Linear Programming Relaxations for Graphical Models - Part II

Tommi Jaakkola (MIT) , Amir Globerson (Hebrew Univ.)

Based on joint work with
M. Collins, M. Fromer, T. Koo,
M. Meila, O. Meshi, A. Rush, D. Sontag

Outline

- Solving LP relaxations:
  - Subgradient algorithms
  - Coordinate descent algorithms
- Parameter learning with LP relaxations:
  - Structured Prediction
  - Structured LP perceptron.
  - Coordinate descent algorithms.
  - Cutting plane algorithms.
  - Pseudo-max algorithms.

Recap

Recap

- Want to solve: $\max_y \sum_{f \in F} \theta_f(y_f) + \sum_i \theta_i(y_i)$

Recap

- Want to solve: $\max_y \sum_{f \in F} \theta_f(y_f) + \sum_i \theta_i(y_i)$
- Approximate with a linear program:

Recap

- Want to solve: $\max_y \sum_{f \in F} \theta_f(y_f) + \sum_i \theta_i(y_i)$
- Approximate with a linear program:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \sum_f \theta_f(y_f)\mu_f(y_f) + \sum_i \theta_i(y_i)\mu_i(y_i)$$

Recap

- Want to solve: $\max_y \sum_{f \in F} \theta_f(y_f) + \sum_i \theta_i(y_i)$
- Approximate with a linear program:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \sum_f \theta_f(y_f)\mu_f(y_f) + \sum_i \theta_i(y_i)\mu_i(y_i)$$

$$\mathcal{M}_L = \left\{ \boldsymbol{\mu}: \begin{array}{l} \mu_i(y_i) \geq 0 \\ \mu_f(y_f) \geq 0 \\ \sum_{y_i} \mu_i(y_i) = 1 \\ \sum_{y_{f \setminus i}} \mu_f(y_f) = \mu_i(y_i) \ , \ \forall i \in f, y_i \end{array} \right\}$$

Recap

- Want to solve: $\max_y \sum_{f \in F} \theta_f(y_f) + \sum_i \theta_i(y_i)$
- Approximate with a linear program:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta} \cdot \boldsymbol{\mu}$$

$$\mathcal{M}_L = \left\{ \boldsymbol{\mu} : \begin{array}{l} \mu_i(y_i) \geq 0 \\ \mu_f(y_f) \geq 0 \\ \sum_{y_i} \mu_i(y_i) = 1 \\ \sum_{y_{f \setminus i}} \mu_f(y_f) = \mu_i(y_i) \ , \ \forall i \in f, y_i \end{array} \right\}$$

Recap

- Want to solve: $\max_y \sum_{f \in F} \theta_f(y_f) + \sum_i \theta_i(y_i)$
- Approximate with a linear program:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta} \cdot \boldsymbol{\mu}$$

Recap

- Want to solve: $\max_y \sum_{f \in F} \theta_f(y_f) + \sum_i \theta_i(y_i)$
- Approximate with a linear program:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta} \cdot \boldsymbol{\mu}$$

The Primal

Recap

- Want to solve: $\max_y \sum_{f \in F} \theta_f(y_f) + \sum_i \theta_i(y_i)$
- Approximate with a linear program:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta} \cdot \boldsymbol{\mu}$$

The Primal

- Equivalent to the following dual:

Recap

- Want to solve: $\max_y \sum_{f \in F} \theta_f(y_f) + \sum_i \theta_i(y_i)$
- Approximate with a linear program:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta} \cdot \boldsymbol{\mu}$$ The Primal

- Equivalent to the following dual:

$$\min_{\boldsymbol{\delta}} g(\boldsymbol{\delta})$$

Recap

- Want to solve: $\max_y \sum_{f \in F} \theta_f(y_f) + \sum_i \theta_i(y_i)$
- Approximate with a linear program:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta} \cdot \boldsymbol{\mu}$$

The Primal

- Equivalent to the following dual:

$$\min_{\boldsymbol{\delta}} g(\boldsymbol{\delta})$$

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right] + \sum_i \max_{y_i} \left[\theta_i(y_i) + \sum_{f : i \in f} \delta_{fi}(y_i) \right]$$

Optimizing LP Relaxations

Optimizing LP Relaxations

- The LP relaxation is an optimization problem.

Optimizing LP Relaxations

- The LP relaxation is an optimization problem.
- Both primal and dual can be cast as LPs

Optimizing LP Relaxations

- The LP relaxation is an optimization problem.
- Both primal and dual can be cast as LPs
- Can be solved in polynomial time using general purpose solvers (e.g., CPLEX, MOSEK, GLPK etc).

Optimizing LP Relaxations

- The LP relaxation is an optimization problem.
- Both primal and dual can be cast as LPs
- Can be solved in polynomial time using general purpose solvers (e.g., CPLEX, MOSEK, GLPK etc).
- Often do not scale well, since they do not use problem specific structure *[Yanover et al., 05]*.

Optimizing LP Relaxations

- The LP relaxation is an optimization problem.
- Both primal and dual can be cast as LPs
- Can be solved in polynomial time using general purpose solvers (e.g., CPLEX, MOSEK, GLPK etc).
- Often do not scale well, since they do not use problem specific structure *[Yanover et al., 05]*.
- Much work on practical algorithms for large scale problems.

Optimizing LP Relaxations

- The LP relaxation is an optimization problem.
- Both primal and dual can be cast as LPs
- Can be solved in polynomial time using general purpose solvers (e.g., CPLEX, MOSEK, GLPK etc).
- Often do not scale well, since they do not use problem specific structure *[Yanover et al., 05]*.
- Much work on practical algorithms for large scale problems.
- Many of these can handle complex (global) factors.

Optimizing LP Relaxations

- The LP relaxation is an optimization problem.
- Both primal and dual can be cast as LPs
- Can be solved in polynomial time using general purpose solvers (e.g., CPLEX, MOSEK, GLPK etc).
- Often do not scale well, since they do not use problem specific structure *[Yanover et al., 05]*.
- Much work on practical algorithms for large scale problems.
- Many of these can handle complex (global) factors.
- Often solve via the dual.

Decoding a dual solution

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

Decoding a dual solution

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- How do we obtain a solution for y from δ ?

Decoding a dual solution

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- How do we obtain a solution for y from δ ?
- What can we say about it?

Decoding a dual solution

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- How do we obtain a solution for y from $\boldsymbol{\delta}$?
- What can we say about it?

- Say we found an assignment $\hat{y} = [\hat{y}_1, \ldots, \hat{y}_n]$ **s.t.:**

Decoding a dual solution

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- How do we obtain a solution for y from $\boldsymbol{\delta}$?
- What can we say about it?

- Say we found an assignment $\hat{y} = [\hat{y}_1, \ldots, \hat{y}_n]$ **s.t.:**

$$\hat{y}_i \in \arg\max_{y_i} \bar{\theta}^{\delta}_i(y_i) \quad \textit{and} \quad \hat{y}_f \in \arg\max_{y_f} \bar{\theta}^{\delta}_f(y_f)$$

Decoding a dual solution

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- How do we obtain a solution for y from $\boldsymbol{\delta}$?
- What can we say about it?

- Say we found an assignment $\hat{y} = [\hat{y}_1, \ldots, \hat{y}_n]$ **s.t.:**

$$\hat{y}_i \in \arg\max_{y_i} \bar{\theta}^{\delta}_i(y_i) \quad \textit{and} \quad \hat{y}_f \in \arg\max_{y_f} \bar{\theta}^{\delta}_f(y_f)$$

$\hat{y}$ maximizes all factors

Decoding a dual solution

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- How do we obtain a solution for y from $\boldsymbol{\delta}$?
- What can we say about it?

- Say we found an assignment $\hat{y} = [\hat{y}_1, \ldots, \hat{y}_n]$ **s.t.:**

$$\hat{y}_i \in \arg\max_{y_i} \bar{\theta}^{\delta}_i(y_i) \quad \textit{and} \quad \hat{y}_f \in \arg\max_{y_f} \bar{\theta}^{\delta}_f(y_f)$$

Decoding a dual solution

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f: i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- How do we obtain a solution for y from $\boldsymbol{\delta}$?
- What can we say about it?

- Say we found an assignment $\hat{y} = [\hat{y}_1, \ldots, \hat{y}_n]$ **s.t.:**

$$\hat{y}_i \in \arg\max_{y_i} \bar{\theta}^{\delta}_i(y_i) \quad \textit{and} \quad \hat{y}_f \in \arg\max_{y_f} \bar{\theta}^{\delta}_f(y_f)$$

- Then $\hat{y}$ is optimal: It maximizes $\sum_f \theta_f(y_f) + \sum_i \theta_i(y_i)$.

Decoding a dual solution

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- How do we obtain a solution for y from $\boldsymbol{\delta}$?
- What can we say about it?

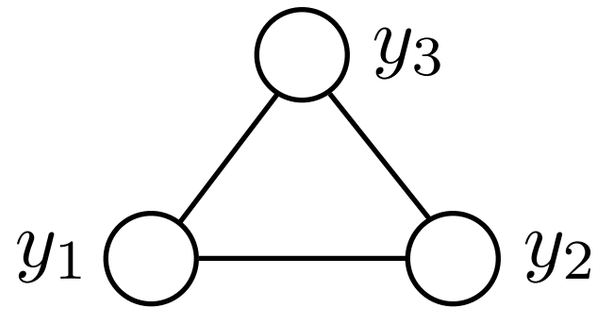
- Say we found an assignment $\hat{y} = [\hat{y}_1, \ldots, \hat{y}_n]$ **s.t.:**

$$\hat{y}_i \in \arg\max_{y_i} \bar{\theta}^{\delta}_i(y_i) \quad \textit{and} \quad \hat{y}_f \in \arg\max_{y_f} \bar{\theta}^{\delta}_f(y_f)$$

- Then $\hat{y}$ is optimal: It maximizes $\sum_f \theta_f(y_f) + \sum_i \theta_i(y_i)$.
- Will only happen if the LP is exact.

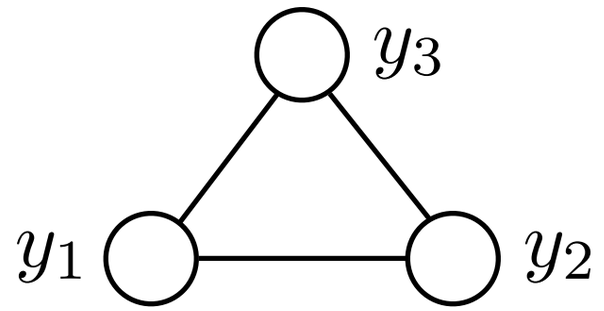
Dual Objective

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \bar{\theta}_f^{\delta}(y_f) + \sum_i \max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$



Dual Objective

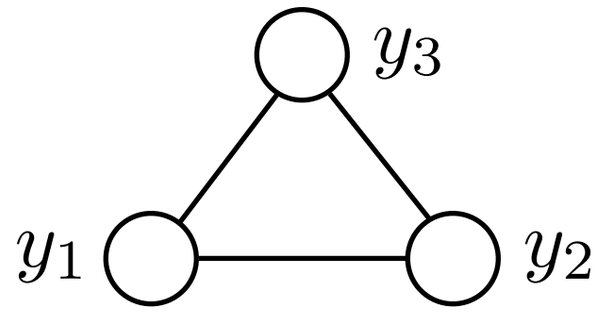
$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \bar{\theta}_f^{\delta}(y_f) + \sum_i \max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$



$$\bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Dual Objective

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \bar{\theta}_f^{\delta}(y_f) + \sum_i \max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

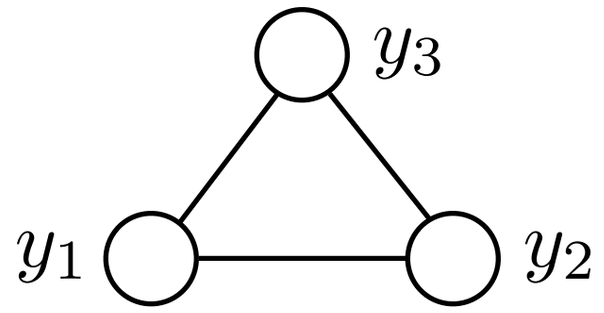


$$\bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\bar{\theta}_2^{\delta}(y_2) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Dual Objective

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \bar{\theta}_f^{\delta}(y_f) + \sum_i \max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$



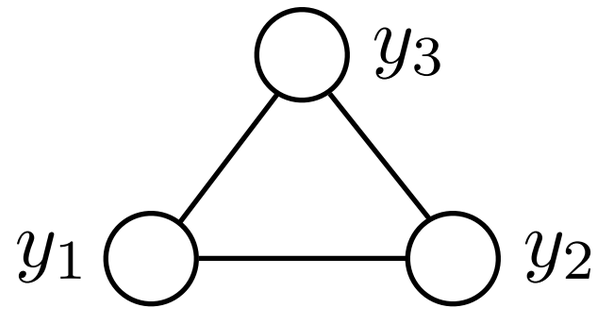
$$\bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\bar{\theta}_2^{\delta}(y_2) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\bar{\theta}_3^{\delta}(y_3) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Dual Objective

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \bar{\theta}^{\delta}_f(y_f) + \sum_i \max_{y_i} \bar{\theta}^{\delta}_i(y_i)$$



$$\bar{\theta}^{\delta}_1(y_1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

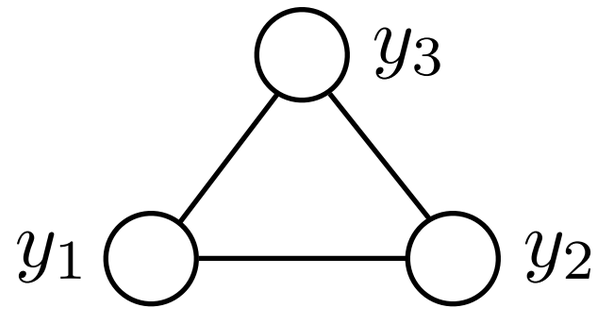
$$\bar{\theta}^{\delta}_{12}(y_1, y_2) = \begin{bmatrix} 2 & 5 \\ 3 & 4 \end{bmatrix}$$

$$\bar{\theta}^{\delta}_2(y_2) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\bar{\theta}^{\delta}_3(y_3) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Dual Objective

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \bar{\theta}^{\delta}_f(y_f) + \sum_i \max_{y_i} \bar{\theta}^{\delta}_i(y_i)$$



$$\bar{\theta}^{\delta}_1(y_1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\bar{\theta}^{\delta}_{12}(y_1, y_2) = \begin{bmatrix} 2 & 5 \\ 3 & 4 \end{bmatrix}$$

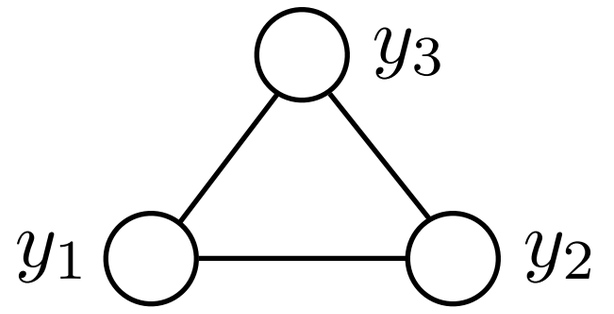
$$\bar{\theta}^{\delta}_2(y_2) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\bar{\theta}^{\delta}_{13}(y_1, y_3) = \begin{bmatrix} 4 & 3 \\ 1 & 2 \end{bmatrix}$$

$$\bar{\theta}^{\delta}_3(y_3) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Dual Objective

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \bar{\theta}^{\delta}_f(y_f) + \sum_i \max_{y_i} \bar{\theta}^{\delta}_i(y_i)$$



$$\bar{\theta}^{\delta}_1(y_1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\bar{\theta}^{\delta}_{12}(y_1, y_2) = \begin{bmatrix} 2 & 5 \\ 3 & 4 \end{bmatrix}$$

$$\bar{\theta}^{\delta}_2(y_2) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

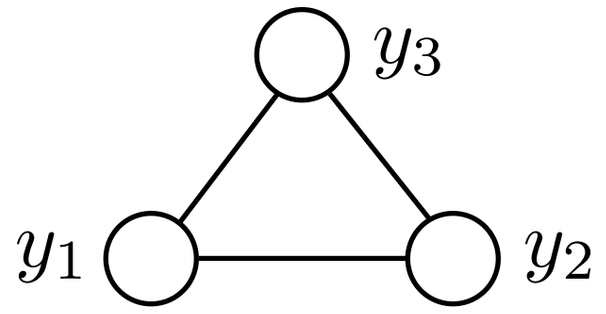
$$\bar{\theta}^{\delta}_{13}(y_1, y_3) = \begin{bmatrix} 4 & 3 \\ 1 & 2 \end{bmatrix}$$

$$\bar{\theta}^{\delta}_3(y_3) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\bar{\theta}^{\delta}_{23}(y_2, y_3) = \begin{bmatrix} 4 & 1 \\ 5 & 0 \end{bmatrix}$$

Dual Objective

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \bar{\theta}_f^{\delta}(y_f) + \sum_i \max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$



$$\bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \bar{\theta}_{12}^{\delta}(y_1, y_2) = \begin{bmatrix} 2 & 5 \\ 3 & 4 \end{bmatrix}$$

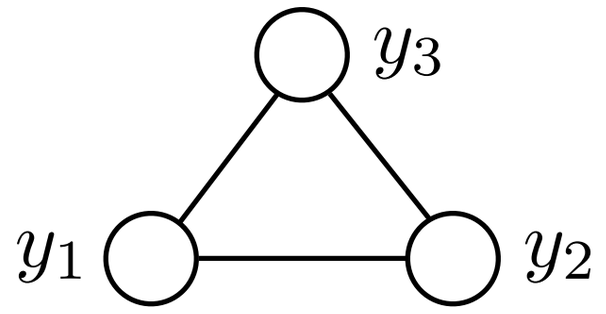
$$\bar{\theta}_2^{\delta}(y_2) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \bar{\theta}_{13}^{\delta}(y_1, y_3) = \begin{bmatrix} 4 & 3 \\ 1 & 2 \end{bmatrix}$$

$$\bar{\theta}_3^{\delta}(y_3) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \bar{\theta}_{23}^{\delta}(y_2, y_3) = \begin{bmatrix} 4 & 1 \\ 5 & 0 \end{bmatrix}$$

- The assignment $\hat{y} = [0, 1, 0]$ is optimal, since it maximizes all local and pairwise factors.

Dual Objective

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \bar{\theta}_f^{\delta}(y_f) + \sum_i \max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$



$$\bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \bar{\theta}_{12}^{\delta}(y_1, y_2) = \begin{bmatrix} 2 & 5 \\ 3 & 4 \end{bmatrix}$$

$$\bar{\theta}_2^{\delta}(y_2) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \bar{\theta}_{13}^{\delta}(y_1, y_3) = \begin{bmatrix} 4 & 3 \\ 1 & 2 \end{bmatrix}$$

$$\bar{\theta}_3^{\delta}(y_3) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \bar{\theta}_{23}^{\delta}(y_2, y_3) = \begin{bmatrix} 4 & 1 \\ 5 & 0 \end{bmatrix}$$

- The assignment $\hat{y} = [0, 1, 0]$ is optimal, since it maximizes all local and pairwise factors.
- The assignment $\hat{y} = [0, 1, 1]$ is not optimal.

Dual Decoding

Dual Decoding

- Goal. Try to find $\hat{y}$ that satisfies:

Dual Decoding

- Goal. Try to find $\hat{y}$ that satisfies:

$$\hat{y}_i \in \arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i) \quad \textit{and} \quad \hat{y}_f \in \arg\max_{y_f} \bar{\theta}_f^{\delta}(y_f)$$

Dual Decoding

- Goal. Try to find $\hat{y}$ that satisfies:

$$\hat{y}_i \in \arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i) \quad \textit{and} \quad \hat{y}_f \in \arg\max_{y_f} \bar{\theta}_f^{\delta}(y_f)$$

- When there are multiple maximizers finding such an assignment could be NP hard *[Sontag et al. 11]*.

Dual Decoding

- Goal. Try to find $\hat{y}$ that satisfies:

$$\hat{y}_i \in \arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i) \quad \textit{and} \quad \hat{y}_f \in \arg\max_{y_f} \bar{\theta}_f^{\delta}(y_f)$$

- When there are multiple maximizers finding such an assignment could be NP hard *[Sontag et al. 11]*.
- In practice, we "guess" some $\hat{y}_i \in \arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$

Dual Decoding

- Goal. Try to find $\hat{y}$ that satisfies:

$$\hat{y}_i \in \arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i) \quad \textit{and} \quad \hat{y}_f \in \arg\max_{y_f} \bar{\theta}_f^{\delta}(y_f)$$

- When there are multiple maximizers finding such an assignment could be NP hard *[Sontag et al. 11]*.
- In practice, we "guess" some $\hat{y}_i \in \arg\max \bar{\theta}_i^{\delta}(y_i)$

- If it satisfies $\hat{y}_f \in \arg\max_{y_f} \bar{\theta}_f^{\delta}(y_f)$ we have an optimal solution.

Dual Decoding

- Goal. Try to find $\hat{y}$ that satisfies:

$$\hat{y}_i \in \arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i) \quad \textit{and} \quad \hat{y}_f \in \arg\max_{y_f} \bar{\theta}_f^{\delta}(y_f)$$

- When there are multiple maximizers finding such an assignment could be NP hard *[Sontag et al. 11]*.
- In practice, we "guess" some $\hat{y}_i \in \arg\max \bar{\theta}_i^{\delta}(y_i)$

- If it satisfies $\hat{y}_f \in \arg\max_{y_f} \bar{\theta}_f^{\delta}(y_f)$ we have an optimal solution.
- This will exist such $\delta, \hat{y}$ if the LP is exact and has a unique maximum *[Sontag et al. 11]*.

Dual Decoding

- Goal. Try to find $\hat{y}$ that satisfies:

$$\hat{y}_i \in \arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i) \quad \textit{and} \quad \hat{y}_f \in \arg\max_{y_f} \bar{\theta}_f^{\delta}(y_f)$$

- When there are multiple maximizers finding such an assignment could be NP hard *[Sontag et al. 11]*.
- In practice, we "guess" some $\hat{y}_i \in \arg\max \bar{\theta}_i^{\delta}(y_i)$

- If it satisfies $\hat{y}_f \in \arg\max_{y_f} \bar{\theta}_f^{\delta}(y_f)$ we have an optimal solution.
- This will exist such $\delta, \hat{y}$ if the LP is exact and has a unique maximum *[Sontag et al. 11]*.
- More clever decoding can be done: e.g., by maximizing over larger factors.

Outline

- Solving LP relaxations:
  - Subgradient algorithms
  - Coordinate descent algorithms
- Parameter learning with LP relaxations:
  - Structured Prediction
  - Structured LP perceptron.
  - Coordinate descent algorithms.
  - Cutting plane algorithms.
  - Pseudo-max algorithms.

Dual Algorithms - Subgradient

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

Dual Algorithms - Subgradient

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- Unconstrained, non differentiable.

Dual Algorithms - Subgradient

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- Unconstrained, non differentiable.
- Can use subgradient descent *[Komodakis 07]*.

Dual Algorithms - Subgradient

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- Unconstrained, non differentiable.
- Can use subgradient descent *[Komodakis 07]*.
- Algorithm:

Dual Algorithms - Subgradient

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- Unconstrained, non differentiable.
- Can use subgradient descent *[Komodakis 07]*.
- Algorithm:
  - Initialize $\boldsymbol{\delta}_0$

Dual Algorithms - Subgradient

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- Unconstrained, non differentiable.
- Can use subgradient descent *[Komodakis 07]*.
- Algorithm:
  - Initialize $\boldsymbol{\delta}_0$
  - Repeat:

Dual Algorithms - Subgradient

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- Unconstrained, non differentiable.
- Can use subgradient descent *[Komodakis 07]*.
- Algorithm:
  - Initialize $\boldsymbol{\delta}_0$
  - Repeat:
    - Calculate a subgradient $\boldsymbol{v}$ at $\boldsymbol{\delta}_t$

Dual Algorithms - Subgradient

- Recall our dual objective:

$$g(\boldsymbol{\delta}) = \sum_f \max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \sum_i \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{f:i \in f} \delta_{fi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- Unconstrained, non differentiable.
- Can use subgradient descent *[Komodakis 07]*.
- Algorithm:
  - Initialize $\boldsymbol{\delta}_0$
  - Repeat:
    - Calculate a subgradient $\boldsymbol{v}$ at $\boldsymbol{\delta}_t$
    - Update: $\boldsymbol{\delta}_{t+1} = \boldsymbol{\delta}_t - \eta_t \boldsymbol{v}$

Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.

Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.

Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.

Calculating the Subgradient

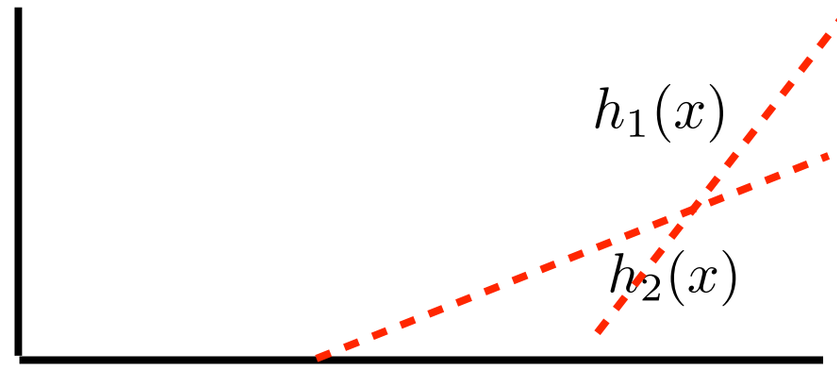
- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.

Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.

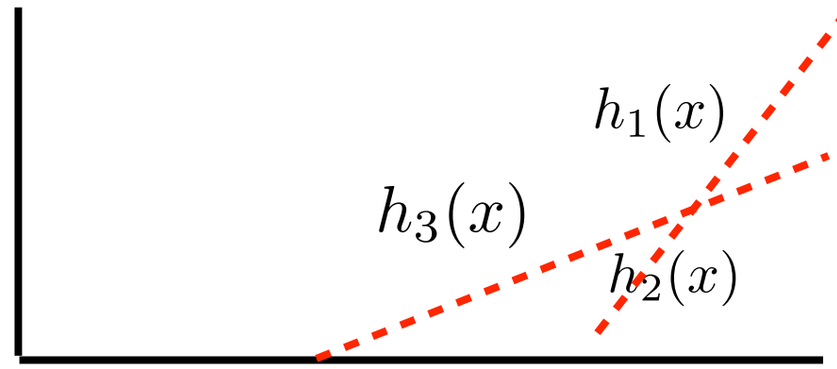
Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.



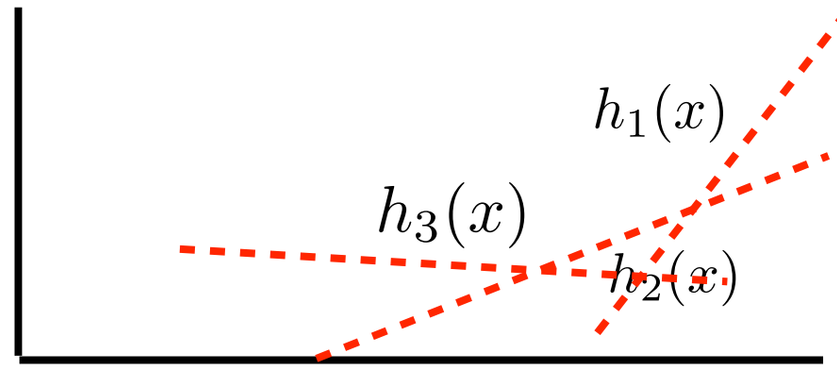
Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.



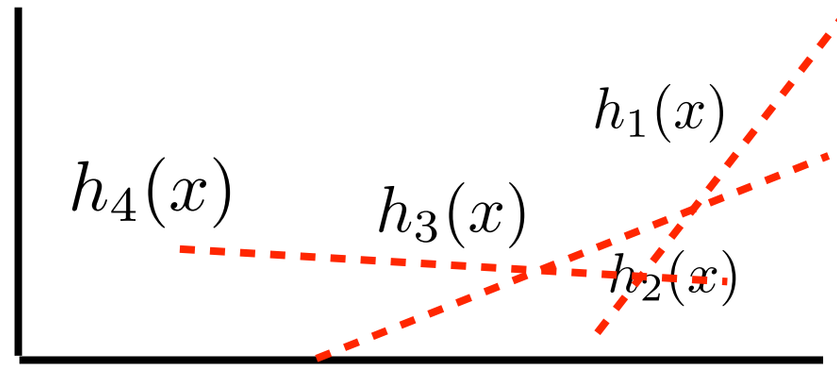
Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.



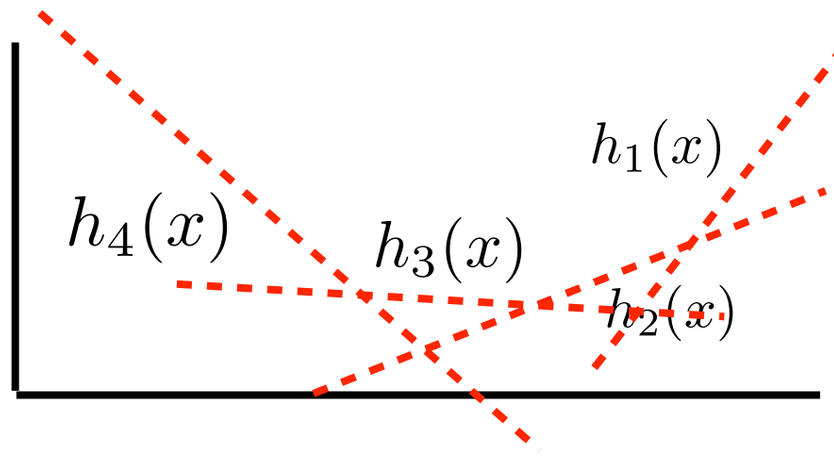
Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.



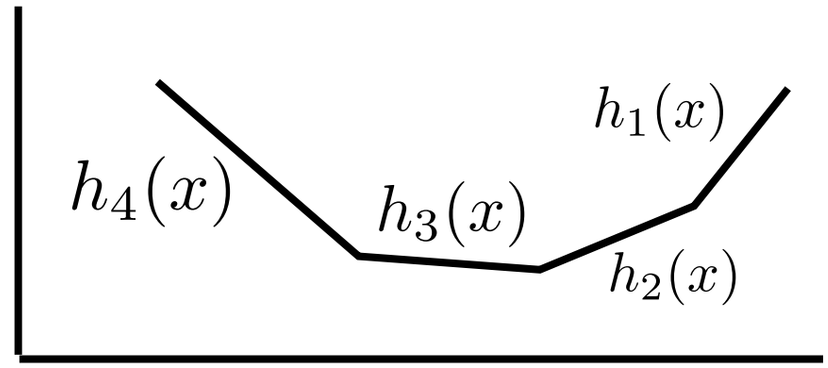
Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.



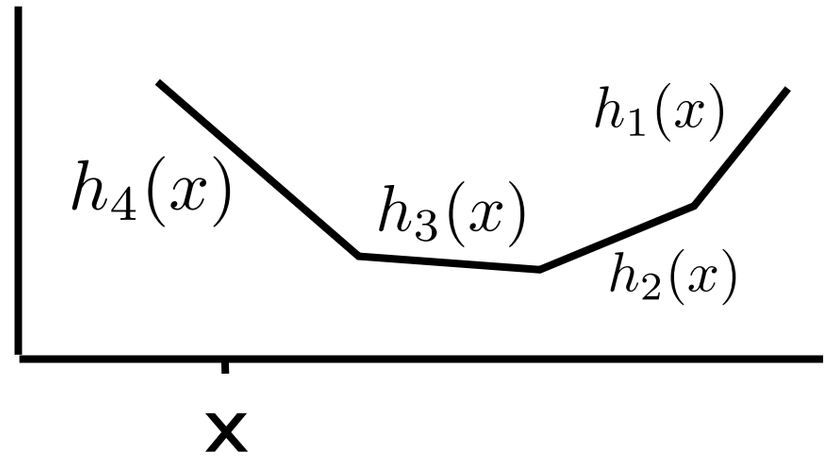
Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.



Calculating the Subgradient

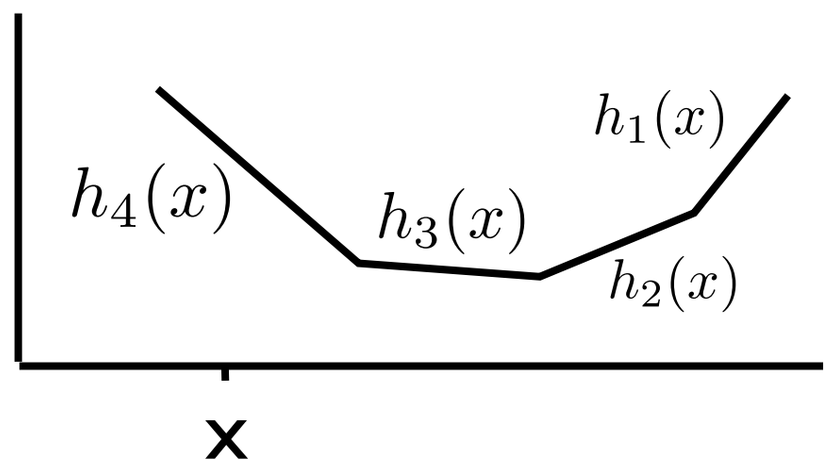
- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.



Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.
- One subgradient of h at x is v:

$$v = \frac{\partial}{\partial x} h_{\hat{i}}(x)$$

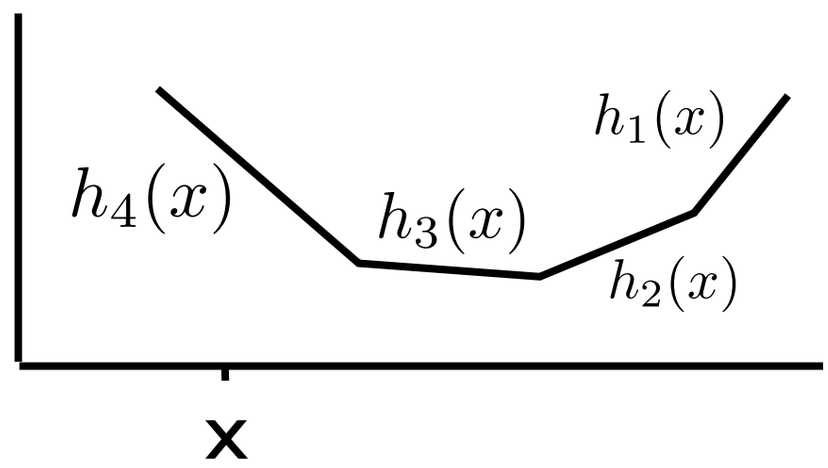


Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.
- One subgradient of h at x is v:

$$v = \frac{\partial}{\partial x} h_{\hat{i}}(x)$$

- Where: $\hat{i} \in \arg\max_i h_i(x)$

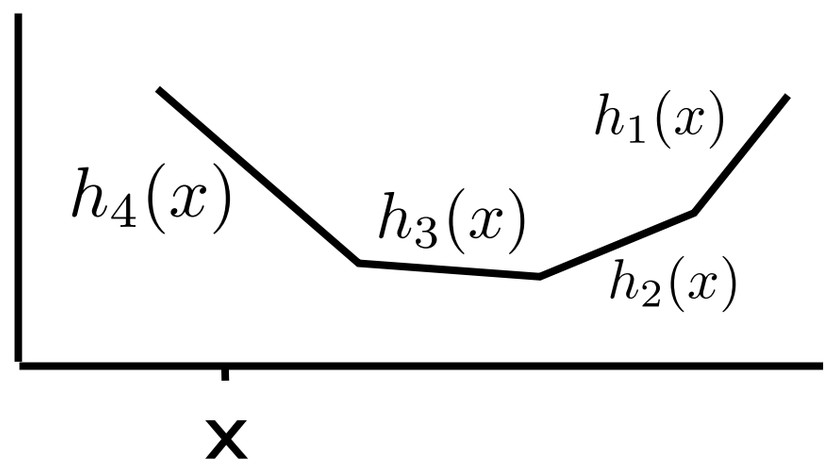


Calculating the Subgradient

- Consider a function $h(x) = \max_i h_i(x)$
- Where the functions $h_i(x)$ are differentiable.
- One subgradient of h at x is v:

$$v = \frac{\partial}{\partial x} h_{\hat{i}}(x)$$

- Where: $\hat{i} \in \arg\max_i h_i(x)$



$\hat{i} = 4$

Calculating the Subgradient

Calculating the Subgradient

- To get the subgradient of $g(\boldsymbol{\delta})$ w.r.t. $\delta_{fi}(y_i)$, the relevant part of the objective is:

Calculating the Subgradient

- To get the subgradient of $g(\boldsymbol{\delta})$ w.r.t. $\delta_{fi}(y_i)$, the relevant part of the objective is:

$$\max_{y_f} \left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j) \right] + \max_{y_i} \left[\theta_i(y_i) + \sum_{g: i \in g} \delta_{gi}(y_i) \right]$$

Calculating the Subgradient

- To get the subgradient of $g(\boldsymbol{\delta})$ w.r.t. $\delta_{fi}(y_i)$, the relevant part of the objective is:

$$\max_{y_f}\left[\theta_f(y_f)-\sum_{j\in f}\delta_{fj}(y_j)\right]+\max_{y_i}\left[\theta_i(y_i)+\sum_{g:i\in g}\delta_{gi}(y_i)\right]$$

$$\downarrow \text{ Maximize}$$

$$y_i^f, y_j^f, y_k^f \;,\; i,j,k\in f$$

Calculating the Subgradient

- To get the subgradient of $g(\boldsymbol{\delta})$ w.r.t. $\delta_{fi}(y_i)$, the relevant part of the objective is:

$$\max_{y_f}\left[\theta_f(y_f)-\sum_{j\in f}\delta_{fj}(y_j)\right]+\max_{y_i}\left[\theta_i(y_i)+\sum_{g:i\in g}\delta_{gi}(y_i)\right]$$

Maximize $\quad y_i^f, y_j^f, y_k^f \;,\; i,j,k\in f$

Maximize $\quad y_i^s$

Calculating the Subgradient

- To get the subgradient of $g(\boldsymbol{\delta})$ w.r.t. $\delta_{fi}(y_i)$, the relevant part of the objective is:

$$\max_{y_f}\left[\theta_f(y_f)-\sum_{j\in f}\delta_{fj}(y_j)\right]+\max_{y_i}\left[\theta_i(y_i)+\sum_{g:i\in g}\delta_{gi}(y_i)\right]$$

Maximize → $y_i^f, y_j^f, y_k^f \;,\; i,j,k\in f$

Maximize → y_i^s

- The subgradient w.r.t. $\delta_{fi}(y_i)$ is:

Calculating the Subgradient

- To get the subgradient of $g(\boldsymbol{\delta})$ w.r.t. $\delta_{fi}(y_i)$, the relevant part of the objective is:

$$\max_{y_f}\left[\theta_f(y_f) - \sum_{j\in f}\delta_{fj}(y_j)\right] + \max_{y_i}\left[\theta_i(y_i) + \sum_{g:i\in g}\delta_{gi}(y_i)\right]$$

Maximize: $y_i^f, y_j^f, y_k^f \;,\; i,j,k \in f$

Maximize: y_i^s

- The subgradient w.r.t. $\delta_{fi}(y_i)$ is:

$$v_{fi}(y_i) = -\mathbf{1}[y_i^f = y_i] + \mathbf{1}[y_i^s = y_i]$$

Calculating the Subgradient

- To get the subgradient of $g(\boldsymbol{\delta})$ w.r.t. $\delta_{fi}(y_i)$, the relevant part of the objective is:

$$\max_{y_f}\left[\theta_f(y_f) - \sum_{j\in f}\delta_{fj}(y_j)\right] + \max_{y_i}\left[\theta_i(y_i) + \sum_{g:i\in g}\delta_{gi}(y_i)\right]$$

Maximize $\quad$ Maximize

$$y_i^f, y_j^f, y_k^f, \; i,j,k \in f \qquad\qquad y_i^s$$

- The subgradient w.r.t. $\delta_{fi}(y_i)$ is:

$$v_{fi}(y_i) = -\mathbf{1}[y_i^f = y_i] + \mathbf{1}[y_i^s = y_i]$$

- All zeros if $y_i^f = y_i^s$ (i.e., maximizers agree).

Calculating the Subgradient

- To get the subgradient of $g(\boldsymbol{\delta})$ w.r.t. $\delta_{fi}(y_i)$, the relevant part of the objective is:

$$\max_{y_f} \left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j) \right] + \max_{y_i} \left[\theta_i(y_i) + \sum_{g: i \in g} \delta_{gi}(y_i) \right]$$

Maximize $y_i^f, y_j^f, y_k^f, \; i,j,k \in f$

Maximize y_i^s

- The subgradient w.r.t. $\delta_{fi}(y_i)$ is:

$$v_{fi}(y_i) = -\mathbf{1}[y_i^f = y_i] + \mathbf{1}[y_i^s = y_i]$$

- All zeros if $y_i^f = y_i^s$ (i.e., maximizers agree).

- Makes sense since this is the optimality condition.

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$

$i = 1$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix}$$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers:

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 0, y_2^f = 1$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 0, y_2^f = 1$ $\qquad$ $y_1^s = 1$

Subgradient - Example

$$\max_{y_f}\underbrace{\left[\theta_f(y_f)-\sum_{j\in f}\delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)}+\max_{y_i}\underbrace{\left[\theta_i(y_i)+\sum_{g:i\in g}\delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f=\{1,2\}$
$i=1$

$$\bar{\theta}_f^{\delta}(y_1,y_2)=\begin{bmatrix}1 & 4\\ 3 & 2\end{bmatrix}\qquad \bar{\theta}_1^{\delta}(y_1)=\begin{bmatrix}1\\ 3\end{bmatrix}$$

Maximizers: $y_1^f=0, y_2^f=1$ $\qquad$ $y_1^s=1$

Do not agree!

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 0, y_2^f = 1$ $\qquad$ $y_1^s = 1$ $\qquad$ Do not agree!

Subgradient w.r.t. $\delta_{f1}(y_1)$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 0, y_2^f = 1$ $\qquad$ $y_1^s = 1$ $\qquad$ Do not agree!

Subgradient w.r.t. $\delta_{f1}(y_1)$

$$v_{f1}(y_1) = [-1, 1]$$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 0, y_2^f = 1$ $\qquad$ $y_1^s = 1$ $\qquad$ Do not agree!

Subgradient w.r.t. $\delta_{f1}(y_1)$

$$v_{f1}(y_1) = [-1, 1]$$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 0, y_2^f = 1$ and $y_1^s = 1$

Do not agree!

Subgradient w.r.t. $\delta_{f1}(y_1)$

$$v_{f1}(y_1) = [-1, 1]$$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$

$i = 1$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers:

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 1, y_2^f = 1$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 1, y_2^f = 1$ $\qquad$ $y_1^s = 1$

Subgradient - Example

$$\max_{y_f}\underbrace{\left[\theta_f(y_f)-\sum_{j\in f}\delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)}+\max_{y_i}\underbrace{\left[\theta_i(y_i)+\sum_{g:i\in g}\delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f=\{1,2\}$
$i=1$

$$\bar{\theta}_f^{\delta}(y_1,y_2)=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$$

$$\bar{\theta}_1^{\delta}(y_1)=\begin{bmatrix}1\\ 3\end{bmatrix}$$

Maximizers: $y_1^f=1, y_2^f=1$ $\quad$ $y_1^s=1$

Agree!

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 1, y_2^f = 1$ $\qquad$ $y_1^s = 1$ $\qquad$ Agree!

Subgradient w.r.t. $\delta_{f1}(y_1)$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 1, y_2^f = 1$ $\qquad$ $y_1^s = 1$ $\qquad$ Agree!

Subgradient w.r.t. $\delta_{f1}(y_1)$

$$v_{f1}(y_1) = [0, 0]$$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 1, y_2^f = 1$ $\qquad$ $y_1^s = 1$ $\qquad$ Agree!

-1

Subgradient w.r.t. $\delta_{f1}(y_1)$

$$v_{f1}(y_1) = [0, 0]$$

Subgradient - Example

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}_f^{\delta}(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}_i^{\delta}(y_i)}$$

$f = \{1, 2\}$
$i = 1$

$$\bar{\theta}_f^{\delta}(y_1, y_2) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \bar{\theta}_1^{\delta}(y_1) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Maximizers: $y_1^f = 1, y_2^f = 1$ and $y_1^s = 1$ — Agree!

-I

+I

Subgradient w.r.t. $\delta_{f1}(y_1)$

$$v_{f1}(y_1) = [0, 0]$$

Subgradient - Computational Aspects

Subgradient - Computational Aspects

- The main computational load in calculating the subgradient is finding:

Subgradient - Computational Aspects

- The main computational load in calculating the subgradient is finding:

$$\arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right]$$

Subgradient - Computational Aspects

- The main computational load in calculating the subgradient is finding:

$$\arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right]$$

- Worst case complexity is $O(2^{|f|})$ where $|f|$ is the number of variables in the set.

Subgradient - Computational Aspects

- The main computational load in calculating the subgradient is finding:

$$\arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right]$$

- Worst case complexity is $O(2^{|f|})$ where $|f|$ is the number of variables in the set.
- Easy for small sets (e.g., pairwise models).

Subgradient - Computational Aspects

- The main computational load in calculating the subgradient is finding:

$$\arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right]$$

- Worst case complexity is $O(2^{|f|})$ where $|f|$ is the number of variables in the set.
- Easy for small sets (e.g., pairwise models).
- But, can be done even for $|f| = n$ in some cases, as we next show.

Tractable Global Factors

Tractable Global Factors

- In some cases, we can apply the method even for factors defined over all variables $y_1, \ldots, y_n$.

Tractable Global Factors

- In some cases, we can apply the method even for factors defined over all variables $y_1, \ldots, y_n$.
- All we need to do is solve maximization problems:

Tractable Global Factors

- In some cases, we can apply the method even for factors defined over all variables $y_1, \ldots, y_n$.
- All we need to do is solve maximization problems:

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right] \qquad f = \{1, \ldots, n\}$$

Tractable Global Factors

- In some cases, we can apply the method even for factors defined over all variables $y_1, \ldots, y_n$.
- All we need to do is solve maximization problems:

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right] \qquad f = \{1, \ldots, n\}$$

- Lets see some examples.

Tractable Global Factors

- In some cases, we can apply the method even for factors defined over all variables $y_1, \ldots, y_n$.
- All we need to do is solve maximization problems:

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right] \qquad f = \{1, \ldots, n\}$$

- Lets see some examples.

- Cardinality potential:

Tractable Global Factors

- In some cases, we can apply the method even for factors defined over all variables $y_1, \ldots, y_n$.
- All we need to do is solve maximization problems:

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right] \qquad f = \{1, \ldots, n\}$$

- Lets see some examples.

- Cardinality potential:

$$\theta_{\mathrm{C}}(y) = h\left(\sum_{i=1}^{n} y_i \right)$$

Tractable Global Factors

- In some cases, we can apply the method even for factors defined over all variables $y_1, \ldots, y_n$.
- All we need to do is solve maximization problems:

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right] \qquad f = \{1, \ldots, n\}$$

- Lets see some examples.

- Cardinality potential:

$$\theta_{\mathrm{C}}(y) = h\left(\sum_{i=1}^{n} y_i \right) \qquad y_i \in \{0, 1\}$$

Tractable Global Factors

- In some cases, we can apply the method even for factors defined over all variables $y_1, \ldots, y_n$.
- All we need to do is solve maximization problems:

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right] \qquad f = \{1, \ldots, n\}$$

- Lets see some examples.

- Cardinality potential:

$$\theta_{\mathrm{C}}(y) = h\left(\sum_{i=1}^{n} y_i \right) \qquad y_i \in \{0, 1\}$$

- Maximization can be done with sorting *[Tarlow et al 10]*

Tractable Global Factors - Binary Variables

Tractable Global Factors - Binary Variables

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right]$$

Tractable Global Factors - Binary Variables

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} \delta_{fi}(y_i) \right]$$

Can assume wlog:

$$\delta_{fi}(y_i) = \begin{bmatrix} 0 \\ \delta_{fi} \end{bmatrix}$$

Tractable Global Factors - Binary Variables

Can assume wlog:

$$\delta_{fi}(y_i) = \begin{bmatrix} 0 \\ \delta_{fi} \end{bmatrix}$$

Tractable Global Factors - Binary Variables

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

Can assume wlog:

$$\delta_{fi}(y_i) = \begin{bmatrix} 0 \\ \delta_{fi} \end{bmatrix}$$

Tractable Global Factors - Binary Variables

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

Can assume wlog:

$$\delta_{fi}(y_i) = \begin{bmatrix} 0 \\ \delta_{fi} \end{bmatrix}$$

- Graph structure constraints

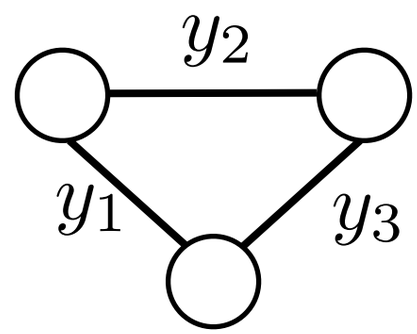
Tractable Global Factors - Binary Variables

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

Can assume wlog:

$$\delta_{fi}(y_i) = \begin{bmatrix} 0 \\ \delta_{fi} \end{bmatrix}$$

- Graph structure constraints



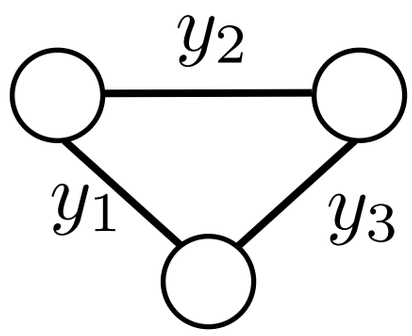
Tractable Global Factors - Binary Variables

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

Can assume wlog:

$$\delta_{fi}(y_i) = \begin{bmatrix} 0 \\ \delta_{fi} \end{bmatrix}$$

- Graph structure constraints



$y_i = 1$ if edge i is in the graph

Tractable Global Factors - Binary Variables

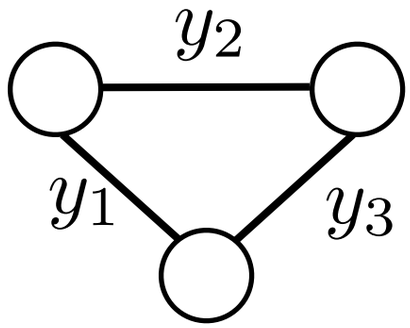
$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

Can assume wlog:

$$\delta_{fi}(y_i) = \begin{bmatrix} 0 \\ \delta_{fi} \end{bmatrix}$$

- Graph structure constraints

$$\theta_T(y) = \begin{cases} 0 & \text{y is a tree} \\ -\infty & \text{Otherwise} \end{cases}$$



$y_i = 1$ if edge i is in the graph

Tractable Global Factors - Binary Variables

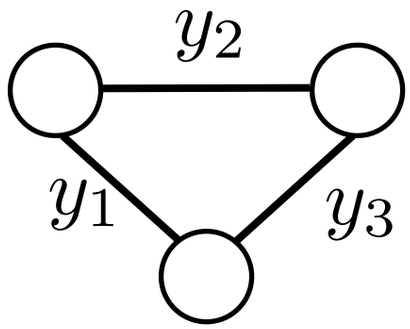
$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

Can assume wlog:

$$\delta_{fi}(y_i) = \begin{bmatrix} 0 \\ \delta_{fi} \end{bmatrix}$$

- Graph structure constraints

$$\theta_T(y) = \begin{cases} 0 & \text{y is a tree} \\ -\infty & \text{Otherwise} \end{cases}$$



Tractable Global Factors - Binary Variables

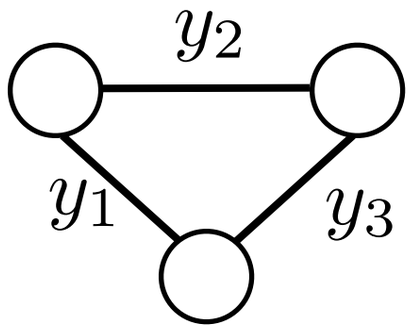
$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

Can assume wlog:

$$\delta_{fi}(y_i) = \begin{bmatrix} 0 \\ \delta_{fi} \end{bmatrix}$$

- Graph structure constraints

$$\theta_T(y) = \begin{cases} 0 & \text{y is a tree} \\ -\infty & \text{Otherwise} \end{cases}$$



- Maximization can be done with maximum weight spanning tree *[McDonald et al 10]*. Directed and undirected versions available.

Tractable Global Factors - Binary Variables

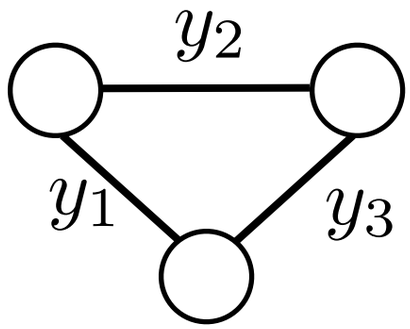
$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

Can assume wlog:

$$\delta_{fi}(y_i) = \begin{bmatrix} 0 \\ \delta_{fi} \end{bmatrix}$$

- Graph structure constraints

$$\theta_T(y) = \begin{cases} 0 & \text{y is a tree} \\ -\infty & \text{Otherwise} \end{cases}$$



- Maximization can be done with maximum weight spanning tree *[McDonald et al 10]*. Directed and undirected versions available.

- Can be generalized to the constraint that y is an independent set of a matroid (or matroid intersection). Maximization can be done with maximum weight independent set in a matroid or matroid intersection.

Tractable Global Factors - Binary Variables

Tractable Global Factors - Binary Variables

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

Tractable Global Factors - Binary Variables

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

- If $\theta_f(y_f)$ is supermodular, the above maximization is also supermodular and tractable. Since:

Tractable Global Factors - Binary Variables

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

- If $\theta_f(y_f)$ is supermodular, the above maximization is also supermodular and tractable. Since:

$\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi}$ supermodular if $\theta_f(y_f)$ is supermodular

Tractable Global Factors - Binary Variables

$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

- If $\theta_f(y_f)$ is supermodular, the above maximization is also supermodular and tractable. Since:

$\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi}$ supermodular if $\theta_f(y_f)$ is supermodular

- Non binary example: when $\theta_f(y_f)$ is a tree shaped MRF *[Komodakis 07]*.

Tractable Global Factors - Binary Variables

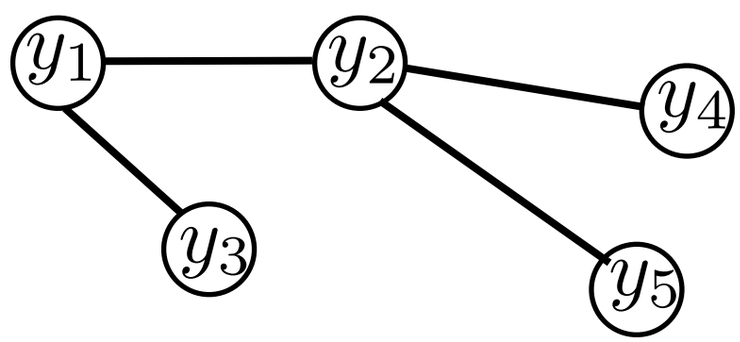
$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

- If $\theta_f(y_f)$ is supermodular, the above maximization is also supermodular and tractable. Since:

$\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi}$ supermodular if $\theta_f(y_f)$ is supermodular

- Non binary example: when $\theta_f(y_f)$ is a tree shaped MRF *[Komodakis 07]*.

$$\theta_f(y_f) = \sum_{ij \in T} \lambda_{ij}(y_i, y_j)$$



Tractable Global Factors - Binary Variables

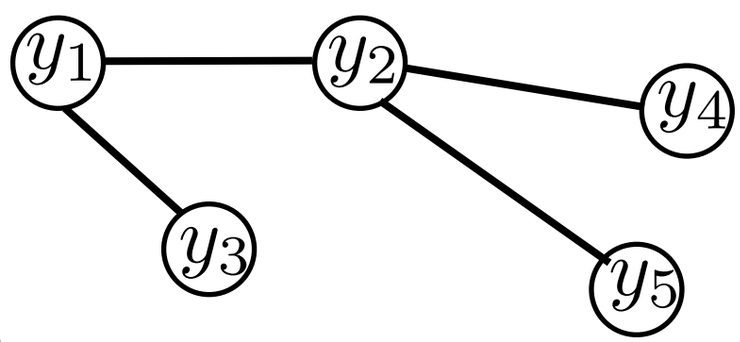
$$y_f^* = \arg\max_{y_f} \left[\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi} \right]$$

- If $\theta_f(y_f)$ is supermodular, the above maximization is also supermodular and tractable. Since:

$\theta_f(y_f) - \sum_{i \in f} y_i \delta_{fi}$ supermodular if $\theta_f(y_f)$ is supermodular

- Non binary example: when $\theta_f(y_f)$ is a tree shaped MRF *[Komodakis 07]*.

$$\theta_f(y_f) = \sum_{ij \in T} \lambda_{ij}(y_i, y_j)$$



- Use dynamic programming to maximize.

Outline

- Solving LP relaxations:
  - Subgradient algorithms
  - Coordinate descent algorithms
- Parameter learning with LP relaxations:
  - Structured Prediction
  - Structured LP perceptron.
  - Coordinate descent algorithms.
  - Cutting plane algorithms.
  - Pseudo-max algorithms.

Dual Algorithm - Coordinate Descent

Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.

Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.
- Alternative approach:

Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block

Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block
  - Find values of block that minimize dual objective

Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block
  - Find values of block that minimize dual objective

Dual Algorithm - Coordinate Descent

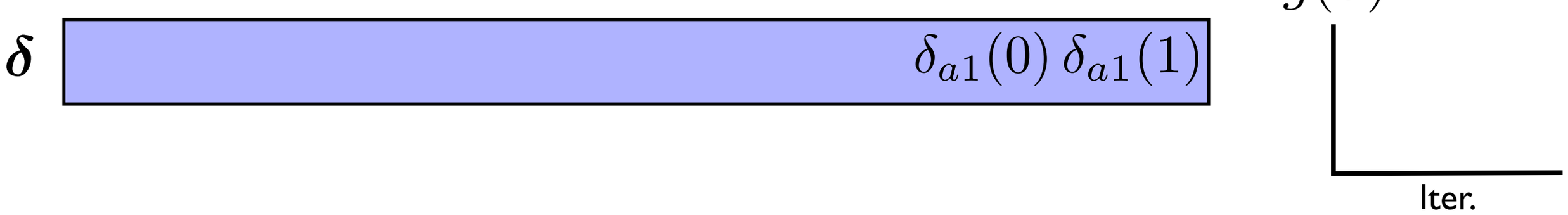
- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block
  - Find values of block that minimize dual objective

Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block
  - Find values of block that minimize dual objective

Dual Algorithm - Coordinate Descent

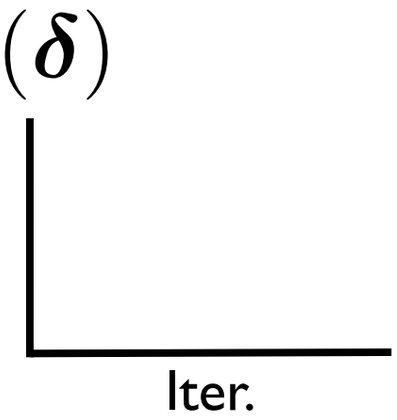
- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block
  - Find values of block that minimize dual objective



Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block
  - Find values of block that minimize dual objective

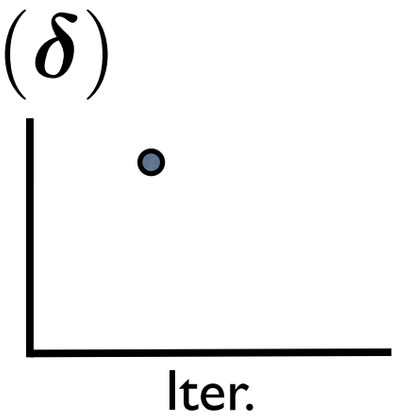
δ | Update | Fix | $\delta_{a1}(0)\ \delta_{a1}(1)$



Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block
  - Find values of block that minimize dual objective

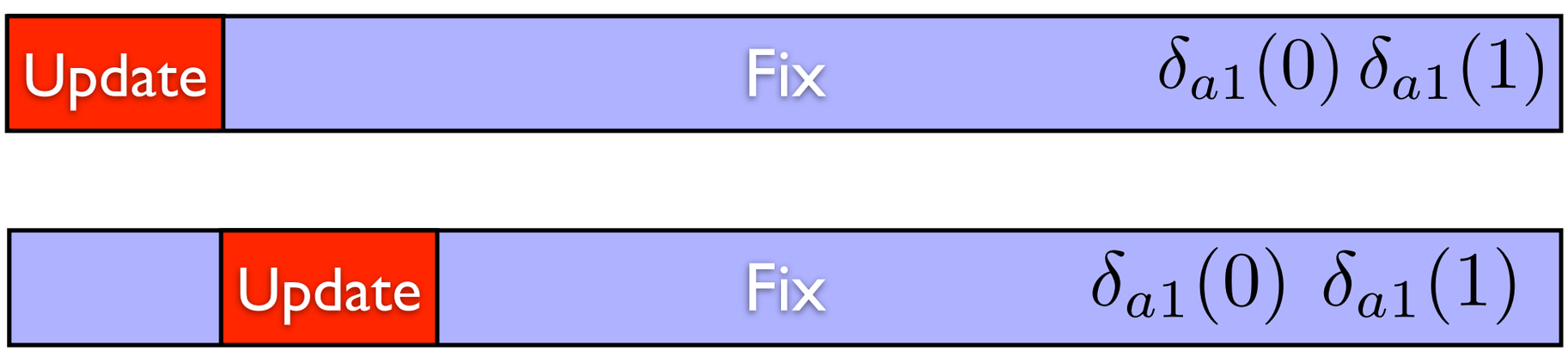
$\boldsymbol{\delta}$ | Update | Fix | $\delta_{a1}(0)\ \delta_{a1}(1)$

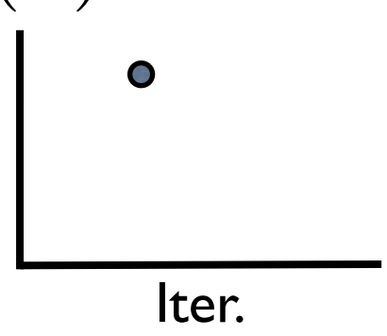


Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block
  - Find values of block that minimize dual objective

$\boldsymbol{\delta}$

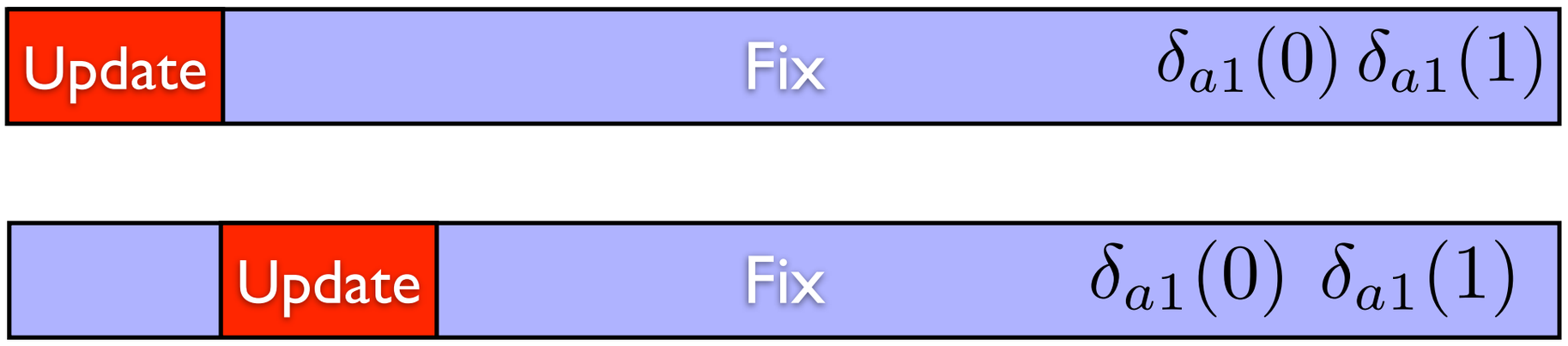


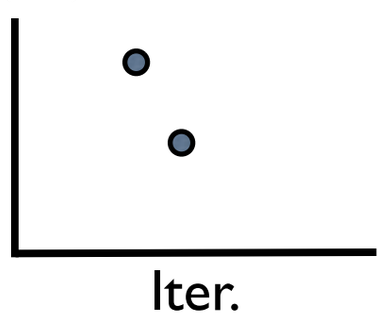


Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block
  - Find values of block that minimize dual objective

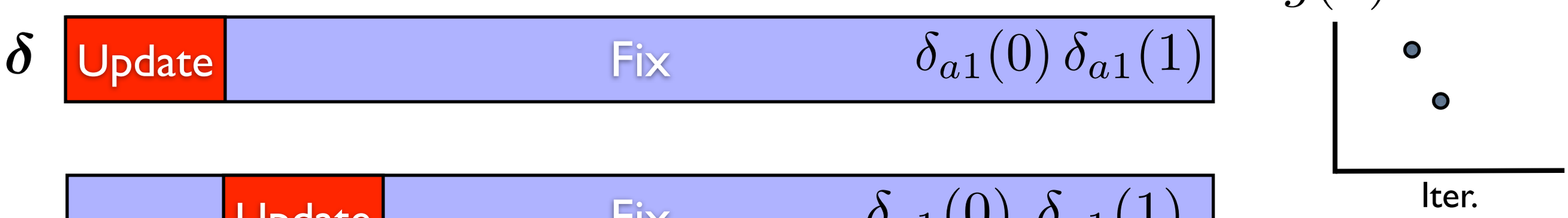
$\boldsymbol{\delta}$





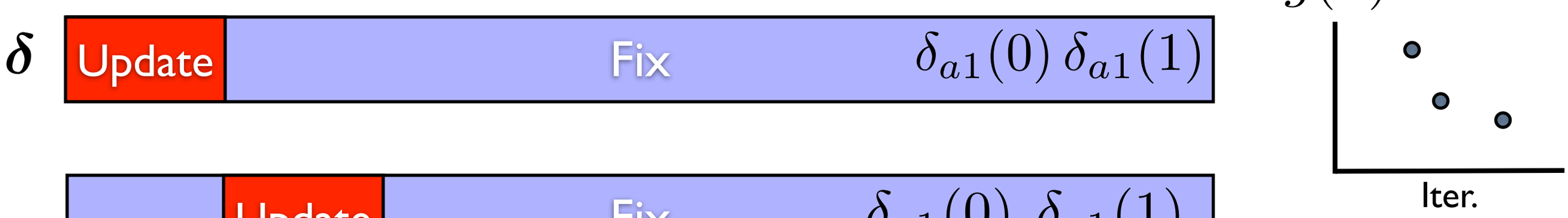
Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block
  - Find values of block that minimize dual objective



Dual Algorithm - Coordinate Descent

- Subgradient descent makes only small updates.
- Alternative approach:
  - Fix all dual variables except some block
  - Find values of block that minimize dual objective



Dual Coordinate Descent

Dual Coordinate Descent

- Design choices:
  - Which coordinate blocks to update?
  - How to optimize over them?

Dual Coordinate Descent

- Design choices:
  - Which coordinate blocks to update?
  - How to optimize over them?
- Some common choices:

Dual Coordinate Descent

- Design choices:
  - Which coordinate blocks to update?
  - How to optimize over them?
- Some common choices:
  - Pick f,i. Update $\delta_{fi}(y_i) \quad , \quad \forall y_i$

Dual Coordinate Descent

- Design choices:
  - Which coordinate blocks to update?
  - How to optimize over them?
- Some common choices:
  - Pick f,i. Update $\delta_{fi}(y_i) \quad , \quad \forall y_i$

Max Sum Diffusion
[Schlesinger 76, Werner 07]

Dual Coordinate Descent

- Design choices:
  - Which coordinate blocks to update?
  - How to optimize over them?
- Some common choices:
  - Pick f,i. Update $\delta_{fi}(y_i) \quad , \quad \forall y_i$

    Max Sum Diffusion
    [Schlesinger 76, Werner 07]
  - Pick f. Update $\delta_{fi}(y_i) \quad , \quad \forall i, y_i$

Dual Coordinate Descent

- Design choices:
  - Which coordinate blocks to update?
  - How to optimize over them?
- Some common choices:
  - Pick f,i. Update $\delta_{fi}(y_i) \quad , \quad \forall y_i$ Max Sum Diffusion *[Schlesinger 76, Werner 07]*
  - Pick f. Update $\delta_{fi}(y_i) \quad , \quad \forall i, y_i$ MPLP *[Globerson & Jaakkola 07]*

Dual Coordinate Descent

- Design choices:
  - Which coordinate blocks to update?
  - How to optimize over them?
- Some common choices:
  - Pick f,i. Update $\delta_{fi}(y_i) \quad , \quad \forall y_i$ Max Sum Diffusion *[Schlesinger 76, Werner 07]*
  - Pick f. Update $\delta_{fi}(y_i) \quad , \quad \forall i, y_i$ MPLP *[Globerson & Jaakkola 07]*

- In these cases, efficient closed form updates are often available.

