Machine Learning

A. G. Schwing & M. Telgarsky

University of Illinois at Urbana-Champaign, 2018

L12: Structured Prediction (ILP, LP relaxation, message passing, graph cut)

Goals of this lecture

Goals of this lecture

- Getting to know structured inference algorithms

Goals of this lecture

- Getting to know structured inference algorithms

Reading material:

Goals of this lecture

- Getting to know structured inference algorithms

Reading material:

- D. Koller and N. Friedman; Probabilistic Graphical Models: Principles and Techniques;

Recap: Inference Program

Recap: Inference Program

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$

Recap: Inference Program

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$

Algorithms:

Recap: Inference Program

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$

Algorithms:

- Exhaustive search
- Dynamic programming

Recap: Inference Program

$$y^* = \arg\max_{\hat{y}} \sum_r f_r(\hat{y}_r)$$

Algorithms:

- Exhaustive search
- Dynamic programming
- Integer linear program
- Linear programming relaxation
- Message passing
- Graph-cut

Integer Linear Program

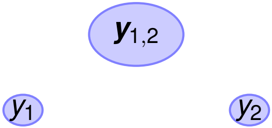
Example:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$

Integer Linear Program

Example:

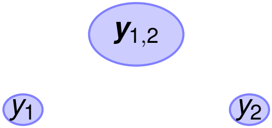
$$\mathbf{y}^* = \arg\max_{\hat{\mathbf{y}}} \sum_r f_r(\hat{\mathbf{y}}_r)$$



Integer Linear Program

Example:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$

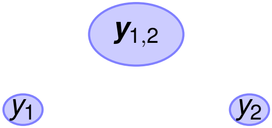


Integer Linear Program (LP) equivalence: variables $b_r(\boldsymbol{y}_r)$

Integer Linear Program

Example:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$



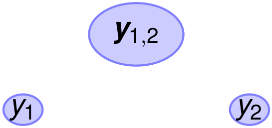
Integer Linear Program (LP) equivalence: variables $b_r(\boldsymbol{y}_r)$

$$\max_{b_1,b_2,b_{12}} \begin{bmatrix} b_1(1) \\ b_1(2) \\ b_2(1) \\ b_2(2) \\ b_{12}(1,1) \\ b_{12}(2,1) \\ b_{12}(1,2) \\ b_{12}(2,2) \end{bmatrix}^\top \begin{bmatrix} f_1(1) \\ f_1(2) \\ f_2(1) \\ f_2(2) \\ f_{12}(1,1) \\ f_{12}(2,1) \\ f_{12}(1,2) \\ f_{12}(2,2) \end{bmatrix}$$

Integer Linear Program

Example:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$



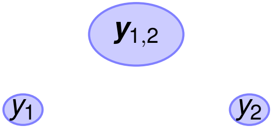
Integer Linear Program (LP) equivalence: variables $b_r(\boldsymbol{y}_r)$

$$\max_{b_1, b_2, b_{12}} \begin{bmatrix} b_1(1) \\ b_1(2) \\ b_2(1) \\ b_2(2) \\ b_{12}(1,1) \\ b_{12}(2,1) \\ b_{12}(1,2) \\ b_{12}(2,2) \end{bmatrix}^\top \begin{bmatrix} f_1(1) \\ f_1(2) \\ f_2(1) \\ f_2(2) \\ f_{12}(1,1) \\ f_{12}(2,1) \\ f_{12}(1,2) \\ f_{12}(2,2) \end{bmatrix} \quad \text{s.t.}$$

Integer Linear Program

Example:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$



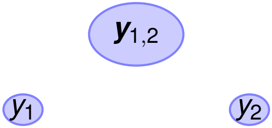
Integer Linear Program (LP) equivalence: variables $b_r(\boldsymbol{y}_r)$

$$\max_{b_1,b_2,b_{12}} \begin{bmatrix} b_1(1) \\ b_1(2) \\ b_2(1) \\ b_2(2) \\ b_{12}(1,1) \\ b_{12}(2,1) \\ b_{12}(1,2) \\ b_{12}(2,2) \end{bmatrix}^\top \begin{bmatrix} f_1(1) \\ f_1(2) \\ f_2(1) \\ f_2(2) \\ f_{12}(1,1) \\ f_{12}(2,1) \\ f_{12}(1,2) \\ f_{12}(2,2) \end{bmatrix} \text{ s.t. } \quad b_r(\boldsymbol{y}_r) \in \{0,1\} \qquad \forall r, \boldsymbol{y}_r$$

