Probabilistic Graphical Models in Computer Vision (IN2329)

Csaba Domokos

Summer Semester 2015/2016

12. Summary * 2
Overview of the course 3

Probability theory **4**
Probability, conditional probability 5
Independence, conditional independence 6
Random variable, probability distribution 7
Joint and marginal distribution 8
Conditional distribution 9
The EM algorithm. 10

Graphical models **11**
Graphical models. 12
Bayesian networks 13
Markov random field 14
Hammersley-Clifford theorem 15
Factor graphs 16
Conditional random fields 17
Potentials and energy functions 18

Energy minimization . . . 19
Overview . . . 20

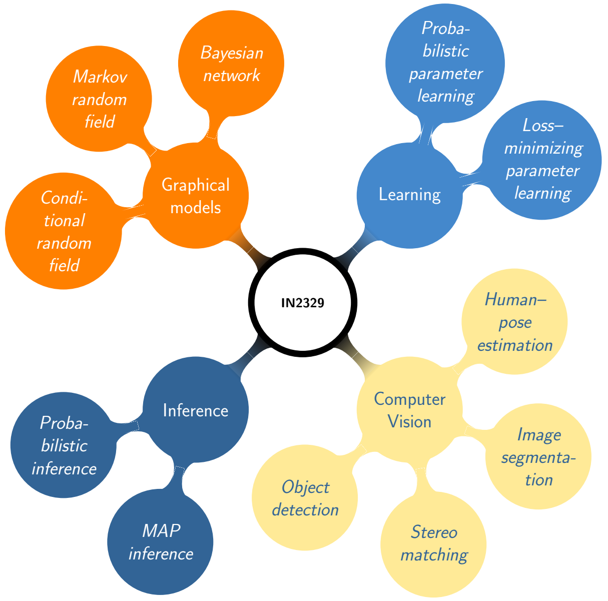
Inference **21**
Inference . . . 22
Graph cut . . . 23
Regular energy functions . . . 24
Energy minimization via minimum $s-t$ cut: unary energies . . . 25
Energy minimization via minimum $s-t$ cut: pairwise energies . . . 26
Energy minimization via minimum $s-t$ cut . . . 27
Multi-label problem . . . 28
Move making algorithms . . . 29
$\alpha-\beta$ swap . . . 30
α-expansion . . . 31
Equivalent integer linear program . . . 32
Interpretation of the constraints . . . 33
Primal-dual LP for multi-label problem . . . 34
Primal-dual principle . . . 35
FastPD . . . 36
Best-first branch-and-bound optimization . . . 37
Belief propagation . . . 38
Messages . . . 39
Factor-to-variable message . . . 40
Inference result . . . 41
Sum-product and Max-sum comparison * . . . 42
Loopy belief propagation . . . 43
Mean field approximation . . . 44
Naïve mean field . . . 45
Naïve mean field: Optimization . . . 46
Gibbs sampling . . . 47
Overview . . . 48

Parameter learning **49**
Parameter learning . . . 50

Probabilistic parameter learning . . . 51
Regularized maximum conditional likelihood training . . . 52
Stochastic gradient descent . . . 53
Using of the output structure . . . 54
Two-stage learning . . . 55
Piecewise learning . . . 56
Loss-minimizing parameter learning . . . 57
Overview . . . 58

Computer vision **59**
Overview . . . 60
Announcement: Computer Vision Group . . . 61

Overview of the course



Probability theory

Probability, conditional probability

A **probability space** is a triple $(\Omega, \mathcal{A}, P)$, where $(\Omega, \mathcal{A})$ is a *measurable space*, and P is a *measure* such that $P(\Omega) = 1$. We called **discrete probability space**, if $\Omega =\neq \varnothing$ is countable.

Let $P(B) > 0$, then the **conditional probability of** A **given** B is defined as

$$P(A \mid B) \triangleq \frac{P(A \cap B)}{P(B)} \,.$$

Independence, conditional independence

If two events A and B are **independent** ($A \perp B$), learning that B happened does not make A more or less likely to occur:

$$P(A \mid B) = P(A)$$

or, equivalently, iff

$$P(A \cap B) = P(A)P(B) .$$

A and B are **conditionally independent** given C means that *once we learned C, learning B gives us no additional information about A.*

$$P(A \mid C) = P(A \mid B \cap C) ,$$

or, equivalently, iff

$$P(A \cap B \mid C) = P(A \mid C)P(B \mid C) .$$

Random variable, probability distribution

A *measurable mapping* $X : (\Omega, \mathcal{A}) \to (\mathbb{R}, \mathcal{A}')$ is called **random variable**.

Let $X : (\Omega, \mathcal{A}) \to (\Omega' \subseteq \mathbb{R}, \mathcal{A}')$ be a *random variable* and P a *measure* over $\mathcal{A}$. Then

$$P'(A') := P_X(A') \triangleq P(X^{-1}(A'))$$

defines a measure over $\mathcal{A}'$. P_X is called the **image measure** of P by X.

The *image measure* P_X of P by X is called **probability distribution**. $F_X : \mathbb{R} \to \mathbb{R}$

$$F_X(x) \triangleq P(X < x) , \quad x \in \mathbb{R}$$

is called **cumulative distribution function** (cdf.) of X.

Joint and marginal distribution

Suppose a probability space $(\Omega, \mathcal{A}, P)$. Let $X : (\Omega, \mathcal{A}) \to (\Omega', \mathcal{A}')$ and $Y : (\Omega, \mathcal{A}) \to (\Omega'', \mathcal{A}'')$ be *discrete* random variables, where $x_1, x_2, \dots$ denote the values of X and $y_1, y_2, \dots$ denote the values of Y.

We introduce the notation

$$p_{ij} \triangleq P(X = x_i, Y = y_j) \quad i, j = 1, 2, \dots$$

for the probability of the *events* $\{X = x_i, Y = y_j\} := \{X = x_i\} \cap \{Y = y_j\}$. These probabilities p_{ij} form a *distribution*, called the **joint distribution** of X and Y.

