Visual Recognition: Inference in Graphical Models

Raquel Urtasun

TTI Chicago

March 1, 2012

Graphical models

- Applications
- Representation
- Inference
  - message passing (LP relaxations)
  - graph cuts
- Learning

Inference with graph cuts

St-mincut and Energy Minimization

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

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

$\Updownarrow$ Equivalent (transformable)

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

[Source: P. Kohli]

How are they equivalent?

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

$$\begin{array}{cc|c|c|} & & \multicolumn{2}{c}{x_j} \\ & & 0 & 1 \\ \hline x_i & 0 & A & B \\ \hline & 1 & C & D \\ \hline \end{array} = A + \begin{array}{c|c|c|} & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & C-A & C-A \\ \hline \end{array} + \begin{array}{c|c|c|} & 0 & 1 \\ \hline 0 & 0 & D-C \\ \hline 1 & 0 & D-C \\ \hline \end{array} + \begin{array}{c|c|c|} & 0 & 1 \\ \hline 0 & 0 & B+C-A-D \\ \hline 1 & 0 & 0 \\ \hline \end{array}$$

if x_1=1 add C-A

if x_2 = 1 add D-C

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

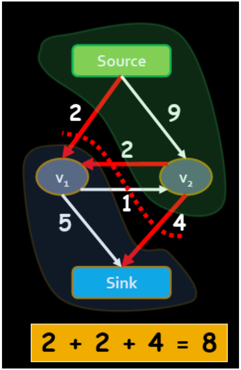
$B+C-A-D \geq 0$ is true from the submodularity of θ_{ij}

[Source: P. Kohli]

Our energy minimization

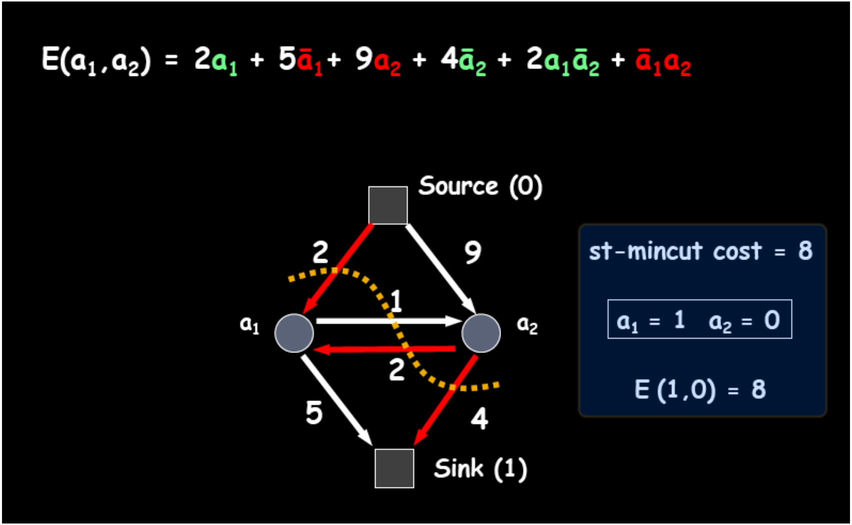
Construct a graph such that

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



[Source: P. Kohli]

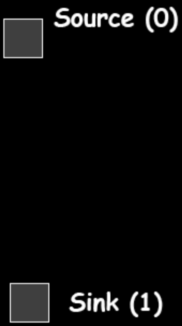
Graph Construction



[Source: P. Kohli]

How does the code look like

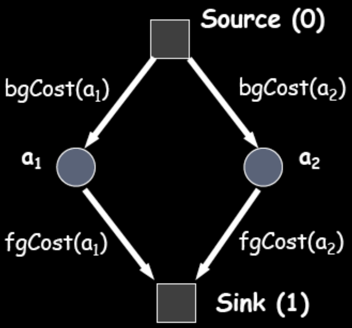
```
Graph *g;

For all pixels p

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

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

end

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

g->compute_maxflow();

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



[Source: P. Kohli]

How does the code look like

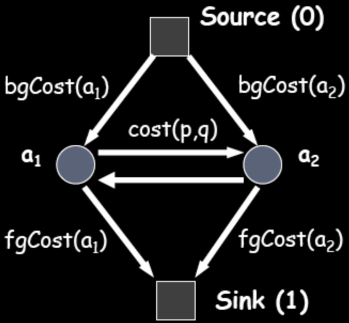
```
Graph *g;
For all pixels p

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

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

end

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

g->compute_maxflow();

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



[Source: P. Kohli]

How does the code look like

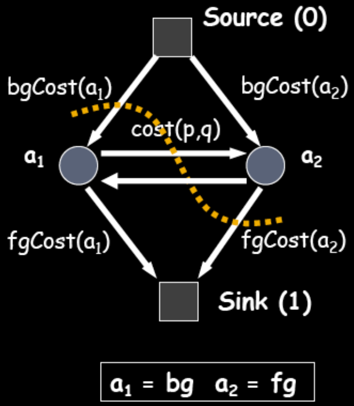
```
Graph *g;

For all pixels p

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

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

end

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

g->compute_maxflow();

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



[Source: P. Kohli]

How does the code look like

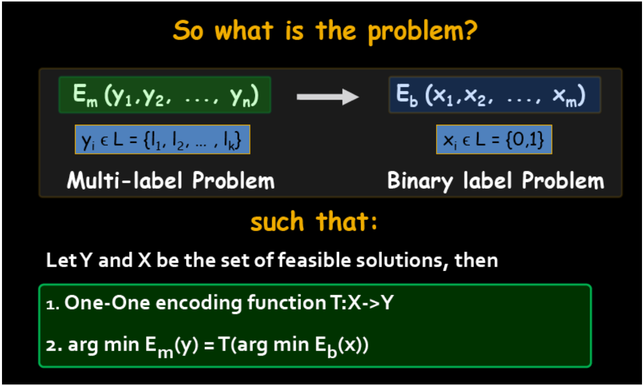
```
Graph *g;

