project onto the space of submodular functions);
- solve a relaxation of the problem using QPBO (Rother et al., 2007) or dual-decomposition (Komodakis et al., 2007).

Approximating Non-submodular Binary Pairwise MRFs

Consider the non-submodular potential

A	B
C	D

with $A + D > B + C$.

We can project onto a submodular potential by modifying the coefficients as follows:

$$\Delta = A + D - C - B$$
$$A \leftarrow A - \frac{\Delta}{3}$$
$$C \leftarrow C + \frac{\Delta}{3}$$
$$B \leftarrow B + \frac{\Delta}{3}$$

QPBO (Roof Duality) [Rother et al., 2007]

Consider the energy function

$$E(\mathbf{y}) = \sum_{i \in \mathcal{V}} \psi_i^U(y_i) + \underbrace{\sum_{ij \in \mathcal{E}} \psi_{ij}^P(y_i, y_j)}_{\text{submodular}} + \underbrace{\sum_{ij \in \mathcal{E}} \tilde{\psi}_{ij}^P(y_i, y_j)}_{\text{non-submodular}}$$

We can introduce duplicate variables $\bar{y}_i$ into the energy function, and write

$$E'(\mathbf{y}, \bar{\mathbf{y}}) = \sum_{i \in \mathcal{V}} \frac{\psi_i^U(y_i) + \psi_i^U(1 - \bar{y}_i)}{2} + \sum_{ij \in \mathcal{E}} \frac{\psi_{ij}^P(y_i, y_j) + \psi_{ij}^P(1 - \bar{y}_i, 1 - \bar{y}_j)}{2} + \sum_{ij \in \mathcal{E}} \frac{\tilde{\psi}_{ij}^P(y_i, 1 - \bar{y}_j) + \tilde{\psi}_{ij}^P(1 - \bar{y}_i, y_j)}{2}$$

QPBO (Roof Duality) [Rother et al., 2007]

Consider the energy function

$$E(\mathbf{y}) = \sum_{i \in \mathcal{V}} \psi_i^U(y_i) + \underbrace{\sum_{ij \in \mathcal{E}} \psi_{ij}^P(y_i, y_j)}_{\text{submodular}} + \underbrace{\sum_{ij \in \mathcal{E}} \tilde{\psi}_{ij}^P(y_i, y_j)}_{\text{non-submodular}}$$

We can introduce duplicate variables $\bar{y}_i$ into the energy function, and write

$$E'(\mathbf{y}, \bar{\mathbf{y}}) = \sum_{i \in \mathcal{V}} \frac{\psi_i^U(y_i) + \psi_i^U(1 - \bar{y}_i)}{2} + \sum_{ij \in \mathcal{E}} \frac{\psi_{ij}^P(y_i, y_j) + \psi_{ij}^P(1 - \bar{y}_i, 1 - \bar{y}_j)}{2} + \sum_{ij \in \mathcal{E}} \frac{\tilde{\psi}_{ij}^P(y_i, 1 - \bar{y}_j) + \tilde{\psi}_{ij}^P(1 - \bar{y}_i, y_j)}{2}$$

QPBO (Roof Duality)

$$E'(\mathbf{y}, \bar{\mathbf{y}}) = \sum_{i \in \mathcal{V}} \tfrac{1}{2}\psi_i^U(y_i) + \tfrac{1}{2}\psi_i^U(1 - \bar{y}_i)$$
$$+ \sum_{ij \in \mathcal{E}} \tfrac{1}{2}\psi_{ij}^P(y_i, y_j) + \tfrac{1}{2}\psi_{ij}^P(1 - \bar{y}_i, 1 - \bar{y}_j)$$
$$+ \sum_{ij \in \mathcal{E}} \tfrac{1}{2}\tilde{\psi}_{ij}^P(y_i, 1 - \bar{y}_j) + \tfrac{1}{2}\tilde{\psi}_{ij}^P(1 - \bar{y}_i, y_j)$$

Observations

- if $y_i = 1 - \bar{y}_i$ for all i, then $E(\mathbf{y}) = E'(\mathbf{y}, \bar{\mathbf{y}})$.
- $E'(\mathbf{y}, \bar{\mathbf{y}})$ is submodular.

Ignore the constraint on $\bar{y}_i$ and solve anyway. Result satisfies *partial optimality*: if $\bar{y}_i = 1 - y_i$ then y_i is the optimal label.

QPBO (Roof Duality)

$$E'(\mathbf{y}, \bar{\mathbf{y}}) = \sum_{i \in \mathcal{V}} \tfrac{1}{2}\psi_i^U(y_i) + \tfrac{1}{2}\psi_i^U(1 - \bar{y}_i) + \sum_{ij \in \mathcal{E}} \tfrac{1}{2}\psi_{ij}^P(y_i, y_j) + \tfrac{1}{2}\psi_{ij}^P(1 - \bar{y}_i, 1 - \bar{y}_j) + \sum_{ij \in \mathcal{E}} \tfrac{1}{2}\tilde{\psi}_{ij}^P(y_i, 1 - \bar{y}_j) + \tfrac{1}{2}\tilde{\psi}_{ij}^P(1 - \bar{y}_i, y_j)$$

Observations

- if $y_i = 1 - \bar{y}_i$ for all i, then $E(\mathbf{y}) = E'(\mathbf{y}, \bar{\mathbf{y}})$.
- $E'(\mathbf{y}, \bar{\mathbf{y}})$ is submodular.

Ignore the constraint on $\bar{y}_i$ and solve anyway. Result satisfies *partial optimality*: if $\bar{y}_i = 1 - y_i$ then y_i is the optimal label.

Multi-label Markov Random Fields

The quadratic pseudo-Boolean optimization techniques described above cannot be applied directly to multi-label MRFs.

However...

- ...for certain MRFs we can transform the multi-label problem into a binary one exactly.
- ...we can project the multi-label problem onto a series of binary problems in a so-called *move-making* algorithm.

Multi-label Markov Random Fields

The quadratic pseudo-Boolean optimization techniques described above cannot be applied directly to multi-label MRFs.

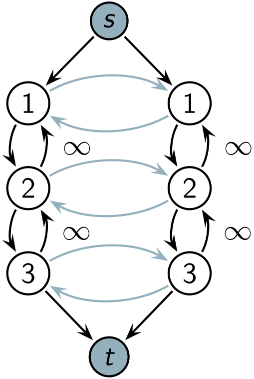
However...

- ...for certain MRFs we can transform the multi-label problem into a binary one exactly.
- ...we can project the multi-label problem onto a series of binary problems in a so-called *move-making* algorithm.

The "Battleship" Transform [Ishikawa, 2003]

If the multi-label MRFs has pairwise potentials that are convex functions over the label differences, i.e., $\psi^P_{ij}(y_i, y_j) = g(|y_i - y_j|)$ where $g(\cdot)$ is convex, then we can transform the energy function into an equivalent binary one.