The *distributions* defined by the probabilities

$$p_i \triangleq P(X = x_i) \quad \text{and} \quad q_j \triangleq P(Y = y_j)$$

are called the **marginal distributions** of X and of Y, respectively.

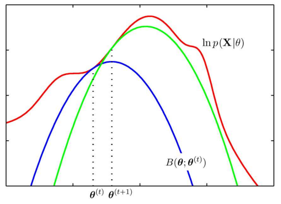
Conditional distribution

Suppose a probability space $(\Omega, \mathcal{A}, P)$. Let X and Y be *discrete random variables*, where $x_1, x_2, \dots$ denote the values of X and $y_1, y_2, \dots$ denote the values of Y.

The **conditional distribution** of X given Y is defined by

$$P(X = x_i \mid Y = y_j) = \frac{P(X = x_i, Y = y_j)}{P(Y = y_j)} = \frac{p_{ij}}{\sum_k p_{kj}} = \frac{p_{ij}}{q_j} .$$

The EM algorithm



```
1: Choose an initial setting for the parameters $\boldsymbol{\theta}^{(0)}$
2: $t \to 0$
3: **repeat**
4:     $t \to t+1$
5:     **E step**. Evaluate $q^{(t-1)}(\mathbf{Z}) \triangleq p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{(t-1)})$
6:     **M step**. Evaluate $\boldsymbol{\theta}^{(t)}$ given by
           $\boldsymbol{\theta}^{(t)} = \operatorname*{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)})$ ,
           where $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)}) \triangleq \mathbb{E}[\ln p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) \mid \mathbf{X}, \boldsymbol{\theta}^{(t-1)}]$
               $= \sum_{\mathbf{Z}} p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{(t-1)}) \ln p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})$
7: **until** convergence of either the parameters $\boldsymbol{\theta}$ or the log likelihood $\mathcal{L}(\boldsymbol{\theta}; \mathbf{X})$
```

Graphical models

Graphical models

Probabilistic graphical models encode a joint $p(\mathbf{x}, \mathbf{y})$ or conditional $p(\mathbf{y} \mid \mathbf{x})$ probability distribution such that given some observations we are provided with a full probability distribution over all feasible solutions.

The graphical models allow us to encode relationships between a set of random variables using a concise language, by means of a **graph**.

- Directed:
  - Bayesian network
- Undirected:
  - Markov random field
    - Factor graphs
    - Conditional random field

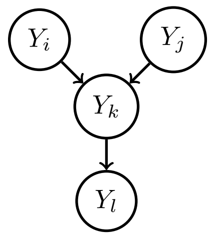
Bayesian networks

Assume a *directed, acyclic* graphical model $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$.

The factorization is given as

$$p(\mathbf{Y} = \mathbf{y}) = \prod_{i \in \mathcal{V}} p(y_i \mid \mathbf{y}_{\mathrm{pa}_G(i)}) \ ,$$

where $p(y_i \mid \mathbf{y}_{\mathrm{pa}_G(i)})$ is a conditional probability distribution on the parents of node $i \in \mathcal{V}$.



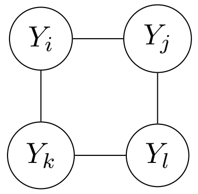
The *conditional independence assumption* is encoded by G that is a variable is conditionally independent of its non-descendants given its parents.

Markov random field

An *undirected graphical model* $G = (\mathcal{V}, \mathcal{E})$ is called **Markov Random Field** (MRF) if two nodes are conditionally independent whenever they are not connected.
In other words, for any node i in the graph, the **local Markov property** holds:

$$p(Y_i \mid Y_{\mathcal{V} \setminus \{i\}}) = p(Y_i \mid Y_{N(i)}) \ ,$$

where $N(i)$ is denotes the neighbors of node i in the graph.



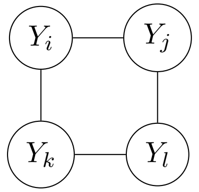
Alternatively, we use the following equivalent notation:

$$Y_i \perp\!\!\!\perp Y_{\mathcal{V} \setminus \mathrm{cl}(i)} \mid Y_{N(i)} \ ,$$

where $\mathrm{cl}(i) = N(i) \cup \{i\}$ is the *closed neighborhood* of i.

Hammersley-Clifford theorem

- An *undirected graphical model* $G = (\mathcal{V}, \mathcal{E})$ is called **Markov random field**, if the **local Markov property** holds, i.e. two nodes are conditionally independent whenever they are not connected.



- A *probability distribution* $p(\mathbf{y})$ on an *undirected graphical model* $G = (\mathcal{V}, \mathcal{E})$ is called **Gibbs distribution** if it can be factorized into potential functions $\psi_c(\mathbf{y}_c) > 0$ defined on cliques:

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{c \in \mathcal{C}_G} \psi_c(\mathbf{y}_c) \text{ , where } Z = \sum_{\mathbf{y} \in \mathcal{Y}} \prod_{c \in \mathcal{C}_G} \psi_c(\mathbf{y}_c) \text{ ,}$$

  and $\mathcal{C}_G$ denotes the set of all (maximal) cliques in G.

The Hammersley-Clifford theorem tells us that the above two definitions are equivalent.

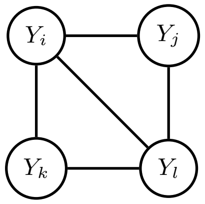
Factor graphs

Factor graphs are *undirected graphical models* that **make the factorization explicit** of the probability function.
A factor graph $G = (\mathcal{V}, \mathcal{F}, \mathcal{E}')$ consists of

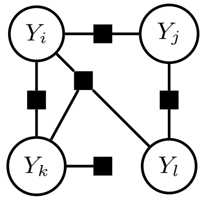
- variable nodes V (◯) and factor nodes $\mathcal{F}$ (■),
- edges $\mathcal{E}' \subseteq V \times \mathcal{F}$ between variable and factor nodes
- $N : \mathcal{F} \to 2^V$ is the *scope of a factor*, defined as the **set of neighboring variables**, i.e. $N(F) = \{i \in V : (i, F) \in \mathcal{E}\}$.

