Probabilistic Graphical Models in Computer Vision (IN2329)

Csaba Domokos

Summer Semester 2015/2016

7. FastPD & Branch-and-MinCut

FastPD

Recall: Primal-dual LP for multi-label problem *

The (relaxed) primal LP:

$$\min_{x_{i:\alpha}, x_{ij:\alpha\beta} \geqslant 0} \sum_{i \in \mathcal{V}} \sum_{\alpha \in \mathcal{L}} E_i(\alpha) x_{i:\alpha} + \sum_{(i,j) \in \mathcal{E}} w_{ij} \sum_{\alpha,\beta \in \mathcal{L}} d(\alpha,\beta) x_{ij:\alpha\beta}$$

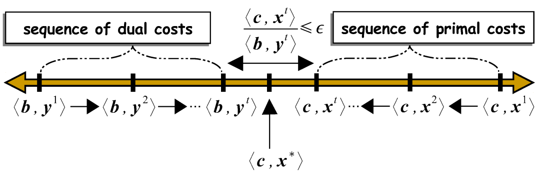
subject to
$$\sum_{\alpha \in \mathcal{L}} x_{i:\alpha} = 1 \quad \forall i \in \mathcal{V}$$
$$\sum_{\alpha \in \mathcal{L}} x_{ij:\alpha\beta} = x_{j:\beta} \quad \forall \beta \in \mathcal{L}, (i,j) \in \mathcal{E}$$
$$\sum_{\beta \in \mathcal{L}} x_{ij:\alpha\beta} = x_{i:\alpha} \quad \forall \alpha \in \mathcal{L}, (i,j) \in \mathcal{E}$$

The dual LP:

$$\max_{y_i, y_{ij:\alpha}, y_{ji:\beta}} \sum_{i \in \mathcal{V}} y_i$$

subject to
$$y_i - \sum_{j \in \mathcal{V}:(i,j) \in \mathcal{E}} y_{ij:\alpha} \leqslant E_i(\alpha) \quad \forall i \in \mathcal{V}, \alpha \in \mathcal{L}$$
$$y_{ij:\alpha} + y_{ji:\beta} \leqslant w_{ij} d(\alpha,\beta) \quad \forall (i,j) \in \mathcal{E}, \alpha, \beta \in \mathcal{L}$$

Recall: Primal-dual schema *



Typically, primal-dual ϵ-approximation algorithms construct a sequence $(\mathbf{x}^k, \mathbf{y}^k)_{k=1,\dots,t}$ of primal and dual solutions until the elements $\mathbf{x}^t$, $\mathbf{y}^t$ of the last pair are both **feasible** and **satisfy the relaxed primal complementary slackness conditions**, hence the condition $\langle \mathbf{c}, \mathbf{x} \rangle \leqslant \epsilon \langle \mathbf{b}, \mathbf{y} \rangle$ will be also fulfilled.

Recall: Complementary slackness conditions *

From now on, in case of Algorithm PD1, we only assume that $d(\alpha,\beta) = 0 \Leftrightarrow \alpha = \beta$, and $d(\alpha,\beta) \geqslant 0$ (i.e. semi-metric).

The *complementary slackness conditions* reduces to

$$y_i - \sum_{j \in \mathcal{V}:(i,j) \in \mathcal{E}} y_{ij:x_i} \geqslant \frac{E_i(x_i)}{\epsilon_1} \quad \Rightarrow \quad y_i \geqslant \frac{E_i(x_i)}{\epsilon_1} + \sum_{j \in \mathcal{V}:(i,j) \in \mathcal{E}} y_{ij:x_i}$$

$$y_{ij:x_i} + y_{ji:x_j} \geqslant \frac{w_{ij} d(x_i, x_j)}{\epsilon_2}$$

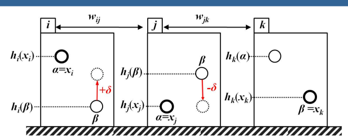
for specific values of $\epsilon_1, \epsilon_2 \geqslant 1$.

If $x_i = x_j = \alpha$ for neighboring pairs $(i,j) \in \mathcal{E}$, then

$$0 = w_{ij} d(\alpha,\alpha) \geqslant y_{ij:\alpha} + y_{ji:\alpha} \geqslant \frac{w_{ij} d(\alpha,\alpha)}{\epsilon_2} = 0 ,$$

therefore we get that $y_{ij:\alpha} = -y_{ji:\alpha}$.