Integer Linear Program

Example:

$$\mathbf{y}^* = \arg\max_{\hat{\mathbf{y}}} \sum_r f_r(\hat{\mathbf{y}}_r)$$



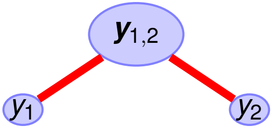
Integer Linear Program (LP) equivalence: variables $b_r(\mathbf{y}_r)$

$$\max_{b_1,b_2,b_{12}} \begin{bmatrix} b_1(1) \\ b_1(2) \\ b_2(1) \\ b_2(2) \\ b_{12}(1,1) \\ b_{12}(2,1) \\ b_{12}(1,2) \\ b_{12}(2,2) \end{bmatrix}^\top \begin{bmatrix} f_1(1) \\ f_1(2) \\ f_2(1) \\ f_2(2) \\ f_{12}(1,1) \\ f_{12}(2,1) \\ f_{12}(1,2) \\ f_{12}(2,2) \end{bmatrix} \quad \text{s.t.} \quad \begin{aligned} & b_r(\mathbf{y}_r) \in \{0,1\} && \forall r, \mathbf{y}_r \\ & \sum_{\mathbf{y}_r} b_r(\mathbf{y}_r) = 1 && \forall r \end{aligned}$$

Integer Linear Program

Example:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$



Integer Linear Program (LP) equivalence: variables $b_r(\boldsymbol{y}_r)$

$$\max_{b_1,b_2,b_{12}} \begin{bmatrix} b_1(1) \\ b_1(2) \\ b_2(1) \\ b_2(2) \\ b_{12}(1,1) \\ b_{12}(2,1) \\ b_{12}(1,2) \\ b_{12}(2,2) \end{bmatrix}^\top \begin{bmatrix} f_1(1) \\ f_1(2) \\ f_2(1) \\ f_2(2) \\ f_{12}(1,1) \\ f_{12}(2,1) \\ f_{12}(1,2) \\ f_{12}(2,2) \end{bmatrix} \quad \text{s.t.} \quad \begin{array}{lr} b_r(\boldsymbol{y}_r) \in \{0,1\} & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \\ \sum_{\boldsymbol{y}_p \setminus \boldsymbol{y}_r} b_p(\boldsymbol{y}_p) = b_r(\boldsymbol{y}_r) & \forall r, \boldsymbol{y}_r, p \in P(r) \end{array}$$

Integer Linear Program

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$

Integer linear program:

$$\max_{b_1,b_2,b_{12}} \begin{bmatrix} b_1(1) \\ b_1(2) \\ b_2(1) \\ b_2(2) \\ b_{12}(1,1) \\ b_{12}(2,1) \\ b_{12}(1,2) \\ b_{12}(2,2) \end{bmatrix}^\top \begin{bmatrix} f_1(1) \\ f_1(2) \\ f_2(1) \\ f_2(2) \\ f_{12}(1,1) \\ f_{12}(2,1) \\ f_{12}(1,2) \\ f_{12}(2,2) \end{bmatrix} \quad \text{s.t.} \quad \begin{array}{ll} b_r(\boldsymbol{y}_r) \in \{0,1\} & \forall r, \boldsymbol{y}_r \\ b_r(\boldsymbol{y}_r) \geq 0 & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \\ \sum_{\boldsymbol{y}_p \setminus \boldsymbol{y}_r} b_p(\boldsymbol{y}_p) = b_r(\boldsymbol{y}_r) & \end{array}$$

Integer Linear Program

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$

Integer linear program:

$$\max_{b_r} \quad \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) \quad \text{s.t.} \quad \begin{array}{ll} b_r(\boldsymbol{y}_r) \in \{0,1\} & \forall r, \boldsymbol{y}_r \\ b_r(\boldsymbol{y}_r) \geq 0 & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \\ \sum_{\boldsymbol{y}_p \backslash \boldsymbol{y}_r} b_p(\boldsymbol{y}_p) = b_r(\boldsymbol{y}_r) & \end{array}$$

Integer Linear Program

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$

Integer linear program:

$$\max_{b_r} \quad \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) \quad \text{s.t.} \quad \begin{array}{ll} b_r(\boldsymbol{y}_r) \in \{0,1\} & \forall r, \boldsymbol{y}_r \\ b_r(\boldsymbol{y}_r) \geq 0 & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \end{array}$$

Marginalization

Integer Linear Program

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$

Integer linear program:

$$\max_{b_r} \quad \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) \quad \text{s.t.} \quad \begin{array}{ll} b_r(\boldsymbol{y}_r) \in \{0,1\} & \forall r, \boldsymbol{y}_r \\ \text{Local probability } b_r & \\ \text{Marginalization} & \end{array}$$

Integer Linear Program

$$y^* = \arg\max_{\hat{y}} \sum_r f_r(\hat{y}_r)$$

Integer linear program:

$$\max_{b_r} \quad \sum_{r,y_r} b_r(y_r) f_r(y_r) \quad \text{s.t.} \quad \begin{array}{l} b_r(y_r) \in \{0,1\} \qquad \forall r, y_r \\ \text{Local probability } b_r \\ \text{Marginalization} \end{array}$$

- **Advantage:** global optimum, very good solvers available
- **Disadvantage:** very slow for larger problems

Linear Programming Relaxation

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$

LP relaxation:

$$\cancel{b_r(\boldsymbol{y}_r) \in \{0,1\} \quad \forall r, \boldsymbol{y}_r}$$

$$\max_{b_r} \quad \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) \quad \underbrace{\text{s.t.} \quad \begin{array}{l} \text{Local probability } b_r \\ \text{Marginalization} \end{array}}_{\text{s.t.} \quad b \in \mathcal{C}}$$