Dual Coordinate Descent

- Design choices:
  - Which coordinate blocks to update?
  - How to optimize over them?
- Some common choices:
  - Pick f,i. Update $\delta_{fi}(y_i) \quad , \quad \forall y_i$ Max Sum Diffusion *[Schlesinger 76, Werner 07]*
  - Pick f. Update $\delta_{fi}(y_i) \quad , \quad \forall i, y_i$ MPLP *[Globerson & Jaakkola 07]*

- In these cases, efficient closed form updates are often available.
- Many works on variations (by Wainwright, Kolmogorov, Werner, Sontag, Hazan among others).

Max-Sum-Diffusion (MSD) Update

- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall y_i$

Max-Sum-Diffusion (MSD) Update

- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall y_i$

- Their optimal value given the others is:

Max-Sum-Diffusion (MSD) Update

- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall y_i$

- Their optimal value given the others is:

$$\delta_{fi}(y_i) = -\frac{1}{2}\delta_i^{-f}(y_i) + \frac{1}{2}\max_{y_{f\setminus i}}\left[\theta_f(y_f) - \sum_{j\in f\setminus i}\delta_{fj}(y_j)\right]$$

- Where: $\delta_i^{-f}(y_i) = \theta_i(y_i) + \sum_{g:i\in g, g\neq f}\delta_{gi}(y_i)$

Max-Sum-Diffusion (MSD) Update

- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall y_i$

- Their optimal value given the others is:

$$\delta_{fi}(y_i) = -\frac{1}{2}\delta_i^{-f}(y_i) + \frac{1}{2}\max_{y_{f\setminus i}}\left[\theta_f(y_f) - \sum_{j\in f\setminus i}\delta_{fj}(y_j)\right]$$

- Where: $\delta_i^{-f}(y_i) = \theta_i(y_i) + \sum_{g:i\in g, g\neq f}\delta_{gi}(y_i)$

- Update depends only on some of the fixed messages.

Max-Sum-Diffusion (MSD) Update

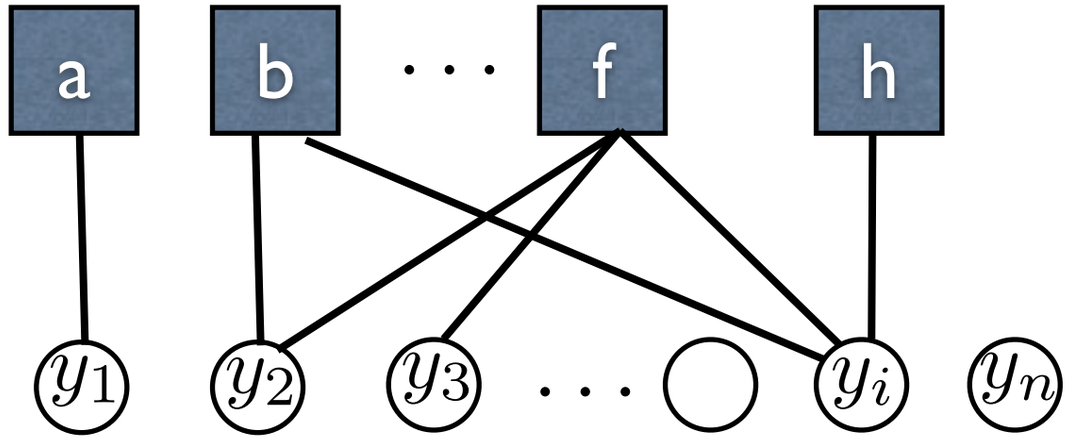
- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall y_i$

- Their optimal value given the others is:

$$\delta_{fi}(y_i) = -\frac{1}{2}\delta_i^{-f}(y_i) + \frac{1}{2}\max_{y_{f\setminus i}}\left[\theta_f(y_f) - \sum_{j\in f\setminus i}\delta_{fj}(y_j)\right]$$

- Where: $\delta_i^{-f}(y_i) = \theta_i(y_i) + \sum_{g:i\in g, g\neq f}\delta_{gi}(y_i)$

- Update depends only on some of the fixed messages.



Max-Sum-Diffusion (MSD) Update

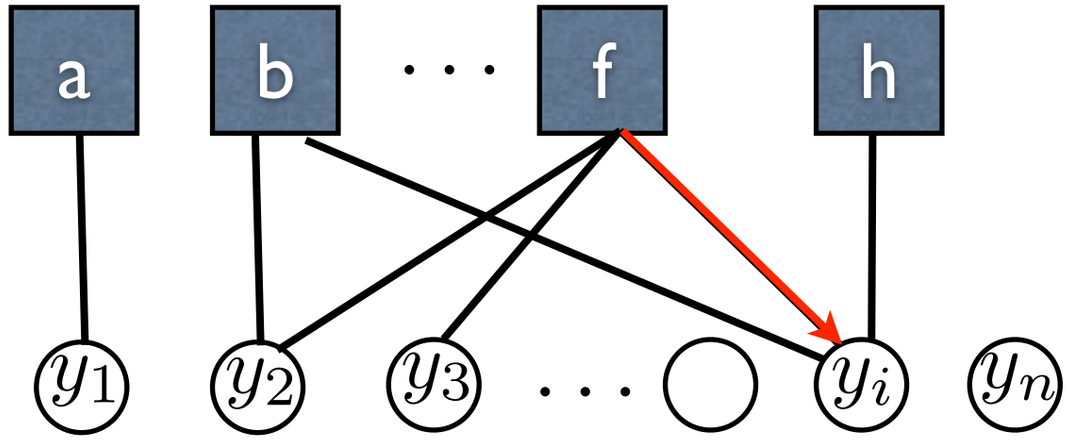
- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall y_i$

- Their optimal value given the others is:

$$\delta_{fi}(y_i) = -\frac{1}{2}\delta_i^{-f}(y_i) + \frac{1}{2}\max_{y_{f\setminus i}}\left[\theta_f(y_f) - \sum_{j\in f\setminus i}\delta_{fj}(y_j)\right]$$

- Where: $\delta_i^{-f}(y_i) = \theta_i(y_i) + \sum_{g:i\in g, g\neq f}\delta_{gi}(y_i)$

- Update depends only on some of the fixed messages.



Max-Sum-Diffusion (MSD) Update

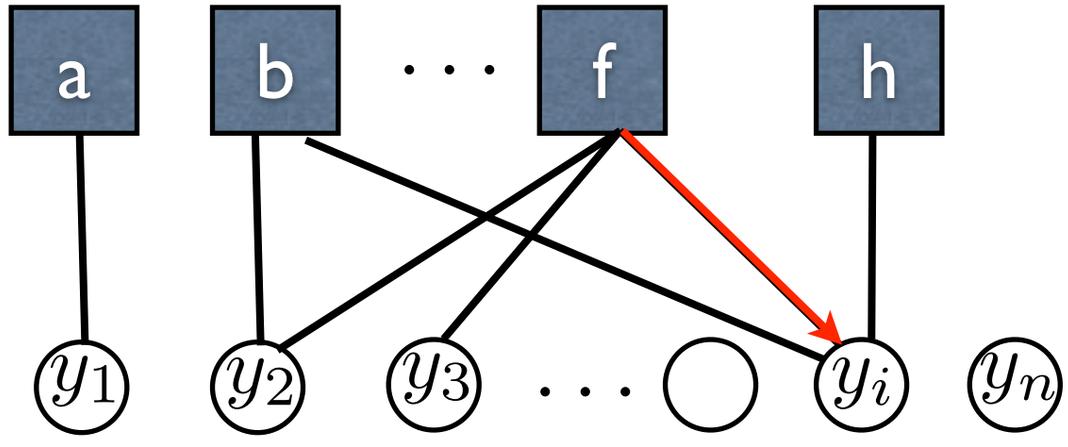
- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall y_i$

- Their optimal value given the others is:

$$\delta_{fi}(y_i) = -\frac{1}{2}\delta_i^{-f}(y_i) + \frac{1}{2}\max_{y_{f\setminus i}}\left[\theta_f(y_f) - \sum_{j\in f\setminus i}\delta_{fj}(y_j)\right]$$

- Where: $\delta_i^{-f}(y_i) = \theta_i(y_i) + \sum_{g:i\in g, g\neq f}\delta_{gi}(y_i)$

- Update depends only on some of the fixed messages.



Max-Sum-Diffusion (MSD) Update

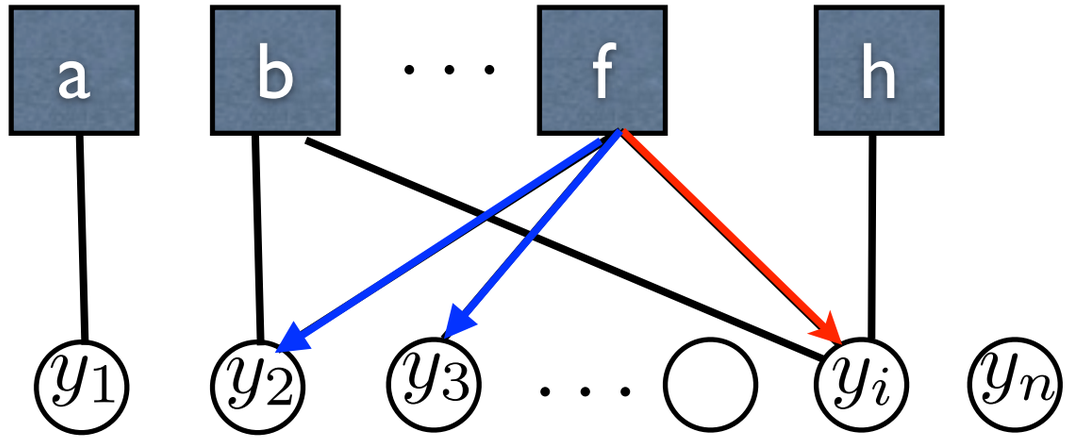
- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall y_i$

- Their optimal value given the others is:

$$\delta_{fi}(y_i) = -\frac{1}{2}\delta_i^{-f}(y_i) + \frac{1}{2}\max_{y_{f\setminus i}}\left[\theta_f(y_f) - \sum_{j\in f\setminus i}\delta_{fj}(y_j)\right]$$

- Where: $\delta_i^{-f}(y_i) = \theta_i(y_i) + \sum_{g:i\in g, g\neq f}\delta_{gi}(y_i)$

- Update depends only on some of the fixed messages.



Max-Sum-Diffusion (MSD) Update

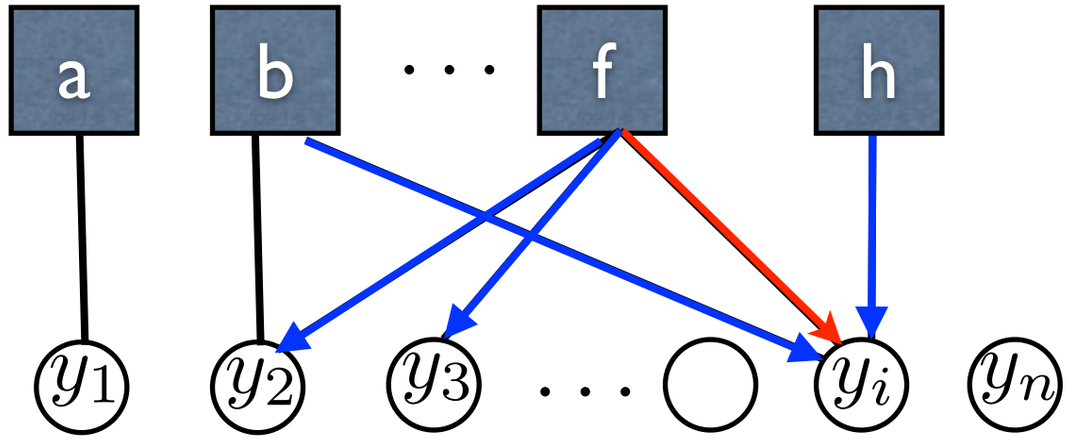
- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall y_i$

- Their optimal value given the others is:

$$\delta_{fi}(y_i) = -\frac{1}{2}\delta_i^{-f}(y_i) + \frac{1}{2}\max_{y_{f\setminus i}}\left[\theta_f(y_f) - \sum_{j\in f\setminus i}\delta_{fj}(y_j)\right]$$

- Where: $\delta_i^{-f}(y_i) = \theta_i(y_i) + \sum_{g:i\in g, g\neq f}\delta_{gi}(y_i)$

- Update depends only on some of the fixed messages.



Max-Sum-Diffusion (MSD) Update

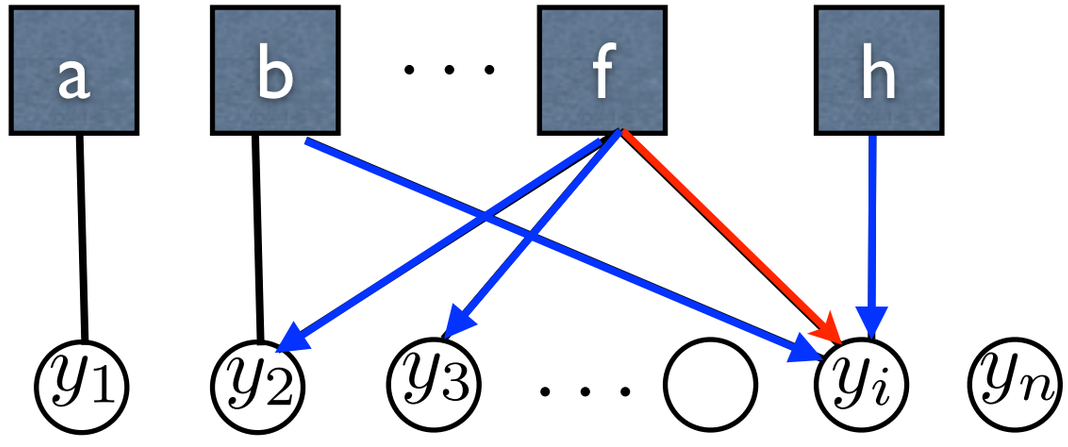
- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall y_i$

- Their optimal value given the others is:

$$\delta_{fi}(y_i) = -\frac{1}{2}\delta_i^{-f}(y_i) + \frac{1}{2}\max_{y_{f\setminus i}}\left[\theta_f(y_f) - \sum_{j\in f\setminus i}\delta_{fj}(y_j)\right]$$

- Where: $\delta_i^{-f}(y_i) = \theta_i(y_i) + \sum_{g:i\in g, g\neq f}\delta_{gi}(y_i)$

- Update depends only on some of the fixed messages.



Updated
Used in Update

MSD and Local Optimality

- The part of the objective that depends on $\delta_{fi}(y_i)$:

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

MSD and Local Optimality

- The part of the objective that depends on $\delta_{fi}(y_i)$:

$$\max_{y_f} \underbrace{\left[\theta_f(y_f) - \sum_{j \in f} \delta_{fj}(y_j)\right]}_{\bar{\theta}^{\delta}_f(y_f)} + \max_{y_i} \underbrace{\left[\theta_i(y_i) + \sum_{g:i \in g} \delta_{gi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- After the update we have: $\max_{y_{f \setminus i}} \bar{\theta}^{\delta}_f(y_f) = \bar{\theta}^{\delta}_i(y_i)$

MSD and Local Optimality

- The part of the objective that depends on $\delta_{fi}(y_i)$:

$$\max_{y_f}\underbrace{\left[\theta_f(y_f)-\sum_{j\in f}\delta_{fj}(y_j)\right]}_{\bar{\theta}^{\delta}_f(y_f)}+\max_{y_i}\underbrace{\left[\theta_i(y_i)+\sum_{g:i\in g}\delta_{gi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- After the update we have: $\max_{y_{f\setminus i}}\bar{\theta}^{\delta}_f(y_f)=\bar{\theta}^{\delta}_i(y_i)$
- The same y_i values maximize $\bar{\theta}^{\delta}_f(y_f)$ and $\bar{\theta}^{\delta}_i(y_i)$,

MSD and Local Optimality

- The part of the objective that depends on $\delta_{fi}(y_i)$:

$$\max_{y_f}\underbrace{\left[\theta_f(y_f)-\sum_{j\in f}\delta_{fj}(y_j)\right]}_{\bar{\theta}^{\delta}_f(y_f)}+\max_{y_i}\underbrace{\left[\theta_i(y_i)+\sum_{g:i\in g}\delta_{gi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- After the update we have: $\max_{y_{f\setminus i}}\bar{\theta}^{\delta}_f(y_f)=\bar{\theta}^{\delta}_i(y_i)$
- The same y_i values maximize $\bar{\theta}^{\delta}_f(y_f)$ and $\bar{\theta}^{\delta}_i(y_i)$,

Recall: $\hat{y}=[\hat{y}_1,\ldots,\hat{y}_n]$ is optimal if:

$$\hat{y}_i\in\arg\max_{y_i}\bar{\theta}^{\delta}_i(y_i) \quad \textit{and} \quad \hat{y}_f\in\arg\max_{y_f}\bar{\theta}^{\delta}_f(y_f)$$

MSD and Local Optimality

- The part of the objective that depends on $\delta_{fi}(y_i)$:

$$\max_{y_f}\underbrace{\left[\theta_f(y_f)-\sum_{j\in f}\delta_{fj}(y_j)\right]}_{\bar{\theta}^{\delta}_f(y_f)}+\max_{y_i}\underbrace{\left[\theta_i(y_i)+\sum_{g:i\in g}\delta_{gi}(y_i)\right]}_{\bar{\theta}^{\delta}_i(y_i)}$$

- After the update we have: $\max_{y_{f\setminus i}}\bar{\theta}^{\delta}_f(y_f)=\bar{\theta}^{\delta}_i(y_i)$
- The same y_i values maximize $\bar{\theta}^{\delta}_f(y_f)$ and $\bar{\theta}^{\delta}_i(y_i)$,

Recall: $\hat{y}=[\hat{y}_1,\ldots,\hat{y}_n]$ is optimal if:

$$\hat{y}_i\in\arg\max_{y_i}\bar{\theta}^{\delta}_i(y_i) \quad \textit{and} \quad \hat{y}_f\in\arg\max_{y_f}\bar{\theta}^{\delta}_f(y_f)$$

- There is a y_i assignment that satisfies this after the update.

MPLP Update

MPLP Update

- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall i, y_i$

MPLP Update

- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall i, y_i$
- Their optimal value given the others is:

MPLP Update

- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall i, y_i$
- Their optimal value given the others is:

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

MPLP Update

- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall i, y_i$
- Their optimal value given the others is:

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

- Where |f| is the number of variables in f (e.g., 2 for pairwise models).

MPLP Update

- Fix all except: $\delta_{fi}(y_i) \quad , \quad \forall i, y_i$
- Their optimal value given the others is:

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

- Where |f| is the number of variables in f (e.g., 2 for pairwise models).
- Must update for all i *simultaneously*.

MPLP Update

- Update is:

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

MPLP Update

- Update is:

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

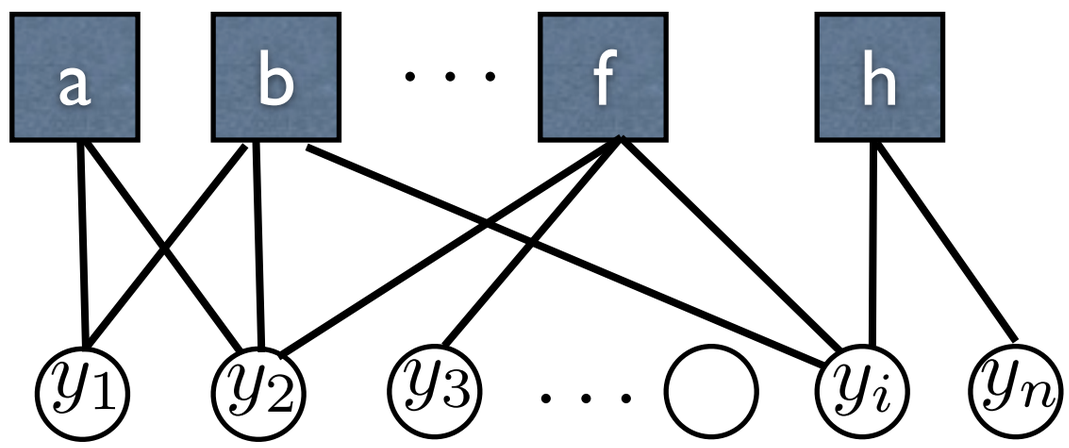
- Depends on a subset of the fixed messages.

MPLP Update

- Update is:

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

- Depends on a subset of the fixed messages.

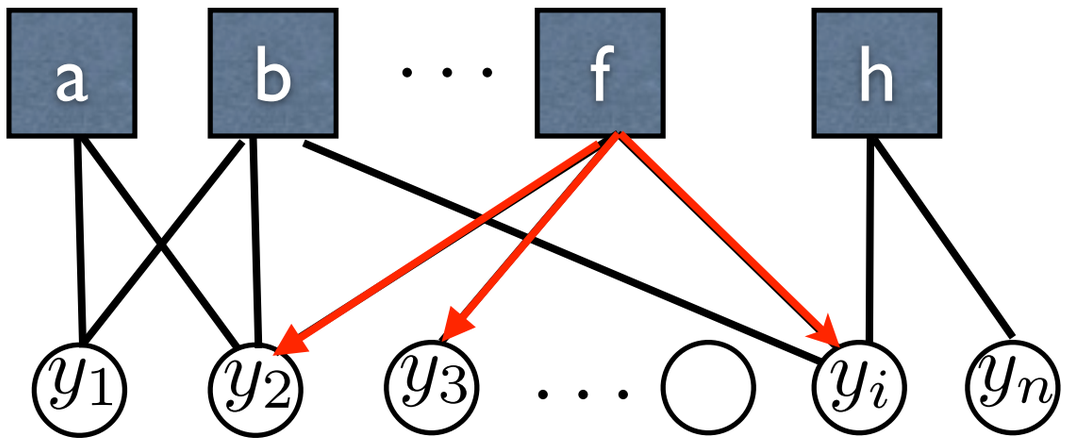


MPLP Update

- Update is:

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

- Depends on a subset of the fixed messages.

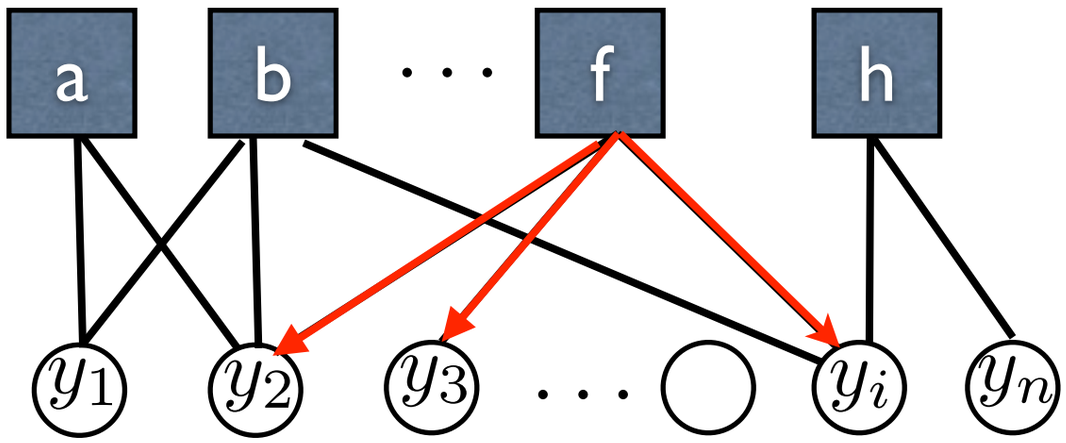


MPLP Update

- Update is:

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

- Depends on a subset of the fixed messages.



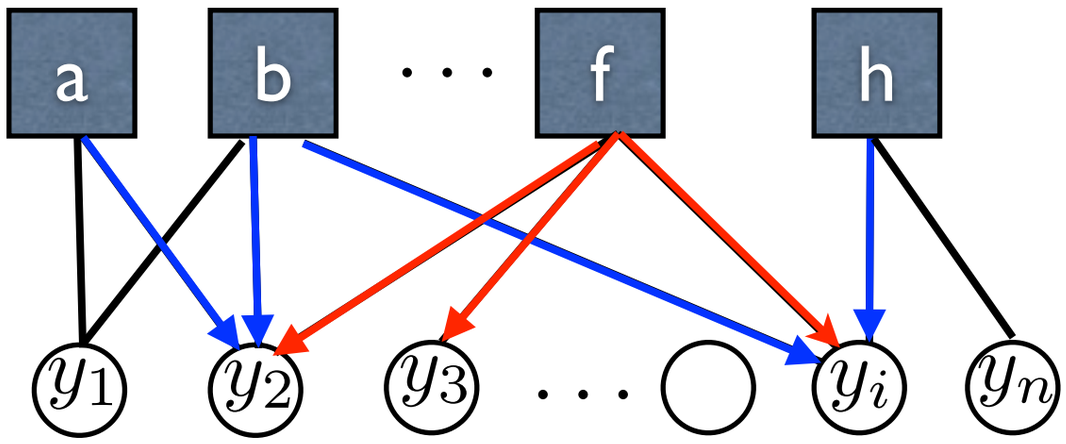
Updated

MPLP Update

- Update is:

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

- Depends on a subset of the fixed messages.



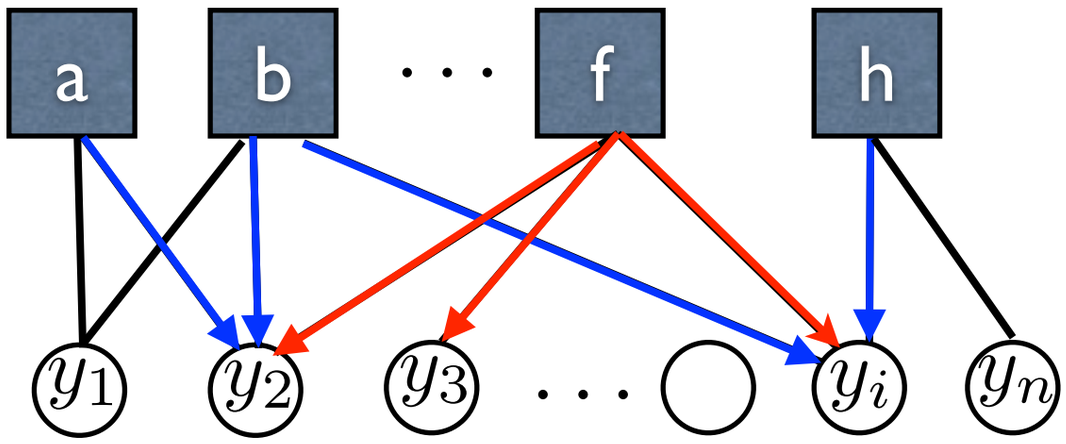
Updated

MPLP Update

- Update is:

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

- Depends on a subset of the fixed messages.



Updated

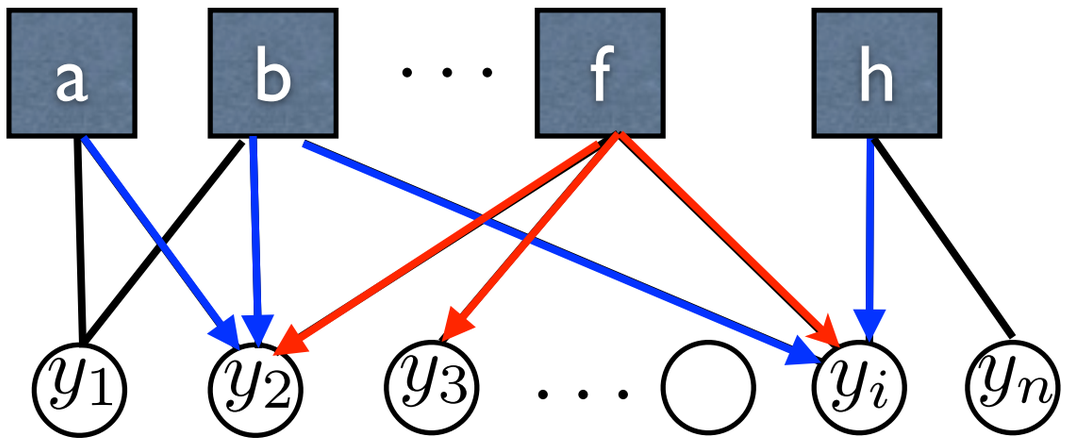
Used in Update

MPLP Update

- Update is:

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

- Depends on a subset of the fixed messages.

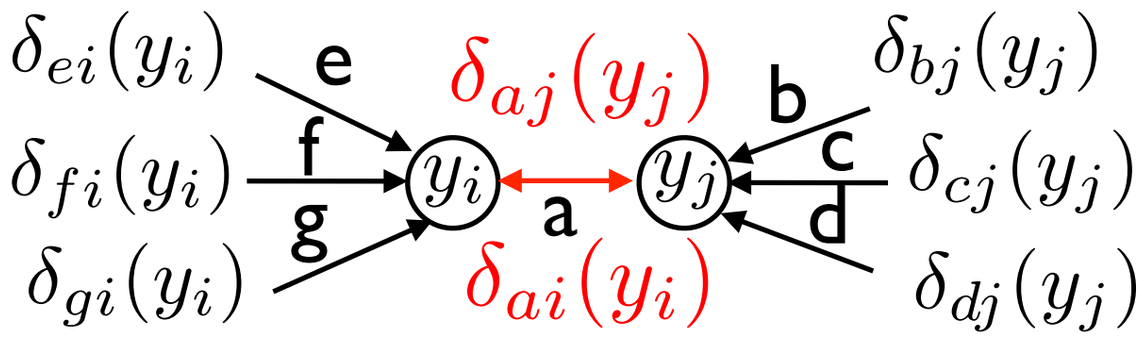


Updated

Used in Update

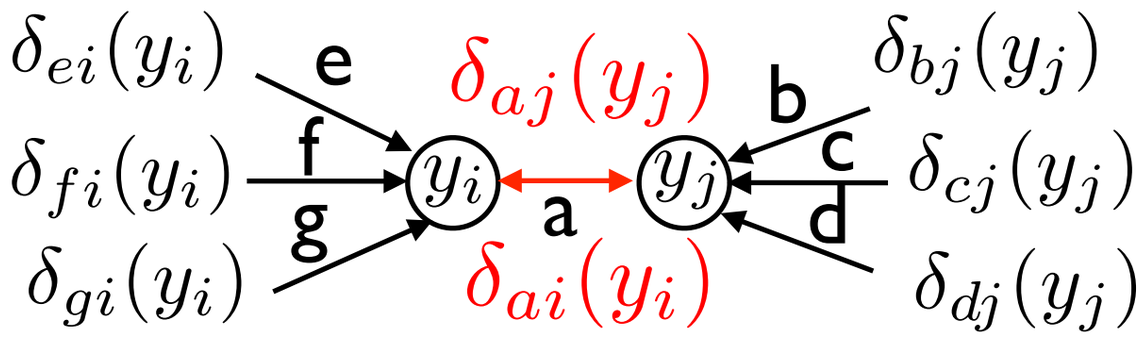
MPLP - Pairwise Case

- For each edge a we have two messages: $\delta_{ai}(y_i), \delta_{aj}(y_j)$



MPLP - Pairwise Case

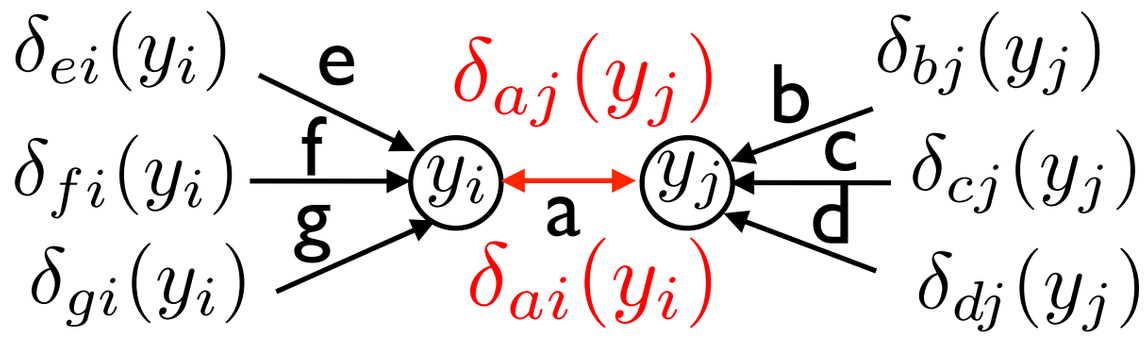
- For each edge a we have two messages: $\delta_{ai}(y_i), \delta_{aj}(y_j)$



- Update:

MPLP - Pairwise Case

- For each edge a we have two messages: $\delta_{ai}(y_i), \delta_{aj}(y_j)$

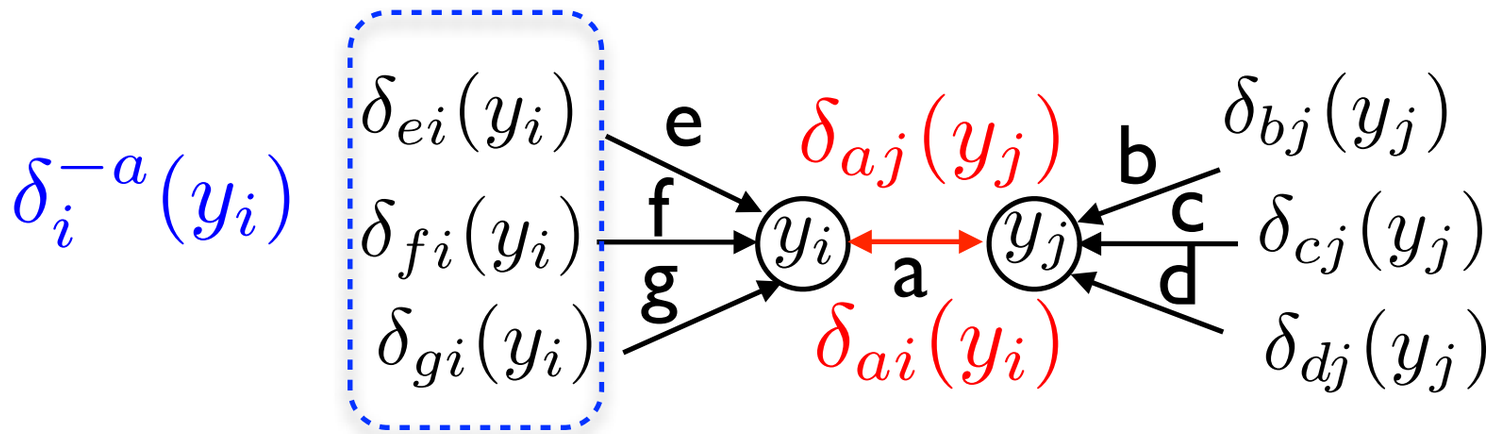


- Update:

$$\delta_{ai}(y_i) = -\frac{1}{2}\delta_i^{-a}(y_i) + \frac{1}{2}\max_{y_j}\left[\theta_a(y_i, y_j) + \delta_j^{-a}(y_j)\right]$$

MPLP - Pairwise Case

- For each edge a we have two messages: $\delta_{ai}(y_i), \delta_{aj}(y_j)$

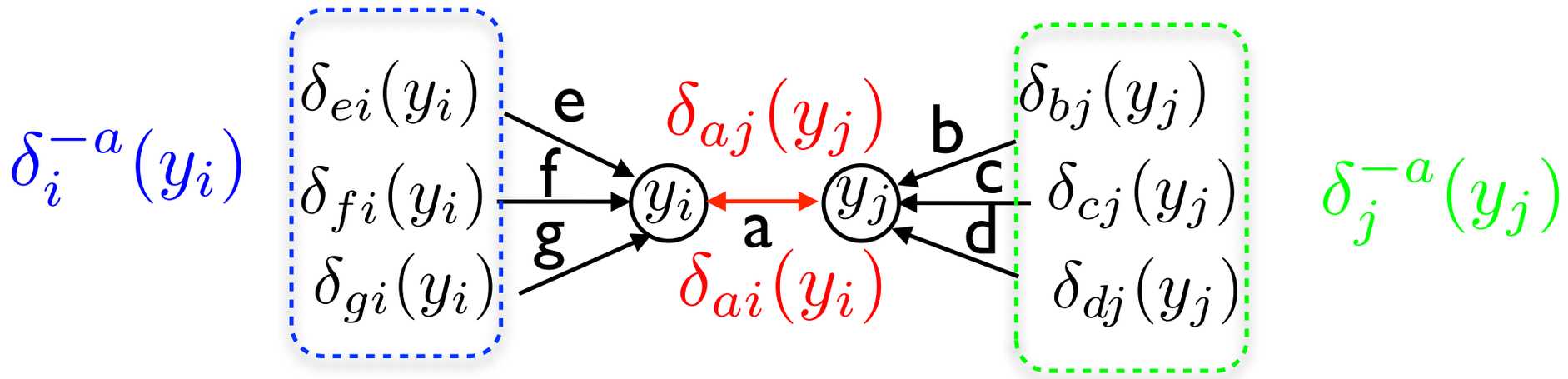


- Update:

$$\delta_{ai}(y_i) = -\frac{1}{2}\delta_i^{-a}(y_i) + \frac{1}{2}\max_{y_j}\left[\theta_a(y_i, y_j) + \delta_j^{-a}(y_j)\right]$$

MPLP - Pairwise Case

- For each edge a we have two messages: $\delta_{ai}(y_i), \delta_{aj}(y_j)$

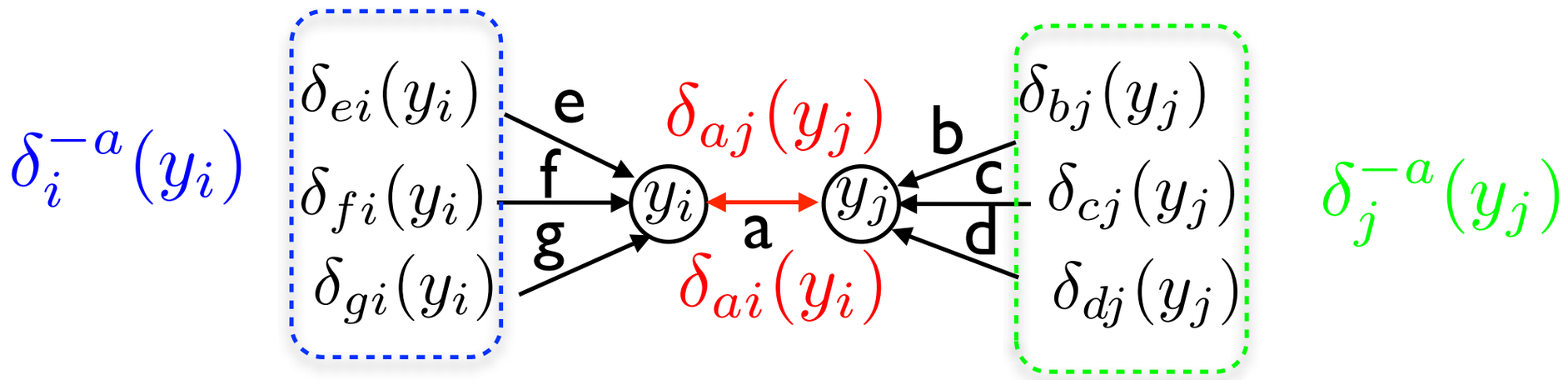


- Update:

$$\delta_{ai}(y_i) = -\frac{1}{2}\delta_i^{-a}(y_i) + \frac{1}{2}\max_{y_j}\left[\theta_a(y_i, y_j) + \delta_j^{-a}(y_j)\right]$$

MPLP - Pairwise Case

- For each edge a we have two messages: $\delta_{ai}(y_i), \delta_{aj}(y_j)$

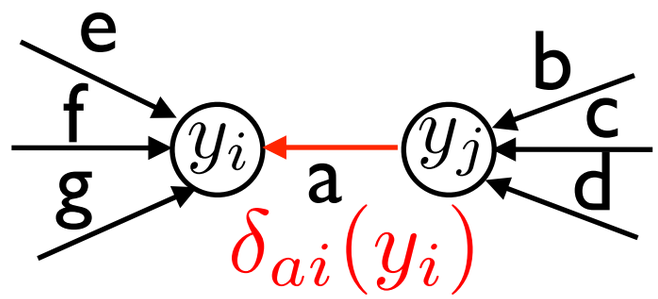


- Update:

$$\delta_{ai}(y_i) = -\frac{1}{2}\delta_i^{-a}(y_i) + \frac{1}{2}\max_{y_j}\left[\theta_a(y_i, y_j) + \delta_j^{-a}(y_j)\right]$$

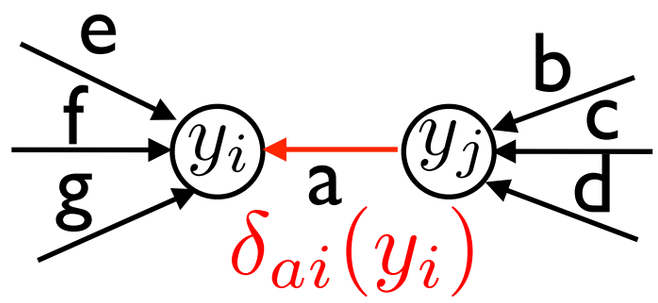
Max Product Update!

MPLP and Max-Product



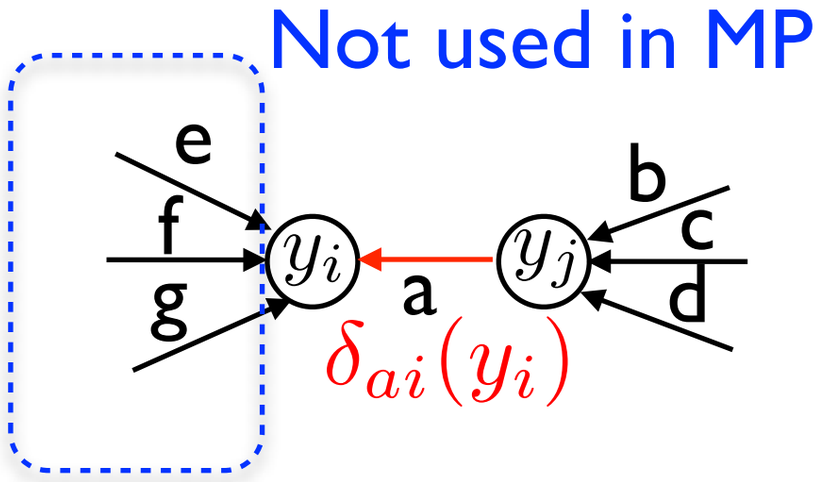
MPLP and Max-Product

- Very similar in form
- MPLP includes "backward" messages



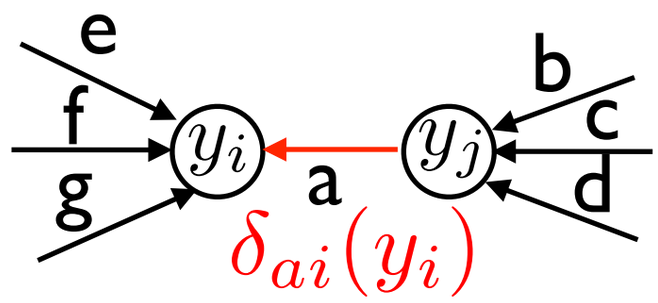
MPLP and Max-Product

- Very similar in form
- MPLP includes "backward" messages



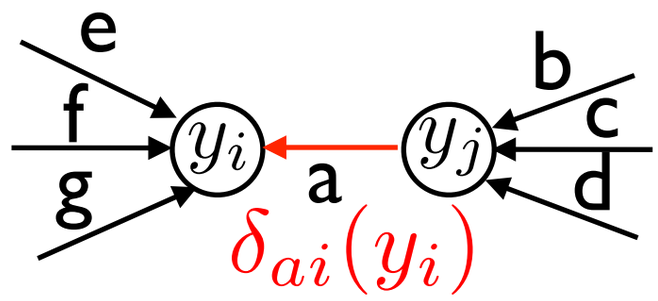
MPLP and Max-Product

- Very similar in form
- MPLP includes "backward" messages



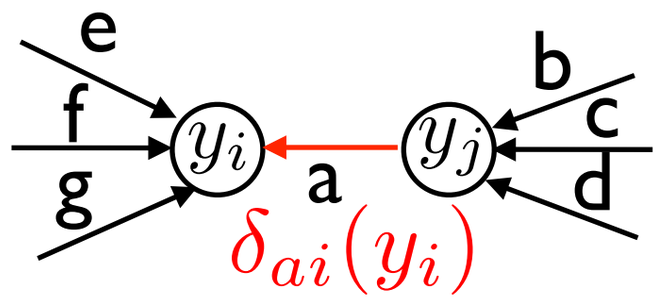
MPLP and Max-Product

- Very similar in form
- MPLP includes "backward" messages
- MPLP decreases dual objective



MPLP and Max-Product

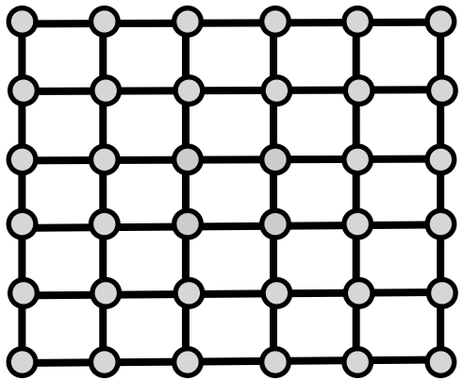
- Very similar in form
- MPLP includes "backward" messages
- MPLP decreases dual objective



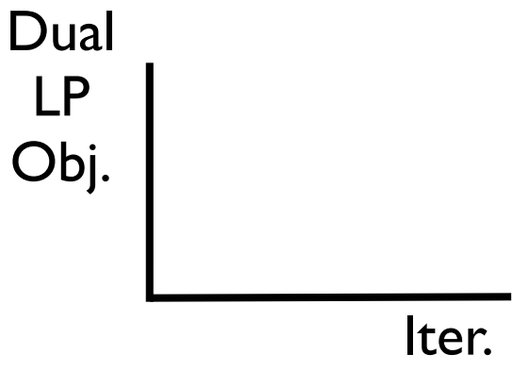
- When LP is exact, max product will have a fixed point corresponding to the LP optimum, but max-product may still not converge to it *[Weiss et al. 07, Bayati et al 08., Tarlow et al. 11]*.

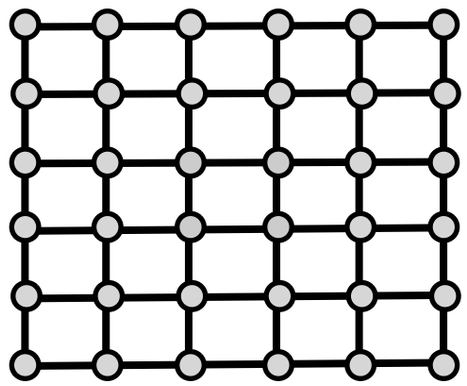
MPLP On Pairwise

MPLP On Pairwise

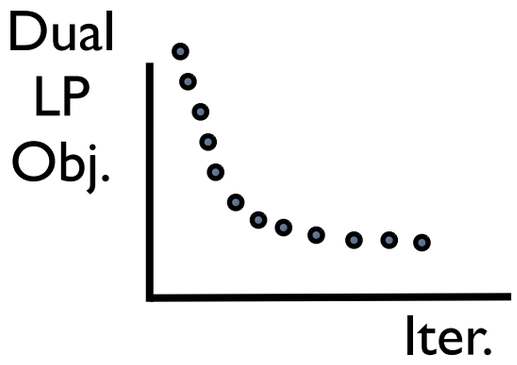


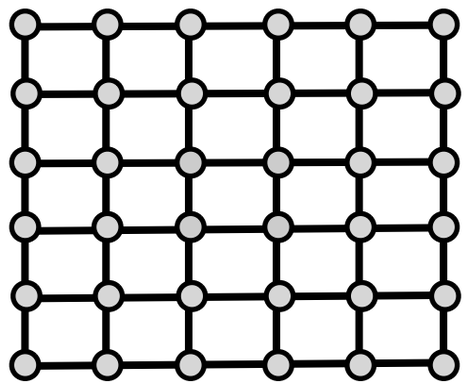
MPLP On Pairwise



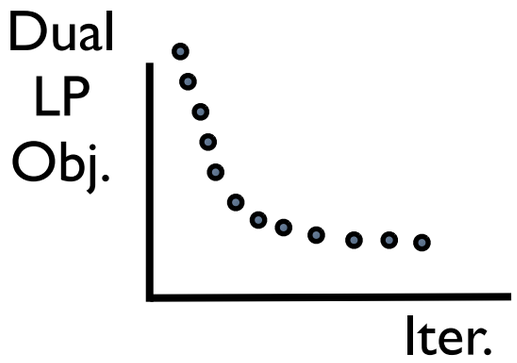


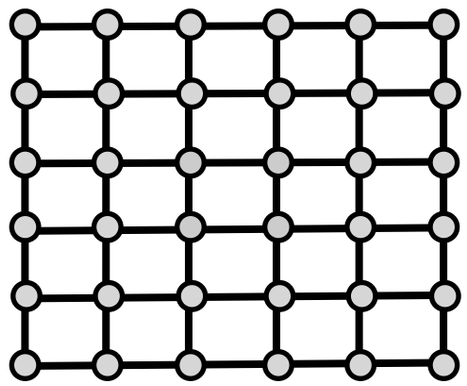
MPLP On Pairwise





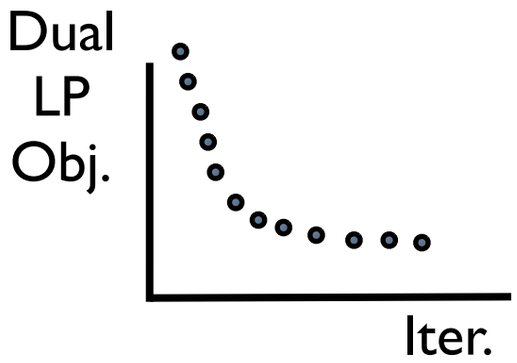
MPLP On Pairwise

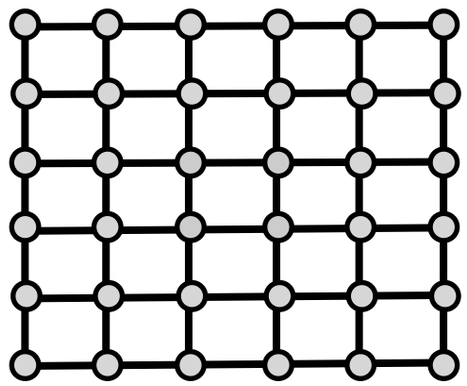




Decoded values

MPLP On Pairwise

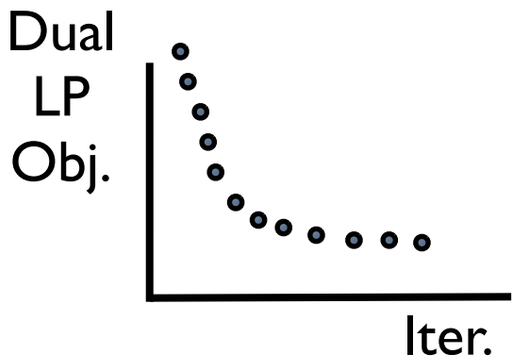


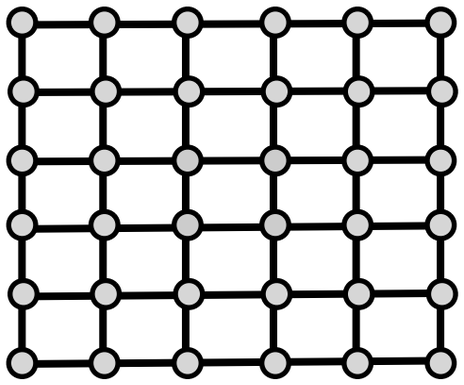


Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise

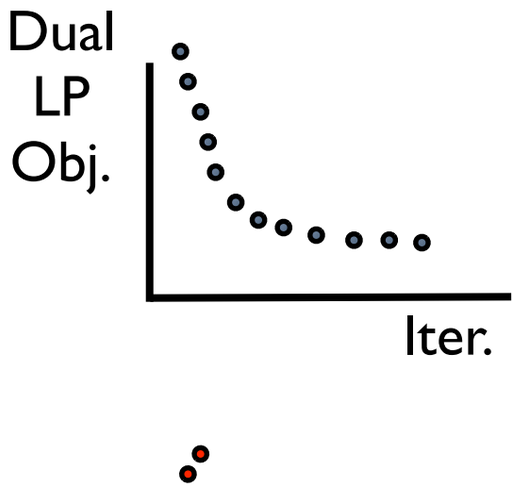


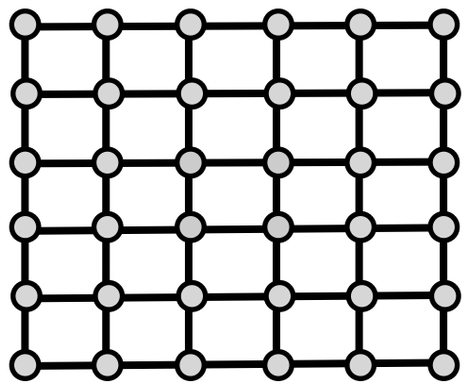


Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise

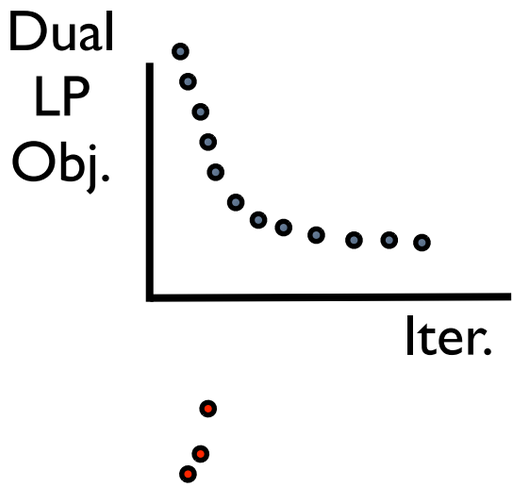


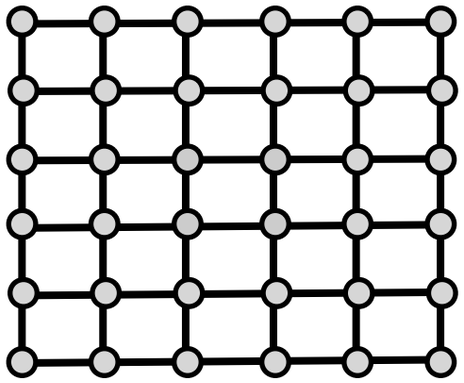


Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise

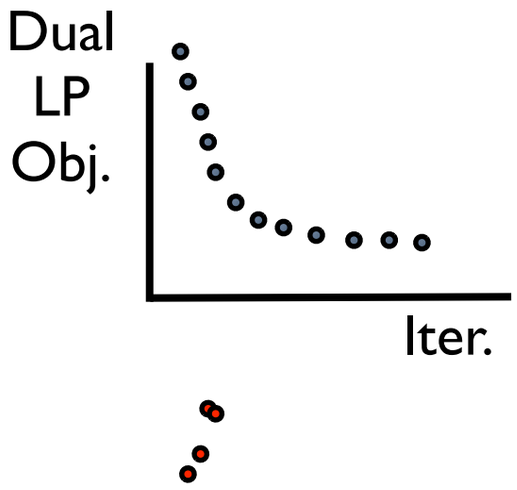


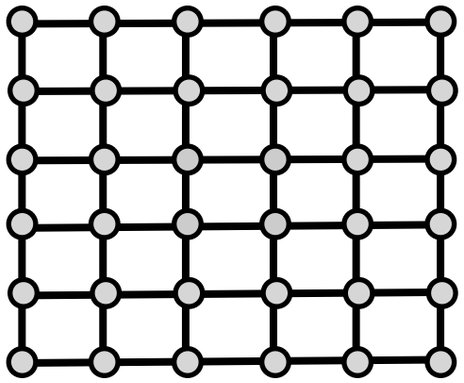


Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise

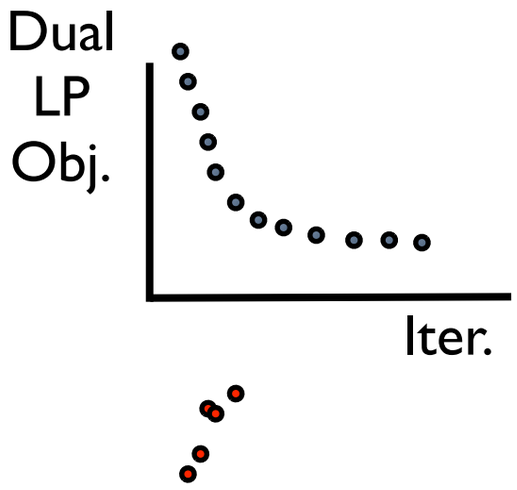


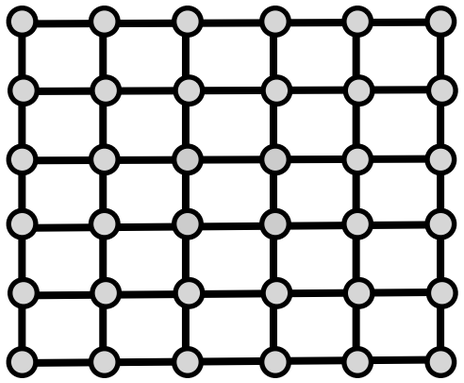


Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise

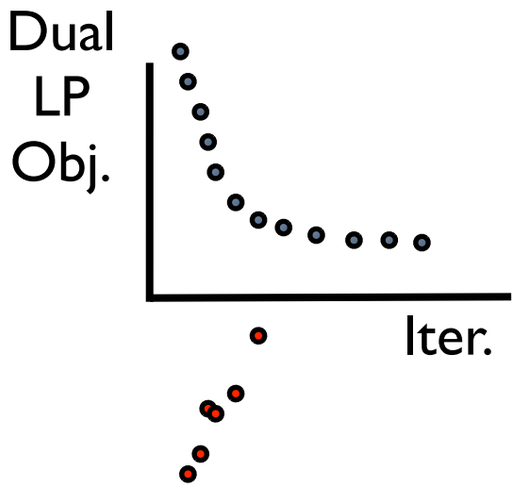


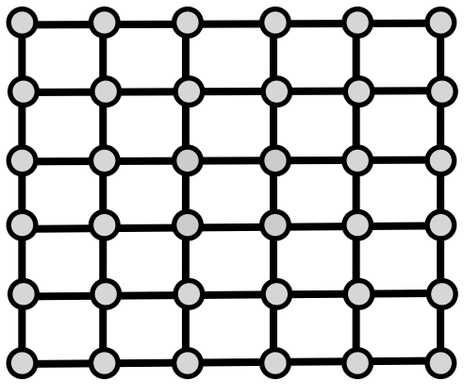


Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise

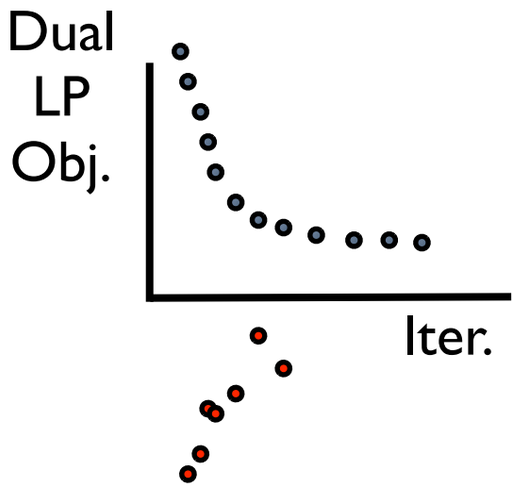


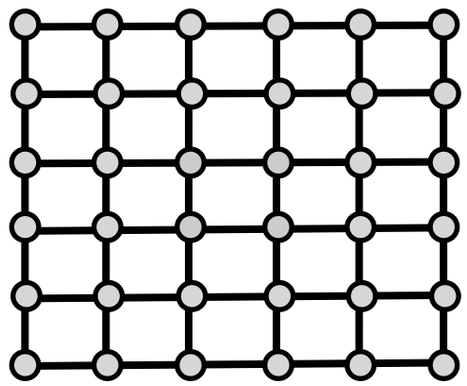


Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise

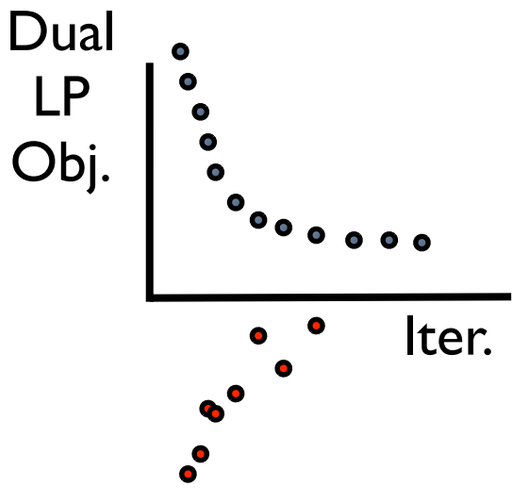


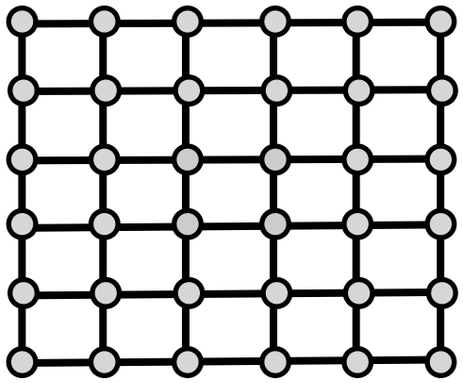


Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise

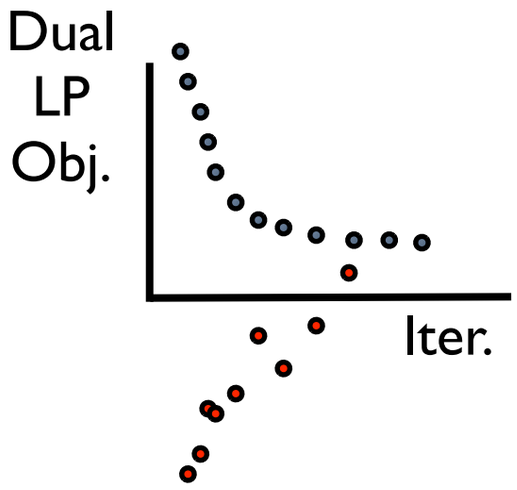


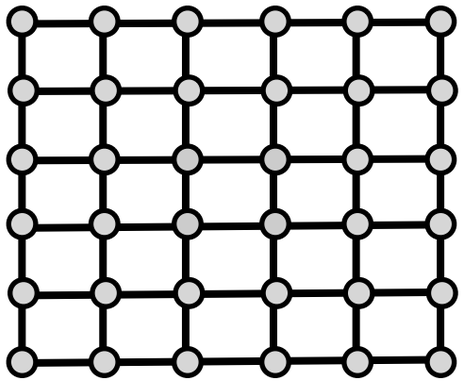


Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise

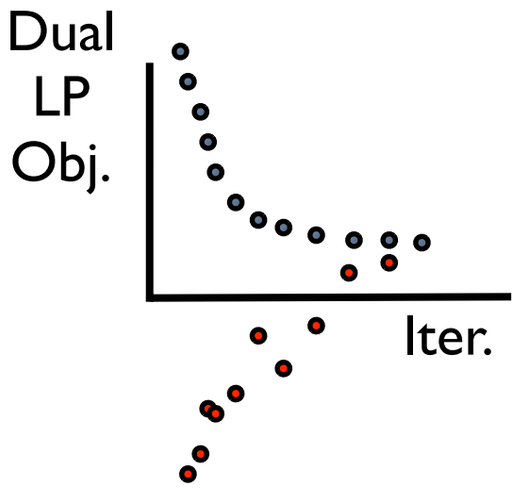


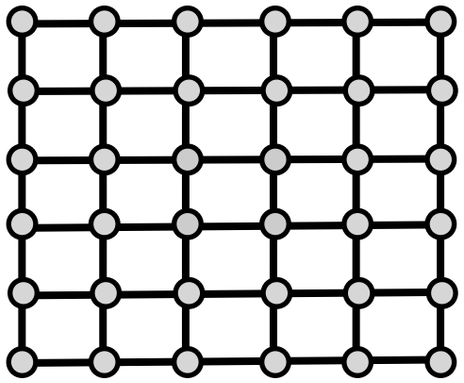


Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise

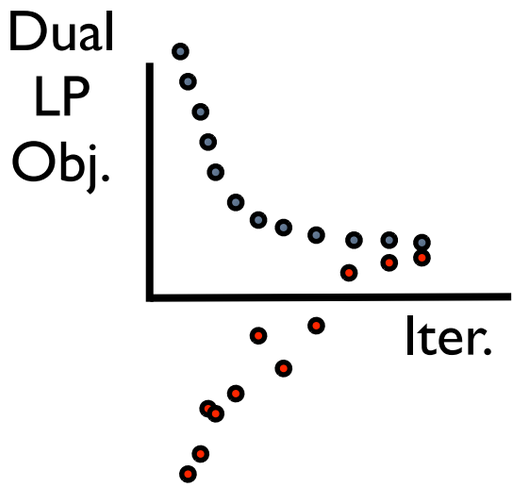


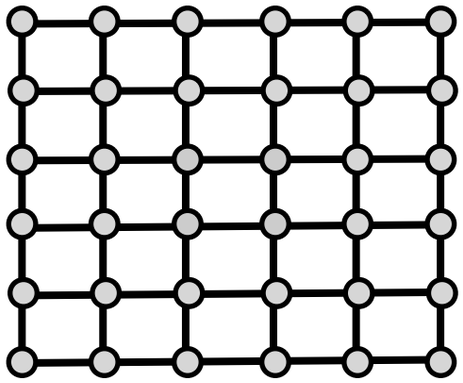


Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise



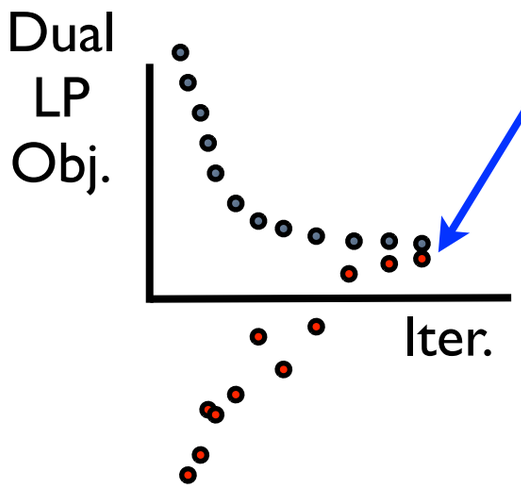


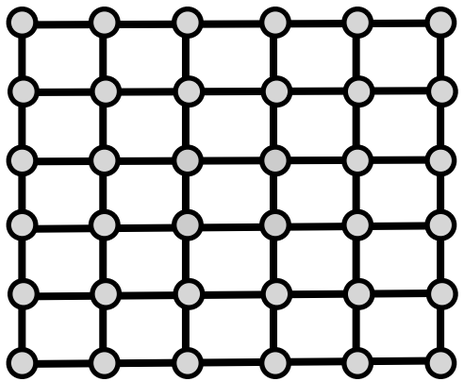
Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

MPLP On Pairwise

Can only happen if LP is exact





Decoded values

$$\arg\max_{y_i} \bar{\theta}_i^{\delta}(y_i)$$

Coordinate Descent - Computational Aspects

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|} \max_{y_{f \setminus i}} \left[\theta_f(y_f) + \sum_{j \in f} \delta_j^{-f}(y_j) \right]$$

Main computational load

Coordinate Descent - Computational Aspects

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|}\max_{y_{f\setminus i}}\left[\theta_f(y_f) + \sum_{j\in f}\delta_j^{-f}(y_j)\right]$$

Main computational load

- For example if f={1,2,3}:

Coordinate Descent - Computational Aspects

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|}\max_{y_{f\setminus i}}\left[\theta_f(y_f) + \sum_{j\in f}\delta_j^{-f}(y_j)\right]$$

Main computational load

- For example if f={1,2,3}:

For $\delta_{f1}(y_1)$ get $\max_{y_2,y_3}\left[\theta_f(y_1,y_2,y_3) + \delta_1^{-f}(y_1) + \delta_2^{-f}(y_2) + \delta_3^{-f}(y_3)\right]$

Coordinate Descent - Computational Aspects

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|}\max_{y_{f\setminus i}}\left[\theta_f(y_f) + \sum_{j\in f}\delta_j^{-f}(y_j)\right]$$

Main computational load

- For example if f={1,2,3}:

For $\delta_{f1}(y_1)$ get $\max_{y_2,y_3}\left[\theta_f(y_1,y_2,y_3) + \delta_1^{-f}(y_1) + \delta_2^{-f}(y_2) + \delta_3^{-f}(y_3)\right]$

For $\delta_{f2}(y_2)$ get $\max_{y_1,y_3}\left[\theta_f(y_1,y_2,y_3) + \delta_1^{-f}(y_1) + \delta_2^{-f}(y_2) + \delta_3^{-f}(y_3)\right]$

Coordinate Descent - Computational Aspects

$$\delta_{fi}(y_i) = -\delta_i^{-f}(y_i) + \frac{1}{|f|}\max_{y_{f\setminus i}}\left[\theta_f(y_f) + \sum_{j\in f}\delta_j^{-f}(y_j)\right]$$

Main computational load

- For example if f={1,2,3}:

For $\delta_{f1}(y_1)$ get $\max_{y_2,y_3}\left[\theta_f(y_1,y_2,y_3) + \delta_1^{-f}(y_1) + \delta_2^{-f}(y_2) + \delta_3^{-f}(y_3)\right]$

For $\delta_{f2}(y_2)$ get $\max_{y_1,y_3}\left[\theta_f(y_1,y_2,y_3) + \delta_1^{-f}(y_1) + \delta_2^{-f}(y_2) + \delta_3^{-f}(y_3)\right]$

- These are known as *max-marginals*.
- Similar complexity to maximizing $\theta_f(y_f)$. May be up to O(n) slower.

