Learning with Inference for Discrete Graphical Models

Nikos Komodakis

Tutorial at ICCV
(Barcelona, Spain, November 2011)

Introduction

Conditional Random Fields (CRFs)

- Ubiquitous in computer vision
  - segmentation, stereo matching, optical flow, image restoration, image completion, object detection/localization, ...

Conditional Random Fields (CRFs)

- Ubiquitous in computer vision
  - segmentation / stereo matching / optical flow / image restoration / image completion / object detection/localization / ...
- and beyond
  - medical imaging, computer graphics, digital communications, physics…

Conditional Random Fields (CRFs)

- Ubiquitous in computer vision
  - segmentation
    optical flow
    image completion
    stereo matching
    image restoration
    object detection/localization
    ...
- and beyond
  - medical imaging, computer graphics, digital communications, physics…
- Really powerful formulation

Conditional Random Fields (CRFs)

- Key task: inference/optimization for CRFs/MRFs

Conditional Random Fields (CRFs)

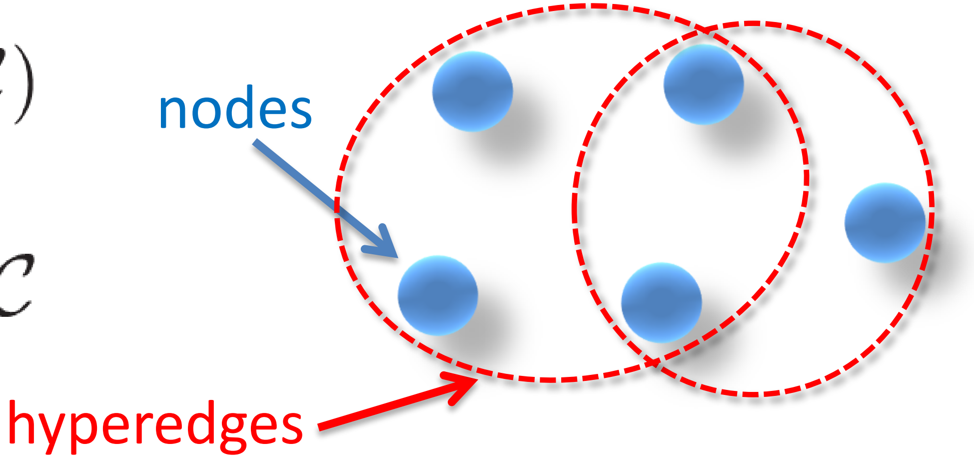
- Key task: inference/optimization for CRFs/MRFs
- Extensive research for more than 20 years

Conditional Random Fields (CRFs)

- Key task: inference/optimization for CRFs/MRFs
- Extensive research for more than 20 years
- Lots of progress
- Many state-of-the-art methods:
  - Graph-cut based algorithms
  - Message-passing methods
  - LP relaxations
  - Dual Decomposition
  - ….

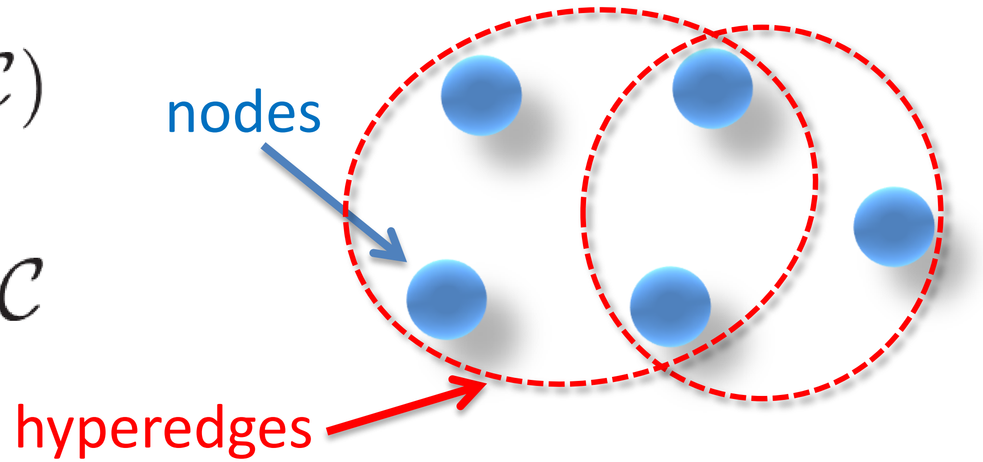
MAP inference for CRFs/MRFs

- Hypergraph $G = (\mathcal{V}, \mathcal{C})$
  - Nodes $\mathcal{V}$
  - Hyperedges/cliques $\mathcal{C}$



MAP inference for CRFs/MRFs

- Hypergraph $G = (\mathcal{V}, \mathcal{C})$
  - Nodes $\mathcal{V}$
  - Hyperedges/cliques $\mathcal{C}$

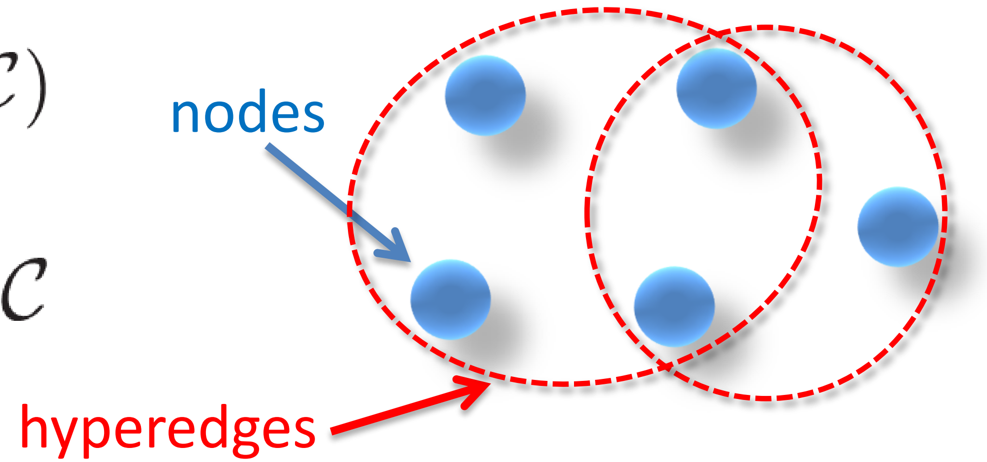


- High-order MRF energy minimization problem

$$\mathrm{MRF}_G(\mathbf{U}, \mathbf{H}) \equiv \min_{\mathbf{x}} \sum_{q \in \mathcal{V}} U_q(x_q) + \sum_{c \in \mathcal{C}} H_c(\mathbf{x}_c)$$

MAP inference for CRFs/MRFs

- Hypergraph $G = (\mathcal{V}, \mathcal{C})$
  - Nodes $\mathcal{V}$
  - Hyperedges/cliques $\mathcal{C}$

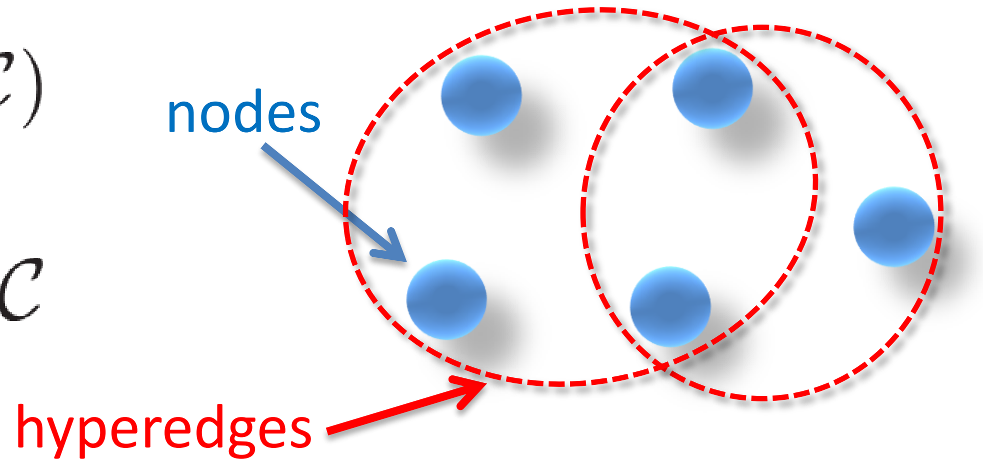


- High-order MRF energy minimization problem

$$\mathrm{MRF}_G(\mathbf{U}, \mathbf{H}) \equiv \min_{\mathbf{x}} \sum_{q \in \mathcal{V}} \underbrace{U_q(x_q)}_{\substack{\text{unary potential} \\ \text{(one per node)}}} + \sum_{c \in \mathcal{C}} H_c(\mathbf{x}_c)$$

MAP inference for CRFs/MRFs

- Hypergraph $G = (\mathcal{V}, \mathcal{C})$
  - Nodes $\mathcal{V}$
  - Hyperedges/cliques $\mathcal{C}$



- High-order MRF energy minimization problem

$$\mathrm{MRF}_G(\mathbf{U}, \mathbf{H}) \equiv \min_{\mathbf{x}} \sum_{q \in \mathcal{V}} \underbrace{U_q(x_q)}_{\substack{\text{unary potential} \\ \text{(one per node)}}} + \sum_{c \in \mathcal{C}} \underbrace{H_c(\mathbf{x}_c)}_{\substack{\text{high-order potential} \\ \text{(one per clique)}}}$$

CRF training

- But how do we choose the CRF potentials?

CRF training

- But how do we choose the CRF potentials?
- Through training
  - Parameterize potentials by **w**
  - Use training data to learn correct **w**

CRF training

- But how do we choose the CRF potentials?
- Through training
  - Parameterize potentials by **w**
  - Use training data to learn correct **w**
- Characteristic example of structured output learning [Taskar], [Tsochantaridis, Joachims]

CRF training

- Equally, if not more, important than MAP inference
  - Better optimize correct energy (even approximately)
  - Than optimize wrong energy exactly

CRF training

- Equally, if not more, important than MAP inference
  - Better optimize correct energy (even approximately)
  - Than optimize wrong energy exactly
- Becomes even more important as we move towards:
  - complex models
  - high-order potentials
  - lots of parameters
  - lots of training data

CRF training

$$f : Z \rightarrow X$$

CRF training

input
space

$$f : Z \longrightarrow X$$

can contain any
kind of data

CRF training

input space

output space

$$f : Z \rightarrow X$$

can contain any kind of data

CRF variables (structured object)

CRF training

input space → output space

$$f : Z \rightarrow X$$

can contain any kind of data

CRF variables (structured object)

Hereafter, we will use:

- symbol z to denote elements of space Z
- symbol x to denote elements of space X

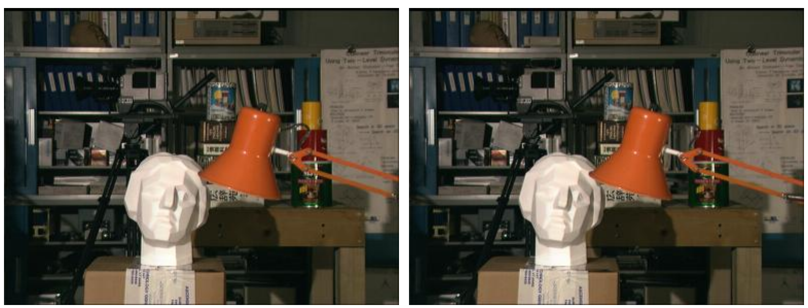
CRF training

- Stereo matching:
  - Z: left, right image
  - X: disparity map

CRF training

- Stereo matching:
  - Z: left, right image
  - X: disparity map

$f:$

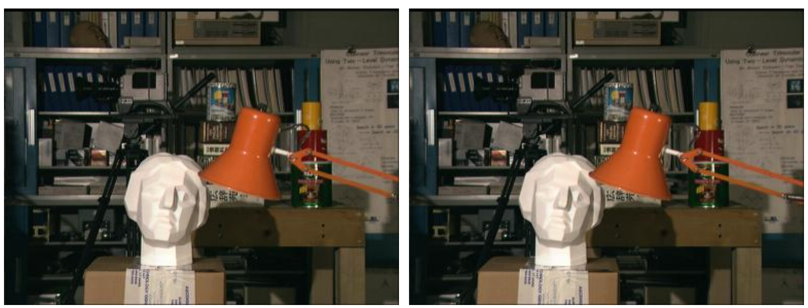


Z

X

CRF training

- Stereo matching:
  - Z: left, right image
  - X: disparity map

$f:$ 

Z

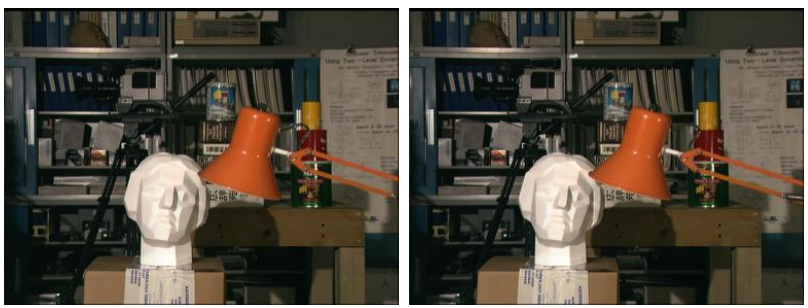
X

$$f = \arg\min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \mathbf{u}, \mathbf{h})$$

CRF training

- Stereo matching:
  - Z: left, right image
  - X: disparity map

$f:$



Z

X

$$f = \arg\min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \mathbf{u}, \mathbf{h})$$

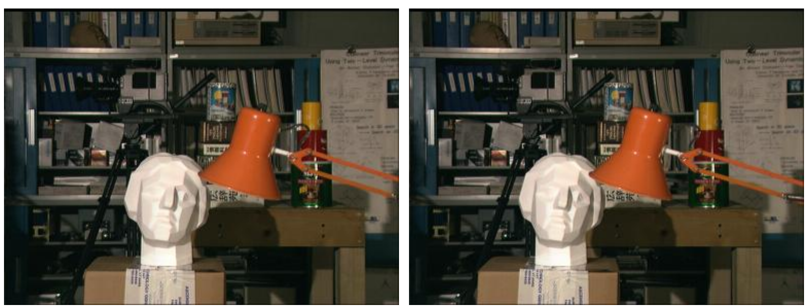
parameterized by **w**

CRF training

- Stereo matching:
  - Z: left, right image
  - X: disparity map

Goal of training:
estimate proper **w**

$f:$



Z

X

$$f = \arg\min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \mathbf{u}, \mathbf{h})$$

parameterized by **w**

CRF training

- Denoising:
  - Z: noisy input image
  - X: denoised output image

Goal of training:
estimate proper **w**

f :

Z

X

$$f = \arg\min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \mathbf{u}, \mathbf{h})$$

parameterized by **w**

CRF training

- Object detection:
  - Z: input image
  - X: position of object parts

Goal of training:
estimate proper **w**

f :

Z

X

$$f = \arg\min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \mathbf{u}, \mathbf{h})$$

parameterized by **w**

CRF training (some further notation)

CRF training (some further notation)

$$\mathrm{MRF}_G(\mathbf{x}; \mathbf{u}^k, \mathbf{h}^k) = \sum_p u_p^k(x_p) + \sum_c h_c^k(\mathbf{x}_c)$$

CRF training (some further notation)

$$\mathrm{MRF}_G(\mathbf{x};\mathbf{u}^k,\mathbf{h}^k) = \sum_p u_p^k(x_p) + \sum_c h_c^k(\mathbf{x}_c)$$

$$u_p^k(x_p) = \mathbf{w}^T g_p(x_p,\mathbf{z}^k),\;\; h_c^k(\mathbf{x}_c) = \mathbf{w}^T g_c(\mathbf{x}_c,\mathbf{z}^k)$$

CRF training (some further notation)

$$\mathrm{MRF}_G(\mathbf{x};\mathbf{u}^k,\mathbf{h}^k) = \sum_p u_p^k(x_p) + \sum_c h_c^k(\mathbf{x}_c)$$

$$u_p^k(x_p) = \mathbf{w}^T g_p(x_p,\mathbf{z}^k), \;\; h_c^k(\mathbf{x}_c) = \mathbf{w}^T g_c(\mathbf{x}_c,\mathbf{z}^k)$$

vector valued feature functions

CRF training (some further notation)

$$\mathrm{MRF}_G(\mathbf{x};\mathbf{u}^k,\mathbf{h}^k) = \sum_p u_p^k(x_p) + \sum_c h_c^k(\mathbf{x}_c)$$

$$u_p^k(x_p) = \mathbf{w}^T g_p(x_p, \mathbf{z}^k), \;\; h_c^k(\mathbf{x}_c) = \mathbf{w}^T g_c(\mathbf{x}_c, \mathbf{z}^k)$$

vector valued feature functions

$$\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k) = \mathbf{w}^T \left(\sum_p g_p(x_p, \mathbf{z}^k) + \sum_c g_c(\mathbf{x}_c, \mathbf{z}^k) \right) = \mathbf{w}^T g(\mathbf{x}, \mathbf{z}^k)$$

Learning formulations

Risk minimization

K training samples $\left\{(\mathbf{x}^k, \mathbf{z}^k)\right\}_{k=1}^{K}$

Risk minimization

$$\min_{\mathbf{w}} \sum_{k=1}^{K} \Delta\left(\mathbf{x}^k, \hat{\mathbf{x}}^k\right)$$

K training samples $\left\{\left(\mathbf{x}^k, \mathbf{z}^k\right)\right\}_{k=1}^{K}$

Risk minimization

$$\hat{\mathbf{x}}^k = \arg\min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k)$$

$$\min_{\mathbf{w}} \sum_{k=1}^{K} \Delta\left(\mathbf{x}^k, \hat{\mathbf{x}}^k\right)$$

K training samples $\left\{(\mathbf{x}^k, \mathbf{z}^k)\right\}_{k=1}^{K}$

Regularized Risk minimization

$$\hat{\mathbf{x}}^k = \arg\min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k)$$

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} \Delta\left(\mathbf{x}^k, \hat{\mathbf{x}}^k\right)$$

$$R(\mathbf{w}) = ||\mathbf{w}||^2, \;\; ||\mathbf{w}||_1, \;\; \text{etc.}$$

Regularized Risk minimization

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} \Delta\left(\mathbf{x}^k, \hat{\mathbf{x}}^k\right)$$

Regularized Risk minimization

Replace Δ(.) with easier to handle upper bound L_G
(e.g., convex w.r.t. **w**)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} \Delta\left(\mathbf{x}^k, \hat{\mathbf{x}}^k\right)$$

Regularized Risk minimization

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

Replace Δ(.) with easier to handle upper bound L_G (e.g., convex w.r.t. **w**)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} \Delta\left(\mathbf{x}^k, \hat{\mathbf{x}}^k\right)$$

Choice 1: Hinge loss

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$\boxed{L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}}\left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)}$$

Choice 1: Hinge loss

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$\boxed{L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}}\left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)}$$

- Upper bounds Δ(.)

Choice 1: Hinge loss

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$\boxed{L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)}$$

- Upper bounds Δ(.)
- Leads to max-margin learning

Choice 1: Hinge loss

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$\boxed{L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)}$$

- Upper bounds Δ(.)
- Leads to **max-margin learning**

Max-margin learning

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)$$

Max-margin learning

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)$$

energy of
ground truth

Max-margin learning

$$\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k) - \Delta(\mathbf{x},\mathbf{x}^k)$$

energy of ground truth

any other energy

Max-margin learning

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)$$

energy of ground truth

any other energy

desired margin

Max-margin learning

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

energy of ground truth	any other energy	desired margin	slack

Max-margin learning

$$\min_{\mathbf{w}} \qquad \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k) \le \mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k) - \Delta(\mathbf{x},\mathbf{x}^k) + \xi_k$$

energy of ground truth	any other energy	desired margin	slack

Max-margin learning

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_k \xi_k$$

subject to the constraints:

$$\underbrace{\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k)}_{\text{energy of ground truth}} \leq \underbrace{\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k)}_{\text{any other energy}} - \underbrace{\Delta(\mathbf{x}, \mathbf{x}^k)}_{\text{desired margin}} + \underbrace{\xi_k}_{\text{slack}}$$

Max-margin learning

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

Max-margin learning

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

or equivalently

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_k \xi_k$$

$$\xi_k = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

Max-margin learning

CONSTRAINED

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

or equivalently

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_k \xi_k$$

$$\xi_k = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

Max-margin learning

CONSTRAINED

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

or equivalently

UNCONSTRAINED

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_k \xi_k$$

$$\xi_k = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) \right)$$

Choice 2: logistic loss

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) + \underbrace{\log \sum_{\mathbf{x}} e^{-\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k)}}_{\text{partition function}}$$

Choice 2: logistic loss

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) + \underbrace{\log \sum_{\mathbf{x}} e^{-\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k)}}_{\text{partition function}}$$

- Can be shown to lead to **maximum likelihood learning**

Max-margin vs Maximum-likelihood

$$L_G\left(\mathbf{x}^k,\mathbf{z}^k;\mathbf{w}\right) = \overbrace{\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k) - \min_{\mathbf{x}}\left(\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k) - \Delta(\mathbf{x},\mathbf{x}^k)\right)}^{\text{max-margin}}$$

$$L_G\left(\mathbf{x}^k,\mathbf{z}^k;\mathbf{w}\right) = \underbrace{\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k) + \log\sum_{\mathbf{x}} e^{-\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k)}}_{\text{maximum likelihood}}$$

Max-margin vs Maximum-likelihood

$$L_G\left(\mathbf{x}^k,\mathbf{z}^k;\mathbf{w}\right) = \overbrace{\boxed{\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k)} - \min_{\mathbf{x}}\left(\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k) - \Delta(\mathbf{x},\mathbf{x}^k)\right)}^{\text{max-margin}}$$

$$L_G\left(\mathbf{x}^k,\mathbf{z}^k;\mathbf{w}\right) = \underbrace{\boxed{\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k)} + \log\sum_{\mathbf{x}} e^{-\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k)}}_{\text{maximum likelihood}}$$

Max-margin vs Maximum-likelihood

$$L_G\left(\mathbf{x}^k,\mathbf{z}^k;\mathbf{w}\right) = \overbrace{\boxed{\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k)}\boxed{-\min_{\mathbf{x}}}\left(\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k) - \Delta(\mathbf{x},\mathbf{x}^k)\right)}^{\text{max-margin}}$$

$$L_G\left(\mathbf{x}^k,\mathbf{z}^k;\mathbf{w}\right) = \underbrace{\boxed{\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k)}+\boxed{\log\sum_{\mathbf{x}} e}^{-\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k)}}_{\text{maximum likelihood}}$$

Max-margin vs Maximum-likelihood

$$L_G\left(\mathbf{x}^k,\mathbf{z}^k;\mathbf{w}\right) = \overbrace{\boxed{\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k)} + \boxed{\max_{\mathbf{x}}}\left(-\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k) + \Delta(\mathbf{x},\mathbf{x}^k)\right)}^{\text{max-margin}}$$

$$L_G\left(\mathbf{x}^k,\mathbf{z}^k;\mathbf{w}\right) = \underbrace{\boxed{\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k)} + \boxed{\log\sum_{\mathbf{x}} e}^{-\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k)}}_{\text{maximum likelihood}}$$

Max-margin vs Maximum-likelihood

$$L_G\left(\mathbf{x}^k,\mathbf{z}^k;\mathbf{w}\right) = \overbrace{\boxed{\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k)} + \boxed{\max_{\mathbf{x}}}\left(-\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k) + \Delta(\mathbf{x},\mathbf{x}^k)\right)}^{\text{max-margin}}$$

soft-max

$$L_G\left(\mathbf{x}^k,\mathbf{z}^k;\mathbf{w}\right) = \underbrace{\boxed{\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k)} + \boxed{\log\sum_{\mathbf{x}}} e^{-\mathrm{MRF}_G(\mathbf{x};\mathbf{w},\mathbf{z}^k)}}_{\text{maximum likelihood}}$$

Solving the learning formulations

Maximum-likelihood learning

Maximum-likelihood learning

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) + \underbrace{\log \sum_{\mathbf{x}} e^{-\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k)}}_{\text{partition function}}$$

Maximum-likelihood learning

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) + \underbrace{\log \sum_{\mathbf{x}} e^{-\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k)}}_{\text{partition function}}$$

- Differentiable & convex

Maximum-likelihood learning

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) + \underbrace{\log \sum_{\mathbf{x}} e^{-\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k)}}_{\text{partition function}}$$

- Differentiable & convex
- Global optimum via e.g. gradient descent

Maximum-likelihood learning

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) + \log \sum_{\mathbf{x}} e^{-\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k)}$$

gradient $\longrightarrow \nabla_{\mathbf{w}} = \mathbf{w} + \sum_{k} \left(g(\mathbf{x}^k, \mathbf{z}^k) - \sum_{\mathbf{x}} p(\mathbf{x}|w, \mathbf{z}^k) g(\mathbf{x}, \mathbf{z}^k) \right)$

Recall that: $\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) = \mathbf{w}^T g(\mathbf{x}, \mathbf{z}^k)$

Maximum-likelihood learning

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) + \log \sum_{\mathbf{x}} e^{-\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k)}$$

gradient $\longrightarrow \nabla_{\mathbf{w}} = \mathbf{w} + \sum_k \left(g(\mathbf{x}^k, \mathbf{z}^k) - \sum_{\mathbf{x}} p(\mathbf{x}|w, \mathbf{z}^k) g(\mathbf{x}, \mathbf{z}^k) \right)$

- Requires MRF probabilistic inference

Maximum-likelihood learning

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) + \log \sum_{\mathbf{x}} e^{-\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k)}$$

$$\text{gradient} \longrightarrow \nabla_{\mathbf{w}} = \mathbf{w} + \sum_k \left(g(\mathbf{x}^k, \mathbf{z}^k) - \underbrace{\sum_{\mathbf{x}} p(\mathbf{x}|w, \mathbf{z}^k) g(\mathbf{x}, \mathbf{z}^k)}_{} \right)$$

- Requires MRF probabilistic inference
- **NP-hard** (exponentially many $\mathbf{x}$): approximation via loopy-BP ???