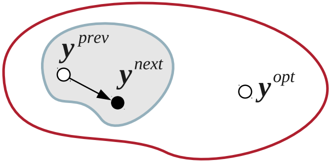
$$y = 1 \Leftrightarrow \mathbf{z} = (0, 0, 0)$$
$$y = 2 \Leftrightarrow \mathbf{z} = (1, 0, 0)$$
$$y = 3 \Leftrightarrow \mathbf{z} = (1, 1, 0)$$
$$y = 4 \Leftrightarrow \mathbf{z} = (1, 1, 1)$$



Move-making Inference

Idea:

- initialize $\mathbf{y}^{\text{prev}}$ to any valid assignment
- restrict the label-space of each variable y_i from $\mathcal{L}$ to $\mathcal{Y}_i \subseteq \mathcal{L}$ (with $y_i^{\text{prev}} \in \mathcal{Y}_i$)
- transform $E : \mathcal{L}^n \to \mathbb{R}$ to $\hat{E} : \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_n \to \mathbb{R}$
- find the optimal assignment $\hat{\mathbf{y}}$ for $\hat{E}$ and repeat



each move results in an assignment with lower energy

Iterated Conditional Modes [Besag, 1986]

Reduce multi-variate inference to solving a series of univariate inference problems.

ICM move

For one of the variables y_i, set $\mathcal{Y}_i = \mathcal{L}$. Set $\mathcal{Y}_j = \{y_j^{\text{prev}}\}$ for all $j \neq i$ (i.e., hold all other variables fixed).

can be used for arbitrary energy functions

Iterated Conditional Modes [Besag, 1986]

Reduce multi-variate inference to solving a series of univariate inference problems.

ICM move

For one of the variables y_i, set $\mathcal{Y}_i = \mathcal{L}$. Set $\mathcal{Y}_j = \{y_j^{\text{prev}}\}$ for all $j \neq i$ (i.e., hold all other variables fixed).

can be used for arbitrary energy functions

Alpha Expansion and Alpha-Beta Swap [Boykov et al., 2001]

Reduce multi-label inference to solving a series of binary (submodular) inference problems.

α-expansion move

Choose some $\alpha \in \mathcal{L}$. Then for all variables, set $\mathcal{Y}_i = \{\alpha, y_i^{\text{prev}}\}$.

$\psi^P_{ij}(\cdot, \cdot)$ must be metric for the resulting move to be submodular

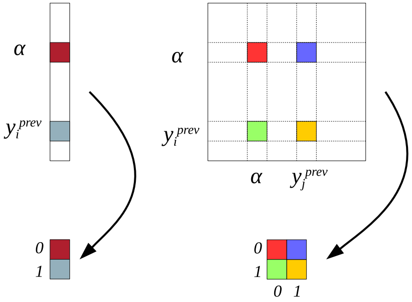
$\alpha\beta$-swap move

Choose two labels $\alpha, \beta \in \mathcal{L}$. Then for each variable y_i such that $y_i^{\text{prev}} \in \{\alpha, \beta\}$, set $\mathcal{Y}_i = \{\alpha, \beta\}$. Otherwise set $\mathcal{Y}_i = \{y_i^{\text{prev}}\}$.

$\psi^P_{ij}(\cdot, \cdot)$ must be semi-metric

Alpha Expansion Potential Construction

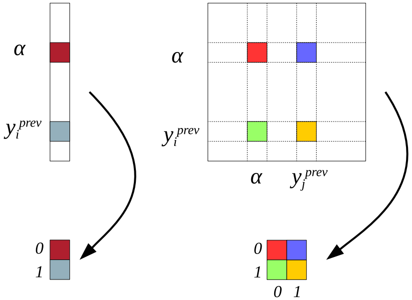
$$y_i^{\text{next}} = \begin{cases} y_i^{\text{prev}} & \text{if } t_i = 1 \\ \alpha & \text{if } t_i = 0 \end{cases}$$



$$E(\mathbf{t}) = \sum_i \psi_i(\alpha)\bar{t}_i + \psi_i(y_i^{\text{prev}})t_i + \sum_{ij} \psi_{ij}(\alpha,\alpha)\bar{t}_i\bar{t}_j$$
$$+ \psi_{ij}(\alpha, y_j^{\text{prev}})\bar{t}_i t_j + \psi_{ij}(y_i^{\text{prev}}, \alpha)t_i\bar{t}_j + \psi_{ij}(y_i^{\text{prev}}, y_j^{\text{prev}})t_i t_j$$

Alpha Expansion Potential Construction

$$y_i^{\text{next}} = \begin{cases} y_i^{\text{prev}} & \text{if } t_i = 1 \\ \alpha & \text{if } t_i = 0 \end{cases}$$



$$E(\mathbf{t}) = \sum_i \psi_i(\alpha)\bar{t}_i + \psi_i(y_i^{\text{prev}})t_i + \sum_{ij} \psi_{ij}(\alpha,\alpha)\bar{t}_i\bar{t}_j + \psi_{ij}(\alpha, y_j^{\text{prev}})\bar{t}_i t_j + \psi_{ij}(y_i^{\text{prev}},\alpha)t_i\bar{t}_j + \psi_{ij}(y_i^{\text{prev}}, y_j^{\text{prev}})t_i t_j$$

A Note on Higher-Order Models

- **Order reduction.** [Ishikawa, 2009]

  Replace $-\prod_{i=1}^{n} y_i$ with $\underbrace{\bar{z} + \sum_{i=1}^{n} \bar{y}_i z - 1}_{*}$.

- **Special forms.** E.g., lower-linear envelopes [Gould, 2011]

$$\psi_c^H(\mathbf{y}_c) \triangleq \min_k \left\{ a_k \sum_{i \in c} y_i + b_k \right\} = \min_k \left\{ f_k(\mathbf{y}_c) \right\}$$

  Assume sorted on a_k. Then replace above with

$$f_1(\mathbf{y}_c) + \underbrace{\sum_k z_k \left(f_{k+1}(\mathbf{y}_c) - f_k(\mathbf{y}_c) \right)}_{\text{* submodular binary pairwise}}$$

A Note on Higher-Order Models

- **Order reduction.** [Ishikawa, 2009]

  Replace $-\prod_{i=1}^{n} y_i$ with $\underbrace{\bar{z} + \sum_{i=1}^{n} \bar{y}_i z - 1}_{*}$.

- **Special forms.** E.g., lower-linear envelopes [Gould, 2011]

  $$\psi_c^H(\mathbf{y}_c) \triangleq \min_k \left\{ a_k \sum_{i \in c} y_i + b_k \right\} = \min_k \{ f_k(\mathbf{y}_c) \}$$

  Assume sorted on a_k. Then replace above with

  $$f_1(\mathbf{y}_c) + \underbrace{\sum_k z_k \left(f_{k+1}(\mathbf{y}_c) - f_k(\mathbf{y}_c) \right)}_{\text{* submodular binary pairwise}}$$

A Note on Higher-Order Models

- **Order reduction.** [Ishikawa, 2009]

  Replace $-\prod_{i=1}^{n} y_i$ with $\underbrace{\bar{z} + \sum_{i=1}^{n} \bar{y}_i z - 1}_{*}$.