Example - Cardinality Potential

- Say $y_i \in \{0, 1\}$ and we have a potential:

$$\theta_{\mathrm{C}}(y_1, \ldots, y_n) = h\left(\sum_i y_i\right)$$

Example - Cardinality Potential

- Say $y_i \in \{0, 1\}$ and we have a potential:

$$\theta_{\mathrm{C}}(y_1, \ldots, y_n) = h\left(\sum_i y_i\right)$$

- For example:

$$h(z) = \begin{cases} -\infty & z = 0 \\ 0 & z \geq 1 \end{cases}$$

Example - Cardinality Potential

- Say $y_i \in \{0, 1\}$ and we have a potential:

$$\theta_{\mathrm{C}}(y_1, \ldots, y_n) = h\left(\sum_i y_i\right)$$

- For example:

$$h(z) = \begin{cases} -\infty & z = 0 \\ 0 & z \geq 1 \end{cases}$$

At least one y_i should be 1.

Example - Cardinality Potential

- Say $y_i \in \{0, 1\}$ and we have a potential:

$$\theta_{\mathrm{C}}(y_1, \ldots, y_n) = h\left(\sum_i y_i\right)$$

- For example:

$$h(z) = \begin{cases} -\infty & z = 0 \\ 0 & z \geq 1 \end{cases}$$

At least one y_i should be 1.

- Usage: control the number of times a label appears:

Example - Cardinality Potential

- Say $y_i \in \{0, 1\}$ and we have a potential:

$$\theta_{\mathrm{C}}(y_1, \ldots, y_n) = h\left(\sum_i y_i\right)$$

- For example:

$$h(z) = \begin{cases} -\infty & z = 0 \\ 0 & z \geq 1 \end{cases}$$

At least one y_i should be 1.

- Usage: control the number of times a label appears:
  - Number of verbs in a sentence.

Example - Cardinality Potential

- Say $y_i \in \{0, 1\}$ and we have a potential:

$$\theta_{\mathrm{C}}(y_1, \ldots, y_n) = h\left(\sum_i y_i\right)$$

- For example:

$$h(z) = \begin{cases} -\infty & z = 0 \\ 0 & z \geq 1 \end{cases}$$

At least one y_i should be 1.

- Usage: control the number of times a label appears:
  - Number of verbs in a sentence.
  - Number of pixels in a segment.

Cardinality Potential - Max Marginals

- To calculate the MPLP message to variable y_i, we need to solve a problem of the following form:

$$m_i(y_i) = \max_{y_{V \setminus i}} \left[\theta_c(y_1, \ldots, y_n) + \sum_i v_i y_i \right]$$

Cardinality Potential - Max Marginals

- To calculate the MPLP message to variable y_i, we need to solve a problem of the following form:

$$m_i(y_i) = \max_{y_{V \setminus i}} \left[\theta_c(y_1, \ldots, y_n) + \sum_i v_i y_i \right]$$

- For all i=1,...,n

Cardinality Potential - Max Marginals

- To calculate the MPLP message to variable y_i, we need to solve a problem of the following form:

$$m_i(y_i) = \max_{y_{V \setminus i}} \left[\theta_c(y_1, \ldots, y_n) + \sum_i v_i y_i \right]$$

- For all i=1,...,n
  - Solution: find largest, and second largest v_i elements.

Cardinality Potential - Max Marginals

- To calculate the MPLP message to variable y_i, we need to solve a problem of the following form:

$$m_i(y_i) = \max_{y_{V \setminus i}} \left[\theta_c(y_1, \ldots, y_n) + \sum_i v_i y_i \right]$$

- For all i=1,...,n
  - Solution: find largest, and second largest v_i elements.
  - Assume they are v_1 and v_2. Then:

Cardinality Potential - Max Marginals

- To calculate the MPLP message to variable y_i, we need to solve a problem of the following form:

$$m_i(y_i) = \max_{y_{V \setminus i}} \left[\theta_c(y_1, \ldots, y_n) + \sum_i v_i y_i \right]$$

- For all i=1,...,n
  - Solution: find largest, and second largest v_i elements.
  - Assume they are v_1 and v_2. Then:

$$m_1(y_1) = \begin{bmatrix} v_2 \\ v_1 \end{bmatrix}$$

Cardinality Potential - Max Marginals

- To calculate the MPLP message to variable y_i, we need to solve a problem of the following form:

$$m_i(y_i) = \max_{y_{V \setminus i}} \left[\theta_c(y_1, \ldots, y_n) + \sum_i v_i y_i \right]$$

- For all i=1,...,n
  - Solution: find largest, and second largest v_i elements.
  - Assume they are v_1 and v_2. Then:

$$m_1(y_1) = \begin{bmatrix} v_2 \\ v_1 \end{bmatrix} \qquad m_i(y_i) = \begin{bmatrix} v_1 \\ v_i \end{bmatrix}$$

Cardinality Potential - Max Marginals

- To calculate the MPLP message to variable y_i, we need to solve a problem of the following form:

$$m_i(y_i) = \max_{y_{V \setminus i}} \left[\theta_c(y_1, \ldots, y_n) + \sum_i v_i y_i \right]$$

- For all i=1,...,n
  - Solution: find largest, and second largest v_i elements.
  - Assume they are v_1 and v_2. Then:

$$m_1(y_1) = \begin{bmatrix} v_2 \\ v_1 \end{bmatrix} \qquad m_i(y_i) = \begin{bmatrix} v_1 \\ v_i \end{bmatrix}$$

  - O(n) complexity for all messages.
  - Extensions to other counting potentials possible (see HopMap algorithms from *[Tarlow et al. 10]*).

Extensions & Other approaches

Extensions & Other approaches

- Optimize over larger sets of dual variables. e.g., those corresponding to trees in pairwise models *[Sontag & Jaakkola 09]*.

Extensions & Other approaches

- Optimize over larger sets of dual variables. e.g., those corresponding to trees in pairwise models *[Sontag & Jaakkola 09]*.
- Accelerated gradient methods *[Jojic et al. 10, Savchynskyy et al. 10]*.

Extensions & Other approaches

- Optimize over larger sets of dual variables. e.g., those corresponding to trees in pairwise models *[Sontag & Jaakkola 09]*.
- Accelerated gradient methods *[Jojic et al. 10, Savchynskyy et al. 10]*.
- For binary variables, the QPBO algorithm solves the LP using mincut, also for non submodular potentials *[Kolmogorov and Rother 07, Hammer et al 84]*).

Extensions & Other approaches

- Optimize over larger sets of dual variables. e.g., those corresponding to trees in pairwise models *[Sontag & Jaakkola 09]*.
- Accelerated gradient methods *[Jojic et al. 10, Savchynskyy et al. 10]*.
- For binary variables, the QPBO algorithm solves the LP using mincut, also for non submodular potentials *[Kolmogorov and Rother 07, Hammer et al 84]*).
- Graph cut methods *[Kumar et al 11,Boykov et al.01]*.

Extensions & Other approaches

- Optimize over larger sets of dual variables. e.g., those corresponding to trees in pairwise models *[Sontag & Jaakkola 09]*.
- Accelerated gradient methods *[Jojic et al. 10, Savchynskyy et al. 10]*.
- For binary variables, the QPBO algorithm solves the LP using mincut, also for non submodular potentials *[Kolmogorov and Rother 07, Hammer et al 84]*).
- Graph cut methods *[Kumar et al 11,Boykov et al.01]*.
- Alternating direction method of multipliers (ADMM) for solving the LP (see survey by Boyd et al). Recent works: [*Martins et al 11, Meshi & Globerson 11*].

Extensions & Other approaches

- Optimize over larger sets of dual variables. e.g., those corresponding to trees in pairwise models *[Sontag & Jaakkola 09]*.
- Accelerated gradient methods *[Jojic et al. 10, Savchynskyy et al. 10]*.
- For binary variables, the QPBO algorithm solves the LP using mincut, also for non submodular potentials *[Kolmogorov and Rother 07, Hammer et al 84]*).
- Graph cut methods *[Kumar et al 11,Boykov et al.01]*.
- Alternating direction method of multipliers (ADMM) for solving the LP (see survey by Boyd et al). Recent works: [*Martins et al 11, Meshi & Globerson 11*].
- Primal solution using proximal point *[Ravikumar et al. 08]*.

Coordinate Descent - Caveat

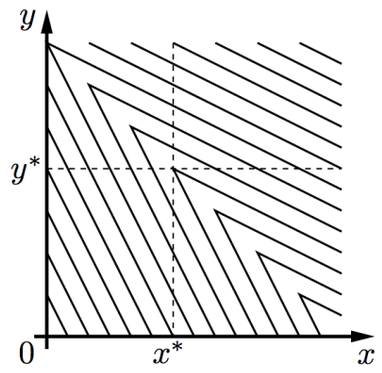
Coordinate Descent - Caveat

- The objective is non-smooth.

Coordinate Descent - Caveat

[Werner & Shekhovtsov, 2007]

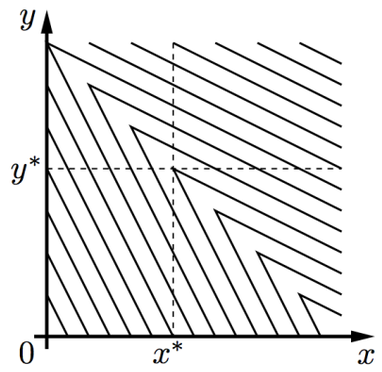
- The objective is non-smooth.



Coordinate Descent - Caveat

[Werner & Shekhovtsov, 2007]

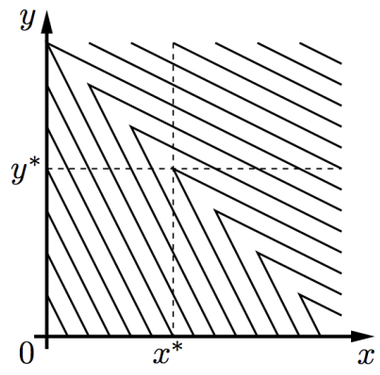
- The objective is non-smooth.
- Coordinate descent may get stuck in "corners"



Coordinate Descent - Caveat

[Werner & Shekhovtsov, 2007]

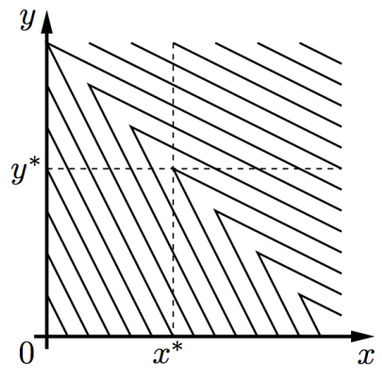
- The objective is non-smooth.
- Coordinate descent may get stuck in "corners"
- No coordinate update improves objective, but we are not at an optimum *[Kolmogorov 04]*.



Coordinate Descent - Caveat

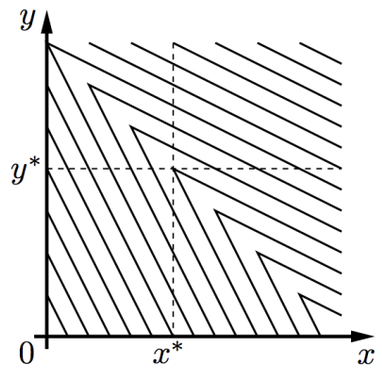
[Werner & Shekhovtsov, 2007]

- The objective is non-smooth.
- Coordinate descent may get stuck in "corners"
- No coordinate update improves objective, but we are not at an optimum *[Kolmogorov 04]*.
- Simplest solution: change max to soft-max:



Coordinate Descent - Caveat

[Werner & Shekhovtsov, 2007]



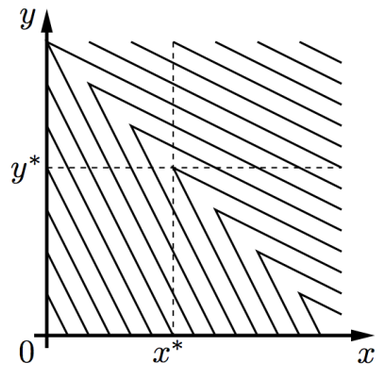
- The objective is non-smooth.
- Coordinate descent may get stuck in "corners"
- No coordinate update improves objective, but we are not at an optimum *[Kolmogorov 04]*.
- Simplest solution: change max to soft-max:

$$\max_i f_i \quad \sim \quad \epsilon \log \sum_i e^{\frac{f_i}{\epsilon}}$$

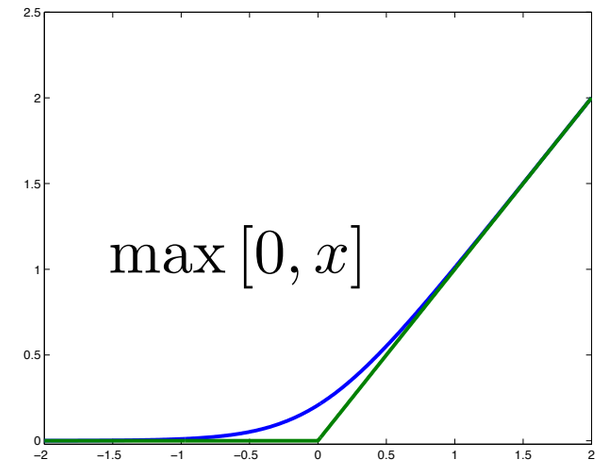
Coordinate Descent - Caveat

[Werner & Shekhovtsov, 2007]

- The objective is non-smooth.
- Coordinate descent may get stuck in "corners"
- No coordinate update improves objective, but we are not at an optimum *[Kolmogorov 04]*.
- Simplest solution: change max to soft-max:



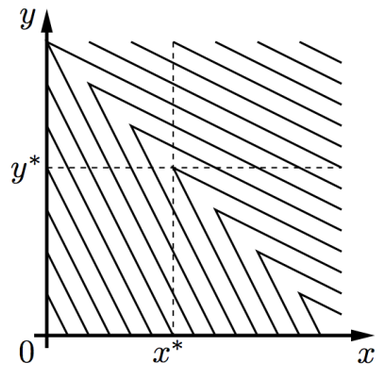
$$\max_i f_i \quad \sim \quad \epsilon \log \sum_i e^{\frac{f_i}{\epsilon}}$$



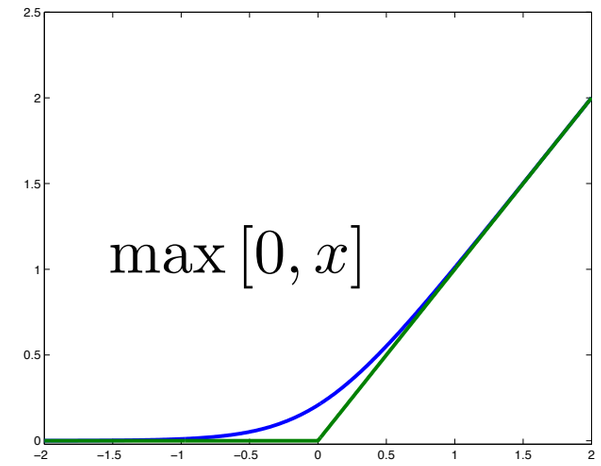
Coordinate Descent - Caveat

[Werner & Shekhovtsov, 2007]

- The objective is non-smooth.
- Coordinate descent may get stuck in "corners"
- No coordinate update improves objective, but we are not at an optimum *[Kolmogorov 04]*.
- Simplest solution: change max to soft-max:



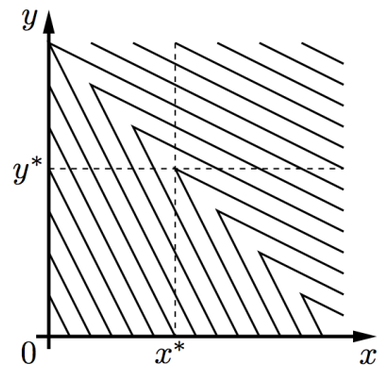
$$\max_i f_i \quad \sim \quad \epsilon \log \sum_i e^{\frac{f_i}{\epsilon}}$$



- As $\epsilon \longrightarrow 0$ becomes equal to max.

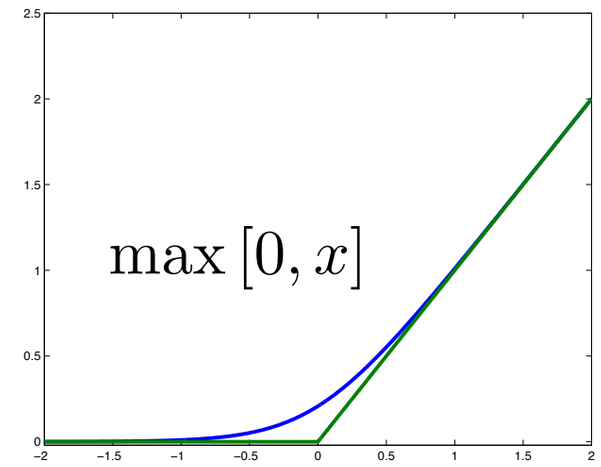
Coordinate Descent - Caveat

[Werner & Shekhovtsov, 2007]



- The objective is non-smooth.
- Coordinate descent may get stuck in "corners"
- No coordinate update improves objective, but we are not at an optimum *[Kolmogorov 04]*.
- Simplest solution: change max to soft-max:

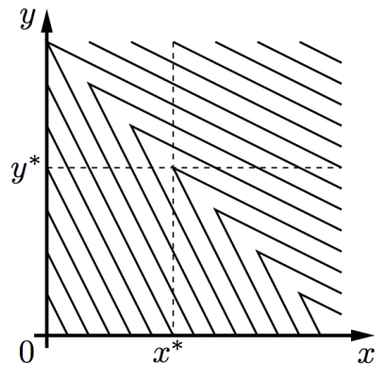
$$\max_i f_i \quad \sim \quad \epsilon \log \sum_i e^{\frac{f_i}{\epsilon}}$$



- As $\epsilon \to 0$ becomes equal to max.
- Is an $O(\epsilon)$ approximation (and upper bound) to the original objective.

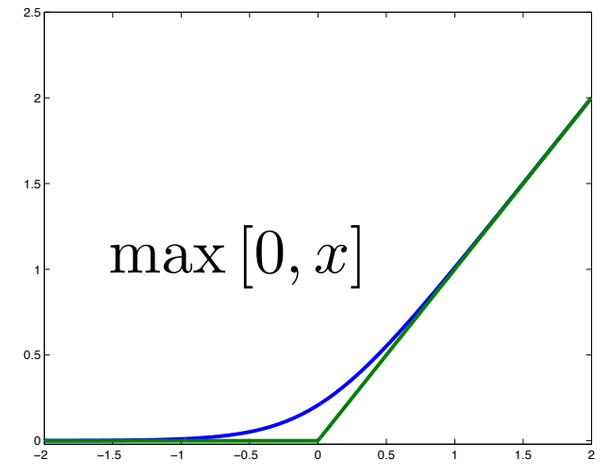
Coordinate Descent - Caveat

[Werner & Shekhovtsov, 2007]



- The objective is non-smooth.
- Coordinate descent may get stuck in "corners"
- No coordinate update improves objective, but we are not at an optimum *[Kolmogorov 04]*.
- Simplest solution: change max to soft-max:

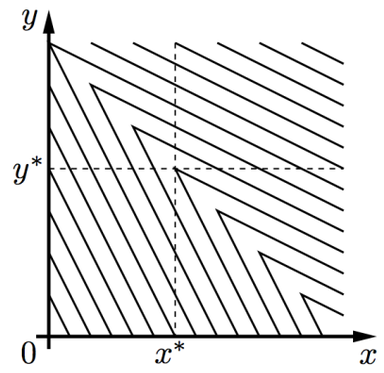
$$\max_i f_i \quad \sim \quad \epsilon \log \sum_i e^{\frac{f_i}{\epsilon}}$$



- As $\epsilon \longrightarrow 0$ becomes equal to max.
- Is an $O(\epsilon)$ approximation (and upper bound) to the original objective.
- Updates are similar, only with marginals replacing max-marginals. Converges globally.

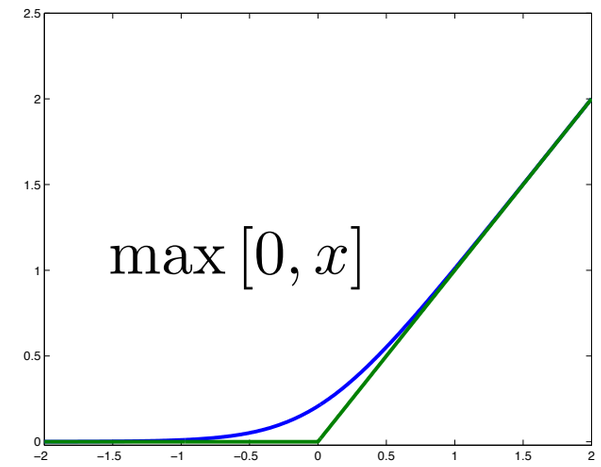
Coordinate Descent - Caveat

[Werner & Shekhovtsov, 2007]



- The objective is non-smooth.
- Coordinate descent may get stuck in "corners"
- No coordinate update improves objective, but we are not at an optimum *[Kolmogorov 04]*.
- Simplest solution: change max to soft-max:

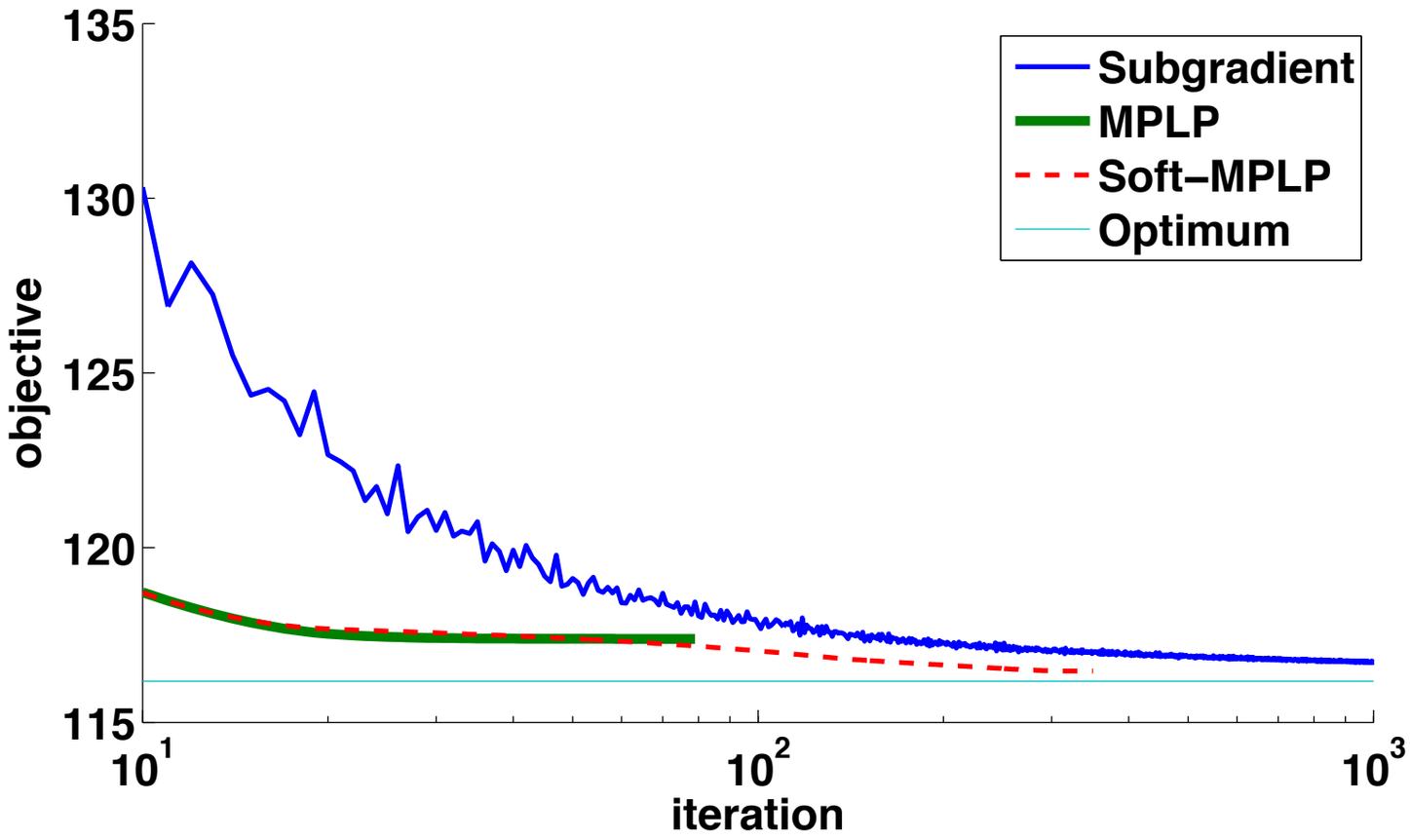
$$\max_i f_i \quad \sim \quad \epsilon \log \sum_i e^{\frac{f_i}{\epsilon}}$$



- As $\epsilon \longrightarrow 0$ becomes equal to max.
- Is an $O(\epsilon)$ approximation (and upper bound) to the original objective.
- Updates are similar, only with marginals replacing max-marginals. Converges globally.
- See *[Hazan & Shashua 07, Johnson et al 07, Werner 09, Savchynskyy et al. 11. Jojic et al. 10, Burshtein 09]*

Smoothing Example

- Model where MPLP gets stuck.



Smoothing as Primal Regularization

Smoothing as Primal Regularization

- Soft-max is obtained as the dual of:

Smoothing as Primal Regularization

- Soft-max is obtained as the dual of:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + \epsilon \sum_i H_i(\mu_i) + \epsilon \sum_f H_f(\mu_f)$$

Smoothing as Primal Regularization

- Soft-max is obtained as the dual of:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + \epsilon \sum_i H_i(\mu_i) + \epsilon \sum_f H_f(\mu_f)$$

- Entropy regularized version of our original LP.

Smoothing as Primal Regularization

- Soft-max is obtained as the dual of:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + \epsilon \sum_i H_i(\mu_i) + \epsilon \sum_f H_f(\mu_f)$$

- Entropy regularized version of our original LP.
- The dual is smooth and has Lipschitz constant: $L = O(\frac{1}{\epsilon})$.

Smoothing as Primal Regularization

- Soft-max is obtained as the dual of:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta} \cdot \boldsymbol{\mu} + \epsilon \sum_i H_i(\mu_i) + \epsilon \sum_f H_f(\mu_f)$$

- Entropy regularized version of our original LP.
- The dual is smooth and has Lipschitz constant: $L = O(\frac{1}{\epsilon})$.
- Can be used to obtain faster subgradient algorithms. See *[Savchynskyy et al. 11. Jojic et al. 10]* and next slide.

Fast Gradient Descent

Fast Gradient Descent

- Smoothing can be used to obtain $O(1/\epsilon)$ time algorithm for solving the dual to accuracy ϵ (compare to $O(1/\epsilon^2)$ without smoothing).

Fast Gradient Descent

- Smoothing can be used to obtain $O(1/\epsilon)$ time algorithm for solving the dual to accuracy ϵ (compare to $O(1/\epsilon^2)$ without smoothing).
- Use the following facts:

Fast Gradient Descent

- Smoothing can be used to obtain $O(1/\epsilon)$ time algorithm for solving the dual to accuracy ϵ (compare to $O(1/\epsilon^2)$ without smoothing).
- Use the following facts:
  - Smoothed version is $O(\epsilon)$ of true objective.

Fast Gradient Descent

- Smoothing can be used to obtain $O(1/\epsilon)$ time algorithm for solving the dual to accuracy ϵ (compare to $O(1/\epsilon^2)$ without smoothing).
- Use the following facts:
  - Smoothed version is $O(\epsilon)$ of true objective.
  - Smoothed version has Lipschitz constant of $L = O(1/\epsilon)$.

Fast Gradient Descent

- Smoothing can be used to obtain $O(1/\epsilon)$ time algorithm for solving the dual to accuracy ϵ (compare to $O(1/\epsilon^2)$ without smoothing).
- Use the following facts:
  - Smoothed version is $O(\epsilon)$ of true objective.
  - Smoothed version has Lipschitz constant of $L = O(1/\epsilon)$.
  - For problems with Lipschitz constant L, Nestertov's accelerated method runs in $O(\sqrt{L/\epsilon}) = O(1/\epsilon)$.

Fast Gradient Descent

- Smoothing can be used to obtain $O(1/\epsilon)$ time algorithm for solving the dual to accuracy ϵ (compare to $O(1/\epsilon^2)$ without smoothing).
- Use the following facts:
  - Smoothed version is $O(\epsilon)$ of true objective.
  - Smoothed version has Lipschitz constant of $L = O(1/\epsilon)$.
  - For problems with Lipschitz constant L, Nestertov's accelerated method runs in $O(\sqrt{L/\epsilon}) = O(1/\epsilon)$.

Original Dual

Non smooth

Fast Gradient Descent

- Smoothing can be used to obtain $O(1/\epsilon)$ time algorithm for solving the dual to accuracy ϵ (compare to $O(1/\epsilon^2)$ without smoothing).
- Use the following facts:
  - Smoothed version is $O(\epsilon)$ of true objective.
  - Smoothed version has Lipschitz constant of $L = O(1/\epsilon)$.
  - For problems with Lipschitz constant L, Nestertov's accelerated method runs in $O(\sqrt{L/\epsilon}) = O(1/\epsilon)$.

Original Dual	Smooth Dual
Non smooth	$L = O(\frac{1}{\epsilon})$

Fast Gradient Descent

- Smoothing can be used to obtain $O(1/\epsilon)$ time algorithm for solving the dual to accuracy ϵ (compare to $O(1/\epsilon^2)$ without smoothing).
- Use the following facts:
  - Smoothed version is $O(\epsilon)$ of true objective.
  - Smoothed version has Lipschitz constant of $L = O(1/\epsilon)$.
  - For problems with Lipschitz constant L, Nestertov's accelerated method runs in $O(\sqrt{L/\epsilon}) = O(1/\epsilon)$.

Original Dual	Smooth Dual	Nesterov's Gradient Method
Non smooth	$L = O(\frac{1}{\epsilon})$	$O(\sqrt{\frac{L}{\epsilon}}) = O(\frac{1}{\epsilon})$

Tighter Relaxations

Tighter Relaxations

- We considered a rather simple relaxation so far.

Tighter Relaxations

- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?

Tighter Relaxations

- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).

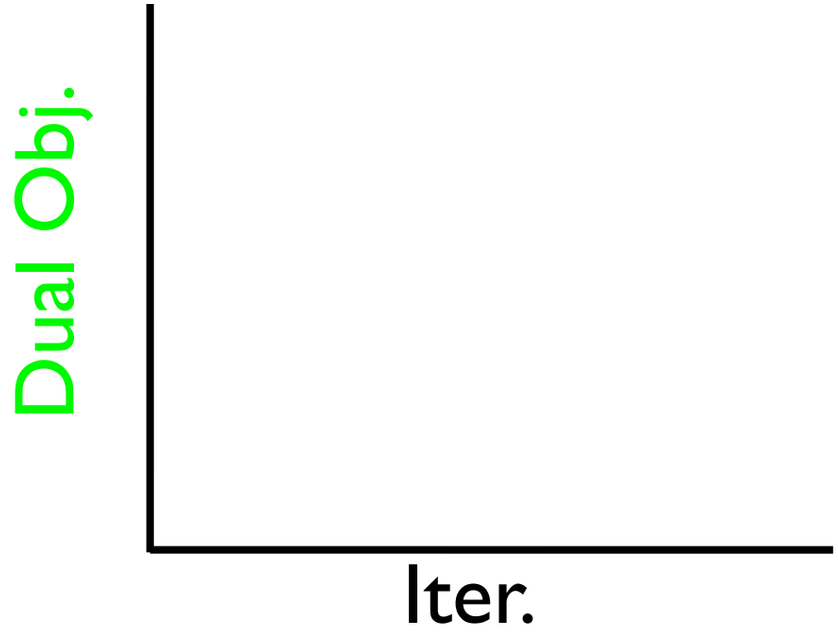
Tighter Relaxations

- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).

$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

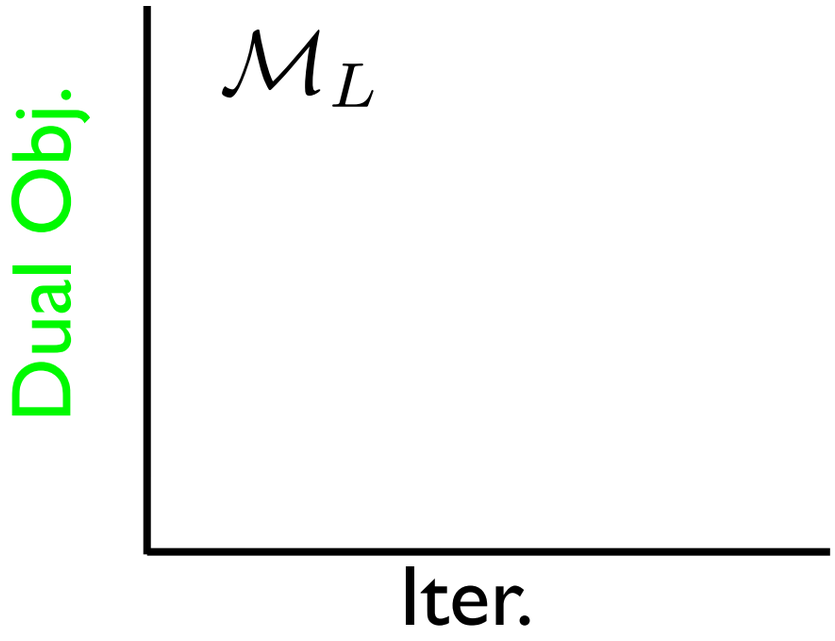
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).

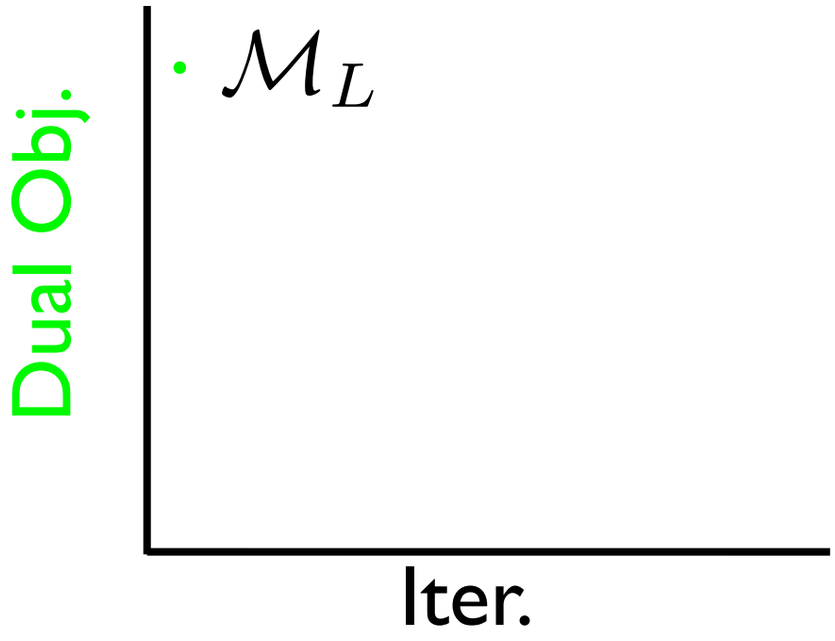


$\mathcal{M}_L$

$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

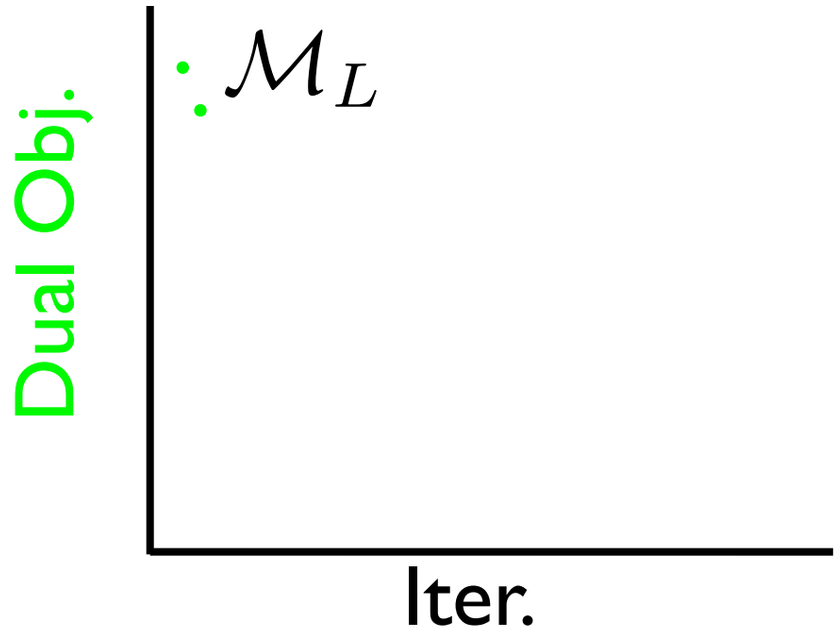
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

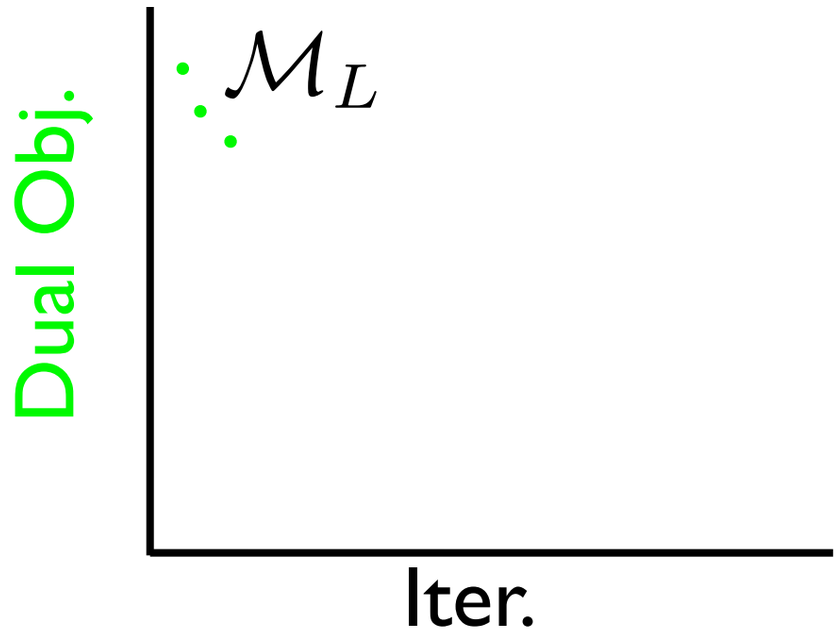
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

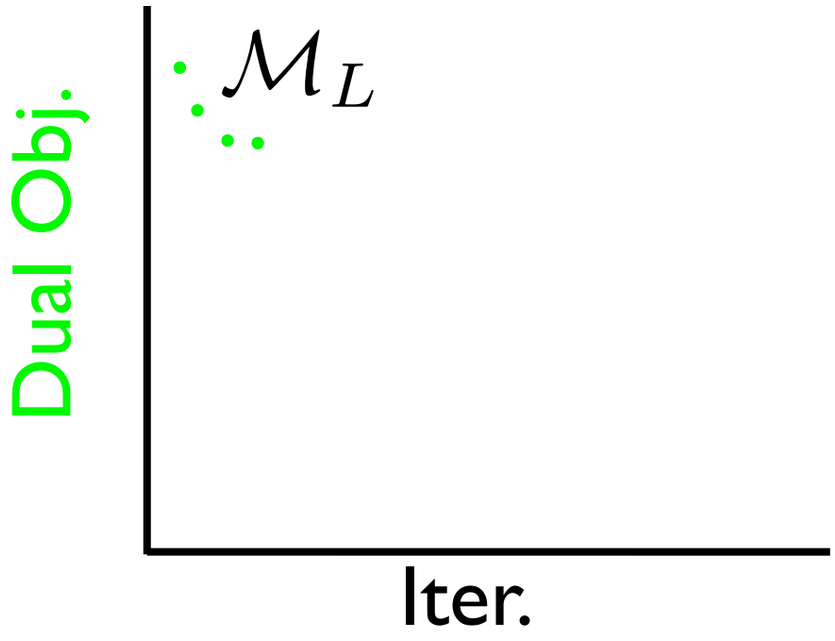
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

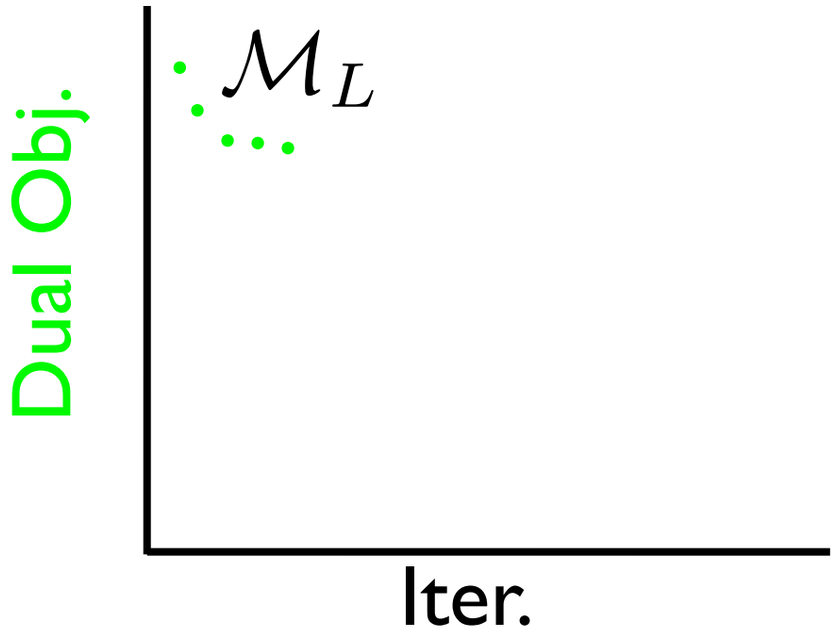
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

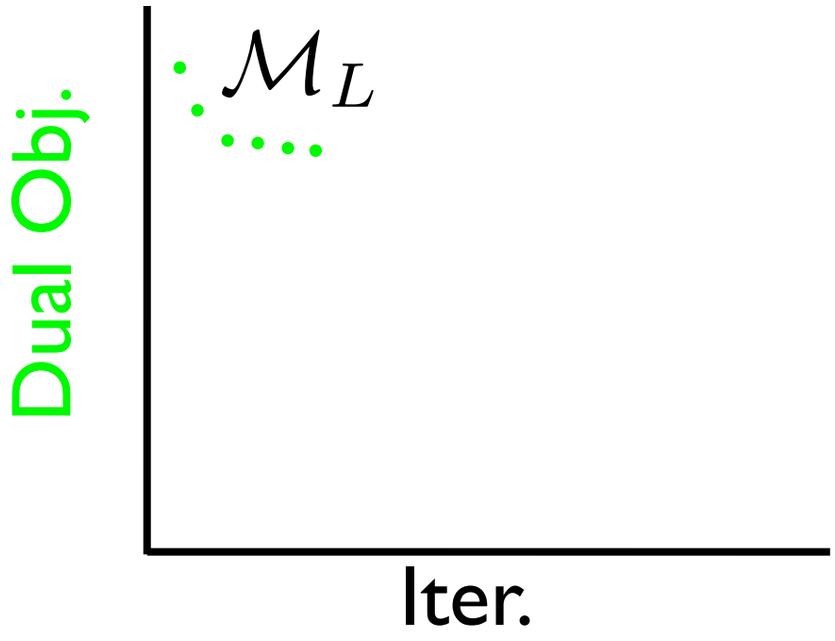
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

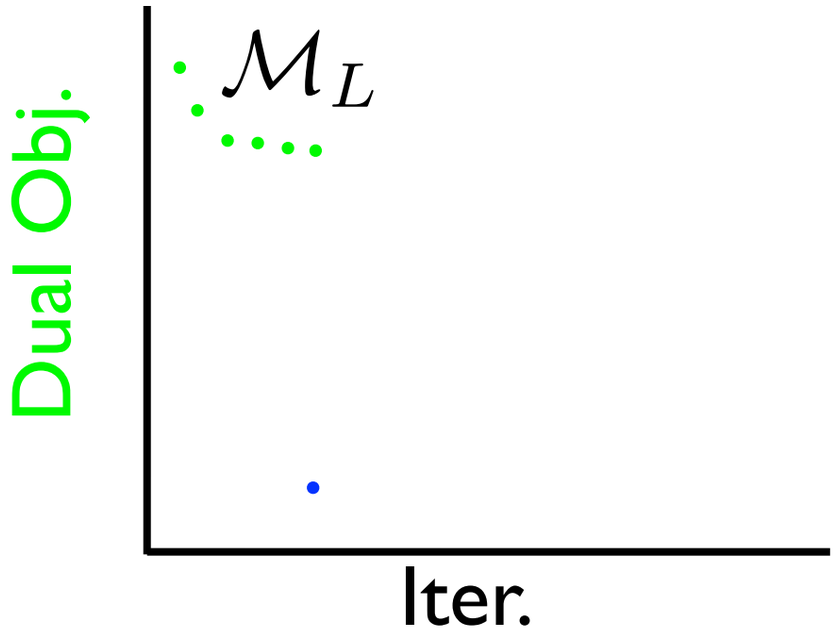
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

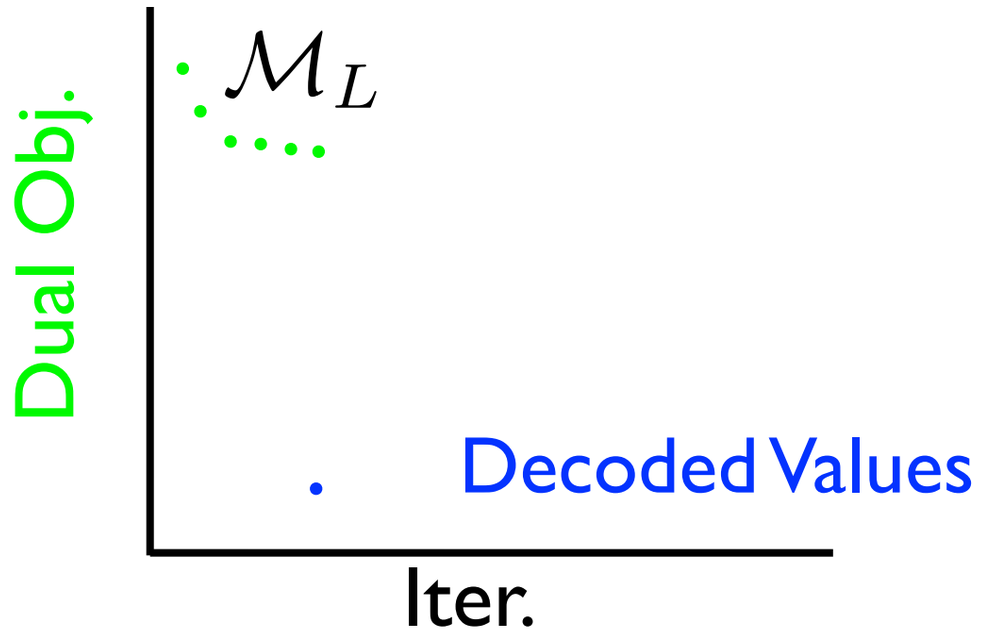
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

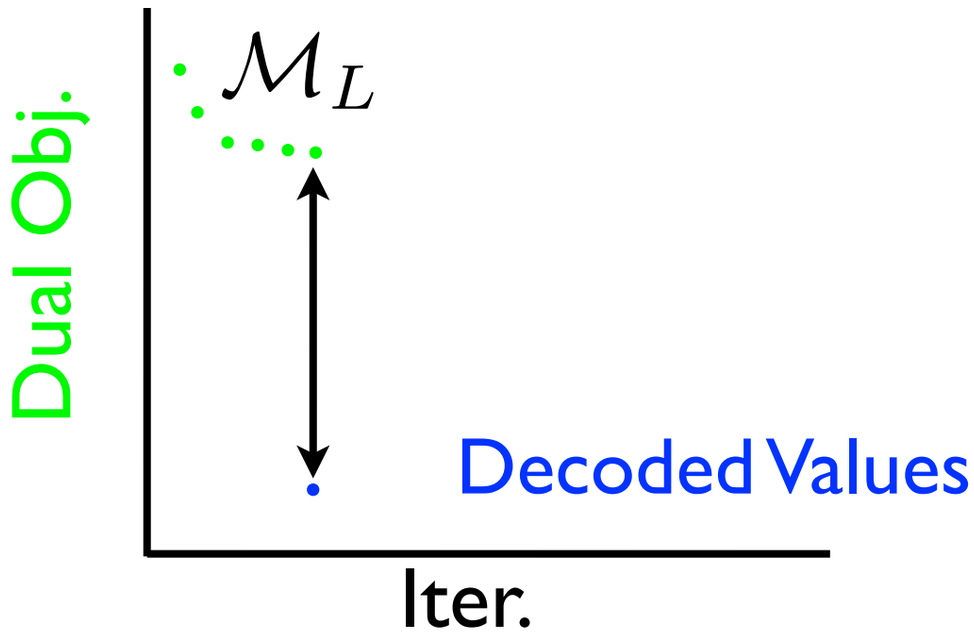
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

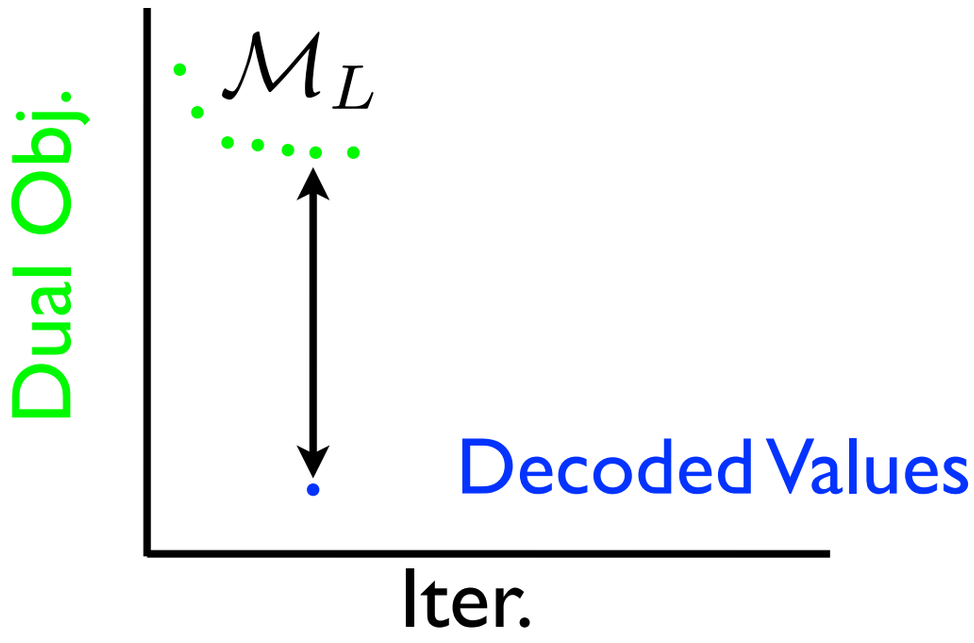
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

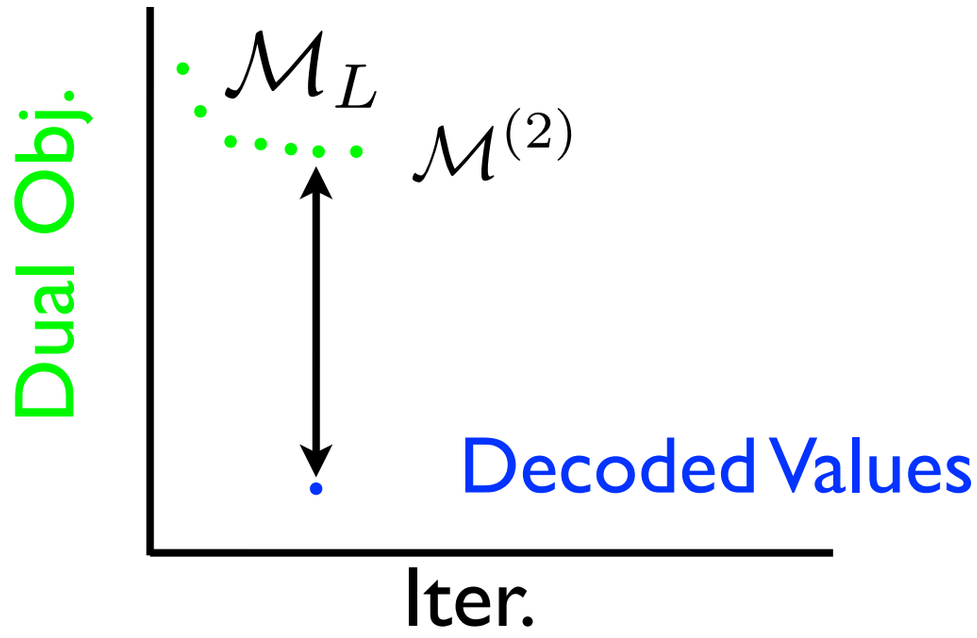
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

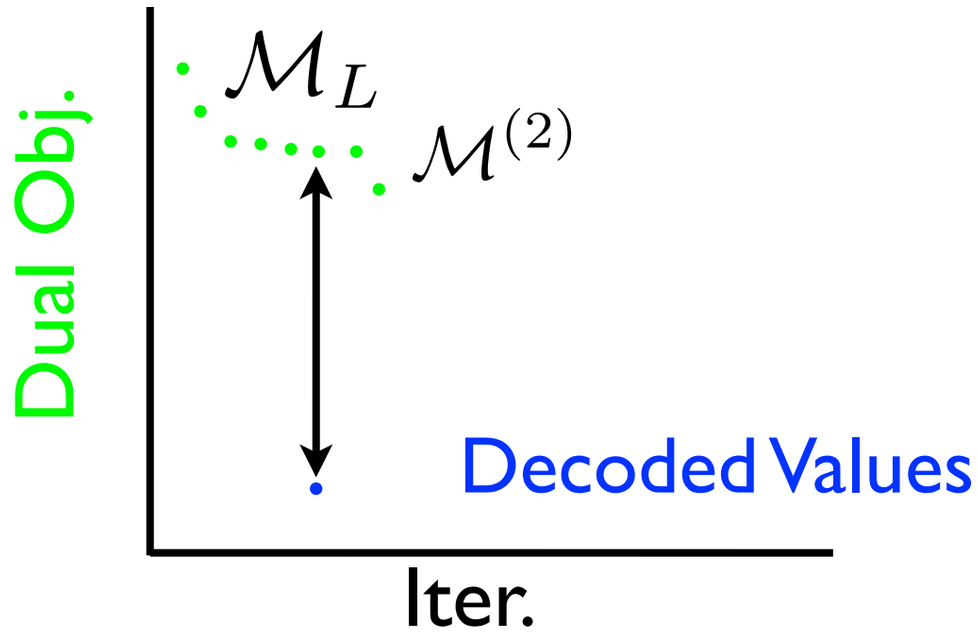
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

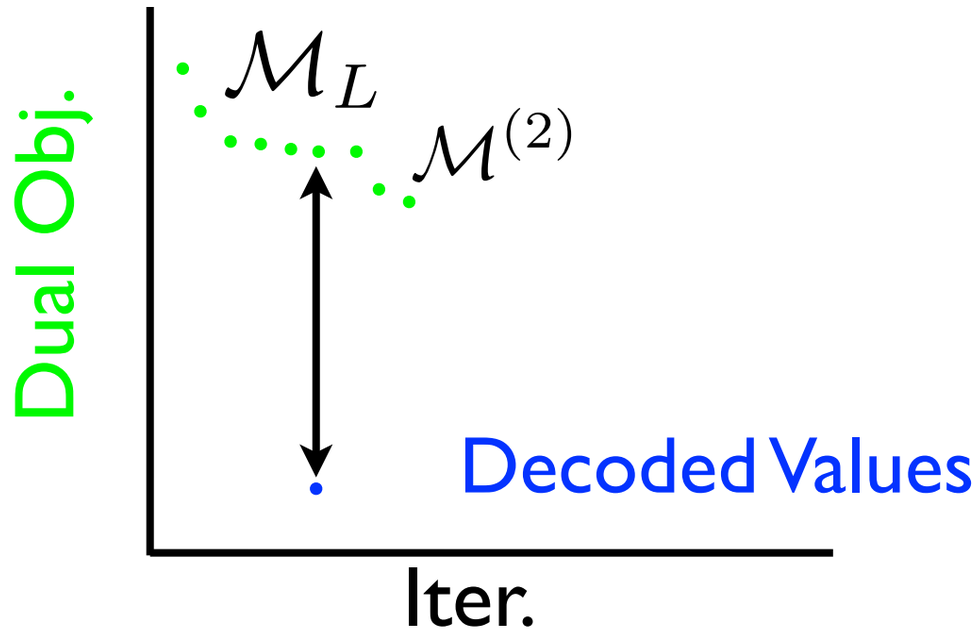
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

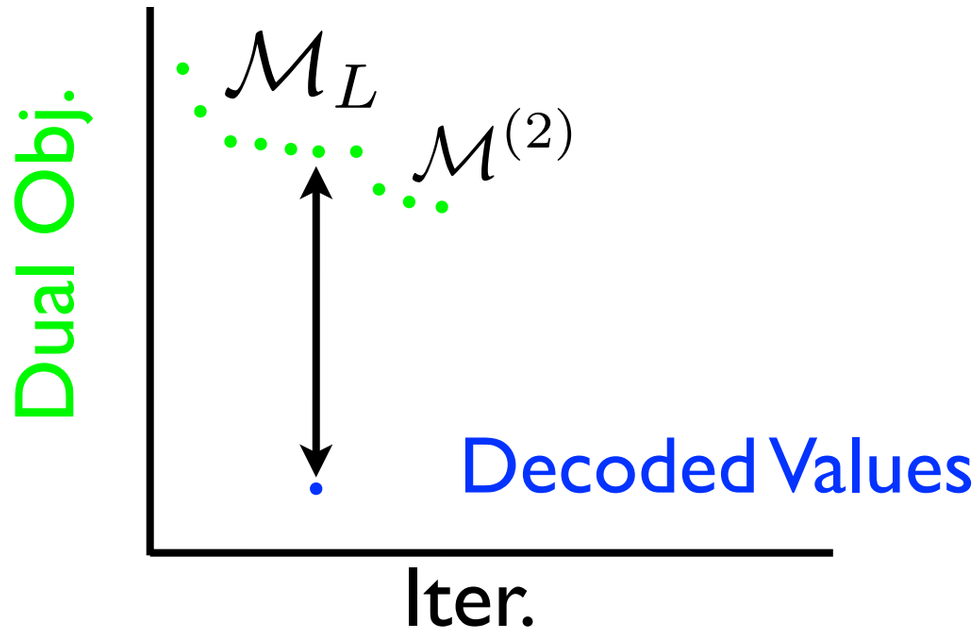
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

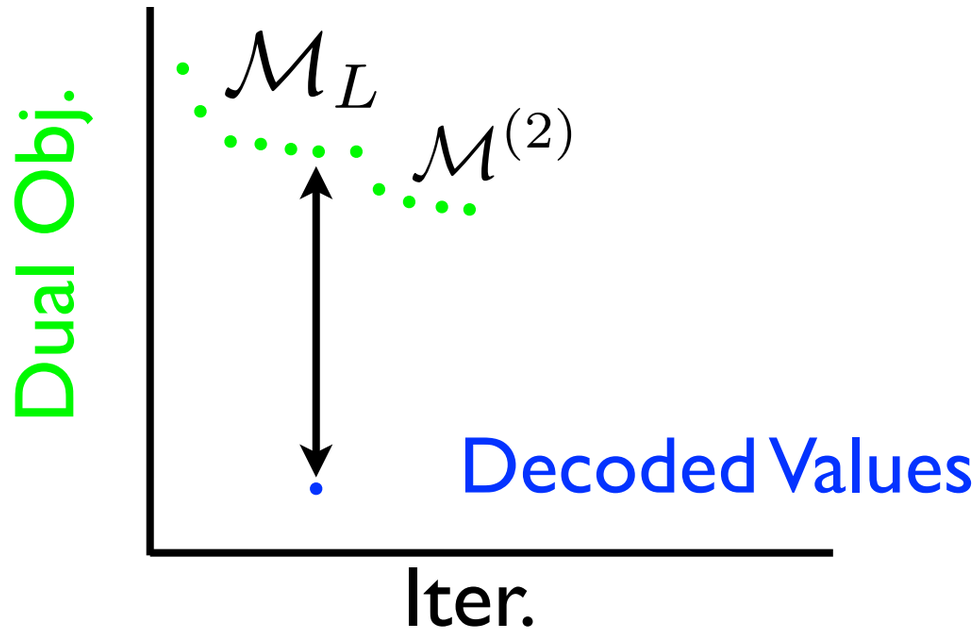
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

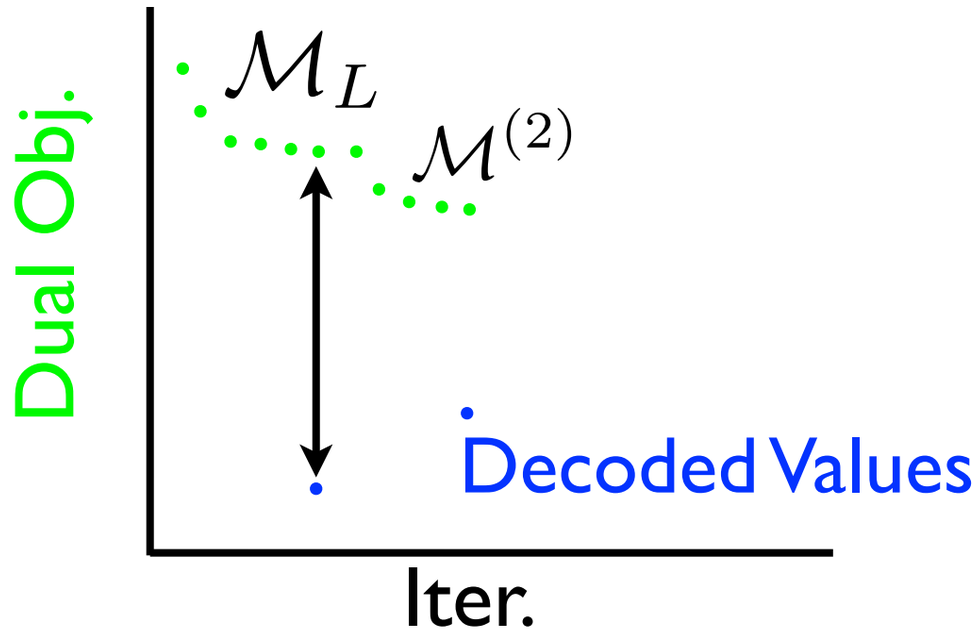
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

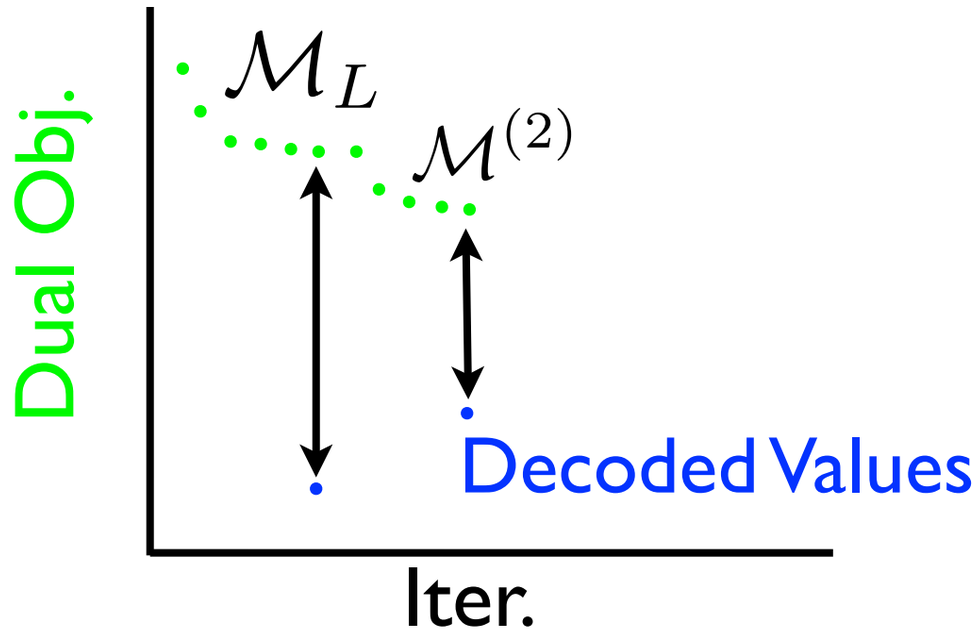
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

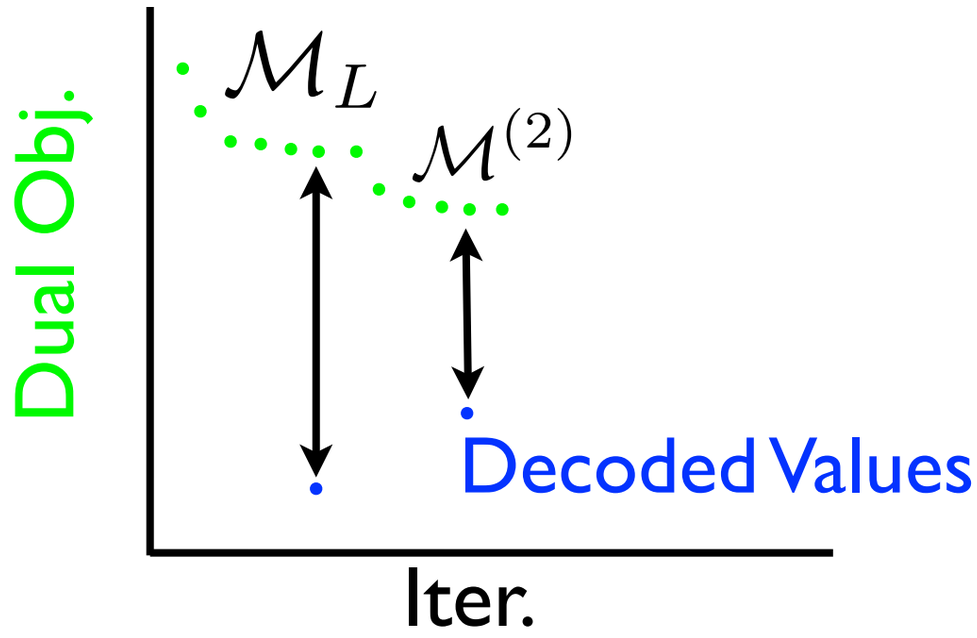
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

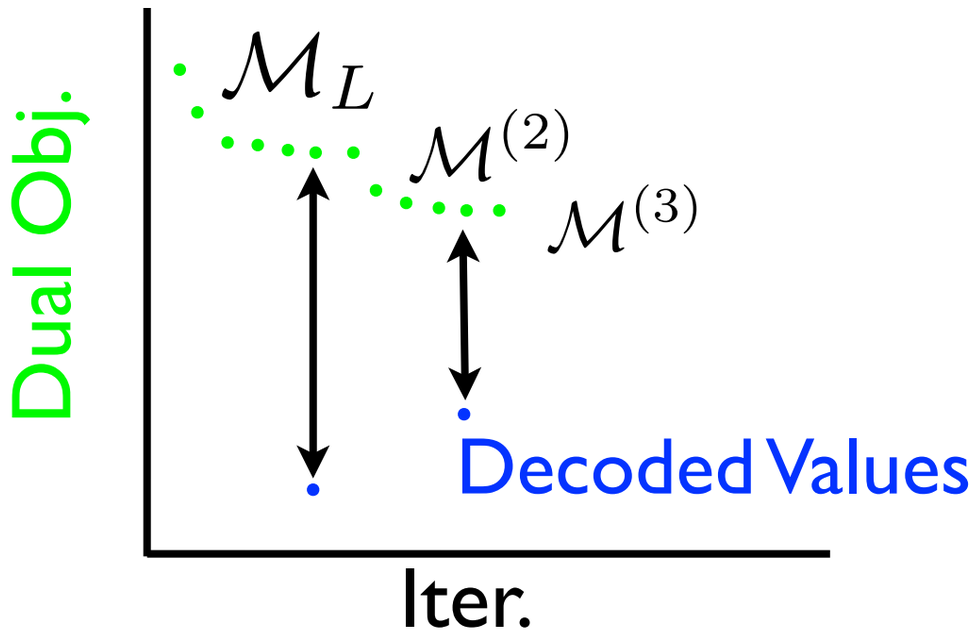
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