Max-margin learning (UNCONSTRAINED)

Max-margin learning (UNCONSTRAINED)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}}\left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

Max-margin learning (UNCONSTRAINED)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

- Convex but non-differentiable

Max-margin learning (UNCONSTRAINED)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

- Convex but non-differentiable
- Global optimum via subgradient method

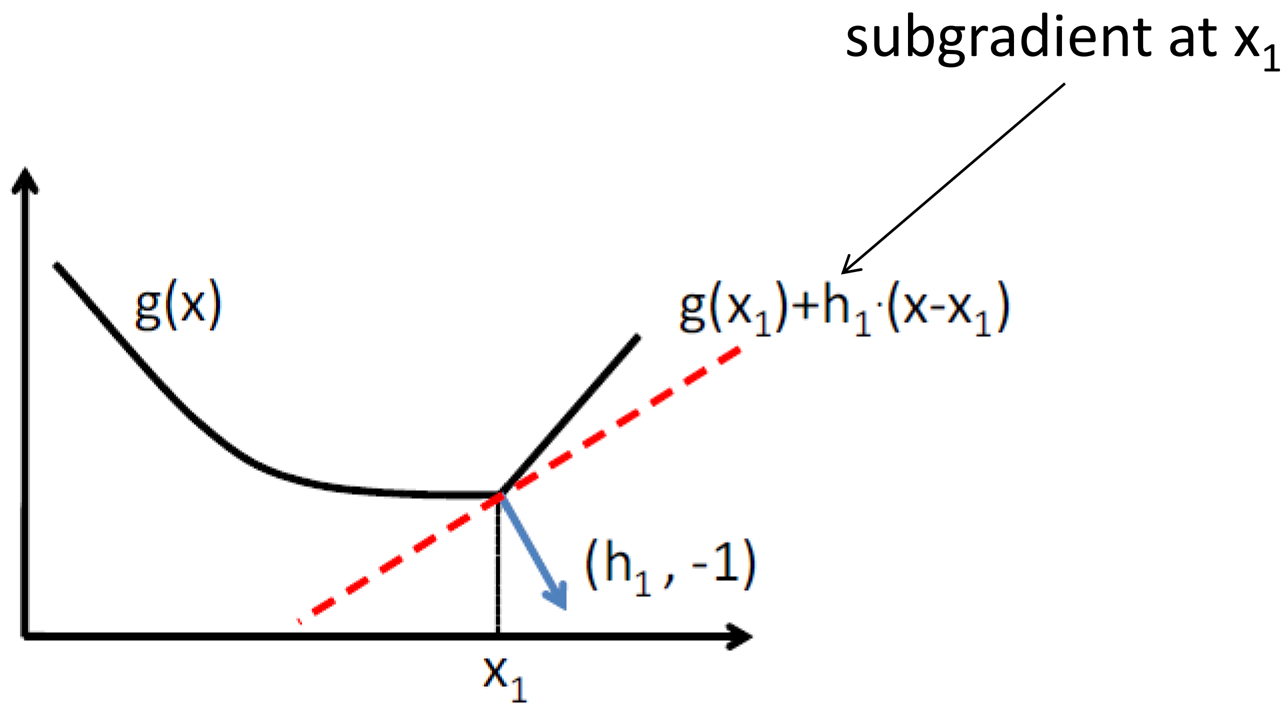
Max-margin learning (UNCONSTRAINED)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

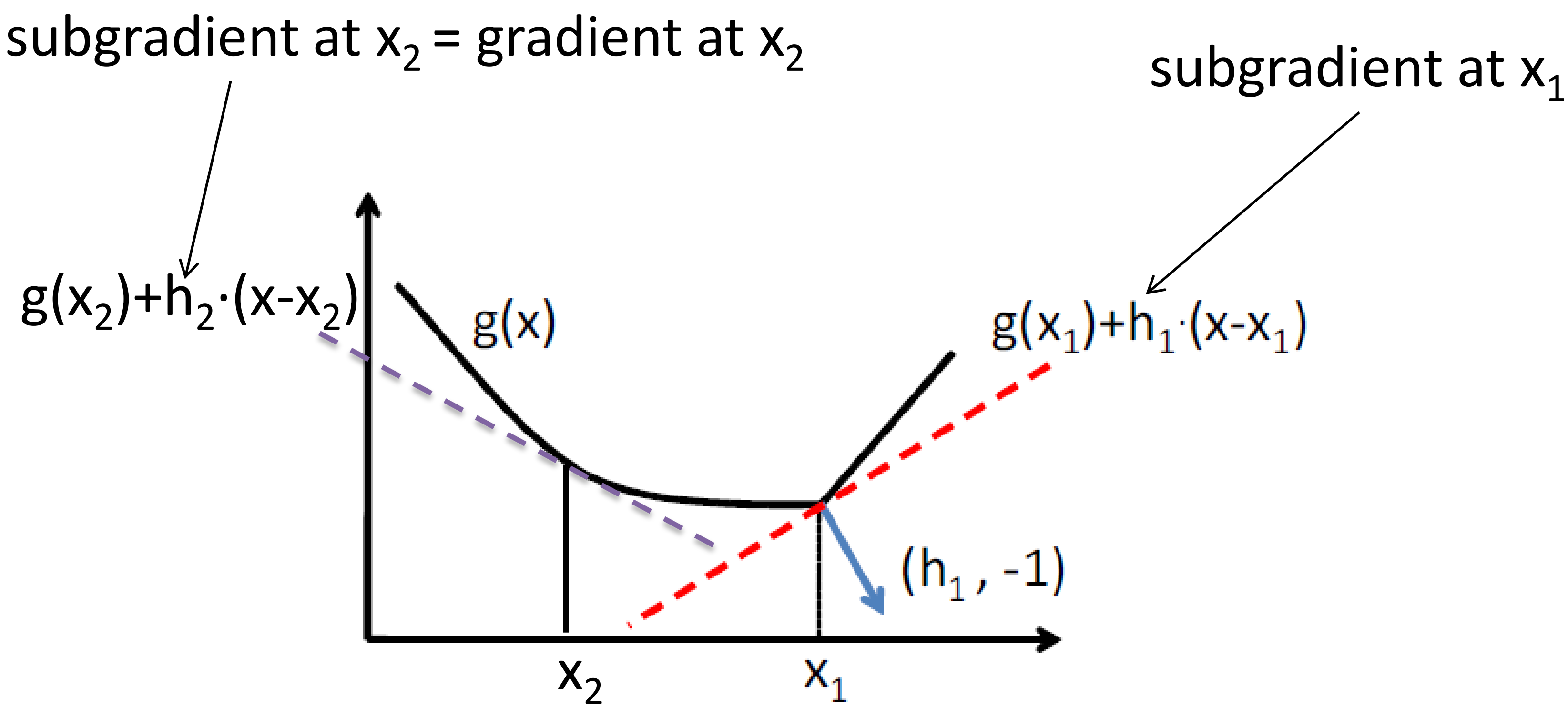
$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}}\left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

- Convex but non-differentiable
- Global optimum via **subgradient method**

Subgradient



Subgradient

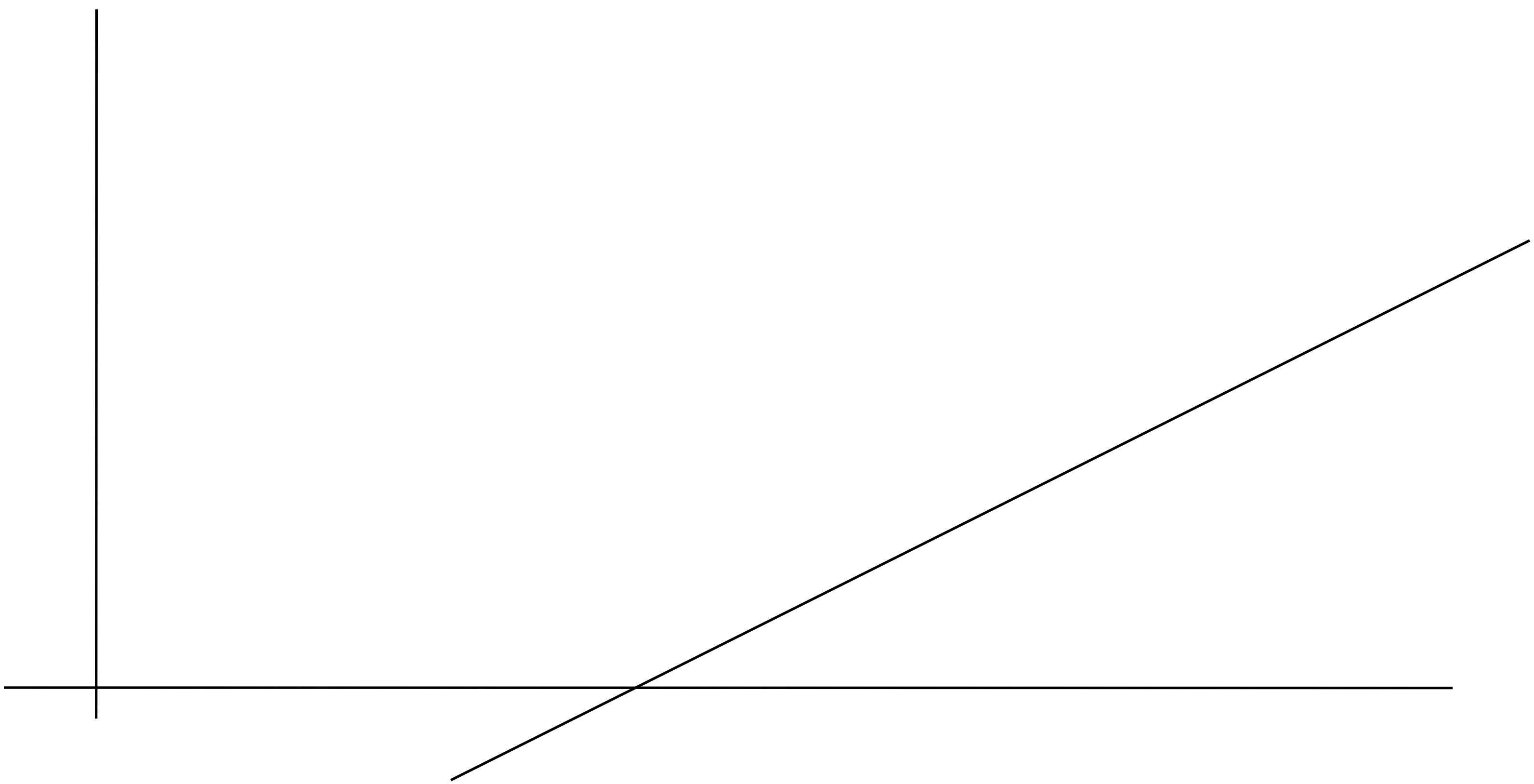


Subgradient

Lemma. *Let* $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, *with* $f_m(\cdot)$ *convex and differentiable. A subgradient of* f *at* $\mathbf{y}$ *is given by* $\nabla f_{\hat{m}}(\mathbf{y})$, *where* $\hat{m}$ *is any index for which* $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.

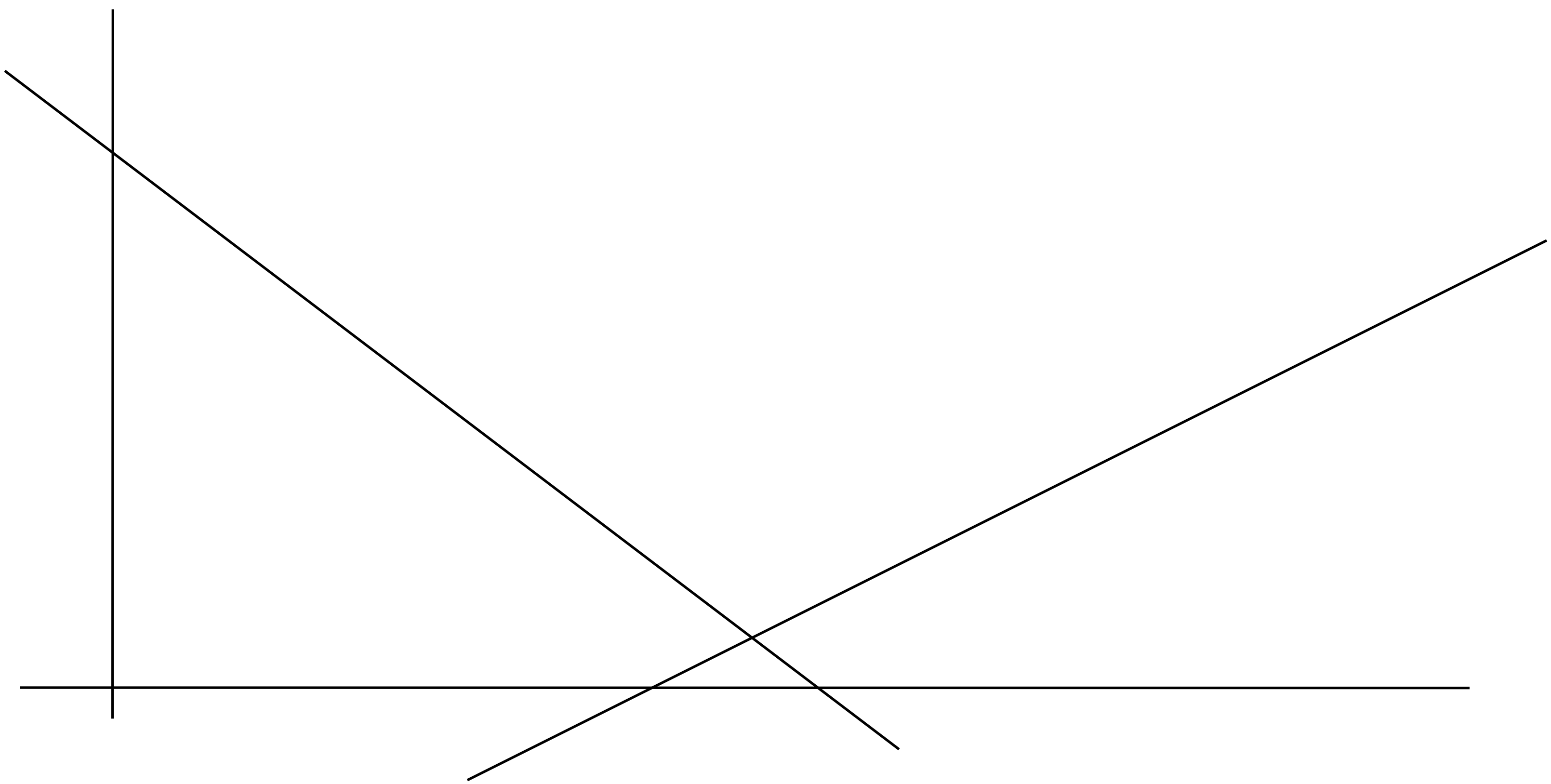
Subgradient

Lemma. *Let $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, with $f_m(\cdot)$ convex and differentiable. A subgradient of f at $\mathbf{y}$ is given by $\nabla f_{\hat{m}}(\mathbf{y})$, where $\hat{m}$ is any index for which $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.*



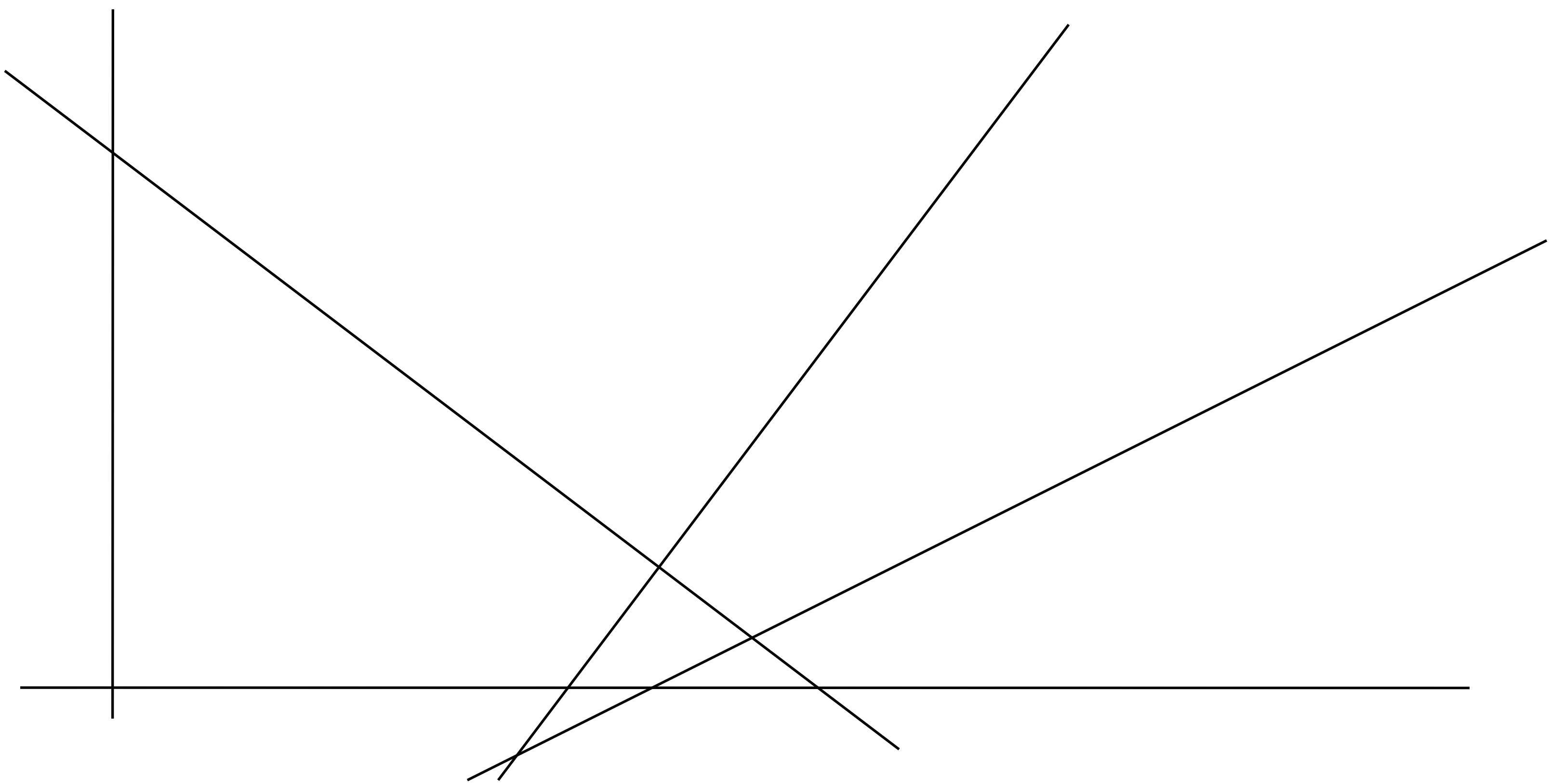
Subgradient

Lemma. *Let* $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, *with* $f_m(\cdot)$ *convex and differentiable. A subgradient of* f *at* $\mathbf{y}$ *is given by* $\nabla f_{\hat{m}}(\mathbf{y})$, *where* $\hat{m}$ *is any index for which* $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.



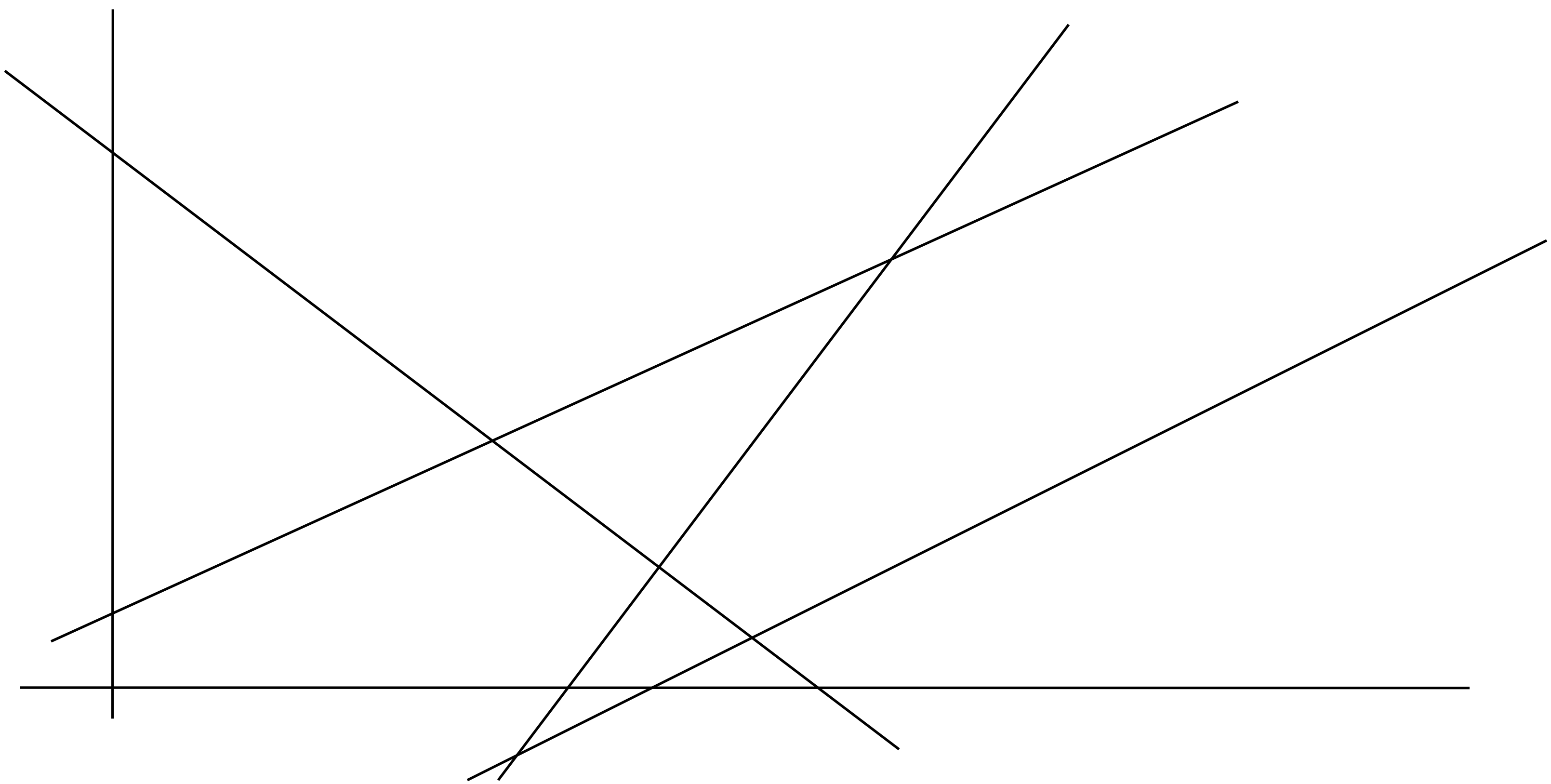
Subgradient

Lemma. *Let* $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, *with* $f_m(\cdot)$ *convex and differentiable. A subgradient of* f *at* $\mathbf{y}$ *is given by* $\nabla f_{\hat{m}}(\mathbf{y})$, *where* $\hat{m}$ *is any index for which* $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.