A family of distribution is defined that factorizes as:

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(\mathbf{y}_{N(F)}) \quad \text{with} \quad Z = \sum_{\mathbf{y} \in \mathcal{Y}} \prod_{F \in \mathcal{F}} \psi_F(\mathbf{y}_{N(F)}) \text{ .}$$



MRF



Factor graph

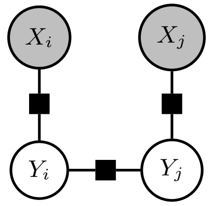
Conditional random fields

We often have access to measurements $\mathbf{X} = \mathbf{x}$, hence the **conditional distribution** $p(\mathbf{Y} = \mathbf{y} \mid \mathbf{X} = \mathbf{x})$ could be directly modeled, too. This can be expressed compactly using **conditional random fields** (CRF) with the factorization

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{p(\mathbf{y}, \mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{y}, \mathbf{x})}{\sum_{\mathbf{y}' \in \mathcal{Y}} p(\mathbf{y}', \mathbf{x})} = \frac{1}{Z(\mathbf{x})} \prod_{F \in \mathcal{F}} \psi_F(\mathbf{y}_{N(F)}; \mathbf{x}_{N(F)}) \ .$$

Note that the potentials become also functions of (part of) $\mathbf{x}$, i.e. $\psi_F(\mathbf{y}_F; \mathbf{x}_F)$ instead of just $\psi_F(\mathbf{y}_F)$.

Nevertheless, $\mathbf{X}$ is **not** part of the probability model, i.e. it is not treated as random vector.



Shaded variables: The observations $\mathbf{X} = \mathbf{x}$.

Potentials and energy functions

We typically would like to infer marginal probabilities $p(\mathbf{Y}_F = \mathbf{y}_F \mid \mathbf{x})$ for some factors $F \in \mathcal{F}$.

Assuming $\psi_F : \mathcal{Y}_F \to \mathbb{R}^+$, where $\mathcal{Y}_F = \times_{i \in N(F)} \mathcal{Y}_i$ is the product domain of the variables adjacent to F, instead of *potentials*, we can also work with **energies**.

We define an **energy function** $E_F : \mathcal{Y}_F \to \mathbb{R}$ for each factor $F \in \mathcal{F}$:

$$E_F(\mathbf{y}_F; \mathbf{x}_F) = -\log(\psi_F(\mathbf{y}_F; \mathbf{x}_F)) \quad \Leftrightarrow \quad \psi_F(\mathbf{y}_F; \mathbf{x}_F) = \exp(-E_F(\mathbf{y}_F; \mathbf{x}_F)) \ .$$

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{x}) &= \frac{1}{Z(\mathbf{x})} \prod_{F \in \mathcal{F}} \psi_F(\mathbf{y}_F; \mathbf{x}_F) = \frac{1}{Z(\mathbf{x})} \exp(-\sum_{F \in \mathcal{F}} E_F(\mathbf{y}_F; \mathbf{x}_F)) \\ &= \frac{1}{Z(\mathbf{x})} \exp(-E(\mathbf{y}; \mathbf{x})) \ . \end{aligned}$$

Hence, $p(\mathbf{y} \mid \mathbf{x})$ is completely determined by $E(\mathbf{y}; \mathbf{x})$.

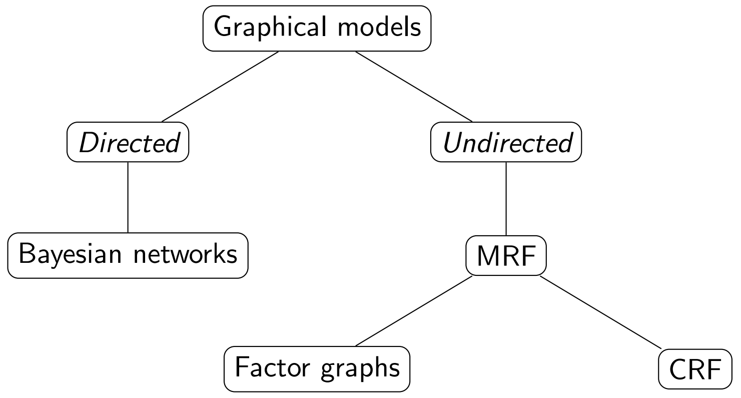
Energy minimization

Assuming a finite $\mathcal{X}$, the goal is to solve $\mathbf{y}^* \in \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y} \mid \mathbf{x})$.

$$\begin{aligned}
\underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\ p(\mathbf{y} \mid \mathbf{x}) &= \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\ \frac{1}{Z(\mathbf{x})} \exp(-E(\mathbf{y};\mathbf{x})) \\
&= \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\ \exp(-E(\mathbf{y};\mathbf{x})) \\
&= \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\ -E(\mathbf{y};\mathbf{x}) \\
&= \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmin}}\ E(\mathbf{y};\mathbf{x}) \ .
\end{aligned}$$

Energy minimization can be interpreted as solving for the most likely state of factor graph, i.e. MAP inference.

Overview



Inference

Inference

The **inference** means the procedure to estimate the *probability distribution*, encoded by the *graphical model*, for a *given data* (or observation).

Probabilistic inference: Given a graphical model and the observation x, find the value of the *log partition function* and the *marginal distributions* for each factor,

$$\log Z(\mathbf{x}) = \log \sum_{\mathbf{y} \in \mathcal{Y}} \exp(-E(\mathbf{y}; \mathbf{x})) \ ,$$

$$\mu_F(y_F) = p(\mathbf{Y}_F = \mathbf{y}_F \mid \mathbf{x}) \quad \forall F \in \mathcal{F}, \ \forall \mathbf{y}_F \in \mathcal{Y}_F \ .$$

Maximum A Posteriori (MAP) inference: Given a graphical model and the observation $\mathbf{x}$, find the *state* $\mathbf{y}^* \in \mathcal{Y}$ *of maximum probability*

$$\mathbf{y}^* \in \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{Y} = \mathbf{y} \mid \mathbf{x}) \ .$$

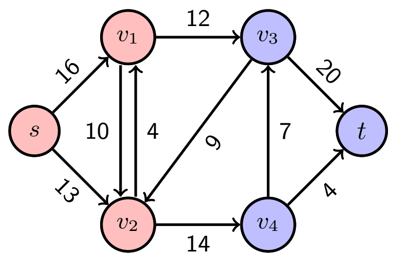
Both inference problems are known to be NP-hard for general graphs and factors, but they can be tractable if the underlying graphical model is suitably restricted.