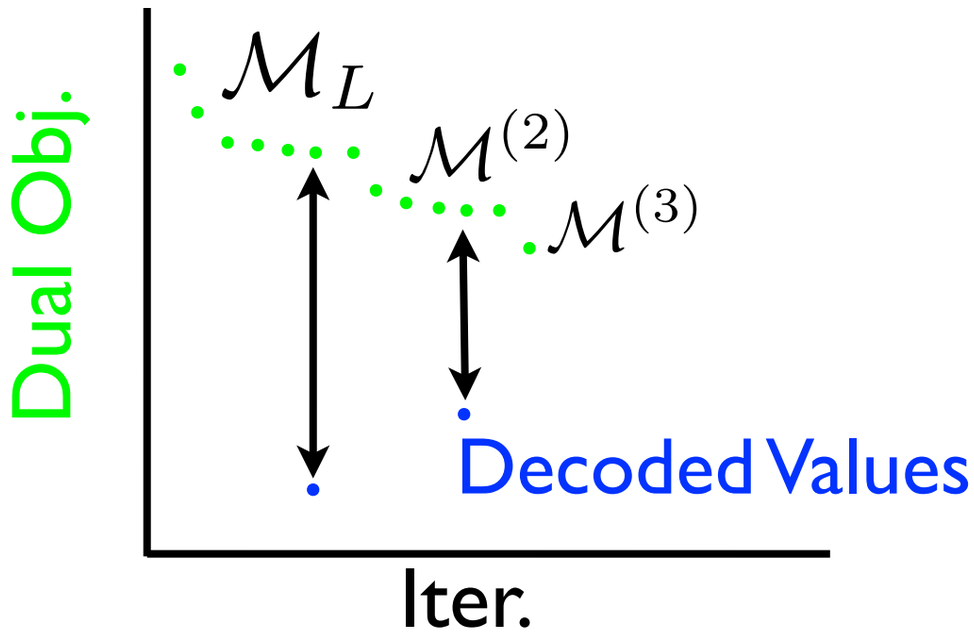
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

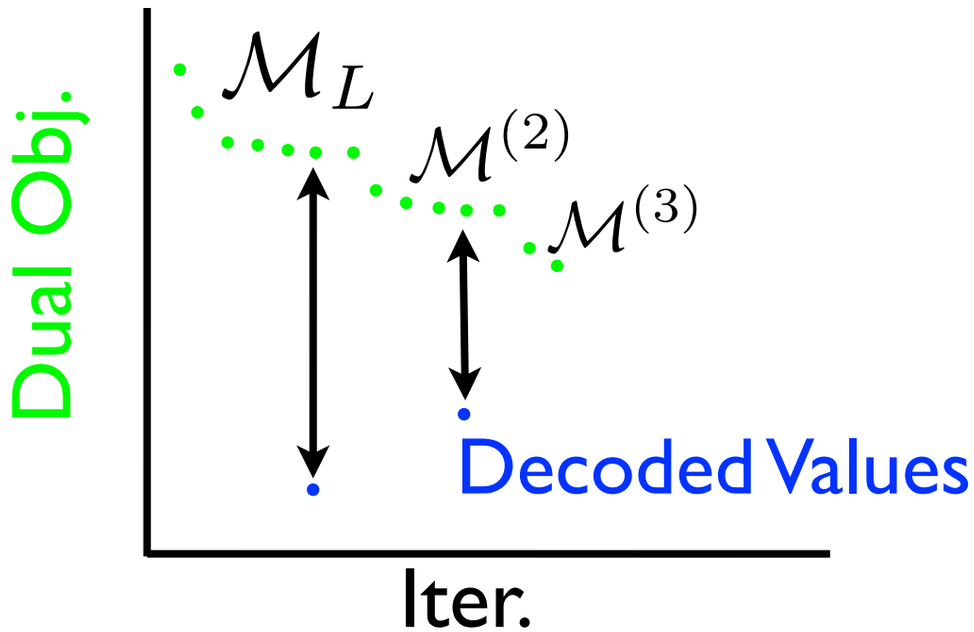
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

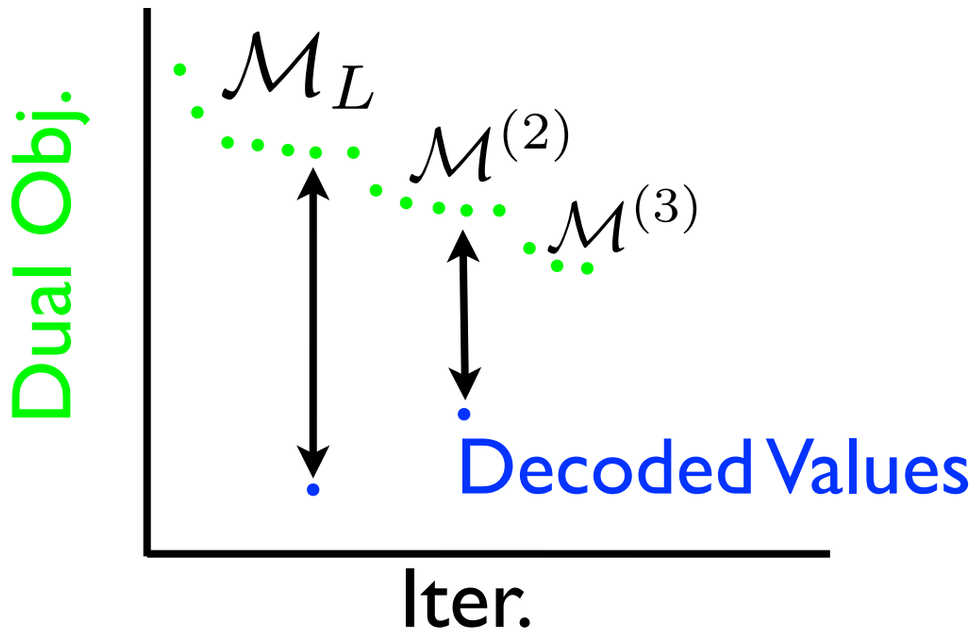
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

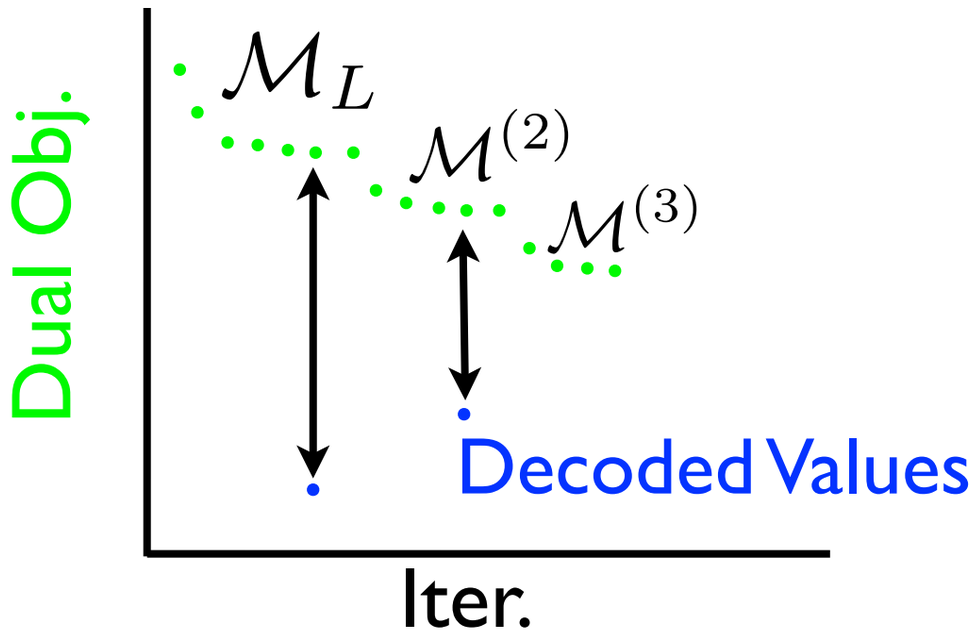
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

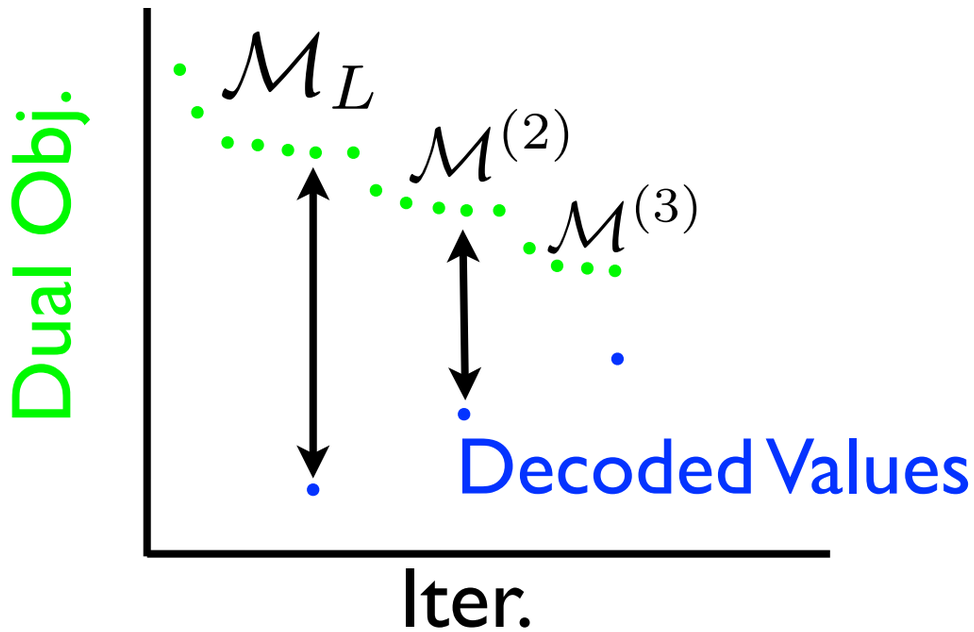
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

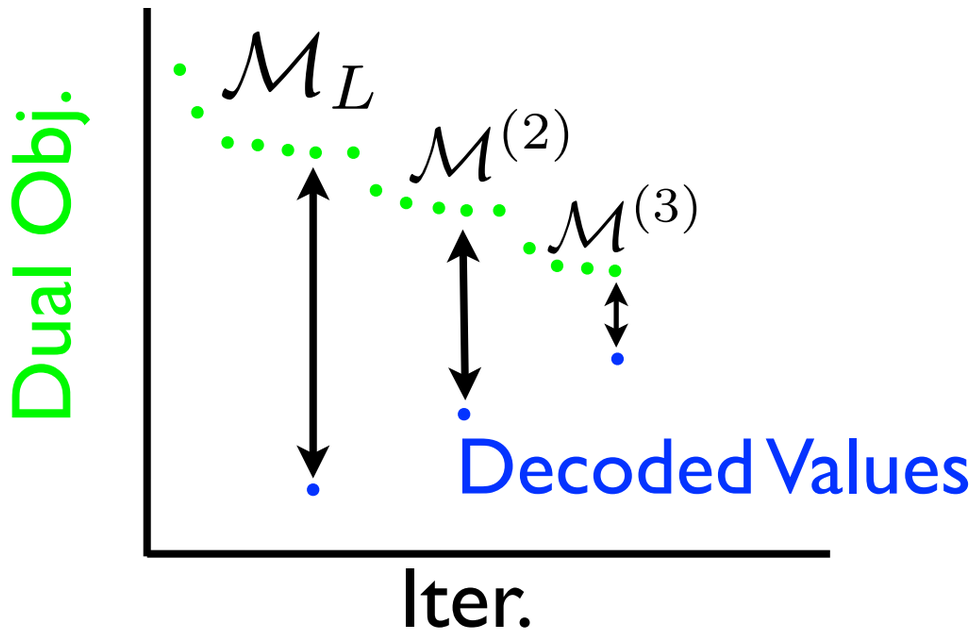
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

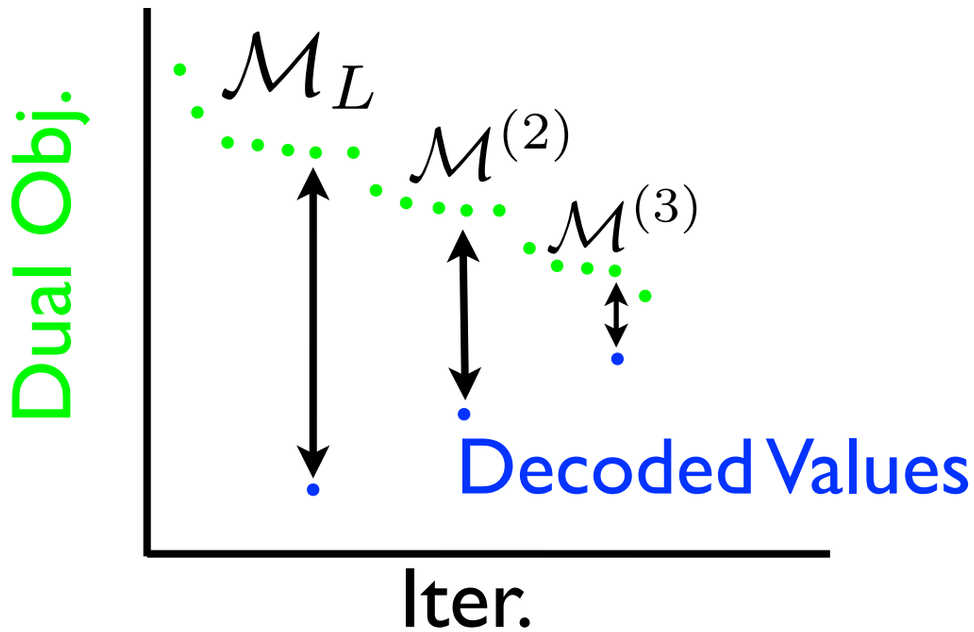
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

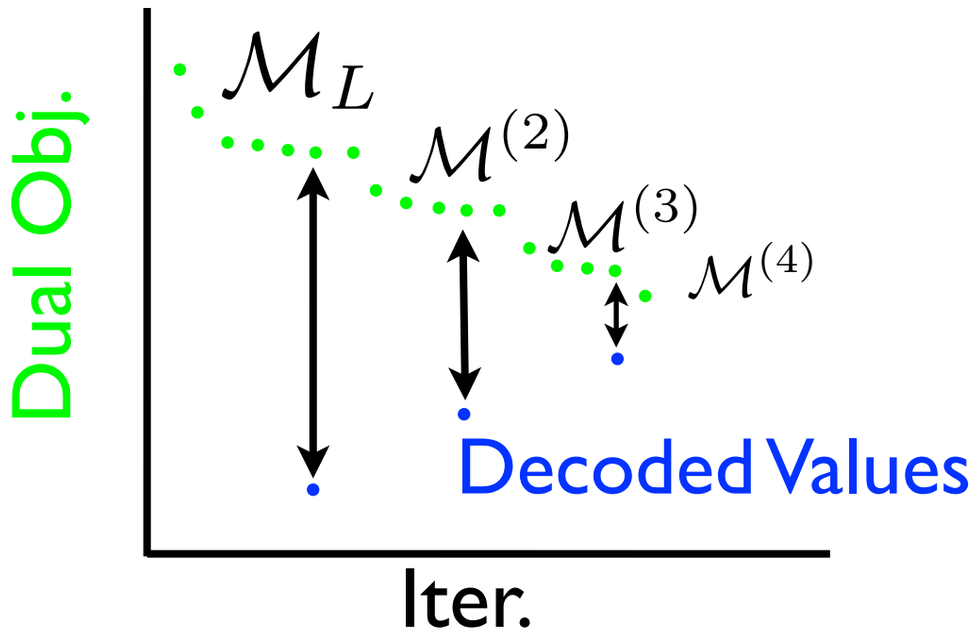
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

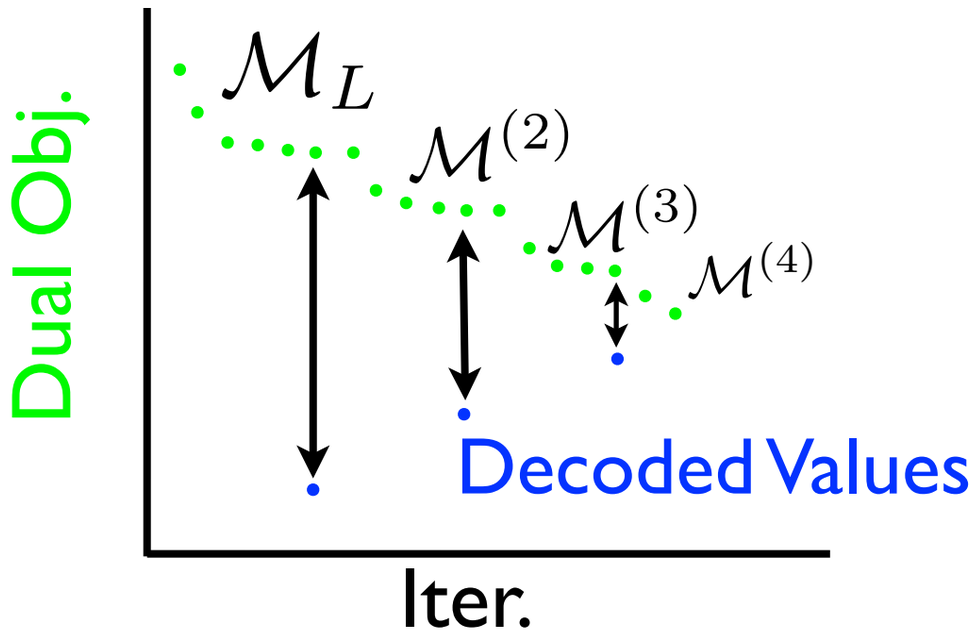
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

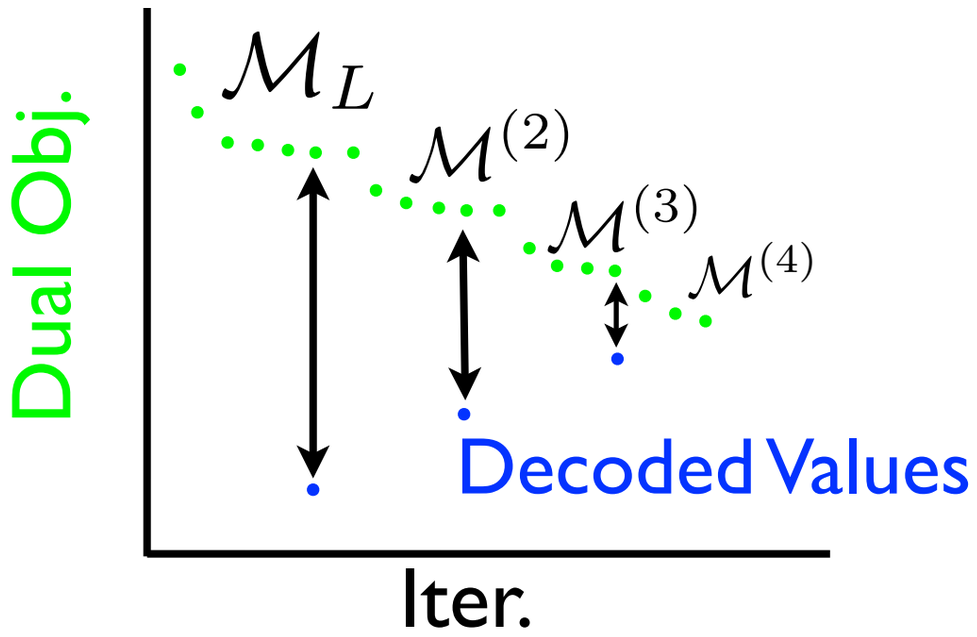
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

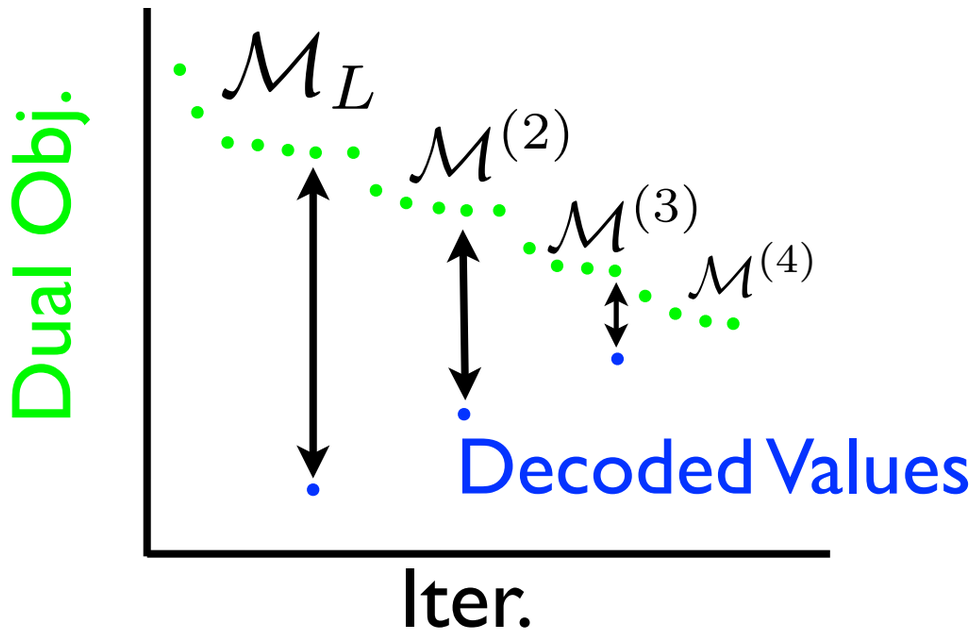
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

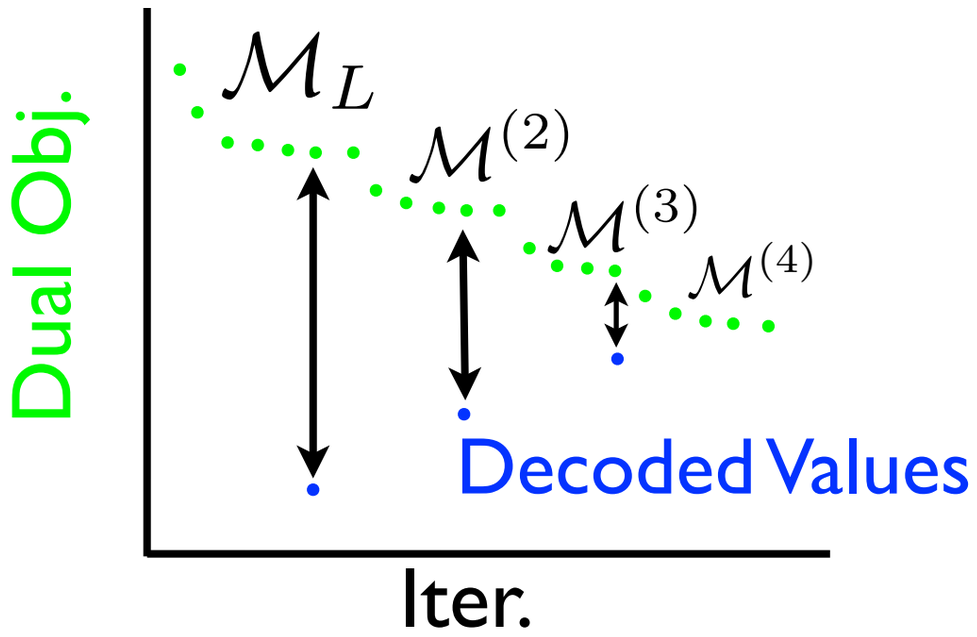
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

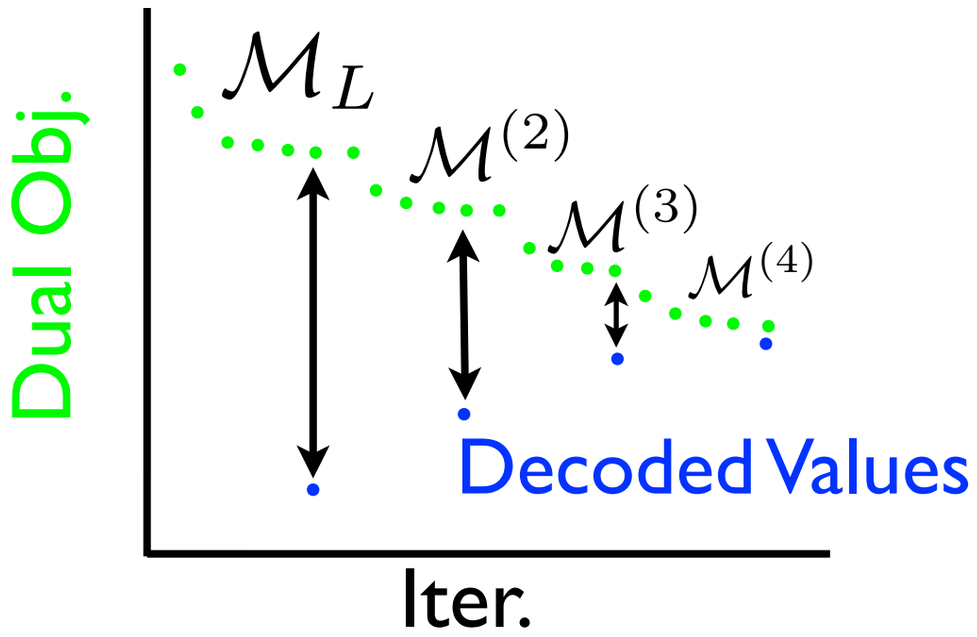
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

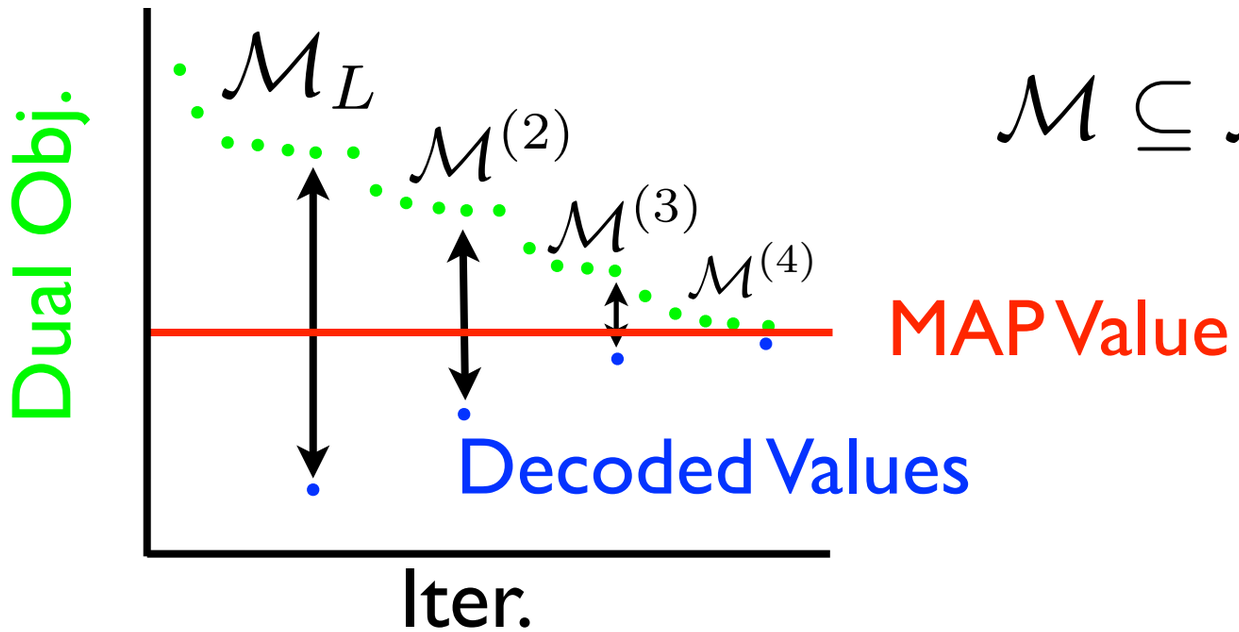
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

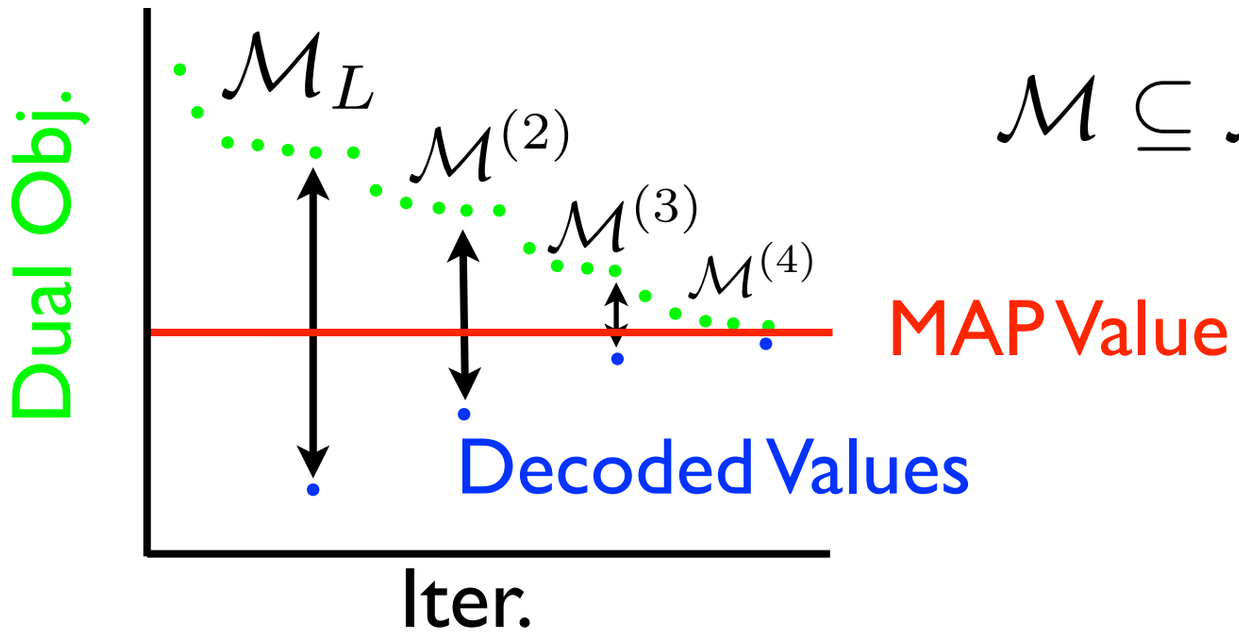
- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

Tighter Relaxations

- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).

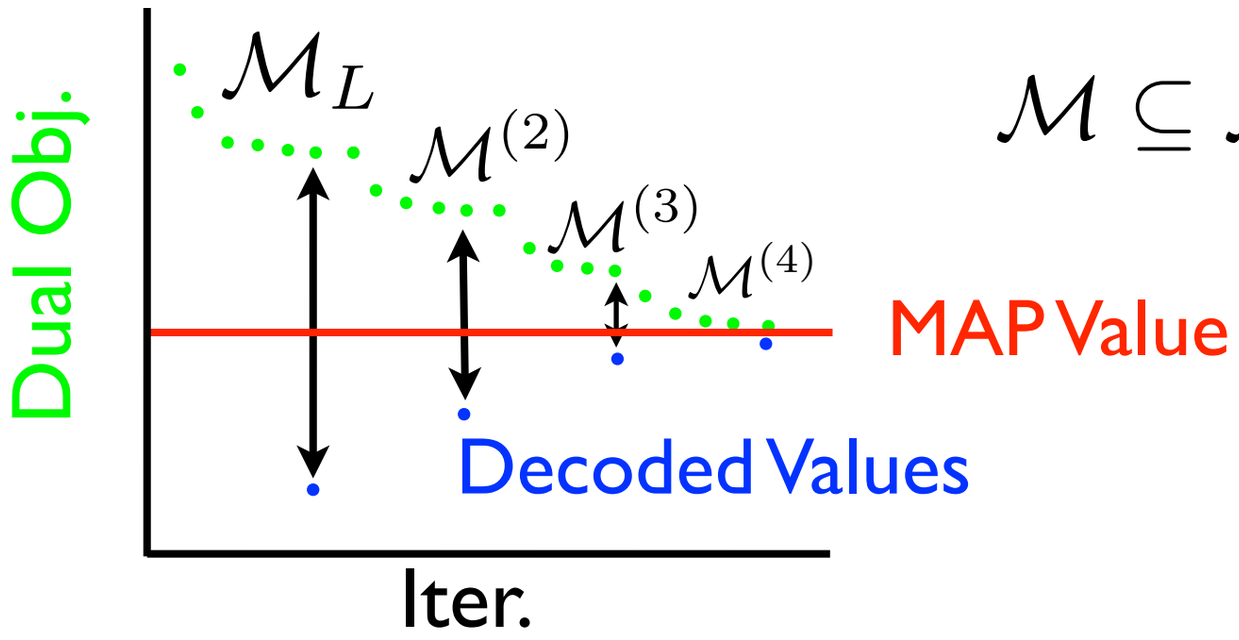


$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

- Simple coordinate descent message passing updates.

Tighter Relaxations

- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).

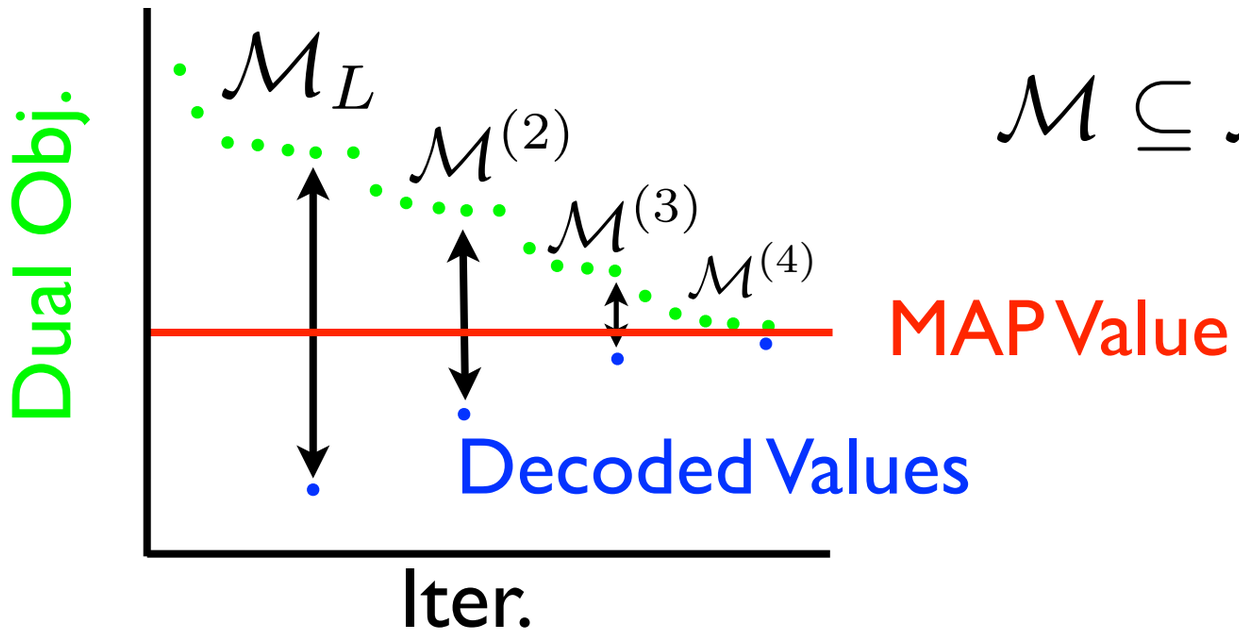


$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

- Simple coordinate descent message passing updates.
- Can solve difficult protein design problems using consistency on pairs of variables *[Sontag et al. 08]*

Tighter Relaxations

- We considered a rather simple relaxation so far.
- What if we have a problem where this relaxation does not result in an integral solution?
- Tighten the relaxation (see first part of the tutorial).



$$\mathcal{M} \subseteq \mathcal{M}^{(k)} \subset \ldots \subset \mathcal{M}^{(2)} \subset \mathcal{M}_L$$

- Simple coordinate descent message passing updates.
- Can solve difficult protein design problems using consistency on pairs of variables *[Sontag et al. 08]*
- *See also [Komodakis 08, Werner 08].*

Outline

- Solving LP relaxations:
  - Subgradient algorithms
  - Coordinate descent algorithms
- Parameter learning with LP relaxations:
  - Structured Prediction
  - Structured LP perceptron.
  - Coordinate descent algorithms.
  - Cutting plane algorithms.
  - Pseudo-max algorithms.

Structured Prediction

- We often want to predict a complex (structured) label from some input.

Structured Prediction

- We often want to predict a complex (structured) label from some input.

Part of speech tagging

x : John hit the ball

y : N V D N

Structured Prediction

- We often want to predict a complex (structured) label from some input.

Part of speech tagging

x : John hit the ball

y : N V D N

Multi-label documents

x :

y : Sports ✓ Politics ✗ Finance ✓

Structured Prediction

- We often want to predict a complex (structured) label from some input.

Part of speech tagging

x : John hit the ball

y : N V D N

Multi-label documents

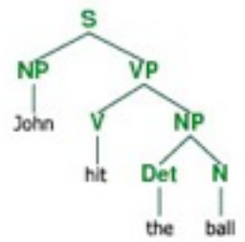
x :

y : Sports ✓ Politics x Finance ✓

Parsing

x : John hit the ball

y :

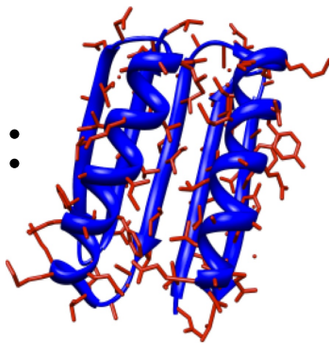


Structured Prediction

Sequence to structure

x : ACGCGGTACT

y :



Structured Prediction

Sequence to structure

x : ACGCGGTACT

y :

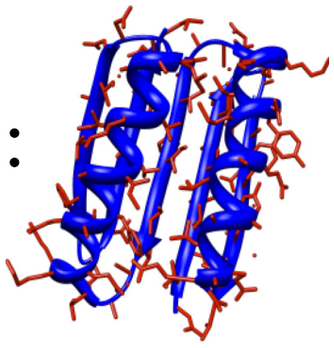


Image segmentation

Structured Prediction

Sequence to structure

x : ACGCGGTACT

y :

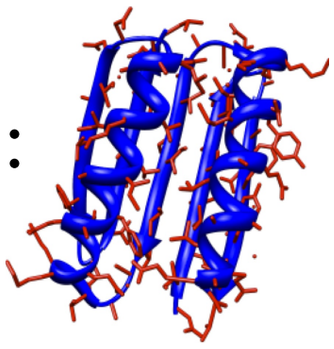


Image segmentation

Stereo Matching

x :

$\mathbf{y}$:

Structured Prediction

- Prediction is often done by maximizing an MRF

Structured Prediction

- Prediction is often done by maximizing an MRF

x

Structured Prediction

- Prediction is often done by maximizing an MRF

x

Structured Prediction

- Prediction is often done by maximizing an MRF

x

$$\longrightarrow \quad s(y;x) = \sum_f \theta_f(y_f;x)$$

Structured Prediction

- Prediction is often done by maximizing an MRF

x

$$\longrightarrow \quad s(y;x) = \sum_f \theta_f(y_f;x) \qquad \max_y s(y;x) \longrightarrow$$

Structured Prediction

- Prediction is often done by maximizing an MRF

x

$$\longrightarrow \quad s(y;x) = \sum_f \theta_f(y_f;x) \qquad \max_y s(y;x) \longrightarrow$$

y

Structured Prediction

- Prediction is often done by maximizing an MRF

x

$$\longrightarrow \quad s(y;x) = \sum_f \theta_f(y_f;x) \qquad \max_y s(y;x) \longrightarrow$$

y

- We'd like to learn these functions from data.

Structured Prediction

- Prediction is often done by maximizing an MRF

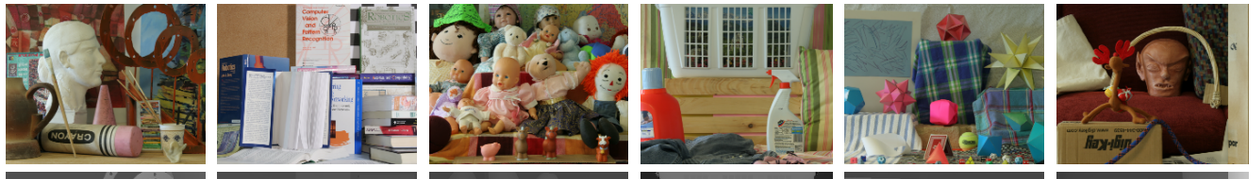
x

$$\longrightarrow \quad s(y;x) = \sum_f \theta_f(y_f;x) \qquad \max_y s(y;x) \longrightarrow$$

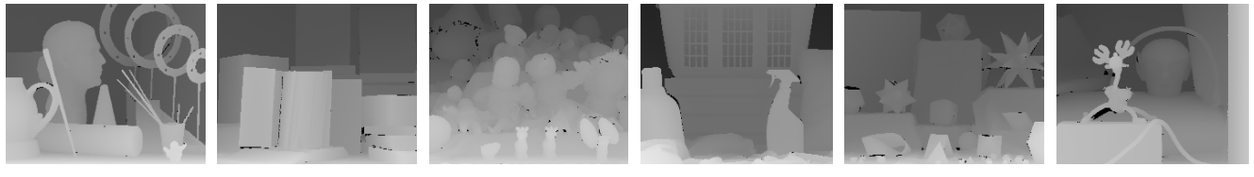
y

- We'd like to learn these functions from data.

x



y



[Scharstein & Pal 07, Middlebury dataset]

Structured Prediction

- Prediction is often done by maximizing an MRF

x

$$\longrightarrow \quad s(y;x) = \sum_f \theta_f(y_f;x) \qquad \max_y s(y;x) \longrightarrow$$

y

- We'd like to learn these functions from data.

x

y

$$s(y;x,\boldsymbol{w})$$

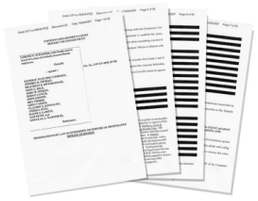
Parameters learned from data

[Scharstein & Pal 07, Middlebury dataset]

Structured Prediction

- Consider document multi-label prediction.

$\boldsymbol{x}$:



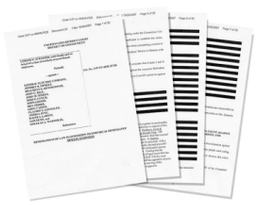
y : $y_1 = \mathrm{Is}Sports$

$y_2 = \mathrm{Is}Politics$

$y_3 = \mathrm{Is}Business$

Structured Prediction

- Consider document multi-label prediction.

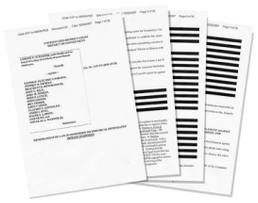
x : 

$$y: \quad \begin{aligned} y_1 &= \mathrm{Is}Sports \\ y_2 &= \mathrm{Is}Politics \\ y_3 &= \mathrm{Is}Business \end{aligned}$$

- Assume vector **x** is bag of words representation of the document.

Structured Prediction

- Consider document multi-label prediction.

$$x: \quad \text{  }$$

$$y: \quad \begin{aligned} y_1 &= \mathrm{Is}Sports \\ y_2 &= \mathrm{Is}Politics \\ y_3 &= \mathrm{Is}Business \end{aligned}$$

- Assume vector **x** is bag of words representation of the document.
- Naive approach, build binary classifier for each label:

Structured Prediction

- Consider document multi-label prediction.

$$\boldsymbol{x}: \quad \begin{array}{ll} y: & y_1 = \mathrm{Is}Sports \\ & y_2 = \mathrm{Is}Politics \\ & y_3 = \mathrm{Is}Business \end{array}$$

- Assume vector **x** is bag of words representation of the document.
- Naive approach, build binary classifier for each label:

$$y_1 = \begin{cases} 0 & \boldsymbol{w}_1 \cdot \boldsymbol{x} < 0 \\ 1 & \boldsymbol{w}_1 \cdot \boldsymbol{x} \geq 0 \end{cases}$$

Structured Prediction

- Consider document multi-label prediction.

$$\boldsymbol{x}: \quad \begin{array}{ll} y: & y_1 = \mathrm{Is}Sports \\ & y_2 = \mathrm{Is}Politics \\ & y_3 = \mathrm{Is}Business \end{array}$$

- Assume vector **x** is bag of words representation of the document.
- Naive approach, build binary classifier for each label:

$$y_1 = \begin{cases} 0 & \boldsymbol{w}_1 \cdot \boldsymbol{x} < 0 \\ 1 & \boldsymbol{w}_1 \cdot \boldsymbol{x} \geq 0 \end{cases} \quad \Longleftrightarrow$$

Structured Prediction

- Consider document multi-label prediction.

$$\boldsymbol{x}: \quad \begin{array}{ll} y: & y_1 = \mathrm{Is}Sports \\ & y_2 = \mathrm{Is}Politics \\ & y_3 = \mathrm{Is}Business \end{array}$$

- Assume vector **x** is bag of words representation of the document.
- Naive approach, build binary classifier for each label:

$$y_1 = \begin{cases} 0 & \boldsymbol{w}_1 \cdot \boldsymbol{x} < 0 \\ 1 & \boldsymbol{w}_1 \cdot \boldsymbol{x} \geq 0 \end{cases} \quad \Longleftrightarrow \quad y_1(\boldsymbol{x}) = \max_{y_1} y_1 \boldsymbol{w}_1 \cdot \boldsymbol{x}$$

Structured Prediction

- Consider document multi-label prediction.

$$\boldsymbol{x}: \quad \begin{array}{ll} y: & y_1 = \mathrm{Is}Sports \\ & y_2 = \mathrm{Is}Politics \\ & y_3 = \mathrm{Is}Business \end{array}$$

- Assume vector **x** is bag of words representation of the document.
- Naive approach, build binary classifier for each label:

$$y_1 = \begin{cases} 0 & \boldsymbol{w}_1 \cdot \boldsymbol{x} < 0 \\ 1 & \boldsymbol{w}_1 \cdot \boldsymbol{x} \geq 0 \end{cases} \quad \Longleftrightarrow \quad y_1(\boldsymbol{x}) = \max_{y_1} y_1 \boldsymbol{w}_1 \cdot \boldsymbol{x}$$

$$y(\boldsymbol{x}) = \arg \max_{y_1, y_2, y_3} \sum_{i=1}^{3} y_i \boldsymbol{w}_i \cdot \boldsymbol{x}$$

Structured Prediction

- Independent predictors:

$$y(\boldsymbol{x}) = \arg \max_{y_1, y_2, y_3} \sum_{i=1}^{3} y_i \boldsymbol{w}_i \cdot \boldsymbol{x}$$

Structured Prediction

- Independent predictors:

$$y(\boldsymbol{x}) = \arg \max_{y_1, y_2, y_3} \sum_{i=1}^{3} y_i \boldsymbol{w}_i \cdot \boldsymbol{x}$$

- Want to use structure on labels via pairwise interactions.

Structured Prediction

- Independent predictors:

$$y(\boldsymbol{x}) = \arg \max_{y_1, y_2, y_3} \sum_{i=1}^{3} y_i \boldsymbol{w}_i \cdot \boldsymbol{x}$$

- Want to use structure on labels via pairwise interactions.

$$y(\boldsymbol{x}) = \arg \max_{y} \sum_{i=1}^{3} y_i \boldsymbol{w}_i \cdot \boldsymbol{x} + w_{12} y_1 y_2 + w_{13} y_1 y_3 + w_{23} y_2 y_3$$

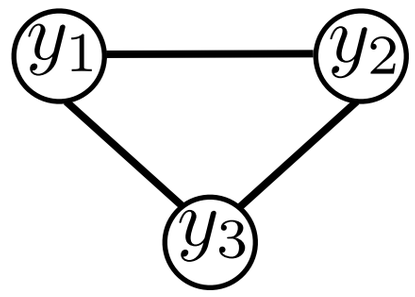
Structured Prediction

- Independent predictors:

$$y(\boldsymbol{x}) = \arg \max_{y_1, y_2, y_3} \sum_{i=1}^{3} y_i \boldsymbol{w}_i \cdot \boldsymbol{x}$$

- Want to use structure on labels via pairwise interactions.

$$y(\boldsymbol{x}) = \arg \max_{y} \sum_{i=1}^{3} y_i \boldsymbol{w}_i \cdot \boldsymbol{x} + w_{12} y_1 y_2 + w_{13} y_1 y_3 + w_{23} y_2 y_3$$



Structured Prediction

- Independent predictors:

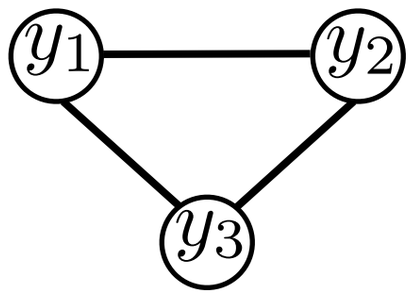
$$y(\boldsymbol{x}) = \arg \max_{y_1, y_2, y_3} \sum_{i=1}^{3} y_i \boldsymbol{w}_i \cdot \boldsymbol{x}$$

y_1 y_2 y_3

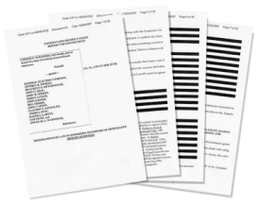
- Want to use structure on labels via pairwise interactions.

$$y(\boldsymbol{x}) = \arg \max_{y} \sum_{i=1}^{3} y_i \boldsymbol{w}_i \cdot \boldsymbol{x} + w_{12} y_1 y_2 + w_{13} y_1 y_3 + w_{23} y_2 y_3$$

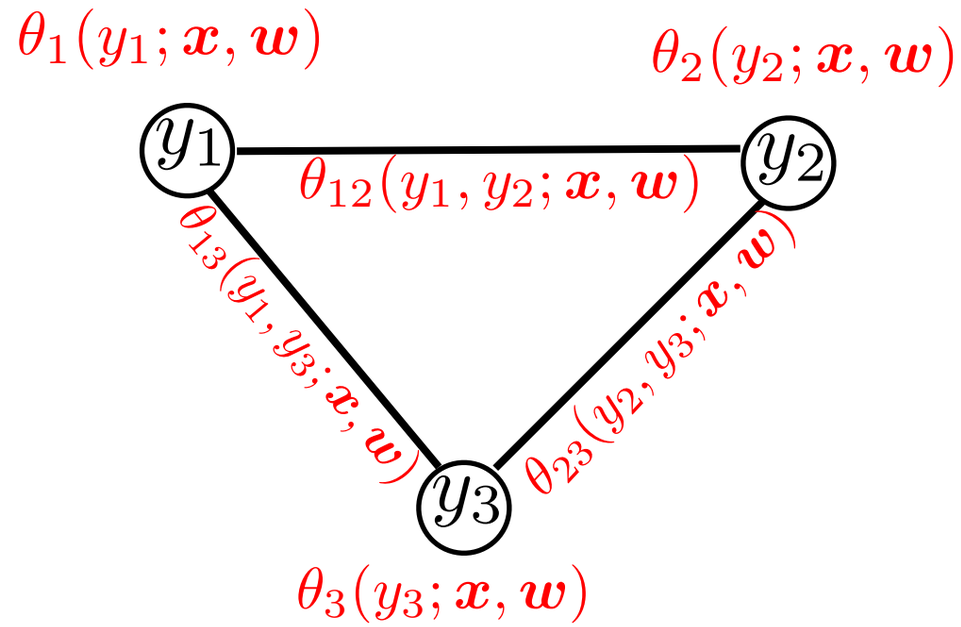
$\theta_i(y_i; \boldsymbol{x}, \boldsymbol{w})$ $\theta_{12}(y_1, y_2; \boldsymbol{x}, \boldsymbol{w})$



Structured Prediction

$\boldsymbol{x}$ 

$\boldsymbol{w}$ $\longrightarrow$



$$\longrightarrow \quad s(y; \boldsymbol{x}, \boldsymbol{w}) = \sum_f \theta_f(y_f; \boldsymbol{x}, \boldsymbol{w})$$

$$\longrightarrow \quad y(\boldsymbol{x}) = \arg\max_y s(y; \boldsymbol{x}, \boldsymbol{w})$$

The Score Function

The Score Function

- Assume score decomposes:

The Score Function

- Assume score decomposes:

$$s(y; x, \boldsymbol{w}) = \sum_f \theta_f(y_c; x, \boldsymbol{w})$$

The Score Function

- Assume score decomposes:

$$s(y; x, \boldsymbol{w}) = \sum_f \theta_f(y_c; x, \boldsymbol{w})$$

- Assume θ_f are linear in **w**.

The Score Function

- Assume score decomposes:

$$s(y; x, \boldsymbol{w}) = \sum_f \theta_f(y_c; x, \boldsymbol{w})$$

- Assume θ_f are linear in **w**.
- For each f, assume we have a function $\boldsymbol{\phi}_f(y_f, x)$

The Score Function

- Assume score decomposes:

$$s(y; x, \boldsymbol{w}) = \sum_f \theta_f(y_c; x, \boldsymbol{w})$$

- Assume θ_f are linear in **w**.
- For each f, assume we have a function $\boldsymbol{\phi}_f(y_f, x)$
- Defines features of y_f and x that might be relevant for prediction.

The Score Function

- Assume score decomposes:

$$s(y; x, \boldsymbol{w}) = \sum_f \theta_f(y_c; x, \boldsymbol{w})$$

- Assume θ_f are linear in **w**.
- For each f, assume we have a function $\boldsymbol{\phi}_f(y_f, x)$
- Defines features of y_f and x that might be relevant for prediction.
- Then: $\theta_f(y_f; x, \boldsymbol{w}) = \boldsymbol{w}_f \cdot \boldsymbol{\phi}_f(y_f, x)$

The Score Function

- Assume score decomposes:

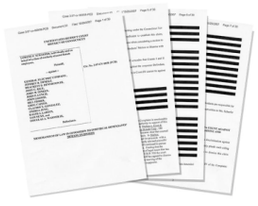
$$s(y; x, \boldsymbol{w}) = \sum_f \theta_f(y_c; x, \boldsymbol{w})$$

- Assume θ_f are linear in **w**.
- For each f, assume we have a function $\boldsymbol{\phi}_f(y_f, x)$
  - Defines features of y_f and x that might be relevant for prediction.
  - Then: $\theta_f(y_f; x, \boldsymbol{w}) = \boldsymbol{w}_f \cdot \boldsymbol{\phi}_f(y_f, x)$
  - The $\boldsymbol{w}_f$ can be different for different factors, but may also be tied.

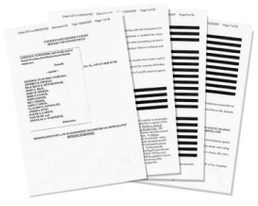
Structured Prediction

Structured Prediction

x

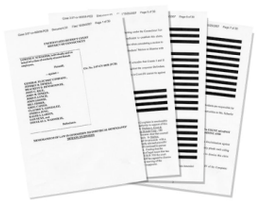


Structured Prediction

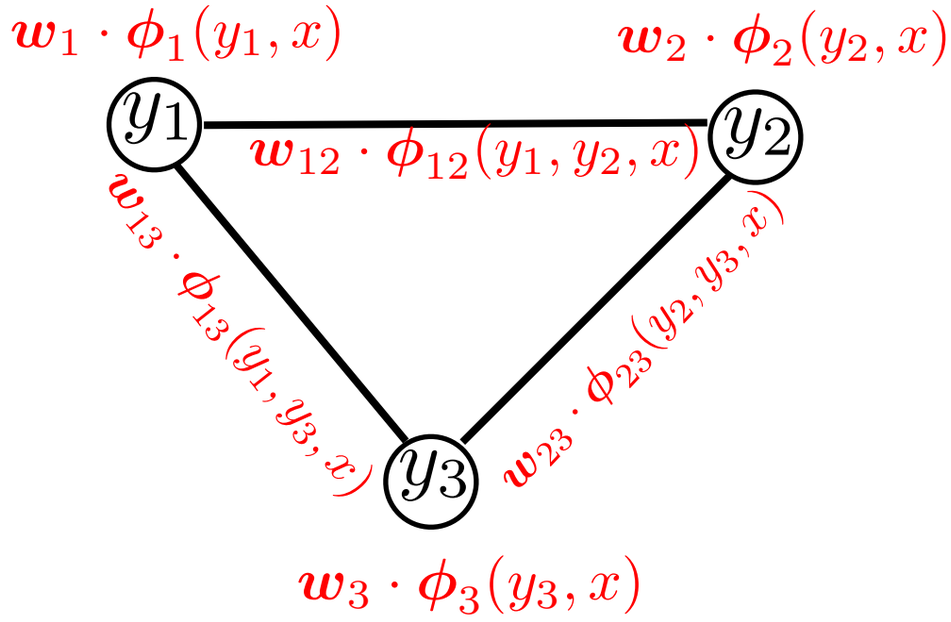
x  $\xrightarrow{w}$

Structured Prediction

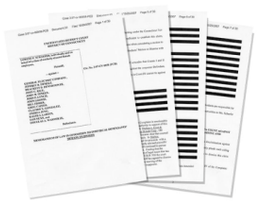
x



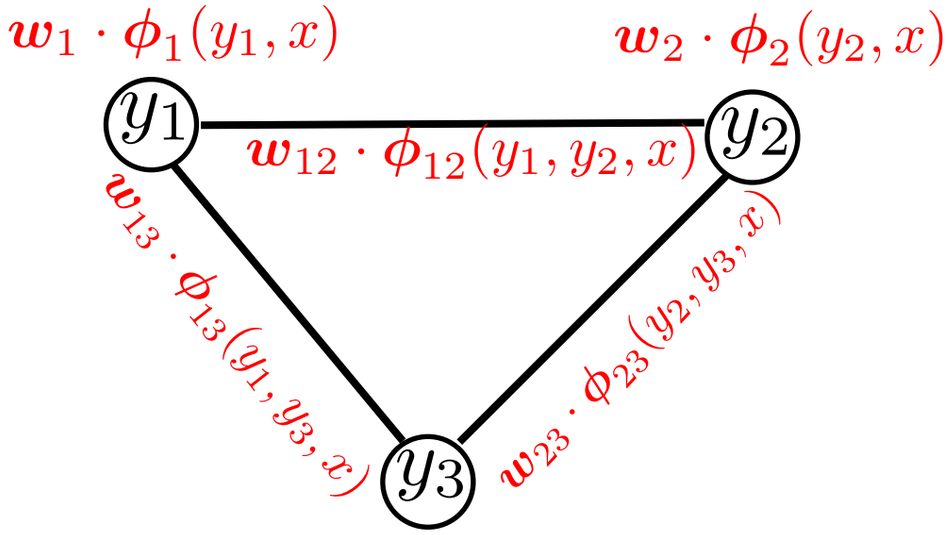
$\boldsymbol{w}$



Structured Prediction

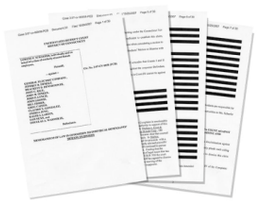
$\boldsymbol{x}$ 

$\boldsymbol{w}$ $\longrightarrow$

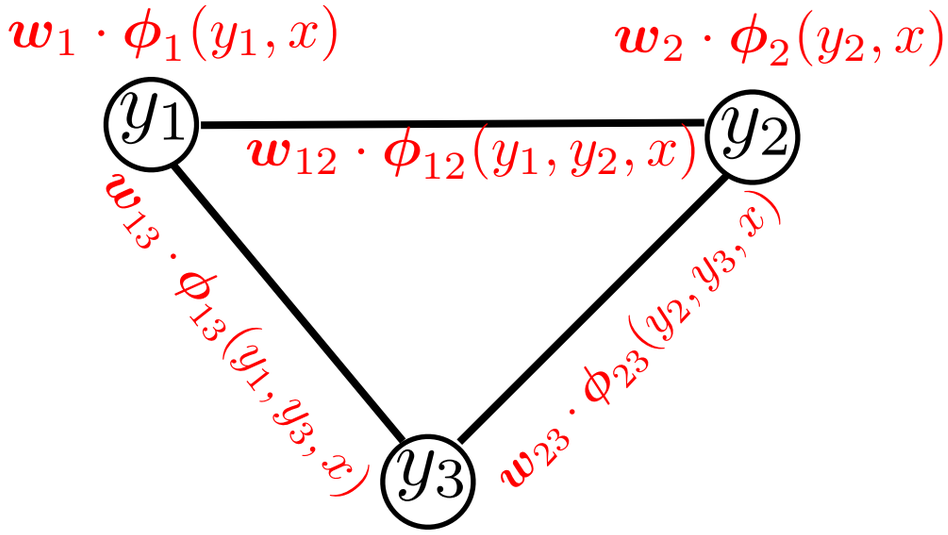


$$\longrightarrow \quad s(y; \boldsymbol{x}, \boldsymbol{w}) = \sum_f \boldsymbol{w}_f \cdot \boldsymbol{\phi}_f(y_f, \boldsymbol{x})$$

Structured Prediction

$\boldsymbol{x}$ 

$\boldsymbol{w}$ $\longrightarrow$



$$\longrightarrow \quad s(y; \boldsymbol{x}, \boldsymbol{w}) = \sum_f \boldsymbol{w}_f \cdot \boldsymbol{\phi}_f(y_f, \boldsymbol{x})$$

$$\longrightarrow \quad y(\boldsymbol{x}) = \arg\max_y s(y; \boldsymbol{x}, \boldsymbol{w})$$

Score Function - Marginal Polytope

- It will be convenient to express things in terms of points in the marginal polytope.

Score Function - Marginal Polytope

- It will be convenient to express things in terms of points in the marginal polytope.

$$\max_{y} \sum_{f} \theta_f(y_f; x, \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}} \sum_{f} \sum_{y_f} \theta_f(y_f; x, \boldsymbol{w}) \mu_f(y_f)$$

Score Function - Marginal Polytope

- It will be convenient to express things in terms of points in the marginal polytope.

$$\max_{y} \sum_{f} \theta_f(y_f; x, \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}} \sum_{f} \sum_{y_f} \theta_f(y_f; x, \boldsymbol{w}) \mu_f(y_f)$$

$$= \max_{\boldsymbol{\mu} \in \mathcal{M}} \sum_{f} \boldsymbol{w}_f \cdot \sum_{y_f} \boldsymbol{\phi}_f(y_f, x) \mu_f(y_f)$$

Score Function - Marginal Polytope

- It will be convenient to express things in terms of points in the marginal polytope.

$$\max_y \sum_f \theta_f(y_f; x, \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}} \sum_f \sum_{y_f} \theta_f(y_f; x, \boldsymbol{w}) \mu_f(y_f)$$

$$= \max_{\boldsymbol{\mu} \in \mathcal{M}} \sum_f \boldsymbol{w}_f \cdot \sum_{y_f} \boldsymbol{\phi}_f(y_f, x) \mu_f(y_f)$$

$$= \max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$

Score Function - Marginal Polytope

- It will be convenient to express things in terms of points in the marginal polytope.

$$\max_y \sum_f \theta_f(y_f; x, \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}} \sum_f \sum_{y_f} \theta_f(y_f; x, \boldsymbol{w}) \mu_f(y_f)$$

$$= \max_{\boldsymbol{\mu} \in \mathcal{M}} \sum_f \boldsymbol{w}_f \cdot \sum_{y_f} \boldsymbol{\phi}_f(y_f, x) \mu_f(y_f)$$

$$= \max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$

$$\boldsymbol{w} = [\ldots, \boldsymbol{w}_f, \ldots, \boldsymbol{w}_g, \ldots]$$

Score Function - Marginal Polytope

- It will be convenient to express things in terms of points in the marginal polytope.

$$\max_y \sum_f \theta_f(y_f; x, \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}} \sum_f \sum_{y_f} \theta_f(y_f; x, \boldsymbol{w}) \mu_f(y_f)$$

$$= \max_{\boldsymbol{\mu} \in \mathcal{M}} \sum_f \boldsymbol{w}_f \cdot \sum_{y_f} \boldsymbol{\phi}_f(y_f, x) \mu_f(y_f)$$

$$= \max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$

$$\boldsymbol{\phi}(\boldsymbol{\mu}, x) = [\ldots, \sum_{y_f} \boldsymbol{\phi}_f(y_f, x) \mu_f(y_f), \ldots, \sum_{y_g} \boldsymbol{\phi}_g(y_g, x) \mu_g(y_g), \ldots]$$

$$\boldsymbol{w} = [\ldots, \boldsymbol{w}_f, \ldots, \boldsymbol{w}_g, \ldots]$$

The Learning Problem

The Learning Problem

- Given training sample D: $\{x^m, y^m\}_{m=1}^{M}$

The Learning Problem

- Given training sample D: $\{x^m, y^m\}_{m=1}^{M}$
- Each y^m is a complete label: $y^m = [y_1^m, \ldots, y_n^m]$

The Learning Problem

- Given training sample D: $\{x^m, y^m\}_{m=1}^{M}$
- Each y^m is a complete label: $y^m = [y_1^m, \ldots, y_n^m]$
- Goal: learn parameters **w**.

The Learning Problem

- Given training sample D: $\{x^m, y^m\}_{m=1}^M$
- Each y^m is a complete label: $y^m = [y_1^m, \ldots, y_n^m]$
- Goal: learn parameters **w**.
- Typical approach: minimize prediction loss on training data (plus regularization).

The Learning Problem

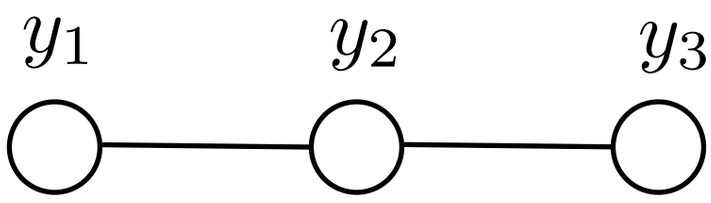
- Given training sample D: $\{x^m, y^m\}_{m=1}^M$
- Each y^m is a complete label: $y^m = [y_1^m, \ldots, y_n^m]$
- Goal: learn parameters **w**.
- Typical approach: minimize prediction loss on training data (plus regularization).

- Represent labels y^m via their corresponding vertices in the marginal polytope $\boldsymbol{\mu}^m$

The Learning Problem

- Given training sample D: $\{x^m, y^m\}_{m=1}^M$
- Each y^m is a complete label: $y^m = [y_1^m, \ldots, y_n^m]$
- Goal: learn parameters **w**.
- Typical approach: minimize prediction loss on training data (plus regularization).

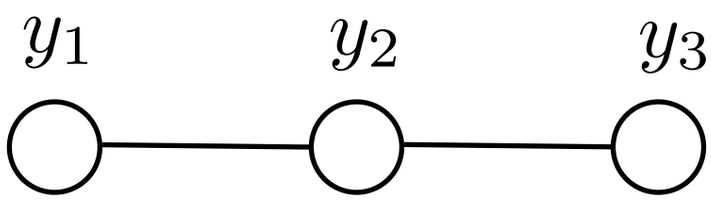
- Represent labels y^m via their corresponding vertices in the marginal polytope $\boldsymbol{\mu}^m$



The Learning Problem

- Given training sample D: $\{x^m, y^m\}_{m=1}^M$
- Each y^m is a complete label: $y^m = [y_1^m, \ldots, y_n^m]$
- Goal: learn parameters **w**.
- Typical approach: minimize prediction loss on training data (plus regularization).

- Represent labels y^m via their corresponding vertices in the marginal polytope $\boldsymbol{\mu}^m$

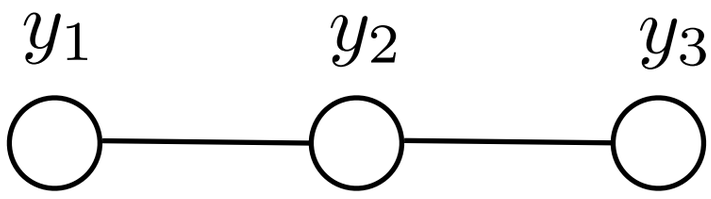


$y^m = [1, 0, 1]$

The Learning Problem

- Given training sample D: $\{x^m, y^m\}_{m=1}^M$
- Each y^m is a complete label: $y^m = [y_1^m, \ldots, y_n^m]$
- Goal: learn parameters **w**.
- Typical approach: minimize prediction loss on training data (plus regularization).