For all pixels p

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

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

end

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

g->compute_maxflow();

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



[Source: P. Kohli]

Graph cuts for multi-label problems

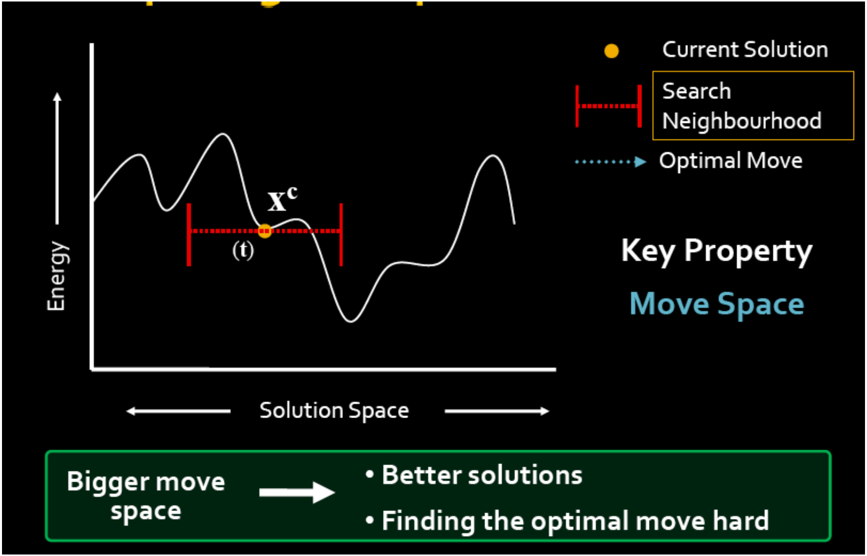
- Exact Transformation to QPBF [Roy and Cox 98] [Ishikawa 03] [Schlesinger et al. 06] [Ramalingam et al. 08]



- Very high computational cost

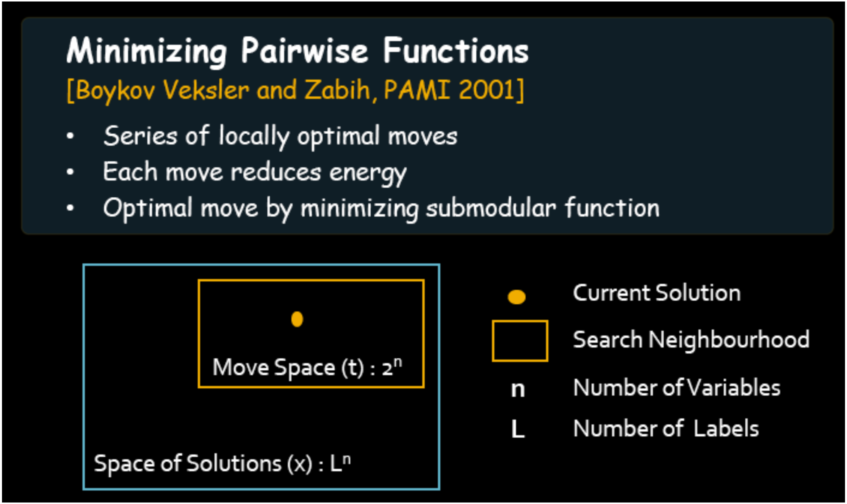
[Source: P. Kohli]

Computing the Optimal Move



[Source: P. Kohli]

Move Making Algorithms



[Source: P. Kohli]

Energy Minimization

- Consider pairwise MRFs

$$E(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) + \sum_p D_p(f_p)$$

with $\mathcal{N}$ defining the interactions between nodes, e.g., pixels

- D_p non-negative, but arbitrary.

Energy Minimization

- Consider pairwise MRFs

$$E(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) + \sum_p D_p(f_p)$$

  with $\mathcal{N}$ defining the interactions between nodes, e.g., pixels

- D_p non-negative, but arbitrary.
- Same as before, where $V_{p,q} \equiv -\theta_\alpha$ and $D_p \equiv -\theta_p$.

Energy Minimization

- Consider pairwise MRFs

$$E(f) = \sum_{\{p,q\}\in\mathcal{N}} V_{p,q}(f_p, f_q) + \sum_p D_p(f_p)$$

  with $\mathcal{N}$ defining the interactions between nodes, e.g., pixels

- D_p non-negative, but arbitrary.
- Same as before, where $V_{p,q} \equiv -\theta_\alpha$ and $D_p \equiv -\theta_p$.
- This is the graph-cuts notation.
- Important to notice it's the same thing.

Energy Minimization

- Consider pairwise MRFs

$$E(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) + \sum_p D_p(f_p)$$

  with $\mathcal{N}$ defining the interactions between nodes, e.g., pixels

- D_p non-negative, but arbitrary.
- Same as before, where $V_{p,q} \equiv -\theta_\alpha$ and $D_p \equiv -\theta_p$.
- This is the graph-cuts notation.
- Important to notice it's the same thing.