Graph cut

Assume a *weighted directed graph* $G = (\mathcal{V}, \mathcal{E}, c)$. A **cut** $(\mathcal{S}, \mathcal{T})$ of G is a *disjoint* partition of $\mathcal{V}$ into $\mathcal{S}$ and $\mathcal{T} = \mathcal{V} \backslash \mathcal{S}$.

The **capacity** of the cut $(\mathcal{S}, \mathcal{T})$ is defined as

$$\text{cut}(\mathcal{S}, \mathcal{T}) = \sum_{(i,j) \in \mathcal{S} \times \mathcal{T}} c(i, j) \ .$$



Assume distinct nodes $s, t \in \mathcal{V}$, a cut $(\mathcal{S}, \mathcal{T})$ is called $s - t$ **cut** if $s \in \mathcal{S}$ and $t \in \mathcal{T}$.

The **minimum** $s - t$ **cut problem** is to find an $s - t$ cut with the lowest cost.

Regular energy functions

Let us consider an *energy function* E of n **binary variables** which can be written as the sum of functions of up to two variables, that is

$$E(y_1, \dots, y_n) = \sum_i E_i(y_i) + \sum_{i<j} E_{ij}(y_i, y_j) \; .$$

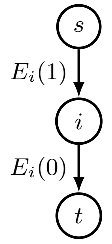
E is *regular*, if each term E_{ij} $(i < j)$ satisfies

$$E_{ij}(0,0) + E_{ij}(1,1) \leqslant E_{ij}(0,1) + E_{ij}(1,0) \; .$$

If each term E_{ij} is *regular*, then it is possible to find the **global** minimum of E in *polynomial time* by solving a *minimum $s-t$ cut problem*.

Energy minimization via minimum $s-t$ cut: unary energies

Let us consider the unary energy function $E_i : \{0,1\} \to \mathbb{R}$.

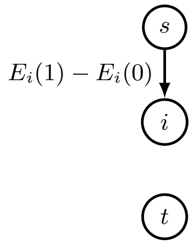


Obviously, the minimum $s-t$ cut of the flow network will correspond to

$$\operatorname*{argmin}_{y_i \in \{0,1\}} E_i(y_i) \; .$$

Without loss of generality we can assume that $E_i(1) > E_i(0)$, then we can write

$$\operatorname*{argmin}_{y_i \in \{0,1\}} E_i(y_i) = \operatorname*{argmin}_{y_i \in \{0,1\}} E_i(y_i) - E_i(0) \; .$$



Energy minimization via minimum $s-t$ cut: pairwise energies

Let us consider the pairwise energy function $E_{ij}(y_i, y_j) : \{0,1\}^2 \to \mathbb{R}$. The possible values of $E_{ij}(y_i, y_j)$ are shown in the table:

E_{ij}	$y_j = 0$	$y_j = 1$
$y_i = 0$	A	B
$y_i = 1$	C	D

We furthermore assume that $E_{ij}(y_i, y_j)$ is regular, that is

$$\begin{aligned} E_{ij}(0,0) + E_{ij}(1,1) \leqslant & E_{ij}(0,1) + E_{ij}(1,0) \\ A + D \leqslant & B + C \ . \end{aligned}$$

Let us note that $E_{ij}(y_i, y_j)$ can be decomposed as:

$$\begin{array}{|c|c|}\hline A & B \\ \hline C & D \\ \hline \end{array} = A + \underbrace{\begin{array}{|c|c|}\hline 0 & 0 \\ \hline C-A & C-A \\ \hline \end{array}}_{E_i(1)} + \underbrace{\begin{array}{|c|c|}\hline 0 & D-C \\ \hline 0 & D-C \\ \hline \end{array}}_{E_j(1)} + \underbrace{\begin{array}{|c|c|}\hline 0 & B+C-A-D \\ \hline 0 & 0 \\ \hline \end{array}}_{B+C-A-D \geqslant 0}$$

Energy minimization via minimum $s-t$ cut

Putting all together we get that

Unaries

$$\operatorname*{argmin}_{\mathbf{y}} E_i(y_i) + E_j(y_j)$$

$$\begin{tikzcd}
 & s \arrow[dl, "E_i(1)"'] \arrow[dr, "E_j(1)"] & \\
i \arrow[dr, "E_i(0)"'] & & j \arrow[dl, "E_j(0)"] \\
 & t &
\end{tikzcd}$$

Pairwise

$$\operatorname*{argmin}_{\mathbf{y}} E_{ij}(y_i, y_j)$$

$$\begin{tikzcd}
 & s \arrow[dl, "C"'] \arrow[dr, "D"] & \\
i \arrow[rr, "B+C-A-D"] \arrow[dr, "A"'] & & j \arrow[dl, "C"] \\
 & t &
\end{tikzcd}$$

Overall energy

$$\operatorname*{argmin}_{\mathbf{y}} E_i(y_i) + E_j(y_j)$$
$$+ E_{ij}(y_i, y_j)$$

$$\begin{tikzcd}
 & s \arrow[dl, "E_i(1)+C"'] \arrow[dr, "E_j(1)+D"] & \\
i \arrow[rr, "B+C-A-D"] \arrow[dr, "E_i(0)+A"'] & & j \arrow[dl, "E_j(0)+C"] \\
 & t &
\end{tikzcd}$$

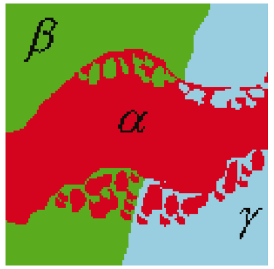
Multi-label problem

We define a label set $\mathcal{L} = \{1, 2, \dots, L\}$, where L is a (finite) constant. Therefore the output domain is defined as $\mathcal{Y} = \mathcal{L}^{\mathcal{V}}$. The *energy function* has the following form

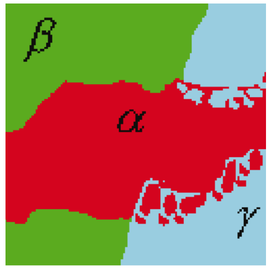
$$E(\mathbf{y};\mathbf{x}) = \sum_{i \in \mathcal{V}} E_i(y_i;\mathbf{x}) + \sum_{(i,j) \in \mathcal{E}} E_{ij}(y_i, y_j;\mathbf{x}) \ ,$$

where $\mathbf{x}$ consists of an input image.