- Represent labels y^m via their corresponding vertices in the marginal polytope $\boldsymbol{\mu}^m$



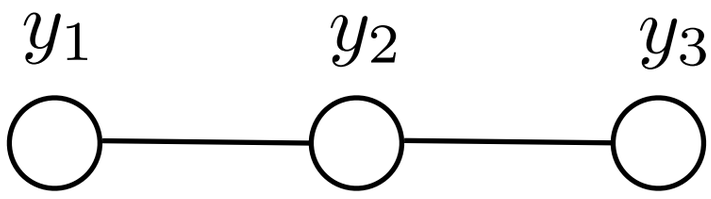
$$y^m = [1, 0, 1]$$

$$\boldsymbol{\mu}^m = [0, 1, 0, 0, 0, 0, 1, 0]$$

The Learning Problem

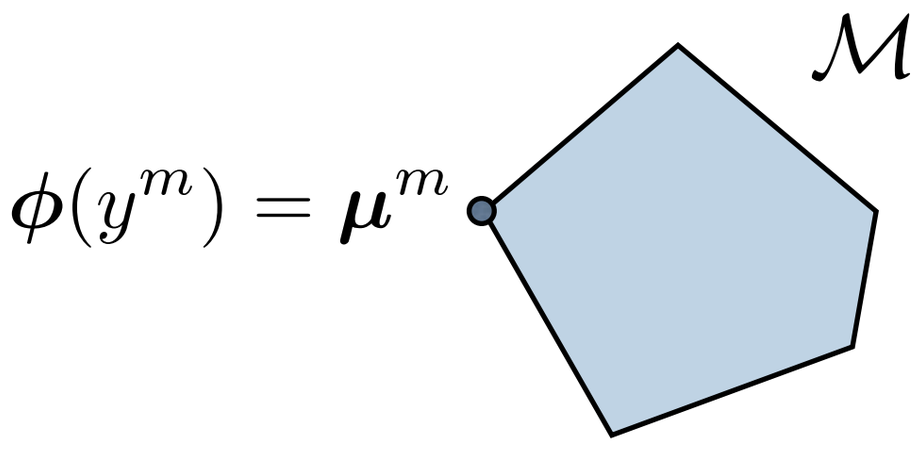
- Given training sample D: $\{x^m, y^m\}_{m=1}^M$
- Each y^m is a complete label: $y^m = [y_1^m, \ldots, y_n^m]$
- Goal: learn parameters **w**.
- Typical approach: minimize prediction loss on training data (plus regularization).

- Represent labels y^m via their corresponding vertices in the marginal polytope $\boldsymbol{\mu}^m$



$$y^m = [1, 0, 1]$$

$$\boldsymbol{\mu}^m = [0, 1, 0, 0, 0, 0, 1, 0]$$



Tractable Predictors

- Our desired predictor has the form:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$

Tractable Predictors

- Our desired predictor has the form:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$

- When the MRF has complex structure, there are two problems:
  - This maximization is intractable.
  - Learning is intractable.

Tractable Predictors

- Our desired predictor has the form:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$

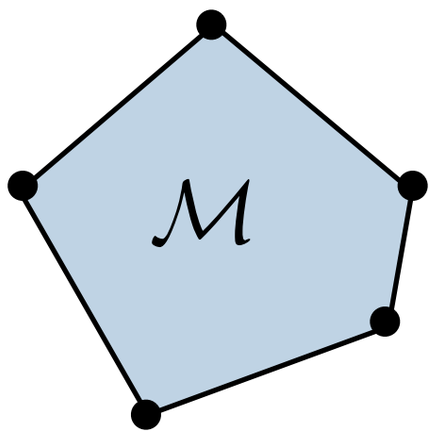
- When the MRF has complex structure, there are two problems:
  - This maximization is intractable.
  - Learning is intractable.
- Solution: use LP relaxation in maximization:

Tractable Predictors

- Our desired predictor has the form:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$

- When the MRF has complex structure, there are two problems:
  - This maximization is intractable.
  - Learning is intractable.
- Solution: use LP relaxation in maximization:

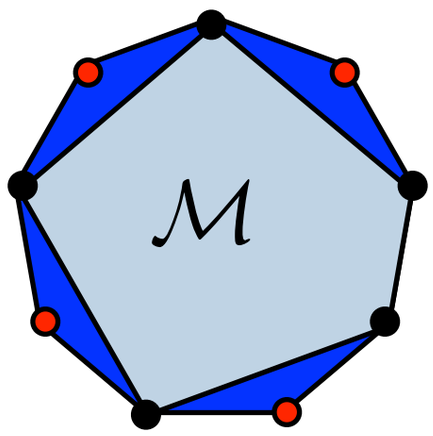


Tractable Predictors

- Our desired predictor has the form:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$

- When the MRF has complex structure, there are two problems:
  - This maximization is intractable.
  - Learning is intractable.
- Solution: use LP relaxation in maximization:

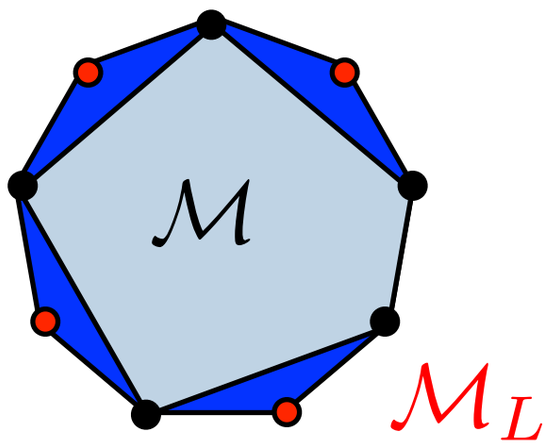


Tractable Predictors

- Our desired predictor has the form:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$

- When the MRF has complex structure, there are two problems:
  - This maximization is intractable.
  - Learning is intractable.
- Solution: use LP relaxation in maximization:



Tractable Predictors

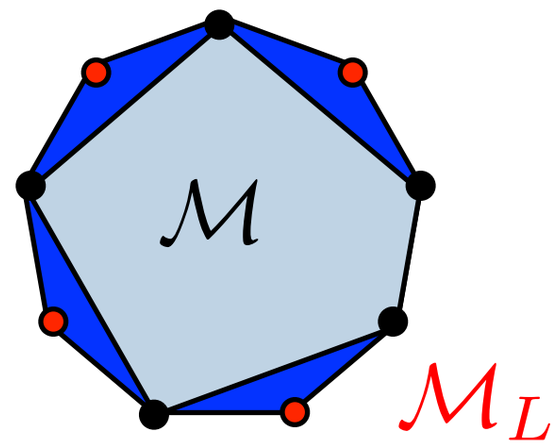
- Our desired predictor has the form:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$

- When the MRF has complex structure, there are two problems:
  - This maximization is intractable.
  - Learning is intractable.

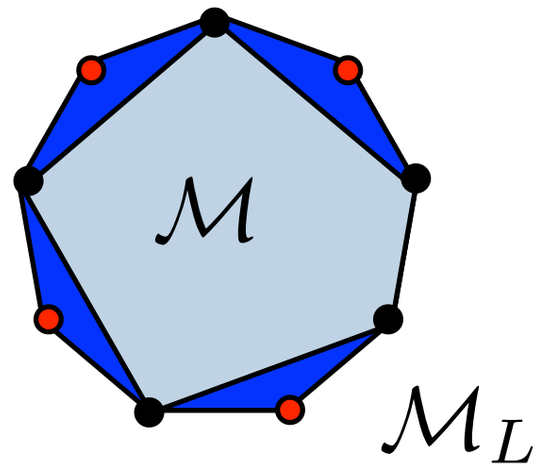
- Solution: use LP relaxation in maximization:

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$



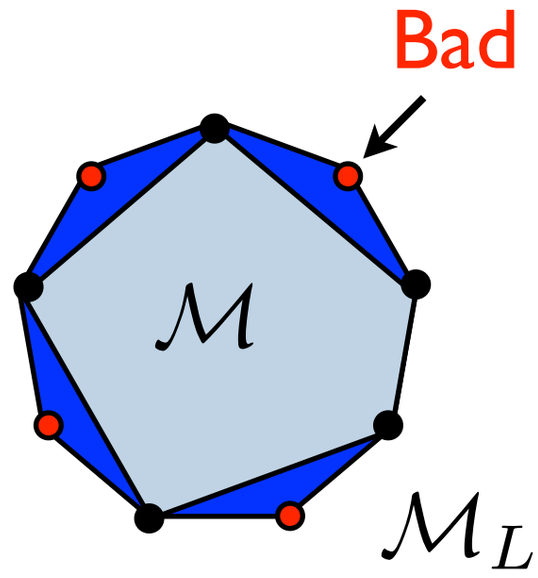
Tractable Predictors

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$



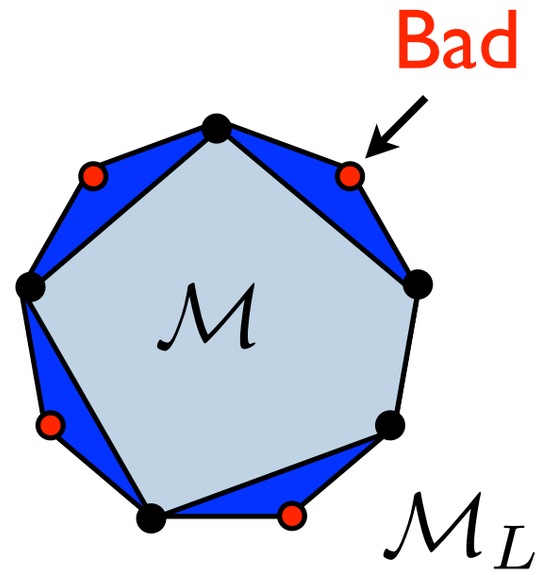
Tractable Predictors

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$



Tractable Predictors

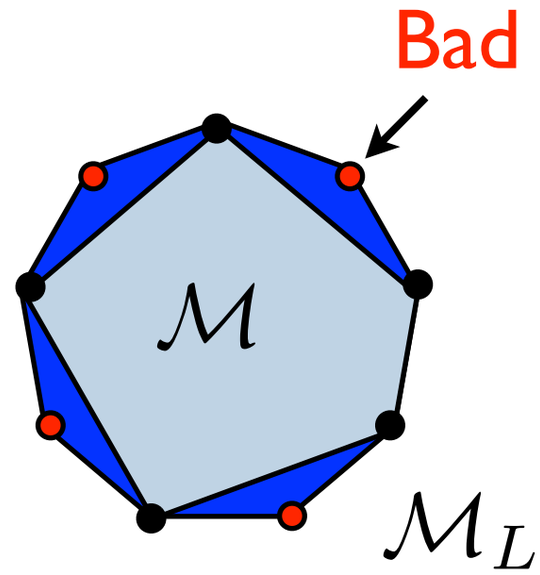
$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$



- We have added new possible labels (the fractional vertices).

Tractable Predictors

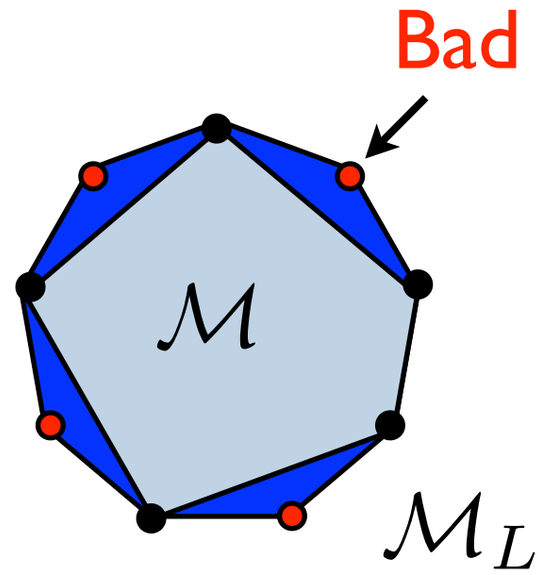
$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$



- We have added new possible labels (the fractional vertices).
- Turns the problem into "standard" structured prediction, with an extended label set.

Tractable Predictors

$$\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x)$$



- We have added new possible labels (the fractional vertices).
- Turns the problem into "standard" structured prediction, with an extended label set.
- The new vertices will never appear as true labels, so we would ideally never obtain them as maxima.

Outline

- Solving LP relaxations:
  - Subgradient algorithms
  - Coordinate descent algorithms
- Parameter learning with LP relaxations:
  - Structured Prediction
  - Structured LP perceptron.
  - Coordinate descent algorithms.
  - Cutting plane algorithms.
  - Pseudo-max algorithms.

Algorithmic Separability

- Suppose our training data $(x^m, \boldsymbol{\mu}^m)$ is such that there exists a weight vector $\boldsymbol{w}^*$:

Algorithmic Separability

- Suppose our training data $(x^m, \boldsymbol{\mu}^m)$ is such that there exists a weight vector $\boldsymbol{w}^*$:

$$\boldsymbol{\mu}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) \quad , \quad \forall m$$

Algorithmic Separability

- Suppose our training data $(x^m, \boldsymbol{\mu}^m)$ is such that there exists a weight vector $\boldsymbol{w}^*$:

$$\boldsymbol{\mu}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) \quad , \quad \forall m$$

- Called *algorithmically separable* data *[Kulesza & Pereira 07]*.

Algorithmic Separability

- Suppose our training data $(x^m, \boldsymbol{\mu}^m)$ is such that there exists a weight vector $\boldsymbol{w}^*$:

$$\boldsymbol{\mu}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) \quad , \quad \forall m$$

- Called *algorithmically separable* data *[Kulesza & Pereira 07]*.

- True vertex $\boldsymbol{\mu}^m$ has higher scores than all local polytope vertices (integral *and* fractional)!

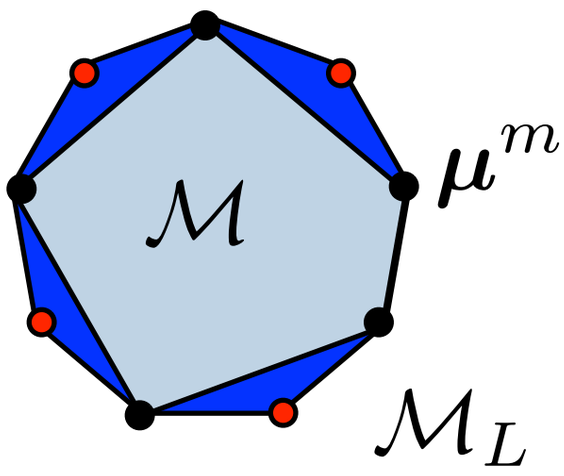
Algorithmic Separability

- Suppose our training data $(x^m, \boldsymbol{\mu}^m)$ is such that there exists a weight vector $\boldsymbol{w}^*$:

$$\boldsymbol{\mu}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) \quad , \quad \forall m$$

- Called *algorithmically separable* data *[Kulesza & Pereira 07]*.

- True vertex $\boldsymbol{\mu}^m$ has higher scores than all local polytope vertices (integral *and* fractional)!



Algorithmic Separability

Algorithmic Separability

- Assume the data is algorithmically separable.

Algorithmic Separability

- Assume the data is algorithmically separable.
- Can we find a **w** that attains zero error:

Algorithmic Separability

- Assume the data is algorithmically separable.
- Can we find a **w** that attains zero error:

$$\boldsymbol{\mu}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) \quad , \quad \forall m$$

Algorithmic Separability

- Assume the data is algorithmically separable.
- Can we find a **w** that attains zero error:

$$\boldsymbol{\mu}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) \quad , \quad \forall m$$

- We know there exists at least one such **w** (i.e., $\boldsymbol{w}^*$).

Algorithmic Separability

- Assume the data is algorithmically separable.
- Can we find a **w** that attains zero error:

$$\boldsymbol{\mu}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) \quad , \quad \forall m$$

- We know there exists at least one such **w** (i.e., $\boldsymbol{w}^*$).
- Equivalent to the following linear constraints:

Algorithmic Separability

- Assume the data is algorithmically separable.
- Can we find a **w** that attains zero error:

$$\boldsymbol{\mu}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) \quad , \quad \forall m$$

- We know there exists at least one such **w** (i.e., $\boldsymbol{w}^*$).
- Equivalent to the following linear constraints:

$$\forall \boldsymbol{\mu} \in \text{Vertex}(\mathcal{M}_L), \boldsymbol{\mu} \neq \boldsymbol{\mu}^m$$

$$\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) > \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$$

Algorithmic Separability

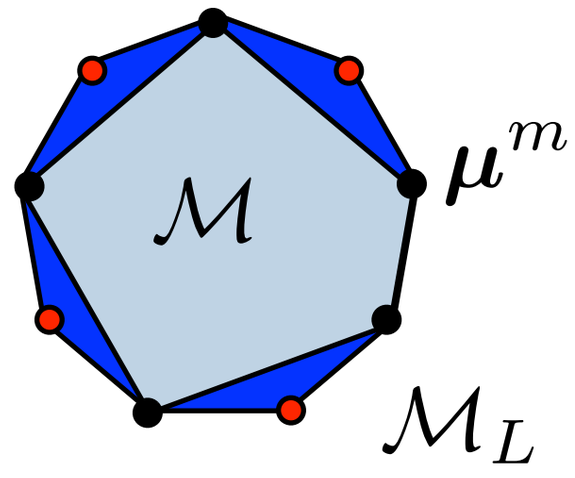
- Assume the data is algorithmically separable.
- Can we find a **w** that attains zero error:

$$\boldsymbol{\mu}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) \quad , \quad \forall m$$

- We know there exists at least one such **w** (i.e., $\boldsymbol{w}^*$).
- Equivalent to the following linear constraints:

$$\forall \boldsymbol{\mu} \in \text{Vertex}(\mathcal{M}_L), \boldsymbol{\mu} \neq \boldsymbol{\mu}^m$$

$$\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) > \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$$



Algorithmic Separability

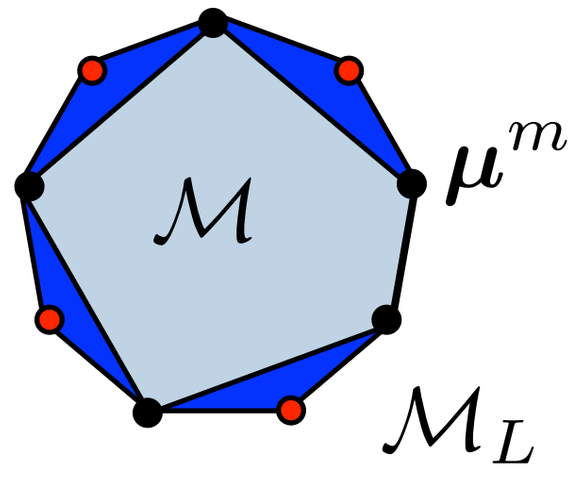
- Assume the data is algorithmically separable.
- Can we find a **w** that attains zero error:

$$\boldsymbol{\mu}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) \quad , \quad \forall m$$

- We know there exists at least one such **w** (i.e., $\boldsymbol{w}^*$).
- Equivalent to the following linear constraints:

$$\forall \boldsymbol{\mu} \in \text{Vertex}(\mathcal{M}_L), \boldsymbol{\mu} \neq \boldsymbol{\mu}^m$$

$$\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) > \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$$



- Can be solved using structured perceptron *[Collins 99, Kulesza and Pereira 07]*.

Structured LP Perceptron

Structured LP Perceptron

- Algorithm:

Structured LP Perceptron

- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.

Structured LP Perceptron

- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.
  - Repeat:

Structured LP Perceptron

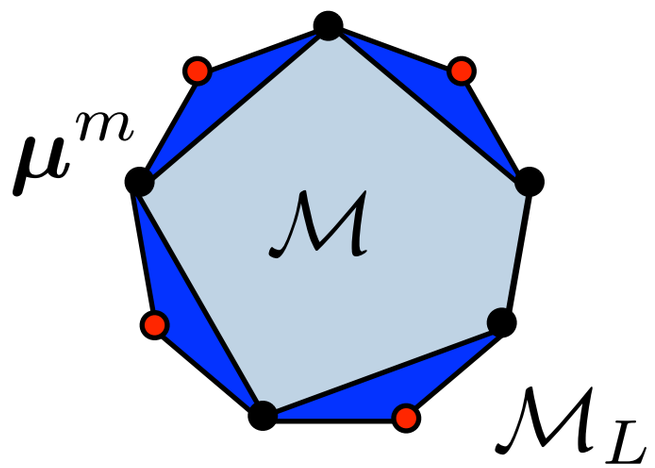
- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.
  - Repeat:
    - For m=1:M

Structured LP Perceptron

- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.
  - Repeat:
    - For m=1:M
    - Calculate: $\bar{\boldsymbol{\mu}}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}_t \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$

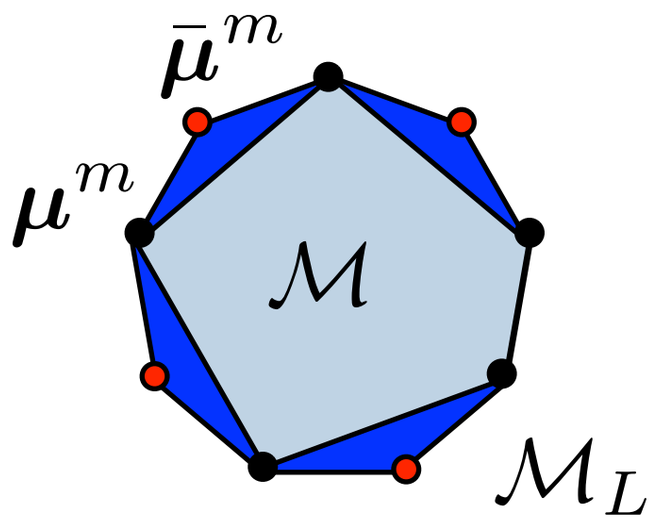
Structured LP Perceptron

- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.
  - Repeat:
    - For m=1:M
    - Calculate: $\bar{\boldsymbol{\mu}}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}_t \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$



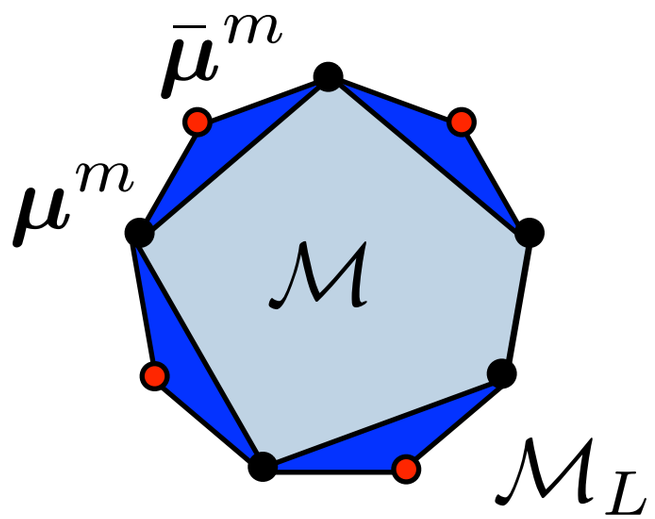
Structured LP Perceptron

- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.
  - Repeat:
    - For m=1:M
    - Calculate: $\bar{\boldsymbol{\mu}}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}_t \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$



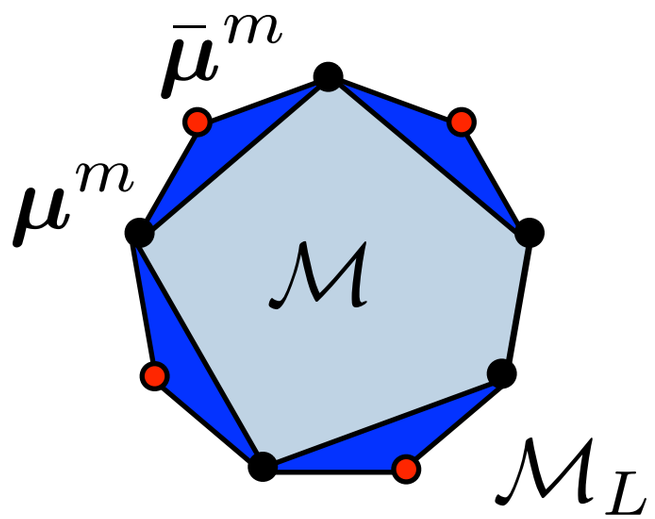
Structured LP Perceptron

- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.
  - Repeat:
    - For m=1:M
    - Calculate: $\bar{\boldsymbol{\mu}}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}_t \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$
    - If $\bar{\boldsymbol{\mu}}^m \neq \boldsymbol{\mu}^m$



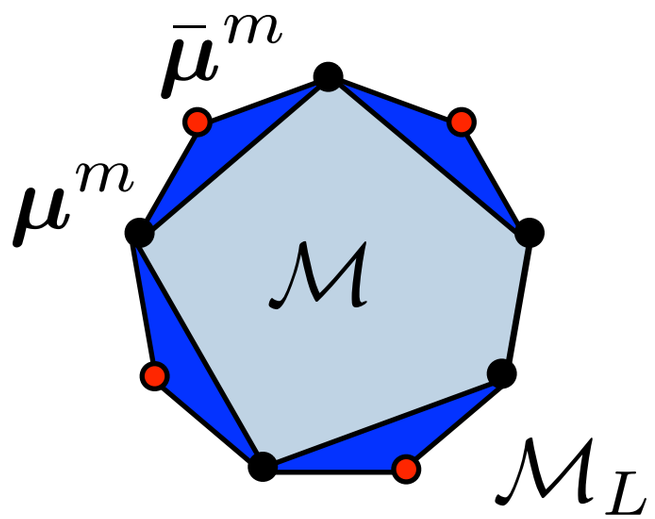
Structured LP Perceptron

- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.
  - Repeat:
    - For m=1:M
      - Calculate: $\bar{\boldsymbol{\mu}}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}_t \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$
      - If $\bar{\boldsymbol{\mu}}^m \neq \boldsymbol{\mu}^m$
        - Update: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) - \boldsymbol{\phi}(\bar{\boldsymbol{\mu}}^m, x^m)$

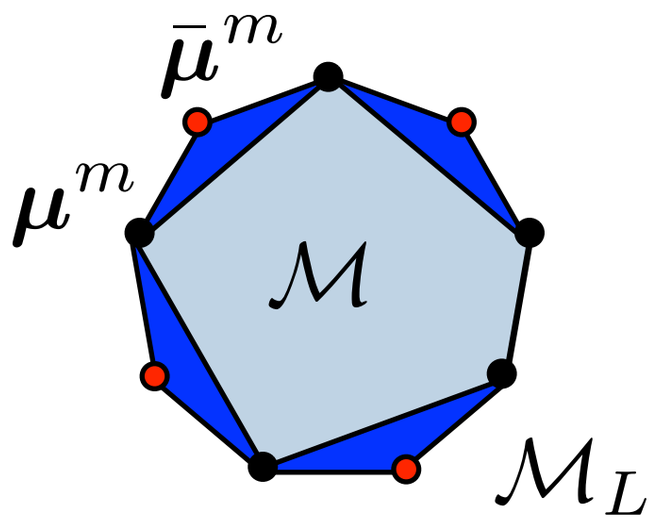


Structured LP Perceptron

- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.
  - Repeat:
    - For m=1:M
      - Calculate: $\bar{\boldsymbol{\mu}}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}_t \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$
      - If $\bar{\boldsymbol{\mu}}^m \neq \boldsymbol{\mu}^m$
        - Update: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) - \boldsymbol{\phi}(\bar{\boldsymbol{\mu}}^m, x^m)$
      - else: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$

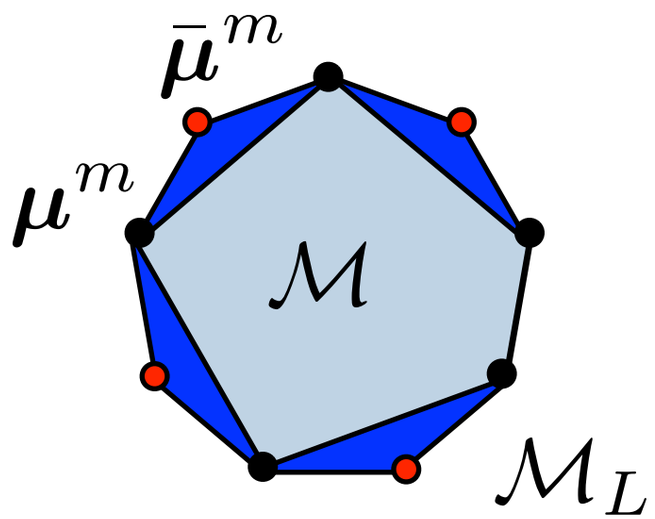


Structured LP Perceptron



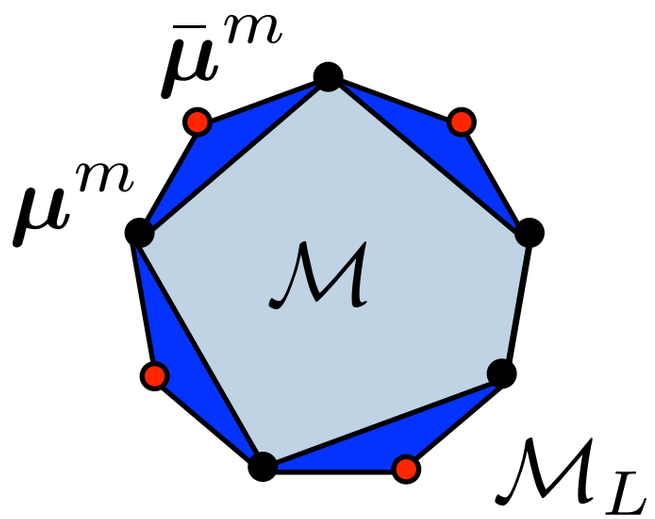
- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.
  - Repeat:
    - For m=1:M
      - Calculate: $\bar{\boldsymbol{\mu}}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}_t \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$
      - If $\bar{\boldsymbol{\mu}}^m \neq \boldsymbol{\mu}^m$
        - Update: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) - \boldsymbol{\phi}(\bar{\boldsymbol{\mu}}^m, x^m)$
      - else: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$
      - t = t+1

Structured LP Perceptron



- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.
  - Repeat:
    - For m=1:M
      - Calculate: $\bar{\boldsymbol{\mu}}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}_t \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$
      - If $\bar{\boldsymbol{\mu}}^m \neq \boldsymbol{\mu}^m$
        - Update: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) - \boldsymbol{\phi}(\bar{\boldsymbol{\mu}}^m, x^m)$
      - else: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$
      - t = t+1
  - If no update performed, break.

Structured LP Perceptron



- Algorithm:
- Initialize: $\boldsymbol{w}_0 = \mathbf{0}$, t=0.
  - Repeat:
    - For m=1:M
      - Calculate: $\bar{\boldsymbol{\mu}}^m = \arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w}_t \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m)$
      - If $\bar{\boldsymbol{\mu}}^m \neq \boldsymbol{\mu}^m$
        - Update: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) - \boldsymbol{\phi}(\bar{\boldsymbol{\mu}}^m, x^m)$
      - else: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$
      - t = t+1
  - If no update performed, break.

May be fractional

Structured LP Perceptron

Structured LP Perceptron

- Will find a **w** that separates the data, after a finite number of iterations, assuming algorithmic separability.

Structured LP Perceptron

- Will find a **w** that separates the data, after a finite number of iterations, assuming algorithmic separability.
- Essentially a standard structured prediction problem, but with an extended label set (including fractional vertices).

Structured LP Perceptron

- Will find a **w** that separates the data, after a finite number of iterations, assuming algorithmic separability.
- Essentially a standard structured prediction problem, but with an extended label set (including fractional vertices).
- The true labels are always integral, but this is not required for the algorithm to work.

Structured LP Perceptron

- Will find a **w** that separates the data, after a finite number of iterations, assuming algorithmic separability.
- Essentially a standard structured prediction problem, but with an extended label set (including fractional vertices).
- The true labels are always integral, but this is not required for the algorithm to work.
- Requires solving an LP at every sample point.

Outline

- Solving LP relaxations:
  - Subgradient algorithms
  - Coordinate descent algorithms
- Parameter learning with LP relaxations:
  - Structured Prediction
  - Structured LP perceptron.
  - Coordinate descent algorithms.
  - Cutting plane algorithms.
  - Pseudo-max algorithms.

Non-separable case

Non-separable case

- Generally we cannot guarantee even algorithmic separability.

Non-separable case

- Generally we cannot guarantee even algorithmic separability.
- Would like to find **w** with minimum error on the training set (plus regularization).

Non-separable case

- Generally we cannot guarantee even algorithmic separability.
- Would like to find **w** with minimum error on the training set (plus regularization).
- How do we define error?

Non-separable case

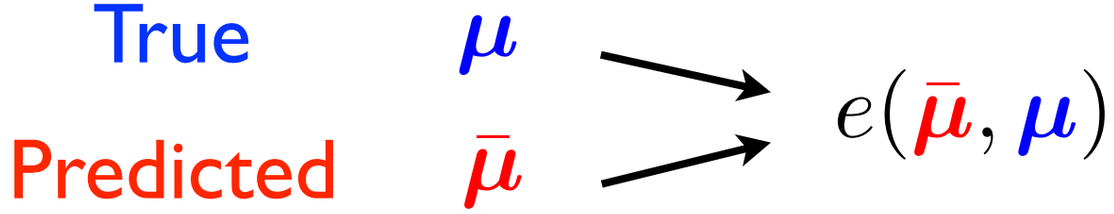
- Generally we cannot guarantee even algorithmic separability.
- Would like to find **w** with minimum error on the training set (plus regularization).
- How do we define error?

True μ

Predicted $\bar{\mu}$

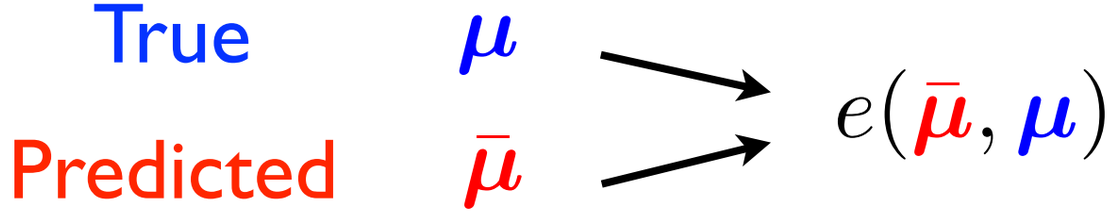
Non-separable case

- Generally we cannot guarantee even algorithmic separability.
- Would like to find **w** with minimum error on the training set (plus regularization).
- How do we define error?



Non-separable case

- Generally we cannot guarantee even algorithmic separability.
- Would like to find **w** with minimum error on the training set (plus regularization).
- How do we define error?



How much do we pay for predicting $\bar{\boldsymbol{\mu}}$ instead of $\boldsymbol{\mu}$?

Non-separable case

- Generally we cannot guarantee even algorithmic separability.
- Would like to find **w** with minimum error on the training set (plus regularization).
- How do we define error?

True $\boldsymbol{\mu}$

Predicted $\bar{\boldsymbol{\mu}}$

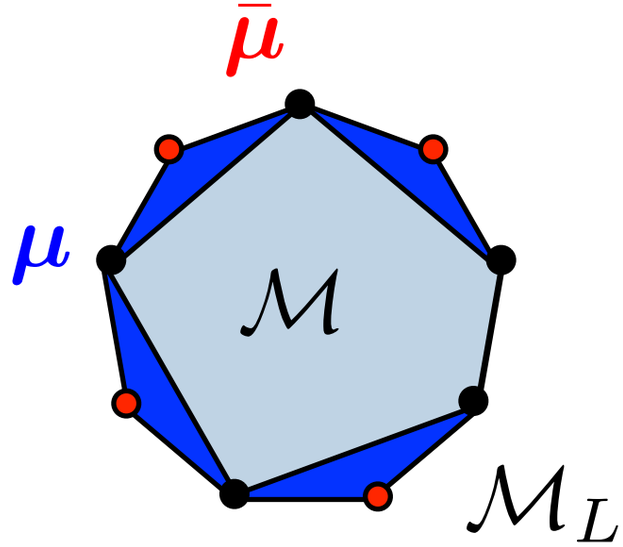
$\rightarrow e(\bar{\boldsymbol{\mu}}, \boldsymbol{\mu})$

How much do we pay for predicting $\bar{\boldsymbol{\mu}}$ instead of $\boldsymbol{\mu}$?

$$e(\bar{\boldsymbol{\mu}}, \boldsymbol{\mu}) = \begin{cases} 1 & \boldsymbol{\mu} \neq \bar{\boldsymbol{\mu}} \\ 0 & \boldsymbol{\mu} = \bar{\boldsymbol{\mu}} \end{cases}$$

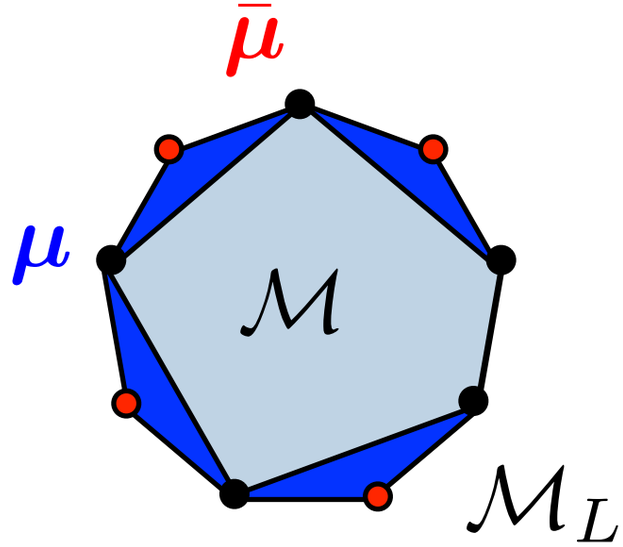
Zero one loss

Hamming Loss



Hamming Loss

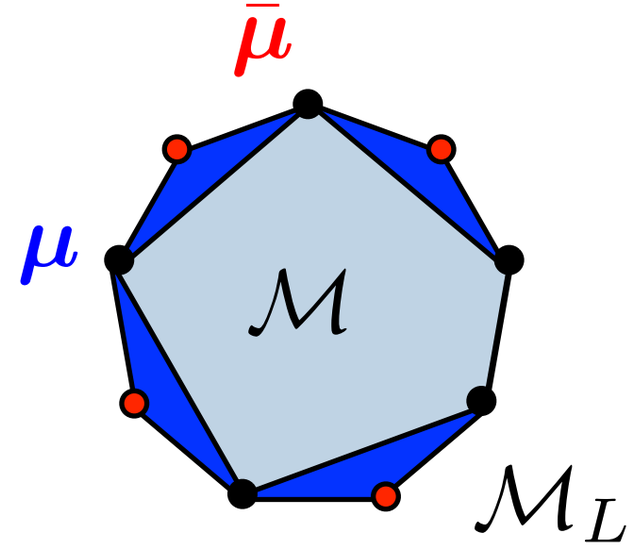
$$\begin{bmatrix} \mu_1(0) \\ \mu_1(1) \\ \mu_2(0) \\ \mu_2(1) \end{bmatrix}$$



Hamming Loss

$$y_1 = 1, y_2 = 0$$

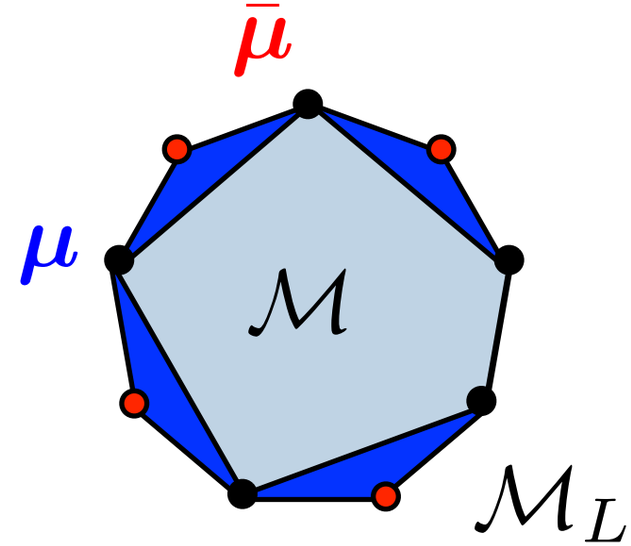
$$\begin{bmatrix} \mu_1(0) \\ \mu_1(1) \\ \mu_2(0) \\ \mu_2(1) \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$



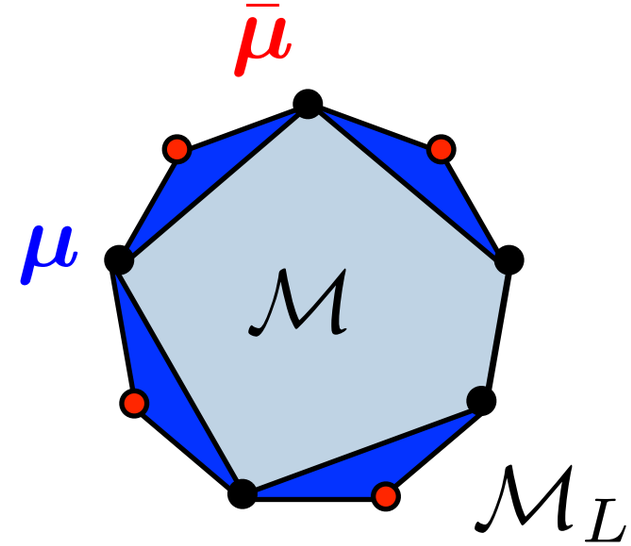
Hamming Loss

$$y_1 = 1, y_2 = 0 \quad y_1 = 0, y_2 = 0$$

$$\begin{bmatrix} \mu_1(0) \\ \mu_1(1) \\ \mu_2(0) \\ \mu_2(1) \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$



Hamming Loss

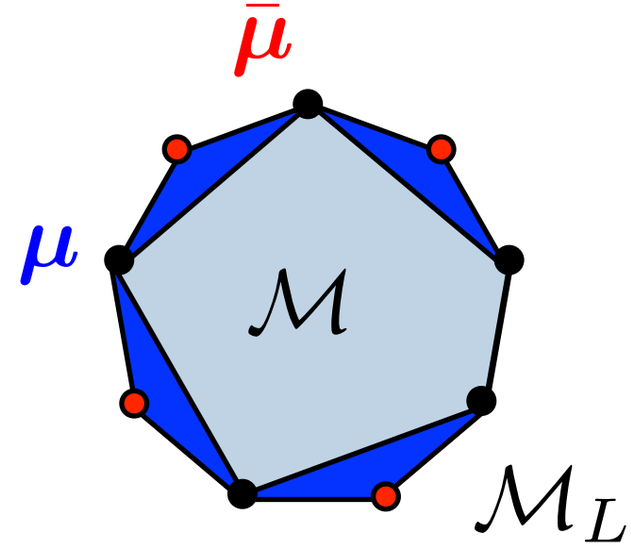


$y_1 = 1, y_2 = 0 \quad y_1 = 0, y_2 = 0$

$$\begin{bmatrix} \mu_1(0) \\ \mu_1(1) \\ \mu_2(0) \\ \mu_2(1) \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$\longrightarrow$ Loss=1

Hamming Loss



$y_1 = 1, y_2 = 0$ $\quad y_1 = 0, y_2 = 0$

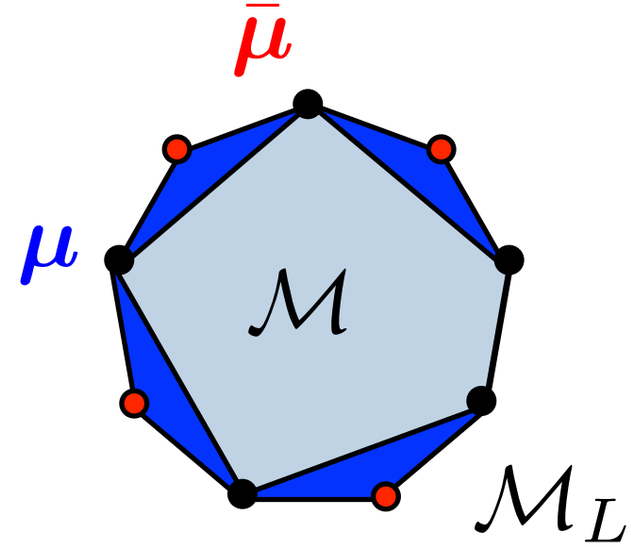
$$\begin{bmatrix} \mu_1(0) \\ \mu_1(1) \\ \mu_2(0) \\ \mu_2(1) \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$\longrightarrow$ Loss=1

$y_1 = 1, y_2 = 0$

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Hamming Loss



$y_1 = 1, y_2 = 0$ $y_1 = 0, y_2 = 0$

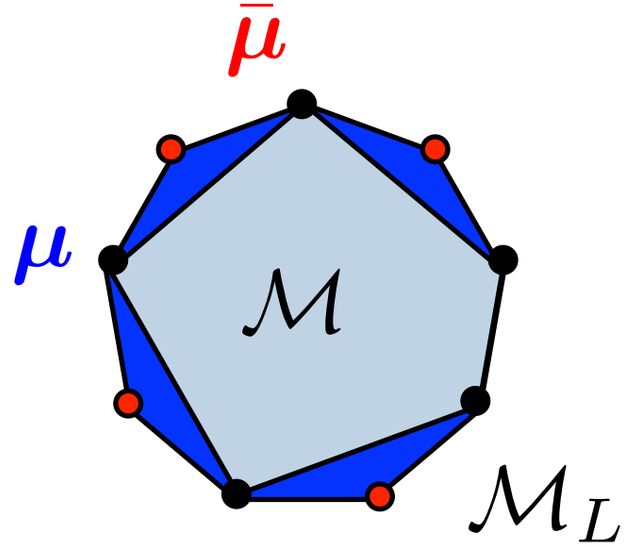
$$\begin{bmatrix} \mu_1(0) \\ \mu_1(1) \\ \mu_2(0) \\ \mu_2(1) \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \longrightarrow$$

Loss=1

$y_1 = 1, y_2 = 0$

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0.3 \\ 0.7 \\ 0.2 \\ 0.8 \end{bmatrix}$$

Hamming Loss



$y_1 = 1, y_2 = 0 \quad y_1 = 0, y_2 = 0$

$$\begin{bmatrix} \mu_1(0) \\ \mu_1(1) \\ \mu_2(0) \\ \mu_2(1) \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

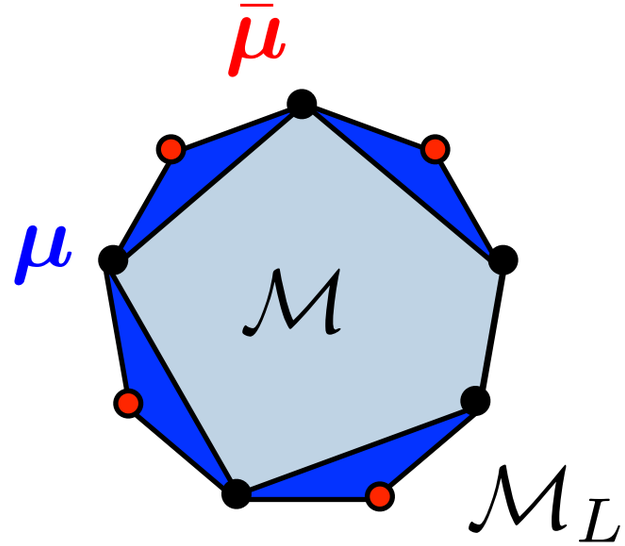
$\longrightarrow$ Loss=1

$y_1 = 1, y_2 = 0$

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0.3 \\ 0.7 \\ 0.2 \\ 0.8 \end{bmatrix}$$

$\longrightarrow$ Loss=0.3+0.8

Hamming Loss



$y_1 = 1, y_2 = 0$ $\quad y_1 = 0, y_2 = 0$

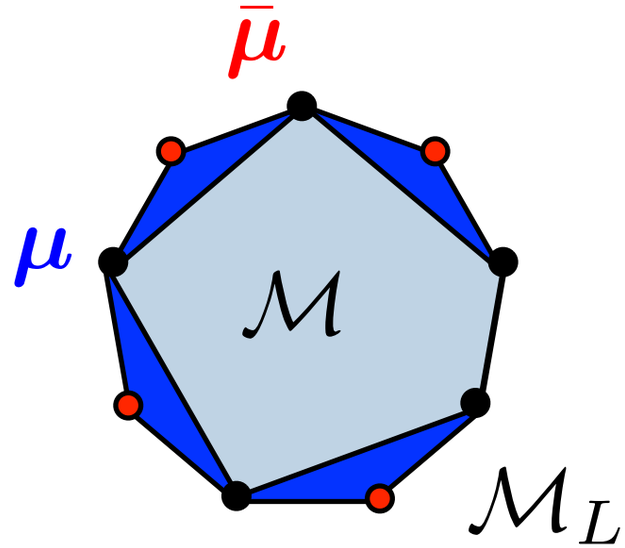
$$\begin{bmatrix} \mu_1(0) \\ \mu_1(1) \\ \mu_2(0) \\ \mu_2(1) \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \longrightarrow \text{Loss=1}$$

$y_1 = 1, y_2 = 0$

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0.3 \\ 0.7 \\ 0.2 \\ 0.8 \end{bmatrix} \longrightarrow \text{Loss=0.3+0.8}$$

$$e(\bar{\boldsymbol{\mu}}, \boldsymbol{\mu}) = \sum_i \sum_{y_i} \bar{\mu}_i(y_i) \left(1 - \mu_i(y_i)\right)$$ Hamming loss

Hamming Loss



$y_1 = 1, y_2 = 0$ $\quad y_1 = 0, y_2 = 0$

$$\begin{bmatrix} \mu_1(0) \\ \mu_1(1) \\ \mu_2(0) \\ \mu_2(1) \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \longrightarrow$$

Loss=1

$y_1 = 1, y_2 = 0$

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0.3 \\ 0.7 \\ 0.2 \\ 0.8 \end{bmatrix} \longrightarrow$$

Loss=0.3+0.8

Factors nicely!

$$e(\bar{\boldsymbol{\mu}}, \boldsymbol{\mu}) = \sum_i \sum_{y_i} \bar{\mu}_i(y_i) \left(1 - \mu_i(y_i)\right)$$

Hamming loss

Hinge Loss

- We would like to find a **w** that minimizes the error:

$$e(\arg\max_{\boldsymbol{\mu}\in\mathcal{M}_L} \boldsymbol{w}\cdot\boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m), \boldsymbol{\mu}^m)$$

Predicted

True

Hinge Loss

- We would like to find a **w** that minimizes the error:

$$e(\arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m), \boldsymbol{\mu}^m)$$

Predicted

True

- NP hard (even for binary classification). Use surrogate hinge loss instead *[Taskar et al. 03]*

Hinge Loss

- We would like to find a **w** that minimizes the error:

$$e(\arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m), \boldsymbol{\mu}^m)$$

Predicted

True

- NP hard (even for binary classification). Use surrogate hinge loss instead *[Taskar et al. 03]*

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} [\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) + e(\boldsymbol{\mu}, \boldsymbol{\mu}^m)]$$

Hinge Loss

- We would like to find a **w** that minimizes the error:

$$e(\arg \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m), \boldsymbol{\mu}^m)$$

Predicted

True

- NP hard (even for binary classification). Use surrogate hinge loss instead *[Taskar et al. 03]*

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} [\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) + e(\boldsymbol{\mu}, \boldsymbol{\mu}^m)]$$

- Upper bounds the true loss above (as a function of **w**).

Hinge Loss

- The hinge loss is:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} [\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) + e(\boldsymbol{\mu}, \boldsymbol{\mu}^m)]$$

Hinge Loss

- The hinge loss is:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} [\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) + e(\boldsymbol{\mu}, \boldsymbol{\mu}^m)]$$

- Can be written as:

Hinge Loss

- The hinge loss is:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \left[\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) + e(\boldsymbol{\mu}, \boldsymbol{\mu}^m)\right]$$

- Can be written as:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \left[\sum_f \theta_f(y_f; x^m, y^m, \boldsymbol{w}) \mu_f(y_f) + \sum_i \theta_i(y_i; x^m, y^m, \boldsymbol{w}) \mu_i(y_i)\right]$$

Hinge Loss

- The hinge loss is:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \left[\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) + e(\boldsymbol{\mu}, \boldsymbol{\mu}^m)\right]$$

- Can be written as:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \left[\sum_f \theta_f(y_f; x^m, y^m, \boldsymbol{w}) \mu_f(y_f) + \sum_i \theta_i(y_i; x^m, y^m, \boldsymbol{w}) \mu_i(y_i)\right]$$

- Where:

Hinge Loss

- The hinge loss is:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \left[\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) + e(\boldsymbol{\mu}, \boldsymbol{\mu}^m)\right]$$

- Can be written as:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \left[\sum_f \theta_f(y_f; x^m, y^m, \boldsymbol{w}) \mu_f(y_f) + \sum_i \theta_i(y_i; x^m, y^m, \boldsymbol{w}) \mu_i(y_i)\right]$$

- Where:

$$\theta_f(y_f; x^m, y^m, \boldsymbol{w}) = \boldsymbol{w}_f \cdot \boldsymbol{\phi}_f(y_f; x^m) - \boldsymbol{w}_f \cdot \boldsymbol{\phi}_f(y_f^m; x^m)$$

Hinge Loss

- The hinge loss is:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \left[\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) + e(\boldsymbol{\mu}, \boldsymbol{\mu}^m)\right]$$

- Can be written as:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \left[\sum_f \theta_f(y_f; x^m, y^m, \boldsymbol{w}) \mu_f(y_f) + \sum_i \theta_i(y_i; x^m, y^m, \boldsymbol{w}) \mu_i(y_i)\right]$$

- Where:

$$\theta_f(y_f; x^m, y^m, \boldsymbol{w}) = \boldsymbol{w}_f \cdot \boldsymbol{\phi}_f(y_f; x^m) - \boldsymbol{w}_f \cdot \boldsymbol{\phi}_f(y_f^m; x^m)$$

$$\theta_i(y_i; x^m, y^m, \boldsymbol{w}) = \boldsymbol{w}_i \cdot \boldsymbol{\phi}_i(y_i, x^m) - \boldsymbol{w}_i \cdot \boldsymbol{\phi}_i(y_i^m; x^m, \boldsymbol{w}) + 1 - \mu_i^m(y_i)$$

Hinge Loss

- The hinge loss is:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \left[\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}, x^m) - \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{\mu}^m, x^m) + e(\boldsymbol{\mu}, \boldsymbol{\mu}^m)\right]$$

- Can be written as:

$$\ell_h(x^m, \boldsymbol{\mu}^m; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \underbrace{\left[\sum_f \theta_f(y_f; x^m, y^m, \boldsymbol{w})\mu_f(y_f) + \sum_i \theta_i(y_i; x^m, y^m, \boldsymbol{w})\mu_i(y_i)\right]}_{\boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu}}$$

- Where:

$$\theta_f(y_f; x^m, y^m, \boldsymbol{w}) = \boldsymbol{w}_f \cdot \boldsymbol{\phi}_f(y_f; x^m) - \boldsymbol{w}_f \cdot \boldsymbol{\phi}_f(y_f^m; x^m)$$

$$\theta_i(y_i; x^m, y^m, \boldsymbol{w}) = \boldsymbol{w}_i \cdot \boldsymbol{\phi}_i(y_i, x^m) - \boldsymbol{w}_i \cdot \boldsymbol{\phi}_i(y_i^m; x^m, \boldsymbol{w}) + 1 - \mu_i^m(y_i)$$

The Learning Objective

- We want to find a **w** that minimizes the regularized hinge loss:

$$J(\boldsymbol{w}) = \frac{1}{M} \sum_{m} \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$$

- How do we minimize it?
- Tractable QP, but generic solvers will not scale well.

Stochastic Subgradient

- Minimize: $J(\boldsymbol{w}) = \frac{1}{M} \sum_{m} \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$

Stochastic Subgradient

- Minimize: $J(\boldsymbol{w}) = \frac{1}{M} \sum_{m} \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$
- Algorithm (e.g., Pegasos *[Shalev-Shwartz et al. 07]*):

Stochastic Subgradient

- Minimize: $J(\boldsymbol{w}) = \frac{1}{M} \sum_{m} \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$
- Algorithm (e.g., Pegasos *[Shalev-Shwartz et al. 07]*):
  - Initialize $\mathbf{w}_0$

Stochastic Subgradient

- Minimize: $J(\boldsymbol{w}) = \frac{1}{M} \sum_m \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$
- Algorithm (e.g., Pegasos *[Shalev-Shwartz et al. 07]*):
  - Initialize $\mathbf{w}_0$
  - Repeat:

Stochastic Subgradient

- Minimize: $J(\boldsymbol{w}) = \frac{1}{M} \sum_{m} \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$
- Algorithm (e.g., Pegasos *[Shalev-Shwartz et al. 07]*):
  - Initialize $\mathbf{w}_0$
  - Repeat:
    - Pick sample m

Stochastic Subgradient

- Minimize: $J(\boldsymbol{w}) = \frac{1}{M} \sum_m \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$
- Algorithm (e.g., Pegasos *[Shalev-Shwartz et al. 07]*):
  - Initialize $\mathbf{w}_0$
  - Repeat:
    - Pick sample m
    - Calculate subgradient $\mathbf{v}$ of $\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$

Stochastic Subgradient

- Minimize: $J(\boldsymbol{w}) = \frac{1}{M} \sum_m \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$
- Algorithm (e.g., Pegasos *[Shalev-Shwartz et al. 07]*):
  - Initialize $\mathbf{w}_0$
  - Repeat:
    - Pick sample m
    - Calculate subgradient $\mathbf{v}$ of $\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$
    - Update: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta \boldsymbol{v}$

Stochastic Subgradient

- Minimize: $J(\boldsymbol{w}) = \frac{1}{M} \sum_m \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$
- Algorithm (e.g., Pegasos *[Shalev-Shwartz et al. 07]*):
  - Initialize $\mathbf{w}_0$
  - Repeat:
    - Pick sample m
    - Calculate subgradient $\mathbf{v}$ of $\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$
    - Update: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta \boldsymbol{v}$
- Good: Guaranteed to Converge

Stochastic Subgradient

- Minimize: $J(\boldsymbol{w}) = \frac{1}{M} \sum_m \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$
- Algorithm (e.g., Pegasos *[Shalev-Shwartz et al. 07]*):
  - Initialize $\mathbf{w}_0$
  - Repeat:
    - Pick sample m
    - Calculate subgradient $\mathbf{v}$ of $\max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$
    - Update: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta \boldsymbol{v}$
- Good: Guaranteed to Converge
- Bad: Subgradient calculation requires solving LP

Duality Again

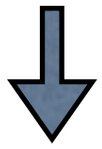
- Lets take the dual of the max part of the objective:

$$J(\boldsymbol{w}) = \frac{1}{M} \sum_{m} \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$$

Duality Again

- Lets take the dual of the max part of the objective:

$$J(\boldsymbol{w}) = \frac{1}{M} \sum_{m} \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$$



Duality

Duality Again

- Lets take the dual of the max part of the objective:

$$J(\boldsymbol{w}) = \frac{1}{M} \sum_m \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$$

⇩ Duality

$$\min_{\boldsymbol{w}} \frac{1}{M} \sum_m \min_{\boldsymbol{\delta}^m} g(\boldsymbol{\delta}^m, \boldsymbol{w}; x^m, \boldsymbol{\mu}^m) + \frac{C}{2} \|\boldsymbol{w}\|^2$$

Duality Again

- Lets take the dual of the max part of the objective:

$$J(\boldsymbol{w}) = \frac{1}{M} \sum_m \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$$

⇓ Duality

$$\min_{\boldsymbol{w}, \boldsymbol{\delta}^1, \ldots, \boldsymbol{\delta}^M} \frac{1}{M} \sum_m g(\boldsymbol{\delta}^m, \boldsymbol{w}; x^m, \boldsymbol{\mu}^m) + \frac{C}{2} \|\boldsymbol{w}\|^2$$

Duality Again

- Lets take the dual of the max part of the objective:

$$J(\boldsymbol{w}) = \frac{1}{M}\sum_m \max_{\boldsymbol{\mu}\in\mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu} + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

⇓ Duality

$$\min_{\boldsymbol{w},\boldsymbol{\delta}^1,\ldots,\boldsymbol{\delta}^M} \frac{1}{M}\sum_m g(\boldsymbol{\delta}^m, \boldsymbol{w}; x^m, \boldsymbol{\mu}^m) + \frac{C}{2}\|\boldsymbol{w}\|^2$$

Upper bound on hinge loss. Tuned by $\boldsymbol{\delta}$

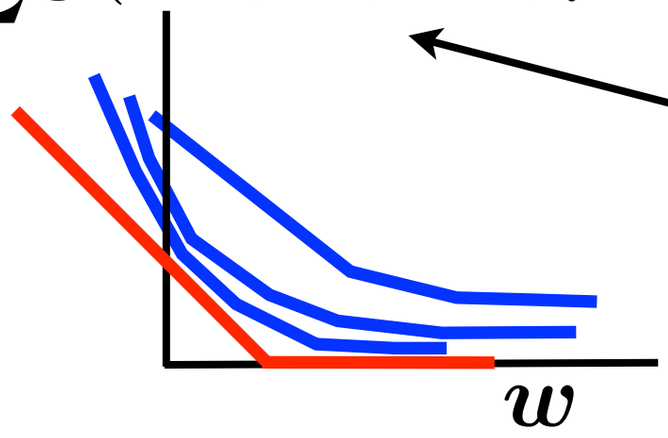
Duality Again

- Lets take the dual of the max part of the objective:

$$J(\boldsymbol{w}) = \frac{1}{M}\sum_m \max_{\boldsymbol{\mu}\in\mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu} + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

⇓ Duality

$$\min_{\boldsymbol{w},\boldsymbol{\delta}^1,\ldots,\boldsymbol{\delta}^M} \frac{1}{M}\sum_m g(\boldsymbol{\delta}^m, \boldsymbol{w}; x^m, \boldsymbol{\mu}^m) + \frac{C}{2}\|\boldsymbol{w}\|^2$$



Upper bound on hinge loss. Tuned by $\boldsymbol{\delta}$

Duality Again

- Lets take the dual of the max part of the objective:

$$J(\boldsymbol{w}) = \frac{1}{M} \sum_m \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$$

⇓ Duality

$$\min_{\boldsymbol{w}, \boldsymbol{\delta}^1, \ldots, \boldsymbol{\delta}^M} \frac{1}{M} \sum_m g(\boldsymbol{\delta}^m, \boldsymbol{w}; x^m, \boldsymbol{\mu}^m) + \frac{C}{2} \|\boldsymbol{w}\|^2$$

Duality Again

- Lets take the dual of the max part of the objective:

$$J(\boldsymbol{w}) = \frac{1}{M}\sum_m \max_{\boldsymbol{\mu}\in\mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu} + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

⇓ Duality

$$\min_{\boldsymbol{w},\boldsymbol{\delta}^1,\ldots,\boldsymbol{\delta}^M} \frac{1}{M}\sum_m g(\boldsymbol{\delta}^m, \boldsymbol{w}; x^m, \boldsymbol{\mu}^m) + \frac{C}{2}\|\boldsymbol{w}\|^2$$

- Dual variable: $\delta_{fi}^m(y_i)$. Message from factor f to variable i, for the m'th sample point.

Duality Again

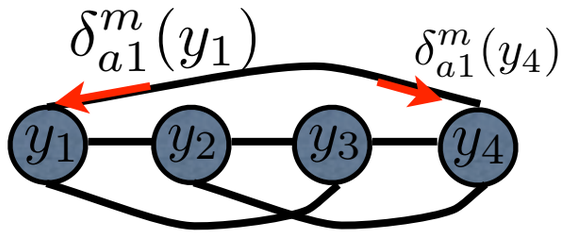
- Lets take the dual of the max part of the objective:

$$J(\boldsymbol{w}) = \frac{1}{M}\sum_m \max_{\boldsymbol{\mu}\in\mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu} + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

⇓ Duality

$$\min_{\boldsymbol{w},\boldsymbol{\delta}^1,\ldots,\boldsymbol{\delta}^M} \frac{1}{M}\sum_m g(\boldsymbol{\delta}^m, \boldsymbol{w}; x^m, \boldsymbol{\mu}^m) + \frac{C}{2}\|\boldsymbol{w}\|^2$$

- Dual variable: $\delta_{fi}^m(y_i)$. Message from factor f to variable i, for the m'th sample point.



Duality Again

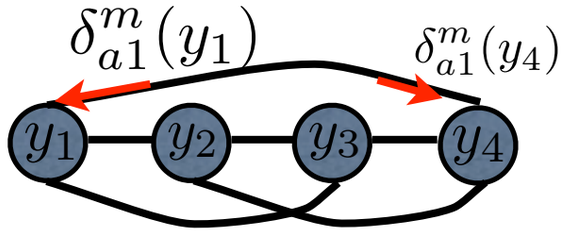
- Lets take the dual of the max part of the objective:

$$J(\boldsymbol{w}) = \frac{1}{M} \sum_m \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$$

⇓ Duality

$$\min_{\boldsymbol{w}, \boldsymbol{\delta}^1, \ldots, \boldsymbol{\delta}^M} \frac{1}{M} \sum_m g(\boldsymbol{\delta}^m, \boldsymbol{w}; x^m, \boldsymbol{\mu}^m) + \frac{C}{2} \|\boldsymbol{w}\|^2$$

- Dual variable: $\delta_{fi}^m(y_i)$. Message from factor f to variable i, for the m'th sample point.