- **Special forms.** E.g., lower-linear envelopes [Gould, 2011]

$$\psi_c^H(\mathbf{y}_c) \triangleq \min_k \left\{ a_k \sum_{i \in c} y_i + b_k \right\} = \min_k \left\{ f_k(\mathbf{y}_c) \right\}$$

  Assume sorted on a_k. Then replace above with

$$f_1(\mathbf{y}_c) + \underbrace{\sum_k z_k \left(f_{k+1}(\mathbf{y}_c) - f_k(\mathbf{y}_c) \right)}_{\text{* submodular binary pairwise}}$$

relaxations and dual decomposition

Mathematical Programming Formulation

- Let $\theta_{c,\mathbf{y}_c} \triangleq \psi_c(\mathbf{y}_c)$ and let $\mu_{c,\mathbf{y}_c} \triangleq \begin{cases} 1, & \text{if } \mathbf{Y}_c = \mathbf{y}_c \\ 0, & \text{otherwise} \end{cases}$

$$\operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} \sum_c \psi_c(\mathbf{y}_c)$$

$$\Updownarrow$$

$$\begin{array}{lll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} & \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in \{0,1\}, & \forall c, \mathbf{y}_c \in \mathcal{Y}_c \\ & \sum_{\mathbf{y}_c} \mu_{c,\mathbf{y}_c} = 1, & \forall c \\ & \sum_{\mathbf{y}_c \setminus y_i} \mu_{c,\mathbf{y}_c} = \mu_{i,y_i}, & \forall i \in c, y_i \in \mathcal{Y}_i \end{array}$$

Mathematical Programming Formulation

- Let $\theta_{c,\mathbf{y}_c} \triangleq \psi_c(\mathbf{y}_c)$ and let $\mu_{c,\mathbf{y}_c} \triangleq \begin{cases} 1, & \text{if } \mathbf{Y}_c = \mathbf{y}_c \\ 0, & \text{otherwise} \end{cases}$

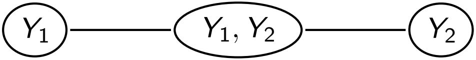
$$\underset{\mathbf{y} \in \mathcal{Y}}{\text{argmin}} \sum_c \psi_c(\mathbf{y}_c)$$

$$\Updownarrow$$

$$\begin{array}{lll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} & \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in \{0,1\}, & \forall c, \mathbf{y}_c \in \mathcal{Y}_c \\ & \sum_{\mathbf{y}_c} \mu_{c,\mathbf{y}_c} = 1, & \forall c \\ & \sum_{\mathbf{y}_c \setminus y_i} \mu_{c,\mathbf{y}_c} = \mu_{i,y_i}, & \forall i \in c, y_i \in \mathcal{Y}_i \end{array}$$

Binary Integer Program: Example

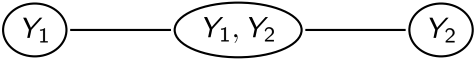
Consider energy function $E(y_1, y_2) = \psi_1(y_1) + \psi_{12}(y_1, y_2) + \psi_2(y_2)$ for binary variables y_1 and y_2.



$$\boldsymbol{\theta} = \begin{bmatrix} \psi_1(0) \\ \psi_1(1) \\ \psi_2(0) \\ \psi_2(1) \\ \psi_{12}(0,0) \\ \psi_{12}(1,0) \\ \psi_{12}(0,1) \\ \psi_{12}(1,1) \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} \mu_{1,0} \\ \mu_{1,1} \\ \mu_{2,0} \\ \mu_{2,1} \\ \mu_{12,00} \\ \mu_{12,10} \\ \mu_{12,01} \\ \mu_{12,11} \end{bmatrix} \qquad \text{s.t.} \begin{cases} \mu_{1,0} + \mu_{1,1} = 1 \\ \mu_{2,0} + \mu_{2,1} = 1 \\ \mu_{12,00} + \mu_{12,10} \\ \quad + \mu_{12,01} + \mu_{12,11} = 1 \\ \mu_{12,00} + \mu_{12,01} = \mu_{1,0} \\ \mu_{12,10} + \mu_{12,11} = \mu_{1,1} \\ \mu_{12,00} + \mu_{12,10} = \mu_{2,0} \\ \mu_{12,01} + \mu_{12,11} = \mu_{2,1} \end{cases}$$

Binary Integer Program: Example

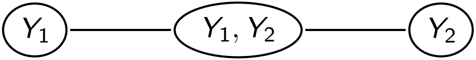
Consider energy function $E(y_1, y_2) = \psi_1(y_1) + \psi_{12}(y_1, y_2) + \psi_2(y_2)$ for binary variables y_1 and y_2.



$$\boldsymbol{\theta} = \begin{bmatrix} \psi_1(0) \\ \psi_1(1) \\ \psi_2(0) \\ \psi_2(1) \\ \psi_{12}(0,0) \\ \psi_{12}(1,0) \\ \psi_{12}(0,1) \\ \psi_{12}(1,1) \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} \mu_{1,0} \\ \mu_{1,1} \\ \mu_{2,0} \\ \mu_{2,1} \\ \mu_{12,00} \\ \mu_{12,10} \\ \mu_{12,01} \\ \mu_{12,11} \end{bmatrix} \qquad \text{s.t.} \begin{cases} \mu_{1,0} + \mu_{1,1} = 1 \\ \mu_{2,0} + \mu_{2,1} = 1 \\ \mu_{12,00} + \mu_{12,10} \\ \quad + \mu_{12,01} + \mu_{12,11} = 1 \\ \mu_{12,00} + \mu_{12,01} = \mu_{1,0} \\ \mu_{12,10} + \mu_{12,11} = \mu_{1,1} \\ \mu_{12,00} + \mu_{12,10} = \mu_{2,0} \\ \mu_{12,01} + \mu_{12,11} = \mu_{2,1} \end{cases}$$

Binary Integer Program: Example

Consider energy function $E(y_1, y_2) = \psi_1(y_1) + \psi_{12}(y_1, y_2) + \psi_2(y_2)$ for binary variables y_1 and y_2.



$$\boldsymbol{\theta} = \begin{bmatrix} \psi_1(0) \\ \psi_1(1) \\ \psi_2(0) \\ \psi_2(1) \\ \psi_{12}(0,0) \\ \psi_{12}(1,0) \\ \psi_{12}(0,1) \\ \psi_{12}(1,1) \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} \mu_{1,0} \\ \mu_{1,1} \\ \mu_{2,0} \\ \mu_{2,1} \\ \mu_{12,00} \\ \mu_{12,10} \\ \mu_{12,01} \\ \mu_{12,11} \end{bmatrix} \qquad \text{s.t.} \left\{ \begin{aligned} \mu_{1,0} + \mu_{1,1} &= 1 \\ \mu_{2,0} + \mu_{2,1} &= 1 \\ \mu_{12,00} + \mu_{12,10} & \\ + \mu_{12,01} + \mu_{12,11} &= 1 \\ \mu_{12,00} + \mu_{12,01} &= \mu_{1,0} \\ \mu_{12,10} + \mu_{12,11} &= \mu_{1,1} \\ \mu_{12,00} + \mu_{12,10} &= \mu_{2,0} \\ \mu_{12,01} + \mu_{12,11} &= \mu_{2,1} \end{aligned} \right.$$

Binary Integer Program: Example

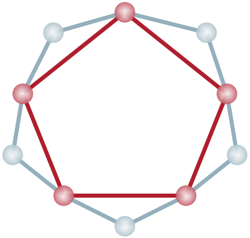
Let $y_1 = 1$ and $y_2 = 0$. Then

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_{1,0} \\ \mu_{1,1} \\ \mu_{2,0} \\ \mu_{2,1} \\ \mu_{12,00} \\ \mu_{12,10} \\ \mu_{12,01} \\ \mu_{12,11} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \cdot \quad \boldsymbol{\theta} = \begin{bmatrix} \psi_1(0) \\ \psi_1(1) \\ \psi_2(0) \\ \psi_2(1) \\ \psi_{12}(0,0) \\ \psi_{12}(1,0) \\ \psi_{12}(0,1) \\ \psi_{12}(1,1) \end{bmatrix}$$

So $\boldsymbol{\theta}^T \boldsymbol{\mu} = \psi_1(1) + \psi_2(0) + \psi_{12}(1,0)$.