Recall: Update primal and dual variables *



Dual variables update: Given the current active labels, any non-active label is raised, until it either reaches the active label, or attains the maximum raise allowed by the upper bound.

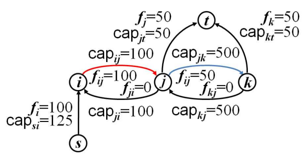
Primal variables update: Given the new heights, there might still be vertices whose active labels are not at the lowest height. For each such vertex i, we select a non-active label, which is below x_i, but has already reached the maximum raise allowed by the upper bound.

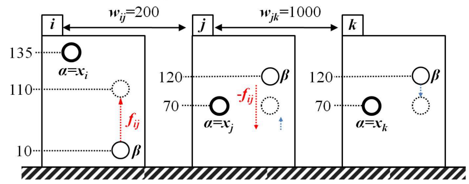
The optimal update of the α-heights can be simulated by pushing the **maximum amount of flow** through a directed graph $G' = (\mathcal{V} \cup \{s,t\}, \mathcal{E}', c, s, t)$.

Recall: Reassign rule *

Label α will be the new label of i (i.e. $x_i' = \alpha$) iff there exists *unsaturated path* (i.e. $f_{ij} < \text{cap}_{ij}$) between the source node s and node i. In all other cases, i keeps its current label (i.e. $x_i' = x_i$).

$$f_{ij} < \text{cap}_{ij}$$
$$h_i'(\alpha) - h_i(\alpha) < h_i(x_i) - h_i(\alpha)$$
$$h_i'(\alpha) < h_i(x_i) = h_i'(x_i)$$





The APF function *

$APF^{\mathbf{x}',\mathbf{y}'} \leqslant APF^{\mathbf{x},\mathbf{y}}$, where $APF^{\mathbf{x},\mathbf{y}}$ is defined as

$$\begin{aligned} APF^{\mathbf{x},\mathbf{y}} &\triangleq \sum_{i\in\mathcal{V}} h_i(x_i) = \sum_{i\in\mathcal{V}} \Big(E_i(x_i) + \sum_{j\in\mathcal{V},(i,j)\in\mathcal{E}} \text{load}_{ij} \Big) \\ &= \sum_{i\in\mathcal{V}} E_i(x_i) + \sum_{(i,j)\in\mathcal{E}} (y_{ij:x_i} + y_{ji:x_j}) \\ &\leqslant \sum_{i\in\mathcal{V}} E_i(x_i) + \sum_{(i,j)\in\mathcal{E}} w_{ij} d(x_i,x_j) = E(\mathbf{x}) \ . \end{aligned}$$

This condition shows that the algorithm terminates (assuming integer capacities), due to the reassign rule, which ensures that a new active label has always lower height than the previous active label, i.e. $h_i'(x_i') \leqslant h_i(x_i)$.

PD2

Parameterization of the PD2 algorithm

We now assume that d is a *metric*.

In fact, PD2 represents a family of algorithms parameterized by $\mu \in [\frac{1}{\epsilon_{\text{app}}}, 1]$. Algorithm PD2_μ will achieve *complementary slackness conditions* with

$$\epsilon_1 \triangleq \mu\epsilon_{\text{app}} \geqslant \frac{1}{\epsilon_{\text{app}}}\epsilon_{\text{app}} \geqslant 1 \quad \text{and} \quad \epsilon_2 = \epsilon_{\text{app}} \ .$$

Algorithm PD1 always generates a feasible dual solution at any of its inner iterations, whereas PD2_μ **may allow any such dual solution to become infeasible**.

Dual-fitting: PD2_μ ensures that the (probably infeasible) final dual solution is "not too far away from feasibility", which practically means that if that solution is divided by a suitable factor, it will become feasible again.