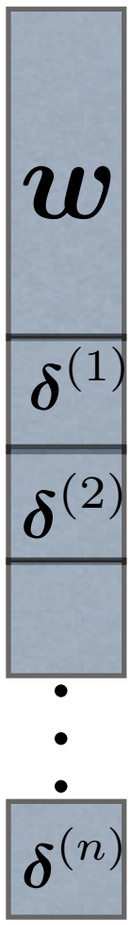
- Objective is jointly convex in **w**, $\boldsymbol{\delta}$

The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*

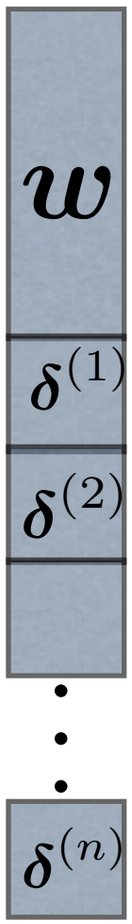
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



The DLPW Algorithm

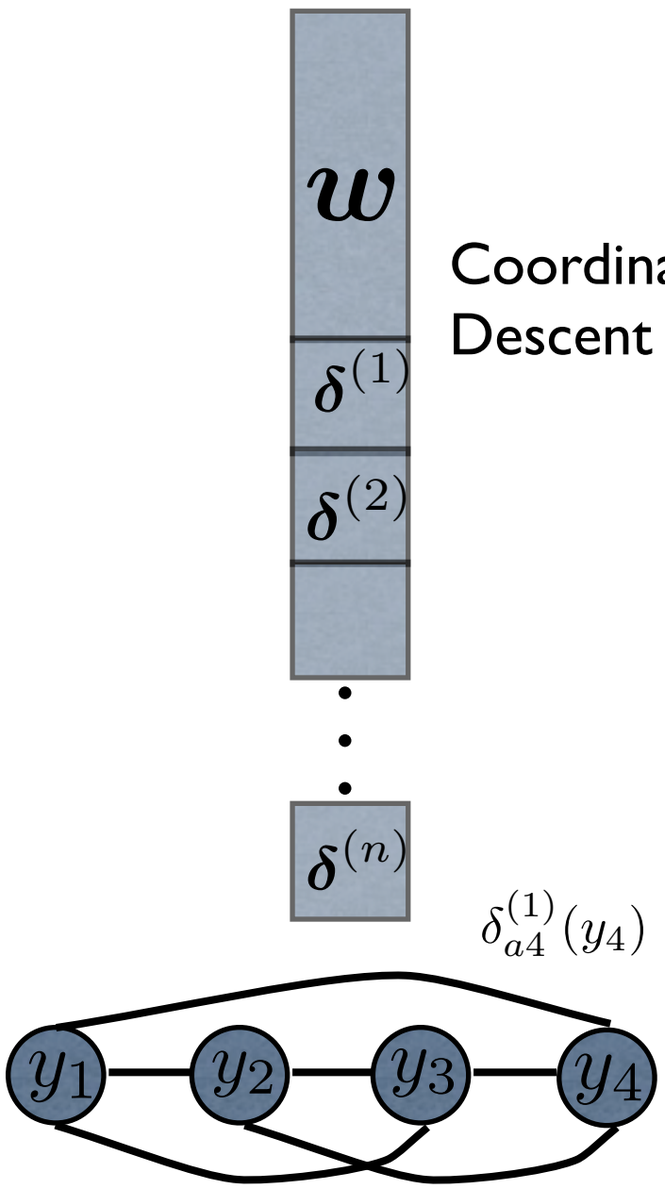
- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



Coordinate Descent

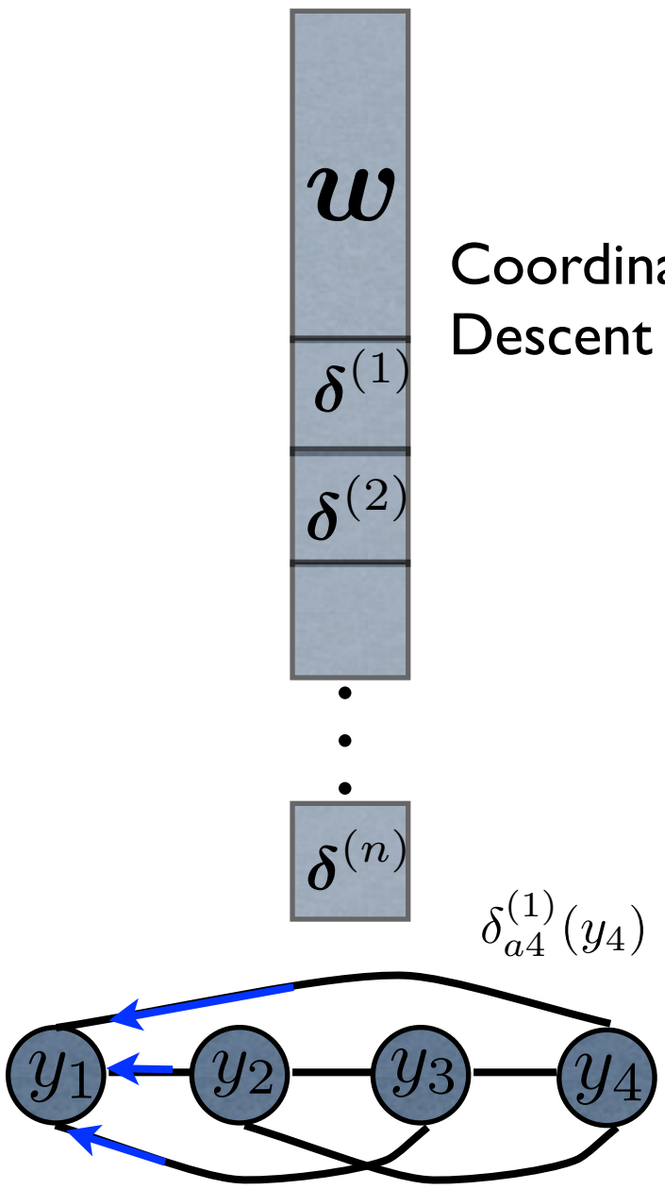
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



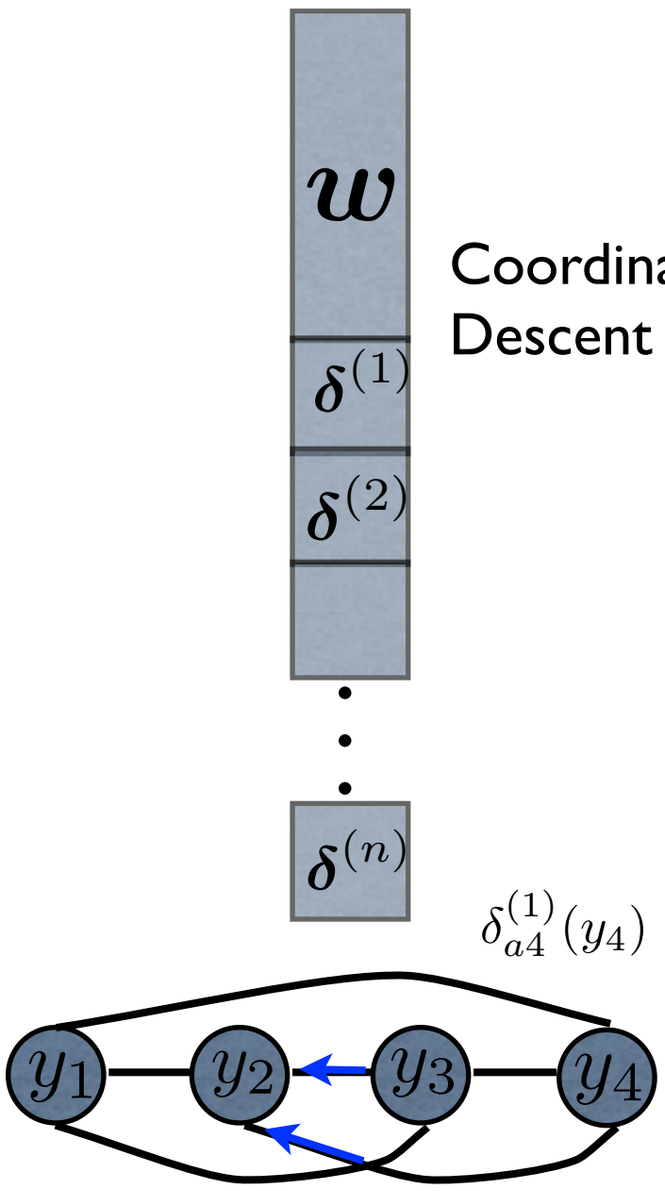
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



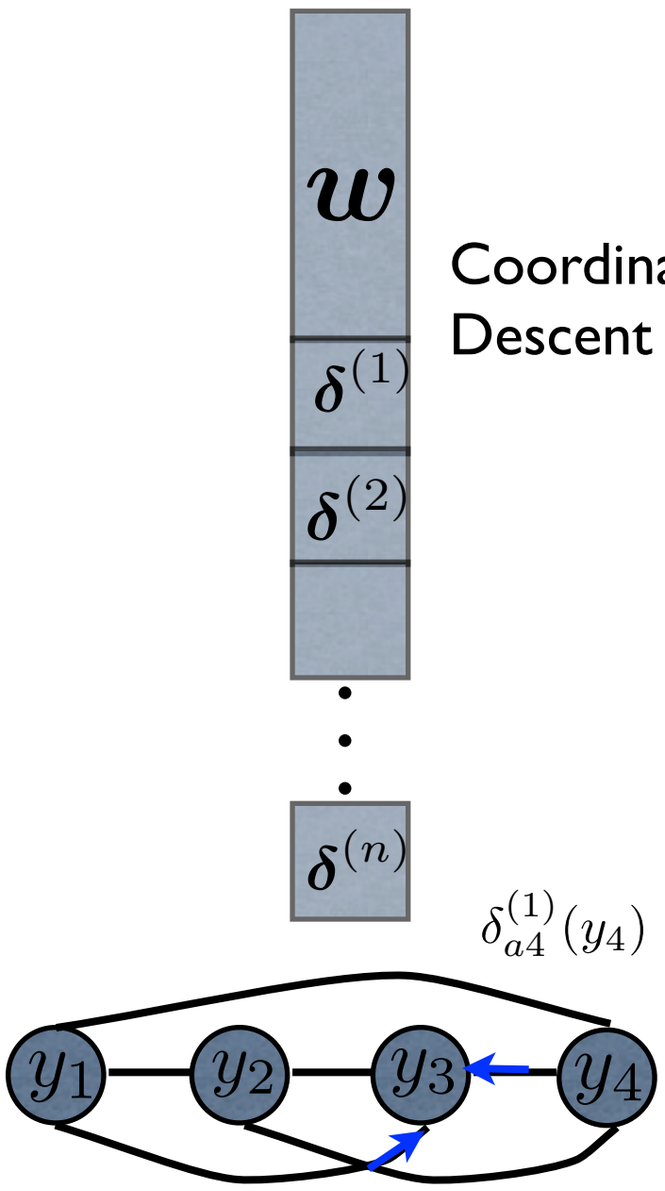
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



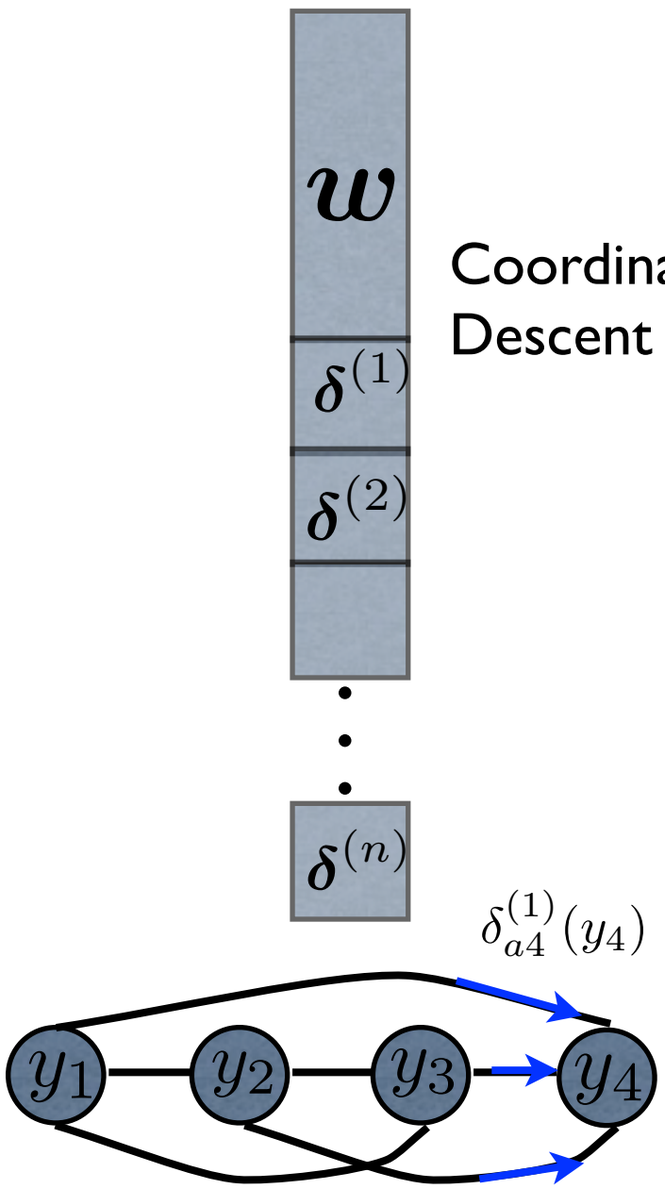
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



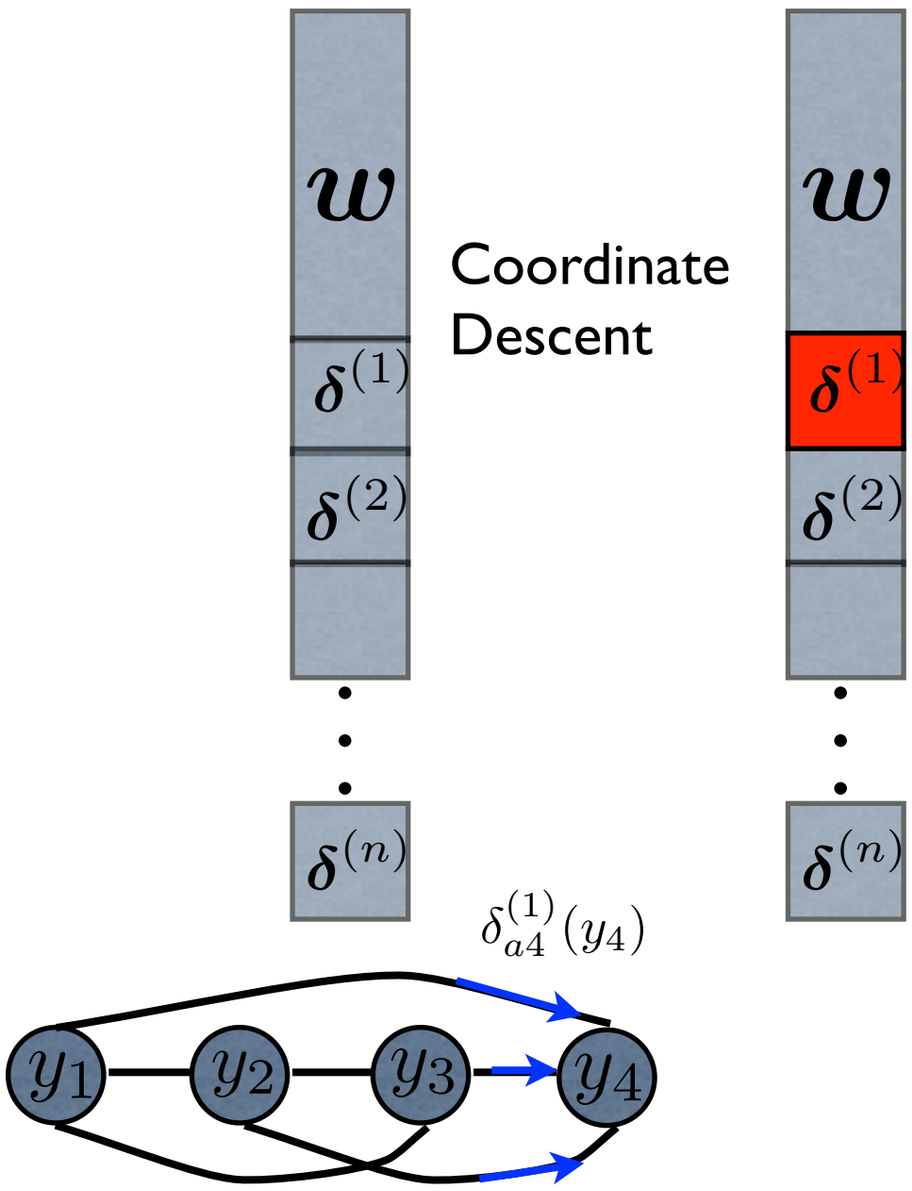
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



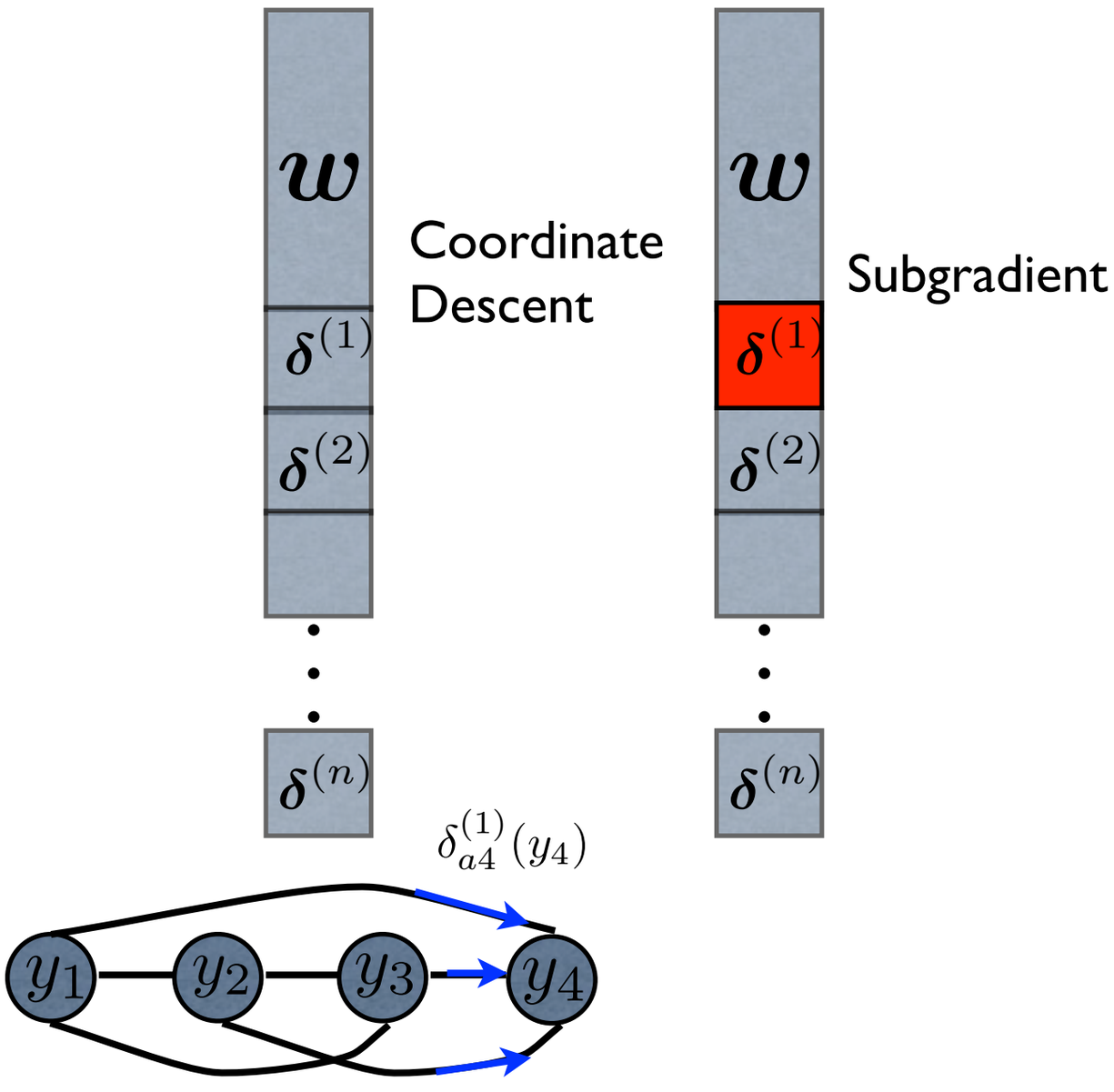
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



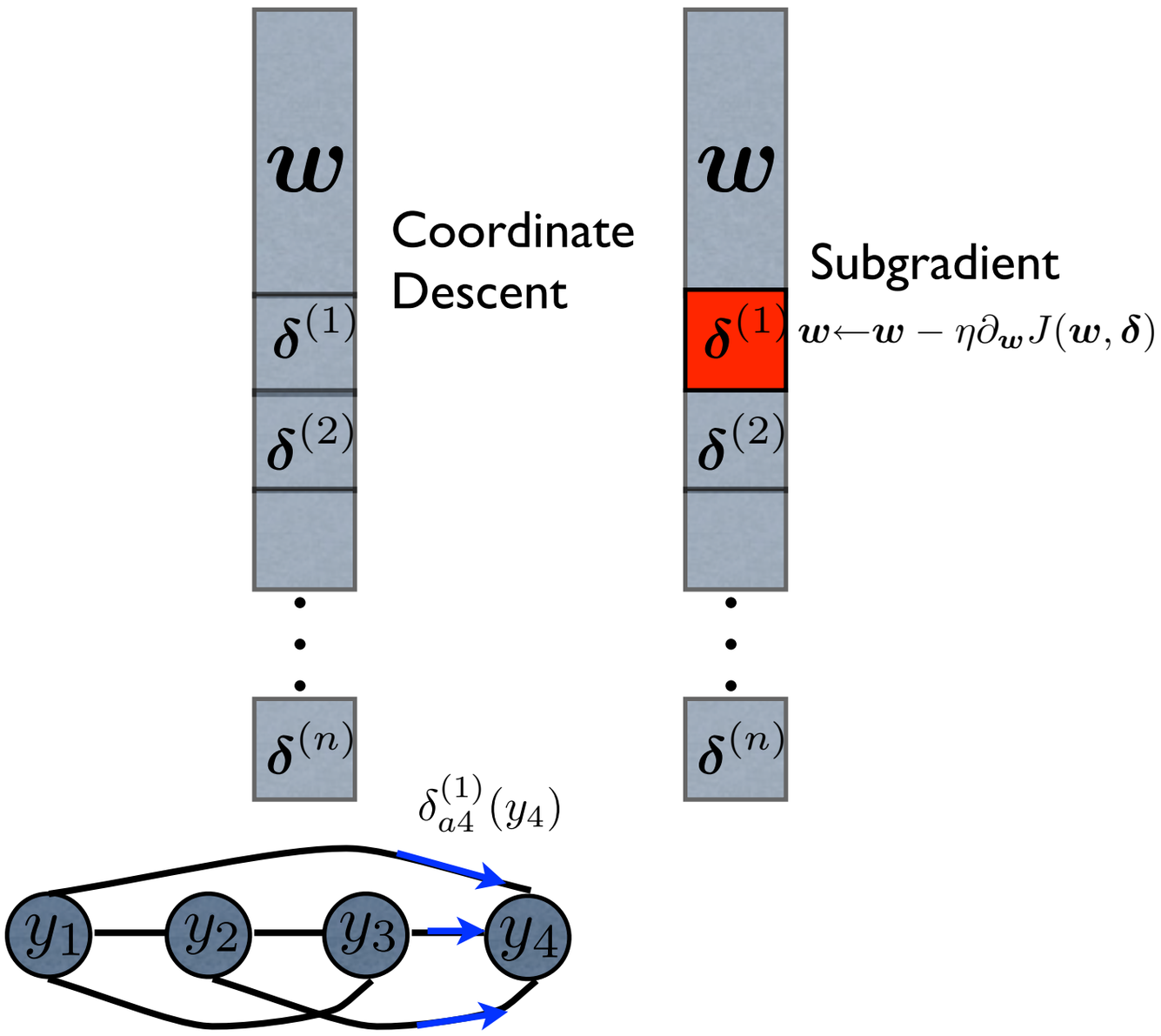
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



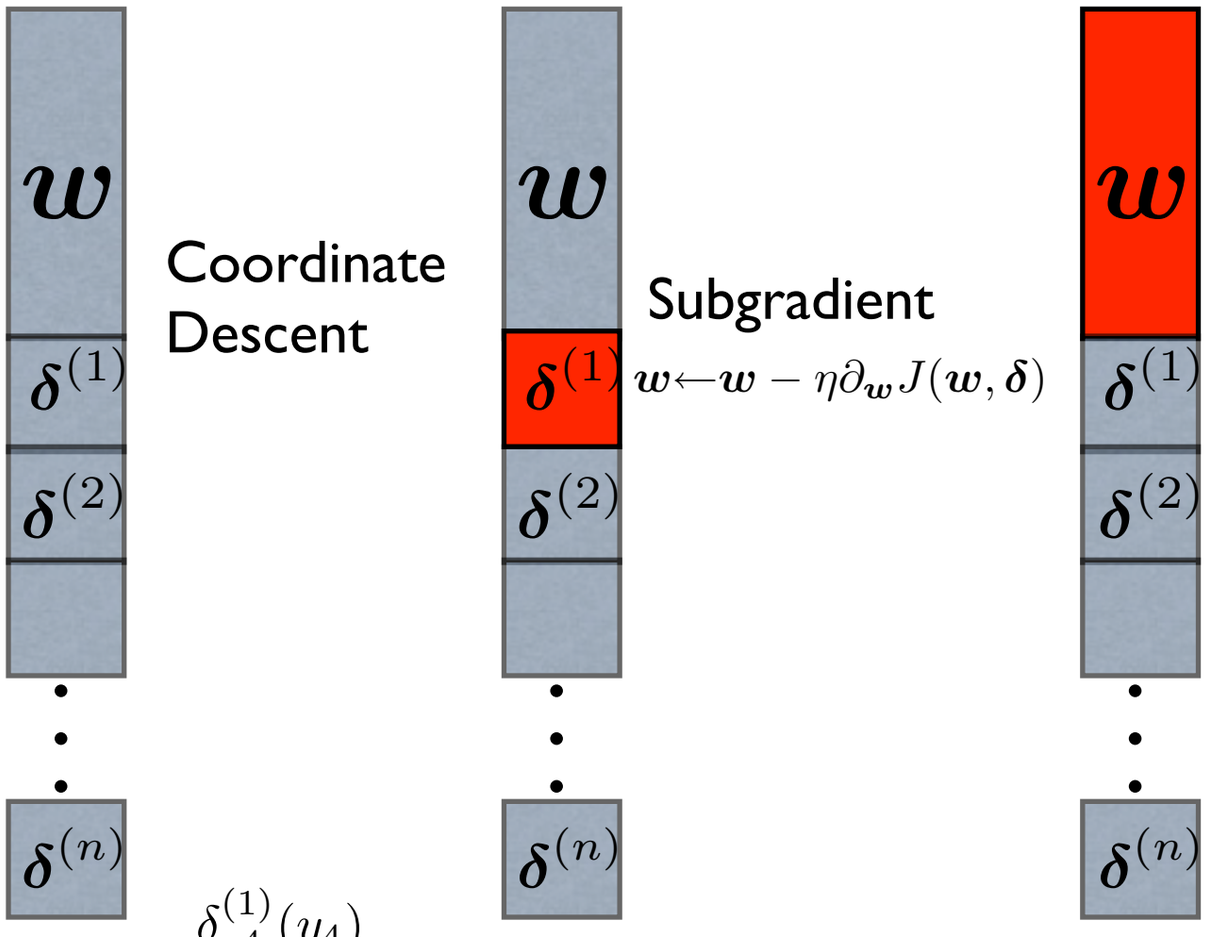
The DLPW Algorithm

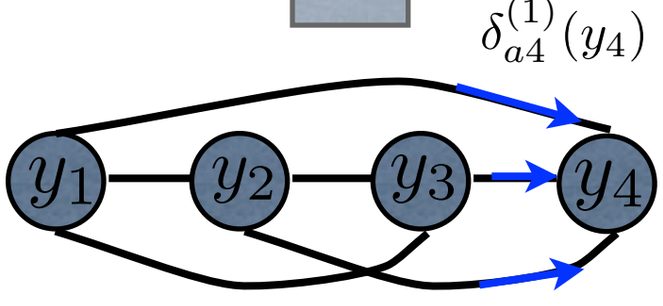
- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



The DLPW Algorithm

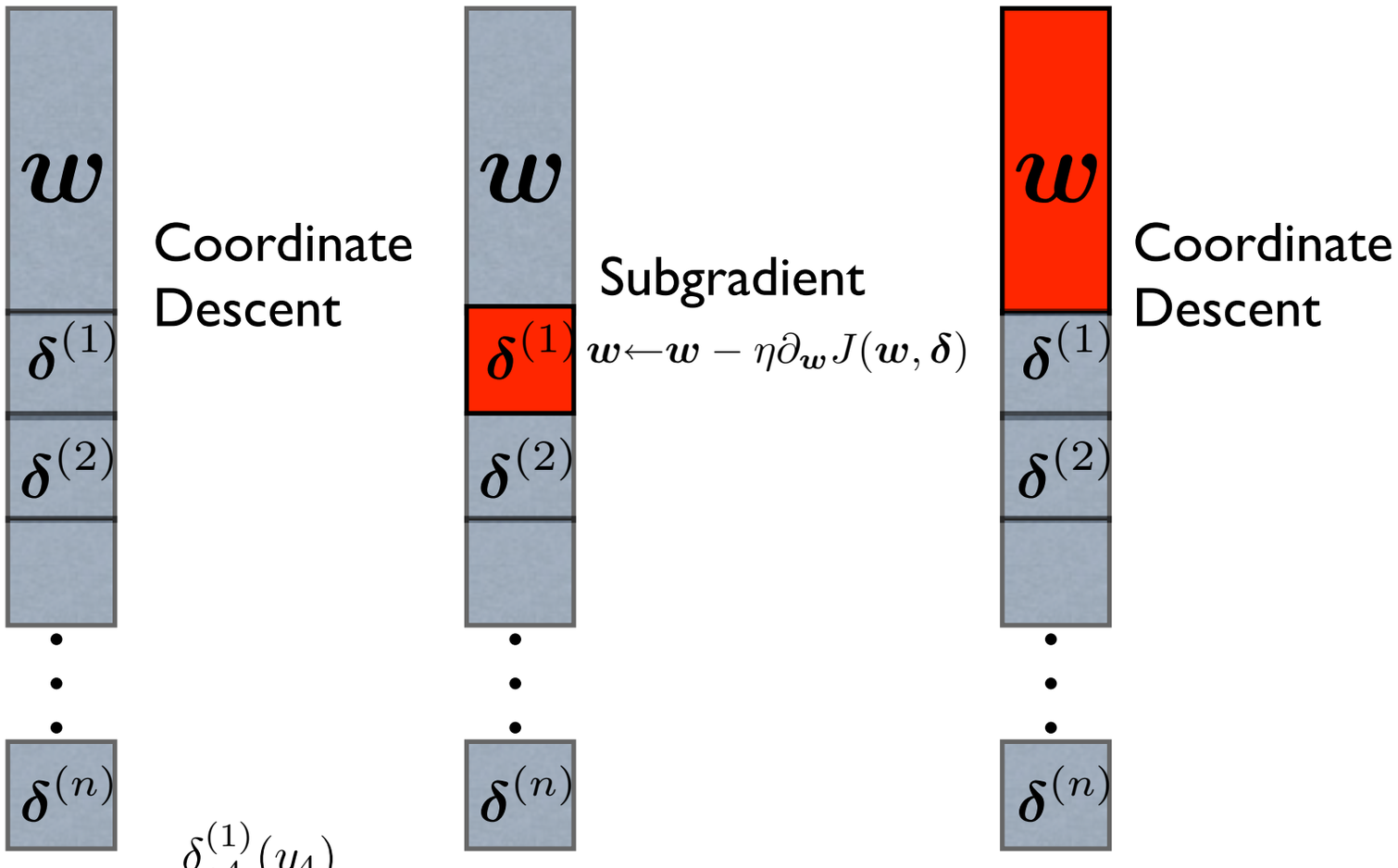
- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*

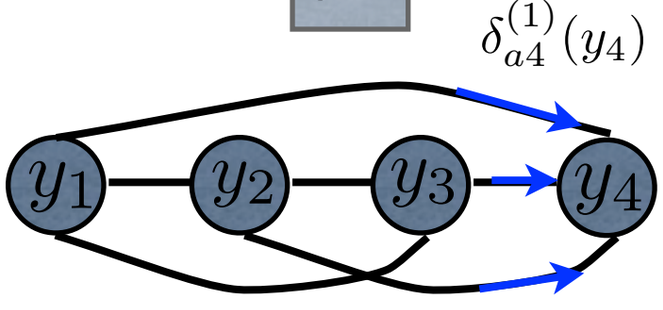




The DLPW Algorithm

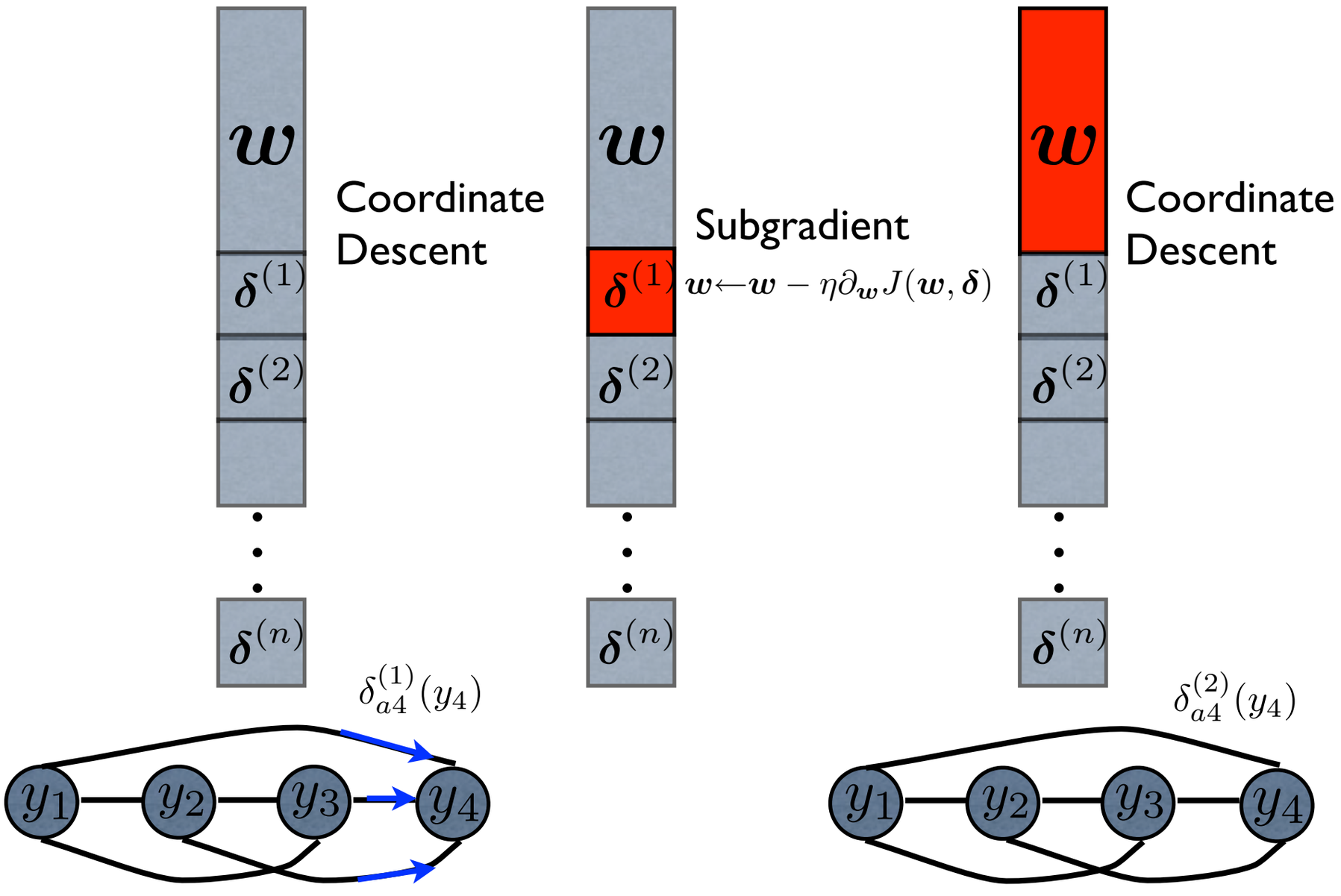
- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*





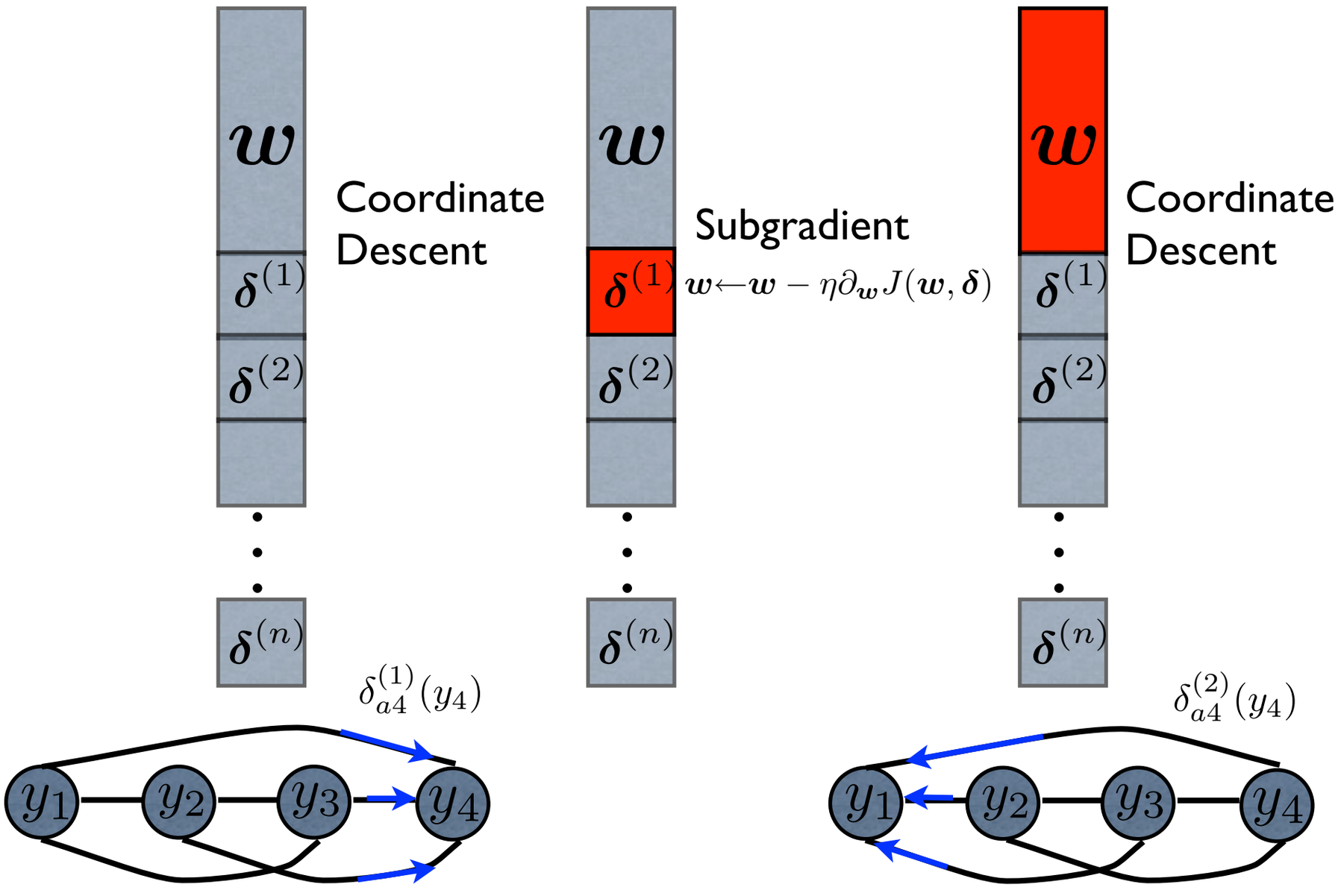
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



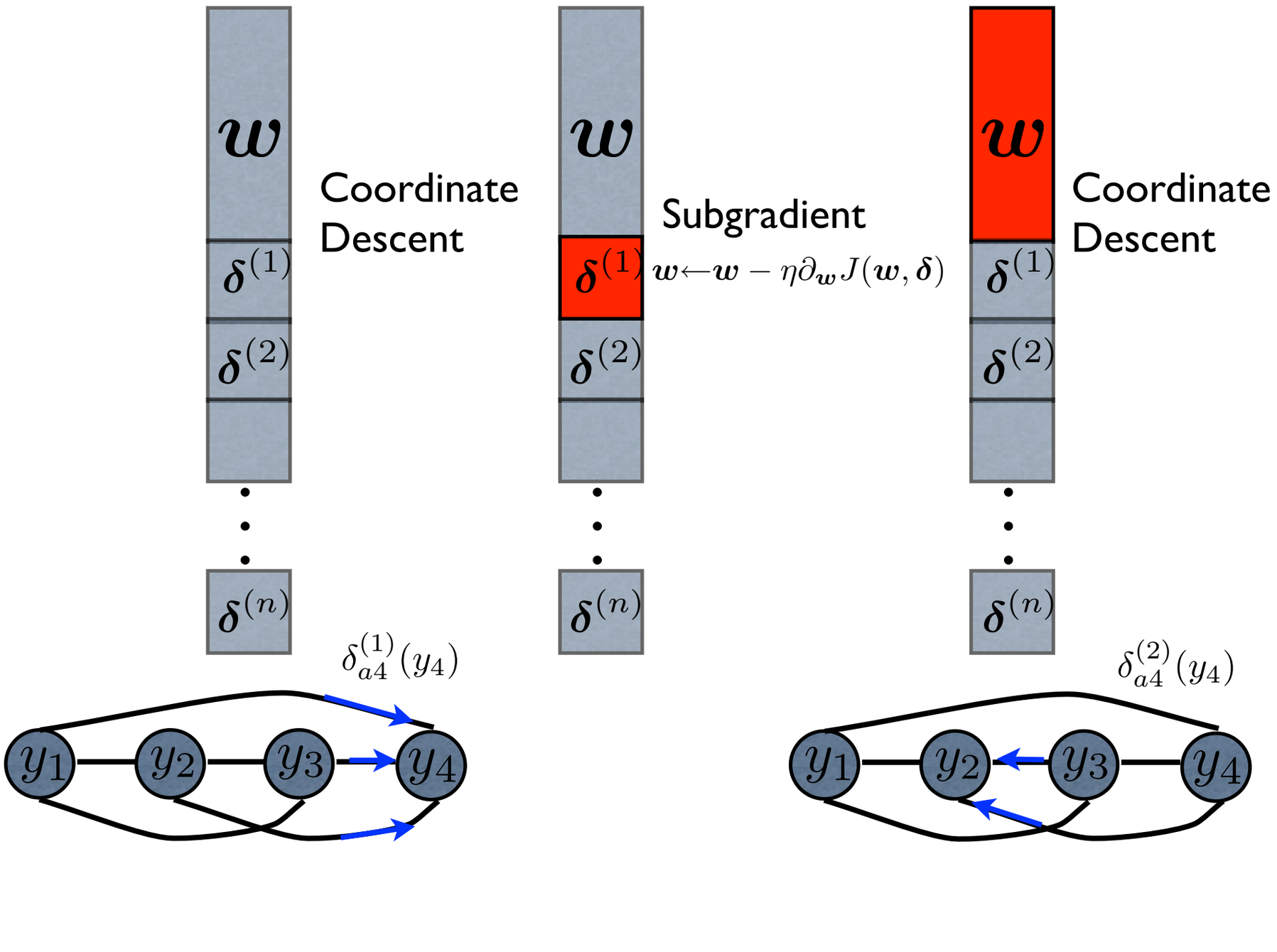
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



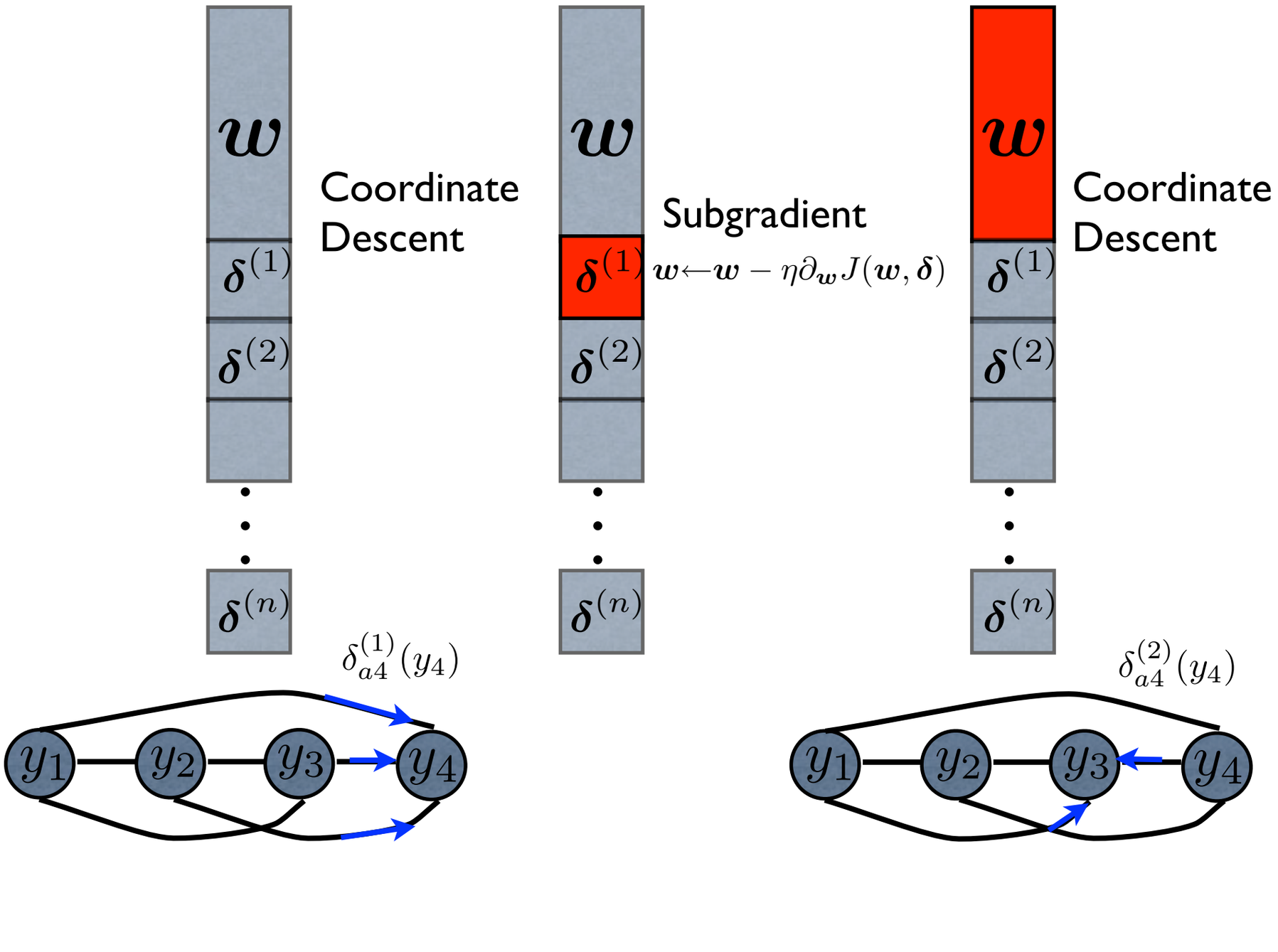
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



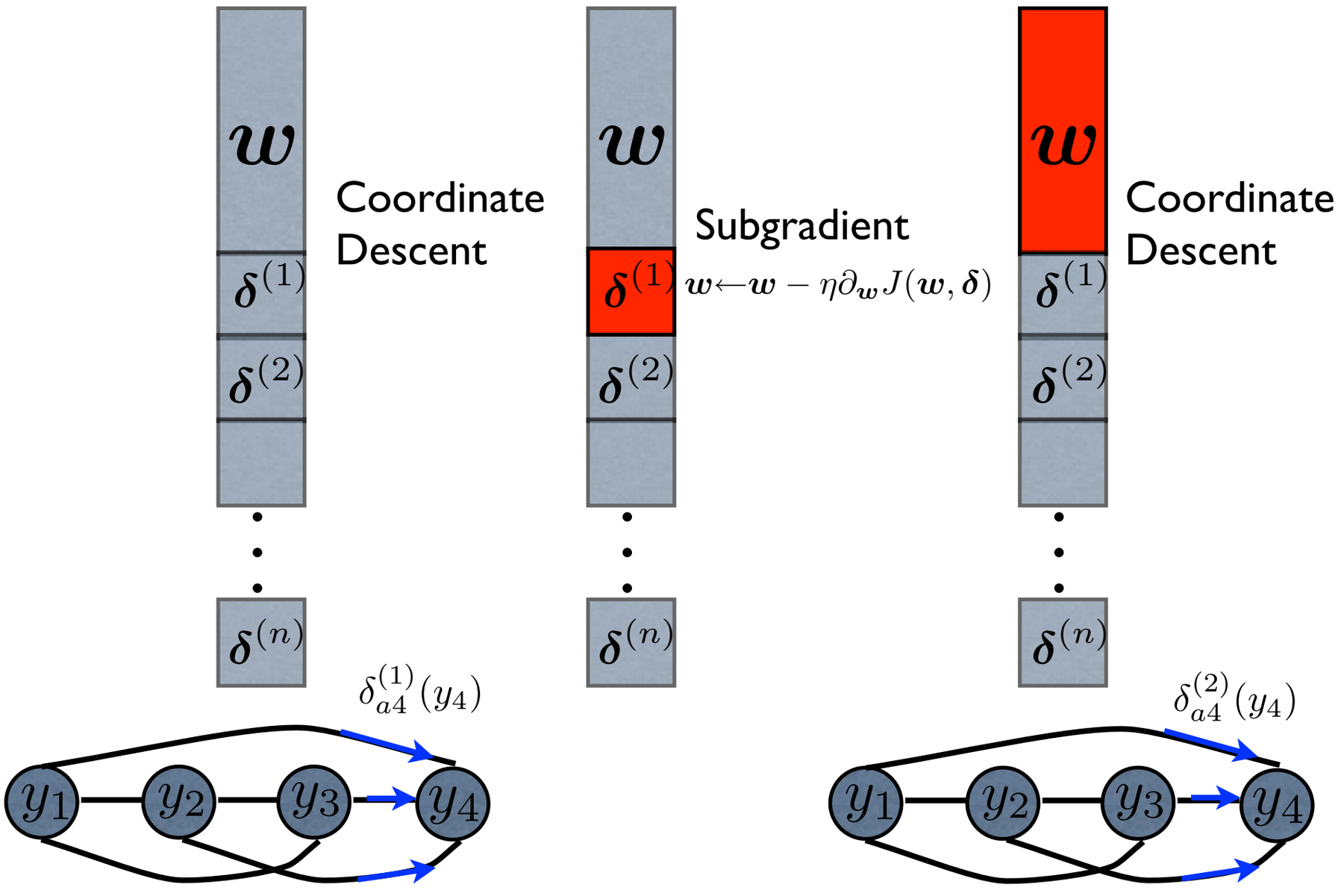
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



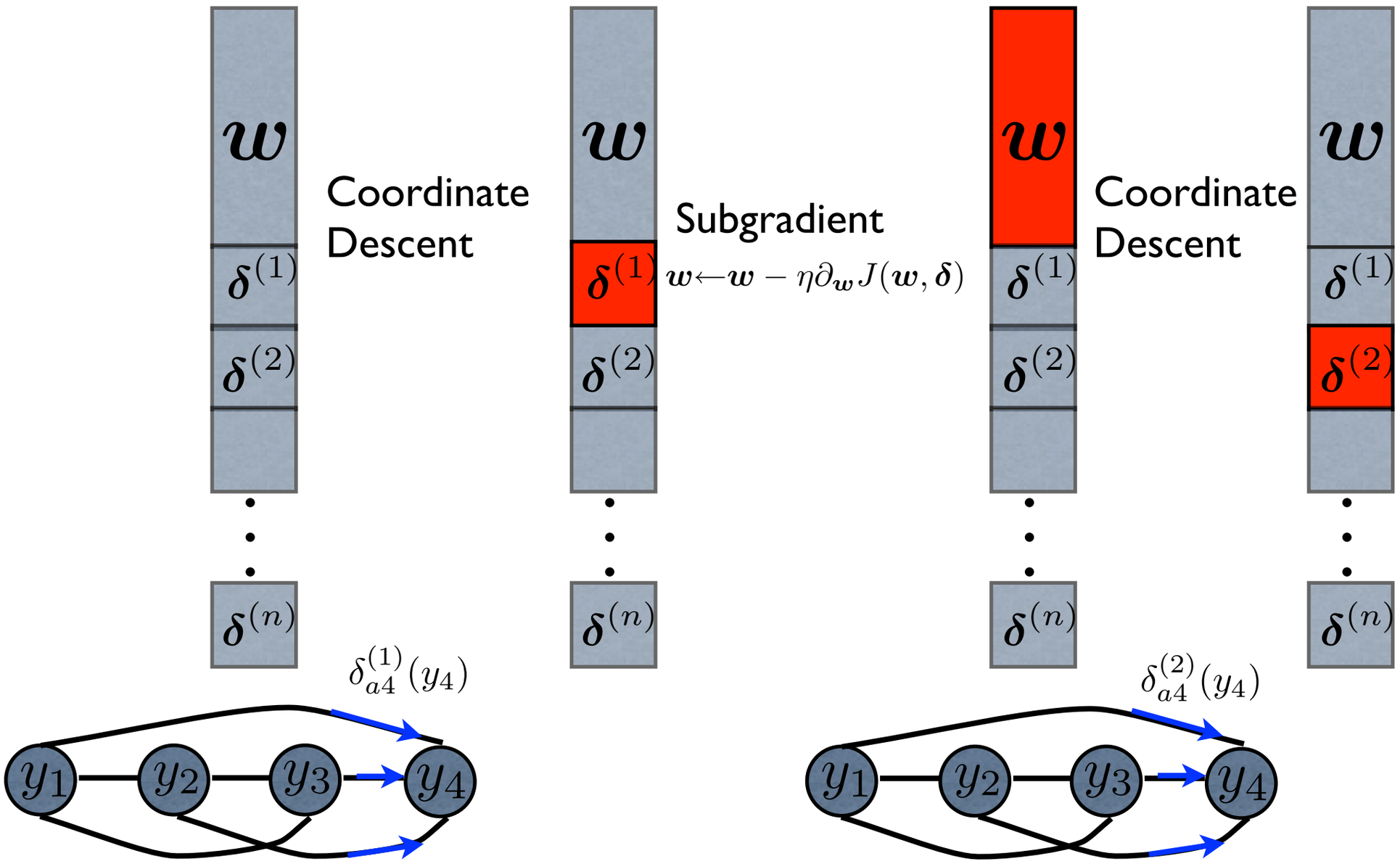
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



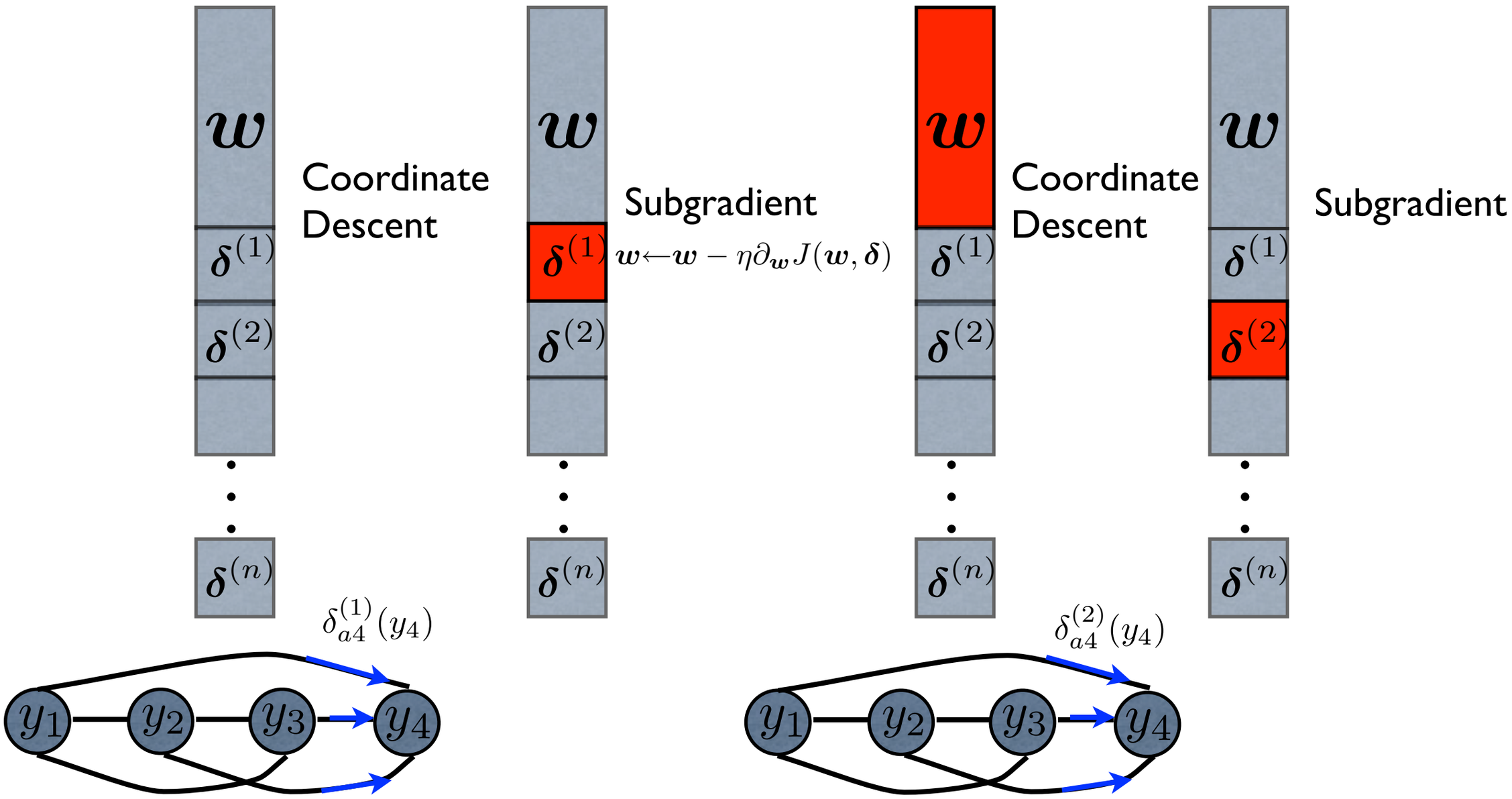
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



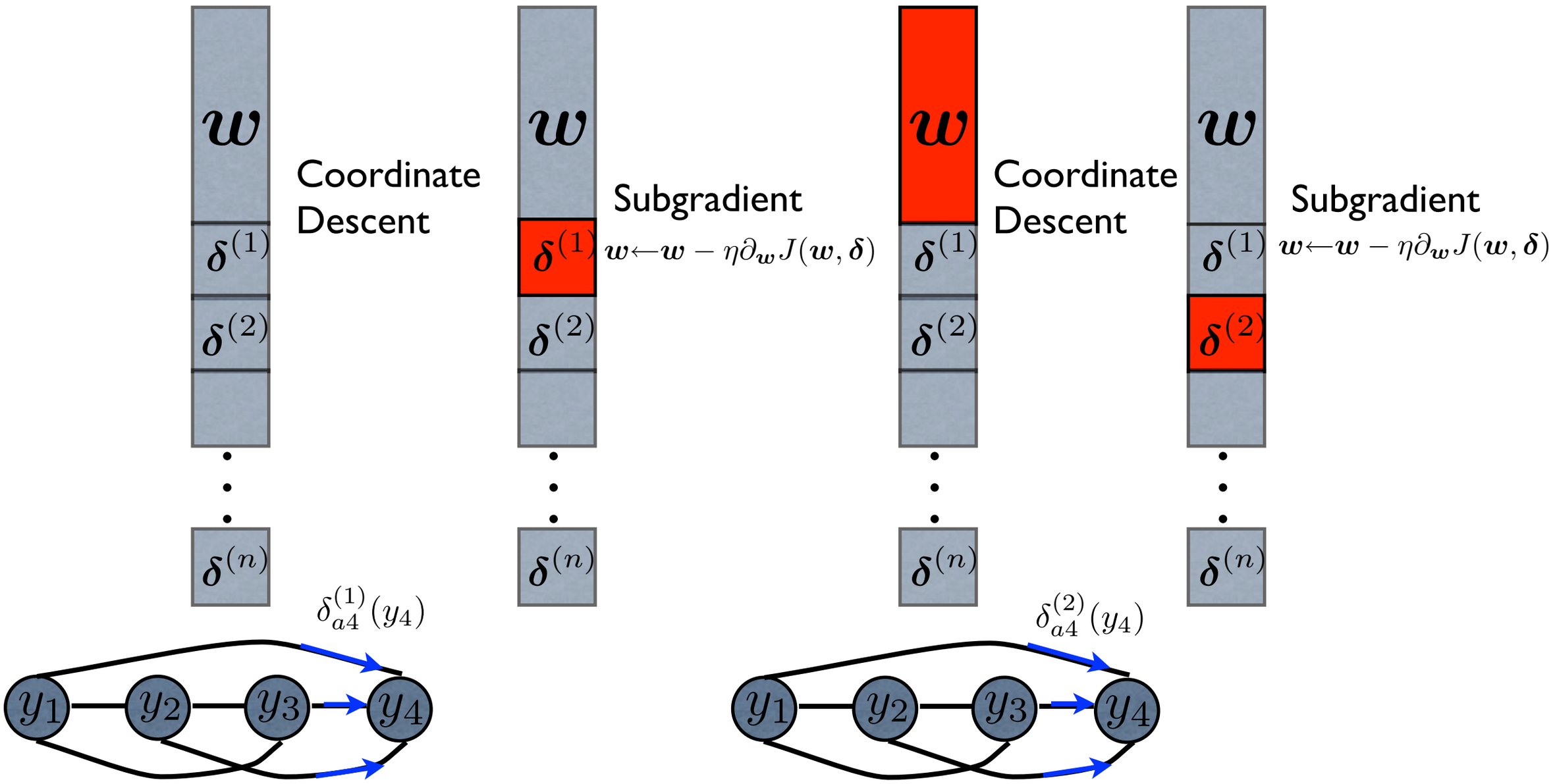
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



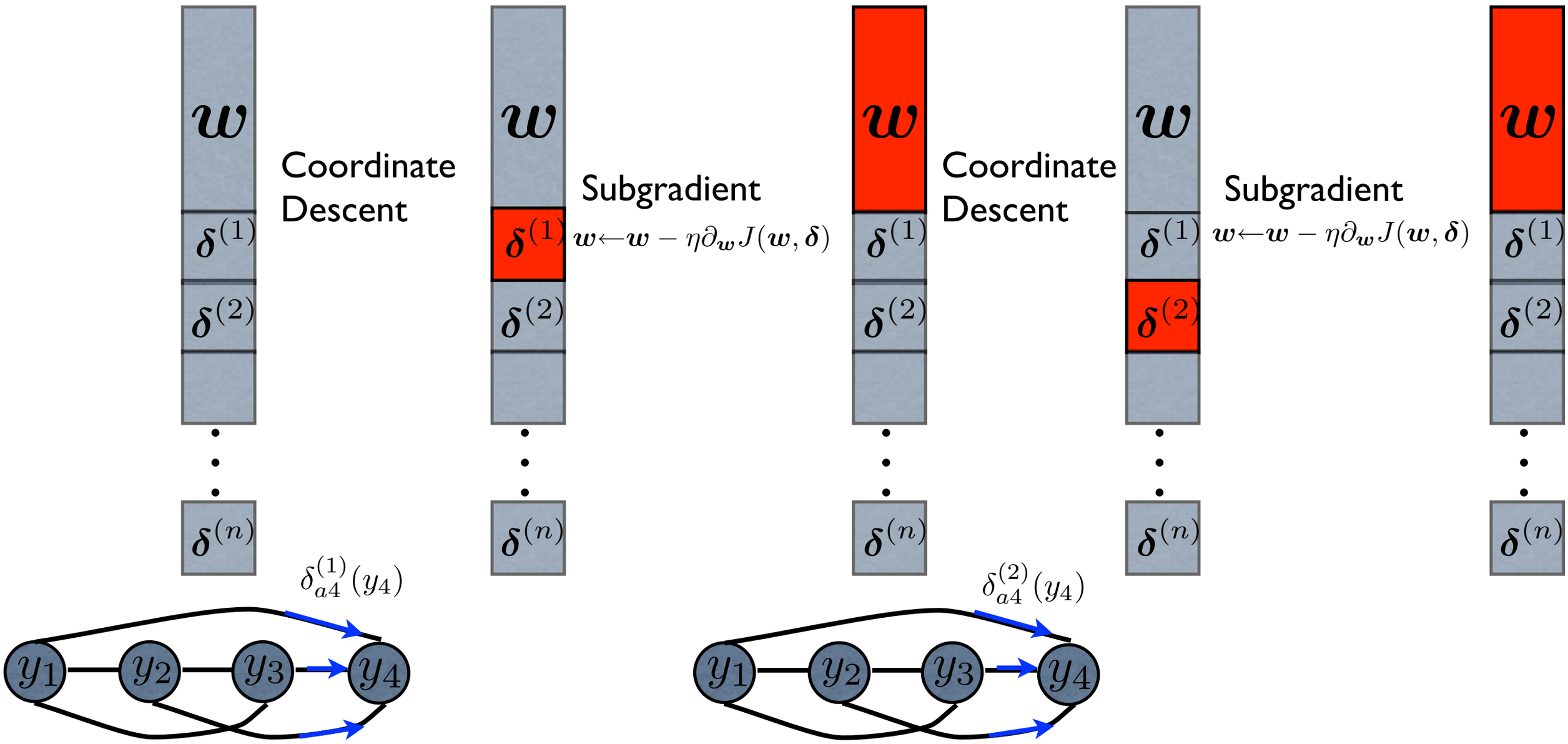
The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



The DLPW Algorithm

- **D**ual **L**oss **P**rimal **W**eights *[Meshi et al. 10]*



Comments

Comments

- Converges to the optimum with appropriate smoothing of dual.

Comments

- Converges to the optimum with appropriate smoothing of dual.
- Significantly faster than solving LP from scratch every time (efficient warm start).

Comments

- Converges to the optimum with appropriate smoothing of dual.
- Significantly faster than solving LP from scratch every time (efficient warm start).
- First application of such duality in *[Taskar 04]*.

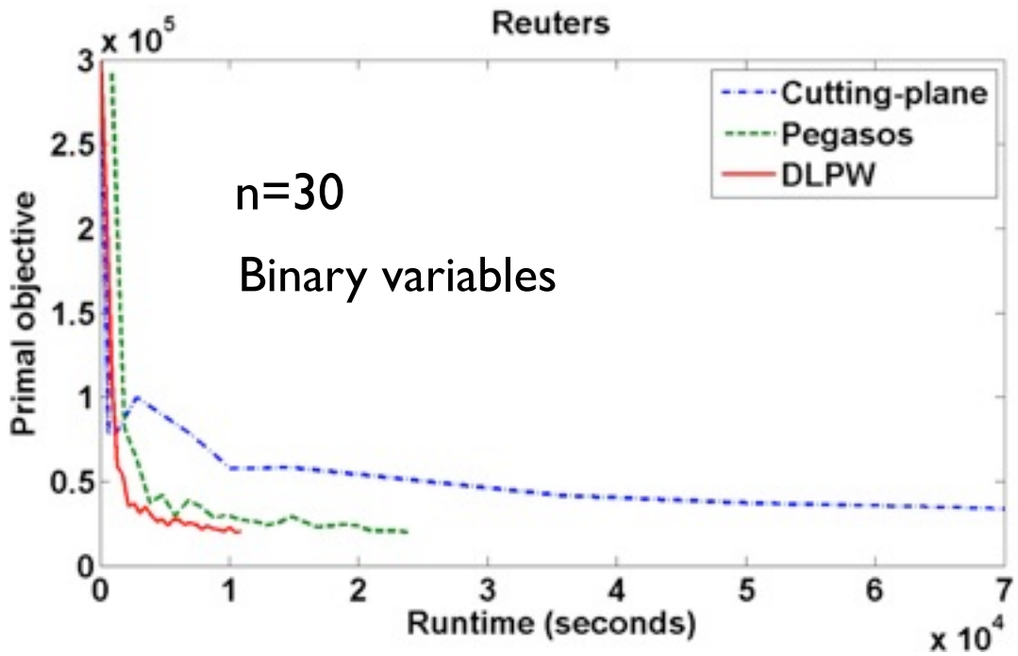
Comments

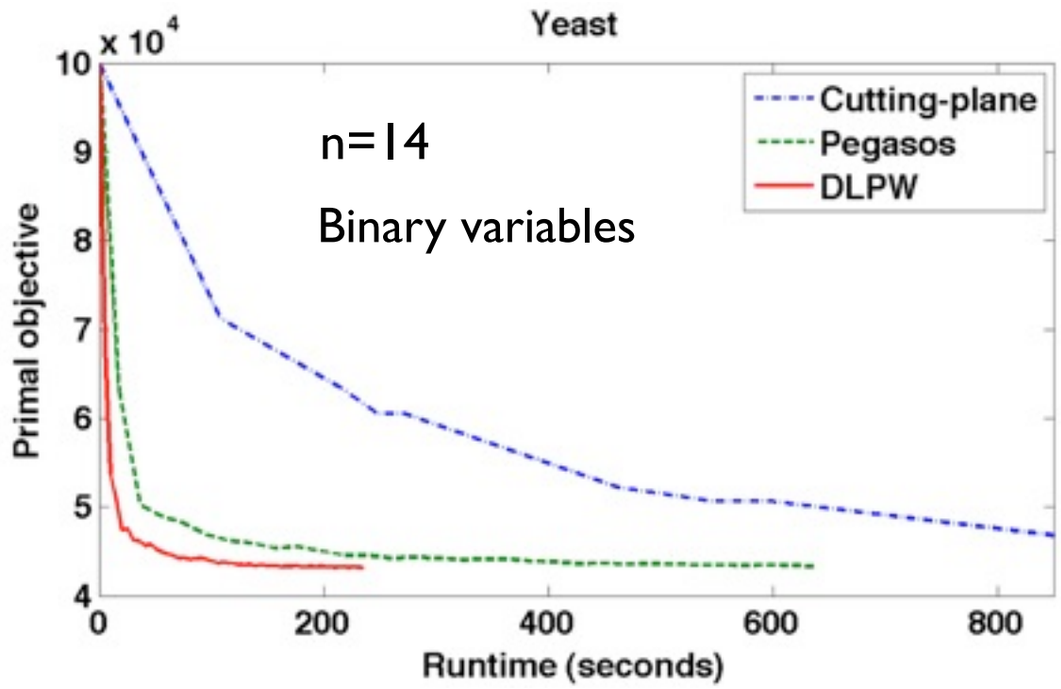
- Converges to the optimum with appropriate smoothing of dual.
- Significantly faster than solving LP from scratch every time (efficient warm start).
- First application of such duality in *[Taskar 04]*.
- Similar method proposed recently by *[Hazan et al. 10]*

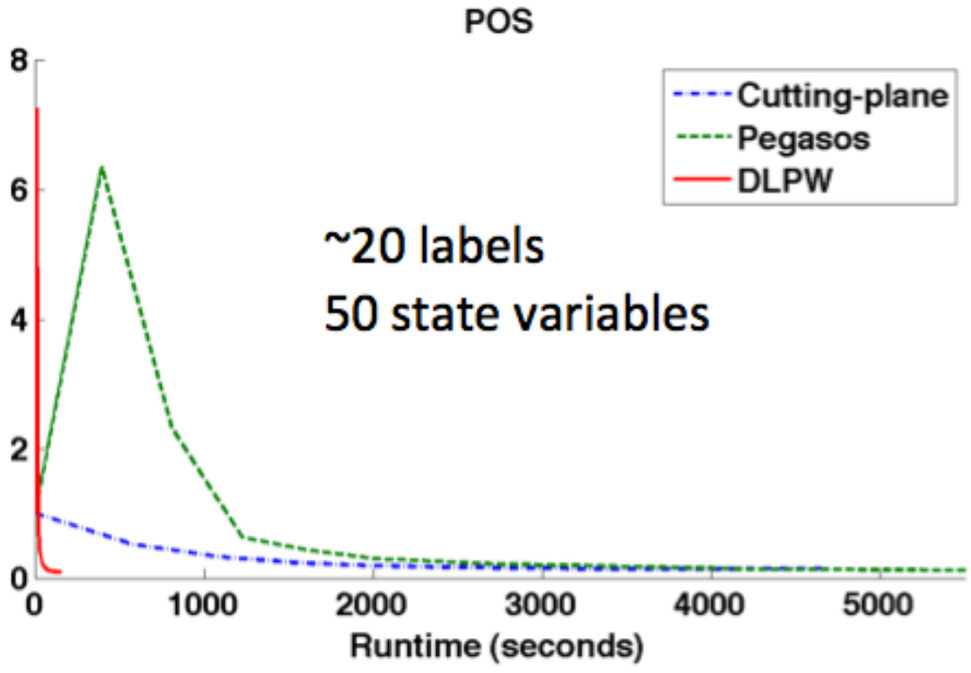
Comments

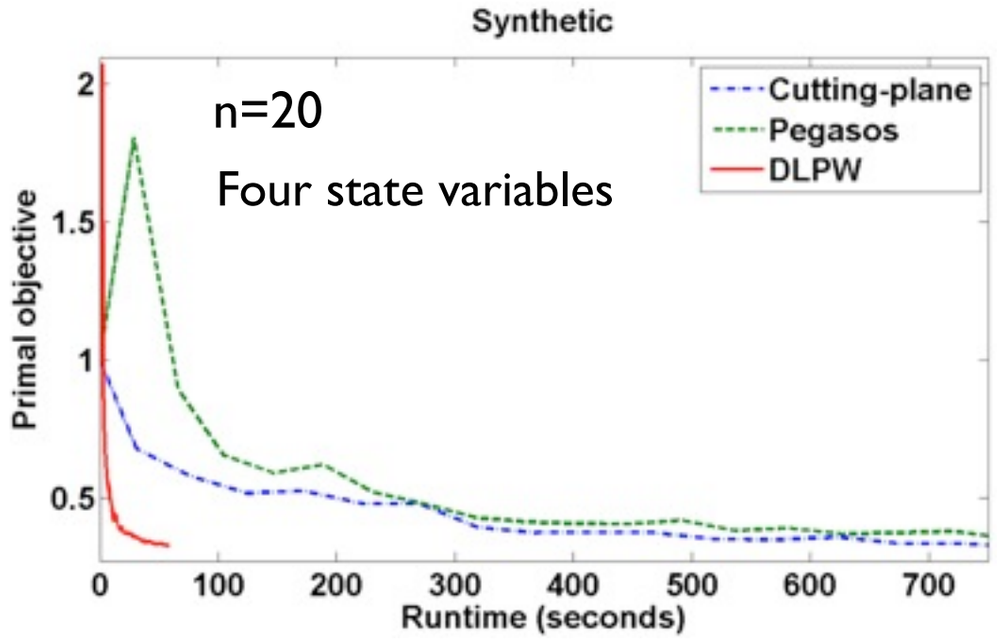
- Converges to the optimum with appropriate smoothing of dual.
- Significantly faster than solving LP from scratch every time (efficient warm start).
- First application of such duality in *[Taskar 04]*.
- Similar method proposed recently by *[Hazan et al. 10]*
- Alternative approach: gradient descent on both w and δ variables *[Komodakis 11]*.