Local Marginal Polytope

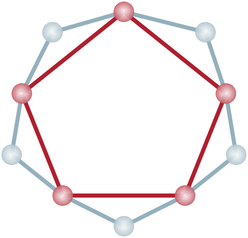
$$\mathcal{M} = \left\{ \boldsymbol{\mu} \geq \mathbf{0} \;\middle|\; \begin{array}{ll} \sum_{y_i} \mu_{i,y_i} = 1, & \forall i \\ \sum_{\mathbf{y}_c \backslash y_i} \mu_{c,\mathbf{y}_c} = \mu_{i,y_i}, & \forall i \in c, y_i \in \mathcal{Y}_i \end{array} \right\}$$



- $\mathcal{M}$ is tight if factor graph is a tree
- for cyclic graphs $\mathcal{M}$ may contain fractional vertices
- for submodular energies, factional solutions are never optimal

Local Marginal Polytope

$$\mathcal{M} = \left\{ \boldsymbol{\mu} \geq \mathbf{0} \;\middle|\; \begin{array}{ll} \sum_{y_i} \mu_{i,y_i} = 1, & \forall i \\ \sum_{\mathbf{y}_c \setminus y_i} \mu_{c,\mathbf{y}_c} = \mu_{i,y_i}, & \forall i \in c, y_i \in \mathcal{Y}_i \end{array} \right\}$$



- $\mathcal{M}$ is tight if factor graph is a tree
- for cyclic graphs $\mathcal{M}$ may contain fractional vertices
- for submodular energies, factional solutions are never optimal

Linear Programming (LP) Relaxation

- Binary integer program

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in \{0, 1\} \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Linear program

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in [0, 1] \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Solution by standard LP solvers typically infeasible due to large number of variables and constraints
- More easily solved via coordinate ascent of the dual
- Solutions need to be rounded or decoded

Linear Programming (LP) Relaxation

- Binary integer program

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in \{0, 1\} \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Linear program

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in [0, 1] \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Solution by standard LP solvers typically infeasible due to large number of variables and constraints
- More easily solved via coordinate ascent of the dual
- Solutions need to be rounded or decoded

Linear Programming (LP) Relaxation

- Binary integer program

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in \{0, 1\} \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Linear program

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in [0, 1] \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Solution by standard LP solvers typically infeasible due to large number of variables and constraints
- More easily solved via coordinate ascent of the dual
- Solutions need to be rounded or decoded

Linear Programming (LP) Relaxation

- Binary integer program

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in \{0, 1\} \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Linear program

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in [0, 1] \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Solution by standard LP solvers typically infeasible due to large number of variables and constraints
- More easily solved via coordinate ascent of the dual
- Solutions need to be rounded or decoded

Linear Programming (LP) Relaxation

- Binary integer program

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in \{0, 1\} \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Linear program

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \boldsymbol{\theta}^T \boldsymbol{\mu} \\ \text{subject to} & \mu_{c,\mathbf{y}_c} \in [0, 1] \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Solution by standard LP solvers typically infeasible due to large number of variables and constraints
- More easily solved via coordinate ascent of the dual
- Solutions need to be rounded or decoded

Dual Decomposition: Rewriting the Primal

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \sum_c \boldsymbol{\theta}_c^T \boldsymbol{\mu}_c \\ \text{subject to} & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

$\Updownarrow$ (pad $\boldsymbol{\theta}_c$)

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu} \\ \text{subject to} & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

$\Updownarrow$ (introduce copies of $\boldsymbol{\mu}$)

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}, \{\boldsymbol{\mu}^c\}) & \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu}^c \\ \text{subject to} & \boldsymbol{\mu}^c = \boldsymbol{\mu} \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

Dual Decomposition: Rewriting the Primal

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \sum_c \boldsymbol{\theta}_c^T \boldsymbol{\mu}_c \\ \text{subject to} & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

$\Updownarrow$ (pad $\boldsymbol{\theta}_c$)

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu} \\ \text{subject to} & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

$\Updownarrow$ (introduce copies of $\boldsymbol{\mu}$)

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}, \{\boldsymbol{\mu}^c\}) & \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu}^c \\ \text{subject to} & \boldsymbol{\mu}^c = \boldsymbol{\mu} \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

Dual Decomposition: Rewriting the Primal

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \sum_c \boldsymbol{\theta}_c^T \boldsymbol{\mu}_c \\ \text{subject to} & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

$\Updownarrow$ (pad $\boldsymbol{\theta}_c$)

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}) & \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu} \\ \text{subject to} & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

$\Updownarrow$ (introduce copies of $\boldsymbol{\mu}$)

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}, \{\boldsymbol{\mu}^c\}) & \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu}^c \\ \text{subject to} & \boldsymbol{\mu}^c = \boldsymbol{\mu} \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

Dual Decomposition: Forming the Dual

- Primal problem

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}, \{\boldsymbol{\mu}^c\}) & \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu}^c \\ \text{subject to} & \boldsymbol{\mu}^c = \boldsymbol{\mu} \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Introducing dual variables $\boldsymbol{\lambda}_c$ we have Lagrangian

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mu}, \{\boldsymbol{\mu}^c\}, \{\boldsymbol{\lambda}_c\}) &= \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu}^c + \sum_c \boldsymbol{\lambda}_c^T (\boldsymbol{\mu}^c - \boldsymbol{\mu}) \\ &= \sum_c (\tilde{\boldsymbol{\theta}}_c + \boldsymbol{\lambda}_c)^T \boldsymbol{\mu}^c - \sum_c \boldsymbol{\lambda}_c^T \boldsymbol{\mu} \end{aligned}$$

Dual Decomposition: Forming the Dual

- Primal problem

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}, \{\boldsymbol{\mu}^c\}) & \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu}^c \\ \text{subject to} & \boldsymbol{\mu}^c = \boldsymbol{\mu} \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Introducing dual variables $\boldsymbol{\lambda}_c$ we have Lagrangian

$$\mathcal{L}(\boldsymbol{\mu}, \{\boldsymbol{\mu}^c\}, \{\boldsymbol{\lambda}_c\}) = \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu}^c + \sum_c \boldsymbol{\lambda}_c^T (\boldsymbol{\mu}^c - \boldsymbol{\mu})$$

$$= \sum_c (\tilde{\theta}_c + \lambda_c)^T \mu^c - \sum_c \lambda_c^T \mu$$

Dual Decomposition: Forming the Dual

- Primal problem

$$\begin{array}{ll} \text{minimize (over } \boldsymbol{\mu}, \{\boldsymbol{\mu}^c\}) & \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu}^c \\ \text{subject to} & \boldsymbol{\mu}^c = \boldsymbol{\mu} \\ & \boldsymbol{\mu} \in \mathcal{M} \end{array}$$

- Introducing dual variables $\boldsymbol{\lambda}_c$ we have Lagrangian

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mu}, \{\boldsymbol{\mu}^c\}, \{\boldsymbol{\lambda}_c\}) &= \sum_c \tilde{\boldsymbol{\theta}}_c^T \boldsymbol{\mu}^c + \sum_c \boldsymbol{\lambda}_c^T (\boldsymbol{\mu}^c - \boldsymbol{\mu}) \\ &= \sum_c (\tilde{\boldsymbol{\theta}}_c + \boldsymbol{\lambda}_c)^T \boldsymbol{\mu}^c - \sum_c \boldsymbol{\lambda}_c^T \boldsymbol{\mu} \end{aligned}$$