Complementary slackness conditions *

Similarly to Algorithm PD1, the equalities will hold for $i \in \mathcal{V}$

$$y_i = \min_{\alpha\in\mathcal{L}} h_i(\alpha) = h_i(x_i) = E_i(x_i) + \sum_{i\in\mathcal{V},(i,j)\in\mathcal{E}} y_{ij:x_i} \ .$$

PD2_μ generates a series of intermediate pairs satisfying *complementary slackness conditions* for $\epsilon_1 \geqslant 1$ and $\epsilon_2 \geqslant \frac{1}{\mu} = \frac{1}{1/\epsilon_{\text{app}}} = \epsilon_{\text{app}}$:

$$\frac{E_i(x_i)}{\epsilon_1} + \sum_{i\in\mathcal{V},(i,j)\in\mathcal{E}} y_{ij:x_i} \leqslant E_i(x_i) + \sum_{i\in\mathcal{V},(i,j)\in\mathcal{E}} y_{ij:x_i} = h_i(x_i) = y_i \qquad \forall i \in \mathcal{V} \ .$$

$$\frac{w_{ij}d(x_i,x_j)}{\epsilon_2} \leqslant \mu w_{ij} d(x_i,x_j) = \text{load}_{ij} \qquad \forall (i,j) \in \mathcal{E} \ .$$

Like PD1, PD2_μ also maintains non-negativity of *active balance variables*.

Dual fitting

The dual solution of the last intermediate pair may be infeasible, since, instead of the feasibility condition $y_{ij:\alpha} + y_{ji:\beta} \leqslant w_{ij}d(\alpha,\beta)$, PD2_μ maintains the conditions:

$$y_{ij:\alpha} + y_{ji:\beta} \leqslant 2\mu w_{ij} d_{\max} \qquad \forall (i,j) \in \mathcal{E},\ \forall \alpha,\beta \in \mathcal{L} \ .$$

These conditions also ensure that the last dual solution $\mathbf{y}$, is not "too far away from feasibility".By replacing $\mathbf{y}$ with $\mathbf{y}^{\text{fit}} = \frac{\mathbf{y}}{\mu\epsilon_{\text{app}}}$ we get that

$$y_{ij:\alpha}^{\text{fit}} + y_{ji:\beta}^{\text{fit}} = \frac{y_{ij:\alpha} + y_{ji:\beta}}{\mu\epsilon_{\text{app}}} \leqslant \frac{2\mu w_{ij} d_{\max}}{\mu\epsilon_{\text{app}}} = \frac{2\mu w_{ij} d_{\max}}{\mu 2 d_{\max}/d_{\min}} = w_{ij} d_{\min} \leqslant w_{ij} d(\alpha,\beta).$$

This means that $\mathbf{y}^{\text{fit}}$ is **feasible**.

```
1: function DUAL_FIT(y)
2:     return y^fit ← y / (μ ε_app)
3: end function
```


Update primal and dual variables

The main/only difference in the subroutine Update_Duals_Primals(α,$\mathbf{x}$,$\mathbf{y}$) is the definition of the capacities corresponding to the *n-edges*. More precisely, assuming an α-iteration, where $x_i = \beta \neq \alpha$ and $x_j = \gamma \neq \alpha$ for a given $(i,j) \in \mathcal{E}$:

$$\text{cap}_{ij} = \mu w_{ij}(d(\beta,\alpha) + d(\alpha,\gamma) - d(\beta,\gamma)) \ , \qquad (1)$$
$$\text{cap}_{ji} = 0 \ .$$

All the capacities in the flow must be non-negative. This motivates that d must be a metric.

By applying $\text{load}_{ij} = y_{ij:\beta} + y_{ji:\gamma} = \mu w_{ij} d(\beta,\gamma)$ one can get

$$\begin{aligned} y'_{ij:\alpha} &= y_{ij:\alpha} + \text{cap}_{ij} = y_{ij:\alpha} + \mu w_{ij}(d(\beta,\alpha) + d(\alpha,\gamma) - d(\beta,\gamma)) \\ &= y_{ij:\alpha} + y_{ij:\beta} + y_{ji:\alpha} + \mu w_{ij} d(\alpha,\gamma) - y_{ij:\beta} - y_{ji:\gamma} = \mu w_{ij} d(\alpha,\gamma) - y_{ji:\gamma} \ , \end{aligned}$$

which ensures that

$$\text{load}_{ij}^{\mathbf{x},\mathbf{y}} = y_{ij:\alpha} + y_{ji:\gamma} = \big(\mu w_{ij} d(\alpha,\gamma) - y_{ji:\gamma}\big) + y_{ji:\gamma} = \mu w_{ij} d(\alpha,\gamma) \ .$$