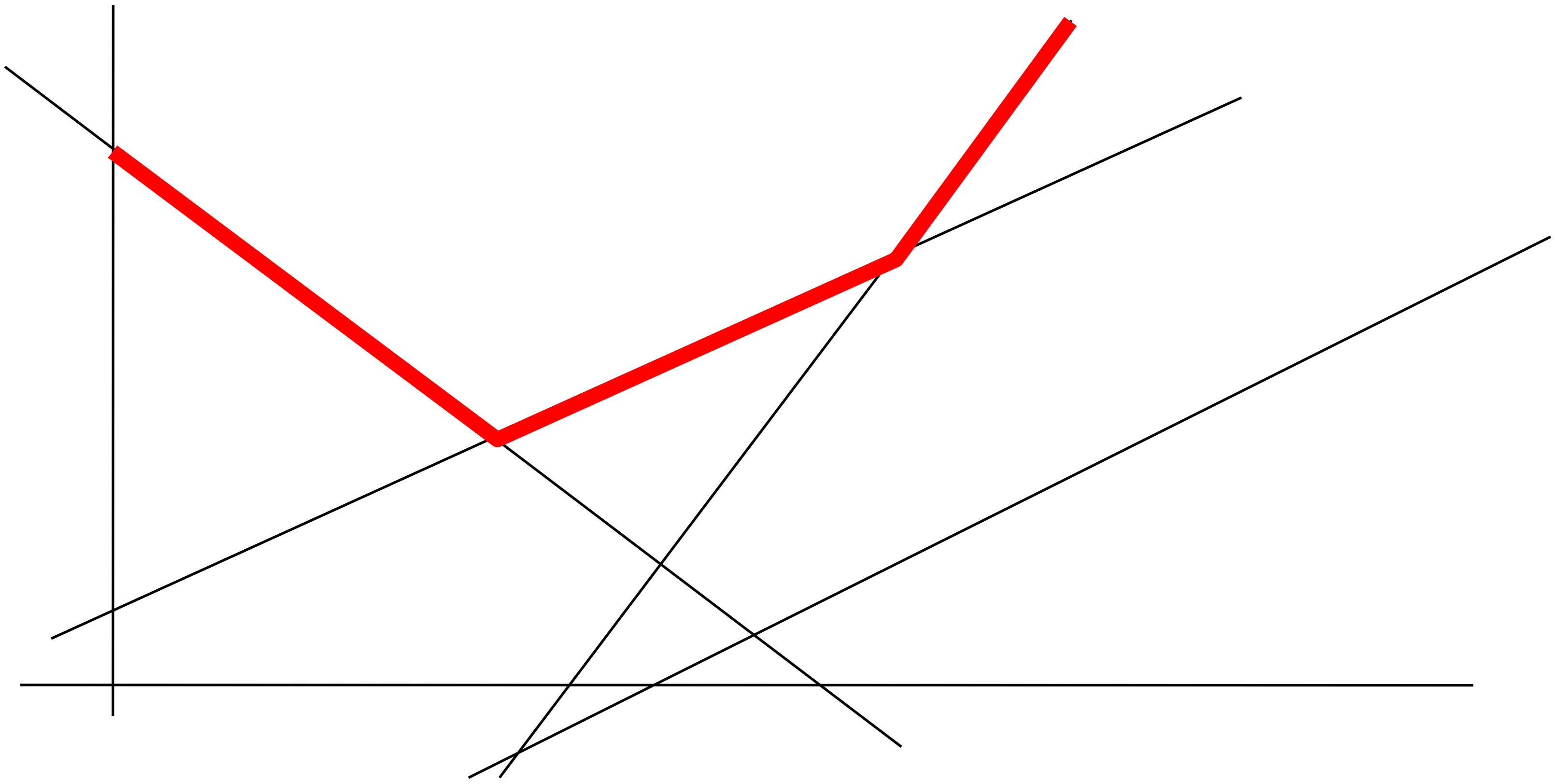
Subgradient

Lemma. *Let* $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, *with* $f_m(\cdot)$ *convex and differentiable. A subgradient of* f *at* $\mathbf{y}$ *is given by* $\nabla f_{\hat{m}}(\mathbf{y})$, *where* $\hat{m}$ *is any index for which* $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.



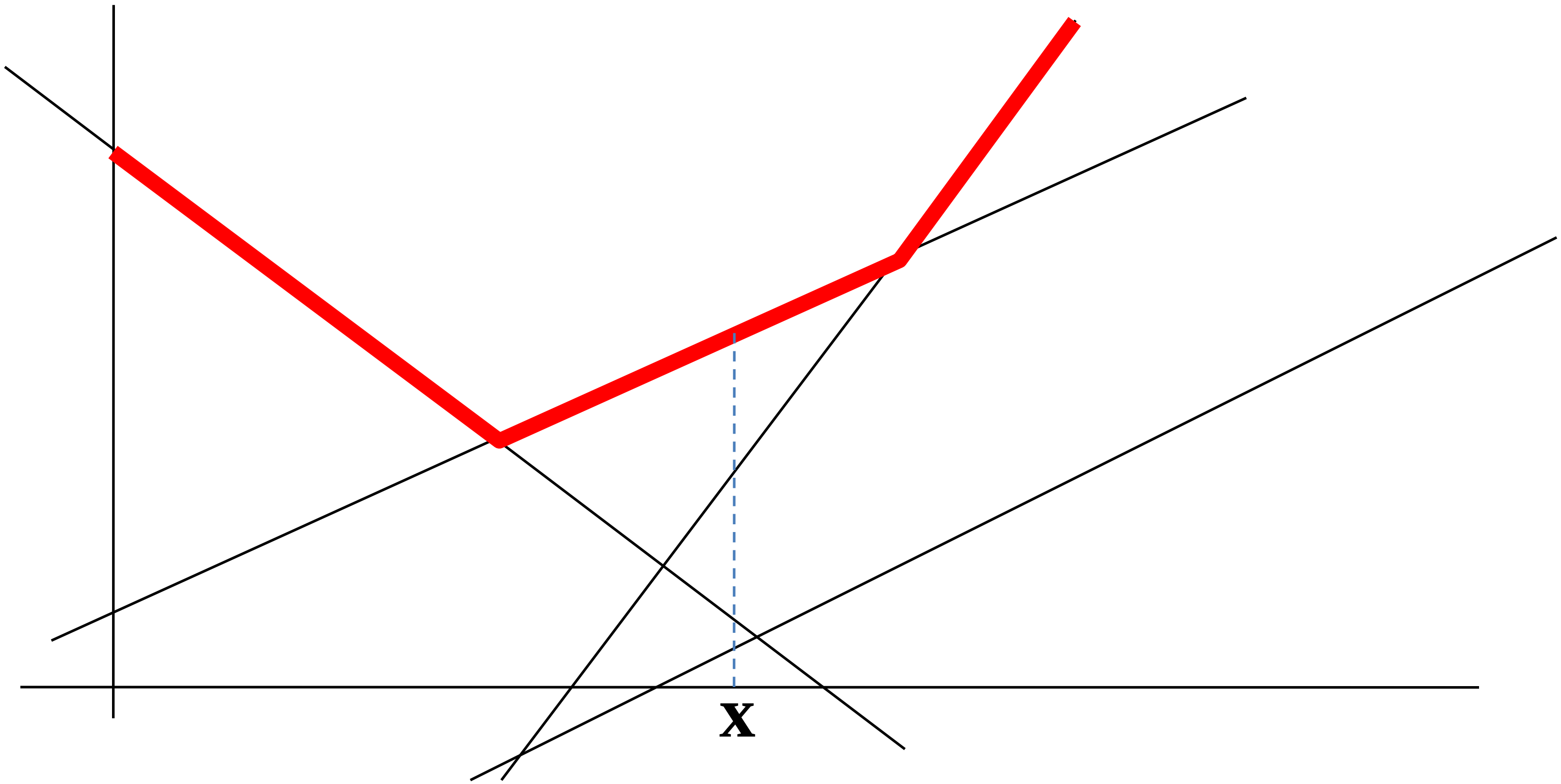
Subgradient

Lemma. *Let* $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, *with* $f_m(\cdot)$ *convex and differentiable. A subgradient of* f *at* $\mathbf{y}$ *is given by* $\nabla f_{\hat{m}}(\mathbf{y})$, *where* $\hat{m}$ *is any index for which* $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.



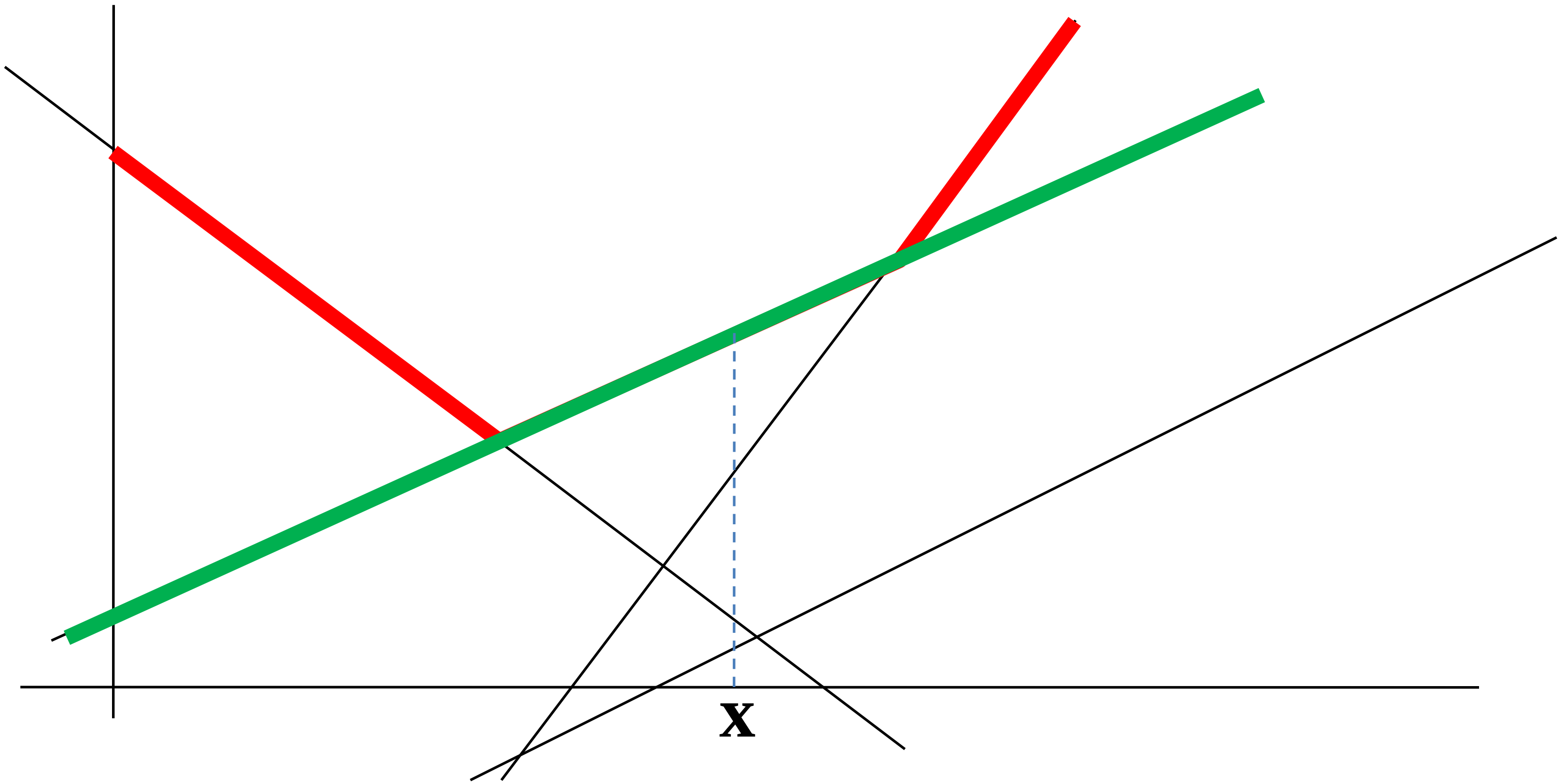
Subgradient

Lemma. *Let* $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, *with* $f_m(\cdot)$ *convex and differentiable. A subgradient of* f *at* $\mathbf{y}$ *is given by* $\nabla f_{\hat{m}}(\mathbf{y})$, *where* $\hat{m}$ *is any index for which* $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.



Subgradient

Lemma. *Let* $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, *with* $f_m(\cdot)$ *convex and differentiable. A subgradient of* f *at* $\mathbf{y}$ *is given by* $\nabla f_{\hat{m}}(\mathbf{y})$, *where* $\hat{m}$ *is any index for which* $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.



Subgradient

Lemma. *Let* $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, *with* $f_m(\cdot)$ *convex and differentiable. A subgradient of* f *at* $\mathbf{y}$ *is given by* $\nabla f_{\hat{m}}(\mathbf{y})$, *where* $\hat{m}$ *is any index for which* $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

Subgradient

Lemma. *Let* $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, *with* $f_m(\cdot)$ *convex and differentiable. A subgradient of* f *at* $\mathbf{y}$ *is given by* $\nabla f_{\hat{m}}(\mathbf{y})$, *where* $\hat{m}$ *is any index for which* $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

$$\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) = \mathbf{w}^T g(\mathbf{x}, \mathbf{z}^k)$$

Subgradient

Lemma. *Let* $f(\cdot) = \max_{m=1,\ldots,M} f_m(\cdot)$, *with* $f_m(\cdot)$ *convex and differentiable. A subgradient of* f *at* $\mathbf{y}$ *is given by* $\nabla f_{\hat{m}}(\mathbf{y})$, *where* $\hat{m}$ *is any index for which* $f(\mathbf{y}) = f_{\hat{m}}(\mathbf{y})$.

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

$$\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) = \mathbf{w}^T g(\mathbf{x}, \mathbf{z}^k)$$

subgradient of L_G = $g(\mathbf{x}^k, \mathbf{z}^k) - g(\hat{\mathbf{x}}^k, \mathbf{z}^k)$

$$\hat{\mathbf{x}}^k = \arg\min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

Max-margin learning (UNCONSTRAINED)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

total subgr. $= \mathrm{subgradient}_{\mathbf{w}}[R(\mathbf{w})] + \sum_k \left(g(\mathbf{x}^k, \mathbf{z}^k) - g(\hat{\mathbf{x}}^k, \mathbf{z}^k)\right)$

Max-margin learning (UNCONSTRAINED)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}}\left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

Subgradient algorithm

Repeat

1. compute global minimizers $\hat{\mathbf{x}}^k$ at current $\mathbf{w}$
2. compute **total subgradient** at current $\mathbf{w}$
3. update $\mathbf{w}$ by taking a step in the negative total subgradient direction

until convergence

total subgr. = $\mathrm{subgradient}_{\mathbf{w}}[R(\mathbf{w})] + \sum_k \left(g(\mathbf{x}^k, \mathbf{z}^k) - g(\hat{\mathbf{x}}^k, \mathbf{z}^k)\right)$

Max-margin learning (UNCONSTRAINED)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}}\left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

Stochastic subgradient algorithm
Repeat 1. pick k at random 2. compute global minimizer $\hat{\mathbf{x}}^k$ at current $\mathbf{w}$ 3. compute **partial subgradient** at current $\mathbf{w}$ 4. update $\mathbf{w}$ by taking a step in the negative partial subgradient direction **until convergence**

partial subgradient = $\mathrm{subgradient}_{\mathbf{w}}[R(\mathbf{w})] + g(\mathbf{x}^k, \mathbf{z}^k) - g(\hat{\mathbf{x}}^k, \mathbf{z}^k)$

Max-margin learning (UNCONSTRAINED)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \boxed{\min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)}$$

Stochastic subgradient algorithm

Repeat

1. pick k at random
2. compute $\boxed{\text{global minimizer } \hat{\mathbf{x}}^k}$ at current $\mathbf{w}$
3. compute **partial subgradient** at current $\mathbf{w}$
4. update $\mathbf{w}$ by taking a step in the negative partial subgradient direction

until convergence

MRF-MAP estimation per iteration

partial subgradient = $\mathrm{subgradient}_{\mathbf{w}}[R(\mathbf{w})] + g(\mathbf{x}^k, \mathbf{z}^k) - g(\hat{\mathbf{x}}^k, \mathbf{z}^k)$

Max-margin learning (UNCONSTRAINED)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \boxed{\min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)}$$

Stochastic subgradient algorithm

Repeat

1. pick k at random
2. compute global minimizer $\hat{\mathbf{x}}^k$ at current $\mathbf{w}$
3. compute **partial subgradient** at current $\mathbf{w}$
4. update $\mathbf{w}$ by taking a step in the negative partial subgradient direction

until convergence

MRF-MAP estimation per iteration
(unfortunately NP-hard)

partial subgradient = $\mathrm{subgradient}_{\mathbf{w}}[R(\mathbf{w})] + g(\mathbf{x}^k, \mathbf{z}^k) - g(\hat{\mathbf{x}}^k, \mathbf{z}^k)$

Max-margin learning (CONSTRAINED)

Max-margin learning (CONSTRAINED)

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

Max-margin learning (CONSTRAINED)

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

Max-margin learning (CONSTRAINED)

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

linear in $\mathbf{w}$

Max-margin learning (CONSTRAINED)

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

linear in $\mathbf{w}$

- Quadratic program (great!)

Max-margin learning (CONSTRAINED)

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

linear in $\mathbf{w}$

- Quadratic program (great!)
- But exponentially many constraints (not so great)

Max-margin learning (CONSTRAINED)

- What if we use only a small number of constrains?
  - Resulting QP can be solved
  - But solution may be infeasible

Max-margin learning (CONSTRAINED)

- What if we use only a small number of constrains?
  - Resulting QP can be solved
  - But solution may be infeasible

- **Constraint generation** to the rescue

Max-margin learning (CONSTRAINED)

- What if we use only a small number of constrains?
  - Resulting QP can be solved
  - But solution may be infeasible

- **Constraint generation** to the rescue
  - only few constraints **active** at optimal solution !! (variables much fewer than constraints)

Max-margin learning (CONSTRAINED)

- What if we use only a small number of constrains?
  - Resulting QP can be solved
  - But solution may be infeasible

- **Constraint generation** to the rescue
  - only few constraints **active** at optimal solution !! (variables much fewer than constraints)
  - Given the active constraints, rest can be ignored

Max-margin learning (CONSTRAINED)

- What if we use only a small number of constrains?
  - Resulting QP can be solved
  - But solution may be infeasible

- **Constraint generation** to the rescue
  - only few constraints **active** at optimal solution !! (variables much fewer than constraints)
  - Given the active constraints, rest can be ignored
  - Then let’s try to find them!

Constraint generation

1. Start with some constraints

Constraint generation

1. Start with some constraints
2. Solve QP

Constraint generation

1. Start with some constraints
2. Solve QP
3. Check if solution is feasible w.r.t. to **all** constraints

Constraint generation

1. Start with some constraints
2. Solve QP
3. Check if solution is feasible w.r.t. to **all** constraints
4. If yes, we are done!

Constraint generation

1. Start with some constraints
2. Solve QP
3. Check if solution is feasible w.r.t. to **all** constraints
4. If yes, we are done!
5. If no, pick a violated constraint and add it to the current set of constraints. Go to step 2

   (optionally, we can also remove inactive constraints)

Constraint generation

- **Key issue:** we must always be able to find a violated constraint if one exists

Constraint generation

- **Key issue:** we must always be able to find a violated constraint if one exists

- Recall the constraints for max-margin learning

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

Constraint generation

- **Key issue:** we must always be able to find a violated constraint if one exists

- Recall the constraints for max-margin learning

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

- To find violated constraint, we therefore need to compute:

$$\hat{\mathbf{x}}^k = \arg\min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) \right)$$

(just like subgradient method!)

Constraint generation

1. Initialize set of constraints C to empty

Constraint generation

1. Initialize set of constraints C to empty

2. Solve QP using current constraints C and obtain new $(\mathbf{w},\xi)$

Constraint generation

1. Initialize set of constraints C to empty
2. Solve QP using current constraints C and obtain new $(\mathbf{w},\xi)$
3. Compute global minimizers $\hat{\mathbf{x}}^k$ at current $\mathbf{w}$

Constraint generation

1. Initialize set of constraints C to empty
2. Solve QP using current constraints C and obtain new $(\mathbf{w},\xi)$
3. Compute global minimizers $\hat{\mathbf{x}}^k$ at current $\mathbf{w}$
4. For each k, if the following constraint is violated then add it to set C:

$$\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k) \leq \mathrm{MRF}_G(\hat{\mathbf{x}}^k;\mathbf{w},\mathbf{z}^k) - \Delta(\hat{\mathbf{x}}^k,\mathbf{x}^k) + \xi_k$$

Constraint generation

1. Initialize set of constraints C to empty
2. Solve QP using current constraints C and obtain new $(\mathbf{w},\xi)$
3. Compute global minimizers $\hat{\mathbf{x}}^k$ at current $\mathbf{w}$
4. For each k, if the following constraint is violated then add it to set C:
   $$\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k) \leq \mathrm{MRF}_G(\hat{\mathbf{x}}^k;\mathbf{w},\mathbf{z}^k) - \Delta(\hat{\mathbf{x}}^k,\mathbf{x}^k) + \xi_k$$
5. If no new constraint was added then terminate. Otherwise go to step 2.

Constraint generation

1. Initialize set of constraints C to empty
2. Solve QP using current constraints C and obtain new $(\mathbf{w},\xi)$
3. Compute $\boxed{\text{global minimizers } \hat{\mathbf{x}}^k}$ at current $\mathbf{w}$
4. For each k, if the following constraint is violated then add it to set C:
$$\mathrm{MRF}_G(\mathbf{x}^k;\mathbf{w},\mathbf{z}^k) \leq \mathrm{MRF}_G(\hat{\mathbf{x}}^k;\mathbf{w},\mathbf{z}^k) - \Delta(\hat{\mathbf{x}}^k,\mathbf{x}^k) + \xi_k$$
5. If no new constraint was added then terminate. Otherwise go to step 2.

MRF-MAP estimation **per sample**
(unfortunately **NP-hard**)

Max-margin learning (CONSTRAINED)

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

- Alternatively, we can solve above QP in the **dual domain**

Max-margin learning (CONSTRAINED)

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

- Alternatively, we can solve above QP in the **dual domain**
- dual variables $\leftrightarrow$ primal constraints

Max-margin learning (CONSTRAINED)

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

- Alternatively, we can solve above QP in the **dual domain**
- dual variables $\leftrightarrow$ primal constraints
- Too many variables, but most of them zero at optimal solution

Max-margin learning (CONSTRAINED)

$$\min_{\mathbf{w}} \frac{\mu}{2}||\mathbf{w}||^2 + \sum_k \xi_k$$

subject to the constraints:

$$\mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) \leq \mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k) + \xi_k$$

- Alternatively, we can solve above QP in the **dual domain**
- dual variables $\leftrightarrow$ primal constraints
- Too many variables, but most of them zero at optimal solution
- Use a **working-set** method
(essentially dual to constraint generation)

CRF Training via Dual Decomposition [CVPR 2011]

CRF training

- Existing max-margin (maximum likelihood) methods:

CRF training

- Existing max-margin (maximum likelihood) methods:
  - use MAP inference (probabilistic inference) w.r.t. an **equally complex CRF** as subroutine

CRF training

- Existing max-margin (maximum likelihood) methods:
  - use MAP inference (probabilistic inference) w.r.t. an **equally complex CRF** as subroutine
  - have to call subroutine **many times** during learning

CRF training

- Existing max-margin (maximum likelihood) methods:
  - ~~use MAP inference (probabilistic inference) w.r.t. an **equally complex CRF** as subroutine~~
  - ~~have to call subroutine **many times** during learning~~
- Suboptimal

CRF training

- Existing max-margin (maximum likelihood) methods:
  - ~~use MAP inference (probabilistic inference) w.r.t. an **equally complex CRF** as subroutine~~
  - ~~have to call subroutine **many times** during learning~~
- Suboptimal
  - computational efficiency ?

CRF training

- Existing max-margin (maximum likelihood) methods:
  - ~~use MAP inference (probabilistic inference) w.r.t. an **equally complex CRF** as subroutine~~
  - ~~have to call subroutine **many times** during learning~~
- Suboptimal
  - computational efficiency ?
  - accuracy ?

CRF training

- Existing max-margin (maximum likelihood) methods:
  - ~~use MAP inference (probabilistic inference) w.r.t. an **equally complex CRF** as subroutine~~
  - ~~have to call subroutine **many times** during learning~~
- Suboptimal
  - computational efficiency ?
  - accuracy ?
  - theoretical guarantees/properties ?

CRF training

- Existing max-margin (maximum likelihood) methods:
  - ~~use MAP inference (probabilistic inference) w.r.t. an **equally complex CRF** as subroutine~~
  - ~~have to call subroutine **many times** during learning~~

- Suboptimal
  - computational efficiency ?
  - accuracy ?
  - theoretical guarantees/properties ?

- **Key issue**: can we more properly exploit CRF structure during training?