Linear Programming Relaxation

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\hat{\boldsymbol{y}}_r)$$

LP relaxation:

$$\cancel{b_r(\boldsymbol{y}_r) \in \{0,1\} \quad \forall r, \boldsymbol{y}_r}$$

$$\max_{b_r} \quad \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) \quad \underbrace{\text{s.t.} \quad \begin{array}{l}\text{Local probability } b_r \\ \text{Marginalization}\end{array}}_{\text{s.t.} \quad b \in \mathcal{C}}$$

- **Advantage:** global optimum for LP, very good solvers available
- **Disadvantage:** no global optimum for ILP, slow for larger problems

Message Passing ([Loopy] Belief Propagation)

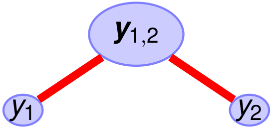
Exploit:

Message Passing ([Loopy] Belief Propagation)

Exploit: Graph structure defined via marginalization constraints

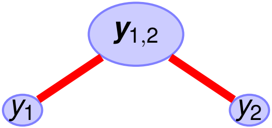
Message Passing ([Loopy] Belief Propagation)

Exploit: Graph structure defined via marginalization constraints



Message Passing ([Loopy] Belief Propagation)

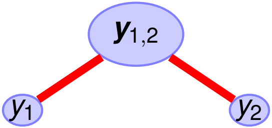
Exploit: Graph structure defined via marginalization constraints



How:

Message Passing ([Loopy] Belief Propagation)

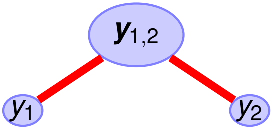
Exploit: Graph structure defined via marginalization constraints



How: Compute the dual function

Message Passing ([Loopy] Belief Propagation)

Exploit: Graph structure defined via marginalization constraints



How: Compute the dual function

Message passing solvers:

- **Advantage:** Efficient due to analytically computable sub-problems
- **Disadvantage:** Special care required to find LP relaxation optimum

Computing the dual of

$$\max_b \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) \quad \text{s.t.} \quad \left\{ \begin{array}{ll} b_r(\boldsymbol{y}_r) \geq 0 & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \\ \sum_{\boldsymbol{y}_p \setminus \boldsymbol{y}_r} b_p(\boldsymbol{y}_p) = b_r(\boldsymbol{y}_r) & \forall r, p \in P(r), \boldsymbol{y}_r \end{array} \right.$$

Computing the dual of

$$\max_{b} \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) \quad \text{s.t.} \quad \left\{ \begin{array}{ll} b_r(\boldsymbol{y}_r) \geq 0 & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \\ \sum_{\boldsymbol{y}_p \setminus \boldsymbol{y}_r} b_p(\boldsymbol{y}_p) = b_r(\boldsymbol{y}_r) & \forall r, p \in P(r), \boldsymbol{y}_r \end{array} \right.$$

Lagrangian:

Computing the dual of

$$\max_b \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) \quad \text{s.t.} \quad \left\{ \begin{array}{ll} b_r(\boldsymbol{y}_r) \geq 0 & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \\ \sum_{\boldsymbol{y}_p \backslash \boldsymbol{y}_r} b_p(\boldsymbol{y}_p) = b_r(\boldsymbol{y}_r) & \forall r, p \in P(r), \boldsymbol{y}_r \end{array} \right.$$

Lagrangian:

$$\begin{aligned} L() &= \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) + \sum_{r,p \in P(r),\boldsymbol{y}_r} \lambda_{r\to p}(\boldsymbol{y}_r) \left(\sum_{\boldsymbol{y}_p \backslash \boldsymbol{y}_r} b_p(\boldsymbol{y}_p) - b_r(\boldsymbol{y}_r) \right) \\ &= \end{aligned}$$

Computing the dual of

$$\max_b \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) \quad \text{s.t.} \quad \left\{ \begin{array}{ll} b_r(\boldsymbol{y}_r) \geq 0 & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \\ \sum_{\boldsymbol{y}_p \setminus \boldsymbol{y}_r} b_p(\boldsymbol{y}_p) = b_r(\boldsymbol{y}_r) & \forall r, p \in P(r), \boldsymbol{y}_r \end{array} \right.$$

Lagrangian:

$$\begin{aligned} L() &= \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) + \sum_{r,p\in P(r),\boldsymbol{y}_r} \lambda_{r\to p}(\boldsymbol{y}_r) \left(\sum_{\boldsymbol{y}_p \setminus \boldsymbol{y}_r} b_p(\boldsymbol{y}_p) - b_r(\boldsymbol{y}_r) \right) \\ &= \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) \left(f_r(\boldsymbol{y}_r) - \sum_{p\in P(r)} \lambda_{r\to p}(\boldsymbol{y}_r) + \sum_{c\in C(r)} \lambda_{c\to r}(\boldsymbol{y}_c) \right) \end{aligned}$$