Subroutine PreEdit_Duals(α,$\mathbf{x}$,$\mathbf{y}$) *

The role of this routine is to edit current solution $\mathbf{y}$, before the subroutine Update_Duals_Primals(α,$\mathbf{x}$), so that

$$\text{load}_{ij}^{\mathbf{x},\mathbf{y}} = y_{ij:\alpha} + y_{ji:\gamma} = \mu w_{ij} d(\alpha,\gamma) \ .$$

```
1: function PREEDIT_DUALS(α,x,y)
2:     for all (i,j) ∈ E with x_i ≠ α, x_j ≠ α do
3:         y_ij:α ← μ w_ij d(α,γ) − y_ji:γ
4:         y_ji:α ← y_ji:γ − μ w_ij d(α,γ)
5:     end for
6:     return y
7: end function
```


Equivalence of $\text{PD2}_{\mu=1}$ and α-expansion

One can show that $\text{PD2}_{\mu=1}$ indeed generates an ϵ_{app} solution.

If $\mu = 1$, then $\text{load}_{ij} = w_{ij} d(x_i,x_j)$. It can be shown that $APF^{\mathbf{x},\mathbf{y}} = E(\mathbf{x})$, whereas in any other case $APF^{\mathbf{x},\mathbf{y}} \leqslant E(\mathbf{x})$.
If $\mu < 1$, then the primal (dual) objective function necessarily decreases (increases) per iteration. Instead, APF constantly decreases.

Recall that APF is the sum of active labels' heights and $\text{PD2}_{\mu=1}$ always tries to choose the *lowest* label among x_i and α. During an α-iteration, $\text{PD2}_{\mu=1}$ chooses an $\mathbf{x}'$ that minimizes APF with respect to any other α-expansion $\bar{\mathbf{x}}$ of current solution $\mathbf{x}$.

Theorem 1. *Let $(\mathbf{x}',\mathbf{y}')$ denote the next primal-dual pair due to an α-iteration and let $\bar{\mathbf{x}}$ denote α-expansion of the current primal. Then*

$$E(\mathbf{x}') = APF^{\mathbf{x}',\mathbf{y}'} \leqslant APF^{\bar{\mathbf{x}},\mathbf{y}'} \leqslant E(\bar{\mathbf{x}}) \ .$$

$E(\mathbf{x}') \leqslant E(\bar{\mathbf{x}})$ shows that the α-expansion algorithm is equivalent to $\text{PD2}_{\mu=1}$.

PD3

Algorithm $\texttt{PD3}_a$ *

By modifying the Algorithm $\texttt{PD2}_{\mu=1}$, we will get Algorithm PD3, which can be applied even if d is non-metric function.

Recall that $\texttt{PD2}_{\mu=1}$ maintains the *optimality criterion*: $\text{load}_{ij} \leqslant w_{ij} d(x_i, x_j)$.

Since d is not metric, we have **conflicting label-triplet** (α, β, γ):

$$d(\beta, \gamma) > d(\beta, \alpha) + d(\alpha, \gamma) \; .$$

Algorithm $\texttt{PD3}_a$: During the primal-dual variable update, in an α-iteration, when $x_i \neq \alpha$ and $x_j \neq \alpha$, i.e. in (1), we set $\text{cap}_{ij} = 0$.

It can be shown that for a *conflicting triplet*

$$\text{load}_{ij} = w_{ij}\big(d(\beta, \gamma) - d(\beta, \alpha)\big) \geqslant w_{ij} d(\alpha, \gamma) \; .$$

Intuitively, $\texttt{PD3}_a$ **overestimates the distance** between labels α, γ in order to restore the triangle inequality for the current conflicting label-triplet (α, β, γ).

$\texttt{PD3}_b$ *

We choose to set $\text{cap}_{ij} = +\infty$ and no further differences between $\texttt{PD3}_b$ and $\texttt{PD2}_{\mu=1}$ exist.

This has the following important effect: the solution $\mathbf{x}'$ produced at the current iteration, can never assign the pair of labels γ, β to the objects i, j respectively.