Metric vs Semimetric

Two general classes of pairwise interactions

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

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

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

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

Metric vs Semimetric

Two general classes of pairwise interactions

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

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

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

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

Binary Moves

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

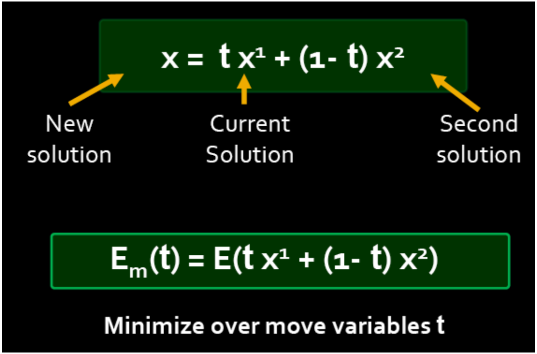


Figure: Figure from P. Kohli tutorial on graph-cuts

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

Swap Move

[Source: P. Kohli]

Swap Move

- Variables labeled α, β can swap their labels

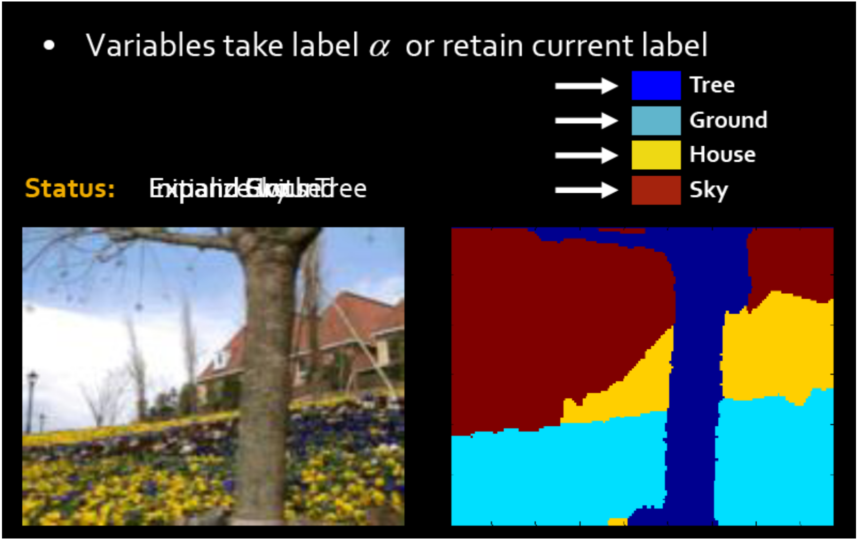
- Move energy is submodular if:
  - Unary Potentials: Arbitrary
  - Pairwise potentials: Semi-metric

$$\boxed{\begin{array}{c} \theta_{ij}(l_a, l_b) \geq 0 \\ \theta_{ij}(l_a, l_b) = 0 \iff a = b \end{array}}$$

Examples: Potts model, Truncated Convex

[Source: P. Kohli]

Expansion Move



[Source: P. Kohli]

Expansion Move

- Variables take label α or retain current label

- Move energy is submodular if:
  - Unary Potentials: Arbitrary
  - Pairwise potentials: Metric

Semi metric
+
Triangle Inequality

$$\boxed{\theta_{ij}(l_a, l_b) + \theta_{ij}(l_b, l_c) \geq \theta_{ij}(l_a, l_c)}$$

Examples: Potts model, Truncated linear

Cannot solve truncated quadratic

[Source: P. Kohli]

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.
- Given a label l, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}'$ (labeling f') is called an α-**expansion** if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.
- Given a label l, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}'$ (labeling f') is called an α-**expansion** if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$.
- An α-**expansion** move allows any set of image pixels to change their labels to α.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.
- Given a label l, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}'$ (labeling f') is called an α-**expansion** if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$.
- An α-**expansion** move allows any set of image pixels to change their labels to α.

Example

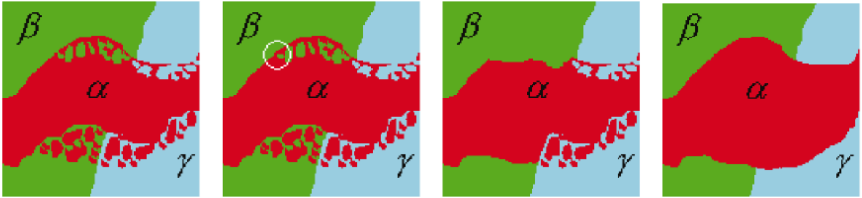


Figure: (a) Current partition (b) local move (c) $\alpha - \beta$-swap (d) α-expansion.

Algorithms

1. Start with an arbitrary labeling f
2. Set success := 0
3. For each pair of labels $\{\alpha, \beta\} \subset \mathcal{L}$
   3.1. Find $\hat{f} = \arg\min E(f')$ among f' within one α-β swap of f
   3.2. If $E(\hat{f}) < E(f)$, set f := $\hat{f}$ and success := 1
4. If success = 1 goto 2
5. Return f

---

1. Start with an arbitrary labeling f
2. Set success := 0
3. For each label $\alpha \in \mathcal{L}$
   3.1. Find $\hat{f} = \arg\min E(f')$ among f' within one α-expansion of f
   3.2. If $E(\hat{f}) < E(f)$, set f := $\hat{f}$ and success := 1
4. If success = 1 goto 2
5. Return f

Finding optimal Swap move

- Given an input labeling f (partition $\mathcal{P}$) and a pair of labels α, β we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - \beta$-swap of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_{\alpha\beta} = (\mathcal{V}_{\alpha\beta}, \mathcal{E}_{\alpha\beta})$.