Lagrangian:

$$L() = \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r)\left(f_r(\boldsymbol{y}_r) - \sum_{p\in P(r)} \lambda_{r\to p}(\boldsymbol{y}_r) + \sum_{c\in C(r)} \lambda_{c\to r}(\boldsymbol{y}_c)\right)$$

Lagrangian:

$$L() \;=\; \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r)\left(f_r(\boldsymbol{y}_r) - \sum_{p\in P(r)} \lambda_{r\to p}(\boldsymbol{y}_r) + \sum_{c\in C(r)} \lambda_{c\to r}(\boldsymbol{y}_c)\right)$$

Maximize Lagrangian w.r.t. primal variables subject to remaining constraints:

Lagrangian:

$$L() = \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r)\left(f_r(\boldsymbol{y}_r) - \sum_{p\in P(r)} \lambda_{r\to p}(\boldsymbol{y}_r) + \sum_{c\in C(r)} \lambda_{c\to r}(\boldsymbol{y}_c)\right)$$

Maximize Lagrangian w.r.t. primal variables subject to remaining constraints:

$$\max_b L() \quad \text{s.t.} \quad \begin{cases} b_r(\boldsymbol{y}_r) \geq 0 & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \end{cases}$$

Lagrangian:

$$L() = \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r)\left(f_r(\boldsymbol{y}_r) - \sum_{p\in P(r)} \lambda_{r\to p}(\boldsymbol{y}_r) + \sum_{c\in C(r)} \lambda_{c\to r}(\boldsymbol{y}_c)\right)$$

Maximize Lagrangian w.r.t. primal variables subject to remaining constraints:

$$\max_b L() \quad \text{s.t.} \quad \begin{cases} b_r(\boldsymbol{y}_r) \geq 0 & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \end{cases}$$

Dual function:

Lagrangian:

$$L() = \sum_{r,\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) \left(f_r(\boldsymbol{y}_r) - \sum_{p \in P(r)} \lambda_{r \to p}(\boldsymbol{y}_r) + \sum_{c \in C(r)} \lambda_{c \to r}(\boldsymbol{y}_c) \right)$$

Maximize Lagrangian w.r.t. primal variables subject to remaining constraints:

$$\max_b L() \quad \text{s.t.} \quad \begin{cases} b_r(\boldsymbol{y}_r) \geq 0 & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \end{cases}$$

Dual function:

$$g(\lambda) = \sum_r \max_{\boldsymbol{y}_r} \left(f_r(\boldsymbol{y}_r) - \sum_{p \in P(r)} \lambda_{r \to p}(\boldsymbol{y}_r) + \sum_{c \in C(r)} \lambda_{c \to r}(\boldsymbol{y}_c) \right)$$

Dual function:

$$g(\lambda) = \sum_r \max_{\boldsymbol{y}_r} \left(f_r(\boldsymbol{y}_r) - \sum_{p \in P(r)} \lambda_{r \to p}(\boldsymbol{y}_r) + \sum_{c \in C(r)} \lambda_{c \to r}(\boldsymbol{y}_c) \right)$$

Dual function:

$$g(\lambda) = \sum_r \max_{\boldsymbol{y}_r} \left(f_r(\boldsymbol{y}_r) - \sum_{p \in P(r)} \lambda_{r \to p}(\boldsymbol{y}_r) + \sum_{c \in C(r)} \lambda_{c \to r}(\boldsymbol{y}_c) \right)$$

Dual program:

Dual function:

$$g(\lambda) = \sum_r \max_{\boldsymbol{y}_r} \left(f_r(\boldsymbol{y}_r) - \sum_{p \in P(r)} \lambda_{r \to p}(\boldsymbol{y}_r) + \sum_{c \in C(r)} \lambda_{c \to r}(\boldsymbol{y}_c) \right)$$

Dual program:

$$\min_\lambda g(\lambda)$$

Dual function:

$$g(\lambda) = \sum_r \max_{\boldsymbol{y}_r} \left(f_r(\boldsymbol{y}_r) - \sum_{p \in P(r)} \lambda_{r \to p}(\boldsymbol{y}_r) + \sum_{c \in C(r)} \lambda_{c \to r}(\boldsymbol{y}_c) \right)$$

Dual program:

$$\min_{\lambda} g(\lambda)$$

Original primal:

Dual function:

$$g(\lambda) = \sum_r \max_{\boldsymbol{y}_r} \left(f_r(\boldsymbol{y}_r) - \sum_{p \in P(r)} \lambda_{r \to p}(\boldsymbol{y}_r) + \sum_{c \in C(r)} \lambda_{c \to r}(\boldsymbol{y}_c) \right)$$