$\texttt{PD3}_c$ *

$\texttt{PD3}_c$ first adjusts the dual solution $\mathbf{y}$ for any $(i, j) \in \mathcal{E}$:

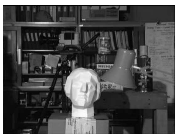
$$\text{load}_{ij} \leqslant w_{ij} d(\alpha, \gamma) + d(\gamma, \beta) \; .$$

After this initial adjustment, $\texttt{PD3}_c$ proceeds exactly as $\texttt{PD2}_{\mu=1}$, except for the fact that the term $d(\alpha, \beta)$ (1) is replaced by

$$\bar{d}(\beta, \gamma) \triangleq \frac{\text{load}_{ij}}{w_{ij}} \leqslant d(\beta, \alpha) + d(\alpha, \gamma) < d(\beta, \gamma) \; .$$

Intuitively, $\texttt{PD3}_c$ works in a complementary way to $\texttt{PD3}_a$ algorithm, i.e. in order to restore the triangle inequality for the conflicting label-triplet (α, β, γ), it chooses to **underestimate the distance** between labels (β, γ) (instead of overestimating the distance between either labels α, γ or α, β).

Results: Stereo matching *


Original (left)


PD1


$\texttt{PD2}_{\mu=1}$ with Potts

Distance $d(\alpha, \beta)$	$\epsilon_{\text{app}}^{\text{PD1}}$	$\epsilon_{\text{app}}^{\text{PD2}_{\mu=1}}$	$\epsilon_{\text{app}}^{\text{PD3}_a}$	$\epsilon_{\text{app}}^{\text{PD3}_b}$	$\epsilon_{\text{app}}^{\text{PD3}_c}$	ϵ_{app}
$[\![\alpha \neq \beta]\!]$	1.0104	1.0058	1.0058	1.0058	1.0058	2
$\min(5, \lvert\alpha - \beta\rvert)$	1.0226	1.0104	1.0104	1.0104	1.0104	10
$\min(5, \lvert\alpha - \beta\rvert^2)$	1.0280	-	1.0143	1.0158	1.0183	10

Branch-and-MinCut

Introduction

We address the problem of **binary image segmentation**, where we also consider *non-local parameters* that are known *a priori*.

For example, one can assume prior knowledge about the **shape** of the foreground segment or the **color distribution** of the foreground and/or background.

Let us consider an *undirected graphical model* $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of pixels and $\mathcal{E}$ consists of 8-connected pairs of pixels. We define the energy function $E : \{0, 1\}^{\mathcal{V}} \times \Omega \to \mathbb{R}$ for non-local parameter $\omega \in \Omega$:

$$E(\mathbf{y}, \omega) = C(\omega) + \sum_{i \in \mathcal{V}} F^i(\omega) \cdot y_i + \sum_{i \in \mathcal{V}} B^i(\omega) \cdot (1 - y_i) + \sum_{(i,j) \in \mathcal{E}} P^{ij}(\omega) \cdot |y_i - y_j| \; ,$$

where $C(\omega)$ is a *constant energy* w.r.t. $\mathbf{y}$, and $F^i(\omega)$ and $B^i(\omega)$ are the *unary energies* defining the cost of assigning the pixel i to the foreground and to the background, respectively. $P^{ij}(\omega) \in \mathbb{R}_0^+$ is **non-negative** for each $(i, j) \in \mathcal{E}$ ensuring the tractability of $E(\mathbf{x}, \omega)$.

Globally optimal segmentation *

The segmentation is given by binary labeling $\mathbf{y} \in \mathbb{B}^{\mathcal{V}} = \{0, 1\}^{\mathcal{V}}$, where individual pixel labels are denoted by $y_i \in \mathbb{B}$ (1:foreground, 0:background). We assume that non-local parameter $\omega \in \Omega$ are taken from a **discrete set**.

Shape priors will be encoded as a product space of various poses and deformations of the *template*, while color priors will correspond to the set of parametric color distributions.

The goal is to achieve a **globally optimal** segmentation under non-local priors. The applied optimization method relies on two techniques: **graph cuts** and **branch-and-bound**.

Although a global minimum can be achieved, the worst case complexity of the method is large (essentially, the same as the exhaustive search over the space of non-local parameters).

An alternative way to solve the problem is to apply *alternating minimization*.