Results


Reuters
x 10^5
Primal objective
Runtime (seconds)
x 10^4
Cutting-plane
Pegasos
DLPW
n=30
Binary variables


Yeast
x 10^4
Primal objective
Runtime (seconds)
Cutting-plane
Pegasos
DLPW
n=14
Binary variables


POS
Runtime (seconds)
Cutting-plane
Pegasos
DLPW
~20 labels
50 state variables


Synthetic
Primal objective
Runtime (seconds)
Cutting-plane
Pegasos
DLPW
n=20
Four state variables

Outline

- Solving LP relaxations:
  - Subgradient algorithms
  - Coordinate descent algorithms
- Parameter learning with LP relaxations:
  - Structured Prediction
  - Structured LP perceptron.
  - Coordinate descent algorithms.
  - Cutting plane algorithms.
  - Pseudo-max algorithms.

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w}} \frac{1}{M} \sum_{m} \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$$

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w}} \frac{1}{M} \sum_{m} \max_{\boldsymbol{\mu} \in \mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} + \frac{C}{2} \|\boldsymbol{w}\|_2^2$$

$$\begin{aligned} \min_{\boldsymbol{w},\xi} \quad & \tfrac{1}{M} \textstyle\sum_m \xi_m + \tfrac{C}{2} \|\boldsymbol{w}\|_2^2 \\ \text{s.}t. \quad & \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} \cdot \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{M}_L \end{aligned}$$

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\begin{aligned} \min_{\boldsymbol{w},\xi} \quad & \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.}t. \quad & \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu}\cdot \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{M}_L \end{aligned}$$

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\begin{aligned} \min_{\boldsymbol{w},\xi} \quad & \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.}t. \quad & \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathrm{V}ert(\mathcal{M}_L) \end{aligned}$$

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\begin{aligned} \min_{\boldsymbol{w},\xi} \quad & \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.}t. \quad & \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathrm{V}ert(\mathcal{M}_L) \end{aligned}$$

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w},\xi} \quad \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.}t. \quad \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathrm{V}ert(\mathcal{M}_L)$$

Many Vertices!

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w},\xi} \quad \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.t.} \quad \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{V}ert(\mathcal{M}_L)$$

Many Vertices!

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\begin{aligned} \min_{\boldsymbol{w},\xi} \quad & \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.}t. \quad & \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{V}ert(\mathcal{M}_L) \end{aligned}$$

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w},\xi} \quad \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.}t. \quad \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{V}ert(\mathcal{M}_L) \quad \boldsymbol{\mu} \in S_m$$

$$S_m \subseteq \mathcal{M}_L$$

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w},\xi} \quad \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.}t. \quad \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{V}ert(\mathcal{M}_L) \quad \boldsymbol{\mu} \in S_m$$

$$S_m \subseteq \mathcal{M}_L$$

- Initialize S_m

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w},\xi} \quad \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.}t. \quad \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{V}ert(\mathcal{M}_L) \quad \boldsymbol{\mu} \in S_m$$

$$S_m \subseteq \mathcal{M}_L$$

- Initialize S_m
- Repeat:

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w},\xi} \quad \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.}t. \quad \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{V}ert(\mathcal{M}_L) \quad \boldsymbol{\mu} \in S_m$$

$$S_m \subseteq \mathcal{M}_L$$

- Initialize S_m
- Repeat:
  - Solve for $\boldsymbol{w}, \xi$ given current S_m

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w},\xi} \quad \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.t.} \quad \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{V}ert(\mathcal{M}_L) \quad \boldsymbol{\mu} \in S_m$$

$$S_m \subseteq \mathcal{M}_L$$

- Initialize S_m
- Repeat:
  - Solve for $\boldsymbol{w}, \xi$ given current S_m
  - If exists an m s.t.:

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w},\xi} \quad \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.}t. \quad \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{V}ert(\mathcal{M}_L) \quad \boldsymbol{\mu} \in S_m$$

$$S_m \subseteq \mathcal{M}_L$$

- Initialize S_m
- Repeat:
  - Solve for $\boldsymbol{w}, \xi$ given current S_m
  - If exists an m s.t.: $\max_{\boldsymbol{\mu}\in\mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu} > \max_{\boldsymbol{\mu}\in S_m} \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu}$

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w},\xi} \quad \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.}t. \quad \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{V}ert(\mathcal{M}_L) \quad \boldsymbol{\mu} \in S_m$$

$$S_m \subseteq \mathcal{M}_L$$

- Initialize S_m
- Repeat:
  - Solve for $\boldsymbol{w}, \xi$ given current S_m
  - If exists an m s.t.: $\max_{\boldsymbol{\mu}\in\mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu} > \max_{\boldsymbol{\mu}\in S_m} \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu}$
    - do: $S_m = S_m \cup \arg\max_{\boldsymbol{\mu}\in\mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w})\cdot\boldsymbol{\mu}$

Cutting Plane

- Proposed in *[Finley and Joachims 08]*.

$$\min_{\boldsymbol{w},\xi} \quad \frac{1}{M}\sum_m \xi_m + \frac{C}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.t.} \quad \xi_m \geq \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} \quad , \quad \forall \boldsymbol{\mu} \in \mathcal{V}ert(\mathcal{M}_L) \quad \boldsymbol{\mu} \in S_m$$

$$S_m \subseteq \mathcal{M}_L$$

- Initialize S_m
- Repeat:
  - Solve for $\boldsymbol{w}, \xi$ given current S_m
  - If exists an m s.t.: $\max_{\boldsymbol{\mu}\in\mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu} > \max_{\boldsymbol{\mu}\in S_m} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu}$
    - do: $S_m = S_m \cup \arg\max_{\boldsymbol{\mu}\in\mathcal{M}_L} \boldsymbol{\theta}^m(\boldsymbol{w}) \cdot \boldsymbol{\mu}$
  - else *done*

Cutting Plane

Cutting Plane

- Nice property: Requires only $O(\frac{1}{\epsilon^2})$ added constraints to get ϵ accuracy.

Cutting Plane

- Nice property: Requires only $O(\frac{1}{\epsilon^2})$ added constraints to get ϵ accuracy.
- Requires repeatedly solving a QP over the current constraints, but can be warm-started from previous iterations.

Cutting Plane

- Nice property: Requires only $O(\frac{1}{\epsilon^2})$ added constraints to get ϵ accuracy.
- Requires repeatedly solving a QP over the current constraints, but can be warm-started from previous iterations.
- Especially useful when maximum over local polytope is fast (e.g., binary variables using the QPBO algorithm, *[Kolmogorov and Rother 07, Hammer et al 84]*).

Outline

- Solving LP relaxations:
  - Subgradient algorithms
  - Coordinate descent algorithms
- Parameter learning with LP relaxations:
  - Structured Prediction
  - Structured LP perceptron.
  - Coordinate descent algorithms.
  - Cutting plane algorithms.
  - Pseudo-max algorithms.

Using Less Constraints

- The cutting plane method iteratively adds vertices of the local polytope as constraints.

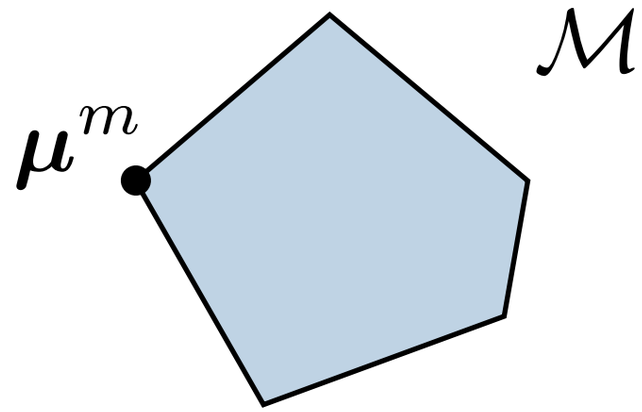
Using Less Constraints

- The cutting plane method iteratively adds vertices of the local polytope as constraints.
- Can we work with a small subset of the marginal polytope vertices?

Using Less Constraints

- The cutting plane method iteratively adds vertices of the local polytope as constraints.

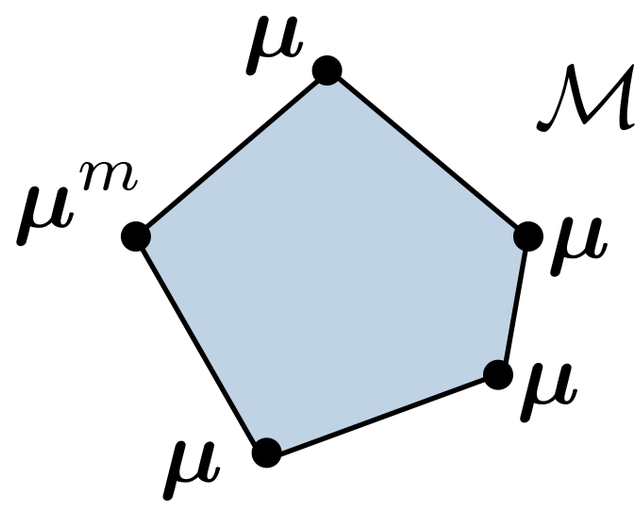
- Can we work with a small subset of the marginal polytope vertices?



Using Less Constraints

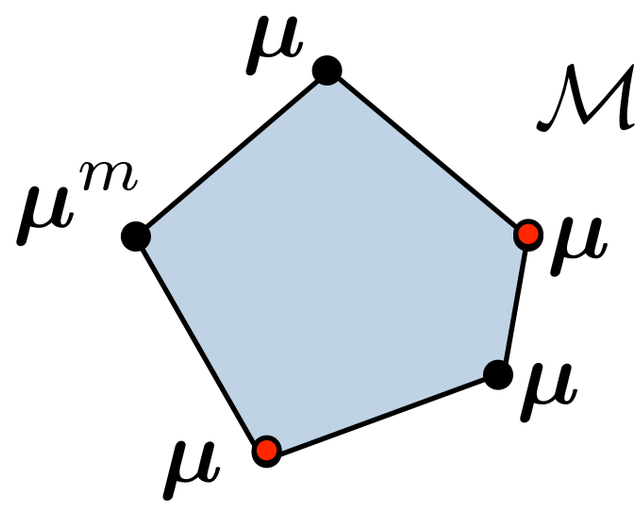
- The cutting plane method iteratively adds vertices of the local polytope as constraints.

- Can we work with a small subset of the marginal polytope vertices?



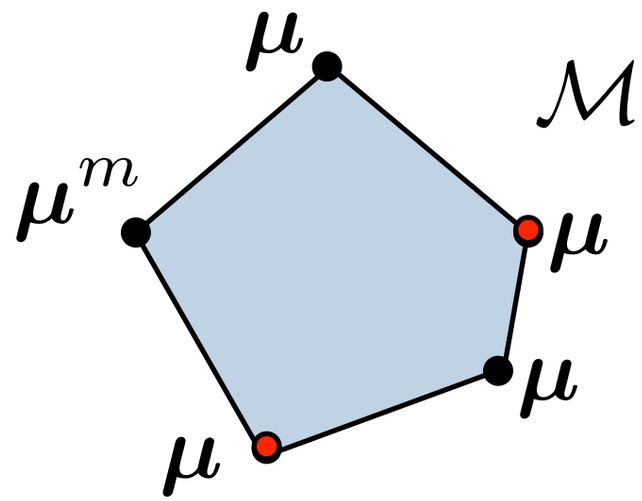
Using Less Constraints

- The cutting plane method iteratively adds vertices of the local polytope as constraints.
- Can we work with a small subset of the marginal polytope vertices?



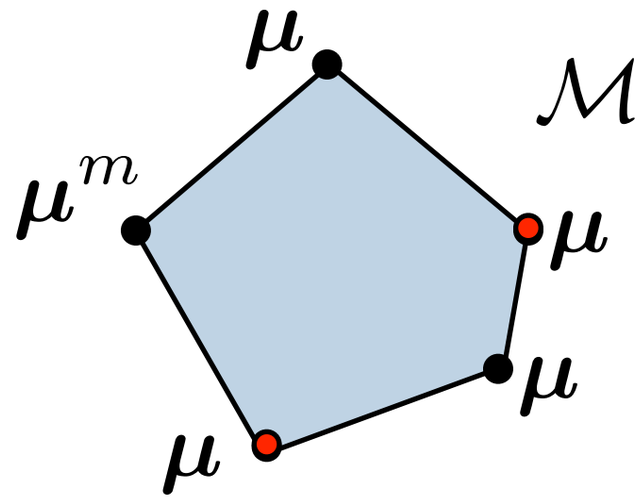
Using Less Constraints

- The cutting plane method iteratively adds vertices of the local polytope as constraints.
- Can we work with a small subset of the marginal polytope vertices?
- Clearly generally not optimal.



Using Less Constraints

- The cutting plane method iteratively adds vertices of the local polytope as constraints.



- Can we work with a small subset of the marginal polytope vertices?

- Clearly generally not optimal.

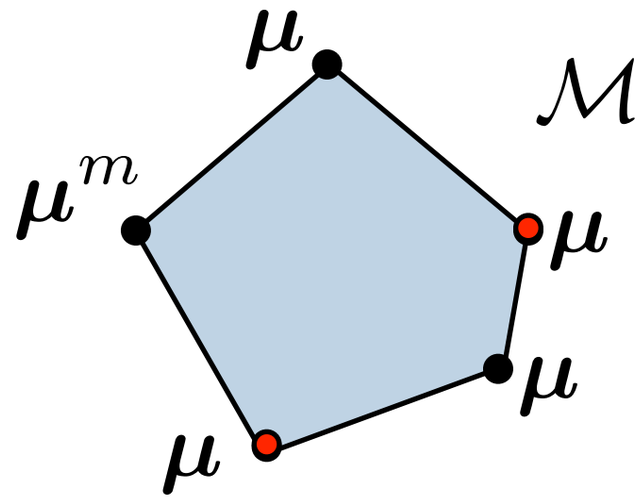
- What's a good way of choosing these vertices, and what can we guarantee about this?

Using Less Constraints

- The cutting plane method iteratively adds vertices of the local polytope as constraints.

- Can we work with a small subset of the marginal polytope vertices?

- Clearly generally not optimal.



- What's a good way of choosing these vertices, and what can we guarantee about this?
- In *[Sontag et al. 10]* we propose a simple 'pseudo-max' method and analyze it. Related works in *[Alahari et al. 10, Vickrey et al. 10]*.

The Pseudo-Max Approach

The Pseudo-Max Approach

- Learning is NP hard when inference is hard

The Pseudo-Max Approach

- Learning is NP hard when inference is hard
  - Hard to find a **w** that separates the data in the general case.

The Pseudo-Max Approach

- Learning is NP hard when inference is hard
  - Hard to find a **w** that separates the data in the general case.
- True for any finite dataset.

The Pseudo-Max Approach

- Learning is NP hard when inference is hard
  - Hard to find a **w** that separates the data in the general case.
- True for any finite dataset.
- What if we had unlimited data?

The Pseudo-Max Approach

- Learning is NP hard when inference is hard
  - Hard to find a **w** that separates the data in the general case.
- True for any finite dataset.
- What if we had unlimited data?
- For maximum likelihood in graphical models, pseudo-likelihood *[Besag 75]* can find *true* parameters even when inference is intractable.

The Pseudo-Max Approach

- Learning is NP hard when inference is hard
  - Hard to find a **w** that separates the data in the general case.
- True for any finite dataset.
- What if we had unlimited data?
- For maximum likelihood in graphical models, pseudo-likelihood *[Besag 75]* can find *true* parameters even when inference is intractable.
- Can we do it for structured prediction?

The Pseudo-Max Approach

- Learning is NP hard when inference is hard
  - Hard to find a **w** that separates the data in the general case.
- True for any finite dataset.
- What if we had unlimited data?
- For maximum likelihood in graphical models, pseudo-likelihood *[Besag 75]* can find *true* parameters even when inference is intractable.
- Can we do it for structured prediction?
- In a sense, we can, using the pseudo-max approach *[Sontag et al. 10]*.

The Pseudo-Max Approach

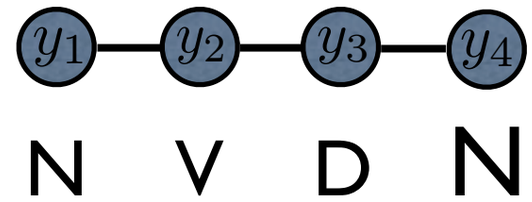
- Learning is NP hard when inference is hard
  - Hard to find a **w** that separates the data in the general case.
- True for any finite dataset.
- What if we had unlimited data?
- For maximum likelihood in graphical models, pseudo-likelihood *[Besag 75]* can find *true* parameters even when inference is intractable.
- Can we do it for structured prediction?
- In a sense, we can, using the pseudo-max approach *[Sontag et al. 10]*.
- Related approaches: *[Alahari et al. 10, Vickrey et al. 10]*

Predicting Single Variables

- Given an example:

x : John hit the ball

$\mathbf{y}$:

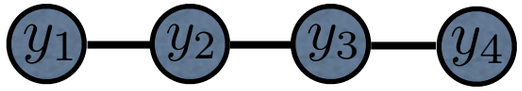


Predicting Single Variables

- Given an example:

x : John hit the ball

$\mathbf{y}$:



N V D N

- Assume data is separable with some $\mathbf{w}^*$:

Predicting Single Variables

- Given an example:

x : John hit the ball

$\mathbf{y}$: y_1 — y_2 — y_3 — y_4

N V D N

- Assume data is separable with some **w***:

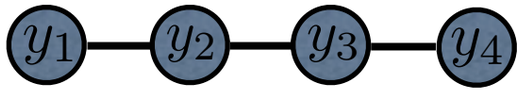
$$y = \arg\max_{\hat{y}} \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\hat{y}, x)$$

Predicting Single Variables

- Given an example:

x : John hit the ball

$\mathbf{y}$:



N V D N

- Assume data is separable with some $\mathbf{w}^*$:

$$y = \arg\max_{\hat{y}} \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\hat{y}, x)$$

- What if we "hide" one variable?

Predicting Single Variables

- Given an example:

x : John hit the ball

$\mathbf{y}$: y_1 — y_2 — y_3 — y_4

N V D N

- Assume data is separable with some **w***:

$$y = \arg\max_{\hat{y}} \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\hat{y}, x)$$

- What if we "hide" one variable?

$$? = \arg\max_{y_4} \boldsymbol{w}^* \cdot \boldsymbol{\phi}([y_1 = N, y_2 = V, y_3 = D, y_4], x)$$

Predicting Single Variables

- Given an example:

x : John hit the ball

$\mathbf{y}$: y_1 — y_2 — y_3 — y_4

N V D ?

- Assume data is separable with some **w***:

$$y = \arg\max_{\hat{y}} \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\hat{y}, x)$$

- What if we "hide" one variable?

$$? = \arg\max_{y_4} \boldsymbol{w}^* \cdot \boldsymbol{\phi}([y_1 = N, y_2 = V, y_3 = D, y_4], x)$$

Predicting Single Variables

- Given an example:

x : John hit the ball

$\mathbf{y}$: y_1 — y_2 — y_3 — y_4

N V D ?

- Assume data is separable with some **w***:

$$y = \arg\max_{\hat{y}} \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\hat{y}, x)$$

- What if we "hide" one variable?

$$? = \arg\max_{y_4} \boldsymbol{w}^* \cdot \boldsymbol{\phi}([y_1 = N, y_2 = V, y_3 = D, y_4], x)$$

- Each variable should be correctly predicted given the others .

Predicting Single Variables

- Given an example:

x : John hit the ball

$\mathbf{y}$: y_1 — y_2 — y_3 — y_4

N V D ?

- Assume data is separable with some **w***:

$$y = \arg\max_{\hat{y}} \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\hat{y}, x)$$

- What if we "hide" one variable?

$$= \arg\max_{y_4} \boldsymbol{w}^* \cdot \boldsymbol{\phi}([y_1 = N, y_2 = V, y_3 = D, y_4], x)$$

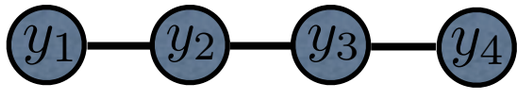
- Each variable should be correctly predicted given the others .

Predicting Single Variables

- Given an example:

x : John hit the ball

$\mathbf{y}$:



N V D N

- Assume data is separable with some **w***:

$$y = \arg\max_{\hat{y}} \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\hat{y}, x)$$

- What if we "hide" one variable?

$$\mathsf{N} = \arg\max_{y_4} \boldsymbol{w}^* \cdot \boldsymbol{\phi}([y_1 = N, y_2 = V, y_3 = D, y_4], x)$$

- Each variable should be correctly predicted given the others .

The Pseudo-max Approach

- Find **w** such that for all sample points (x,y):

$$y_i = \arg\max_{\hat{y}_i} \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x)$$

The Pseudo-max Approach

- Find **w** such that for all sample points (x,y):

$$y_i = \arg\max_{\hat{y}_i} \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x)$$

$$i = 3$$

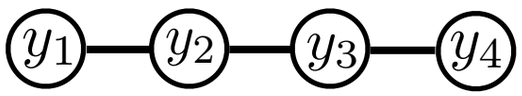
$$y^{-3} = [N, V, ?, N]$$

$$\left[y^{-3}, \hat{y}_3\right] = [N, V, \hat{y}_3, N]$$

The Pseudo-max Approach

- Find **w** such that for all sample points (x,y):

$$y_i = \arg\max_{\hat{y}_i} \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x)$$



N V D N

$i = 3$

$$y^{-3} = [N, V, ?, N]$$

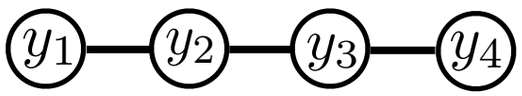
$$\left[y^{-3}, \hat{y}_3\right] = [N, V, \hat{y}_3, N]$$

- Turns the problem into standard multi-class with a single label.

The Pseudo-max Approach

- Find **w** such that for all sample points (x,y):

$$y_i = \arg\max_{\hat{y}_i} \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x)$$



N V D N

$$i = 3$$

$$y^{-3} = [N, V, ?, N]$$

$$\left[y^{-3}, \hat{y}_3\right] = [N, V, \hat{y}_3, N]$$

- Turns the problem into standard multi-class with a single label.
- Easy to optimize (e.g., by minimizing hinge loss).

The Pseudo-max Approach

- Instead of the original constraints:

$$\forall \bar{y} \neq y : \boldsymbol{w} \cdot \boldsymbol{\phi}(y, x) > \boldsymbol{w} \cdot \boldsymbol{\phi}(\bar{y}, x)$$

The Pseudo-max Approach

- Instead of the original constraints:

$$\forall \bar{y} \neq y : \boldsymbol{w} \cdot \boldsymbol{\phi}(y, x) > \boldsymbol{w} \cdot \boldsymbol{\phi}(\bar{y}, x)$$

- Use just O(n) constraints:

$$\boldsymbol{w} \cdot \boldsymbol{\phi}(y, x) > \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x) \quad , \quad \forall i, \hat{y}_i \neq y_i$$

The Pseudo-max Approach

- Instead of the original constraints:

$$\forall \bar{y} \neq y : \boldsymbol{w} \cdot \boldsymbol{\phi}(y, x) > \boldsymbol{w} \cdot \boldsymbol{\phi}(\bar{y}, x)$$

- Use just O(n) constraints:

$$\boldsymbol{w} \cdot \boldsymbol{\phi}(y, x) > \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x) \quad , \quad \forall i, \hat{y}_i \neq y_i$$

$$\boldsymbol{w} \cdot \boldsymbol{\phi}([N, V, D, N], x) > \boldsymbol{w} \cdot \boldsymbol{\phi}([D, V, D, N]], x)$$

$$\boldsymbol{w} \cdot \boldsymbol{\phi}([N, V, D, N], x) > \boldsymbol{w} \cdot \boldsymbol{\phi}([V, V, D, N]], x)$$

$$\boldsymbol{w} \cdot \boldsymbol{\phi}([N, V, D, N], x) > \boldsymbol{w} \cdot \boldsymbol{\phi}([D, D, D, N]], x)$$

$$\boldsymbol{w} \cdot \boldsymbol{\phi}([N, V, D, N], x) > \boldsymbol{w} \cdot \boldsymbol{\phi}([D, N, D, N]], x)$$

$$\vdots$$

The Pseudo-max Approach

- Instead of the original constraints:

$$\forall \bar{y} \neq y : \boldsymbol{w} \cdot \boldsymbol{\phi}(y, x) > \boldsymbol{w} \cdot \boldsymbol{\phi}(\bar{y}, x)$$

- Use just O(n) constraints:

$$\boldsymbol{w} \cdot \boldsymbol{\phi}(y, x) > \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x) \quad , \quad \forall i, \hat{y}_i \neq y_i$$

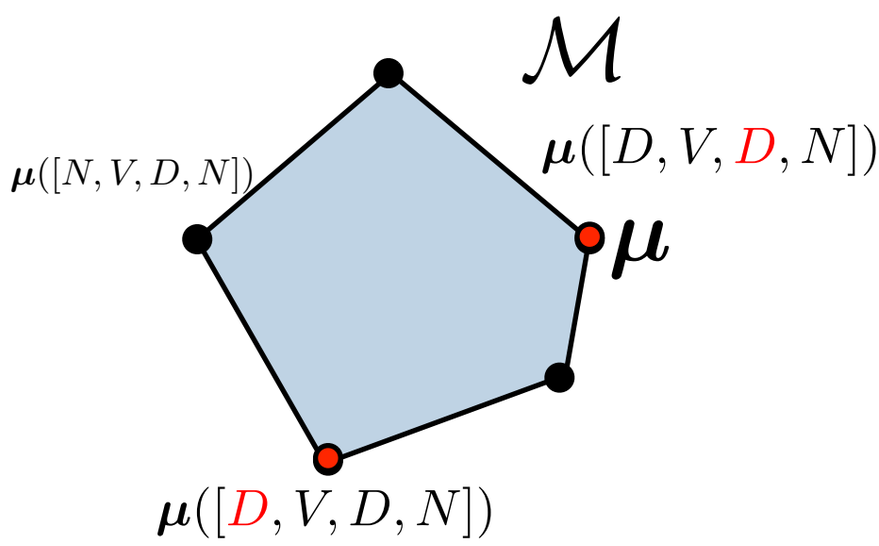
$$\boldsymbol{w} \cdot \boldsymbol{\phi}([N, V, D, N], x) > \boldsymbol{w} \cdot \boldsymbol{\phi}([D, V, D, N]], x)$$

$$\boldsymbol{w} \cdot \boldsymbol{\phi}([N, V, D, N], x) > \boldsymbol{w} \cdot \boldsymbol{\phi}([V, V, D, N]], x)$$

$$\boldsymbol{w} \cdot \boldsymbol{\phi}([N, V, D, N], x) > \boldsymbol{w} \cdot \boldsymbol{\phi}([D, D, D, N]], x)$$

$$\boldsymbol{w} \cdot \boldsymbol{\phi}([N, V, D, N], x) > \boldsymbol{w} \cdot \boldsymbol{\phi}([D, N, D, N]], x)$$

$$\vdots$$



Does it Work?

- Given a training set, define two sets.

Does it Work?

- Given a training set, define two sets.
- The set W of **w** that give correct *global* predictions:

Does it Work?

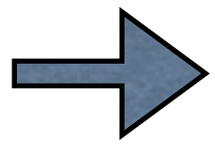
- Given a training set, define two sets.
- The set W of **w** that give correct *global* predictions:

$$\forall (x, y) \in D: \;\; y = \arg\max_{\hat{y}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\hat{y}, x)$$

Does it Work?

- Given a training set, define two sets.
- The set W of **w** that give correct *global* predictions:

$$\forall (x, y) \in D : \; y = \arg\max_{\hat{y}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\hat{y}, x)$$



Does it Work?

- Given a training set, define two sets.
- The set W of **w** that give correct *global* predictions:

$$\forall (x,y) \in D: \;\; y = \arg\max_{\hat{y}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\hat{y}, x) \quad \Rightarrow \quad \boldsymbol{w} \in W$$

Does it Work?

- Given a training set, define two sets.
- The set W of **w** that give correct *global* predictions:

$$\forall (x,y) \in D: \;\; y = \arg\max_{\hat{y}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\hat{y}, x) \quad \Rightarrow \quad \boldsymbol{w} \in W$$

Does it Work?

- Given a training set, define two sets.
- The set W of **w** that give correct *global* predictions:

$$\forall (x, y) \in D: \;\; y = \arg\max_{\hat{y}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\hat{y}, x) \quad \Rightarrow \quad \boldsymbol{w} \in W$$

- The set W_{ps} of **w** that give correct *local* predictions:

Does it Work?

- Given a training set, define two sets.

- The set W of **w** that give correct *global* predictions:

$$\forall (x,y) \in D: \;\; y = \arg\max_{\hat{y}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\hat{y}, x) \quad \Rightarrow \quad \boldsymbol{w} \in W$$

- The set W_{ps} of **w** that give correct *local* predictions:

$$\forall (x,y) \in D, \forall i : y_i = \arg\max_{\hat{y}_i} \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x)$$

Does it Work?

- Given a training set, define two sets.
- The set W of **w** that give correct *global* predictions:

$$\forall (x,y) \in D: \;\; y = \arg\max_{\hat{y}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\hat{y}, x) \quad \Rightarrow \quad \boldsymbol{w} \in W$$

- The set W_{ps} of **w** that give correct *local* predictions:

$$\forall (x,y) \in D, \forall i : y_i = \arg\max_{\hat{y}_i} \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x) \Rightarrow$$

Does it Work?

- Given a training set, define two sets.

- The set W of **w** that give correct *global* predictions:

$$\forall (x,y) \in D: \;\; y = \arg\max_{\hat{y}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\hat{y}, x) \quad \Rightarrow \quad \boldsymbol{w} \in W$$

- The set W_{ps} of **w** that give correct *local* predictions:

$$\forall (x,y) \in D, \forall i : y_i = \arg\max_{\hat{y}_i} \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x) \quad \Rightarrow \quad \boldsymbol{w} \in W_{ps}$$

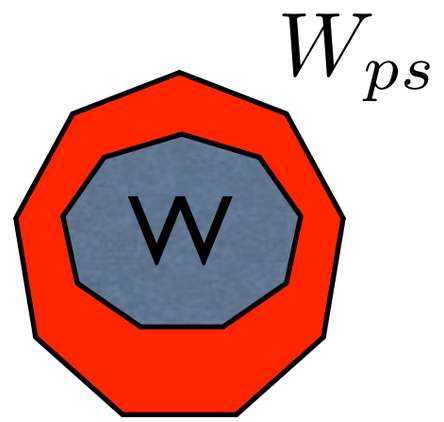
Does it Work?

- Given a training set, define two sets.
- The set W of **w** that give correct *global* predictions:

$$\forall (x,y) \in D: \;\; y = \arg\max_{\hat{y}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\hat{y}, x) \quad \Rightarrow \quad \boldsymbol{w} \in W$$

- The set W_{ps} of **w** that give correct *local* predictions:

$$\forall (x,y) \in D, \forall i : y_i = \arg\max_{\hat{y}_i} \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x) \quad \Rightarrow \quad \boldsymbol{w} \in W_{ps}$$



Does it Work?

- Given a training set, define two sets.

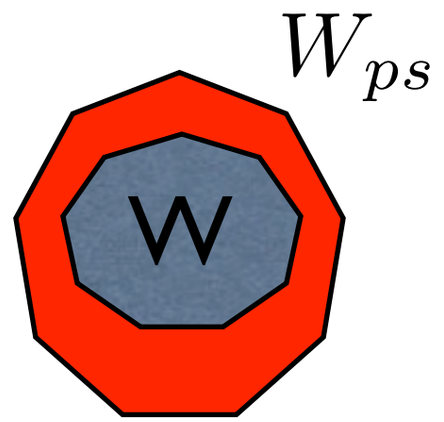
- The set W of **w** that give correct *global* predictions:

$$\forall (x,y) \in D: \;\; y = \arg\max_{\hat{y}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\hat{y}, x) \quad \Rightarrow \quad \boldsymbol{w} \in W$$

- The set W_{ps} of **w** that give correct *local* predictions:

$$\forall (x,y) \in D, \forall i : y_i = \arg\max_{\hat{y}_i} \boldsymbol{w} \cdot \boldsymbol{\phi}(\left[y^{-i}, \hat{y}_i\right], x) \quad \Rightarrow \quad \boldsymbol{w} \in W_{ps}$$

- Generally these sets are different, since *local* optimality does not imply *global* optimality...



Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

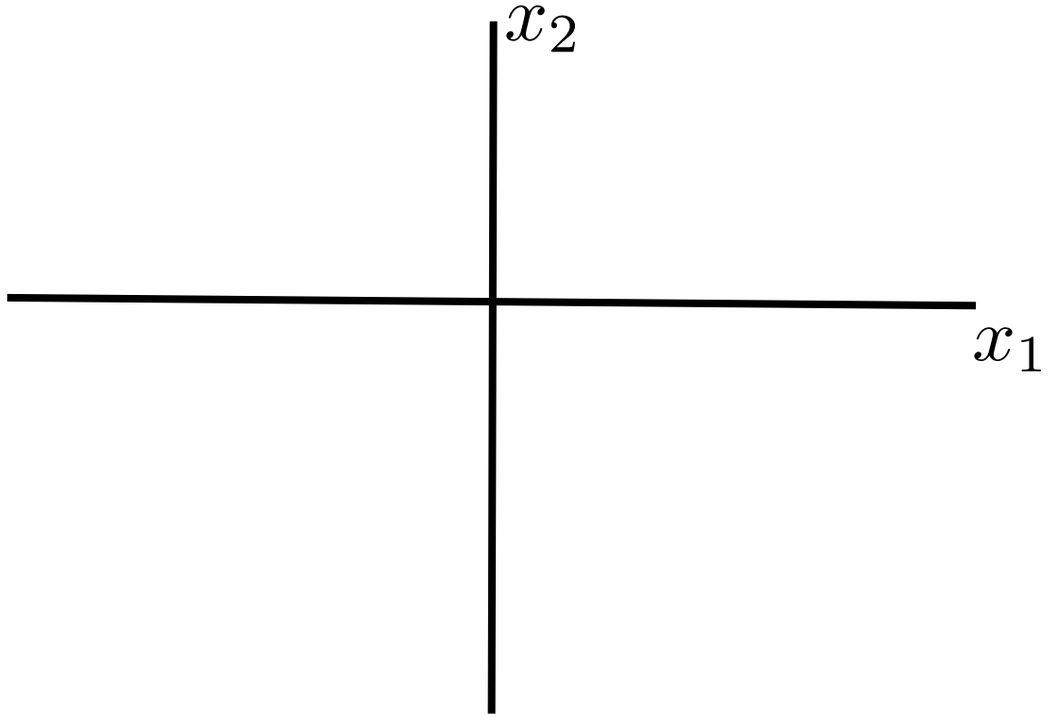
$$\max_{y_1, y_2} [w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2]$$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$

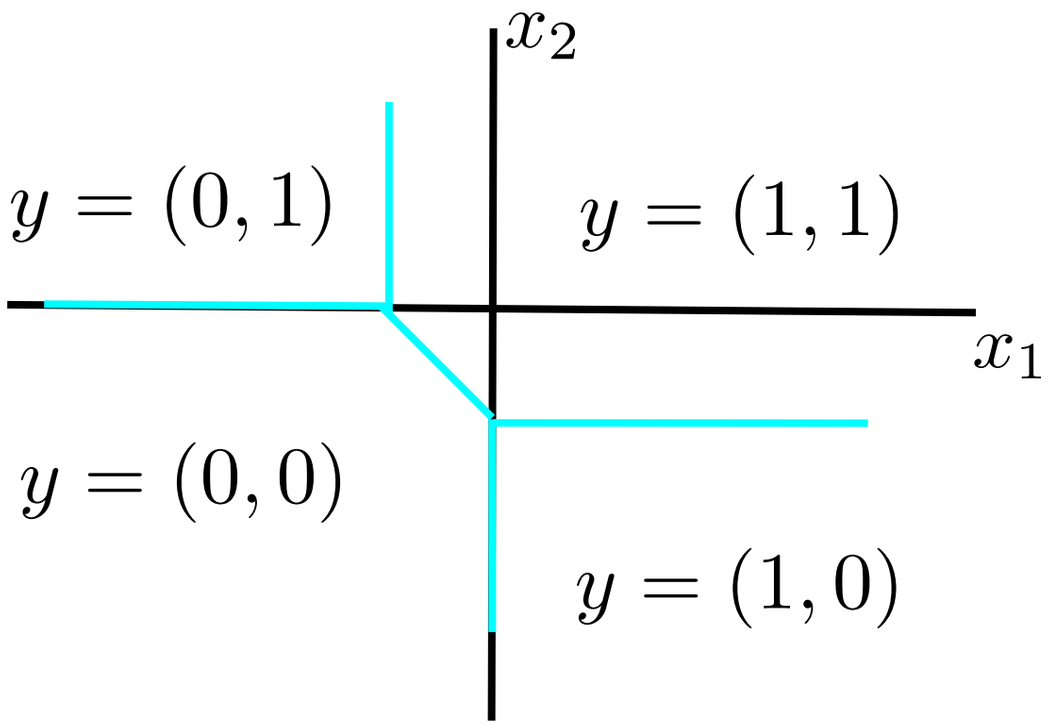


Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$

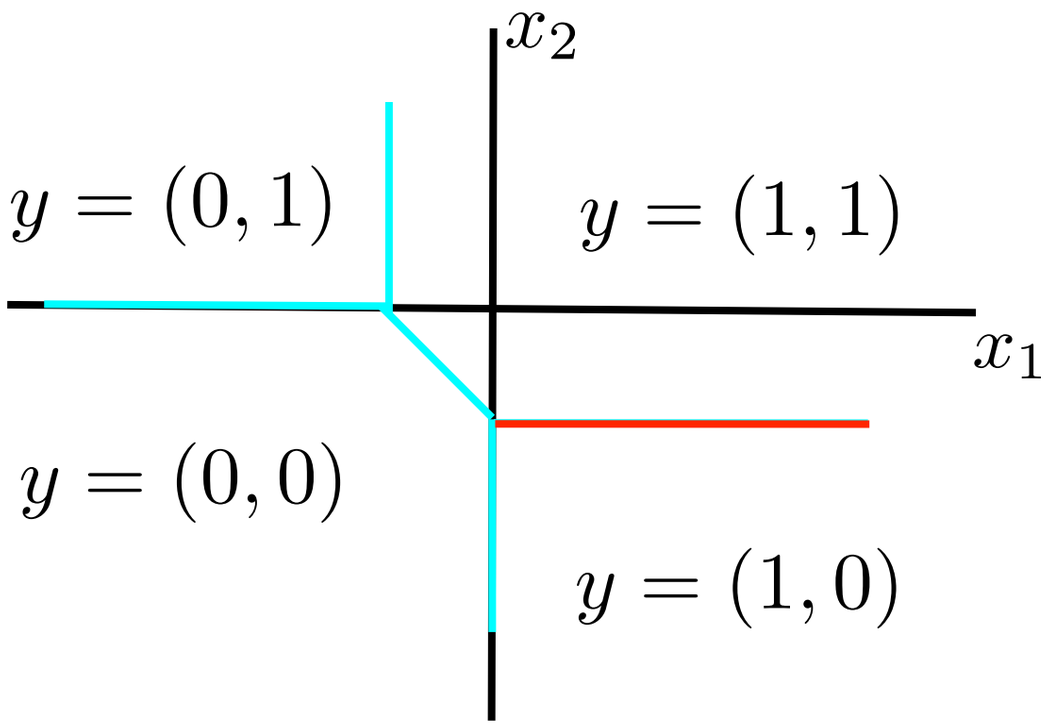


Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$

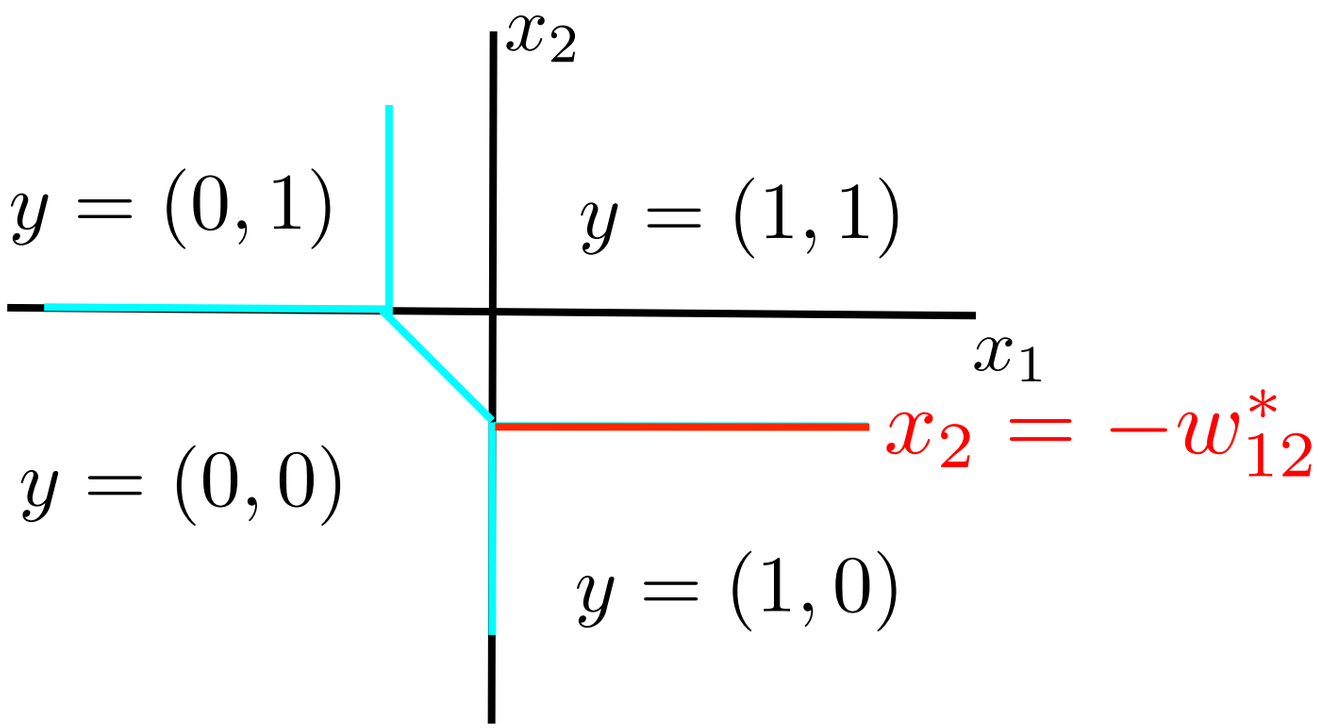


Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$

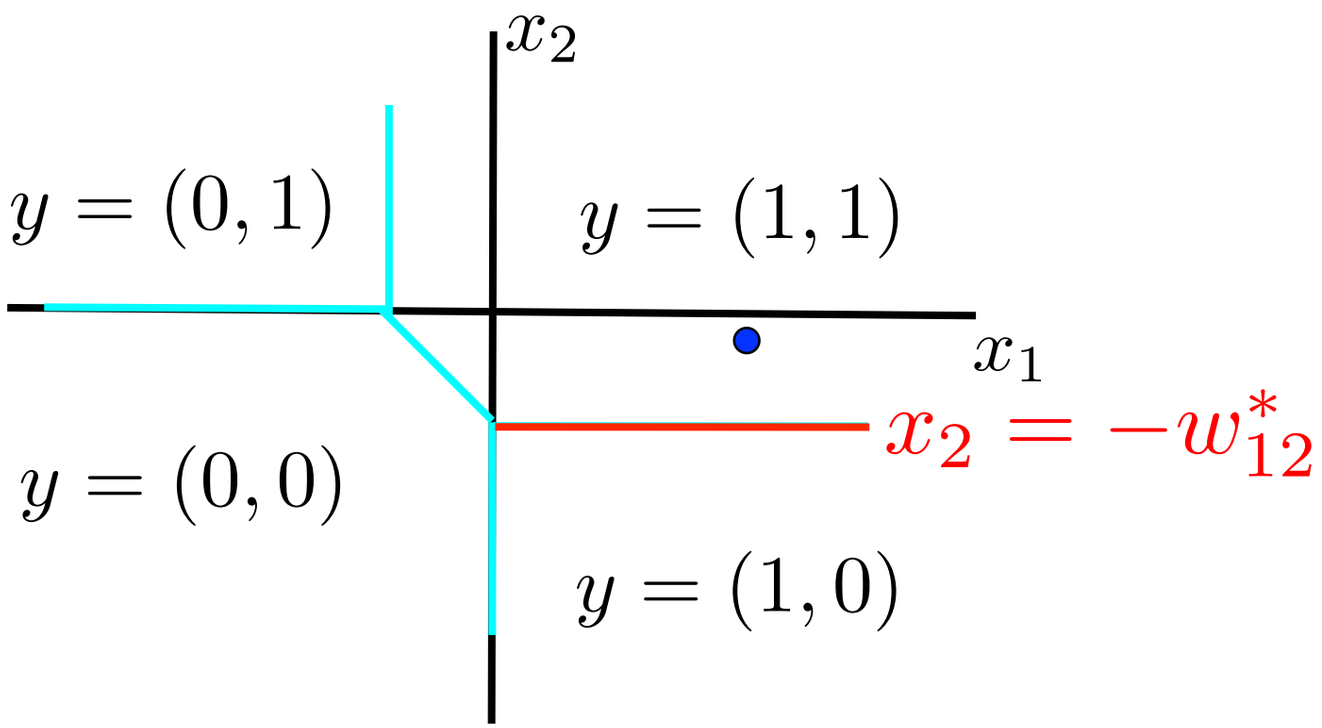


Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$

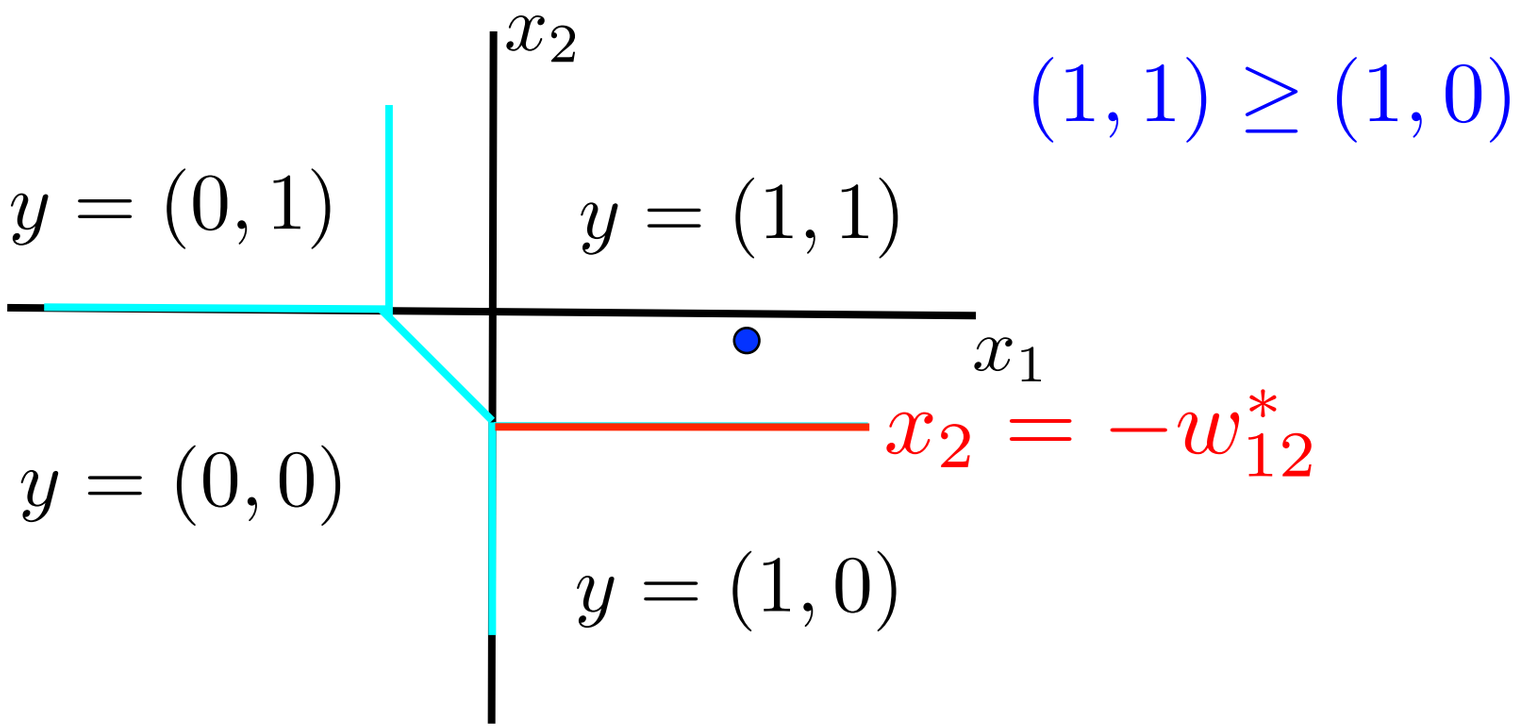


Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$

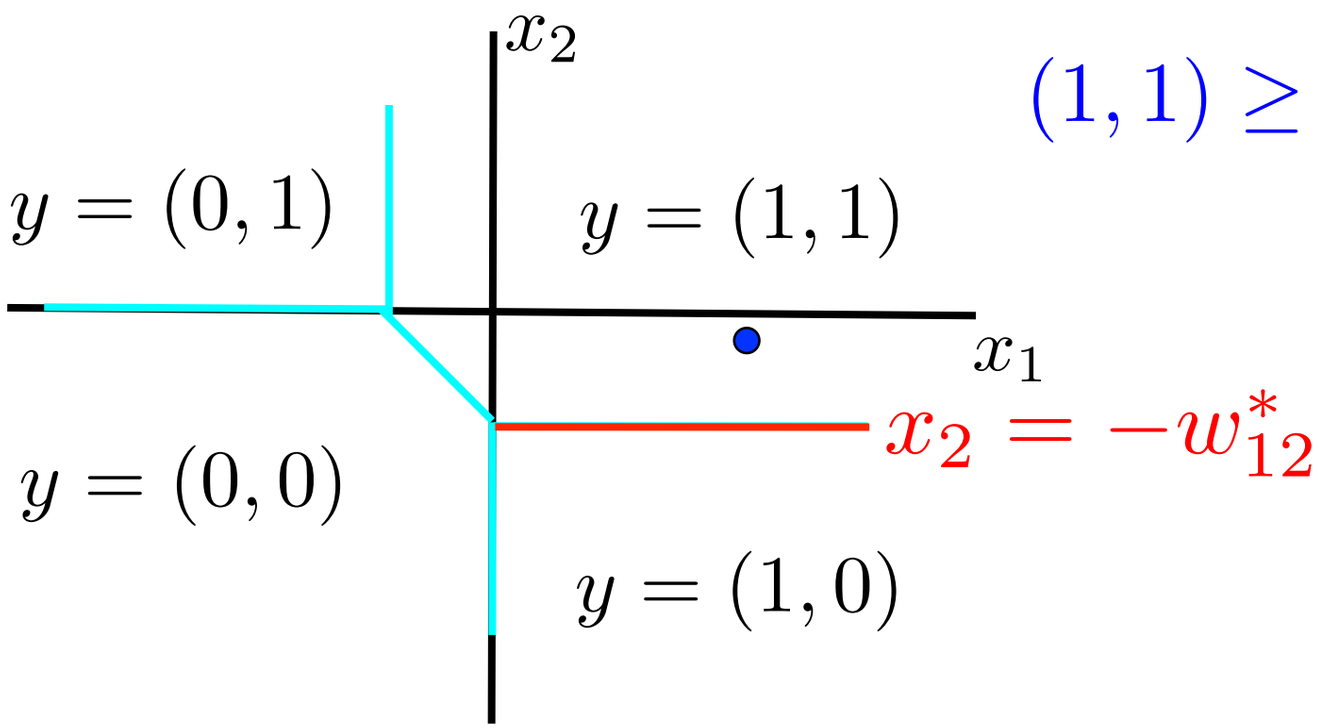


Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



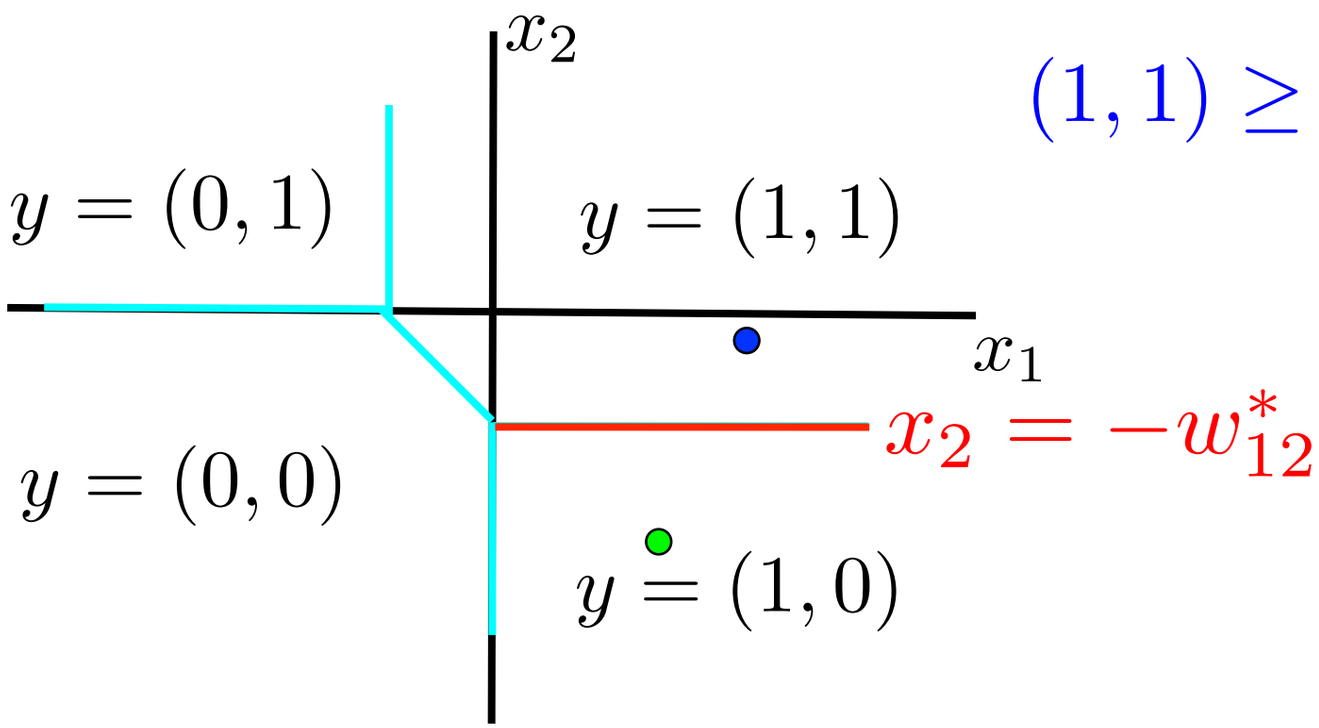
$$(1, 1) \geq (1, 0) \quad w_{12} \geq -x_2$$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



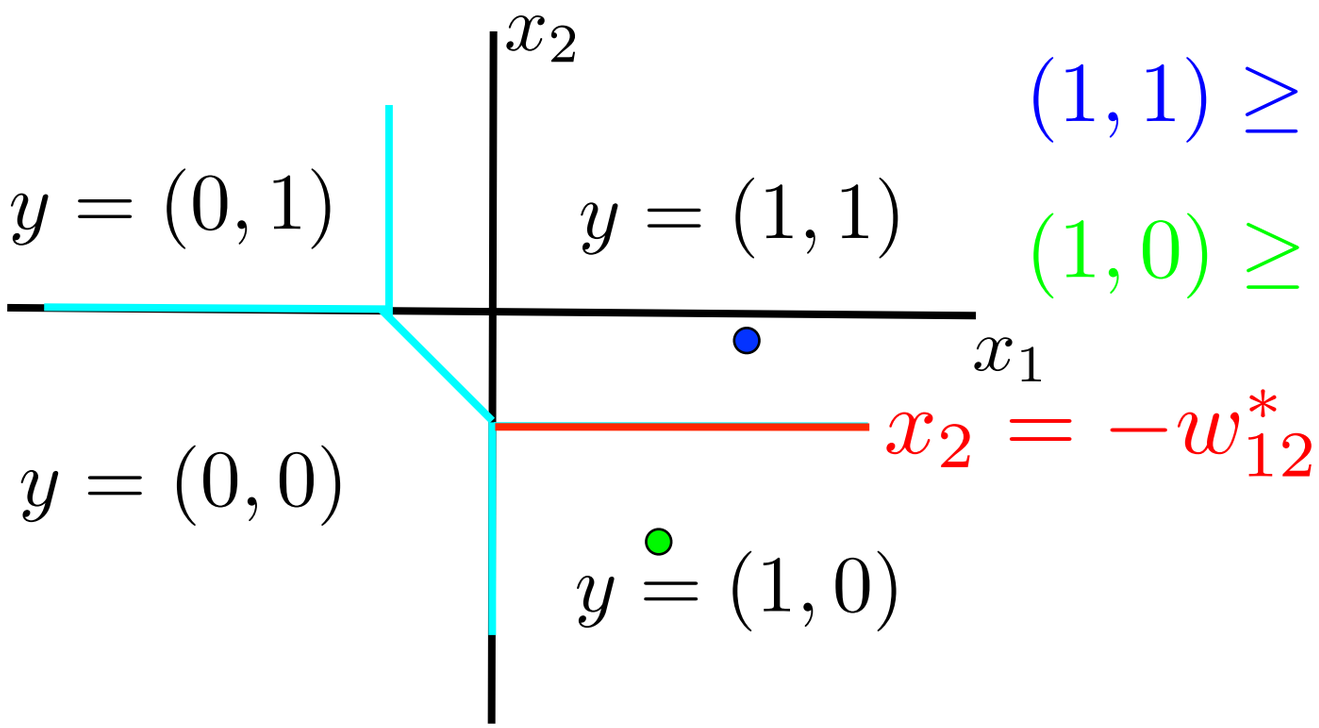
$$(1, 1) \geq (1, 0) \quad w_{12} \geq -x_2$$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$

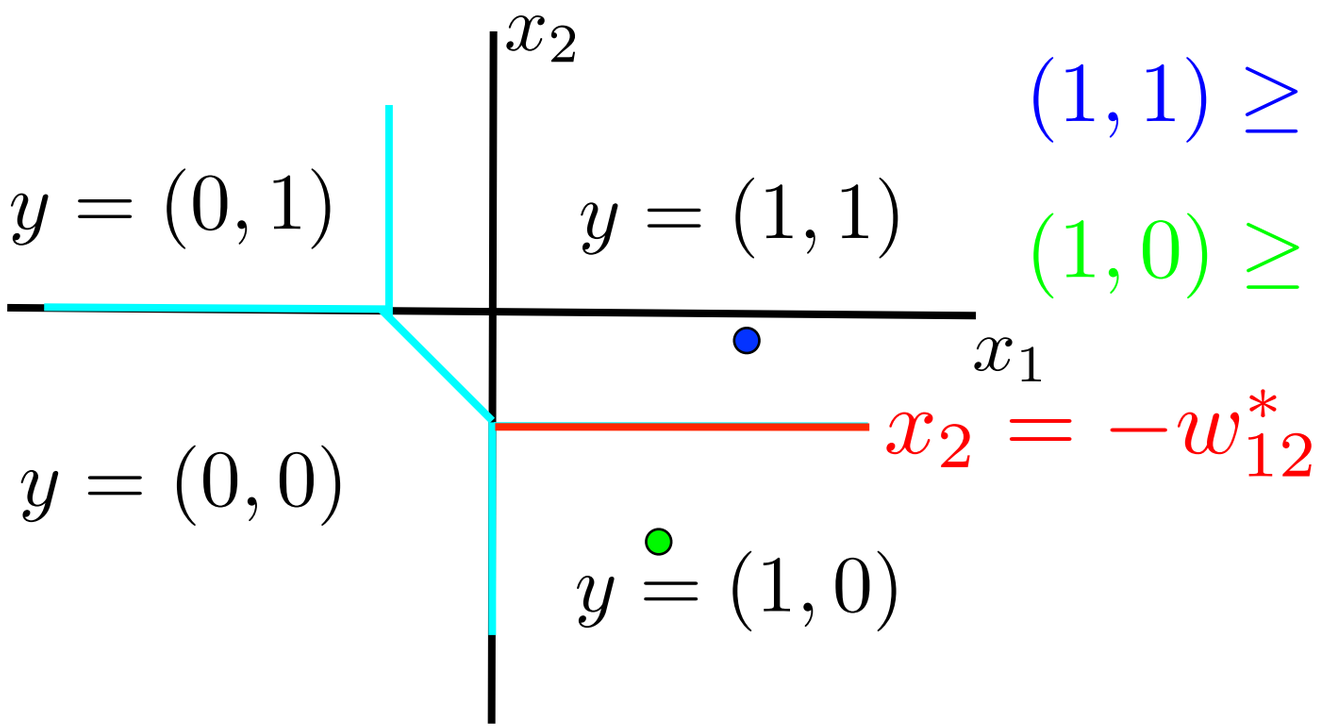
$(1,0) \geq (1,1)$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$

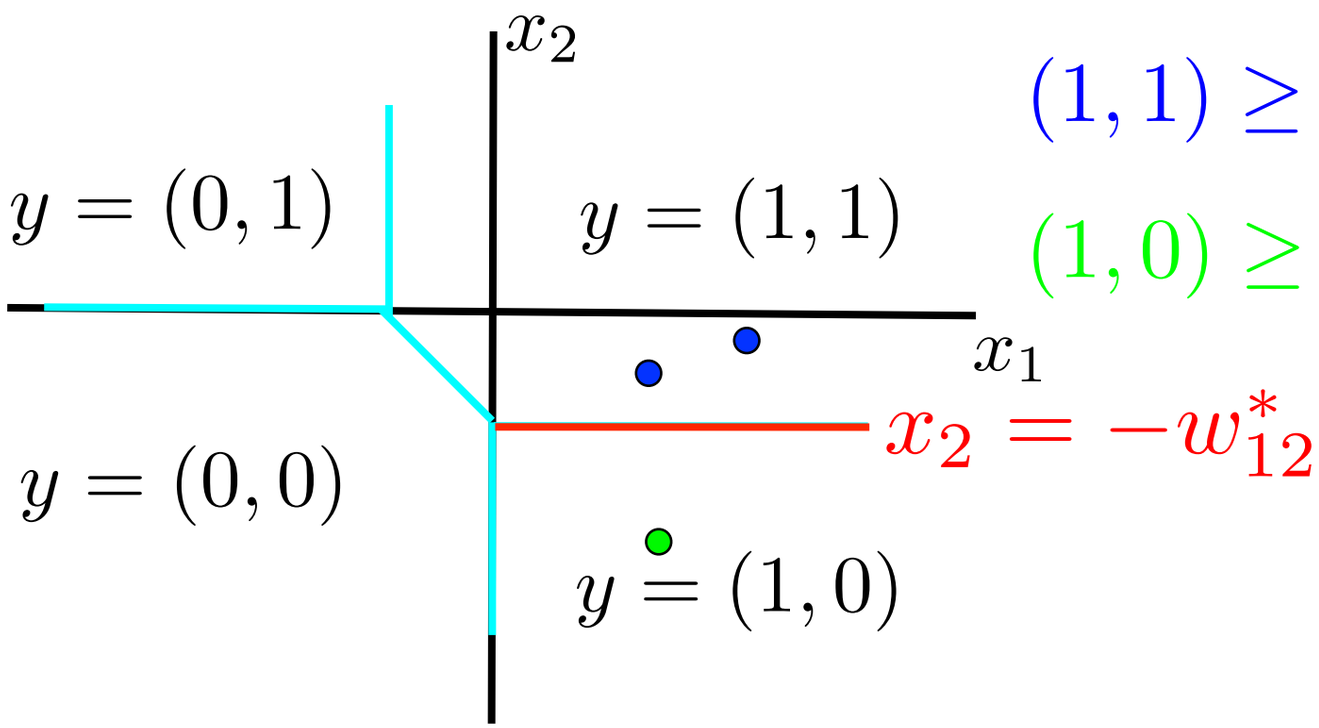
$(1,0) \geq (1,1) \quad w_{12} \leq -x_2$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$

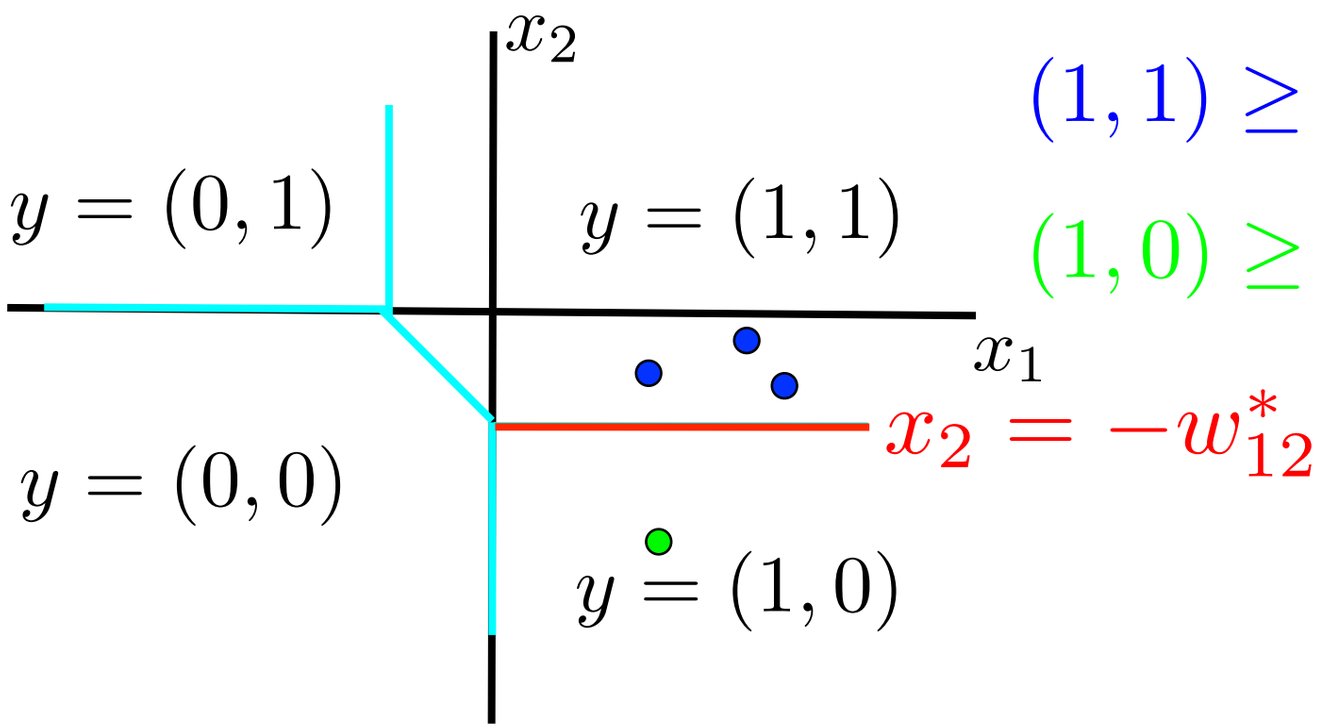
$(1,0) \geq (1,1) \quad w_{12} \leq -x_2$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$