Finding optimal Swap move

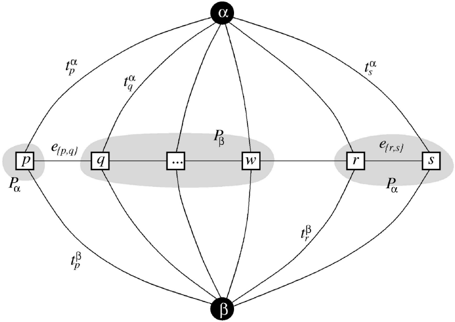
- Given an input labeling f (partition $\mathcal{P}$) and a pair of labels α, β we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - \beta$-swap of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_{\alpha\beta} = (\mathcal{V}_{\alpha\beta}, \mathcal{E}_{\alpha\beta})$.
- The structure of this graph is dynamically determined by the current partition $\mathcal{P}$ and by the labels α, β.

Finding optimal Swap move

- Given an input labeling f (partition $\mathcal{P}$) and a pair of labels α, β we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - \beta$-swap of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_{\alpha\beta} = (\mathcal{V}_{\alpha\beta}, \mathcal{E}_{\alpha\beta})$.
- The structure of this graph is dynamically determined by the current partition $\mathcal{P}$ and by the labels α, β.

Graph Construction

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



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

Computing the Cut

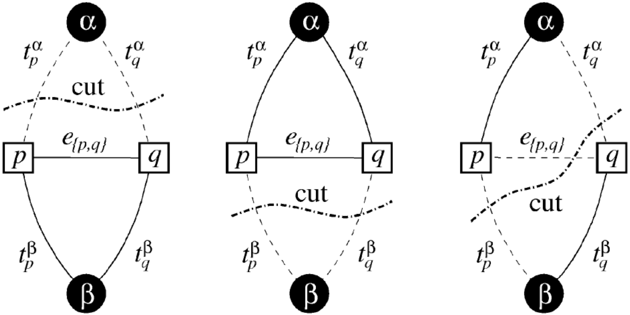
- Any cut must have a single *t*-link not cut.
- This defines a labeling

$$f_p^{\mathcal{C}} = \begin{cases} \alpha & \text{if } \ t_p^{\alpha} \in \mathcal{C} \ \text{ for } \ p \in \mathcal{P}_{\alpha\beta} \\ \beta & \text{if } \ t_p^{\beta} \in \mathcal{C} \ \text{ for } \ p \in \mathcal{P}_{\alpha\beta} \\ f_p & \text{for } \ p \in \mathcal{P}, \ p \notin \mathcal{P}_{\alpha\beta}. \end{cases}$$

- There is a one-to-one correspondences between a cut and a labeling.
- The energy of the cut is the energy of the labeling.
- See Boykov et al, "*fast approximate energy minimization via graph cuts*" PAMI 2001.

- For any cut, then

$$\begin{array}{lllll}
(a) & \textit{If} & t_p^\alpha, t_q^\alpha \in \mathcal{C} & \textit{then} & e_{\{p,q\}} \notin \mathcal{C}. \\
(b) & \textit{If} & t_p^\beta, t_q^\beta \in \mathcal{C} & \textit{then} & e_{\{p,q\}} \notin \mathcal{C}. \\
(c) & \textit{If} & t_p^\beta, t_q^\alpha \in \mathcal{C} & \textit{then} & e_{\{p,q\}} \in \mathcal{C}. \\
(d) & \textit{If} & t_p^\alpha, t_q^\beta \in \mathcal{C} & \textit{then} & e_{\{p,q\}} \in \mathcal{C}.
\end{array}$$



Finding the optimal α expansion

- Given an input labeling f (partition $\mathcal{P}$) and a label α we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one α-expansion of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$.

Finding the optimal α expansion

- Given an input labeling f (partition $\mathcal{P}$) and a label α we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one α-expansion of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$.
- The structure of this graph is dynamically determined by the current partition $\mathcal{P}$ and by the label α.

Finding the optimal α expansion

- Given an input labeling f (partition $\mathcal{P}$) and a label α we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one α-expansion of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$.
- The structure of this graph is dynamically determined by the current partition $\mathcal{P}$ and by the label α.
- Different graph than the $\alpha - \beta$ swap.

Finding the optimal α expansion

- Given an input labeling f (partition $\mathcal{P}$) and a label α we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one α-expansion of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$.
- The structure of this graph is dynamically determined by the current partition $\mathcal{P}$ and by the label α.
- Different graph than the $\alpha - \beta$ swap.

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.
- Each pixel p is connected to the terminals α and $\bar{\alpha}$, called t-links.

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.
- Each pixel p is connected to the terminals α and $\bar{\alpha}$, called t-links.
- Each set of pixels p, q which are neighbors and $f_p = f_q$, we connect with and n-link.

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.
- Each pixel p is connected to the terminals α and $\bar{\alpha}$, called t-links.
- Each set of pixels p, q which are neighbors and $f_p = f_q$, we connect with and n-link.
- For each pair of neighboring pixels such that $f_p \neq f_q$, we create a triplet $\{e_{p,a}, e_{a,q}, t_a^{\bar{\alpha}}\}$.