CRF Training via Dual Decomposition

- Efficient max-margin training method

CRF Training via Dual Decomposition

- Efficient max-margin training method
- Reduces training of complex CRF to **parallel training of a series of easy-to-handle slave CRFs**

CRF Training via Dual Decomposition

- Efficient max-margin training method
- Reduces training of complex CRF to **parallel training of a series of easy-to-handle slave CRFs**
- Handles arbitrary **pairwise or higher-order** CRFs

CRF Training via Dual Decomposition

- Efficient max-margin training method
- Reduces training of complex CRF to **parallel training of a series of easy-to-handle slave CRFs**
- Handles arbitrary **pairwise or higher-order** CRFs
- Uses **very efficient** projected subgradient learning scheme

CRF Training via Dual Decomposition

- Efficient max-margin training method
- Reduces training of complex CRF to **parallel training of a series of easy-to-handle slave CRFs**
- Handles arbitrary **pairwise or higher-order** CRFs
- Uses **very efficient** projected subgradient learning scheme
- Allows hierarchy of structured prediction learning algorithms of **increasing accuracy**

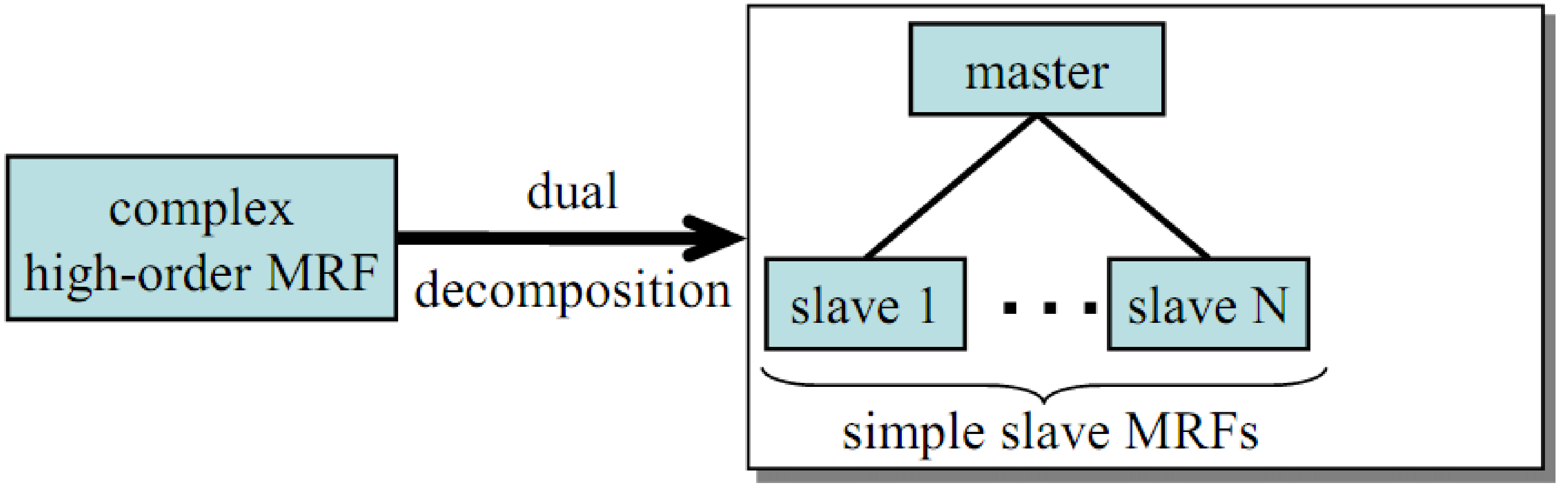
CRF Training via Dual Decomposition

- Efficient max-margin training method
- Reduces training of complex CRF to **parallel training of a series of easy-to-handle slave CRFs**
- Handles arbitrary **pairwise or higher-order** CRFs
- Uses **very efficient** projected subgradient learning scheme
- Allows hierarchy of structured prediction learning algorithms of **increasing accuracy**
- Very **flexible and adaptable**
  - Easily adjusted to fully exploit additional structure in any class of CRFs (no matter if they contain very high order cliques or not)

Dual Decomposition for MRF Optimization
(short review)

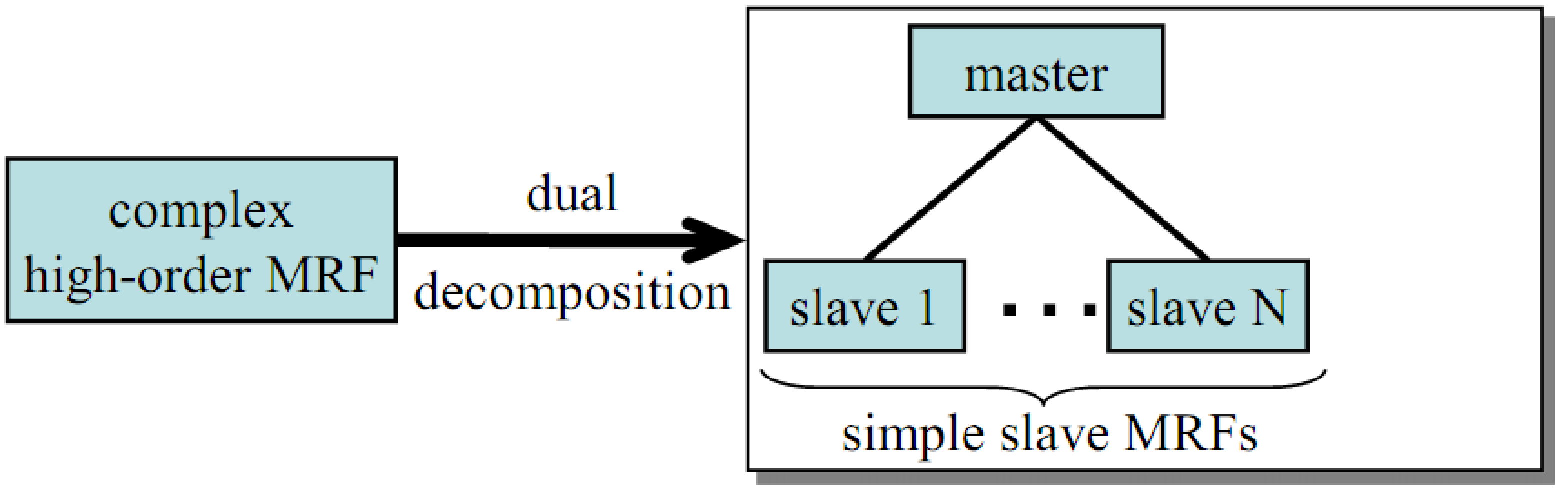
MRF Optimization via Dual Decomposition

- Very general framework for MAP inference [Komodakis et al. ICCV07, PAMI11]



MRF Optimization via Dual Decomposition

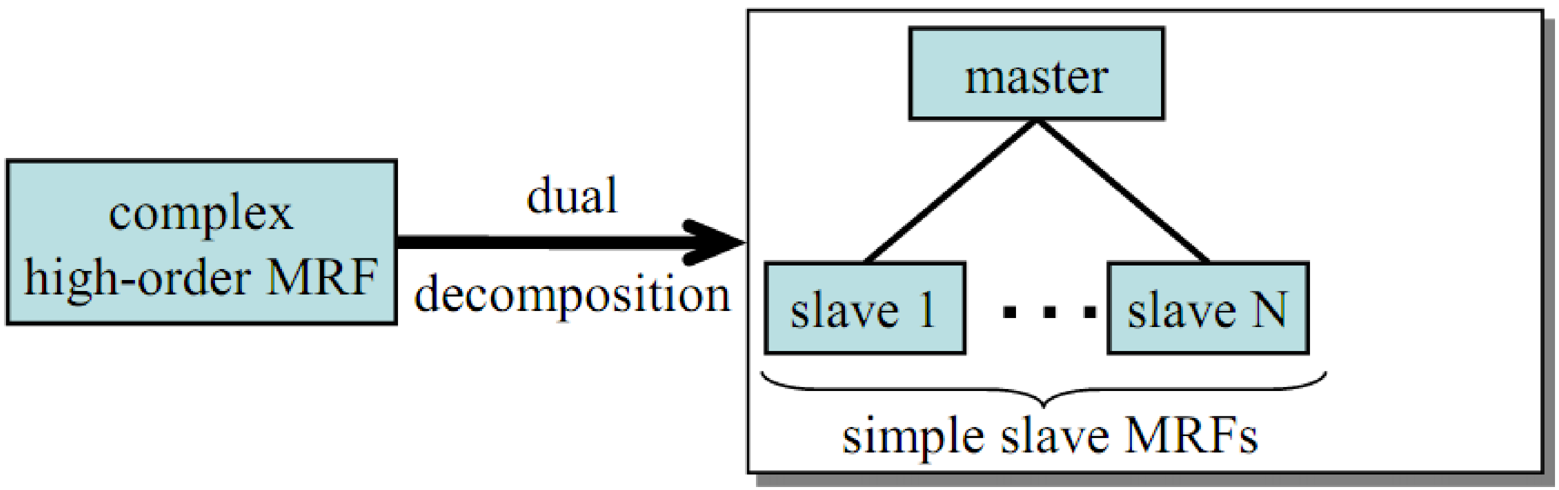
- Very general framework for MAP inference [Komodakis et al. ICCV07, PAMI11]



- Master = coordinator (has global view)
  Slaves = subproblems (have only local view)

MRF Optimization via Dual Decomposition

- Very general framework for MAP inference [Komodakis et al. ICCV07, PAMI11]

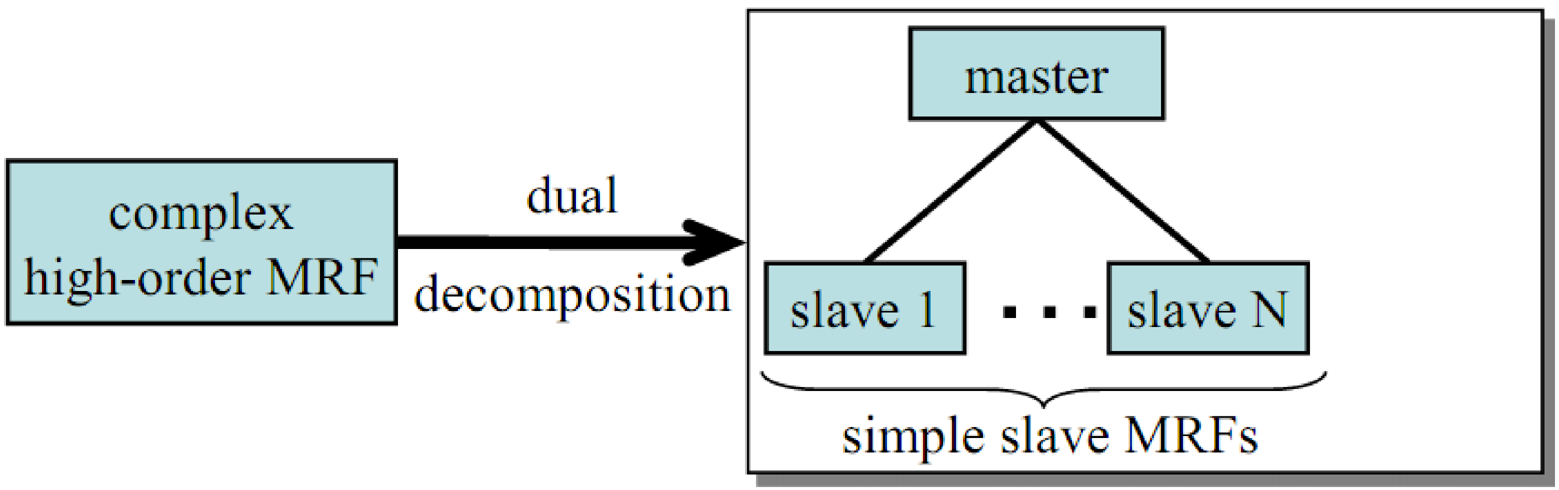


- Master $= \mathrm{MRF}_G(\mathbf{u},\mathbf{h}) \longleftarrow$ (MAP-MRF on hypergraph *G*)

  $= \min \; \mathrm{MRF}_G(\mathbf{x};\mathbf{u},\mathbf{h}) := \sum_{p\in\mathcal{V}} u_p(x_p) + \sum_{c\in\mathcal{C}} h_c(\mathbf{x}_c)$

MRF Optimization via Dual Decomposition

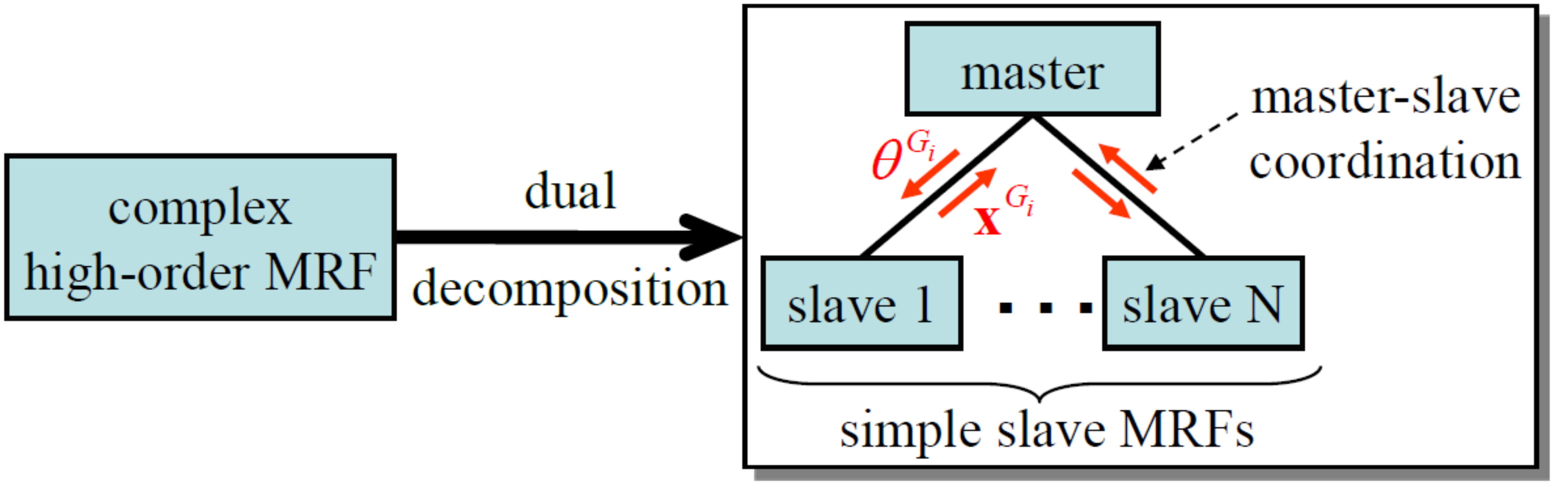
- Very general framework for MAP inference [Komodakis et al. ICCV07, PAMI11]



- Set of slaves = $\{\mathrm{MRF}_{G_i}(\boldsymbol{\theta}^i, \mathbf{h})\}$
  (MRFs on sub-hypergraphs G_i whose union covers G)
- Many other choices possible as well

MRF Optimization via Dual Decomposition

- Very general framework for MAP inference [Komodakis et al. ICCV07, PAMI11]



- Optimization proceeds in an iterative fashion via **master-slave coordination**

MRF Optimization via Dual Decomposition

Set of slave MRFs

$$\{\mathrm{MRF}_{G_i}(\boldsymbol{\theta}^i, \mathbf{h})\}$$

$\xrightarrow{\text{convex dual relaxation}}$

$$\mathrm{DUAL}_{\{G_i\}}(\mathbf{u}, \mathbf{h}) = \max_{\{\boldsymbol{\theta}^i\}} \sum\nolimits_i \mathrm{MRF}_{G_i}(\boldsymbol{\theta}^i, \mathbf{h})$$

$$\text{s.t.} \sum_{i \in \mathcal{I}_p} \theta_p^i(\cdot) = u_p(\cdot)$$

For each choice of slaves, master solves (possibly different) dual relaxation

MRF Optimization via Dual Decomposition

Set of slave MRFs
$$\{\mathrm{MRF}_{G_i}(\boldsymbol{\theta}^i, \mathbf{h})\}$$

convex dual relaxation

$$\mathrm{DUAL}_{\{G_i\}}(\mathbf{u}, \mathbf{h}) = \max_{\{\boldsymbol{\theta}^i\}} \sum\nolimits_i \mathrm{MRF}_{G_i}(\boldsymbol{\theta}^i, \mathbf{h})$$
$$\text{s.t.} \sum_{i \in \mathcal{I}_p} \theta_p^i(\cdot) = u_p(\cdot)$$

For each choice of slaves, master solves (possibly different) dual relaxation

- Sum of slave energies = lower bound on MRF optimum

MRF Optimization via Dual Decomposition

Set of slave MRFs
$$\{\mathrm{MRF}_{G_i}(\boldsymbol{\theta}^i, \mathbf{h})\}$$

convex dual relaxation

$$\mathrm{DUAL}_{\{G_i\}}(\mathbf{u}, \mathbf{h}) = \max_{\{\boldsymbol{\theta}^i\}} \sum\nolimits_i \mathrm{MRF}_{G_i}(\boldsymbol{\theta}^i, \mathbf{h})$$
$$\text{s.t.} \sum_{i \in \mathcal{I}_p} \theta_p^i(\cdot) = u_p(\cdot)$$

For each choice of slaves, master solves (possibly different) dual relaxation

- Sum of slave energies = lower bound on MRF optimum
- Dual relaxation = maximum such bound

MRF Optimization via Dual Decomposition

Set of slave MRFs

$$\{\mathrm{MRF}_{G_i}(\boldsymbol{\theta}^i, \mathbf{h})\}$$

convex dual relaxation

$$\mathrm{DUAL}_{\{G_i\}}(\mathbf{u}, \mathbf{h}) = \max_{\{\boldsymbol{\theta}^i\}} \sum\nolimits_i \mathrm{MRF}_{G_i}(\boldsymbol{\theta}^i, \mathbf{h})$$

$$\text{s.t.} \sum_{i \in \mathcal{I}_p} \theta_p^i(\cdot) = u_p(\cdot)$$

Choosing more difficult slaves $\Longrightarrow$ tighter lower bounds

$\Longrightarrow$ tighter dual relaxations

Dual Decomposition for MRF Optimization
(short review finished)

Max-margin learning via dual decomposition

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{w}, \mathbf{z}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{w}, \mathbf{z}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

Max-margin learning via dual decomposition

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{u}^k, \mathbf{h}^k) - \min_{\mathbf{x}} \left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{u}^k, \mathbf{h}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

Max-margin learning via dual decomposition

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{u}^k, \mathbf{h}^k) - \min_{\mathbf{x}}\left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{u}^k, \mathbf{h}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

$$\Delta(\mathbf{x}, \mathbf{x}^k) = \sum_p \delta_p(x_p, x_p^k) + \sum_c \delta_c(\mathbf{x}_c, \mathbf{x}_c^k) \qquad \boxed{\Delta(\mathbf{x}, \mathbf{x}) = 0}$$

Max-margin learning via dual decomposition

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{u}^k, \mathbf{h}^k) - \min_{\mathbf{x}}\left(\mathrm{MRF}_G(\mathbf{x}; \mathbf{u}^k, \mathbf{h}^k) - \Delta(\mathbf{x}, \mathbf{x}^k)\right)$$

$$\Delta(\mathbf{x}, \mathbf{x}^k) = \sum_p \delta_p(x_p, x_p^k) + \sum_c \delta_c(\mathbf{x}_c, \mathbf{x}_c^k) \qquad \boxed{\Delta(\mathbf{x}, \mathbf{x}) = 0}$$

$$\boxed{\begin{aligned} \bar{u}_p^k(\cdot) &= u_p^k(\cdot) - \delta_p(\cdot, x_p^k) \\ \bar{h}_c^k(\cdot) &= h_c^k(\cdot) - \delta_c(\cdot, \mathbf{x}_c^k) \end{aligned}}$$

loss-augmented potentials

Max-margin learning via dual decomposition

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{u}^k, \mathbf{h}^k) - \min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$$

$$\Delta(\mathbf{x}, \mathbf{x}^k) = \sum_p \delta_p(x_p, x_p^k) + \sum_c \delta_c(\mathbf{x}_c, \mathbf{x}_c^k) \qquad \boxed{\Delta(\mathbf{x}, \mathbf{x}) = 0}$$

$$\boxed{\begin{aligned} \bar{u}_p^k(\cdot) &= u_p^k(\cdot) - \delta_p(\cdot, x_p^k) \\ \bar{h}_c^k(\cdot) &= h_c^k(\cdot) - \delta_c(\cdot, \mathbf{x}_c^k) \end{aligned}}$$

loss-augmented potentials

Max-margin learning via dual decomposition

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \mathbf{u}^k, \mathbf{h}^k) - \min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$$

$$\Delta(\mathbf{x}, \mathbf{x}^k) = \sum_p \delta_p(x_p, x_p^k) + \sum_c \delta_c(\mathbf{x}_c, \mathbf{x}_c^k) \qquad \boxed{\Delta(\mathbf{x}, \mathbf{x}) = 0}$$

$$\boxed{\begin{aligned} \bar{u}_p^k(\cdot) &= u_p^k(\cdot) - \delta_p(\cdot, x_p^k) \\ \bar{h}_c^k(\cdot) &= h_c^k(\cdot) - \delta_c(\cdot, \mathbf{x}_c^k) \end{aligned}}$$

$$\delta_p(x_p, x_p) = 0$$
$$\delta_c(\mathbf{x}_c, \mathbf{x}_c) = 0$$

loss-augmented potentials

Max-margin learning via dual decomposition

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right)$$

$$L_G\left(\mathbf{x}^k, \mathbf{z}^k; \mathbf{w}\right) = \mathrm{MRF}_G(\mathbf{x}^k; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) - \min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$$

$$\Delta(\mathbf{x}, \mathbf{x}^k) = \sum_p \delta_p(x_p, x_p^k) + \sum_c \delta_c(\mathbf{x}_c, \mathbf{x}_c^k) \qquad \boxed{\Delta(\mathbf{x}, \mathbf{x}) = 0}$$

$$\boxed{\begin{aligned} \bar{u}_p^k(\cdot) &= u_p^k(\cdot) - \delta_p(\cdot, x_p^k) \\ \bar{h}_c^k(\cdot) &= h_c^k(\cdot) - \delta_c(\cdot, \mathbf{x}_c^k) \end{aligned}}$$

$$\Downarrow$$

$$\delta_p(x_p, x_p) = 0$$

$$\delta_c(\mathbf{x}_c, \mathbf{x}_c) = 0$$

loss-augmented potentials

Max-margin learning via dual decomposition

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G(\mathbf{x}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w})$$

$$L_G(\mathbf{x}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w}) = \mathrm{MRF}_G(\mathbf{x}^k; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) - \min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$$

$$\Delta(\mathbf{x}, \mathbf{x}^k) = \sum_p \delta_p(x_p, x_p^k) + \sum_c \delta_c(\mathbf{x}_c, \mathbf{x}_c^k) \qquad \boxed{\Delta(\mathbf{x}, \mathbf{x}) = 0}$$

$$\boxed{\begin{aligned} \bar{u}_p^k(\cdot) &= u_p^k(\cdot) - \delta_p(\cdot, x_p^k) \\ \bar{h}_c^k(\cdot) &= h_c^k(\cdot) - \delta_c(\cdot, \mathbf{x}_c^k) \end{aligned}}$$

$$\delta_p(x_p, x_p) = 0$$

$$\delta_c(\mathbf{x}_c, \mathbf{x}_c) = 0$$

loss-augmented potentials

Max-margin learning via dual decomposition

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G(\mathbf{x}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w})$$

$$L_G(\mathbf{x}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w}) = \mathrm{MRF}_G(\mathbf{x}^k; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) - \min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$$

Problem

Learning objective intractable due to this term

Max-margin learning via dual decomposition

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G(\mathbf{x}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w})$$

$$L_G(\mathbf{x}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w}) = \mathrm{MRF}_G(\mathbf{x}^k; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) - \min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$$

Solution: approximate this term with dual relaxation from decomposition $\{G_i = (\mathcal{V}_i, \mathcal{C}_i)\}_{i=1}^{N}$

$$\min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) \approx \mathrm{DUAL}_{\{G_i\}}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$$

Max-margin learning via dual decomposition

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G(\mathbf{x}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w})$$

$$L_G(\mathbf{x}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w}) = \mathrm{MRF}_G(\mathbf{x}^k; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) - \min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$$

Solution: approximate this term with dual relaxation from decomposition $\{G_i = (\mathcal{V}_i, \mathcal{C}_i)\}_{i=1}^{N}$

$$\min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) \approx \mathrm{DUAL}_{\{G_i\}}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$$

$$\mathrm{DUAL}_{\{G_i\}}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) = \max_{\{\boldsymbol{\theta}^{(i,k)}\}} \sum_i \mathrm{MRF}_{G_i}(\boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k)$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot)$$

Max-margin learning via dual decomposition

now

$$\min_{\mathbf{w},\{\boldsymbol{\theta}^{(i,k)}\}} R(\mathbf{w}) + \sum_k \sum_i L_{G_i}(\mathbf{x}^k, \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k; \mathbf{w})$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot) \quad .$$

Solution: approximate this term with dual relaxation from decomposition $\{G_i = (\mathcal{V}_i, \mathcal{C}_i)\}_{i=1}^N$

$$\min_{\mathbf{x}} \mathrm{MRF}_G(\mathbf{x}; \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) \approx \mathrm{DUAL}_{\{G_i\}}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k)$$

$$\mathrm{DUAL}_{\{G_i\}}(\bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k) = \max_{\{\boldsymbol{\theta}^{(i,k)}\}} \sum_i \mathrm{MRF}_{G_i}(\boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k)$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot)$$

Max-margin learning via dual decomposition

now

$$\min_{\mathbf{w},\{\boldsymbol{\theta}^{(i,k)}\}} R(\mathbf{w}) + \sum_k \sum_i L_{G_i}(\mathbf{x}^k, \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k; \mathbf{w})$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot) \quad .$$

before

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G(\mathbf{x}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w})$$

Max-margin learning via dual decomposition

now

$$\min_{\mathbf{w},\{\boldsymbol{\theta}^{(i,k)}\}} R(\mathbf{w}) + \sum_k \sum_i L_{G_i}(\mathbf{x}^k, \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k; \mathbf{w})$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot) \quad .$$

before

$$\min_{\mathbf{w}} R(\mathbf{w}) + \sum_{k=1}^{K} L_G(\mathbf{x}^k, \bar{\mathbf{u}}^k, \bar{\mathbf{h}}^k; \mathbf{w})$$

Essentially, training of complex CRF decomposed to parallel training of easy-to-handle slave CRFs !!!