Lower bound

$L(\Omega)$ denotes the lower bound for $E(\mathbf{y},\omega)$ over $\mathbb{B}^{\mathcal{V}} \times \Omega$:

$$\begin{aligned}
&\min_{\mathbf{y}\in\mathbb{B}^{\mathcal{V}},\omega\in\Omega} E(\mathbf{y},\omega) \\
&= \min_{\mathbf{y}\in\mathbb{B}^{\mathcal{V}},\omega\in\Omega} \Big\{ C(\omega) + \sum_{i\in\mathcal{V}} F^i(\omega)\cdot y_i + \sum_{i\in\mathcal{V}} B^i(\omega)\cdot(1-y_i) + \sum_{(i,j)\in\mathcal{E}} P^{ij}(\omega)\cdot|y_i-y_j| \Big\} \\
&\geqslant \min_{\mathbf{y}\in\mathbb{B}^{\mathcal{V}}} \Big\{ \min_{\omega\in\Omega} C(\omega) + \sum_{i\in\mathcal{V}} \min_{\omega\in\Omega} F^i(\omega)\cdot y_i + \sum_{i\in\mathcal{V}} \min_{\omega\in\Omega} B^i(\omega)\cdot(1-y_i) + \\
&\quad \sum_{(i,j)\in\mathcal{E}} \min_{\omega\in\Omega} P^{ij}(\omega)\cdot|y_i-y_j| \Big\} \\
&= \min_{\mathbf{y}\in\mathbb{B}^{\mathcal{V}}} \Big\{ C_\Omega + \sum_{i\in\mathcal{V}} F^i_\Omega(\omega)\cdot y_i + \sum_{i\in\mathcal{V}} B^i_\Omega(\omega)\cdot(1-y_i) + \sum_{(i,j)\in\mathcal{E}} P^{ij}_\Omega(\omega)\cdot|y_i-y_j| \Big\} \\
&= L(\Omega)\ .
\end{aligned}$$

C_Ω, F^i_Ω, B^i_Ω, P^{ij}_Ω denote the minima of $C(\omega)$, $F^i(\omega)$, $B^i(\omega)$, $P^{ij}(\omega)$ over $\omega \in \Omega$ referred to as **aggregated energies**.

Monotonicity

Suppose $\Omega_1 \subset \Omega_2$, then the inequality $L(\Omega_1) \geqslant L(\Omega_2)$ holds.

Proof. Let us define $A(\mathbf{y},\Omega)$ as

$$\begin{aligned}
A(\mathbf{y},\Omega) \triangleq& \min_{\omega\in\Omega} C(\omega) + \sum_{i\in\mathcal{V}} \min_{\omega\in\Omega} F^i(\omega)\cdot y_i + \sum_{i\in\mathcal{V}} \min_{\omega\in\Omega} B^i(\omega)\cdot(1-y_i) \\
&+ \sum_{(i,j)\in\mathcal{E}} \min_{\omega\in\Omega} P^{ij}(\omega)\cdot|y_i-y_j|\ .
\end{aligned}$$

Assume $\Omega_1 \subset \Omega_2$. Then, for any $\mathbf{y} \in \mathbb{B}^{\mathcal{V}}$

$$\begin{aligned}
&A(\mathbf{x},\Omega_1) \\
&= \min_{\omega\in\Omega_1} C(\omega) + \sum_{i\in\mathcal{V}} \min_{\omega\in\Omega_1} F^i(\omega)y_i + \sum_{i\in\mathcal{V}} \min_{\omega\in\Omega_1} B^i(\omega)(1-y_i) + \sum_{(p,q)\in\mathcal{E}} \min_{\omega\in\Omega_1} P^{ij}(\omega)|y_i-y_j| \\
&\geqslant \min_{\omega\in\Omega_2} C(\omega) + \sum_{i\in\mathcal{V}} \min_{\omega\in\Omega_2} F^i(\omega)y_i + \sum_{i\in\mathcal{V}} \min_{\omega\in\Omega_2} B^i(\omega)(1-y_i) + \sum_{(i,j)\in\mathcal{E}} \min_{\omega\in\Omega_2} P^{ij}(\omega)|y_i-y_j| \\
&= A(\mathbf{y},\Omega_2)\ .
\end{aligned}$$

Monotonicity

Proof. Continued

Note that $L(\Omega) = \min_{\mathbf{y}\in\mathbb{B}^{\mathcal{V}}} A(\mathbf{y},\Omega)$.