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.
- Each pixel p is connected to the terminals α and $\bar{\alpha}$, called t-links.
- Each set of pixels p, q which are neighbors and $f_p = f_q$, we connect with and n-link.
- For each pair of neighboring pixels such that $f_p \neq f_q$, we create a triplet $\{e_{p,a}, e_{a,q}, t_a^{\bar{\alpha}}\}$.
- The set of edges is then

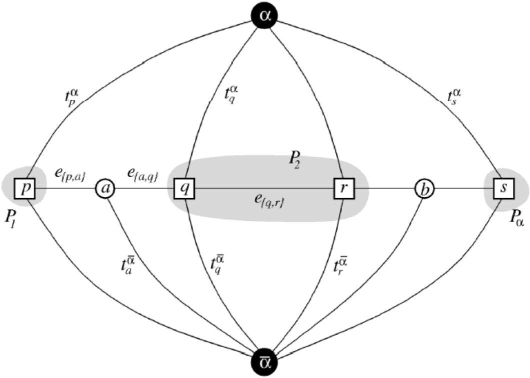
$$\mathcal{E}_\alpha = \left\{ \bigcup_{p \in \mathcal{P}} \{t_p^\alpha, t_p^{\bar{\alpha}}\}, \bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p \neq f_q}} \mathcal{E}_{\{p,q\}} \;, \bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p = f_q}} e_{\{p,q\}} \right\}$$

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.
- Each pixel p is connected to the terminals α and $\bar{\alpha}$, called t-links.
- Each set of pixels p, q which are neighbors and $f_p = f_q$, we connect with and n-link.
- For each pair of neighboring pixels such that $f_p \neq f_q$, we create a triplet $\{e_{p,a}, e_{a,q}, t_a^{\bar{\alpha}}\}$.
- The set of edges is then

$$\mathcal{E}_\alpha = \left\{ \bigcup_{p \in \mathcal{P}} \{t_p^\alpha, t_p^{\bar{\alpha}}\}, \bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p \neq f_q}} \mathcal{E}_{\{p,q\}} \ , \bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p = f_q}} e_{\{p,q\}} \right\}$$

Graph Construction



edge	weight	for
$t_p^{\bar\alpha}$	∞	$p \in \mathcal{P}_\alpha$
$t_p^{\bar\alpha}$	$D_p(f_p)$	$p \notin \mathcal{P}_\alpha$
t_p^{α}	$D_p(\alpha)$	$p \in \mathcal{P}$
$e_{\{p,a\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N},\ f_p \neq f_q$
$e_{\{a,q\}}$	$V(\alpha, f_q)$	
$t_a^{\bar\alpha}$	$V(f_p, f_q)$	
$e_{\{p,q\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N},\ f_p = f_q$

Properties

- There is a one-to-one correspondences between a cut and a labeling.

$$f_p^{\mathcal{C}} = \begin{cases} \alpha & \text{if} \quad t_p^{\alpha} \in \mathcal{C} \\ f_p & \text{if} \quad t_p^{\bar{\alpha}} \in \mathcal{C} \end{cases} \qquad \forall p \in \mathcal{P}.$$

- The energy of the cut is the energy of the labeling.
- See Boykov et al, "*fast approximate energy minimization via graph cuts*" PAMI 2001.

Property 5.2. *If* $\{p,q\} \in \mathcal{N}$ *and* $f_p \neq f_q$, *then a minimum cut* $\mathcal{C}$ *on* $\mathcal{G}_\alpha$ *satisfies:*

$$\begin{array}{llllll} (a) & \mathit{If} & t_p^{\alpha}, t_q^{\alpha} \in \mathcal{C} & \mathit{then} & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = \emptyset. \\ (b) & \mathit{If} & t_p^{\bar{\alpha}}, t_q^{\bar{\alpha}} \in \mathcal{C} & \mathit{then} & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = t_a^{\bar{\alpha}}. \\ (c) & \mathit{If} & t_p^{\bar{\alpha}}, t_q^{\alpha} \in \mathcal{C} & \mathit{then} & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{p,a\}}. \\ (d) & \mathit{If} & t_p^{\alpha}, t_q^{\bar{\alpha}} \in \mathcal{C} & \mathit{then} & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{a,q\}}. \end{array}$$

Learning in graphical models

Parameter learning

- The MAP problem was defined as

$$\max_{y_1,\cdots,y_n} \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- Learn parameters **w** for more accurate prediction

$$\max_{y_1,\cdots,y_n} \sum_i \mathbf{w}_i \phi_i(y_i) + \sum_\alpha \mathbf{w}_\alpha \phi_\alpha(y_\alpha)$$

Loss functions

- Regularized loss minimization: Given input pairs $(x, y) \in \mathcal{S}$, minimize

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

- Different learning frameworks depending on the surrogate loss $\hat{\ell}(\mathbf{w}, x, y)$
  - Hinge for Structural SVMs [Tsochantaridis et al. 05, Taskar et al. 04]
  - log-loss for Conditional Random Fields [Lafferty et al. 01]
- Unified by [Hazan and Urtasun, 10]

Loss functions

- Regularized loss minimization: Given input pairs $(x, y) \in \mathcal{S}$, minimize

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

- Different learning frameworks depending on the surrogate loss $\hat{\ell}(\mathbf{w}, x, y)$
  - Hinge for Structural SVMs [Tsochantaridis et al. 05, Taskar et al. 04]
  - log-loss for Conditional Random Fields [Lafferty et al. 01]
- Unified by [Hazan and Urtasun, 10]

Recall SVM

- In SVMs we minimize the following program

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \xi_i$$

$$\text{subject to} \;\; y_i(b + \mathbf{w}^T\mathbf{x}_i) - 1 + \xi_i \;\geq\; 0, \quad \forall i = 1, \ldots, N.$$

  with $y_i \in \{-1, 1\}$ binary.

- We need to extend this to reason about more complex structures, not just binary variables.

Structural SVM [Tsochantaridis et al., 05]

- We want to construct a function

$$f(x, y) = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^T \phi(x, y)$$

which is parameterized in terms of **w**, the parameters to learn.

- We will like to minimize the empirical risk

$$R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w))$$

Structural SVM [Tsochantaridis et al., 05]

- We want to construct a function

$$f(x, y) = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^T \phi(x, y)$$

which is parameterized in terms of $\mathbf{w}$, the parameters to learn.