Dual Decomposition

$$\begin{array}{ll} \underset{\{\boldsymbol{\lambda}_c\}}{\text{maximize}} & \min_{\{\boldsymbol{\mu}^c\}} \sum_c (\tilde{\boldsymbol{\theta}}_c + \boldsymbol{\lambda}_c)^T \boldsymbol{\mu}^c \\ \text{subject to} & \sum_c \boldsymbol{\lambda}_c = 0 \end{array}$$

$$\Updownarrow$$

$$\begin{array}{ll} \underset{\{\boldsymbol{\lambda}_c\}}{\text{maximize}} & \sum_c \min_{\boldsymbol{\mu}^c} (\tilde{\boldsymbol{\theta}}_c + \boldsymbol{\lambda}_c)^T \boldsymbol{\mu}^c \\ \text{subject to} & \sum_c \boldsymbol{\lambda}_c = 0 \end{array}$$

$$\Updownarrow$$

$$\begin{array}{ll} \underset{\{\lambda_c\}}{\text{maximize}} & \sum_c \min_{\mathbf{y}_c} \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c) \\ \text{subject to} & \sum_c \boldsymbol{\lambda}_c = 0 \end{array}$$

Dual Decomposition

$$\begin{array}{ll} \underset{\{\boldsymbol{\lambda}_c\}}{\text{maximize}} & \min_{\{\boldsymbol{\mu}^c\}} \sum_c (\tilde{\boldsymbol{\theta}}_c + \boldsymbol{\lambda}_c)^T \boldsymbol{\mu}^c \\ \text{subject to} & \sum_c \boldsymbol{\lambda}_c = 0 \end{array}$$

$$\Updownarrow$$

$$\begin{array}{ll} \underset{\{\boldsymbol{\lambda}_c\}}{\text{maximize}} & \sum_c \min_{\boldsymbol{\mu}^c} (\tilde{\boldsymbol{\theta}}_c + \boldsymbol{\lambda}_c)^T \boldsymbol{\mu}^c \\ \text{subject to} & \sum_c \boldsymbol{\lambda}_c = 0 \end{array}$$

$$\Updownarrow$$

$$\begin{array}{ll} \underset{\{\lambda_c\}}{\text{maximize}} & \sum_c \min_{\mathbf{y}_c} \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c) \\ \text{subject to} & \sum_c \boldsymbol{\lambda}_c = 0 \end{array}$$

Dual Decomposition

$$\begin{array}{ll} \underset{\{\boldsymbol{\lambda}_c\}}{\text{maximize}} & \min_{\{\boldsymbol{\mu}^c\}} \sum_c (\tilde{\boldsymbol{\theta}}_c + \boldsymbol{\lambda}_c)^T \boldsymbol{\mu}^c \\ \text{subject to} & \sum_c \boldsymbol{\lambda}_c = 0 \end{array}$$

$$\Updownarrow$$

$$\begin{array}{ll} \underset{\{\boldsymbol{\lambda}_c\}}{\text{maximize}} & \sum_c \min_{\boldsymbol{\mu}^c} (\tilde{\boldsymbol{\theta}}_c + \boldsymbol{\lambda}_c)^T \boldsymbol{\mu}^c \\ \text{subject to} & \sum_c \boldsymbol{\lambda}_c = 0 \end{array}$$

$$\Updownarrow$$

$$\begin{array}{ll} \underset{\{\boldsymbol{\lambda}_c\}}{\text{maximize}} & \sum_c \min_{\mathbf{y}_c} \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c) \\ \text{subject to} & \sum_c \boldsymbol{\lambda}_c = 0 \end{array}$$

Dual Lower Bound

$$E(\mathbf{y}) = \sum_c \psi_c(\mathbf{y}_c)$$

$$= \sum_c \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c) \quad \left(\text{iff } \sum_c \lambda_c(\mathbf{y}_c) = 0\right)$$

$$\min_{\mathbf{y}} E(\mathbf{y}) \geq \sum_c \min_{\mathbf{y}_c} \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c)$$

$$\min_{\mathbf{y}} E(\mathbf{y}) \geq \max_{\{\lambda_c\}: \sum_c \lambda_c = 0} \sum_c \min_{\mathbf{y}_c} \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c)$$

Dual Lower Bound

$$E(\mathbf{y}) = \sum_c \psi_c(\mathbf{y}_c)$$

$$= \sum_c \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c) \quad \left(\text{iff } \sum_c \lambda_c(\mathbf{y}_c) = 0\right)$$

$$\min_{\mathbf{y}} E(\mathbf{y}) \;\geq\; \sum_c \min_{\mathbf{y}_c} \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c)$$

$$\min_{\mathbf{y}} E(\mathbf{y}) \;\geq\; \max_{\{\lambda_c\}:\sum_c \lambda_c = 0} \sum_c \min_{\mathbf{y}_c} \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c)$$

Dual Lower Bound

$$E(\mathbf{y}) = \sum_c \psi_c(\mathbf{y}_c)$$

$$= \sum_c \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c) \quad \left(\text{iff } \sum_c \lambda_c(\mathbf{y}_c) = 0\right)$$

$$\min_{\mathbf{y}} E(\mathbf{y}) \;\geq\; \sum_c \min_{\mathbf{y}_c} \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c)$$

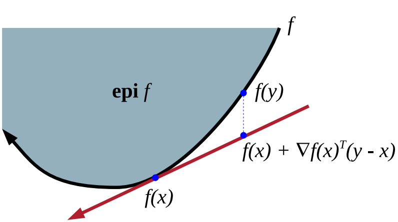
$$\min_{\mathbf{y}} E(\mathbf{y}) \;\geq\; \max_{\{\lambda_c\}:\sum_c \lambda_c = 0} \sum_c \min_{\mathbf{y}_c} \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c)$$

Subgradients

Subgradient

A subgradient of a function f at x is *any* vector g satisfying

$$f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y$$



Subgradient Method

The basic subgradient method is a algorithm for minimizing a nondifferentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$.

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$$

- $x^{(k)}$ is the k-th iterate
- $g^{(k)}$ is any subgradient of f at $x^{(k)}$
- $\alpha_k > 0$ is the k-th step size

It is possible that $-g^{(k)}$ is not a descent direction for f at $x^{(k)}$, so we keep track of the best point found so far

$$f_{\text{best}}^{(k)} = \min\left\{ f_{\text{best}}^{(k-1)}, f(x^{(k)}) \right\}$$

Subgradient Method

The basic subgradient method is a algorithm for minimizing a nondifferentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$.