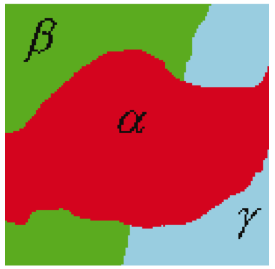
Move making algorithms



Regions



α-β swap



α-expansion

Assumptions:

- $\alpha - \beta$ swap: E_{ij} is a semi-metric.
- α-expansion: E_{ij} is a metric.

$$\mathcal{Z}_{\alpha\beta}(\mathbf{y}, \alpha, \beta) = \{\mathbf{z} \in \mathcal{Y} : z_i = y_i, \text{ if } y_i \notin \{\alpha, \beta\}, \text{ otherwise } z_i \in \{\alpha, \beta\}\} \ .$$

$$\mathcal{Z}_{\alpha}(\mathbf{y}, \alpha) = \{\mathbf{z} \in \mathcal{Y} : z_i \in \{y_i, \alpha\} \text{ for all } i \in \mathcal{V}\} \ .$$

$\alpha - \beta$ **swap**

$\alpha - \beta$ **swap** changes the variables that are labeled as $\ell \in \{\alpha, \beta\}$. Each of these variables can choose either α or β. We introduce the following notation

$$\mathcal{Z}_{\alpha\beta}(\mathbf{y}, \alpha, \beta) = \{\mathbf{z} \in \mathcal{Y} : z_i = y_i, \text{ if } y_i \notin \{\alpha, \beta\}, \text{ otherwise } z_i \in \{\alpha, \beta\}\} .$$

The minimization of the *energy function* E can be reformulated as follows:

$$\begin{aligned}
\hat{\mathbf{z}} \in & \operatorname*{argmin}_{\mathbf{z} \in \mathcal{Z}_{\alpha\beta}(\mathbf{y},\alpha,\beta)} E(\mathbf{z}) = \operatorname*{argmin}_{\mathbf{z} \in \mathcal{Z}_{\alpha\beta}(\mathbf{y},\alpha,\beta)} \sum_{i \in \mathcal{V}} E_i(z_i) + \sum_{(i,j) \in \mathcal{E}} E_{ij}(z_i, z_j) \\
= & \operatorname*{argmin}_{\mathbf{z} \in \mathcal{Z}_{\alpha\beta}(\mathbf{y},\alpha,\beta)} \Bigg[\underbrace{\sum_{i \in \mathcal{V},\, y_i \notin \{\alpha,\beta\}} E_i(y_i)}_{\text{constant}} + \underbrace{\sum_{i \in \mathcal{V},\, y_i \in \{\alpha,\beta\}} E_i(z_i)}_{\text{unary}} \\
& + \underbrace{\sum_{\substack{(i,j) \in \mathcal{E} \\ y_i, y_j \notin \{\alpha,\beta\}}} E_{ij}(y_i, y_j)}_{\text{constant}} + \underbrace{\sum_{\substack{(i,j) \in \mathcal{E} \\ y_i \in \{\alpha,\beta\},\, y_j \notin \{\alpha,\beta\}}} E_{ij}(z_i, y_j)}_{\text{unary}} + \underbrace{\sum_{\substack{(i,j) \in \mathcal{E} \\ y_i \notin \{\alpha,\beta\},\, y_j \in \{\alpha,\beta\}}} E_{ij}(y_i, z_j)}_{\text{unary}} + \underbrace{\sum_{\substack{(i,j) \in \mathcal{E} \\ y_i, y_j \in \{\alpha,\beta\}}} E_{ij}(z_i, z_j)}_{\text{pairwise}} \Bigg].
\end{aligned}$$

α-expansion

α-**expansion** allows each variable either to keep its current label or to change it to the label $\alpha \in \mathcal{L}$. We introduce the following notation

$$\mathcal{Z}_\alpha(\mathbf{y}, \alpha) = \{\mathbf{z} \in \mathcal{Y} : z_i \in \{y_i, \alpha\} \text{ for all } i \in \mathcal{V}\} \ .$$

The minimization of the *energy function* E can be reformulated as follows:

$$\begin{aligned}
\hat{\mathbf{z}} \in \underset{\mathbf{z} \in \mathcal{Z}_\alpha(\mathbf{y},\alpha)}{\operatorname{argmin}} \ E(\mathbf{z}) &= \underset{\mathbf{z} \in \mathcal{Z}_\alpha(\mathbf{y},\alpha)}{\operatorname{argmin}} \sum_{i \in \mathcal{V}} E_i(z_i) + \sum_{(i,j) \in \mathcal{E}} E_{ij}(z_i, z_j) \\
&= \underset{\mathbf{z} \in \mathcal{Z}_\alpha(\mathbf{y},\alpha)}{\operatorname{argmin}} \Bigg[\underbrace{\sum_{i \in \mathcal{V},\, y_i = \alpha} E_i(\alpha)}_{\text{constant}} + \underbrace{\sum_{i \in \mathcal{V},\, y_i \neq \alpha} E_i(z_i)}_{\text{unary}} \\
&\quad + \underbrace{\sum_{\substack{(i,j) \in \mathcal{E} \\ y_i = \alpha,\, y_j = \alpha}} E_{ij}(\alpha, \alpha)}_{\text{constant}} + \underbrace{\sum_{\substack{(i,j) \in \mathcal{E} \\ y_i = \alpha,\, y_j \neq \alpha}} E_{ij}(\alpha, z_j)}_{\text{unary}} + \underbrace{\sum_{\substack{(i,j) \in \mathcal{E} \\ y_i \neq \alpha,\, y_j = \alpha}} E_{ij}(z_i, \alpha)}_{\text{unary}} + \underbrace{\sum_{\substack{(i,j) \in \mathcal{E} \\ y_i \neq \alpha,\, y_j \neq \alpha}} E_{ij}(z_i, z_j)}_{\text{pairwise}} \Bigg] .
\end{aligned}$$