- We will like to minimize the empirical risk

$$R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w))$$

- This is the expected loss under the empirical distribution induced

Structural SVM [Tsochantaridis et al., 05]

- We want to construct a function

$$f(x, y) = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^T \phi(x, y)$$

  which is parameterized in terms of **w**, the parameters to learn.

- We will like to minimize the empirical risk

$$R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w))$$

- This is the expected loss under the empirical distribution induced
- $\Delta(y_i, f(x_i, w))$ is the "task loss" which depends on the application
  - segmentation: per pixel segmentation error
  - detection: intersection over the union

Structural SVM [Tsochantaridis et al., 05]

- We want to construct a function

$$f(x,y) = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^T \phi(x,y)$$

  which is parameterized in terms of **w**, the parameters to learn.

- We will like to minimize the empirical risk

$$R_s(f,w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i,w))$$

- This is the expected loss under the empirical distribution induced
- $\Delta(y_i, f(x_i,w))$ is the "task loss" which depends on the application
  - segmentation: per pixel segmentation error
  - detection: intersection over the union
- Typically, $\Delta(y,y') = 0$ if $y = y'$

Structural SVM [Tsochantaridis et al., 05]

- We want to construct a function

$$f(x,y) = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^T \phi(x,y)$$

  which is parameterized in terms of **w**, the parameters to learn.

- We will like to minimize the empirical risk

$$R_s(f,w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w))$$

- This is the expected loss under the empirical distribution induced
- $\Delta(y_i, f(x_i, w))$ is the "task loss" which depends on the application
  - segmentation: per pixel segmentation error
  - detection: intersection over the union
- Typically, $\Delta(y, y') = 0$ if $y = y'$

Separable case

- We will like to minimize the empirical risk

$$R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w))$$

- We will have 0 train error if we satisfy

$$\max_{y \in \mathcal{Y} \setminus y_i} \{\mathbf{w}^T \phi(x_i, y)\} \leq \mathbf{w}^T \phi(x_i, y_i)$$

since $\Delta(y_i, y_i) = 0$ and $\Delta(y_i, y) > 0, \forall y \in \mathcal{Y} \setminus y_i$.

Separable case

- We will like to minimize the empirical risk

$$R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w))$$

- We will have 0 train error if we satisfy

$$\max_{y \in \mathcal{Y} \setminus y_i} \{\mathbf{w}^T \phi(x_i, y)\} \leq \mathbf{w}^T \phi(x_i, y_i)$$

since $\Delta(y_i, y_i) = 0$ and $\Delta(y_i, y) > 0, \forall y \in \mathcal{Y} \setminus y_i$.

- This can be replaced by $|\mathcal{Y}| - 1$ inequalities

$$\forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i : \quad \mathbf{w}^T \phi(x_i, y_i) - \mathbf{w}^T \phi(x_i, y) \geq 0$$

Separable case

- We will like to minimize the empirical risk

$$R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w))$$

- We will have 0 train error if we satisfy

$$\max_{y \in \mathcal{Y} \setminus y_i} \{\mathbf{w}^T \phi(x_i, y)\} \leq \mathbf{w}^T \phi(x_i, y_i)$$

since $\Delta(y_i, y_i) = 0$ and $\Delta(y_i, y) > 0, \forall y \in \mathcal{Y} \setminus y_i$.

- This can be replaced by $|\mathcal{Y}| - 1$ inequalities

$$\forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i : \quad \mathbf{w}^T \phi(x_i, y_i) - \mathbf{w}^T \phi(x_i, y) \geq 0$$

- What's the problem of this?

Separable case

- We will like to minimize the empirical risk

$$R_s(f, w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f(x_i, w))$$

- We will have 0 train error if we satisfy

$$\max_{y \in \mathcal{Y} \setminus y_i} \{\mathbf{w}^T \phi(x_i, y)\} \leq \mathbf{w}^T \phi(x_i, y_i)$$

since $\Delta(y_i, y_i) = 0$ and $\Delta(y_i, y) > 0, \forall y \in \mathcal{Y} \setminus y_i$.

- This can be replaced by $|\mathcal{Y}| - 1$ inequalities

$$\forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i : \quad \mathbf{w}^T \phi(x_i, y_i) - \mathbf{w}^T \phi(x_i, y) \geq 0$$

- What's the problem of this?

Separable case

- Satisfying the inequalities might have more than one solution.
- Select the $\mathbf{w}$ with the maximum margin.

Separable case

- Satisfying the inequalities might have more than one solution.
- Select the **w** with the maximum margin.
- We can thus form the following optimization problem

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

Separable case

- Satisfying the inequalities might have more than one solution.
- Select the $\mathbf{w}$ with the maximum margin.
- We can thus form the following optimization problem

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

- This is a quadratic program, so it's convex

Separable case

- Satisfying the inequalities might have more than one solution.
- Select the **w** with the maximum margin.
- We can thus form the following optimization problem

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to} \;\; \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

- This is a quadratic program, so it's convex
- But it involves exponentially many constraints!

Separable case

- Satisfying the inequalities might have more than one solution.
- Select the **w** with the maximum margin.
- We can thus form the following optimization problem

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

- This is a quadratic program, so it's convex
- But it involves exponentially many constraints!