Dual program:

$$\min_\lambda g(\lambda)$$

Original primal:

$$\max_b \sum_{r, \boldsymbol{y}_r} b_r(\boldsymbol{y}_r) f_r(\boldsymbol{y}_r) \quad \text{s.t.} \quad \left\{ \begin{array}{ll} b_r(\boldsymbol{y}_r) \geq 0 & \forall r, \boldsymbol{y}_r \\ \sum_{\boldsymbol{y}_r} b_r(\boldsymbol{y}_r) = 1 & \forall r \\ \sum_{\boldsymbol{y}_p \setminus \boldsymbol{y}_r} b_p(\boldsymbol{y}_p) = b_r(\boldsymbol{y}_r) & \forall r, p \in P(r), \boldsymbol{y}_r \end{array} \right.$$

Properties of dual program:

Properties of dual program:

- Convex program

Properties of dual program:

- Convex program
- Not strongly convex

Properties of dual program:

- Convex program
- Not strongly convex
- Piecewise linear

Properties of dual program:

- Convex program
- Not strongly convex
- Piecewise linear
- **Unconstrained**

Properties of dual program:

- Convex program
- Not strongly convex
- Piecewise linear
- **Unconstrained**
- Lagrange multipliers are messages

Properties of dual program:

- Convex program
- Not strongly convex
- Piecewise linear
- **Unconstrained**
- Lagrange multipliers are messages

Lagrange multipliers are messages defined on edges of the graph. They shift 'energy' such that local maximization (dual) is identical to global maximization (primal).

Graph-cut Solvers

Graph-cut Solvers

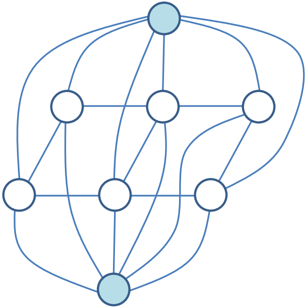
- Efficient algorithms to compute the minimum cost cut in a weighted graph

Graph-cut Solvers

- Efficient algorithms to compute the minimum cost cut in a weighted graph
- Efficient algorithms to compute the maximum flow through a weighted graph

Graph-cut Solvers

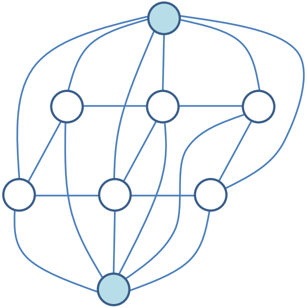
- Efficient algorithms to compute the minimum cost cut in a weighted graph
- Efficient algorithms to compute the maximum flow through a weighted graph



Graph-cut Solvers

For binary problems $y_d \in \{1, 2\}$:

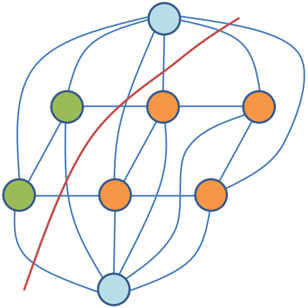
- Convert scoring function F into auxiliary graph (not the same graph as before!)



Graph-cut Solvers

For binary problems $y_d \in \{1, 2\}$:

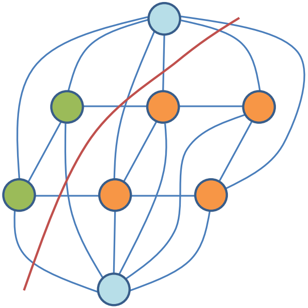
- Convert scoring function F into auxiliary graph (not the same graph as before!)
- Compute a weighted cut cost corresponding to the labeling score



Graph-cut Solvers

For binary problems $y_d \in \{1, 2\}$:

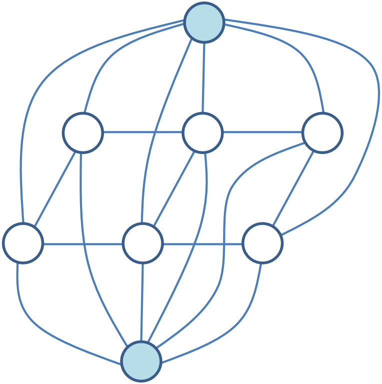
- Convert scoring function F into auxiliary graph (not the same graph as before!)
- Compute a weighted cut cost corresponding to the labeling score



What are the nodes and what are the weights on the edges in this auxiliary graph?

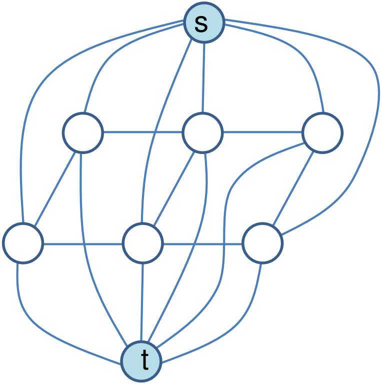
Graph-cut Solvers

What are the nodes in the auxiliary graph?



Graph-cut Solvers

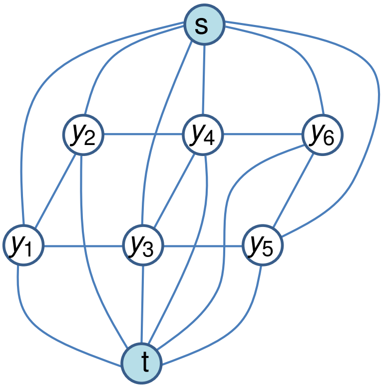
What are the nodes in the auxiliary graph?