Equivalent integer linear program

We are generally interested to find a *MAP labelling* $\mathbf{x}^*$:

$$\mathbf{x}^* \in \operatorname*{argmin}_{\mathbf{x} \in \mathcal{L}^{|\mathcal{V}|}} E(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{L}^{|\mathcal{V}|}} \Big\{ \sum_{i \in \mathcal{V}} E_i(x_i) + \sum_{(i,j) \in \mathcal{E}} w_{ij} \cdot d(x_i, x_j) \Big\} \,.$$

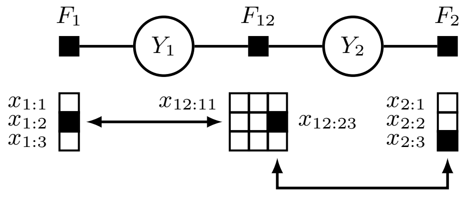
This can be equivalently written as an **integer linear program** (ILP):

$$\begin{aligned} \min_{x_{i:\alpha}, x_{ij:\alpha\beta}} \sum_{i \in \mathcal{V}} \sum_{\alpha \in \mathcal{L}} E_i(\alpha) x_{i:\alpha} + \sum_{(i,j) \in \mathcal{E}} w_{ij} \sum_{\alpha, \beta \in \mathcal{L}} d(\alpha, \beta) x_{ij:\alpha\beta} \\ \text{subject to} \quad & \textstyle\sum_{\alpha \in \mathcal{L}} x_{i:\alpha} && = 1 && \forall i \in \mathcal{V} \\ & \textstyle\sum_{\alpha \in \mathcal{L}} x_{ij:\alpha\beta} && = x_{j:\beta} && \forall \beta \in \mathcal{L}, (i,j) \in \mathcal{E} \\ & \textstyle\sum_{\beta \in \mathcal{L}} x_{ij:\alpha\beta} && = x_{i:\alpha} && \forall \alpha \in \mathcal{L}, (i,j) \in \mathcal{E} \\ & x_{i:\alpha}, x_{ij:\alpha\beta} \in \mathbb{B} && && \forall \alpha, \beta \in \mathcal{L}, (i,j) \in \mathcal{E} \end{aligned}$$

$x_{i:\alpha}$ indicates whether vertex i is assigned label α, while $x_{ij:\alpha\beta}$ indicates whether (neighboring) vertices i, j are assigned labels α, β, respectively.

Interpretation of the constraints

Let us assume that $\mathcal{L} = \{1, 2, 3\}$ and consider the following factor graph example:



Uniqueness: The constraints $\sum_{\alpha \in \mathcal{L}} x_{i:\alpha} = 1$ for all $i \in \mathcal{V}$ simply express the fact that each vertex must receive exactly one label.

Consistency: The constraints

$$\sum_{\alpha \in \mathcal{L}} x_{ij:\alpha\beta} = x_{j:\beta} \quad \text{and} \quad \sum_{\beta \in \mathcal{L}} x_{ij:\alpha\beta} = x_{i:\alpha} \quad \forall \alpha, \beta \in \mathcal{L}\ , (i, j) \in \mathcal{E}$$

maintain consistency between variables, i.e. if $x_{i:\alpha} = 1$ and $x_{j:\beta} = 1$ holds true, then these constraints force $x_{ij:\alpha\beta} = 1$ to hold true as well.

Primal-dual LP for multi-label problem

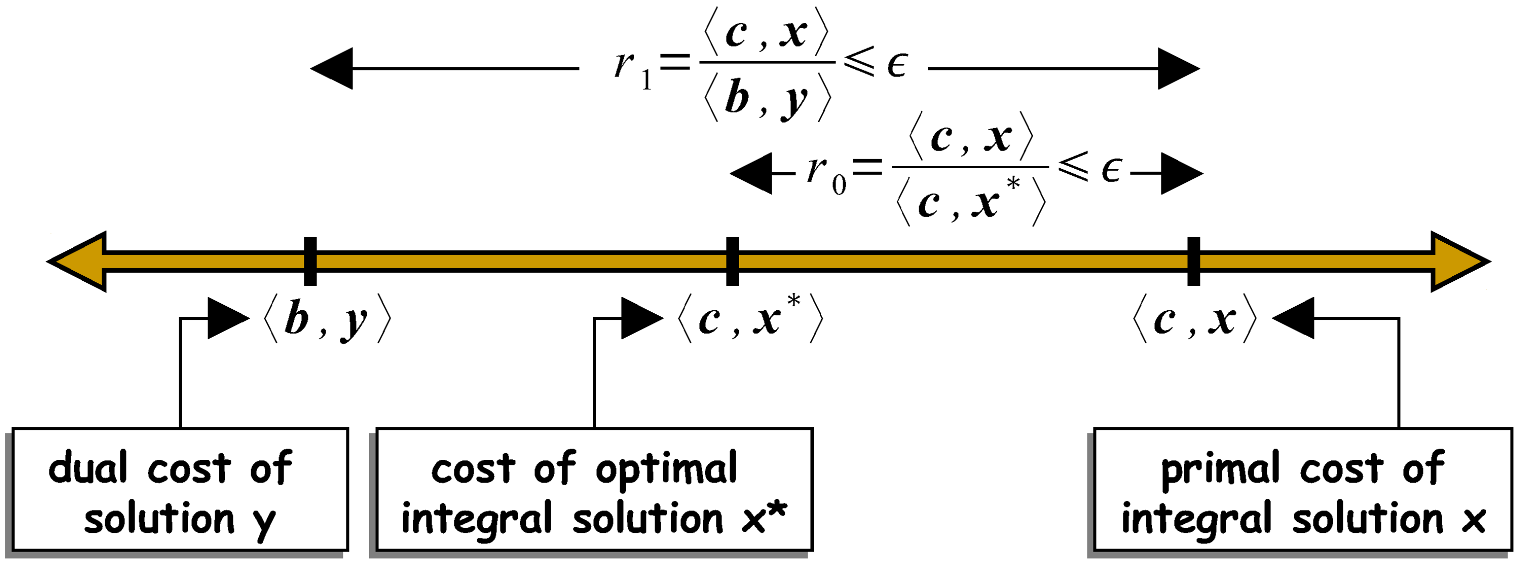
The (relaxed) primal LP:

$$\begin{aligned}
\min_{x_{i:\alpha}, x_{ij:\alpha\beta} \geqslant 0} & \sum_{i \in \mathcal{V}} \sum_{\alpha \in \mathcal{L}} E_i(\alpha) x_{i:\alpha} + \sum_{(i,j) \in \mathcal{E}} w_{ij} \sum_{\alpha, \beta \in \mathcal{L}} d(\alpha, \beta) x_{ij:\alpha\beta} \\
\text{subject to} \quad & \textstyle\sum_{\alpha \in \mathcal{L}} x_{i:\alpha} = 1 \quad \forall i \in \mathcal{V} \\
& \textstyle\sum_{\alpha \in \mathcal{L}} x_{ij:\alpha\beta} = x_{j:\beta} \quad \forall \beta \in \mathcal{L}, (i,j) \in \mathcal{E} \\
& \textstyle\sum_{\beta \in \mathcal{L}} x_{ij:\alpha\beta} = x_{i:\alpha} \quad \forall \alpha \in \mathcal{L}, (i,j) \in \mathcal{E}
\end{aligned}$$

The dual LP:

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

Primal-dual principle



$r_1 = \frac{\langle c, x\rangle}{\langle b, y\rangle} \leqslant \epsilon$
$r_0 = \frac{\langle c, x\rangle}{\langle c, x^*\rangle} \leqslant \epsilon$
$\langle b, y\rangle$
$\langle c, x^*\rangle$
$\langle c, x\rangle$
dual cost of solution y
cost of optimal integral solution x*
primal cost of integral solution x

Theorem 1. *If* $\mathbf{x}$ *and* $\mathbf{y}$ *are integral-primal and dual feasible solutions satisfying:*

$$\langle \mathbf{c}, \mathbf{x}\rangle \leqslant \epsilon \langle \mathbf{b}, \mathbf{y}\rangle$$

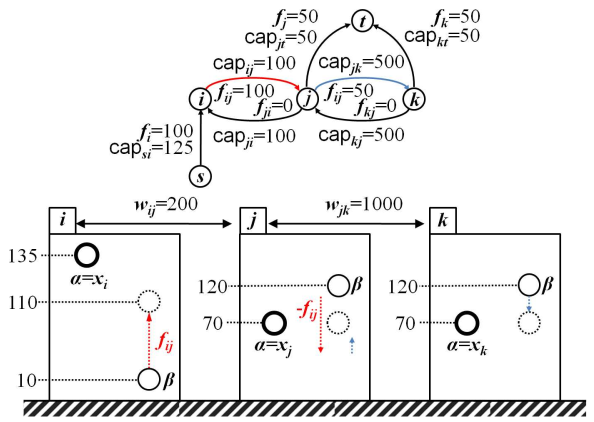
for $\epsilon \geqslant 1$*, then* $\mathbf{x}$ *is an* ϵ**-approximation** *to the optimal integral solution* $\mathbf{x}^*$*, that is*

$$\langle \mathbf{c}, \mathbf{x}^*\rangle \leqslant \langle \mathbf{c}, \mathbf{x}\rangle \leqslant \epsilon \langle \mathbf{c}, \mathbf{x}^*\rangle .$$

FastPD

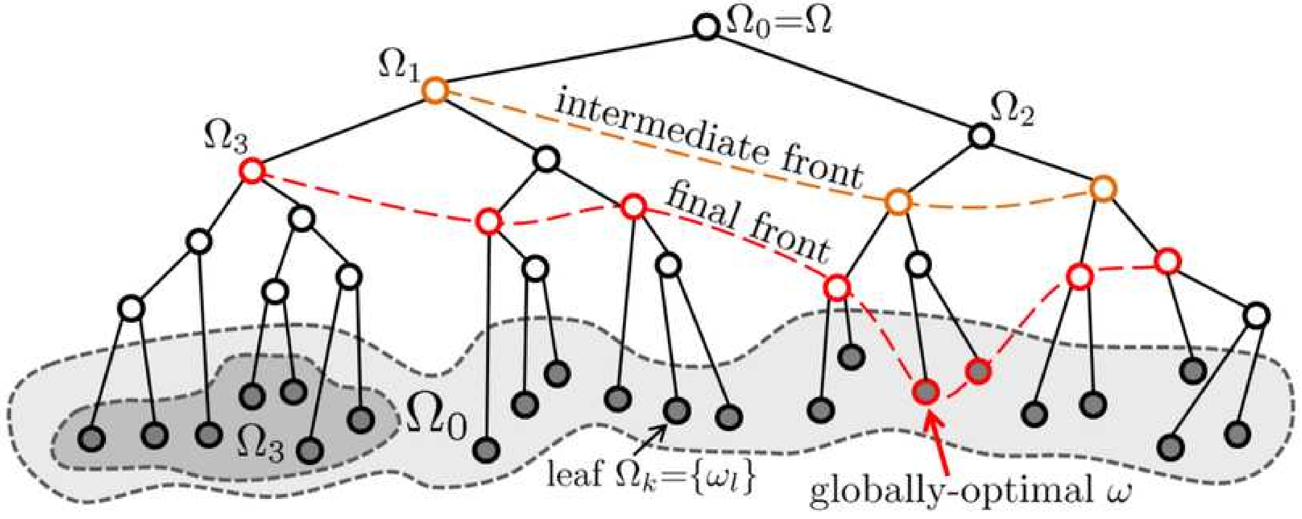
Dual variables update: Given the current active labels, any non-active label is raised, until it either reaches the active label, or attains the maximum raise allowed by the upper bound.