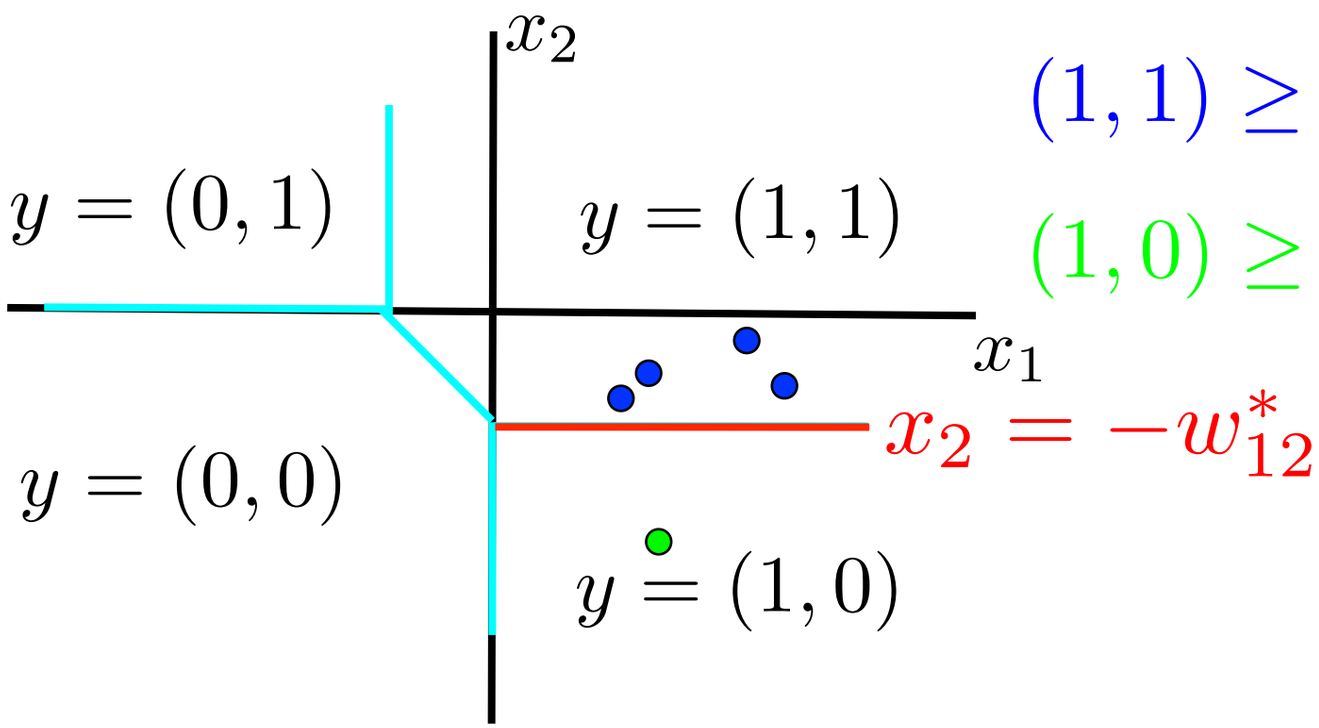
$(1,0) \geq (1,1) \quad w_{12} \leq -x_2$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$

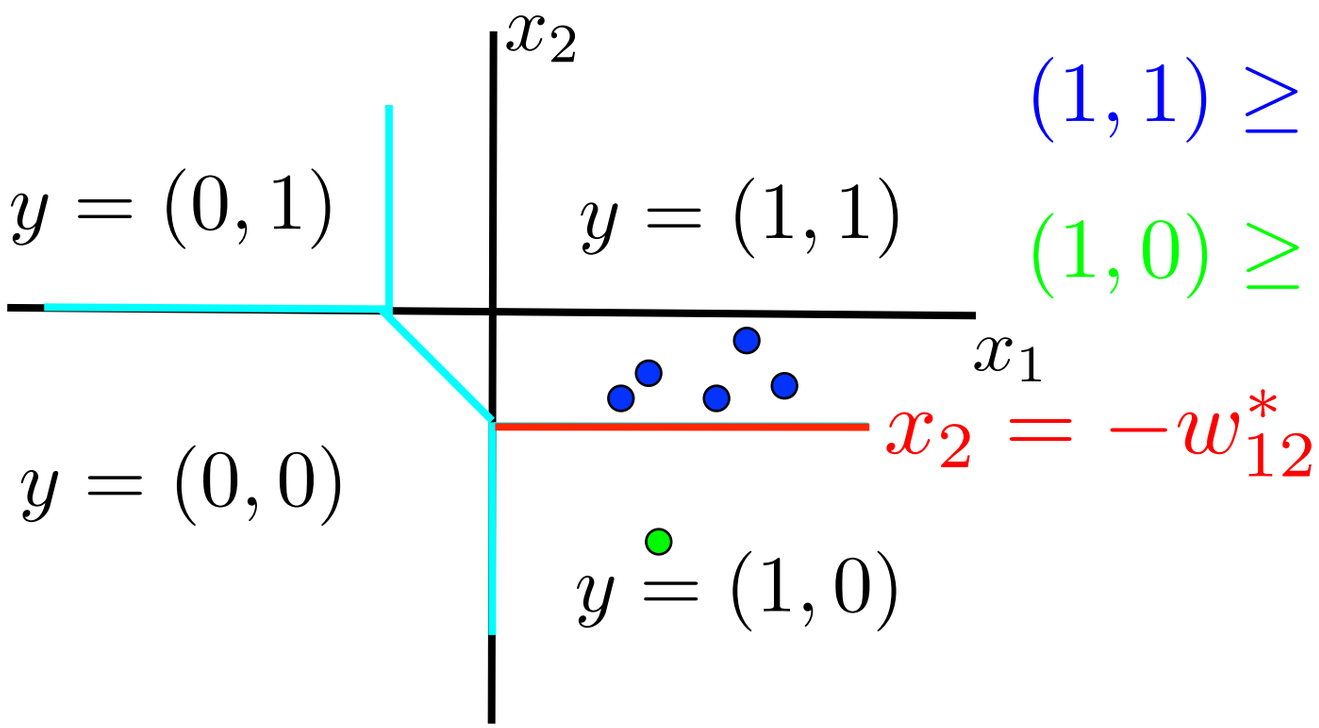
$(1,0) \geq (1,1) \quad w_{12} \leq -x_2$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$

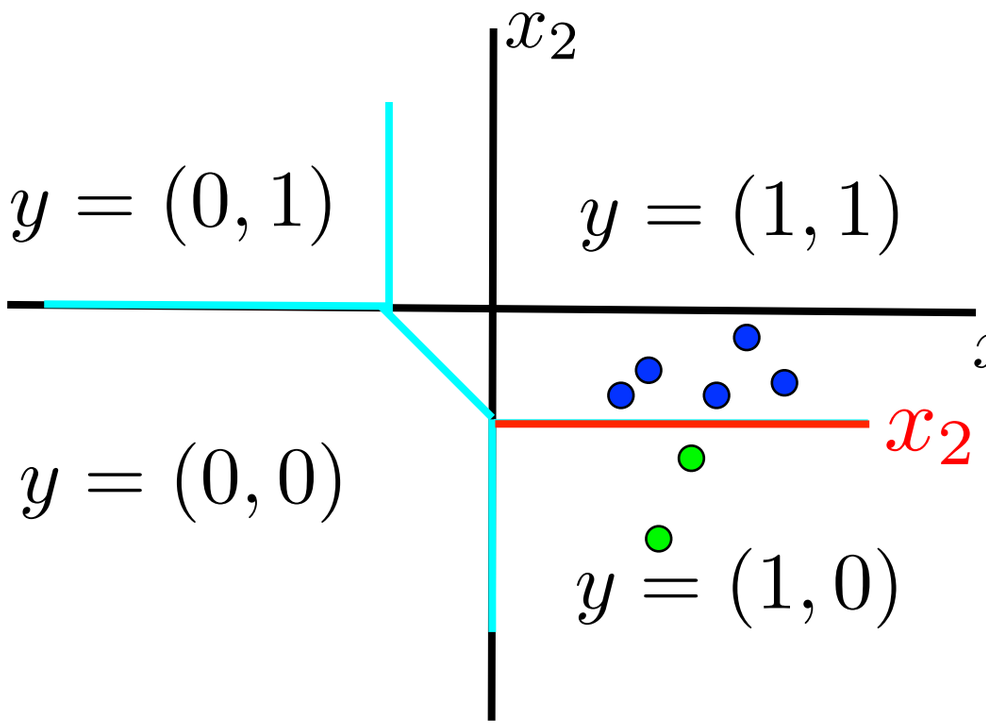
$(1,0) \geq (1,1) \quad w_{12} \leq -x_2$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



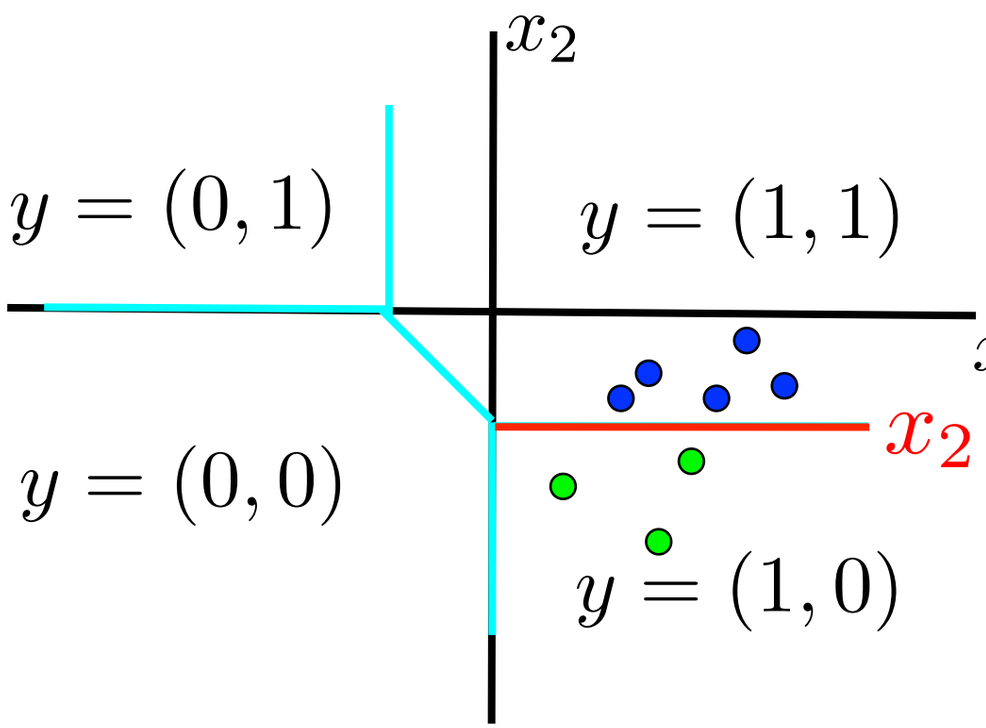
$$(1, 1) \geq (1, 0) \quad w_{12} \geq -x_2$$
$$(1, 0) \geq (1, 1) \quad w_{12} \leq -x_2$$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



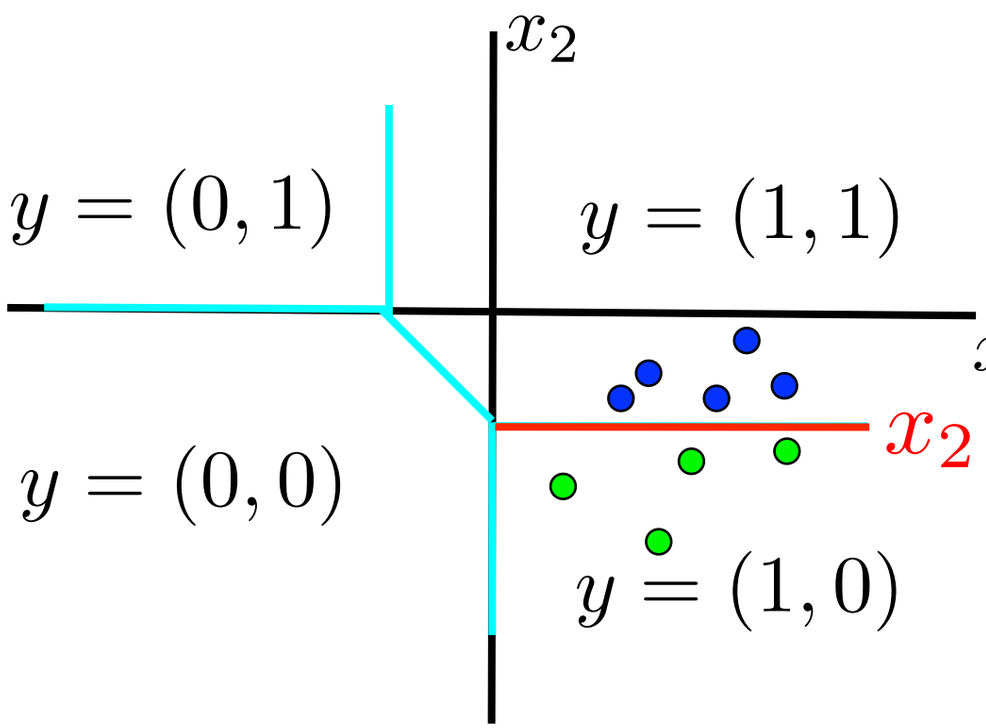
$$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$$
$$(1,0) \geq (1,1) \quad w_{12} \leq -x_2$$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$

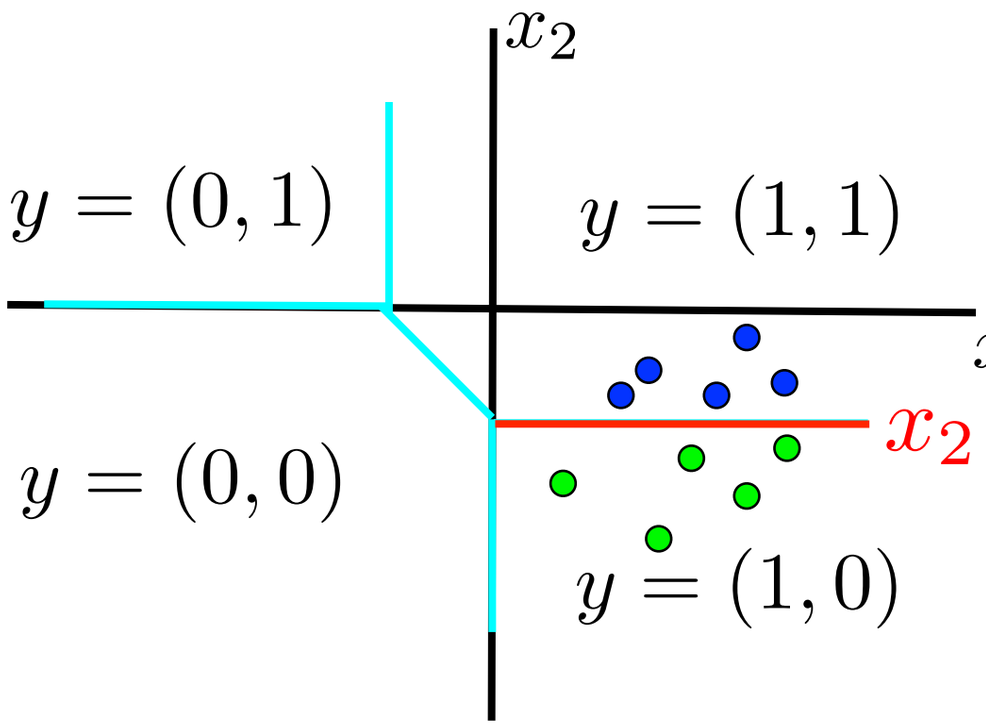
$(1,0) \geq (1,1) \quad w_{12} \leq -x_2$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$

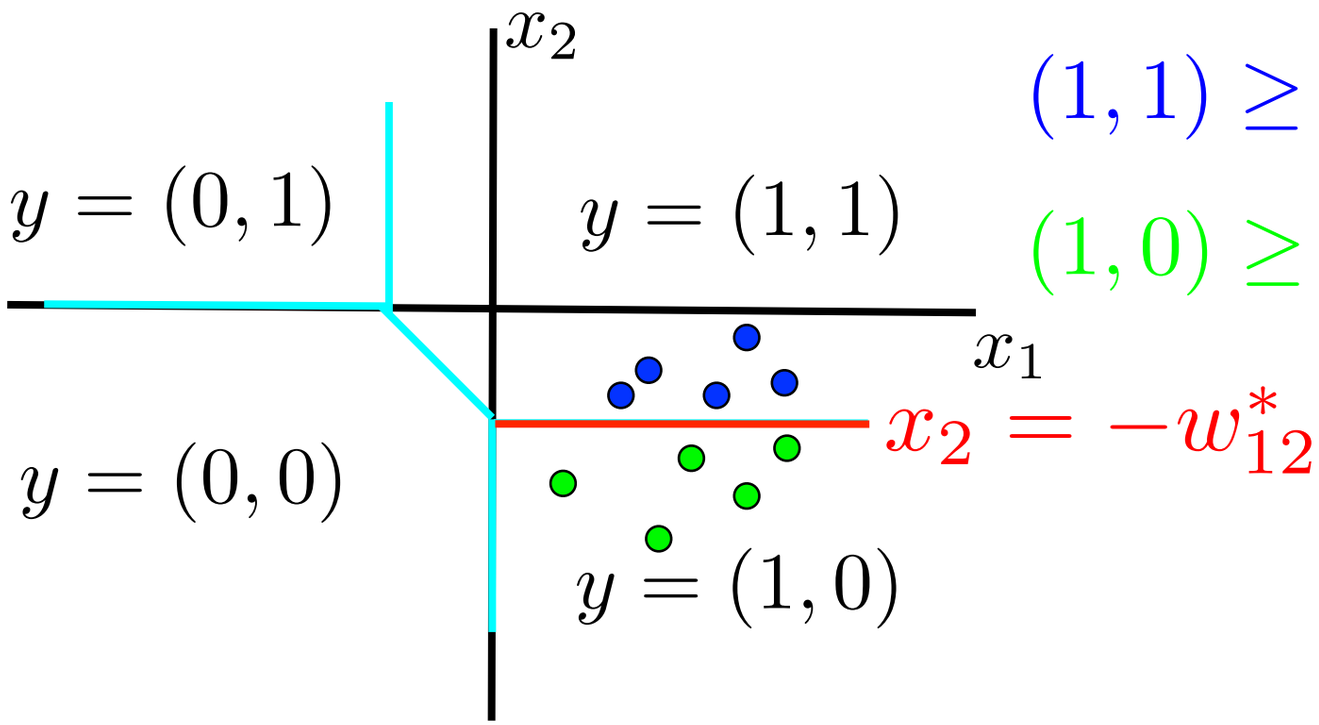
$(1,0) \geq (1,1) \quad w_{12} \leq -x_2$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$

$(1,0) \geq (1,1) \quad w_{12} \leq -x_2$

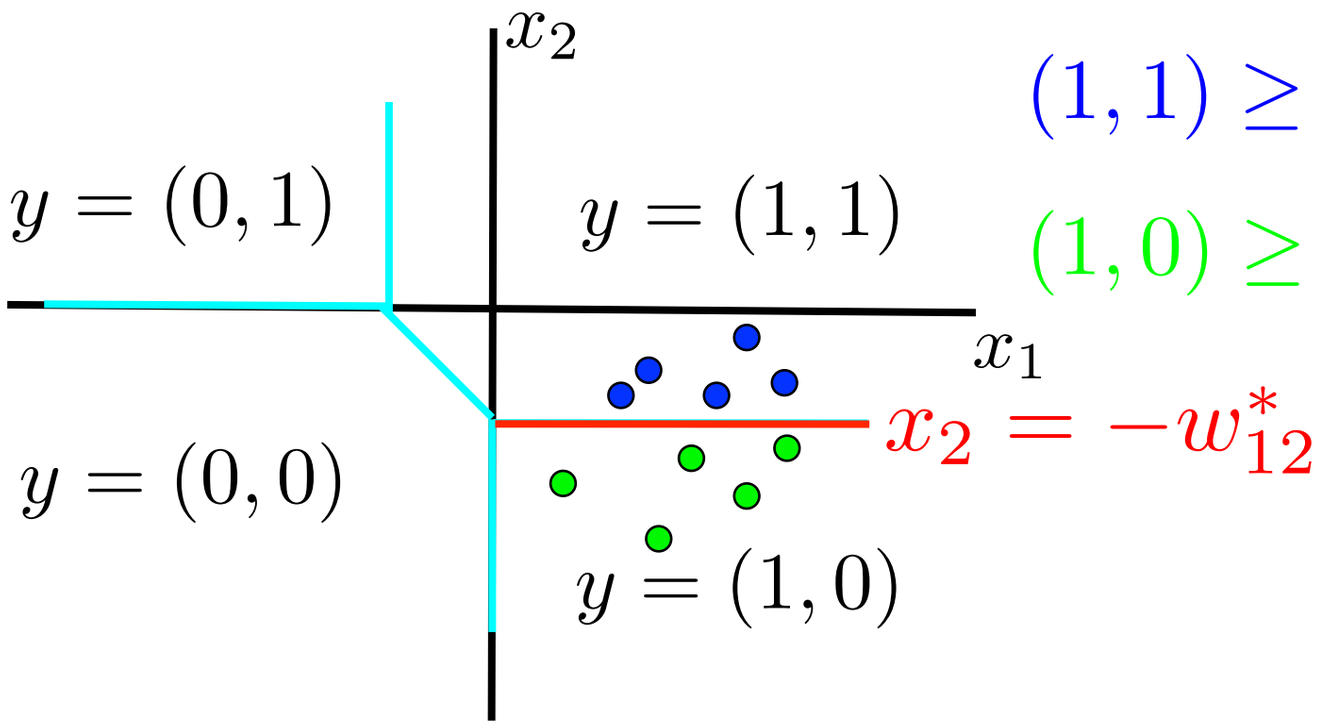
$\downarrow n \longrightarrow \infty$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$

$(1,0) \geq (1,1) \quad w_{12} \leq -x_2$

$\downarrow n \to \infty$

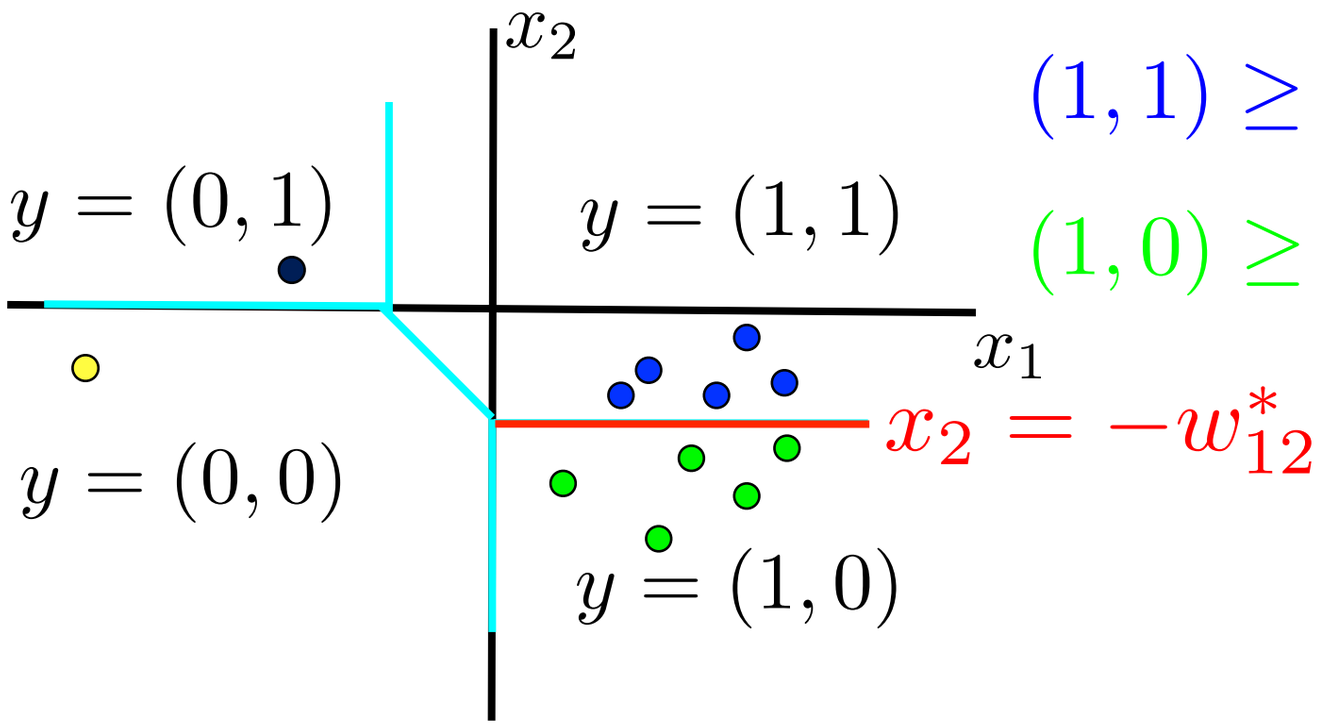
$w_{12} = w_{12}^*$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



$(1, 1) \geq (1, 0) \quad w_{12} \geq -x_2$

$(1, 0) \geq (1, 1) \quad w_{12} \leq -x_2$

$\downarrow n \longrightarrow \infty$

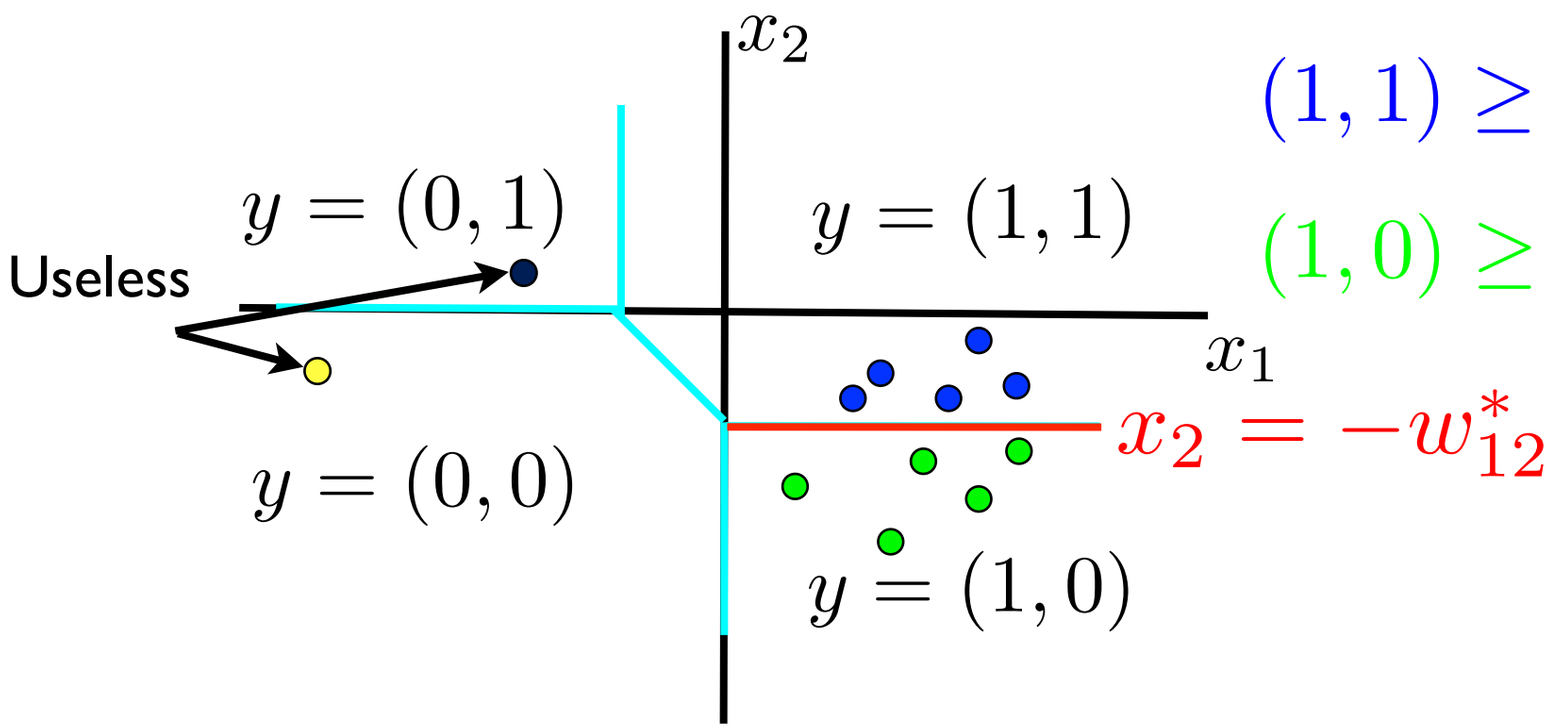
$w_{12} = w_{12}^*$

Does it Ever Work?

- Can we ever find a globally separating **w** with this approach?
- What if we had a lot of data?

$$x_i \in \mathbb{R}$$
$$y_i \in \{0, 1\}$$

$$\max_{y_1, y_2} \left[w_{12} y_1 y_2 + y_1 x_1 + y_2 x_2 \right]$$



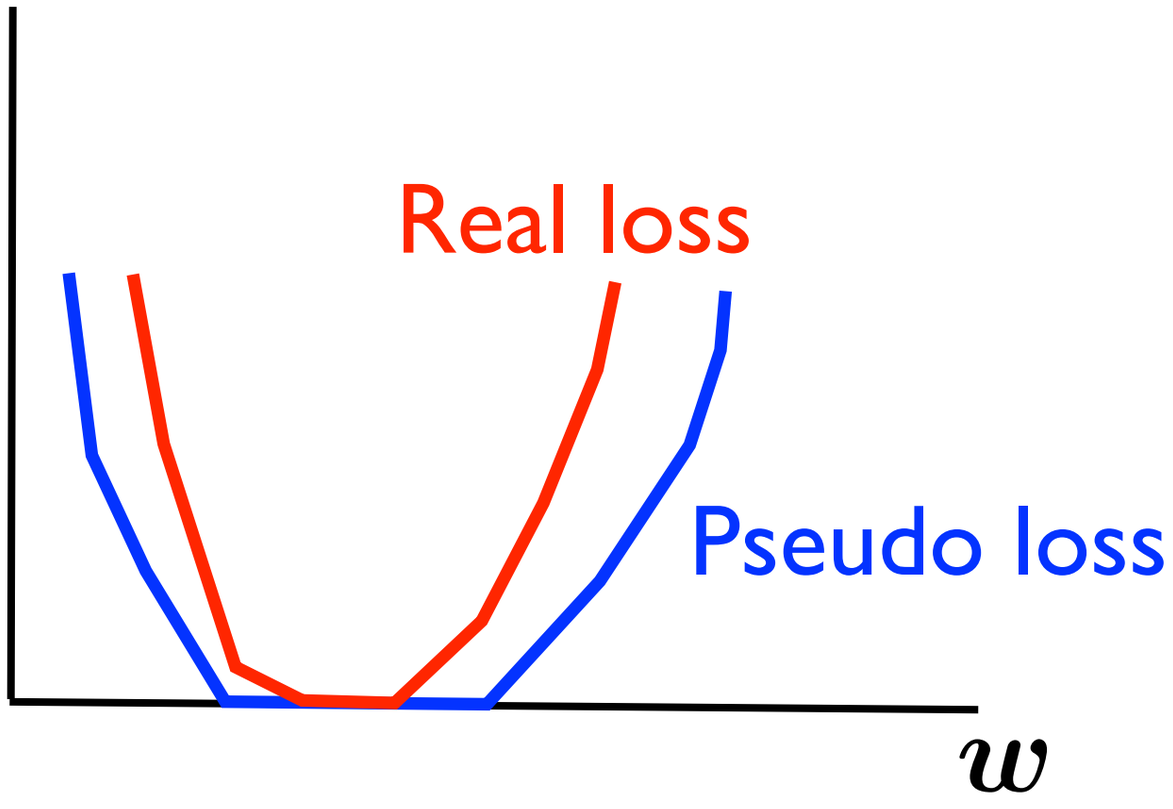
$$(1,1) \geq (1,0) \quad w_{12} \geq -x_2$$
$$(1,0) \geq (1,1) \quad w_{12} \leq -x_2$$
$$\downarrow n \longrightarrow \infty$$
$$w_{12} = w_{12}^*$$

Proof Idea

- Show that the hinge loss for the pseudo-max constraints becomes strictly convex as n goes to infinity.

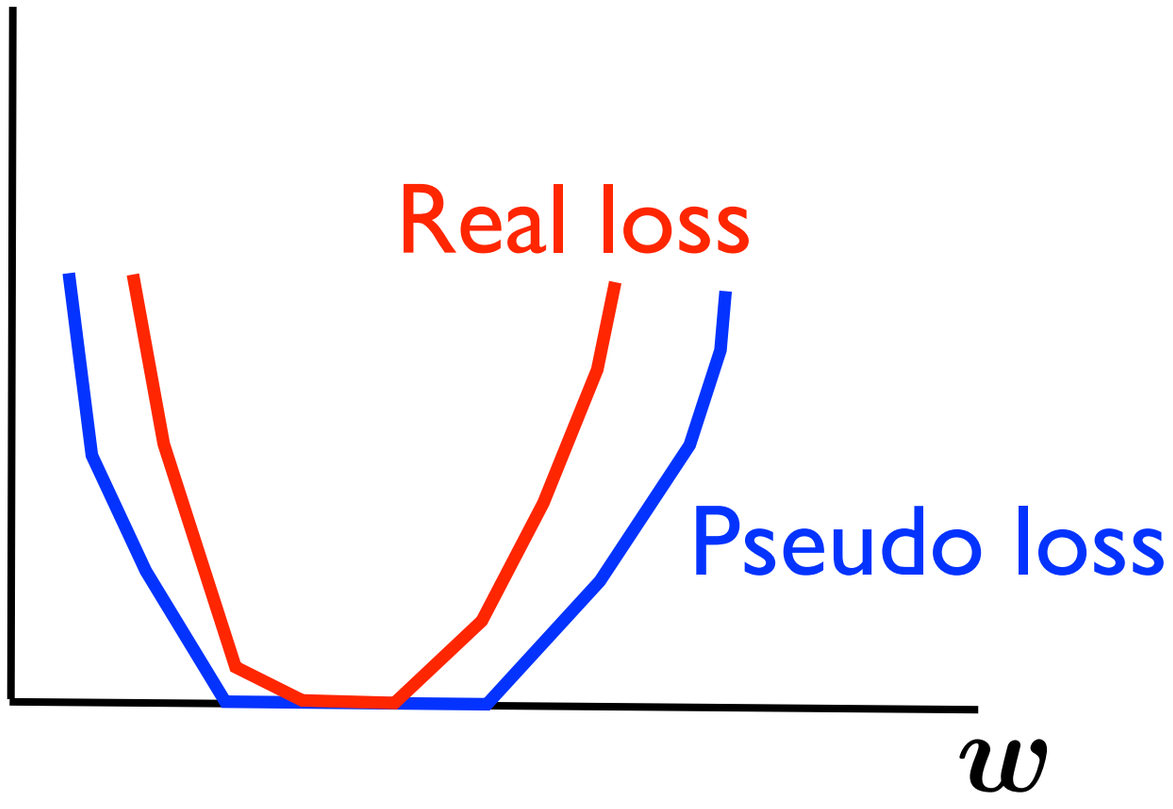
Proof Idea

- Show that the hinge loss for the pseudo-max constraints becomes strictly convex as n goes to infinity.

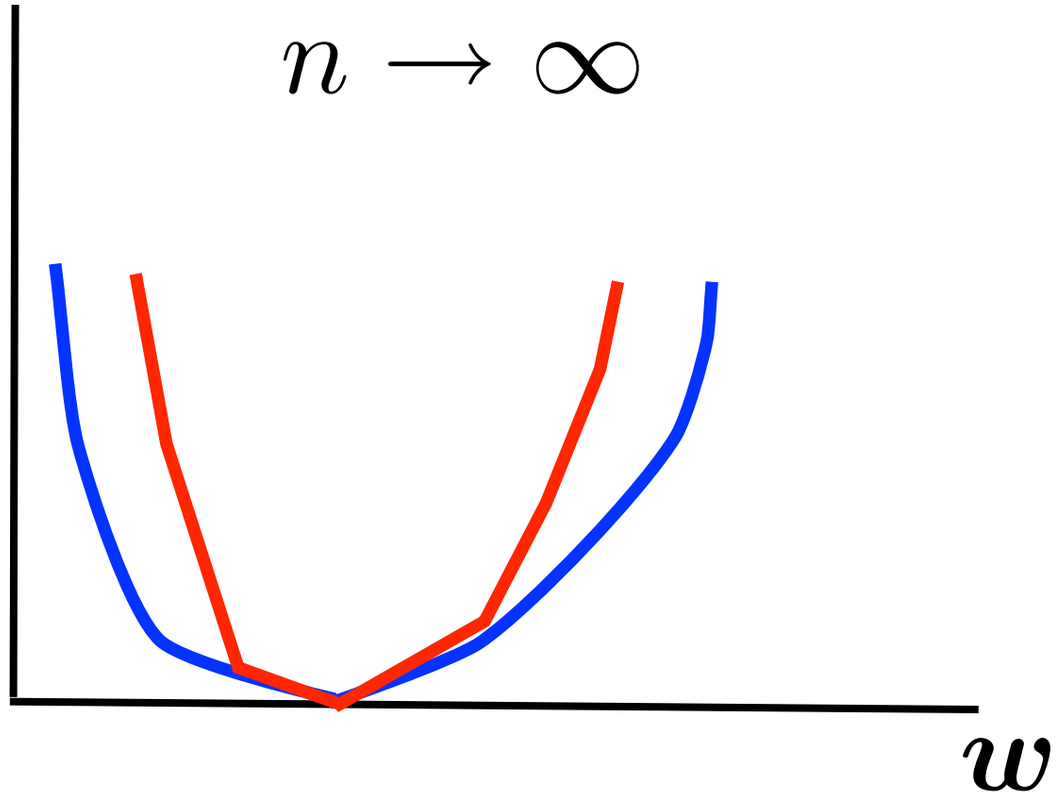


Proof Idea

- Show that the hinge loss for the pseudo-max constraints becomes strictly convex as n goes to infinity.



$$n \longrightarrow \infty$$



Comments

Comments

- Applies quite generally for models with continuous inputs.

Comments

- Applies quite generally for models with continuous inputs.
- Few inputs (polynomial in n) are needed to get good performance.

Comments

- Applies quite generally for models with continuous inputs.
- Few inputs (polynomial in n) are needed to get good performance.
- Very cheap to implement.

Comments

- Applies quite generally for models with continuous inputs.
- Few inputs (polynomial in n) are needed to get good performance.
- Very cheap to implement.
- Caveat: what do you do at prediction time. Options: LP relaxation, greedy (in line with the local optimality).

Comments

- Applies quite generally for models with continuous inputs.
- Few inputs (polynomial in n) are needed to get good performance.
- Very cheap to implement.
- Caveat: what do you do at prediction time. Options: LP relaxation, greedy (in line with the local optimality).
- Conclusion: more data can allow cheaper inference (similar intuition to *[Shalev Shwartz & Srebro 08]*).

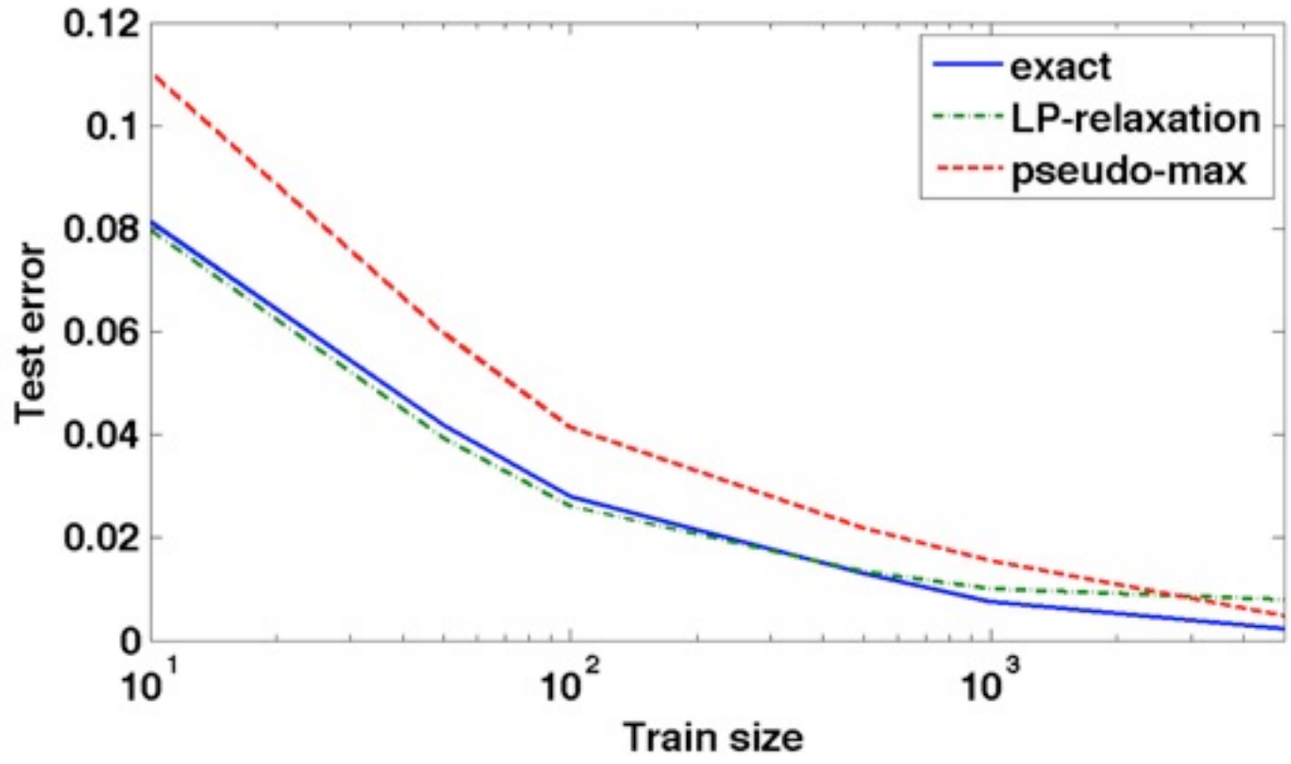
Experiments

Experiments

- Toy problem. Ten binary variables.

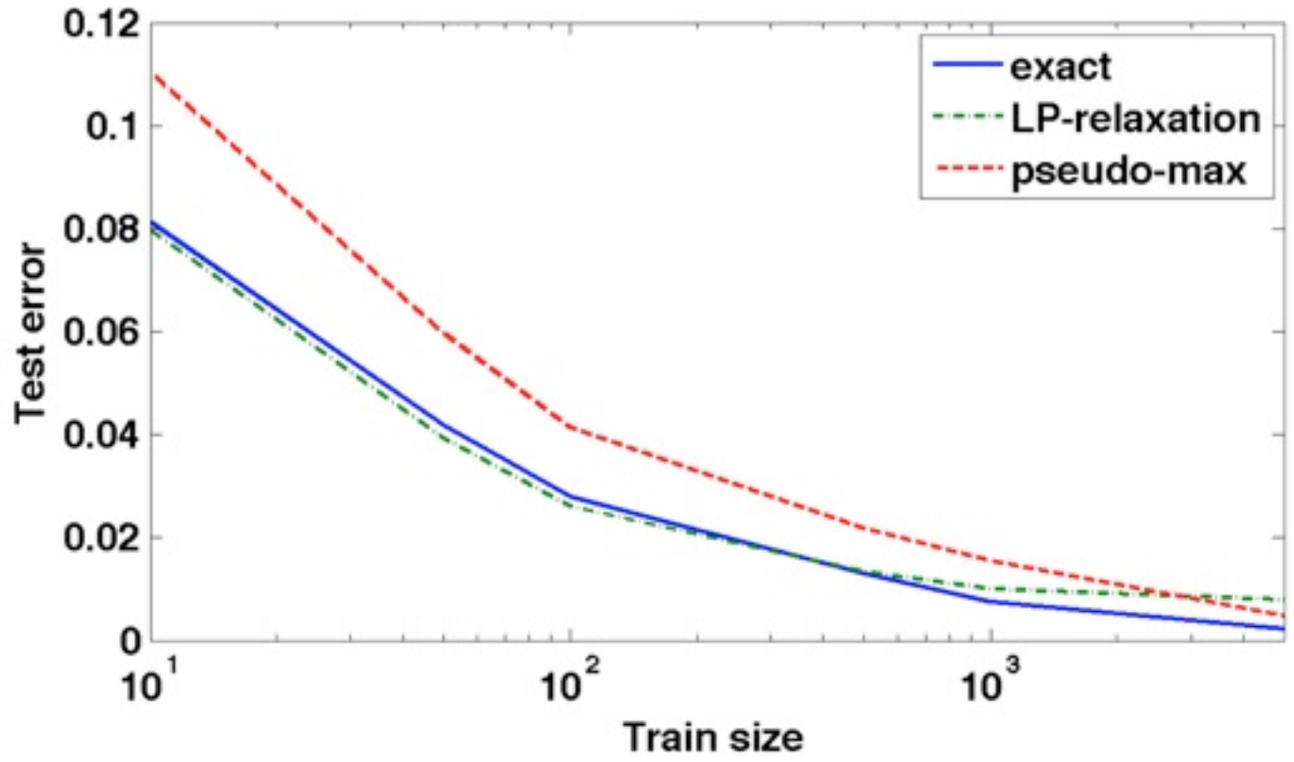
Experiments

- Toy problem. Ten binary variables.



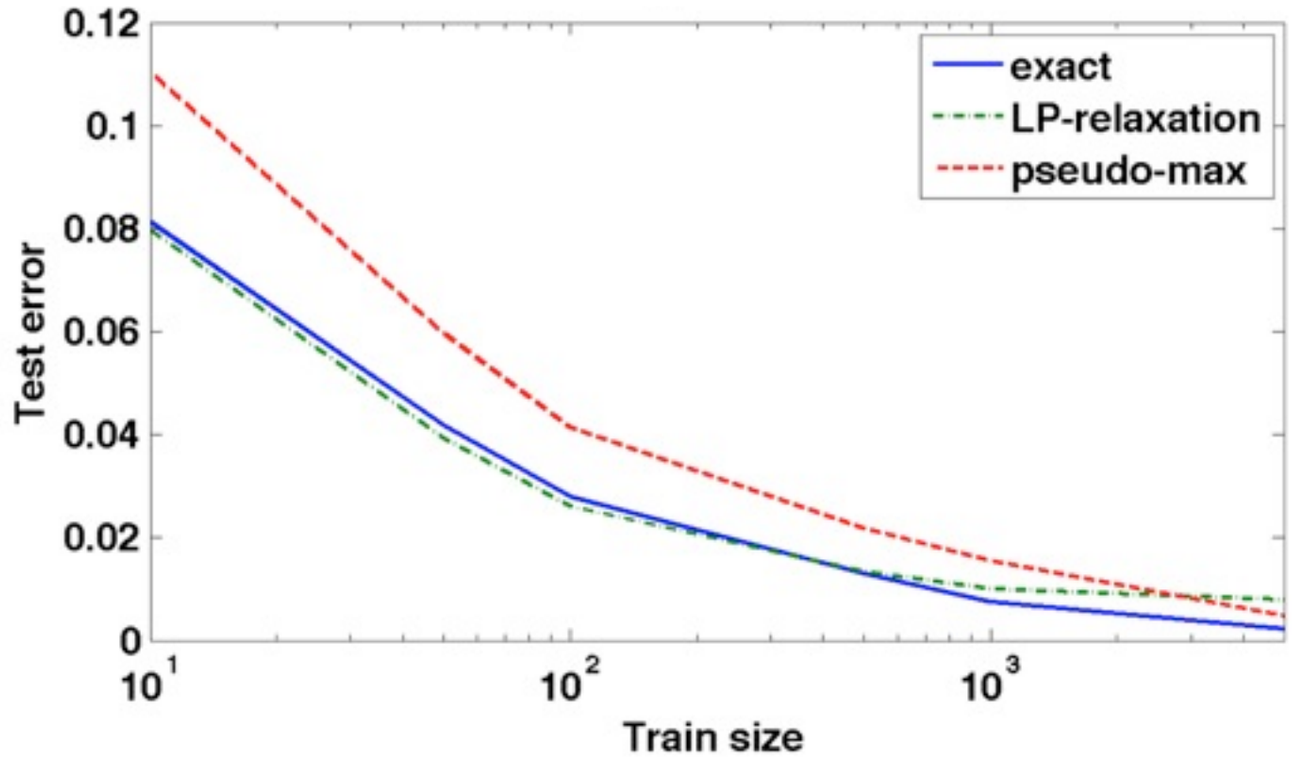
Experiments

- Toy problem. Ten binary variables.
- Pseudo-max asymptotically consistent.



Experiments

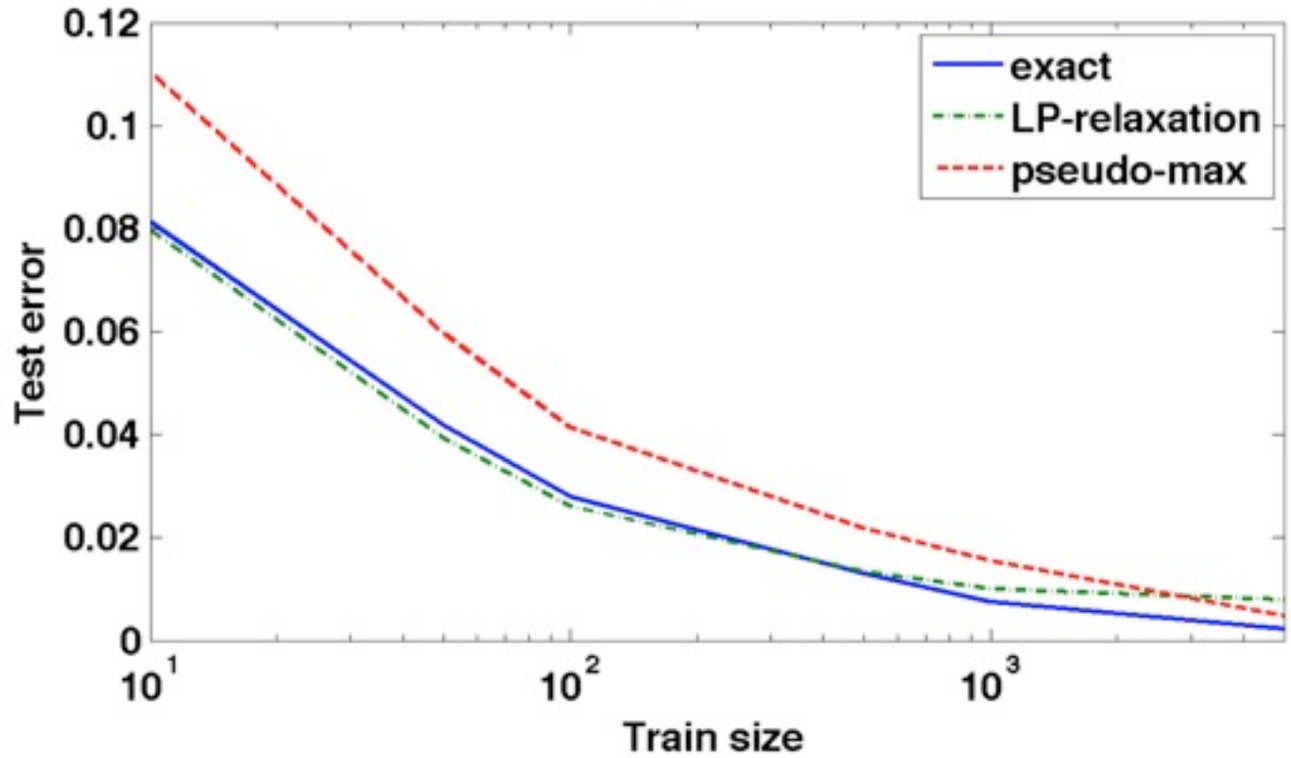
- Toy problem. Ten binary variables.
- Pseudo-max asymptotically consistent.



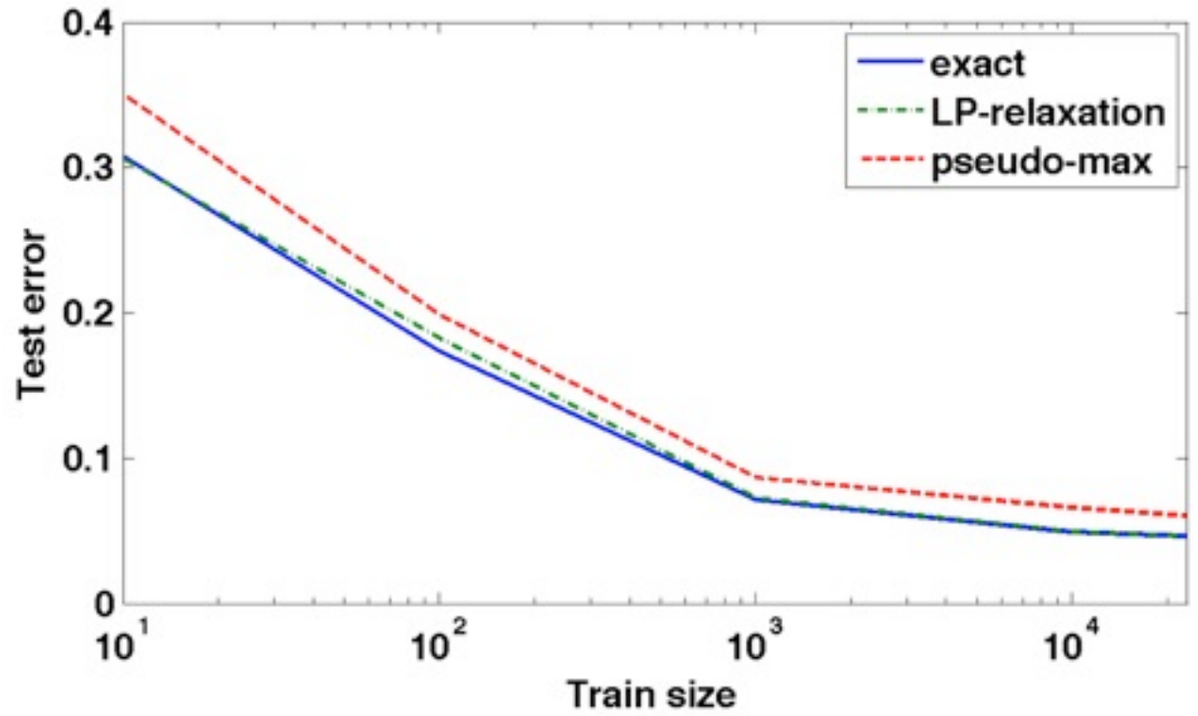
- Reuters with 10 labels.

Experiments

- Toy problem. Ten binary variables.
- Pseudo-max asymptotically consistent.

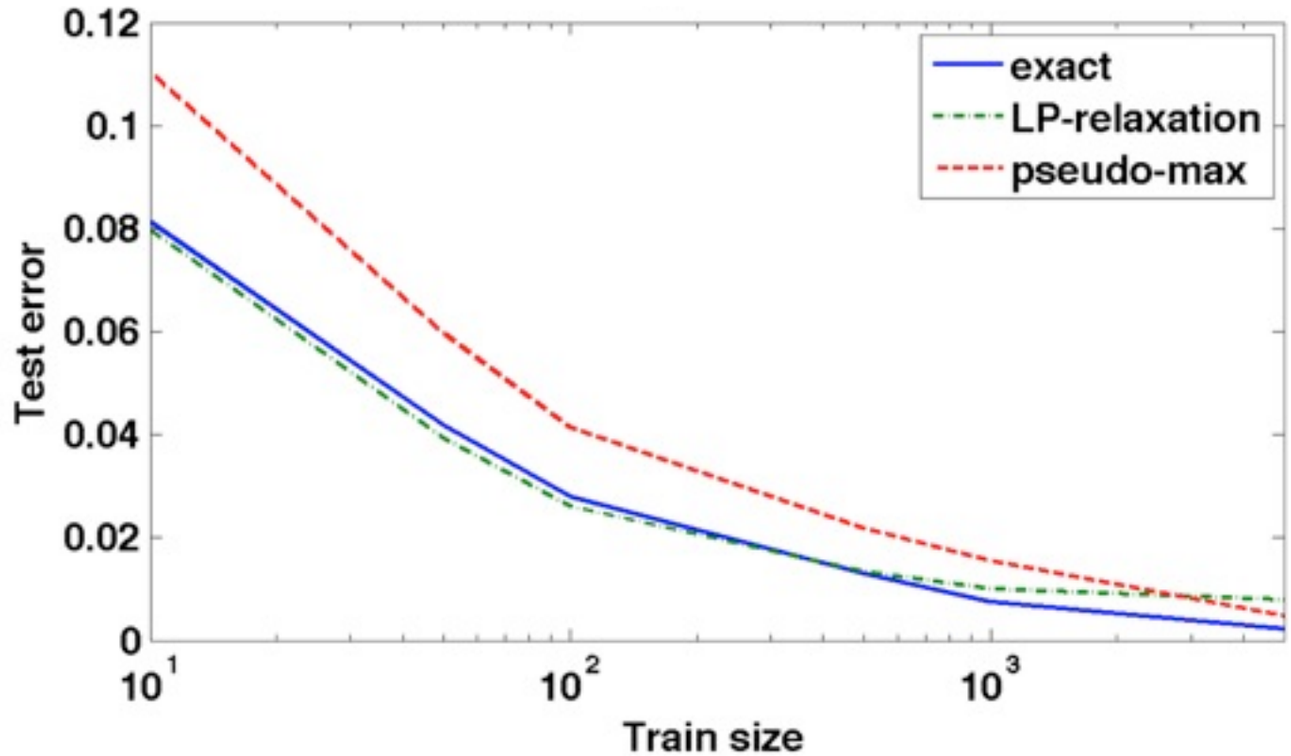


- Reuters with 10 labels.

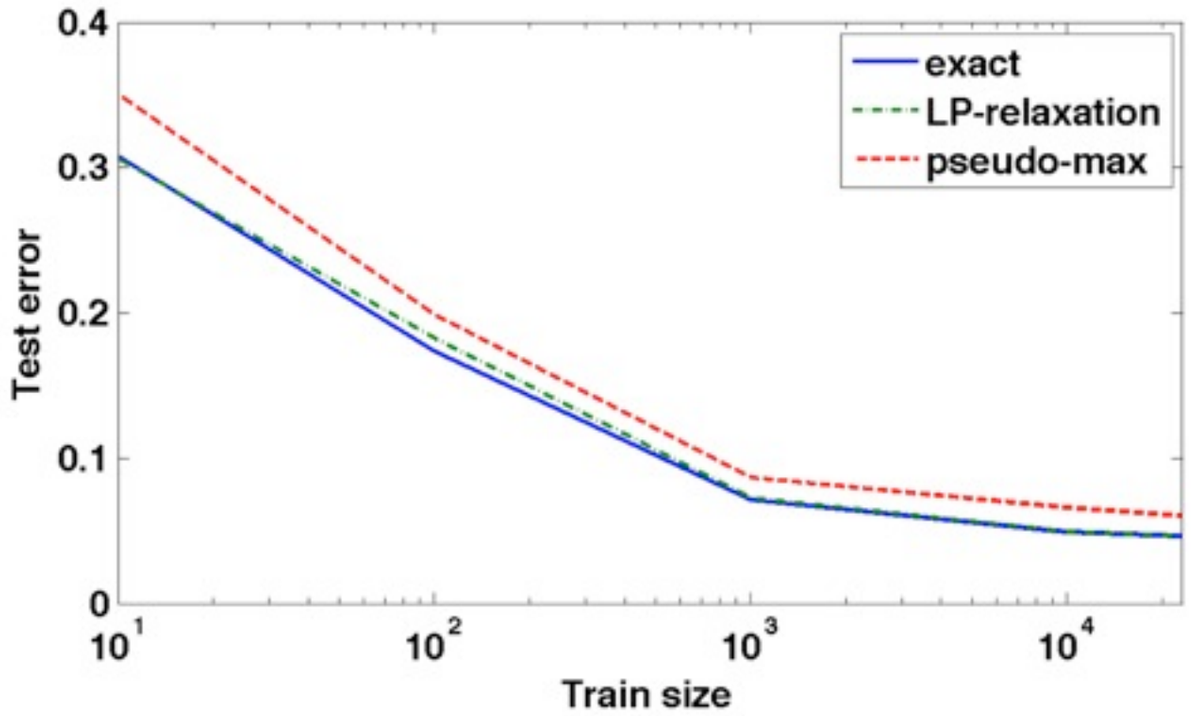


Experiments

- Toy problem. Ten binary variables.
- Pseudo-max asymptotically consistent.



- Reuters with 10 labels.
- Competitive with more costly methods.



Conclusions

Conclusions

- LP Relaxations are a powerful framework for approximating difficult combinatorial problems.

Conclusions

- LP Relaxations are a powerful framework for approximating difficult combinatorial problems.
- Allow us to introduce complex dependencies in settings like structured prediction.

Conclusions

- LP Relaxations are a powerful framework for approximating difficult combinatorial problems.
- Allow us to introduce complex dependencies in settings like structured prediction.
- Although generally approximate, tend to work well in practice.

Conclusions

- LP Relaxations are a powerful framework for approximating difficult combinatorial problems.
- Allow us to introduce complex dependencies in settings like structured prediction.
- Although generally approximate, tend to work well in practice.
- Have nice optimality certificates.

Conclusions

- LP Relaxations are a powerful framework for approximating difficult combinatorial problems.
- Allow us to introduce complex dependencies in settings like structured prediction.
- Although generally approximate, tend to work well in practice.
- Have nice optimality certificates.
- Can be progressively tightened.

Conclusions

- LP Relaxations are a powerful framework for approximating difficult combinatorial problems.
- Allow us to introduce complex dependencies in settings like structured prediction.
- Although generally approximate, tend to work well in practice.
- Have nice optimality certificates.
- Can be progressively tightened.
- Scalable algorithms for learning and inference.

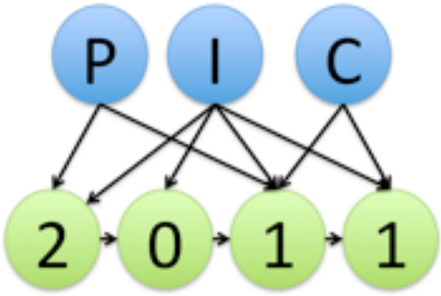
Conclusions

- LP Relaxations are a powerful framework for approximating difficult combinatorial problems.
- Allow us to introduce complex dependencies in settings like structured prediction.
- Although generally approximate, tend to work well in practice.
- Have nice optimality certificates.
- Can be progressively tightened.
- Scalable algorithms for learning and inference.
- Much work left: better algorithms and convergence analysis, better understanding of learning with approximations, why do they work well?

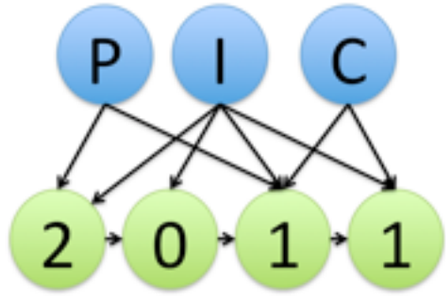
Conclusions

- LP Relaxations are a powerful framework for approximating difficult combinatorial problems.
- Allow us to introduce complex dependencies in settings like structured prediction.
- Although generally approximate, tend to work well in practice.
- Have nice optimality certificates.
- Can be progressively tightened.
- Scalable algorithms for learning and inference.
- Much work left: better algorithms and convergence analysis, better understanding of learning with approximations, why do they work well?
- Much potential for applications: holistic approaches to vision and language.

PASCAL Inference Challenge

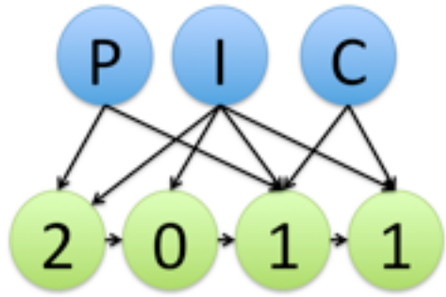


PASCAL Inference Challenge



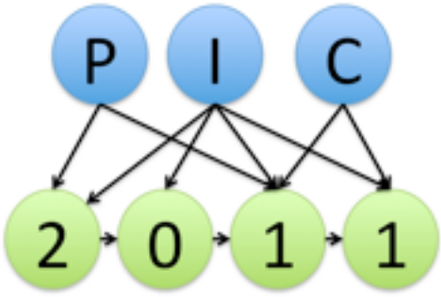
- We (with Gal Elidan) are organizing a challenge for testing inference algorithms.

PASCAL Inference Challenge



- We (with Gal Elidan) are organizing a challenge for testing inference algorithms.
- Sponsored by PASCAL.

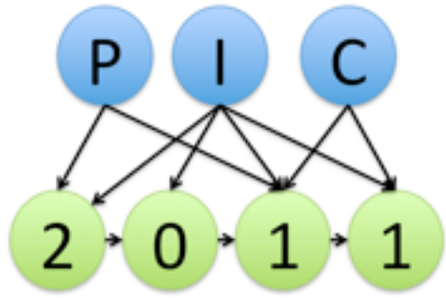
PASCAL Inference Challenge



- We (with Gal Elidan) are organizing a challenge for testing inference algorithms.
- Sponsored by PASCAL.

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

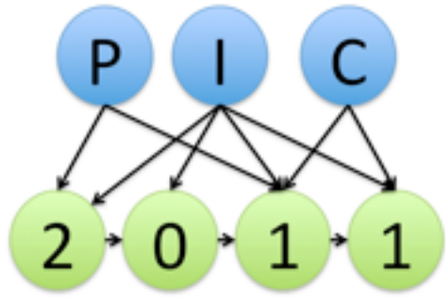
PASCAL Inference Challenge



- We (with Gal Elidan) are organizing a challenge for testing inference algorithms.
- Sponsored by PASCAL.
- Data from many domains.

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

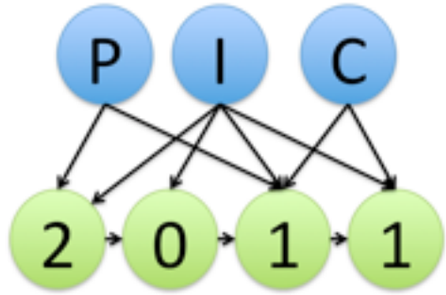
PASCAL Inference Challenge



- We (with Gal Elidan) are organizing a challenge for testing inference algorithms.
- Sponsored by PASCAL.
- Data from many domains.
- Compete on MAP, marginals and partition function.

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

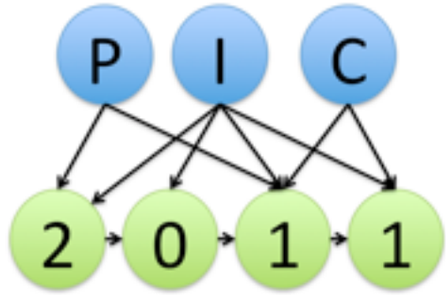
PASCAL Inference Challenge



- We (with Gal Elidan) are organizing a challenge for testing inference algorithms.
- Sponsored by PASCAL.
- Data from many domains.
- Compete on MAP, marginals and partition function.
- Speed and accuracy.

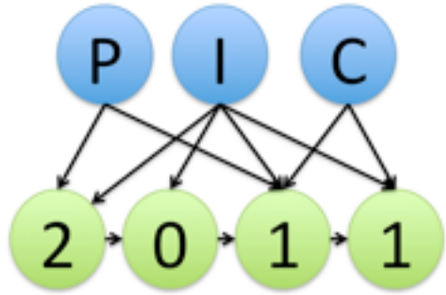
PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

PASCAL Inference Challenge



- We (with Gal Elidan) are organizing a challenge for testing inference algorithms.
- Sponsored by PASCAL.
- Data from many domains.
- Compete on MAP, marginals and partition function.
- Speed and accuracy.
- Modest prizes.

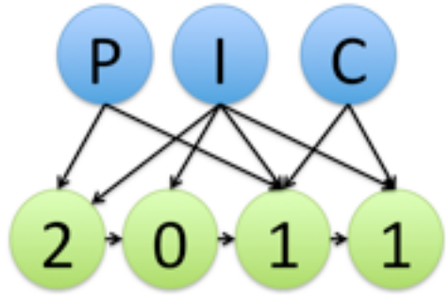
PASCAL Inference Challenge



- We (with Gal Elidan) are organizing a challenge for testing inference algorithms.
- Sponsored by PASCAL.
- Data from many domains.
- Compete on MAP, marginals and partition function.
- Speed and accuracy.
- Modest prizes.
- You are all invited to submit!

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

PASCAL Inference Challenge



- We (with Gal Elidan) are organizing a challenge for testing inference algorithms.
- Sponsored by PASCAL.
- Data from many domains.
- Compete on MAP, marginals and partition function.
- Speed and accuracy.
- Modest prizes.
- You are all invited to submit!
- Link: http://www.cs.huji.ac.il/project/PASCAL/