Max-margin learning via dual decomposition

- Global optimum via projected subgradient method
  (slight variation of subgradient method)

Max-margin learning via dual decomposition

- Global optimum via projected subgradient method (slight variation of subgradient method)

Projected subgradient
Repeat 1. compute subgradient at current $\mathbf{w}$ 2. update $\mathbf{w}$ by taking a step in the negative subgradient direction 3. project into feasible set **until convergence**

Projected subgradient learning algorithm

- **Input:**
  - K training samples $\left\{(\mathbf{x}^k, \mathbf{z}^k)\right\}_{k=1}^{K}$
  - Hypergraph $G = (\mathcal{V}, \mathcal{C})$
    (in general hypergraphs can vary per sample)
  - Vector valued feature functions $\{g_p(\cdot,\cdot)\}$, $\{g_c(\cdot,\cdot)\}$

Projected subgradient learning algorithm

$\forall k,$ choose decomposition $\{G_i = (\mathcal{V}_i, \mathcal{C}_i)\}_{i=1}^{N}$ of hypergraph G

$\forall k, i,$ initialize $\boldsymbol{\theta}^{(i,k)}$ so as to satisfy $\sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot)$

Projected subgradient learning algorithm

$\forall k,$ choose decomposition $\{G_i = (\mathcal{V}_i, \mathcal{C}_i)\}_{i=1}^N$ of hypergraph G

$\forall k, i,$ initialize $\boldsymbol{\theta}^{(i,k)}$ so as to satisfy $\sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot)$

repeat

until convergence

Projected subgradient learning algorithm

$\forall k,$ choose decomposition $\{G_i = (\mathcal{V}_i, \mathcal{C}_i)\}_{i=1}^{N}$ of hypergraph G

$\forall k, i,$ initialize $\boldsymbol{\theta}^{(i,k)}$ so as to satisfy $\sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot)$

repeat

> ***// optimize slave MRFs***
>
> $\forall k, i,$ compute minimizer $\hat{\mathbf{x}}^{(i,k)} = \arg\min_{\mathbf{x}} \mathrm{MRF}_{G_i}(\mathbf{x}; \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k)$

until convergence

Projected subgradient learning algorithm

$\forall k,$ choose decomposition $\{G_i = (\mathcal{V}_i, \mathcal{C}_i)\}_{i=1}^N$ of hypergraph G

$\forall k, i,$ initialize $\boldsymbol{\theta}^{(i,k)}$ so as to satisfy $\sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot)$

repeat

// ***optimize slave MRFs***

$\forall k, i,$ compute minimizer $\hat{\mathbf{x}}^{(i,k)} = \arg\min_{\mathbf{x}} \mathrm{MRF}_{G_i}(\mathbf{x}; \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k)$

// ***update*** $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{w} - \alpha_t \cdot d\mathbf{w}$ fully specified from $\hat{\mathbf{x}}^{(i,k)}$

until convergence

Projected subgradient learning algorithm

$\forall k,$ choose decomposition $\{G_i = (\mathcal{V}_i, \mathcal{C}_i)\}_{i=1}^{N}$ of hypergraph G

$\forall k, i,$ initialize $\boldsymbol{\theta}^{(i,k)}$ so as to satisfy $\sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot)$

repeat

// optimize slave MRFs

$\forall k, i,$ compute minimizer $\hat{\mathbf{x}}^{(i,k)} = \arg\min_{\mathbf{x}} \mathrm{MRF}_{G_i}(\mathbf{x}; \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k)$

// update $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{w} - \alpha_t \cdot d\mathbf{w}$ fully specified from $\hat{\mathbf{x}}^{(i,k)}$

// update $\boldsymbol{\theta}^{(i,k)}$

$$\theta_p^{(i,k)}(\cdot) \mathrel{+}= \alpha_t \left(\left[\hat{x}_p^{(i,k)} = \cdot \right] - \frac{\sum_{j \in \mathcal{I}_p} \left[\hat{x}_p^{(j,k)} = \cdot \right]}{|\mathcal{I}_p|} \right)$$

until convergence

Projected subgradient learning algorithm

$\forall k,$ choose decomposition $\{G_i = (\mathcal{V}_i, \mathcal{C}_i)\}_{i=1}^N$ of hypergraph G

$\forall k, i,$ initialize $\boldsymbol{\theta}^{(i,k)}$ so as to satisfy $\sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot)$

repeat

// optimize slave MRFs

$\forall k, i,$ compute minimizer $\hat{\mathbf{x}}^{(i,k)} = \arg\min_{\mathbf{x}} \mathrm{MRF}_{G_i}(\mathbf{x}; \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k)$

// update $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{w} - \alpha_t \cdot d\mathbf{w}$ ⟸ fully specified from $\hat{\mathbf{x}}^{(i,k)}$

// update $\boldsymbol{\theta}^{(i,k)}$

$$\theta_p^{(i,k)}(\cdot) \mathrel{+}= \alpha_t \left(\left[\hat{x}_p^{(i,k)} = \cdot \right] - \frac{\sum_{j \in \mathcal{I}_p} \left[\hat{x}_p^{(j,k)} = \cdot \right]}{|\mathcal{I}_p|} \right)$$

until convergence

(we only need to know how to optimize slave MRFs !!)

Projected subgradient learning algorithm

- **Incremental subgradient** version:
  - Same as before but considers subset of slaves per iteration
  - Subset chosen
    - deterministically or
    - randomly (**stochastic subgradient**)
  - Further improves computational efficiency
  - Same optimality guarantees & theoretical properties

Projected subgradient learning algorithm

$\forall k,$ choose decomposition $\{G_i = (\mathcal{V}_i, \mathcal{C}_i)\}_{i=1}^{N}$ of hypergraph G

$\forall k, i,$ initialize $\boldsymbol{\theta}^{(i,k)}$ so as to satisfy $\sum_{i \in \mathcal{I}_p} \theta_p^{(i,k)}(\cdot) = \bar{u}_p^k(\cdot)$

repeat

pick k

// ***optimize slave MRFs***

$\forall i,$ compute minimizer $\hat{\mathbf{x}}^{(i,k)} = \arg\min_{\mathbf{x}} \mathrm{MRF}_{G_i}(\mathbf{x}; \boldsymbol{\theta}^{(i,k)}, \bar{\mathbf{h}}^k)$

// ***update*** $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{w} - \alpha_t \cdot d\mathbf{w}$ fully specified from $\hat{\mathbf{x}}^{(i,k)}$

// ***update*** $\boldsymbol{\theta}^{(i,k)}$

$$\theta_p^{(i,k)}(\cdot) \mathrel{+}= \alpha_t \left(\left[\hat{x}_p^{(i,k)} = \cdot \right] - \frac{\sum_{j \in \mathcal{I}_p} \left[\hat{x}_p^{(j,k)} = \cdot \right]}{|\mathcal{I}_p|} \right)$$

until convergence

Projected subgradient learning algorithm

- Resulting learning scheme:
  - ✓ Very efficient and very flexible

Projected subgradient learning algorithm

- Resulting learning scheme:
  - ✓ Very efficient and very flexible
  - ✓ Requires from the user only to provide an optimizer for the slave MRFs

Projected subgradient learning algorithm

- Resulting learning scheme:
  - ✓ Very efficient and very flexible
  - ✓ Requires from the user only to provide an optimizer for the slave MRFs
  - ✓ Slave problems freely chosen by the user

Projected subgradient learning algorithm

- Resulting learning scheme:
  - ✓ Very efficient and very flexible
  - ✓ Requires from the user only to provide an optimizer for the slave MRFs
  - ✓ Slave problems freely chosen by the user
  - ✓ Easily adaptable to further exploit special structure of any class of CRFs

Choice of decompositions $\{G_i\}$

Choice of decompositions $\{G_i\}$

$\mathcal{F}_0$ = true loss (intractable)

$\mathcal{F}_{\{G_i\}}$ = loss when using decomposition $\{G_i\}$

Choice of decompositions $\{G_i\}$

$\mathcal{F}_0$ = true loss (intractable)

$\mathcal{F}_{\{G_i\}}$ = loss when using decomposition $\{G_i\}$

- $\mathcal{F}_0 \leq \mathcal{F}_{\{G_i\}}$

  (upper bound property)

Choice of decompositions $\{G_i\}$

$\mathcal{F}_0$ = true loss (intractable)

$\mathcal{F}_{\{G_i\}}$ = loss when using decomposition $\{G_i\}$

- $\mathcal{F}_0 \leq \mathcal{F}_{\{G_i\}}$

  (upper bound property)

- $\{G_i\} < \{\tilde{G}_j\}$

  (hierarchy of learning algorithms)

Choice of decompositions $\{G_i\}$

- $G_{\text{single}} = \{G_c\}_{c \in \mathcal{C}}$ denotes following decomposition:
  - One slave per clique $c \in \mathcal{C}$
  - Corresponding sub-hypergraph $G_c = (\mathcal{V}_c, \mathcal{C}_c)$:

    $\mathcal{V}_c = \{p | p \in c\}, \mathcal{C}_c = \{c\}$

Choice of decompositions $\{G_i\}$

- $G_{\text{single}} = \{G_c\}_{c \in \mathcal{C}}$ denotes following decomposition:
  - One slave per clique $c \in \mathcal{C}$
  - Corresponding sub-hypergraph $G_c = (\mathcal{V}_c, \mathcal{C}_c)$:

    $\mathcal{V}_c = \{p | p \in c\}, \mathcal{C}_c = \{c\}$

- Resulting slaves often easy (or even trivial) to solve even if global problem is complex and NP-hard
  - leads to widely applicable learning algorithm

Choice of decompositions $\{G_i\}$

- $G_{\text{single}} = \{G_c\}_{c \in \mathcal{C}}$ denotes following decomposition:
  - One slave per clique $c \in \mathcal{C}$
  - Corresponding sub-hypergraph $G_c = (\mathcal{V}_c, \mathcal{C}_c)$:

    $$\mathcal{V}_c = \{p | p \in c\}, \mathcal{C}_c = \{c\}$$

- Resulting slaves often easy (or even trivial) to solve even if global problem is complex and NP-hard
  - leads to widely applicable learning algorithm
- Corresponding dual relaxation is an LP
  - Generalizes well known LP relaxation for pairwise MRFs (at the core of most state-of-the-art methods)

Choice of decompositions $\{G_i\}$

- But we can do better if CRFs have special structure…

Choice of decompositions $\{G_i\}$

- But we can do better if CRFs have special structure…
- Structure means:
  - More efficient optimizer for slaves **(speed)**

Choice of decompositions $\{G_i\}$

- But we can do better if CRFs have special structure…
- Structure means:
  - More efficient optimizer for slaves **(speed)**
  - Optimizer that handles more complex slaves **(accuracy)**

(Almost all known examples fall in one of above two cases)

Choice of decompositions $\{G_i\}$

- But we can do better if CRFs have special structure…
- Structure means:
  - More efficient optimizer for slaves **(speed)**
  - Optimizer that handles more complex slaves **(accuracy)**

  (Almost all known examples fall in one of above two cases)
- We are essentially adapting decomposition to exploit the structure of the problem at hand

Choice of decompositions $\{G_i\}$

- But we can do better if CRFs have special structure…
- E.g., **pattern-based** high-order potentials (for a clique c) [Komodakis & Paragios CVPR09]

$$H_c(\mathbf{x}) = \begin{cases} \psi_c(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{P} \\ \psi_c^{\max} & \text{otherwise} \end{cases}$$

$\mathcal{P}$ subset of $\mathcal{L}^{|c|}$ (its vectors called **patterns**)

Choice of decompositions $\{G_i\}$

- But we can do better if CRFs have special structure…
- E.g., **pattern-based** high-order potentials (for a clique c) [Komodakis & Paragios CVPR09]

$$H_c(\mathbf{x}) = \begin{cases} \psi_c(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{P} \\ \psi_c^{\max} & \text{otherwise} \end{cases}$$

$\mathcal{P}$ subset of $\mathcal{L}^{|c|}$ (its vectors called **patterns**)

- We only assume:
  - Set $\mathcal{P}$ is sparse
  - It holds $\psi_c(\mathbf{x}) \leq \psi_c^{\max},\ \forall \mathbf{x} \in \mathcal{P}$
  - No other restriction

Choice of decompositions $\{G_i\}$

- Tree decomposition $G_{\text{tree}} = \{T_i\}_{i=1}^{N}$
  (T_i are spanning trees that cover the graph)

Choice of decompositions $\{G_i\}$

- Tree decomposition $G_{\text{tree}} = \{T_i\}_{i=1}^{N}$
  (T_i are spanning trees that cover the graph)

- No improvement in accuracy

$$\mathrm{DUAL}_{G_{\text{tree}}} = \mathrm{DUAL}_{G_{\text{single}}} \Rightarrow \mathcal{F}_{G_{\text{tree}}} = \mathcal{F}_{G_{\text{single}}}$$

Choice of decompositions $\{G_i\}$

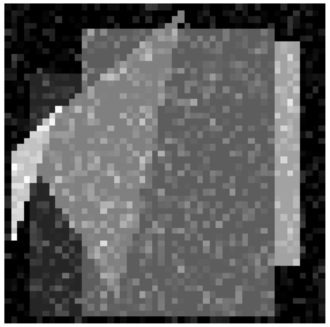
- Tree decomposition $G_{\text{tree}} = \{T_i\}_{i=1}^{N}$
  (T_i are spanning trees that cover the graph)

- No improvement in accuracy
  $\mathrm{DUAL}_{G_{\text{tree}}} = \mathrm{DUAL}_{G_{\text{single}}} \Rightarrow \mathcal{F}_{G_{\text{tree}}} = \mathcal{F}_{G_{\text{single}}}$

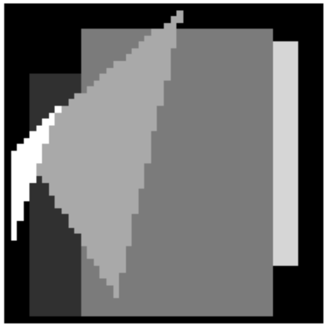
- But improvement in speed
  ($\mathrm{DUAL}_{G_{\text{tree}}}$ converges faster than $\mathrm{DUAL}_{G_{\text{single}}}$)

Image denoising

- Piecewise constant images



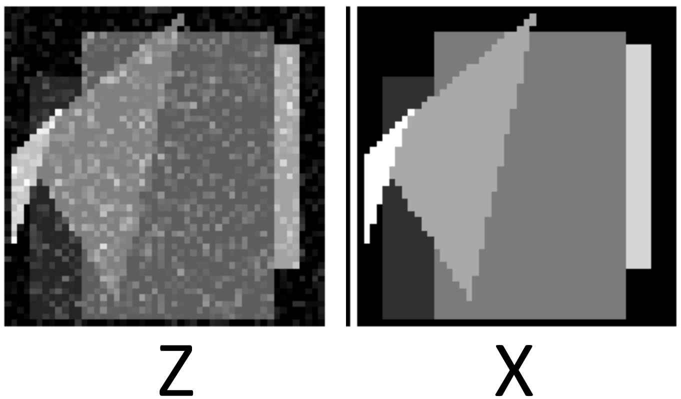
Z



X

Image denoising

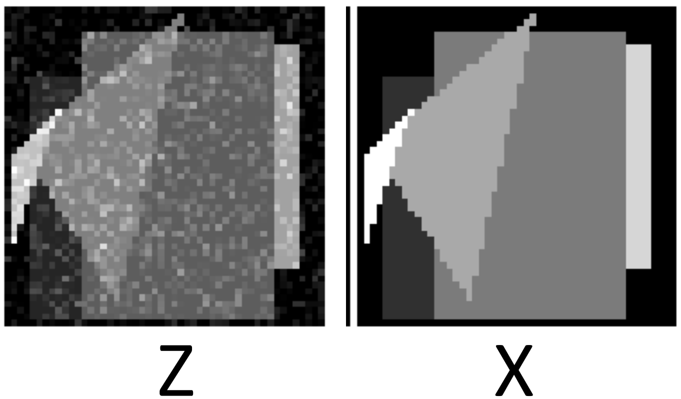
- Piecewise constant images



- Potentials: $u_p^k\left(x_p\right)=\left|x_p-z_p\right| \qquad h_{pq}^k\left(x_p,x_q\right)=V\left(\left|x_p-x_q\right|\right)$

Image denoising

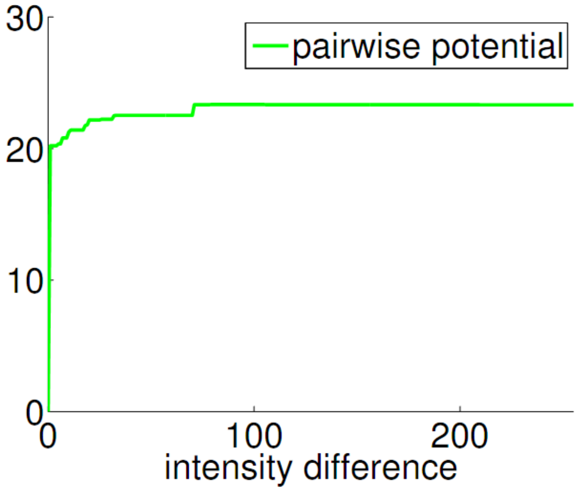
- Piecewise constant images



- Potentials: $u_p^k\left(x_p\right)=\left|x_p-z_p\right| \qquad h_{pq}^k\left(x_p,x_q\right)=V\left(\left|x_p-x_q\right|\right)$

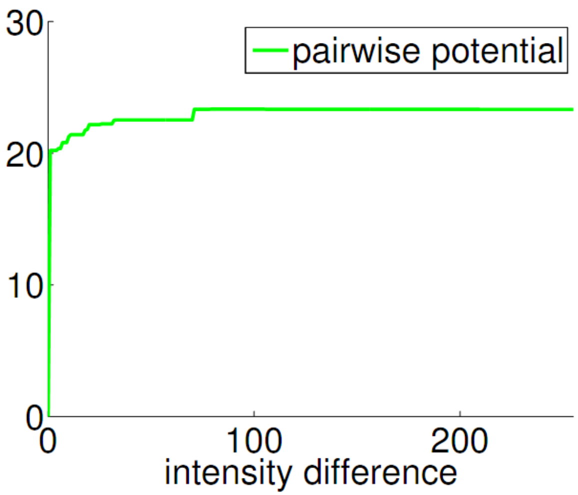
- Goal: learn pairwise potential $V(\cdot)$

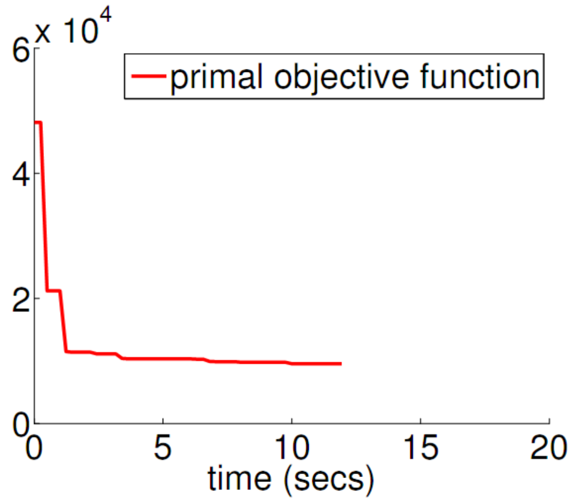
Image denoising



learnt potential

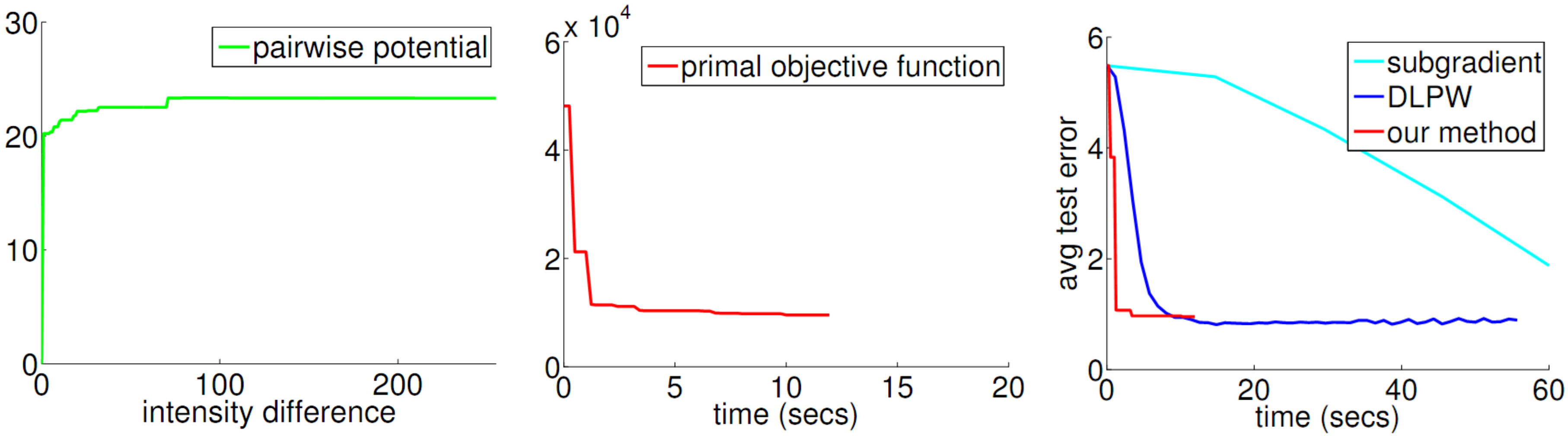
Image denoising





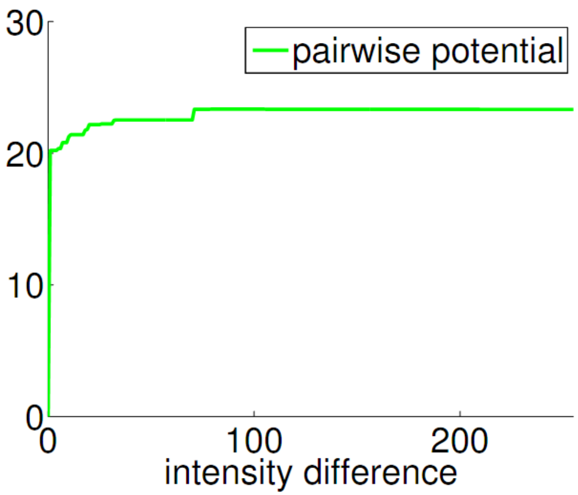
learnt potential

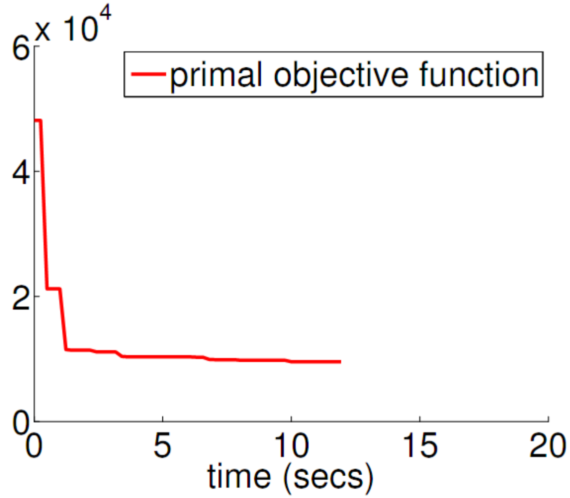
Image denoising

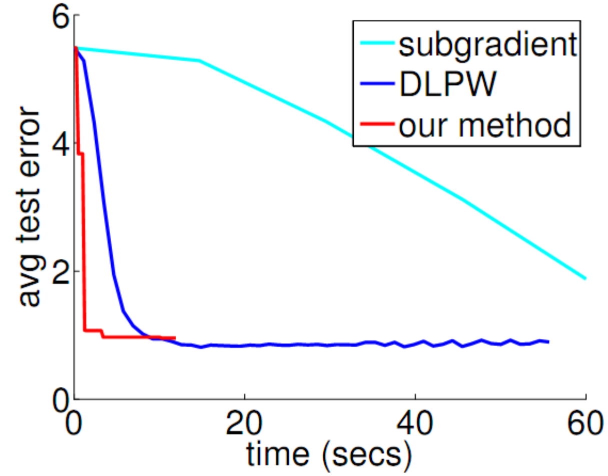


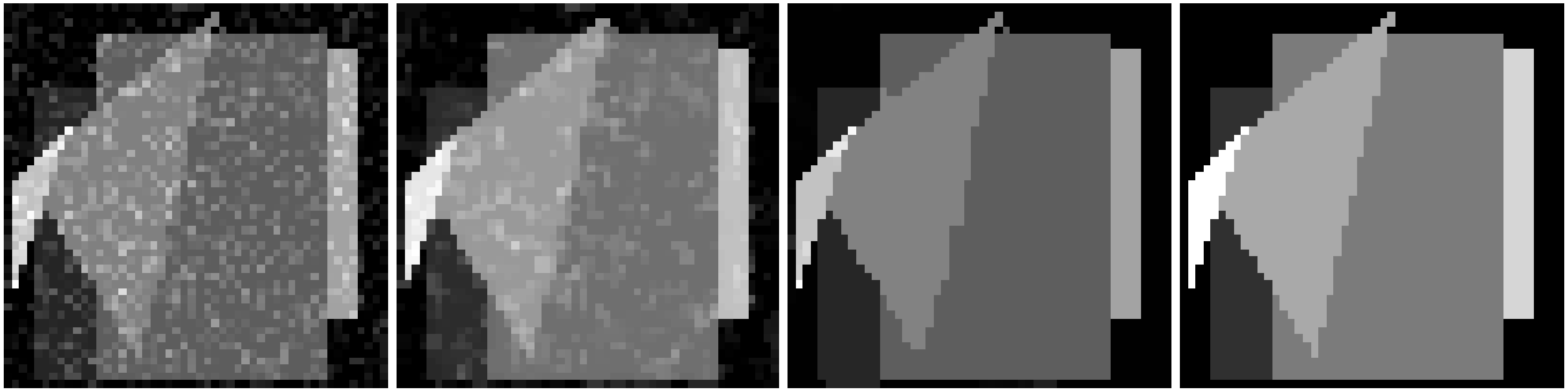
learnt potential

Image denoising









Stereo matching

- Potentials: $$u_p^k\left(x_p\right)=\left|I^{left}\left(p\right)-I^{right}\left(p-x_p\right)\right|$$

$$h_{pq}^k\left(x_p,x_q\right)=f\left(\left|\nabla I^{left}(p)\right|\right)\left[x_p\neq x_q\right]$$