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$$

- $x^{(k)}$ is the k-th iterate
- $g^{(k)}$ is any subgradient of f at $x^{(k)}$
- $\alpha_k > 0$ is the k-th step size

It is possible that $-g^{(k)}$ is not a descent direction for f at $x^{(k)}$, so we keep track of the best point found so far

$$f_{\text{best}}^{(k)} = \min\left\{ f_{\text{best}}^{(k-1)}, f(x^{(k)}) \right\}$$

Step Size Rules

Step sizes are chosen ahead of time (unlike line search is ordinary gradient methods). A few common step size schedules are:

- **constant step size:** $\alpha_k = \alpha$
- **constant step length:** $\alpha_k = \frac{\gamma}{\|g^{(k)}\|_2}$
- **square summable but not summable:**

$$\sum_{k=1}^{\infty} \alpha_k^2 < \infty, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty$$

- **nonsummable diminishing:**

$$\lim_{k\to\infty} \alpha_k = 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty$$

- **nonsummable diminishing step lengths:** $\alpha_k = \frac{\gamma_k}{\|g^{(k)}\|_2}$

$$\lim_{k\to\infty} \gamma_k = 0, \qquad \sum_{k=1}^{\infty} \gamma_k = \infty$$

Step Size Rules

Step sizes are chosen ahead of time (unlike line search is ordinary gradient methods). A few common step size schedules are:

- **constant step size:** $\alpha_k = \alpha$
- **constant step length:** $\alpha_k = \frac{\gamma}{\|g^{(k)}\|_2}$
- **square summable but not summable:**

$$\sum_{k=1}^{\infty} \alpha_k^2 < \infty, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty$$

- **nonsummable diminishing:**

$$\lim_{k \to \infty} \alpha_k = 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty$$

- **nonsummable diminishing step lengths:** $\alpha_k = \frac{\gamma_k}{\|g^{(k)}\|_2}$

$$\lim_{k \to \infty} \gamma_k = 0, \qquad \sum_{k=1}^{\infty} \gamma_k = \infty$$

Step Size Rules

Step sizes are chosen ahead of time (unlike line search is ordinary gradient methods). A few common step size schedules are:

- **constant step size:** $\alpha_k = \alpha$
- **constant step length:** $\alpha_k = \frac{\gamma}{\|g^{(k)}\|_2}$
- **square summable but not summable:**

$$\sum_{k=1}^{\infty} \alpha_k^2 < \infty, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty$$

- **nonsummable diminishing:**

$$\lim_{k \to \infty} \alpha_k = 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty$$

- **nonsummable diminishing step lengths:** $\alpha_k = \frac{\gamma_k}{\|g^{(k)}\|_2}$

$$\lim_{k \to \infty} \gamma_k = 0, \qquad \sum_{k=1}^{\infty} \gamma_k = \infty$$

Step Size Rules

Step sizes are chosen ahead of time (unlike line search is ordinary gradient methods). A few common step size schedules are:

- **constant step size:** $\alpha_k = \alpha$
- **constant step length:** $\alpha_k = \frac{\gamma}{\|g^{(k)}\|_2}$
- **square summable but not summable:**

$$\sum_{k=1}^{\infty} \alpha_k^2 < \infty, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty$$

- **nonsummable diminishing:**

$$\lim_{k \to \infty} \alpha_k = 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty$$

- **nonsummable diminishing step lengths:** $\alpha_k = \frac{\gamma_k}{\|g^{(k)}\|_2}$

$$\lim_{k \to \infty} \gamma_k = 0, \qquad \sum_{k=1}^{\infty} \gamma_k = \infty$$

Step Size Rules

Step sizes are chosen ahead of time (unlike line search is ordinary gradient methods). A few common step size schedules are:

- **constant step size:** $\alpha_k = \alpha$
- **constant step length:** $\alpha_k = \frac{\gamma}{\|g^{(k)}\|_2}$
- **square summable but not summable:**

$$\sum_{k=1}^{\infty} \alpha_k^2 < \infty, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty$$

- **nonsummable diminishing:**

$$\lim_{k\to\infty} \alpha_k = 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty$$

- **nonsummable diminishing step lengths:** $\alpha_k = \frac{\gamma_k}{\|g^{(k)}\|_2}$

$$\lim_{k\to\infty} \gamma_k = 0, \qquad \sum_{k=1}^{\infty} \gamma_k = \infty$$

Convergence Results

For constant step size and constant step length, the subgradient algorithm will converge to within some range of the optimal value,

$$\lim_{k \to \infty} f_{\text{best}}^{(k)} < f^\star + \epsilon$$

For the diminishing step size and step length rules the algorithm converges to the optimal value,

$$\lim_{k \to \infty} f_{\text{best}}^{(k)} = f^\star$$

but may take a very long time to converge.

Optimal Step Size for Known $f^\star$

Assume we know $f^\star$ (we just don't know $x^\star$). Then

$$\alpha_k = \frac{f(x^{(k)}) - f^\star}{\|g^{(k)}\|_2^2}$$

is an optimal step size in some sense. Called the Polyak step size.

A good approximation when $f^\star$ is not known (but non-negative) is

$$\alpha_k = \frac{f(x^{(k)}) - \gamma \cdot f_{\text{best}}^{(k-1)}}{\|g^{(k)}\|_2^2}$$

where $0 < \gamma < 1$.

Optimal Step Size for Known $f^\star$

Assume we know $f^\star$ (we just don't know $x^\star$). Then

$$\alpha_k = \frac{f(x^{(k)}) - f^\star}{\|g^{(k)}\|_2^2}$$

is an optimal step size in some sense. Called the Polyak step size.

A good approximation when $f^\star$ is not known (but non-negative) is

$$\alpha_k = \frac{f(x^{(k)}) - \gamma \cdot f_{\text{best}}^{(k-1)}}{\|g^{(k)}\|_2^2}$$

where $0 < \gamma < 1$.

Projected Subgradient Method

One extension of the subgradient method is the **projected subgradient method** which solves problems of the form

$$\begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C} \end{array}$$

Here the updates are

$$x^{(k+1)} = P_{\mathcal{C}}\left(x^{(k)} - \alpha_k g^{(k)}\right)$$

The projected subgradient method has similar convergence guarantees to the subgradient method.

Projected Subgradient Method

One extension of the subgradient method is the **projected subgradient method** which solves problems of the form

$$\begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C} \end{array}$$

Here the updates are

$$x^{(k+1)} = P_{\mathcal{C}}\left(x^{(k)} - \alpha_k g^{(k)}\right)$$

The projected subgradient method has similar convergence guarantees to the subgradient method.

Projected Subgradient Method

One extension of the subgradient method is the **projected subgradient method** which solves problems of the form

$$\begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C} \end{array}$$

Here the updates are

$$x^{(k+1)} = P_{\mathcal{C}}\left(x^{(k)} - \alpha_k g^{(k)}\right)$$

The projected subgradient method has similar convergence guarantees to the subgradient method.

Supergradient of $\min_i\{a_i^T x + b_i\}$

Consider $f(\mathbf{x}) = \min_i\{\mathbf{a}_i^T\mathbf{x} + b_i\}$ and let $I(\mathbf{x}) = \text{argmin}_i\{\mathbf{a}_i^T x + b_i\}$. Then for any $i \in I(\mathbf{x})$, $\mathbf{g} = \mathbf{a}_i$ is a supergradient of f at $\mathbf{x}$.