Non-separable case

Multiple formulations

- Multi-class classification [Crammer & Singer, 03]
- Slack re-scaling [Tsochantaridis et al. 05]
- Margin re-scaling [Taskar et al. 04]

Let's look at them in more details

Multi-class classification [Crammer & Singer, 03]

- Enforce a large margin and do a batch convex optimization
- The minimization program is then

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 - \xi_i \quad \forall i \in \{1, \cdots, n\}, \forall y \neq y_i$$

- Can also be written in terms of kernels

Structured Output SVMs

- Frame structured prediction as a multiclass problem to predict a single element of Y and pay a penalty for mistakes
- Not all errors are created equally, e.g. in an HMM making only one mistake in a sequence should be penalized less than making 50 mistakes

Structured Output SVMs

- Frame structured prediction as a multiclass problem to predict a single element of Y and pay a penalty for mistakes
- Not all errors are created equally, e.g. in an HMM making only one mistake in a sequence should be penalized less than making 50 mistakes
- Pay a loss proportional to the difference between true and predicted error (task dependent)

$$\Delta(y_i, y)$$

[Source: M. Blaschko]

Structured Output SVMs

- Frame structured prediction as a multiclass problem to predict a single element of Y and pay a penalty for mistakes
- Not all errors are created equally, e.g. in an HMM making only one mistake in a sequence should be penalized less than making 50 mistakes
- Pay a loss proportional to the difference between true and predicted error (task dependent)

$$\Delta(y_i, y)$$

[Source: M. Blaschko]

Example: Data imbalanced

- Suppose that we have highly imbalanced training data: $n_+ \gg n_-$
- We still have a two class problem
- We can use structured output formulation to pay a higher price for misclassification of positives than misclassification of negative, e.g.,

$$\Delta(y_i, y) = \begin{cases} 0 & \text{if } y_i == y \\ \frac{1}{n_+} & \text{if } y_i = 1 \wedge y = -1 \\ \frac{1}{n_-} & \text{if } y_i = -1 \wedge y = 1 \end{cases}$$

[Source: M. Blaschko]

Slack re-scaling

- Re-scale the slack variables according to the loss incurred in each of the linear constraints
- Violating a margin constraint involving a $y \neq y_i$ with high loss $\Delta(y_i, y)$ should be penalized more than a violation involving an output value with smaller loss

Slack re-scaling

- Re-scale the slack variables according to the loss incurred in each of the linear constraints
- Violating a margin constraint involving a $y \neq y_i$ with high loss $\Delta(y_i, y)$ should be penalized more than a violation involving an output value with smaller loss
- The minimization program is then

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 - \frac{\xi_i}{\Delta(y_i, y)} \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

Slack re-scaling

- Re-scale the slack variables according to the loss incurred in each of the linear constraints
- Violating a margin constraint involving a $y \neq y_i$ with high loss $\Delta(y_i, y)$ should be penalized more than a violation involving an output value with smaller loss
- The minimization program is then

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 - \frac{\xi_i}{\Delta(y_i, y)} \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

- The justification is that $\frac{1}{n}\sum_{i=1}^{n} \xi_i$ is an upper-bound on the empirical risk.
- Easy to proof

Slack re-scaling

- Re-scale the slack variables according to the loss incurred in each of the linear constraints
- Violating a margin constraint involving a $y \neq y_i$ with high loss $\Delta(y_i, y)$ should be penalized more than a violation involving an output value with smaller loss
- The minimization program is then

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 - \frac{\xi_i}{\Delta(y_i, y)} \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

- The justification is that $\frac{1}{n}\sum_{i=1}^{n}\xi_i$ is an upper-bound on the empirical risk.
- Easy to proof

Margin re-scaling

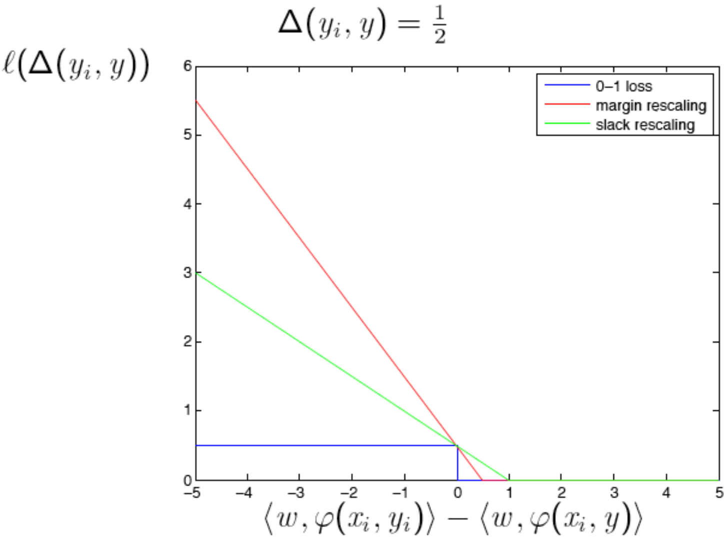
- In this case the minimization problem is formulated as

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq \Delta(y_i, y) - \xi_i \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

- The justification is that $\frac{1}{n}\sum_{i=1}^{n}\xi_i$ is an upper-bound on the empirical risk.
- Also easy to proof.

Margin vs Slack re-scaling



Algorithm

- Problem is the exponential number of constraints
- Derive a cutting plane algorithm, where the most violated constraints are added as we go

Algorithm 1 Algorithm for solving SVM_0 and the loss re-scaling formulations $\text{SVM}_1^{\triangle s}$ and $\text{SVM}_2^{\triangle s}$.