Let $\mathbf{y}_1 \in \operatorname{argmin}_{\mathbf{y}\in\mathbb{B}^{\mathcal{V}}} A(\mathbf{y},\Omega_1)$ and $\mathbf{y}_2 \in \operatorname{argmin}_{\mathbf{y}\in\mathbb{B}^{\mathcal{V}}} A(\mathbf{y},\Omega_2)$, then from the monotonicity, one gets:

$$L(\Omega_1) = A(\mathbf{y}_1,\Omega_1) \geqslant A(\mathbf{y}_1,\Omega_2) \geqslant A(\mathbf{y}_2,\Omega_2) = L(\Omega_2)\ .$$

□

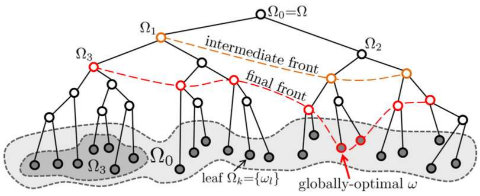
Computability and tightness

Computability: the lower bound $L(\Omega)$ equals the minimum of a *regular function*, which can be globally minimized via graph-cuts.

Tightness: for a singleton $\Omega = \{\omega\}$ (i.e. $|\Omega| = 1$) the bound $L(\Omega)$ is **tight**, that is

$$L(\{\omega\}) = \min_{\mathbf{y}\in\mathbb{B}^{\mathcal{V}}} E(\mathbf{y},\omega)\ .$$

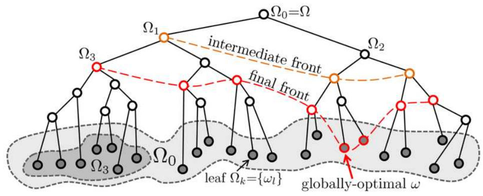
Best-first branch-and-bound optimization



The discrete domain Ω can be hierarchically clustered and the binary tree of its subregions can be considered.

At each step the *active node* with the **smallest** lower bound is removed from the *active front*, while two of its *children* are added to the *active front* (due to *monotonicity property* they have higher or equal lower bounds).

Best-first branch-and-bound optimization



If the *active node* with the smallest lower bound turns out to be a **leaf** ω' and $\mathbf{y}'$ is the corresponding optimal segmentation, then $E(\mathbf{y}',\omega') = L(\omega')$ due to the *tightness property*. Consequently, $(\mathbf{y}',\omega')$ is a **global minimum**.

Remark that in worst-case any optimization has to search exhaustively over Ω.

Pseudo code of Branch-And-Mincut *

```
Front ← ∅                                                      ▷ initializing the priority queue
[C_0, {F_0^i}, {B_0^i}, {P_0^ij}] ←GetAggregPotentials(Ω_0)
LB_0 ←GetMaxFlowValue({F_0^i},{B_0^i},{P_0^ij})+C_0
Front.InsertWithPriority(Ω_0,−LB_0)
while true do                                                  ▷ advancing front
    Ω ← Front.PullHighestPriorityElement()
    if IsSingleton(Ω) then                                     ▷ global minimum found
        ω ← Ω
        [C, {F^i}, {B^i}, {P^ij}] ←GetAggregPotentials(ω)
        x ←FindMinimumViaMincut({F^i},{B^i},{P^ij})
        return (x,ω)
    end if
    [Ω_1, Ω_2] ←GetChildrenSubdomains(Ω)
    [C_1, {F_1^i}, {B_1^i}, {P_1^ij}] ←GetAggregPotentials(Ω_1)
    LB_1 ←GetMaxFlowValue({F_1^i},{B_1^i},{P_1^ij})+C_1
    Front.InsertWithPriority(Ω_1,−LB_1)
    [C_2, {F_2^i}, {B_2^i}, {P_2^ij}] ←GetAggregPotentials(Ω_2)
    LB_2 ←GetMaxFlowValue({F_2^i},{B_2^i},{P_2^ij})+C_2
    Front.InsertWithPriority(Ω_2,−LB_2)
end while
```


Segmentation with shape priors

The prior is defined by the set of exemplar binary segmentations $\{\mathbf{x}^\omega \mid \omega \in \Omega\}$, where Ω is a discrete set indexing the exemplar segmentations.

We define a joint prior over the segmentation and the non-local parameter:

$$E_{\text{prior}}(\mathbf{y},\omega) = \sum_{i\in\mathcal{V}} (1 - x_i^\omega)\cdot y_i + \sum_{i\in\mathcal{V}} x_i^\omega \cdot (1-y_i)\ .$$

This encourages the segmentation $\mathbf{y}$ to be close in the Hamming-distance ($d_{\mathsf{H}}(\mathbf{a},\mathbf{b}) = \frac{1}{N}\sum_{i=1}^{N} [\![a_i \neq b_i]\!]$) to one of the exemplar shapes.