Stereo matching

- Potentials: $u_p^k\left(x_p\right)=\left|I^{left}\left(p\right)-I^{right}\left(p-x_p\right)\right|$

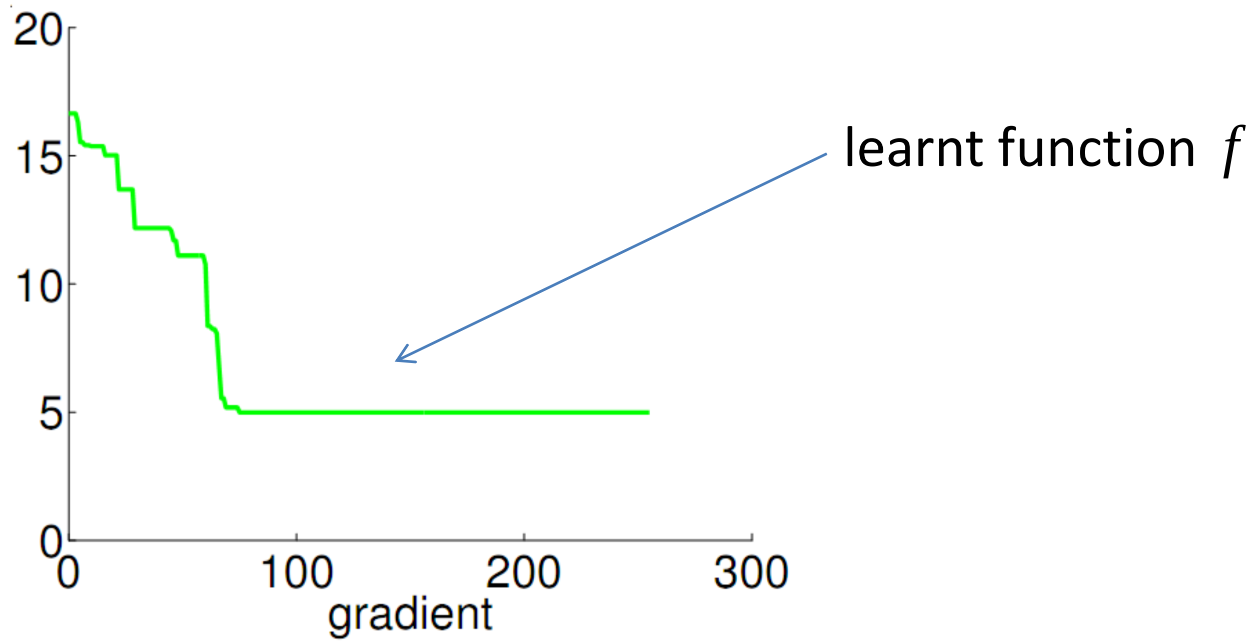
$$h_{pq}^k\left(x_p, x_q\right)= f\left(\left|\nabla I^{left}(p)\right|\right)\left[x_p \neq x_q\right]$$

- Goal: learn function $f(\cdot)$ for gradient-modulated Potts model

Stereo matching

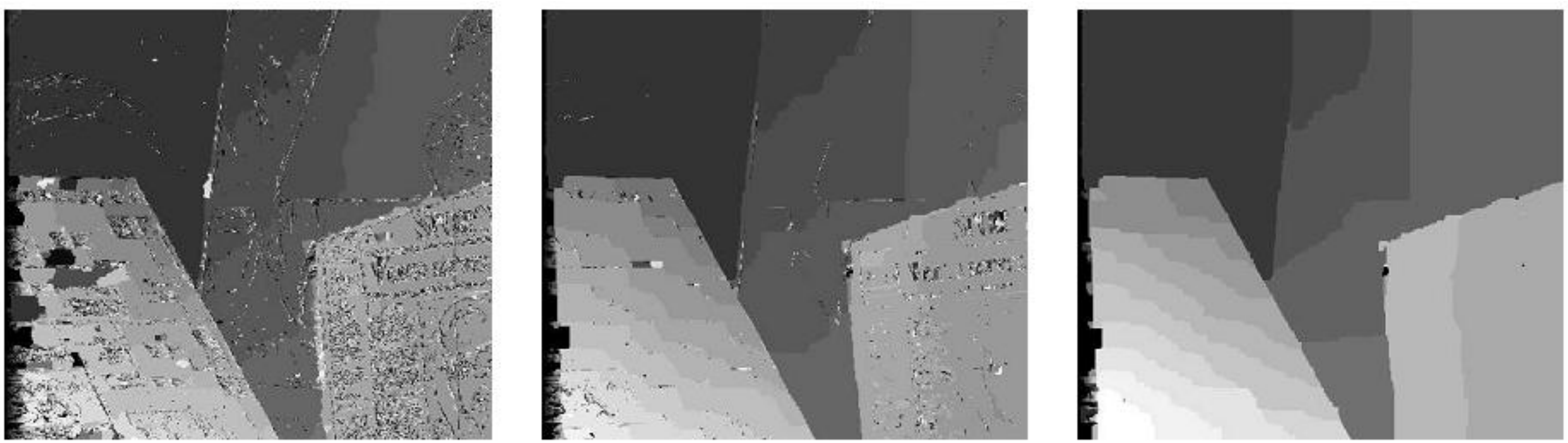
- Potentials: $$u_p^k\left(x_p\right)=\left|I^{left}\left(p\right)-I^{right}\left(p-x_p\right)\right|$$
$$h_{pq}^k\left(x_p,x_q\right)=f\left(\left|\nabla I^{left}(p)\right|\right)\left[x_p\neq x_q\right]$$

- Goal: learn function $f(\cdot)$ for gradient-modulated Potts model



Stereo matching

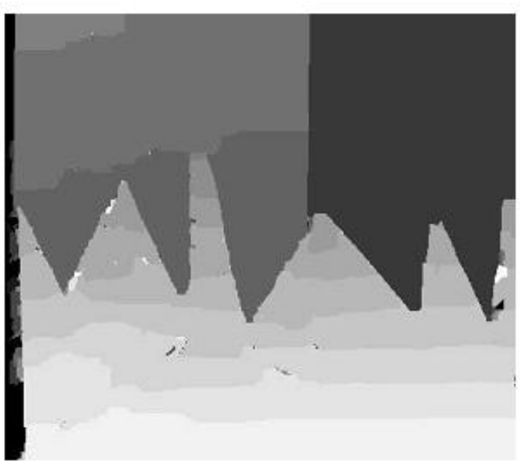
- Potentials: $$u_p^k\left(x_p\right)=\left|I^{left}\left(p\right)-I^{right}\left(p-x_p\right)\right|$$
$$h_{pq}^k\left(x_p,x_q\right)=f\left(\left|\nabla I^{left}(p)\right|\right)\left[x_p\neq x_q\right]$$
- Goal: learn function $f(\cdot)$ for gradient-modulated Potts model



"Venus" disparity using $f(\cdot)$ as estimated at different iterations of learning algorithm

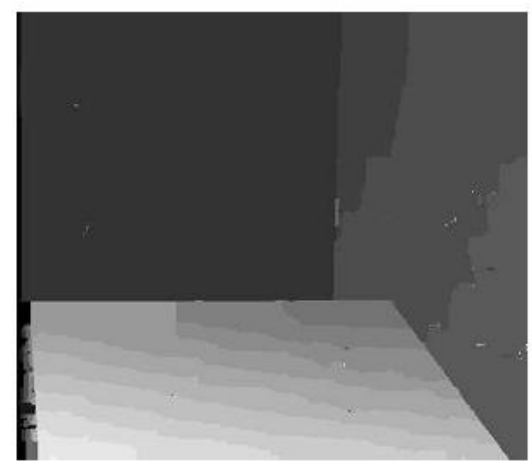
Stereo matching

- Potentials: $$u_p^k\left(x_p\right) = \left| I^{left}\left(p\right) - I^{right}\left(p - x_p\right)\right|$$ $$h_{pq}^k\left(x_p, x_q\right) = f\left(\left|\nabla I^{left}(p)\right|\right)\left[x_p \neq x_q\right]$$
- Goal: learn function $f(\cdot)$ for gradient-modulated Potts model



Sawtooth
4.9%

Poster
3.7%

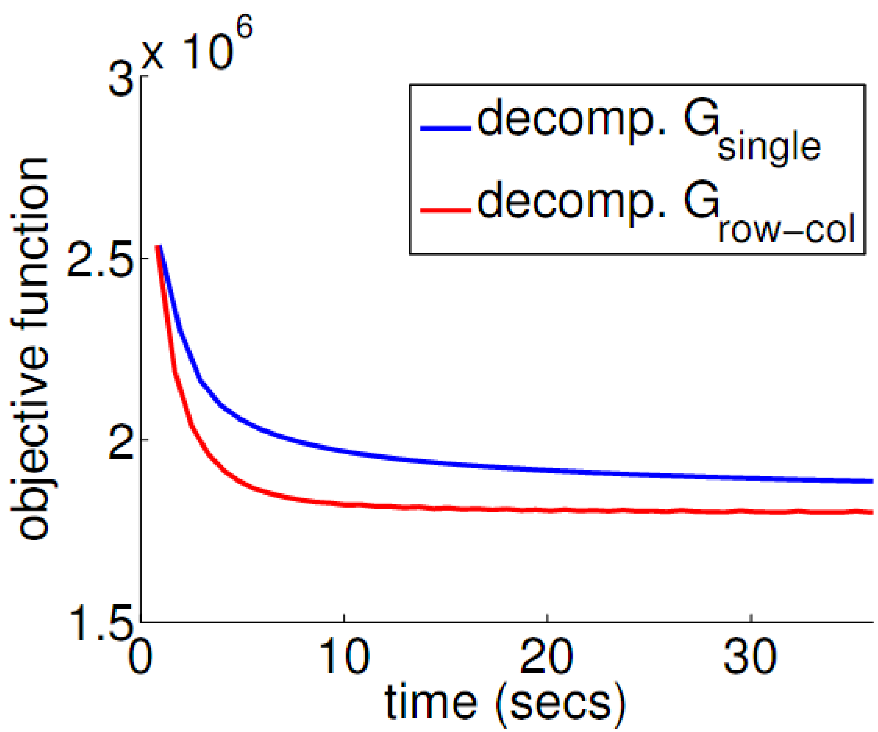


Bull
2.8%

Stereo matching

- Potentials: $$u_p^k\left(x_p\right) = \left|I^{left}\left(p\right) - I^{right}\left(p - x_p\right)\right|$$
$$h_{pq}^k\left(x_p, x_q\right) = f\left(\left|\nabla I^{left}(p)\right|\right)\left[x_p \neq x_q\right]$$

- Goal: learn function $f(\cdot)$ for gradient-modulated Potts model



High-order P^n Potts model

Goal: learn high order CRF with potentials given by

$$h_c(\mathbf{x}) = \begin{cases} \beta_l^c & \text{if } x_p = l,\ \forall p \in c \\ \beta_{\max}^c & \text{otherwise} \end{cases},$$

[Kohli et al. CVPR07]

$$\beta_l^c = \mathbf{w}_l \cdot z_l^c$$

High-order P^n Potts model

Goal: learn high order CRF with potentials given by

$$h_c(\mathbf{x}) = \begin{cases} \beta_l^c & \text{if } x_p = l, \ \forall p \in c \\ \beta_{\max}^c & \text{otherwise} \end{cases},$$

[Kohli et al. CVPR07]

$$\beta_l^c = \mathbf{w}_l \cdot z_l^c$$

Cost for optimizing slave CRF: O(|L|) ⇨ Fast training

High-order P^n Potts model

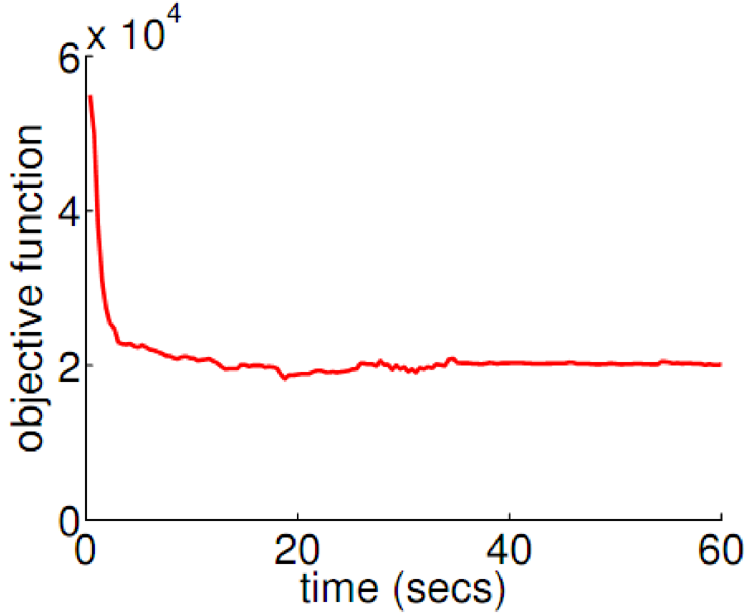
Goal: learn high order CRF with potentials given by

$$h_c(\mathbf{x}) = \begin{cases} \beta_l^c & \text{if } x_p = l, \ \forall p \in c \\ \beta_{\max}^c & \text{otherwise} \end{cases},$$

[Kohli et al. CVPR07]

$$\beta_l^c = \mathbf{w}_l \cdot z_l^c$$

Cost for optimizing slave CRF: O(|L|) ⇨ Fast training



- 100 training samples
- 50x50 grid
- clique size 3x3
- 5 labels (|L|=5)

Learning to cluster [ICCV 2011]

Clustering

- A fundamental task in vision and beyond
- Typically formulated as an optimization problem based on a given distance function between datapoints
- Choice of distance crucial for the success of clustering

Clustering

- A fundamental task in vision and beyond

- Typically formulated as an optimization problem based on a given distance function between datapoints

- Choice of distance crucial for the success of clustering

- **Goal 1:** learn this distance automatically based on training data

Clustering

- A fundamental task in vision and beyond
- Typically formulated as an optimization problem based on a given distance function between datapoints
- Choice of distance crucial for the success of clustering
- **Goal 1:** learn this distance automatically based on training data
- **Goal 2:** learning should also handle the fact that the number of clusters is typically unknown at test time

Exemplar based clustering formulation

$$\min_{Q \subseteq S} E(Q) = \sum_{p \notin Q} \min_{q \in Q} d_{p,q} + \sum_{q \in Q} d_{q,q}$$

set of datapoints

Exemplar based clustering formulation

$$\min_{Q \subseteq S} E(Q) = \sum_{p \notin Q} \min_{q \in Q} d_{p,q} + \sum_{q \in Q} d_{q,q}$$

set of exemplars
(cluster centers)

set of datapoints

Exemplar based clustering formulation

distance between
datapoints p and q

$$\min_{Q \subseteq S} E(Q) = \sum_{p \notin Q} \min_{q \in Q} d_{p,q} + \sum_{q \in Q} d_{q,q}$$

set of exemplars
(cluster centers)

set of datapoints

Exemplar based clustering formulation

distance between
datapoints p and q

penalty for choosing q as
exemplar (cluster center)

$$\min_{Q \subseteq S} E(Q) = \sum_{p \notin Q} \min_{q \in Q} d_{p,q} + \sum_{q \in Q} d_{q,q}$$

set of exemplars
(cluster centers)

set of datapoints

Exemplar based clustering formulation

distance between datapoints p and q

penalty for choosing q as exemplar (cluster center)

$$\min_{Q \subseteq S} E(Q) = \sum_{p \notin Q} \min_{q \in Q} d_{p,q} + \sum_{q \in Q} d_{q,q}$$

set of exemplars (cluster centers)

set of datapoints

The above formulation allows to:

- automatically estimate the number of clusters (i.e. size of Q)

Exemplar based clustering formulation

distance between datapoints p and q

penalty for choosing q as exemplar (cluster center)

$$\min_{Q \subseteq S} E(Q) = \sum_{p \notin Q} \min_{q \in Q} d_{p,q} + \sum_{q \in Q} d_{q,q}$$

set of exemplars (cluster centers)

set of datapoints

The above formulation allows to:

- automatically estimate the number of clusters (i.e. size of Q)
- use arbitrary distances
  (e.g., non-metric, asymmetric, non-differentiable)

Exemplar based clustering formulation

distance between datapoints p and q

penalty for choosing q as exemplar (cluster center)

$$\min_{Q \subseteq S} E(Q) = \sum_{p \notin Q} \min_{q \in Q} d_{p,q} + \sum_{q \in Q} d_{q,q}$$

set of exemplars (cluster centers)

set of datapoints

Inference can be performed efficiently using:
Clustering via LP-based Stabilities [Komodakis et al., NIPS 2008]

Exemplar based clustering as a high-order CRF

$$\min_{\mathbf{x}} \sum_{p,q \in S} d_{p,q} x_{pq}$$

$$\text{s.t.} \sum_{q \in S} x_{pq} = 1, \ \forall p$$

$$x_{pq} \leq x_{qq}, \qquad \forall p, q$$

$$x_{pq} \in \{0, 1\}, \ \forall p, q.$$

Exemplar based clustering as a high-order CRF

$x_{qq} = 1 \Leftrightarrow q$ is chosen as exemplar

$x_{pq} = 1 \Leftrightarrow p$ is assigned to exemplar q

$$\min_{\mathbf{x}} \sum_{p,q \in S} d_{p,q} x_{pq}$$

$$\text{s.t.} \sum_{q \in S} x_{pq} = 1, \quad \forall p$$

$$x_{pq} \leq x_{qq}, \qquad \forall p, q$$

$$x_{pq} \in \{0, 1\}, \quad \forall p, q.$$

Exemplar based clustering as a high-order CRF

$$\min_{\mathbf{x}} \sum_{p,q \in S} d_{p,q} x_{pq}$$

$$\text{s.t.} \sum_{q \in S} x_{pq} = 1, \quad \forall p$$

$$x_{pq} \leq x_{qq}, \qquad \forall p, q$$

$$x_{pq} \in \{0,1\}, \quad \forall p, q.$$

$x_{qq} = 1 \Leftrightarrow q$ is chosen as exemplar

$x_{pq} = 1 \Leftrightarrow p$ is assigned to exemplar q

each datapoint *p* can be assigned to exactly one exemplar

Exemplar based clustering as a high-order CRF

$$\min_{\mathbf{x}} \sum_{p,q \in S} d_{p,q} x_{pq}$$

$$\text{s.t.} \sum_{q \in S} x_{pq} = 1, \quad \forall p$$

$$x_{pq} \leq x_{qq}, \qquad \forall p, q$$

$$x_{pq} \in \{0, 1\}, \quad \forall p, q.$$

$x_{qq} = 1 \Leftrightarrow q$ is chosen as exemplar

$x_{pq} = 1 \Leftrightarrow p$ is assigned to exemplar q

each datapoint p can be assigned to exactly one exemplar

if p is assigned to q, then q must be chosen as exemplar

Exemplar based clustering as a high-order CRF

$$\min_{\mathbf{x}} \sum_{p,q \in S} d_{p,q} x_{pq}$$

$$\text{s.t.} \sum_{q \in S} x_{pq} = 1, \quad \forall p$$

$$x_{pq} \leq x_{qq}, \qquad \forall p, q$$

$$x_{pq} \in \{0, 1\}, \quad \forall p, q.$$

$x_{qq} = 1 \Leftrightarrow q$ is chosen as exemplar

$x_{pq} = 1 \Leftrightarrow p$ is assigned to exemplar q

each datapoint p can be assigned to exactly one exemplar

if p is assigned to q, then q must be chosen as exemplar

$$E(\mathbf{x}; \mathbf{d}) = \sum_{p,q} \underbrace{d_{p,q} x_{pq}}_{\text{unary terms}} + \sum_{p,q} \underbrace{\delta(x_{pq} \leq x_{qq})}_{\text{pairwise terms}} + \sum_{p} \underbrace{\delta\left(\sum_q x_{pq} = 1\right)}_{\text{higher-order terms}}$$

$$\delta(a) = \begin{cases} 0, & a \text{ is true} \\ \infty, & a \text{ is false} \end{cases}$$

Learning to cluster via high-order latent CRFs

- **Goal:** lean distances $d_{p,q}$ and penalties $d_{q,q}$

Learning to cluster via high-order latent CRFs

- **Goal:** lean distances $d_{p,q}$ and penalties $d_{q,q}$
- **Input:**
  - K training samples $\left\{\mathcal{C}^k, \mathbf{z}^k\right\}_{k=1}^K$

Learning to cluster via high-order latent CRFs

- **Goal:** lean distances $d_{p,q}$ and penalties $d_{q,q}$

- **Input:**
  - K training samples $\left\{\mathcal{C}^k, \mathbf{z}^k\right\}_{k=1}^{K}$

$$\mathcal{C}^k = \{C_i^k\}$$

$$\boxed{\text{ground truth partition of } S^k \text{ into clusters}}$$

$$\cup_i C_i^k = S^k, C_i^k \cap C_j^k = \emptyset, \forall i \neq j$$

Learning to cluster via high-order latent CRFs

- **Goal:** lean distances $d_{p,q}$ and penalties $d_{q,q}$

- **Input:**
  - K training samples $\left\{\mathcal{C}^k, \mathbf{z}^k\right\}_{k=1}^{K}$

$$\mathcal{C}^k = \{C_i^k\}$$

$\boxed{\text{ground truth partition of } S^k \text{ into clusters}}$

$$\cup_i C_i^k = S^k, C_i^k \cap C_j^k = \emptyset, \forall i \neq j$$

  - Vector valued feature function $g_{pq}(\cdot)$

$$d_{p,q}^k = \mathbf{w}^T g_{pq}(\mathbf{z}^k)$$

Learning to cluster via high-order latent CRFs

- Loss function for clustering

$$\Delta(\mathbf{x}; \mathcal{C}^k) = \alpha \sum_{C \in \mathcal{C}^k} \left| 1 - \sum_{q \in C} x_{qq} \right| + \beta \sum_{C \in \mathcal{C}^k} \sum_{p \in C} \left(1 - \sum_{q \in C} x_{pq} \right)$$

Learning to cluster via high-order latent CRFs

- Loss function for clustering

$$\Delta(\mathbf{x}; \mathcal{C}^k) = \alpha \sum_{C \in \mathcal{C}^k} \left| 1 - \sum_{q \in C} x_{qq} \right| + \beta \sum_{C \in \mathcal{C}^k} \sum_{p \in C} \left(1 - \sum_{q \in C} x_{pq} \right)$$

measures inconsistency between **x** and partition $\mathcal{C}^k$

Learning to cluster via high-order latent CRFs

- Loss function for clustering

$$\Delta(\mathbf{x};\mathcal{C}^k) = \alpha \sum_{C\in\mathcal{C}^k} \left|1 - \sum_{q\in C} x_{qq}\right| + \beta \sum_{C\in\mathcal{C}^k} \sum_{p\in C} \left(1 - \sum_{q\in C} x_{pq}\right)$$

measures inconsistency between **x** and partition $\mathcal{C}^k$

- Set of clusterings fully consistent with partition C^k

$$\mathcal{X}(\mathcal{C}^k) = \{\mathbf{x} : \Delta(\mathbf{x};\mathcal{C}^k) = 0\}$$

Learning to cluster via high-order latent CRFs

Main problems:

Learning to cluster via high-order latent CRFs

Main problems:

1. Training data do not specify ground truth solution $\boldsymbol{x}^k$ (**all** elements of $\boldsymbol{x}^k$ are **hidden/latent** in this case)

Learning to cluster via high-order latent CRFs

Main problems:

1. Training data do not specify ground truth solution $\boldsymbol{x}^k$ (**all** elements of $\boldsymbol{x}^k$ are **hidden/latent** in this case)
    - they only constraint it: $\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)$

Learning to cluster via high-order latent CRFs

Main problems:

1. Training data do not specify ground truth solution $\boldsymbol{x}^k$ (**all** elements of $\boldsymbol{x}^k$ are **hidden/latent** in this case)
   - they only constraint it: $\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)$

latent CRF model

Learning to cluster via high-order latent CRFs

Main problems:

1. Training data do not specify ground truth solution $\boldsymbol{x}^k$ (**all** elements of $\boldsymbol{x}^k$ are **hidden/latent** in this case)
   - they only constraint it: $\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)$

latent CRF model

2. Both E(.) and Δ(.) are CRF energies of very high order

Learning to cluster via high-order latent CRFs

Main problems:

1. Training data do not specify ground truth solution $\boldsymbol{x}^k$ (**all** elements of $\boldsymbol{x}^k$ are **hidden/latent** in this case)
   - they only constraint it: $\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)$

latent CRF model

2. Both $E(.)$ and $\Delta(.)$ are CRF energies of very high order

$$E(\mathbf{x};\mathbf{d}) = \sum_{p,q} \underbrace{d_{p,q} x_{pq}}_{\text{unary terms}} + \sum_{p,q} \underbrace{\delta(x_{pq} \leq x_{qq})}_{\text{pairwise terms}} + \sum_{p} \underbrace{\delta\left(\sum\nolimits_q x_{pq} = 1\right)}_{\text{higher-order terms}}$$

$$\Delta(\mathbf{x};\mathcal{C}^k) = \alpha \sum_{C \in \mathcal{C}^k} \underbrace{\left|1 - \sum_{q \in C} x_{qq}\right|}_{\text{high-order terms}} + \beta \sum_{C \in \mathcal{C}^k} \sum_{p \in C} \left(1 - \sum_{q \in C} x_{pq}\right)$$

Learning to cluster via high-order latent CRFs

How to efficiently deal with these problems during learning?