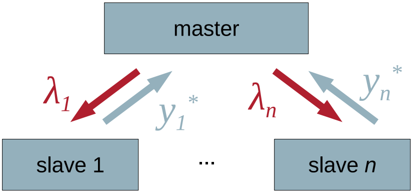
$$\begin{aligned}
f(\mathbf{x}) + \mathbf{g}^T(\mathbf{z} - \mathbf{x}) &= f(\mathbf{x}) - \mathbf{a}_i^T(\mathbf{z} - \mathbf{x}) && i \in I(\mathbf{x}) \\
&= f(\mathbf{x}) - \mathbf{a}_i^T\mathbf{x} - b_i + \mathbf{a}_i^T\mathbf{z} + b_i \\
&= \mathbf{a}_i^T\mathbf{z} + b_i \\
&\geq f(\mathbf{z})
\end{aligned}$$

Supergradient of $\min_i\{a_i^T x + b_i\}$

Consider $f(\mathbf{x}) = \min_i\{\mathbf{a}_i^T \mathbf{x} + b_i\}$ and let $I(\mathbf{x}) = \text{argmin}_i\{\mathbf{a}_i^T x + b_i\}$. Then for any $i \in I(\mathbf{x})$, $\mathbf{g} = \mathbf{a}_i$ is a supergradient of f at $\mathbf{x}$.

$$\begin{aligned} f(\mathbf{x}) + \mathbf{g}^T(\mathbf{z} - \mathbf{x}) &= f(\mathbf{x}) - \mathbf{a}_i^T(\mathbf{z} - \mathbf{x}) && i \in I(\mathbf{x}) \\ &= f(\mathbf{x}) - \mathbf{a}_i^T \mathbf{x} - b_i + \mathbf{a}_i^T \mathbf{z} + b_i \\ &= \mathbf{a}_i^T \mathbf{z} + b_i \\ &\geq f(\mathbf{z}) \end{aligned}$$

Dual Decomposition Inference [Komodakis et al., 2010]

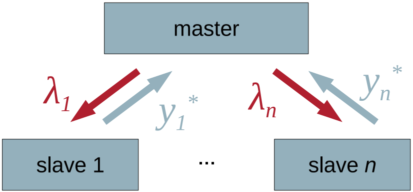


- initialize $\lambda_c = 0$
- loop
  - slaves solve $\min_{\mathbf{y}_c} \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c)$ (to get $\mu_c^\star$)
  - master updates λ_c as

$$\lambda_c \leftarrow \lambda_c + \alpha \left(\mu_c^\star - \frac{1}{C} \sum_{c'} \mu_{c'}^\star \right)$$

- until convergence

Dual Decomposition Inference [Komodakis et al., 2010]



- initialize $\lambda_c = 0$
- loop
  - slaves solve $\min_{\mathbf{y}_c} \psi_c(\mathbf{y}_c) + \lambda_c(\mathbf{y}_c)$ (to get $\boldsymbol{\mu}_c^\star$)
  - master updates λ_c as

$$\lambda_c \leftarrow \lambda_c + \alpha \left(\boldsymbol{\mu}_c^\star - \frac{1}{C} \sum_{c'} \boldsymbol{\mu}_{c'}^\star \right)$$

- until convergence

parameter learning

Max-Margin Learning

- Assume we have an energy function which is linear in its parameters, $E_{\mathbf{w}}(\mathbf{y}; \mathbf{x}) = \mathbf{w}^T \phi(\mathbf{y}; \mathbf{x})$.
- Let $\mathcal{D} = \{(\mathbf{y}_t, \mathbf{x}_t)\}_{t=1}^T$ be our set of training examples.
- Our goal in learning is to find a parameter setting $\mathbf{x}^\star$ so that for each training example $E_{\mathbf{w}}(\mathbf{y}_t; \mathbf{x}_t)$ is lower than the energy of any other assignment $E_{\mathbf{w}}(\mathbf{y}; \mathbf{x}_t)$ by some margin.
- We formalise the notion of margin by defining a loss function $\Delta(\mathbf{y}_t, \mathbf{y})$, which is zero when $\mathbf{y} = \mathbf{y}_t$ and positive otherwise.
- For simplicity let us assume we only have a single training example $(\mathbf{y}^\dagger, \mathbf{x}^\dagger)$.

Max-Margin Learning

- Assume we have an energy function which is linear in its parameters, $E_{\mathbf{w}}(\mathbf{y}; \mathbf{x}) = \mathbf{w}^T \phi(\mathbf{y}; \mathbf{x})$.
- Let $\mathcal{D} = \{(\mathbf{y}_t, \mathbf{x}_t)\}_{t=1}^T$ be our set of training examples.
- Our goal in learning is to find a parameter setting $\mathbf{x}^\star$ so that for each training example $E_{\mathbf{w}}(\mathbf{y}_t; \mathbf{x}_t)$ is lower than the energy of any other assignment $E_{\mathbf{w}}(\mathbf{y}; \mathbf{x}_t)$ by some margin.
- We formalise the notion of margin by defining a loss function $\Delta(\mathbf{y}_t, \mathbf{y})$, which is zero when $\mathbf{y} = \mathbf{y}_t$ and positive otherwise.
- For simplicity let us assume we only have a single training example $(\mathbf{y}^\dagger, \mathbf{x}^\dagger)$.

Max-Margin Learning

- Assume we have an energy function which is linear in its parameters, $E_{\mathbf{w}}(\mathbf{y}; \mathbf{x}) = \mathbf{w}^T \phi(\mathbf{y}; \mathbf{x})$.
- Let $\mathcal{D} = \{(\mathbf{y}_t, \mathbf{x}_t)\}_{t=1}^T$ be our set of training examples.
- Our goal in learning is to find a parameter setting $\mathbf{x}^\star$ so that for each training example $E_{\mathbf{w}}(\mathbf{y}_t; \mathbf{x}_t)$ is lower than the energy of any other assignment $E_{\mathbf{w}}(\mathbf{y}; \mathbf{x}_t)$ by some margin.
- We formalise the notion of margin by defining a loss function $\Delta(\mathbf{y}_t, \mathbf{y})$, which is zero when $\mathbf{y} = \mathbf{y}_t$ and positive otherwise.
- For simplicity let us assume we only have a single training example $(\mathbf{y}^\dagger, \mathbf{x}^\dagger)$.

Max-Margin Learning

- Assume we have an energy function which is linear in its parameters, $E_{\mathbf{w}}(\mathbf{y}; \mathbf{x}) = \mathbf{w}^T \phi(\mathbf{y}; \mathbf{x})$.
- Let $\mathcal{D} = \{(\mathbf{y}_t, \mathbf{x}_t)\}_{t=1}^T$ be our set of training examples.
- Our goal in learning is to find a parameter setting $\mathbf{x}^\star$ so that for each training example $E_{\mathbf{w}}(\mathbf{y}_t; \mathbf{x}_t)$ is lower than the energy of any other assignment $E_{\mathbf{w}}(\mathbf{y}; \mathbf{x}_t)$ by some margin.
- We formalise the notion of margin by defining a loss function $\Delta(\mathbf{y}_t, \mathbf{y})$, which is zero when $\mathbf{y} = \mathbf{y}_t$ and positive otherwise.
- For simplicity let us assume we only have a single training example $(\mathbf{y}^\dagger, \mathbf{x}^\dagger)$.

Max-Margin Learning

- Assume we have an energy function which is linear in its parameters, $E_{\mathbf{w}}(\mathbf{y}; \mathbf{x}) = \mathbf{w}^T \phi(\mathbf{y}; \mathbf{x})$.
- Let $\mathcal{D} = \{(\mathbf{y}_t, \mathbf{x}_t)\}_{t=1}^T$ be our set of training examples.
- Our goal in learning is to find a parameter setting $\mathbf{x}^\star$ so that for each training example $E_{\mathbf{w}}(\mathbf{y}_t; \mathbf{x}_t)$ is lower than the energy of any other assignment $E_{\mathbf{w}}(\mathbf{y}; \mathbf{x}_t)$ by some margin.
- We formalise the notion of margin by defining a loss function $\Delta(\mathbf{y}_t, \mathbf{y})$, which is zero when $\mathbf{y} = \mathbf{y}_t$ and positive otherwise.
- For simplicity let us assume we only have a single training example $(\mathbf{y}^\dagger, \mathbf{x}^\dagger)$.

Max-Margin Quadratic Program

Learning goal: Find $\mathbf{w}$ such that $E_{\mathbf{w}}(\mathbf{y}) - E_{\mathbf{w}}(\mathbf{y}^\dagger) \geq \Delta(\mathbf{y}^\dagger, \mathbf{y})$.