1: Input: $(\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_n,\mathbf{y}_n), C, \varepsilon$

2: $S_i \leftarrow \emptyset$ for all $i = 1,\ldots,n$

3: **repeat**

4: **for** $i = 1,\ldots,n$ **do**

5: /* *prepare cost function for optimization* */

set up cost function

$$H(\mathbf{y}) \equiv \begin{cases} 1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w}\rangle & (\text{SVM}_0) \\ (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w}\rangle)\,\triangle(\mathbf{y}_i,\mathbf{y}) & (\text{SVM}_1^{\triangle s}) \\ \triangle(\mathbf{y}_i,\mathbf{y}) - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w}\rangle & (\text{SVM}_1^{\triangle m}) \\ (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w}\rangle)\sqrt{\triangle(\mathbf{y}_i,\mathbf{y})} & (\text{SVM}_2^{\triangle s}) \\ \sqrt{\triangle(\mathbf{y}_i,\mathbf{y})} - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w}\rangle & (\text{SVM}_2^{\triangle m}) \end{cases}$$

where $\mathbf{w} \equiv \sum_j \sum_{\mathbf{y}' \in S_j} \alpha_{(j\mathbf{y}')} \delta\Psi_j(\mathbf{y}')$.

6: /* *find cutting plane* */

compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y})$

7: /* *determine value of current slack variable* */

compute $\xi_i = \max\{0, \max_{\mathbf{y}\in S_i} H(\mathbf{y})\}$

8: **if** $H(\hat{\mathbf{y}}) > \xi_i + \varepsilon$ **then**

9: /* *add constraint to the working set* */

$S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$

10a: /* *Variant (a): perform full optimization* */

$\alpha_S \leftarrow$ optimize the dual of SVM_0 , SVM_1^{*} or SVM_2^{*} over S, $S = \cup_i S_i$.

10b: /* *Variant (b): perform subspace ascent* */

$\alpha_{S_i} \leftarrow$ optimize the dual of SVM_0 , SVM_1^{*} or SVM_2^{*} over S_i

12: **end if**

13: **end for**

14: **until** no S_i has changed during iteration

Constraint Generation

- To find the most violated constraint, we need to maximize w.r.t. y for margin rescaling

$$\mathbf{w}^T \phi(x_i, y) + \Delta(y_i, y)$$

  and for slack rescaling

$$\{\mathbf{w}^T \phi(x_i, y) + 1 - \mathbf{w}^T \phi(x_i, y_i)\} \Delta(y_i, y)$$

- For arbitrary output spaces, we would need to iterate over all elements in $\mathcal{Y}$

Constraint Generation

- To find the most violated constraint, we need to maximize w.r.t. y for margin rescaling
$$\mathbf{w}^T\phi(x_i, y) + \Delta(y_i, y)$$
and for slack rescaling
$$\{\mathbf{w}^T\phi(x_i, y) + 1 - \mathbf{w}^T\phi(x_i, y_i)\}\Delta(y_i, y)$$

- For arbitrary output spaces, we would need to iterate over all elements in $\mathcal{Y}$
- Use Graph-cuts or message passing

Constraint Generation

- To find the most violated constraint, we need to maximize w.r.t. y for margin rescaling

$$\mathbf{w}^T \phi(x_i, y) + \Delta(y_i, y)$$

and for slack rescaling

$$\{\mathbf{w}^T \phi(x_i, y) + 1 - \mathbf{w}^T \phi(x_i, y_i)\} \Delta(y_i, y)$$

- For arbitrary output spaces, we would need to iterate over all elements in $\mathcal{Y}$
- Use Graph-cuts or message passing
- When the MAP cannot be computed exactly, but only approximately, this algorithm does not behave well [Fidley et al., 08]

Constraint Generation

- To find the most violated constraint, we need to maximize w.r.t. y for margin rescaling

$$\mathbf{w}^T \phi(x_i, y) + \Delta(y_i, y)$$

and for slack rescaling

$$\{\mathbf{w}^T \phi(x_i, y) + 1 - \mathbf{w}^T \phi(x_i, y_i)\} \Delta(y_i, y)$$

- For arbitrary output spaces, we would need to iterate over all elements in $\mathcal{Y}$
- Use Graph-cuts or message passing
- When the MAP cannot be computed exactly, but only approximately, this algorithm does not behave well [Fidley et al., 08]

One Slack Formulation

- Margin rescaling

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\xi$$
$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq \Delta(y_i, y) - \xi \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

- Slack rescaling

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\xi$$
$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 - \frac{\xi}{\Delta(y_i, y)} \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

One Slack Formulation

- Margin rescaling

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\xi$$
$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq \Delta(y_i, y) - \xi \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

- Slack rescaling

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\xi$$
$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 - \frac{\xi}{\Delta(y_i, y)} \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

- Same optima as previous formulation [Joachims et al, 09]

One Slack Formulation

- Margin rescaling

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\xi$$
$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq \Delta(y_i, y) - \xi \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

- Slack rescaling

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\xi$$
$$\text{s.t.} \quad \mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq 1 - \frac{\xi}{\Delta(y_i, y)} \quad \forall i \in \{1, \cdots, n\}, \forall y \in \mathcal{Y} \setminus y_i$$

- Same optima as previous formulation [Joachims et al, 09]

Example: Handwritten Recognition

- Predict text from image of handwritten characters

$\arg\max_{\mathbf{y}} \; \mathbf{w}^\top \mathbf{f}($  $, \mathbf{y}) \; = \;$ "brace"

- Equivalently:

$\mathbf{w}^\top \mathbf{f}($  $,$ "brace"$) \; > \; \mathbf{w}^\top \mathbf{f}($  $,$ "aaaaa"$)$

$\mathbf{w}^\top \mathbf{f}($  $,$ "brace"$) \; > \; \mathbf{w}^\top \mathbf{f}($  $,$ "aaaab"$)$

...

$\mathbf{w}^\top \mathbf{f}($  $,$ "brace"$) \; > \; \mathbf{w}^\top \mathbf{f}($  $,$ "zzzzz"$)$

- Iterate
  - Estimate model parameters **w** using active constraint set
  - Generate the next constraint

[Source: B. Taskar]