Learning to cluster via high-order latent CRFs

How to efficiently deal with these problems during learning?

Solution: CRF training via **dual decomposition** for **latent CRFs**

Learning to cluster via high-order latent CRFs

$$\bar{E}^k(\mathbf{x};\mathbf{w}) := E(\mathbf{x};\mathbf{d^k}) - \Delta(\mathbf{x};\mathcal{C}^k)$$

Learning to cluster via high-order latent CRFs

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_{p,q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \sum_{p} \bar{\phi}_p(\mathbf{x}_p) + \sum_{C \in \mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k|$$

Learning to cluster via high-order latent CRFs

$$\boxed{\left(d_{p,q} + \beta\cdot\left[\exists C \in \mathcal{C}^k : p, q \in C\right]\right)\cdot x_{pq}}$$

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_{p,q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \sum_{p} \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k|$$

Learning to cluster via high-order latent CRFs

$$\boxed{\left(d_{p,q} + \beta\cdot\left[\exists C \in \mathcal{C}^k : p, q \in C\right]\right)\cdot x_{pq}}$$

$$\boxed{\delta\left(x_{pq} \leq x_{qq}\right)}$$

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_{p,q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \sum_{p} \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k|$$

Learning to cluster via high-order latent CRFs

$$\boxed{\left(d_{p,q} + \beta \cdot \left[\exists C \in \mathcal{C}^k : p, q \in C\right]\right) \cdot x_{pq}}$$

$$\boxed{\delta\left(x_{pq} \leq x_{qq}\right)}$$

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_{p,q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \sum_{p} \bar{\phi}_p(\mathbf{x}_p) + \sum_{C \in \mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k|$$

$$\boxed{\delta\left(\sum\nolimits_q x_{pq} = 1\right)}$$

$$\mathbf{x}_p = \{x_{pq} | q \in S\}$$

Learning to cluster via high-order latent CRFs

$$\left(d_{p,q} + \beta \cdot [\exists C \in \mathcal{C}^k : p, q \in C]\right) \cdot x_{pq}$$

$$\delta\left(x_{pq} \leq x_{qq}\right)$$

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_{p,q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) +$$

$$\sum_{p} \bar{\phi}_p(\mathbf{x}_p) + \sum_{C \in \mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k|$$

$$\delta\left(\sum\nolimits_q x_{pq} = 1\right)$$

$$-\alpha\left|1 - \sum\nolimits_{q \in C} x_{qq}\right|$$

$$\mathbf{x}_p = \{x_{pq} | q \in S\}$$

$$\mathbf{x}_C = \{x_{qq} \,|\, q \in C\}$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_{p,q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \sum_{p} \bar{\phi}_p(\mathbf{x}_p) + \sum_{C \in \mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k|$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \bar{u}^k_{qq}(x_{qq}) + \sum_{p,q:p\neq q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \sum_p \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \bar{u}^k_{qq}(x_{qq}) + \sum_{p,q:p\neq q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) +$$
$$\sum_p \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,$$

One slave CRF per ground truth cluster C

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \theta^k_{Cq} x_{qq} + \bar{\phi}_C(\mathbf{x}_C)$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \boxed{\bar{u}^k_{qq}(x_{qq})} + \sum_{p,q:p\neq q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) +$$
$$\sum_p \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \boxed{\bar{\phi}_C(\mathbf{x}_C)} - \beta|S^k| \;,$$

One slave CRF per ground truth cluster C

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \boxed{\theta^k_{Cq} x_{qq}} + \boxed{\bar{\phi}_C(\mathbf{x}_C)}$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \bar{u}^k_{qq}(x_{qq}) + \sum_{p,q:p\neq q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) +$$
$$\sum_p \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,$$

One slave CRF per ground truth cluster C

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \theta^k_{Cq} x_{qq} + \bar{\phi}_C(\mathbf{x}_C)$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \bar{u}^k_{qq}(x_{qq}) + \sum_{p,q:p\neq q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \sum_p \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,$$

One slave CRF per datapoint p

$$\bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_q \theta^k_{pq} x_{qq} + \sum_{q:q\neq p} \bar{u}^k_{pq}(x_{pq}) + \sum_q \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \bar{\phi}_p(\mathbf{x}_p) - \beta$$

One slave CRF per ground truth cluster C

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \theta^k_{Cq} x_{qq} + \bar{\phi}_C(\mathbf{x}_C)$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \bar{u}^k_{qq}(x_{qq}) + \sum_{p,q:p\neq q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \boxed{\bar{\phi}_{pq}(x_{pq},x_{qq})} + \sum_p \boxed{\bar{\phi}_p(\mathbf{x}_p)} + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,$$

One slave CRF per datapoint p

$$\bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_q \theta^k_{pq} x_{qq} + \sum_{q:q\neq p} \bar{u}^k_{pq}(x_{pq}) + \sum_q \boxed{\bar{\phi}_{pq}(x_{pq},x_{qq})} + \boxed{\bar{\phi}_p(\mathbf{x}_p)} - \beta$$

One slave CRF per ground truth cluster C

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \theta^k_{Cq} x_{qq} + \bar{\phi}_C(\mathbf{x}_C)$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \boxed{\bar{u}^k_{qq}(x_{qq})} + \sum_{p,q:p\neq q} \boxed{\bar{u}^k_{pq}(x_{pq})} + \sum_{p,q} \boxed{\bar{\phi}_{pq}(x_{pq},x_{qq})} + \sum_p \boxed{\bar{\phi}_p(\mathbf{x}_p)} + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,$$

One slave CRF per datapoint p

$$\bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_q \boxed{\theta^k_{pq} x_{qq}} + \sum_{q:q\neq p} \boxed{\bar{u}^k_{pq}(x_{pq})} + \sum_q \boxed{\bar{\phi}_{pq}(x_{pq},x_{qq})} + \boxed{\bar{\phi}_p(\mathbf{x}_p)} - \beta$$

One slave CRF per ground truth cluster C

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \theta^k_{Cq} x_{qq} + \bar{\phi}_C(\mathbf{x}_C)$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \bar{u}^k_{qq}(x_{qq}) + \sum_{p,q:p\neq q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) +$$

$$\sum_p \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,$$

One slave CRF per datapoint p

$$\bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_q \theta^k_{pq} x_{qq} + \sum_{q:q\neq p} \bar{u}^k_{pq}(x_{pq}) + \sum_q \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \bar{\phi}_p(\mathbf{x}_p) - \beta$$

One slave CRF per ground truth cluster C

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \theta^k_{Cq} x_{qq} + \bar{\phi}_C(\mathbf{x}_C)$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \bar{u}^k_{qq}(x_{qq}) + \sum_{p,q:p\neq q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) +$$
$$\sum_p \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,$$

+ +

One slave CRF per datapoint p

$$\bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_q \theta^k_{pq} x_{qq} + \sum_{q:q\neq p} \bar{u}^k_{pq}(x_{pq}) + \sum_q \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \bar{\phi}_p(\mathbf{x}_p) - \beta$$

One slave CRF per ground truth cluster C

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \theta^k_{Cq} x_{qq} + \bar{\phi}_C(\mathbf{x}_C)$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \bar{u}^k_{qq}(x_{qq}) + \sum_{p,q:p\neq q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) +$$

$$\sum_p \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,$$

$+ \quad +$

One slave CRF per datapoint p

$$\bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_q \theta^k_{pq} x_{qq} + \sum_{q:q\neq p} \bar{u}^k_{pq}(x_{pq}) + \sum_q \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \bar{\phi}_p(\mathbf{x}_p) - \beta$$

One slave CRF per ground truth cluster C

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \theta^k_{Cq} x_{qq} + \bar{\phi}_C(\mathbf{x}_C)$$

$$\left(\sum_p \theta^k_{pq} + \theta^k_{Cq}\right) x_{qq} = \bar{u}^k_{qq}(x_{qq}), \ \forall C \in \mathcal{C}^k, q \in C$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \bar{u}^k_{qq}(x_{qq}) + \sum_{p,q:p\neq q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) +$$
$$\sum_p \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,$$

$+ \quad +$

One slave CRF per datapoint p

$$\bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_q \theta^k_{pq} x_{qq} + \sum_{q:q\neq p} \bar{u}^k_{pq}(x_{pq}) + \sum_q \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \bar{\phi}_p(\mathbf{x}_p) - \beta$$

One slave CRF per ground truth cluster C

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \theta^k_{Cq} x_{qq} + \bar{\phi}_C(\mathbf{x}_C)$$

$$\sum_p \theta^k_{pq} + \theta^k_{Cq} = d_{q,q} + \beta, \ \forall C \in \mathcal{C}^k, q \in C$$

Choosing decomposition for clustering

$$\bar{E}^k(\mathbf{x};\mathbf{w}) = \sum_q \bar{u}^k_{qq}(x_{qq}) + \sum_{p,q:p\neq q} \bar{u}^k_{pq}(x_{pq}) + \sum_{p,q} \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \sum_p \bar{\phi}_p(\mathbf{x}_p) + \sum_{C\in\mathcal{C}^k} \bar{\phi}_C(\mathbf{x}_C) - \beta|S^k| \ ,$$

$+ \qquad +$

One slave CRF per datapoint p

$$\bar{E}^k_p(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_q \theta^k_{pq} x_{qq} + \sum_{q:q\neq p} \bar{u}^k_{pq}(x_{pq}) + \sum_q \bar{\phi}_{pq}(x_{pq}, x_{qq}) + \bar{\phi}_p(\mathbf{x}_p) - \beta$$

One slave CRF per ground truth cluster C

$$\bar{E}^k_C(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \theta^k_{Cq} x_{qq} + \bar{\phi}_C(\mathbf{x}_C)$$

$$\boldsymbol{\theta}^k \in \Theta^k = \left\{ \boldsymbol{\theta}^k : \sum_p \theta^k_{pq} + \theta^k_{Cq} = d_{q,q} + \beta,\ \forall C \in \mathcal{C}^k, q \in C \right\}$$

Learning to cluster via high-order latent CRFs

$$\min_{\{\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)\},\mathbf{w},\{\boldsymbol{\theta}^k \in \boldsymbol{\Theta}^k\}} R(\mathbf{w}) + \sum_k \sum_{p \in S^k} \mathcal{L}_{\bar{E}_p^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) + \sum_k \sum_{C \in \mathcal{C}^k} \mathcal{L}_{\bar{E}_C^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathcal{L}_{\bar{E}}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) := \bar{E}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) - \min_{\mathbf{x}} \bar{E}(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}^k)$$

Learning to cluster via high-order latent CRFs

$$\min_{\{\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)\}, \mathbf{w}, \{\boldsymbol{\theta}^k \in \boldsymbol{\Theta}^k\}} R(\mathbf{w}) + \sum_k \sum_{p \in S^k} \mathcal{L}_{\bar{E}_p^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) + \sum_k \sum_{C \in \mathcal{C}^k} \mathcal{L}_{\bar{E}_C^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathcal{L}_{\bar{E}}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) := \bar{E}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) - \min_{\mathbf{x}} \bar{E}(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}^k)$$

- Use block coordinate descent
- Alternately optimize
  a. $\{\mathbf{x}^k\}$
  b. $\left\{\mathbf{w}, \{\boldsymbol{\theta}^k \in \boldsymbol{\Theta}^k\}\right\}$

Optimizing over $\{\mathbf{x}^k\}$

$$\min_{\{\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)\}, \cancel{\mathbf{w}, \{\boldsymbol{\theta}^k \in \Theta^k\}}} R(\mathbf{w}) + \sum_k \sum_{p \in S^k} \mathcal{L}_{\bar{E}_p^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) + \sum_k \sum_{C \in \mathcal{C}^k} \mathcal{L}_{\bar{E}_C^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathcal{L}_{\bar{E}}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) := \bar{E}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) - \min_{\mathbf{x}} \bar{E}(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}^k)$$

Optimizing over $\{\mathbf{x}^k\}$

$$\min_{\{\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)\}, \cancel{\mathbf{w}, \{\boldsymbol{\theta}^k \in \Theta^k\}}} R(\mathbf{w}) + \sum_k \sum_{p \in S^k} \mathcal{L}_{\bar{E}_p^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) + \sum_k \sum_{C \in \mathcal{C}^k} \mathcal{L}_{\bar{E}_C^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathcal{L}_{\bar{E}}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) := \bar{E}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) - \min_{\mathbf{x}} \bar{E}(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathbf{x}^k = \arg \min_{\mathbf{x} \in \mathcal{X}(\mathcal{C}^k)} E(\mathbf{x}; \mathbf{d}^k)$$

Optimizing over $\{\mathbf{x}^k\}$

$$\min_{\{\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)\}, \mathbf{w}, \cancel{\{\boldsymbol{\theta}^k \in \Theta^k\}}} R(\mathbf{w}) + \sum_k \sum_{p \in S^k} \mathcal{L}_{\bar{E}_p^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) + \sum_k \sum_{C \in \mathcal{C}^k} \mathcal{L}_{\bar{E}_C^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathcal{L}_{\bar{E}}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) := \bar{E}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) - \min_{\mathbf{x}} \bar{E}(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathbf{x}^k = \arg \min_{\mathbf{x} \in \mathcal{X}(\mathcal{C}^k)} E(\mathbf{x}; \mathbf{d}^k)$$

optimal cluster centers (exemplars)

$$Q^k = \{q_C\}_{C \in \mathcal{C}^k} \qquad q_C = \arg\min_{q \in C} \sum_{p \in C} d_{p,q}^k$$

Optimizing over $\{\mathbf{x}^k\}$

$$\min_{\{\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)\},\mathbf{w},\cancel{\{\boldsymbol{\theta}^k \in \Theta^k\}}} R(\mathbf{w}) + \sum_k \sum_{p \in S^k} \mathcal{L}_{\bar{E}_p^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) + \sum_k \sum_{C \in \mathcal{C}^k} \mathcal{L}_{\bar{E}_C^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathcal{L}_{\bar{E}}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) := \bar{E}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) - \min_{\mathbf{x}} \bar{E}(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathbf{x}^k = \arg \min_{\mathbf{x} \in \mathcal{X}(\mathcal{C}^k)} E(\mathbf{x}; \mathbf{d}^k)$$

optimal cluster centers (exemplars)

$$Q^k = \{q_C\}_{C \in \mathcal{C}^k} \qquad q_C = \arg \min_{q \in C} \sum_{p \in C} d_{p,q}^k$$

$$x_{qq}^k = 1 \Leftrightarrow q \in Q^k \qquad x_{pq}^k = 1 \Leftrightarrow q = \arg \min\nolimits_{q \in Q^k} d_{p,q}^k$$

Optimizing over $\{\mathbf{w}, \{\boldsymbol{\theta}^k \in \boldsymbol{\Theta}^k\}\}$

$$\min_{\cancel{\{\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)\}}, \mathbf{w}, \{\boldsymbol{\theta}^k \in \boldsymbol{\Theta}^k\}} R(\mathbf{w}) + \sum_k \sum_{p \in S^k} \mathcal{L}_{\bar{E}_p^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) + \sum_k \sum_{C \in \mathcal{C}^k} \mathcal{L}_{\bar{E}_C^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathcal{L}_{\bar{E}}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) := \bar{E}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) - \min_{\mathbf{x}} \bar{E}(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}^k)$$

Optimizing over $\{\mathbf{w}, \{\theta^k \in \Theta^k\}\}$

$$\min_{\cancel{\{\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)\}},\mathbf{w},\{\boldsymbol{\theta}^k \in \boldsymbol{\Theta}^k\}} R(\mathbf{w}) + \sum_k \sum_{p \in S^k} \mathcal{L}_{\bar{E}_p^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) + \sum_k \sum_{C \in \mathcal{C}^k} \mathcal{L}_{\bar{E}_C^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathcal{L}_{\bar{E}}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) := \bar{E}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) - \min_{\mathbf{x}} \bar{E}(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}^k)$$

$\{\mathbf{x}^k\}$ is known

- Back to **fully supervised** learning

Optimizing over $\{\mathbf{w}, \{\theta^k \in \boldsymbol{\Theta}^k\}\}$

$$\min_{\cancel{\{\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)\}}, \mathbf{w}, \{\boldsymbol{\theta}^k \in \boldsymbol{\Theta}^k\}} R(\mathbf{w}) + \sum_k \sum_{p \in S^k} \mathcal{L}_{\bar{E}_p^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) + \sum_k \sum_{C \in \mathcal{C}^k} \mathcal{L}_{\bar{E}_C^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathcal{L}_{\bar{E}}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) := \bar{E}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) - \min_{\mathbf{x}} \bar{E}(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}^k)$$

$\{\mathbf{x}^k\}$ is known

- Back to **fully supervised** learning
- As already explained, in this case training requires solving the slave CRFs

Optimizing over $\{\mathrm{w}, \{\theta^k \in \Theta^k\}\}$

$$\min_{\cancel{\{\mathbf{x}^k \in \mathcal{X}(\mathcal{C}^k)\}}, \mathbf{w}, \{\boldsymbol{\theta}^k \in \boldsymbol{\Theta}^k\}} R(\mathbf{w}) + \sum_k \sum_{p \in S^k} \mathcal{L}_{\bar{E}_p^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) + \sum_k \sum_{C \in \mathcal{C}^k} \mathcal{L}_{\bar{E}_C^k}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k)$$

$$\mathcal{L}_{\bar{E}}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) := \bar{E}(\mathbf{x}^k; \mathbf{w}, \boldsymbol{\theta}^k) - \min_{\mathbf{x}} \bar{E}(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}^k)$$

$\{\mathbf{x}^k\}$ is known

- Back to **fully supervised** learning
- As already explained, in this case training requires **solving the slave CRFs**

Solving slave CRF $\bar{E}_C^k$

Solving slave CRF $\bar{E}_C^k$

$$\bar{E}_C^k(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_{q\in C} \theta_{Cq}^k x_{qq} - a \cdot \left|1 - \sum_{q\in C} x_{qq}\right|$$

Solving slave CRF $\bar{E}_C^k$

$$\bar{E}_C^k(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}^k) = \sum_{q \in C} \theta_{Cq}^k x_{qq} - a \cdot \left| 1 - \sum_{q \in C} x_{qq} \right|$$

$$\Downarrow$$

$$\forall q \in C,\ \hat{x}_{qq} = \begin{cases} [\theta_q^k < \alpha], & \textit{if } 2\alpha + \sum_{q' \in C} [\theta_{q'}^k - \alpha]_- < 0 \\ 0, & \textit{otherwise} \end{cases}$$

Solving slave CRF $\bar{E}_p^k$

Solving slave CRF $\bar{E}_p^k$

$$\bar{E}_p^k(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_q \theta_{pq}^k x_{qq} + \sum_{q:q\neq p} \bar{u}_{pq}^k(1) x_{pq} + \sum_q \delta(x_{pq} \leq x_{qq}) + \delta\left(\sum_q x_{pq} = 1\right) - \beta$$

Solving slave CRF $\bar{E}_p^k$

$$\bar{E}_p^k(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_q \theta_{pq}^k x_{qq} + \sum_{q:q\neq p} \bar{u}_{pq}^k(1) x_{pq} + \sum_q \delta(x_{pq} \leq x_{qq}) + \delta\left(\sum_q x_{pq} = 1\right) - \beta$$

$$\Downarrow$$

$$\textit{define}\ \ \bar{\theta}_q^k \equiv \bar{u}_{pq}^k(1) + [\theta_q^k]_+, \forall q \neq p \ \textit{and}\ \bar{\theta}_p^k = \theta_p^k$$

Solving slave CRF $\bar{E}_p^k$

$$\bar{E}_p^k(\mathbf{x};\mathbf{w},\boldsymbol{\theta}^k) = \sum_q \theta_{pq}^k x_{qq} + \sum_{q:q\neq p} \bar{u}_{pq}^k(1) x_{pq} + \sum_q \delta(x_{pq} \leq x_{qq}) + \delta\left(\sum_q x_{pq} = 1\right) - \beta$$

$$\Downarrow$$

$$\textit{define} \ \ \bar{\theta}_q^k \equiv \bar{u}_{pq}^k(1) + [\theta_q^k]_+, \forall q \neq p \ \textit{and} \ \bar{\theta}_p^k = \theta_p^k$$

$$\Downarrow$$

$$\forall q \neq p, \ \hat{x}_{qq} \leftarrow [\theta_q^k < 0]$$

$$\forall q, \ \hat{x}_{pq} \leftarrow [q = \bar{q}], \ \textit{where} \ \bar{q} = \arg\min_q \bar{\theta}_q^k$$

Learning scheme

Data: training samples $\{\mathcal{C}^k, \mathbf{z}^k\}_{k=1}^{K}$, features $\{f_{pq}(\cdot)\}$

repeat

/* Optimize over $\mathbf{x}^k$ */

compute optimal set of exemplars Q^k

set $x_{qq}^k = 1 \Leftrightarrow q \in Q^k, \quad x_{pq}^k = 1 \Leftrightarrow q = \arg\min_{q \in Q^k} d_{p,q}^k, \ \forall p \neq q$

/* Apply T rounds of projected subgradient */

repeat T times $\{$

get solutions $\hat{\mathbf{x}}^{k,p}, \hat{\mathbf{x}}^{k,C}$ of slaves $\bar{E}_p^k, \bar{E}_C^k$ to estimate subgradient

update $\mathbf{w}, \boldsymbol{\theta}^k$ via projected subgradient update

$\}$

until convergence

Training high-order latent CRFs via dual decomposition

- More generally, dual decomposition can be used for training any **high-order latent** model

Training high-order latent CRFs via dual decomposition

- More generally, dual decomposition can be used for training any **high-order latent** model

- K training samples $\{\tilde{\mathbf{x}}^k, \mathbf{z}^k\}_{k=1}^K$

Training high-order latent CRFs via dual decomposition

- More generally, dual decomposition can be used for training any **high-order latent** model

observed variables (per sample)

- K training samples $\{\tilde{\mathbf{x}}^k, \mathbf{z}^k\}_{k=1}^K$

Training high-order latent CRFs via dual decomposition

- More generally, dual decomposition can be used for training any **high-order latent** model

observed variables (per sample)

- K training samples $\{\tilde{\mathbf{x}}^k, \mathbf{z}^k\}_{k=1}^K$

hidden variables

$$\mathrm{MRF}_G((\mathbf{x}, \tilde{\mathbf{x}}); \mathbf{u}^k, \mathbf{h}^k) = \sum_p u_p^k\big((x_p, \tilde{x}_p)\big) + \sum_c h_c^k\big((\mathbf{x}_c, \tilde{\mathbf{x}}_c)\big)$$