Relaxed and regularized learning goal:

$$\begin{array}{ll} \text{minimize} & \overbrace{\frac{1}{2}\|\mathbf{w}\|_2^2}^{\text{regularization}} + \overbrace{C\xi}^{\text{slack}} \\ \text{subject to} & \underbrace{\mathbf{w}^T\phi(\mathbf{y}) - \mathbf{w}^T\phi(\mathbf{y}^\dagger)}_{\text{energy difference}} \geq \underbrace{\Delta(\mathbf{y}, \mathbf{y}^\dagger) - \xi}_{\text{rescaled margin}}, \quad \overbrace{\forall \mathbf{y} \in \mathcal{Y}}^{\text{very large}} \\ & \xi \geq 0 \end{array}$$

Max-Margin Quadratic Program

Learning goal: Find $\mathbf{w}$ such that $E_{\mathbf{w}}(\mathbf{y}) - E_{\mathbf{w}}(\mathbf{y}^\dagger) \geq \Delta(\mathbf{y}^\dagger, \mathbf{y})$.

Relaxed and regularized learning goal:

$$\text{minimize} \quad \overbrace{\frac{1}{2}\|\mathbf{w}\|_2^2}^{\text{regularization}} + \overbrace{C\xi}^{\text{slack}}$$

$$\text{subject to} \quad \underbrace{\mathbf{w}^T\phi(\mathbf{y}) - \mathbf{w}^T\phi(\mathbf{y}^\dagger)}_{\text{energy difference}} \geq \underbrace{\Delta(\mathbf{y}, \mathbf{y}^\dagger) - \xi}_{\text{rescaled margin}}, \quad \overbrace{\forall \mathbf{y} \in \mathcal{Y}}^{\text{very large}}$$

$$\xi \geq 0$$

Re-writing Margin Constraints

Recognize that $\mathbf{w}^T\phi(\mathbf{y}) - \mathbf{w}^T\phi(\mathbf{y}^\dagger) \geq \Delta\left(\mathbf{y}, \mathbf{y}^\dagger\right) - \xi$ for all $\mathbf{y}$ so, in particular, it must hold for the worst case $\mathbf{y}$.

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\xi \\ \text{subject to} & \xi \geq \underbrace{\max_{\mathbf{y}\in\mathcal{Y}} \left\{\Delta(\mathbf{y}, \mathbf{y}^\dagger) - \mathbf{w}^T\phi(\mathbf{y})\right\}}_{\text{loss-augmented inference (for given } \mathbf{w})} + \mathbf{w}^T\phi(\mathbf{y}^\dagger) \\ & \xi \geq 0 \end{array}$$

As long as $\Delta(\mathbf{y}, \mathbf{y}_t)$ decomposes over cliques of E we can use inference to find the most violated constraint (for a fixed $\mathbf{w}$).

Re-writing Margin Constraints

Recognize that $\mathbf{w}^T\phi(\mathbf{y}) - \mathbf{w}^T\phi(\mathbf{y}^\dagger) \geq \Delta\left(\mathbf{y}, \mathbf{y}^\dagger\right) - \xi$ for all $\mathbf{y}$ so, in particular, it must hold for the worst case $\mathbf{y}$.

$$
\begin{array}{ll}
\text{minimize} & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\xi \\
\text{subject to} & \xi \geq \underbrace{\max_{\mathbf{y}\in\mathcal{Y}} \left\{\Delta(\mathbf{y}, \mathbf{y}^\dagger) - \mathbf{w}^T\phi(\mathbf{y})\right\}}_{\text{loss-augmented inference (for given } \mathbf{w})} + \mathbf{w}^T\phi(\mathbf{y}^\dagger) \\
& \xi \geq 0
\end{array}
$$

As long as $\Delta(\mathbf{y}, \mathbf{y}_t)$ decomposes over cliques of E we can use inference to find the most violated constraint (for a fixed $\mathbf{w}$).

Re-writing Margin Constraints

Recognize that $\mathbf{w}^T\phi(\mathbf{y}) - \mathbf{w}^T\phi(\mathbf{y}^\dagger) \geq \Delta\left(\mathbf{y}, \mathbf{y}^\dagger\right) - \xi$ for all $\mathbf{y}$ so, in particular, it must hold for the worst case $\mathbf{y}$.

$$
\begin{array}{ll}
\text{minimize} & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\xi \\
\text{subject to} & \xi \geq \underbrace{\max_{\mathbf{y}\in\mathcal{Y}} \left\{\Delta(\mathbf{y}, \mathbf{y}^\dagger) - \mathbf{w}^T\phi(\mathbf{y})\right\}}_{\text{loss-augmented inference (for given } \mathbf{w})} + \mathbf{w}^T\phi(\mathbf{y}^\dagger) \\
& \xi \geq 0
\end{array}
$$

As long as $\Delta(\mathbf{y}, \mathbf{y}_t)$ decomposes over cliques of E we can use inference to find the most violated constraint (for a fixed $\mathbf{w}$).

Cutting-Plane Max-Margin Learning

- Start with active set $\mathcal{A} = \{\}$.
- Solve for **w** and ξ

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\xi \\ \text{subject to} & \mathbf{w}^T\phi(\mathbf{y}) - \mathbf{w}^T\phi(\mathbf{y}^\dagger) \geq \Delta(\mathbf{y}, \mathbf{y}^\dagger) - \xi, \quad \forall \mathbf{y} \in \mathcal{A} \\ & \xi \geq 0 \end{array}$$

- Find the most violated constraint,

$$\mathbf{y}^\star \in \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} \left\{ \mathbf{w}^T\phi(\mathbf{y}) - \Delta(\mathbf{y}, \mathbf{y}^\dagger) \right\}$$

- Add $\mathbf{y}^\star$ to active set $\mathcal{A}$ and repeat.

Subgradient Descent Max-Margin Learning

Recognize that $\xi^\star = \max_{\mathbf{y} \in \mathcal{Y}} \left\{ \Delta(\mathbf{y}, \mathbf{y}^\dagger) - \mathbf{w}^T \phi(\mathbf{y}) \right\}$. So rewrite the max-margin QP as the non-smooth optimization problem

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 + C \max_{\mathbf{y} \in \mathcal{Y}} \underbrace{\left\{ \Delta(\mathbf{y}, \mathbf{y}^\dagger) - \mathbf{w}^T \phi(\mathbf{y}) \right\}}_{\text{family of linear functions}}$$

which we can solve by the subgradient method.

Tutorial Summary

- Structured prediction models, or energy functions, are pervasive in computer vision (and other fields).
- Often we are interested in finding the energy minimizing assignment.
- Exact and approximate inference algorithms exploit structure:
  - message passing for low treewidth graphs
  - graph-cuts for submodular energies
  - dual decomposition for decomposeable energies
- Parameter learning within a max-margin setting.
- Still very active research in inference and learning.

Any Questions?

stephen.gould@anu.edu.au

Tutorial Summary

- Structured prediction models, or energy functions, are pervasive in computer vision (and other fields).
- Often we are interested in finding the energy minimizing assignment.
- Exact and approximate inference algorithms exploit structure:
  - message passing for low treewidth graphs
  - graph-cuts for submodular energies
  - dual decomposition for decomposeable energies
- Parameter learning within a max-margin setting.
- Still very active research in inference and learning.

Any Questions?

stephen.gould@anu.edu.au