The segmentation energy may be defined by adding a standard *contrast-sensitive Potts-model* for $\lambda, \sigma > 0$:

$$E(\mathbf{y},\omega) = E_{\text{prior}}(\mathbf{y},\omega) + \lambda \sum_{(i,j)\in\mathcal{E}} \frac{e^{-\frac{\|I_i - I_j\|}{\sigma}}}{|i-j|}\cdot|y_i - y_j|\ ,$$

where I_i denotes RGB colors of the pixel i.

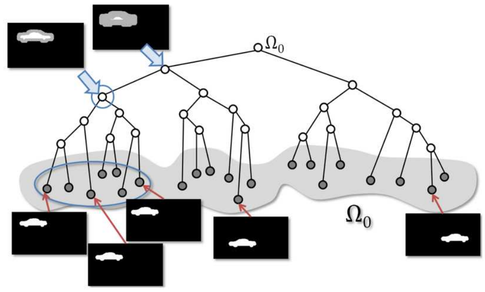
Parameterization: multiple templates × translations

The shape prior is given by a set of templates, whereas each template can be located anywhere within the image.

$\Omega = \Delta \times \Theta$, where the set Δ indexes the set of all exemplar segmentations x_δ and Θ corresponds to translations.

Any exemplar segmentation $\mathbf{x}^\omega$ for $\omega = (\delta, \theta)$ is then defined as some *exemplar segmentation* x_δ centered at the origin and then *translated* by the shift θ.

Clustering tree



For Δ we use agglomerative bottom-up clustering resulting in a (binary) clustering tree $T_\Delta = \{\Delta = \Delta_0, \Delta_1, \ldots, \Delta_N\}$.

To build a clustering tree for Θ, we recursively split along the "longer" dimension. This leads to a (binary) tree $T_\Theta = \{\Theta = \Theta_0, \Theta_1, \ldots, \Theta_N\}$.

Branch operation

Each *nodeset* Ω_t in the combined tree is defined by a pair $\Delta_t \times \Theta_t$.

The **looseness** of a nodeset Ω_t is defined as the number of pixels that change their mask value under different shapes in Ω_t (i.e. neither background nor foreground):

$$\Lambda(\Omega_t) = |\{i \mid \exists \omega_1, \omega_2 : x_i^{\omega_1} = 0 \text{ and } x_i^{\omega_2} = 1\}| .$$

The tree is built in a recursive top-down fashion as follows.

We start by creating a root nodeset $\Omega_0 = \Delta_0 \times \Theta_0$. Given a nodeset $\Omega_t = \Delta_t \times \Theta_t$ we consider (recursively) two possible splits: 1) split along the shape dimension or 2) split along the shift dimension. The split that minimizes the sum of loosenesses is preferred.

The recursion stops when the leaf level is reached within both the *shape* and the *shift* trees.

Results *

Yellow: global minimum of E; Blue: feature-based car detector; Red: global minimum of the combination of E with detection results (detection is included as a constant potential)

The prior set Δ was built by manual segmentation of 60 training images coming with the dataset.

Summary *

- Primal-dual schema:

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} E_i(x_i) + \sum_{(i,j) \in \mathcal{E}} w_{ij} \cdot d(x_i, x_j)$$

  - PD1: d is a semi-metric
  - PD2: d is a metric (equivalent to α-expansion)
  - PD3: d is a non-metric function
- For **binary image segmentation** we learned a global optimal solution, based on *branch and bound* optimization, in the presence of (shape) prior information.

In the **next lecture** we will learn about **exact inference** (probabilistic and MAP) on tree structured factor graphs.

Literature *

FastPD

1. Nikos Komodakis and Georgios Tziritas. Approximate labeling via the primal-dual schema. Technical report, University of Crete, February 2005
2. Nikos Komodakis and Georgios Tziritas. Approximate labeling via graph-cuts based on linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1436–1453, August 2007

Branch-and-MinCut

1. Victor Lempitsky, Andrew Blake, and Carsten Rother. Branch-and-mincut: Global optimization for image segmentation with high-level priors. *Journal of Mathematical Imaging and Vision*, 44(3):315–329, March 2012