- Two special nodes called 'source' and 'terminal'

Graph-cut Solvers

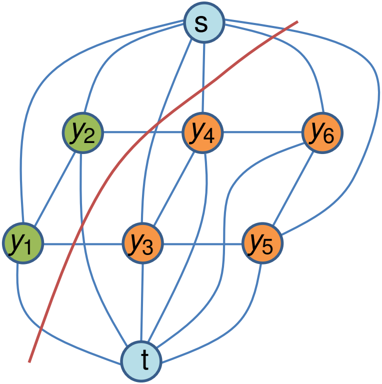
What are the nodes in the auxiliary graph?



- Two special nodes called 'source' and 'terminal'
- Variables y_d as nodes

Graph-cut Solvers

What are the nodes in the auxiliary graph?



- Two special nodes called ‘source’ and ‘terminal’
- Variables y_d as nodes

Graph-cut Solvers

What weights do we assign to edges?

Graph-cut Solvers

What weights do we assign to edges?
Recall that local scoring functions are arrays:

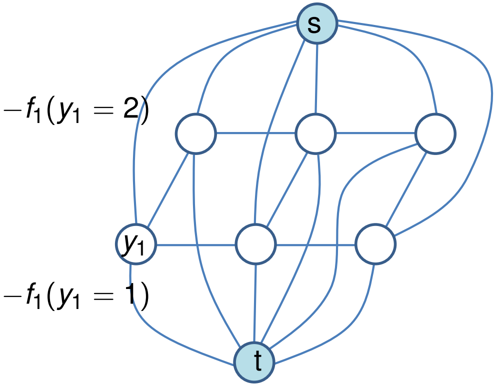
$$\begin{bmatrix} f_1(y_1 = 1) & f_1(y_1 = 2) \end{bmatrix}$$

Graph-cut Solvers

What weights do we assign to edges?
Recall that local scoring functions are arrays:

$$\left[\begin{array}{cc} f_1(y_1 = 1) & f_1(y_1 = 2) \end{array}\right]$$

Graph-cut solvers compute a min-cut:



Graph-cut Solvers

What weights do we assign to edges?
Recall that local scoring functions are arrays:

$$\begin{bmatrix} f_{12}(1,1) & f_{12}(1,2) \\ f_{12}(2,1) & f_{12}(2,2) \end{bmatrix} =$$

Graph-cut Solvers

What weights do we assign to edges?
Recall that local scoring functions are arrays:

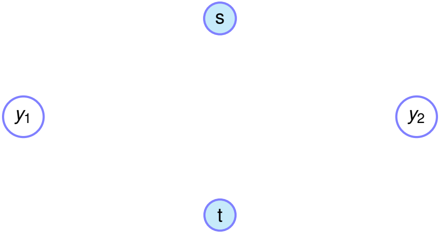
$$
\begin{aligned}
\begin{bmatrix} f_{12}(1,1) & f_{12}(1,2) \\ f_{12}(2,1) & f_{12}(2,2) \end{bmatrix} &= f(1,1) - f(2,1) + f(2,2) \\
&+ \begin{bmatrix} 0 & 0 \\ f(2,1) - f(1,1) & f(2,1) - f(1,1) \end{bmatrix} \\
&+ \begin{bmatrix} f(2,1) - f(2,2) & 0 \\ f(2,1) - f(2,2) & 0 \end{bmatrix} \\
&+ \begin{bmatrix} 0 & f(1,2) + f(2,1) - f(1,1) - f(2,2) \\ 0 & 0 \end{bmatrix}
\end{aligned}
$$

Graph-cut Solvers

What weights do we assign to edges?
Recall that local scoring functions are arrays:

$$\begin{aligned}
\begin{bmatrix} f_{12}(1,1) & f_{12}(1,2) \\ f_{12}(2,1) & f_{12}(2,2) \end{bmatrix} &= f(1,1) - f(2,1) + f(2,2) \\
&+ \begin{bmatrix} 0 & 0 \\ f(2,1) - f(1,1) & f(2,1) - f(1,1) \end{bmatrix} \\
&+ \begin{bmatrix} f(2,1) - f(2,2) & 0 \\ f(2,1) - f(2,2) & 0 \end{bmatrix} \\
&+ \begin{bmatrix} 0 & f(1,2) + f(2,1) - f(1,1) - f(2,2) \\ 0 & 0 \end{bmatrix}
\end{aligned}$$

Graph-cut solvers compute a min-cut:

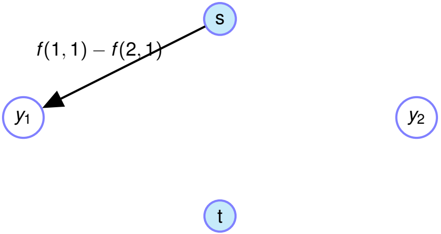


Graph-cut Solvers

What weights do we assign to edges?
Recall that local scoring functions are arrays:

$$\begin{bmatrix} f_{12}(1,1) & f_{12}(1,2) \\ f_{12}(2,1) & f_{12}(2,2) \end{bmatrix} = f(1,1) - f(2,1) + f(2,2)$$
$$+ \begin{bmatrix} 0 & 0 \\ f(2,1) - f(1,1) & f(2,1) - f(1,1) \end{bmatrix}$$
$$+ \begin{bmatrix} f(2,1) - f(2,2) & 0 \\ f(2,1) - f(2,2) & 0 \end{bmatrix}$$
$$+ \begin{bmatrix} 0 & f(1,2) + f(2,1) - f(1,1) - f(2,2) \\ 0 & 0 \end{bmatrix}$$

Graph-cut solvers compute a min-cut:

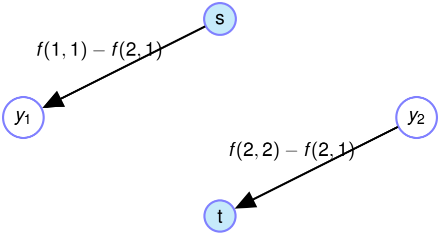


Graph-cut Solvers

What weights do we assign to edges?
Recall that local scoring functions are arrays:

$$\begin{aligned}\begin{bmatrix} f_{12}(1,1) & f_{12}(1,2) \\ f_{12}(2,1) & f_{12}(2,2) \end{bmatrix} &= f(1,1) - f(2,1) + f(2,2) \\ &+ \begin{bmatrix} 0 & 0 \\ f(2,1) - f(1,1) & f(2,1) - f(1,1) \end{bmatrix} \\ &+ \begin{bmatrix} f(2,1) - f(2,2) & 0 \\ f(2,1) - f(2,2) & 0 \end{bmatrix} \\ &+ \begin{bmatrix} 0 & f(1,2) + f(2,1) - f(1,1) - f(2,2) \\ 0 & 0 \end{bmatrix}\end{aligned}$$

Graph-cut solvers compute a min-cut:

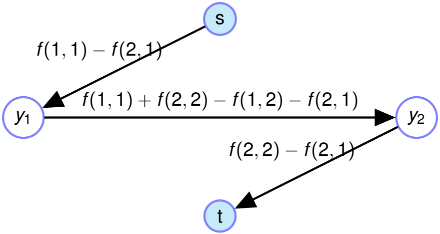


Graph-cut Solvers

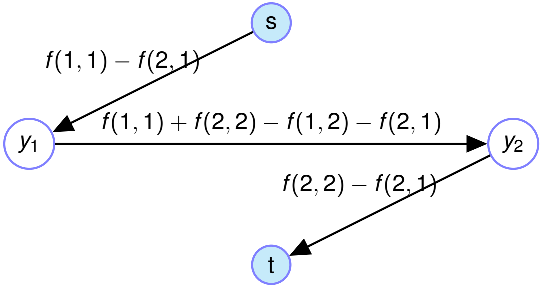
What weights do we assign to edges?
Recall that local scoring functions are arrays:

$$
\begin{aligned}
\begin{bmatrix} f_{12}(1,1) & f_{12}(1,2) \\ f_{12}(2,1) & f_{12}(2,2) \end{bmatrix} &= f(1,1) - f(2,1) + f(2,2) \\
&+ \begin{bmatrix} 0 & 0 \\ f(2,1) - f(1,1) & f(2,1) - f(1,1) \end{bmatrix} \\
&+ \begin{bmatrix} f(2,1) - f(2,2) & 0 \\ f(2,1) - f(2,2) & 0 \end{bmatrix} \\
&+ \begin{bmatrix} 0 & f(1,2) + f(2,1) - f(1,1) - f(2,2) \\ 0 & 0 \end{bmatrix}
\end{aligned}
$$

Graph-cut solvers compute a min-cut:

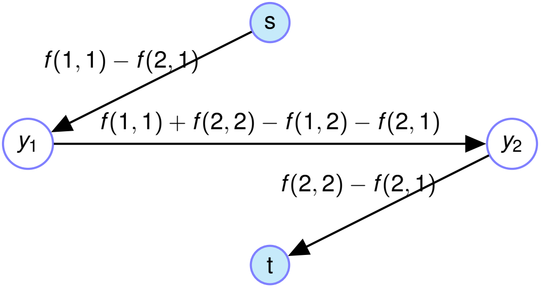


Graph-cut Solvers



Requirement for optimality:

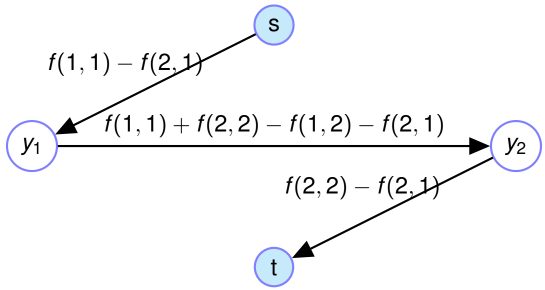
Graph-cut Solvers



Requirement for optimality: Pairwise edge weights are positive

$$f(1,1) + f(2,2) - f(1,2) - f(2,1) \geq 0 \quad \text{sub-modularity}$$

Graph-cut Solvers

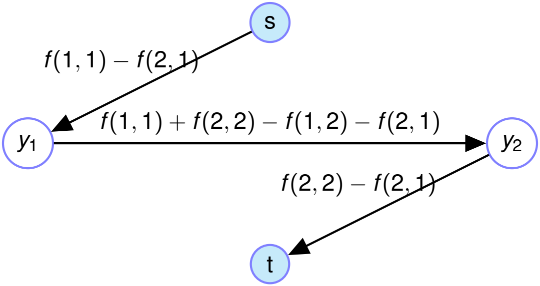


Requirement for optimality: Pairwise edge weights are positive

$$f(1,1) + f(2,2) - f(1,2) - f(2,1) \geq 0 \quad \text{sub-modularity}$$

For higher order functions?

Graph-cut Solvers

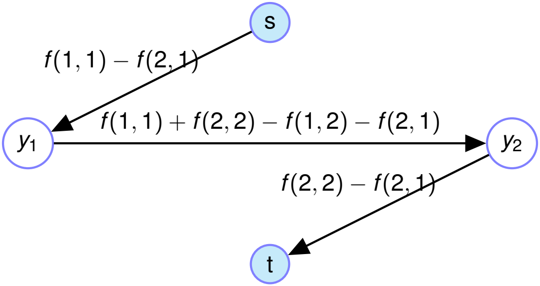


Requirement for optimality: Pairwise edge weights are positive

$$f(1,1) + f(2,2) - f(1,2) - f(2,1) \geq 0 \quad \text{sub-modularity}$$

For higher order functions? More complicated graph constructions

Graph-cut Solvers

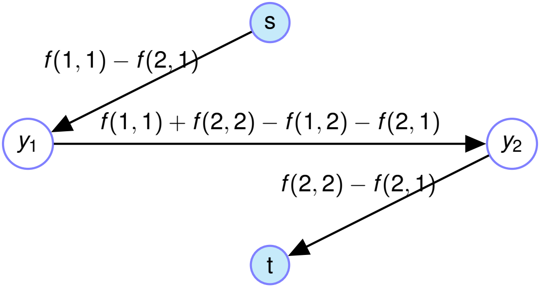


Requirement for optimality: Pairwise edge weights are positive

$$f(1,1) + f(2,2) - f(1,2) - f(2,1) \geq 0 \quad \text{sub-modularity}$$

For higher order functions? More complicated graph constructions
For more than two labels?

Graph-cut Solvers



Requirement for optimality: Pairwise edge weights are positive

$$f(1,1) + f(2,2) - f(1,2) - f(2,1) \geq 0 \quad \text{sub-modularity}$$

For higher order functions? More complicated graph constructions
For more than two labels? Move making algorithms

Structured Prediction

Inference:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\boldsymbol{w}, x, \hat{\boldsymbol{y}}_r)$$

Efficiency and accuracy of inference algorithms is problem dependent:

Structured Prediction

Inference:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\boldsymbol{w}, x, \hat{\boldsymbol{y}}_r)$$

Efficiency and accuracy of inference algorithms is problem dependent:

- Exhaustive search

Structured Prediction

Inference:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\boldsymbol{w}, x, \hat{\boldsymbol{y}}_r)$$

Efficiency and accuracy of inference algorithms is problem dependent:

- Exhaustive search
- Dynamic programming

Structured Prediction

Inference:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\boldsymbol{w}, x, \hat{\boldsymbol{y}}_r)$$

Efficiency and accuracy of inference algorithms is problem dependent:

- Exhaustive search
- Dynamic programming
- Integer linear program

Structured Prediction

Inference:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\boldsymbol{w}, x, \hat{\boldsymbol{y}}_r)$$

Efficiency and accuracy of inference algorithms is problem dependent:

- Exhaustive search
- Dynamic programming
- Integer linear program
- Linear programming relaxation

Structured Prediction

Inference:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\boldsymbol{w}, x, \hat{\boldsymbol{y}}_r)$$

Efficiency and accuracy of inference algorithms is problem dependent:

- Exhaustive search
- Dynamic programming
- Integer linear program
- Linear programming relaxation
- Message passing

Structured Prediction

Inference:

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}} \sum_r f_r(\boldsymbol{w}, x, \hat{\boldsymbol{y}}_r)$$

Efficiency and accuracy of inference algorithms is problem dependent:

- Exhaustive search
- Dynamic programming
- Integer linear program
- Linear programming relaxation
- Message passing
- Graph-cut

Quiz:

Quiz:

- What is the advantage of ILP and LP relaxation compared to dynamic programming and exhaustive search?

Quiz:

- What is the advantage of ILP and LP relaxation compared to dynamic programming and exhaustive search?
- When is a graph-cut algorithm optimal?

Important topics of this lecture

- More inference algorithms for structured spaces

Up next:

- Learning models for structured output spaces