Training high-order latent CRFs via dual decomposition

- More generally, dual decomposition can be used for training any **high-order latent** model

observed variables (per sample)

- K training samples $\{\tilde{\mathbf{x}}^k, \mathbf{z}^k\}_{k=1}^K$

hidden variables

$$\mathrm{MRF}_G((\mathbf{x}, \tilde{\mathbf{x}}); \mathbf{u}^k, \mathbf{h}^k) = \sum_p u_p^k\big((x_p, \tilde{x}_p)\big) + \sum_c h_c^k\big((\mathbf{x}_c, \tilde{\mathbf{x}}_c)\big)$$

$$u_p^k\big((x_p, \tilde{x}_p)\big) = \mathbf{w}^T g_p((x_p, \tilde{x}_p), \mathbf{z}^k)$$

$$h_c^k\big((\mathbf{x}_c, \tilde{\mathbf{x}}_c)\big) = \mathbf{w}^T g_c((\mathbf{x}_c, \tilde{\mathbf{x}}_c), \mathbf{z}^k)$$

vector valued feature functions

Learning a weighted Euclidean distance

- We consider a weighted Euclidean distance d_{pq} for D-dimensional datapoints

$$d_{pq} = \sum_{i=1}^{D} w_i (x_p^i - x_q^i)^2$$

Learning a weighted Euclidean distance

- We consider a weighted Euclidean distance d_{pq} for D-dimensional datapoints

$$d_{pq} = \sum_{i=1}^{D} w_i (x_p^i - x_q^i)^2$$

- **<u>Half</u>** of the D dimensions are assumed to be **<u>noisy</u>**

Learning a weighted Euclidean distance

- We consider a weighted Euclidean distance d_{pq} for D-dimensional datapoints

  $d_{pq} = \sum_{i=1}^{D} w_i (x_p^i - x_q^i)^2$

- **<u>Half</u>** of the D dimensions are assumed to be **<u>noisy</u>**

- **Goal:** learn weights w_i automatically from clustering data

Learning a weighted Euclidean distance

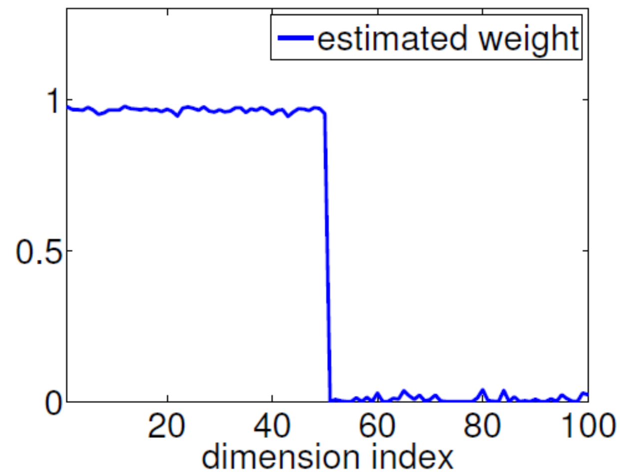
$$d_{pq} = \sum_{i=1}^{D} w_i (x_p^i - x_q^i)^2$$

$\boxed{D = 100}$

Learning a weighted Euclidean distance

$$d_{pq} = \sum_{i=1}^{D} w_i (x_p^i - x_q^i)^2$$

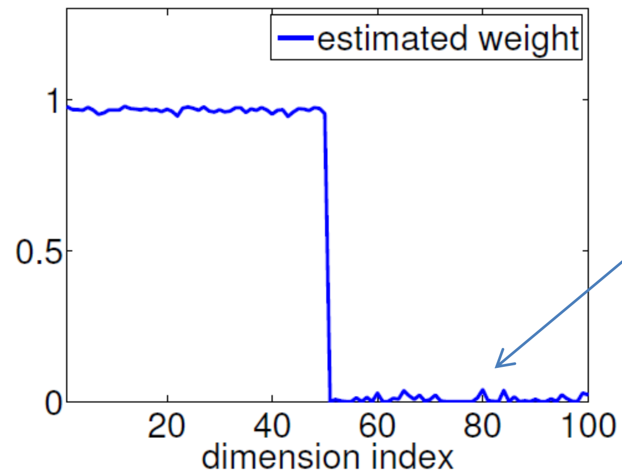
$\boxed{D = 100}$



Learning a weighted Euclidean distance

$$d_{pq} = \sum_{i=1}^{D} w_i (x_p^i - x_q^i)^2$$

$\boxed{D = 100}$

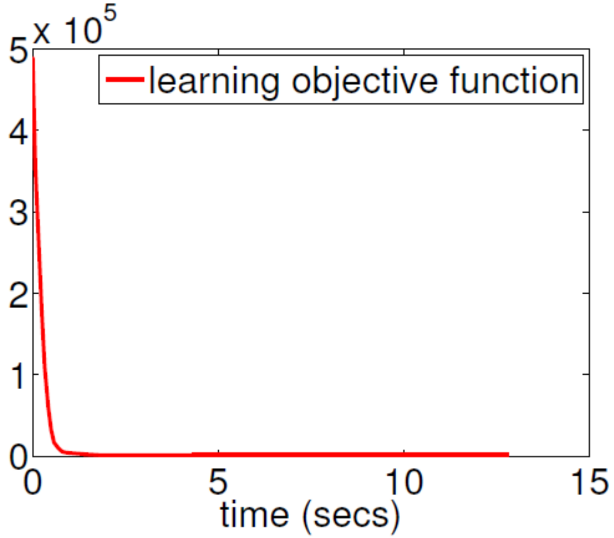


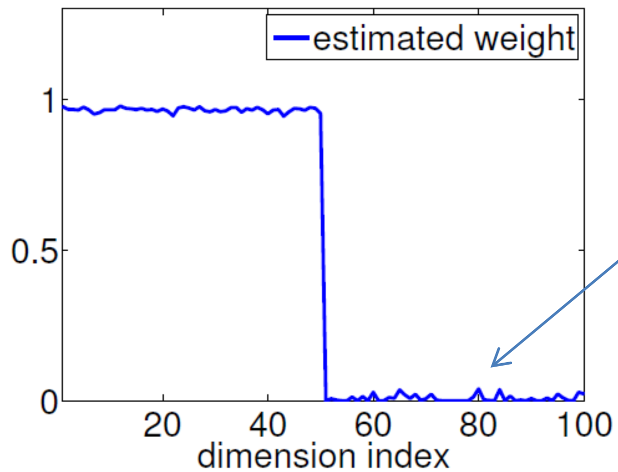
noisy dimensions get suppressed weights after training

Learning a weighted Euclidean distance

$$d_{pq} = \sum_{i=1}^{D} w_i (x_p^i - x_q^i)^2$$

$\boxed{D = 100}$



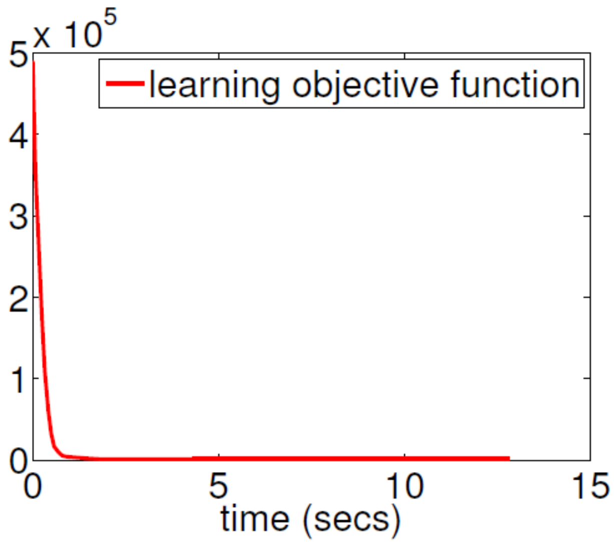


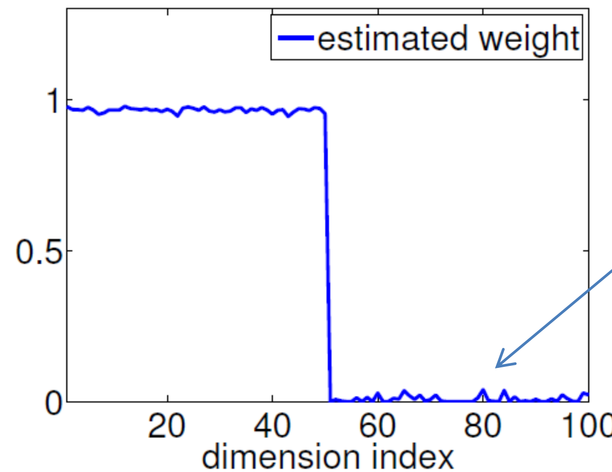
noisy dimensions get suppressed weights after training

Learning a weighted Euclidean distance

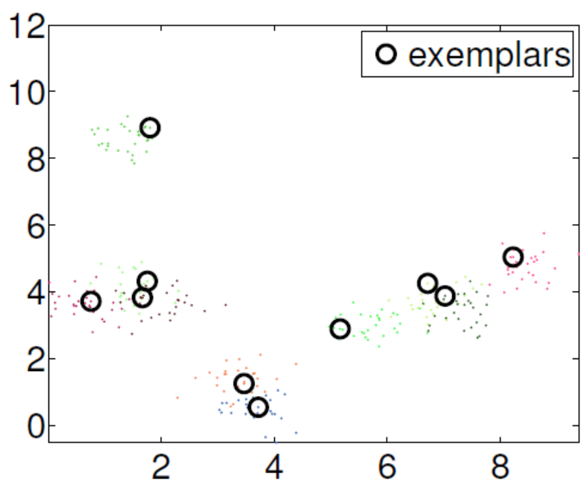
$$d_{pq} = \sum_{i=1}^{D} w_i (x_p^i - x_q^i)^2$$

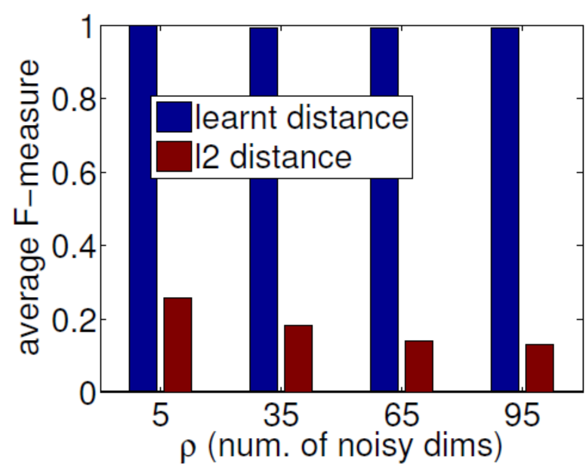
$\boxed{D = 100}$





noisy dimensions get suppressed weights after training



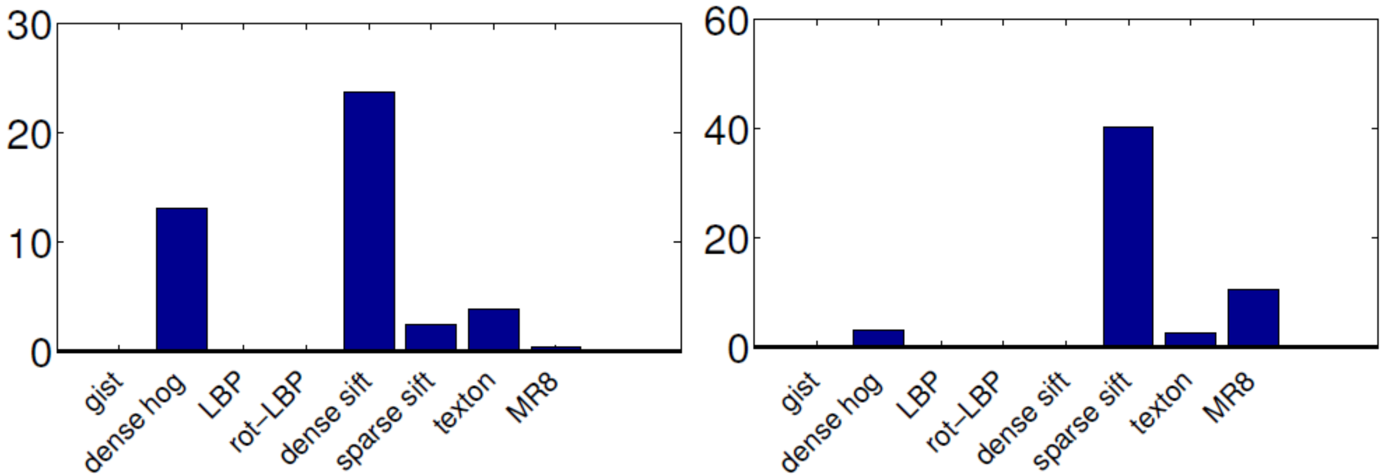


Learning to cluster texture images

Learn weighted comb. of distances between features: $d(\cdot) = \sum_f w_f d^f(\cdot)$

Learning to cluster texture images

Learn weighted comb. of distances between features: $d(\cdot) = \sum_f w_f d^f(\cdot)$



(a) Outex (b) UIUC

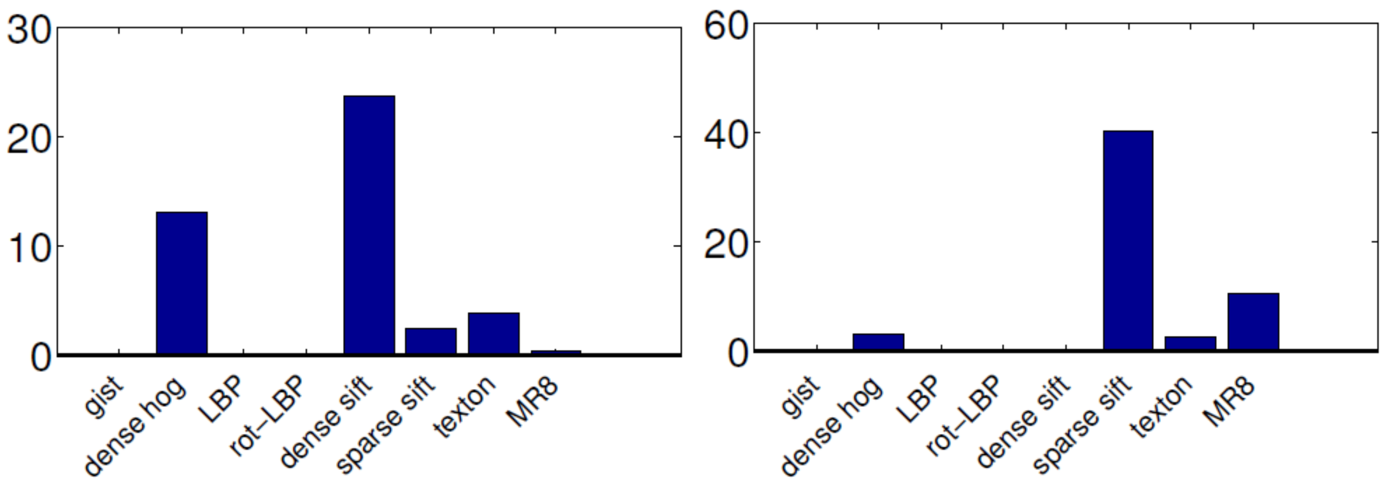
clustering accuracy:

- 100% (Outex)
- 86% (UIUC)

learnt weights

Learning to cluster texture images

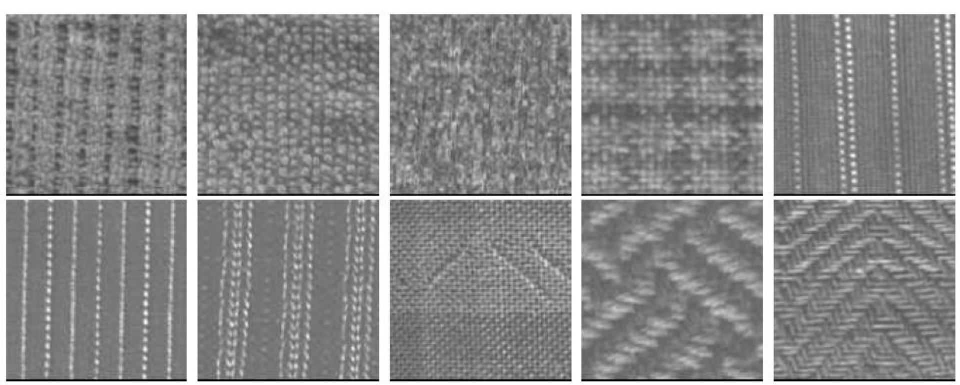
Learn weighted comb. of distances between features: $d(\cdot) = \sum_f w_f d^f(\cdot)$



(a) Outex (b) UIUC

clustering accuracy:

- 100% (Outex)
- 86% (UIUC)



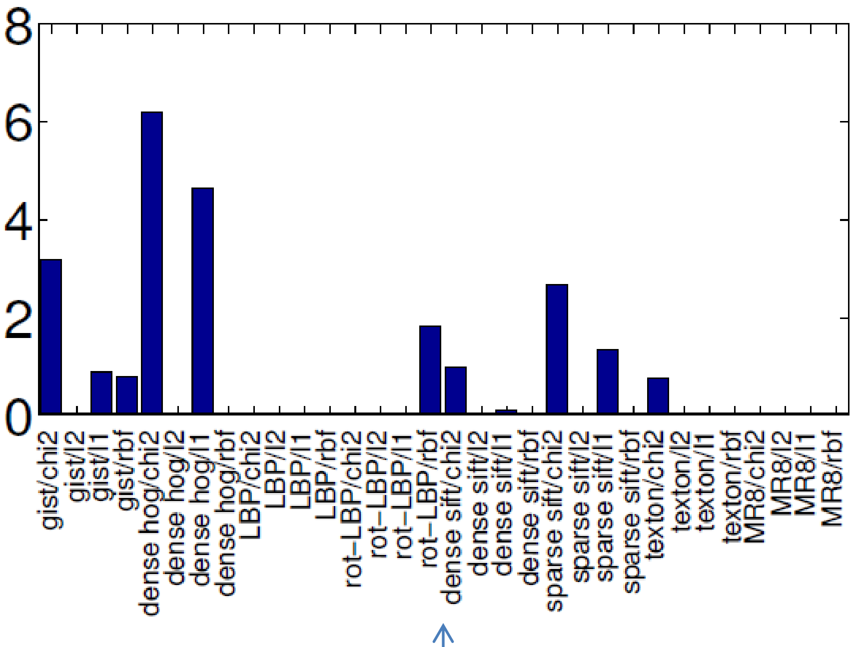
10 of the estimated exemplars for Outex

Learning to cluster scene images

Learn weighted combination of distances (multiple distances per feature, multiple features)

Learning to cluster scene images

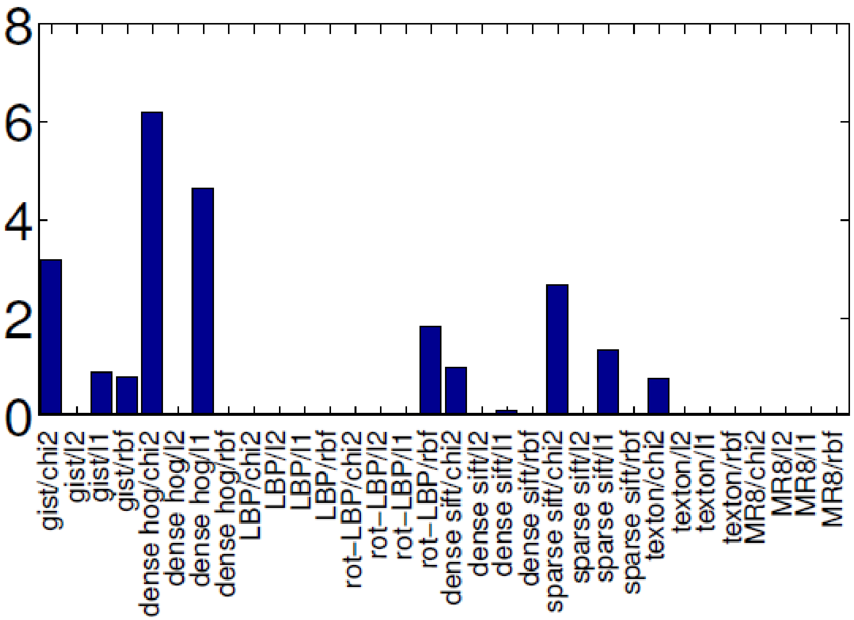
Learn weighted combination of distances (multiple distances per feature, multiple features)

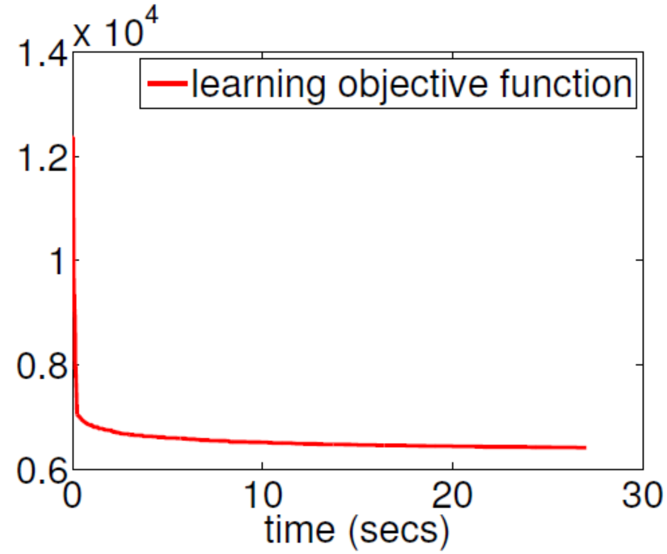


learnt weights

Learning to cluster scene images

Learn weighted combination of distances (multiple distances per feature, multiple features)

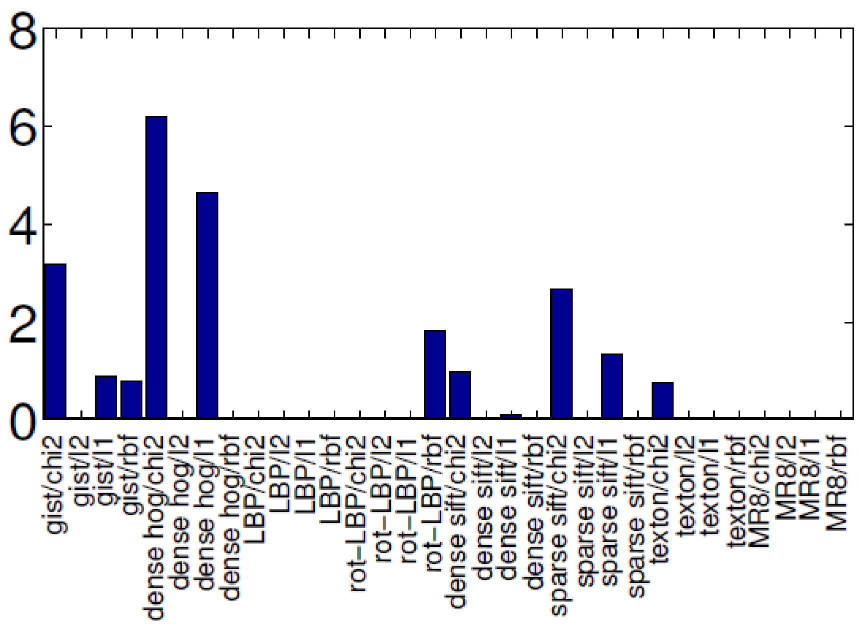


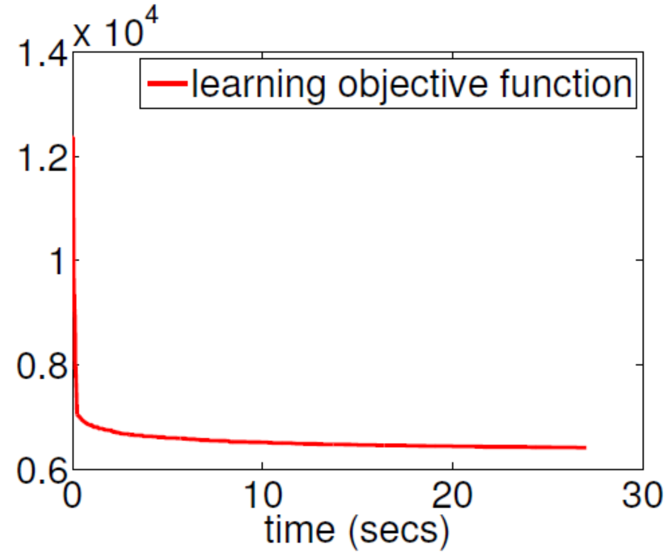


learnt weights

Learning to cluster scene images

Learn weighted combination of distances (multiple distances per feature, multiple features)





clustering accuracy: 63% (Scene)

10 of the estimated exemplars for Outex

Thank you for your attention!

Questions?