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 $G' = (\mathcal{V} \cup \{s,t\}, \mathcal{E}', c, s, t)$.

Best-first branch-and-bound optimization

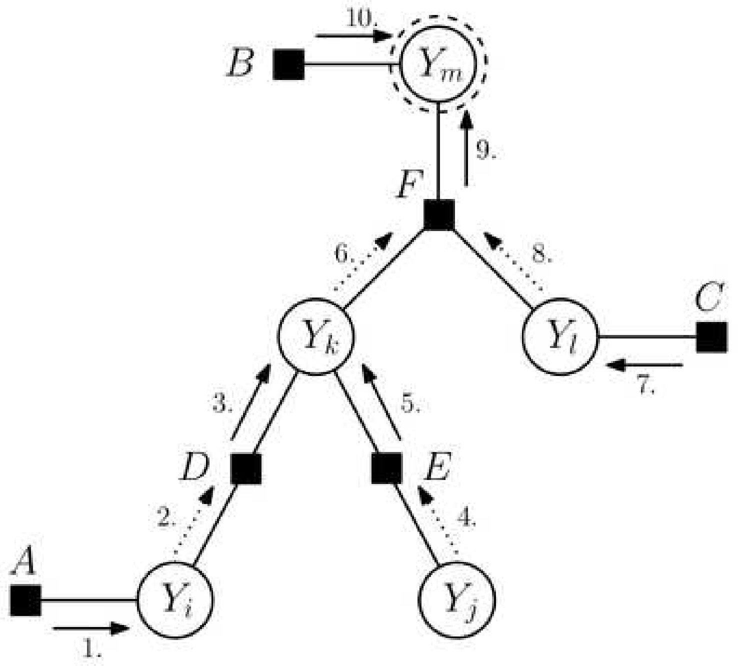


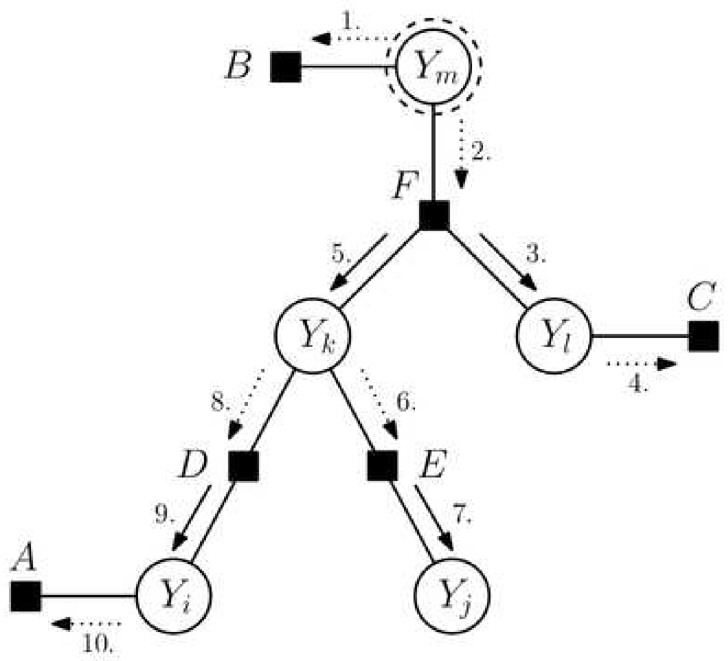
At each step the *active node* with the **smallest** *lower bound* is removed from the *active front*, while two of its *children* are added to the *active front* (due to *monotonicity property* they have higher or equal lower bounds).

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**.

Belief propagation

For tree-structured factor graphs there always exist at least one message that can be computed initially, hence all the dependencies can be resolved.

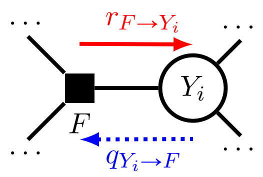




1. Select one variable node as root of the tree (e.g., Y_m)
2. Compute leaf-to-root messages (e.g., by applying depth-first-search)
3. Compute root-to-leaf messages (reverse order as before)

Messages

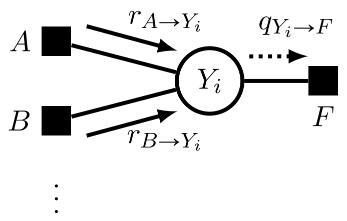
Message: pair of vectors at each factor graph edge $(i, F) \in \mathcal{E}$.



1. **Variable-to-factor** message $q_{Y_i\to F} \in \mathbb{R}^{\mathcal{Y}_i}$ is given by

$$q_{Y_i\to F}(y_i) = \prod_{F'\in M(i)\setminus\{F\}} r_{F'\to Y_i}(y_i)\ ,$$

where $M(i) = \{F \in \mathcal{F} : (i, F) \in \mathcal{E}\}$ denotes the set of factors adjacent to Y_i.



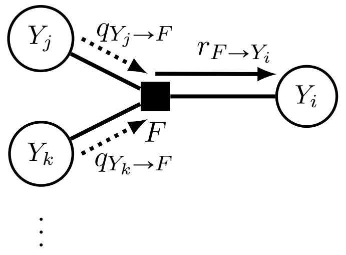
2. **Factor-to-variable** message: $r_{F\to Y_i} \in \mathbb{R}^{\mathcal{Y}_i}$.

Factor-to-variable message

2. **Factor-to-variable** message $r_{F\to Y_i} \in \mathbb{R}^{\mathcal{Y}_i}$ is given by

$$r_{F\to Y_i}(y_i) = \sum_{\substack{\mathbf{y}'_F \in \mathcal{Y}_F,\\ y'_i = y_i}} \left(\exp(-E_F(\mathbf{y}'_F)) \prod_{l \in N(F)\setminus\{i\}} q_{Y_l \to F}(y'_l) \right),$$

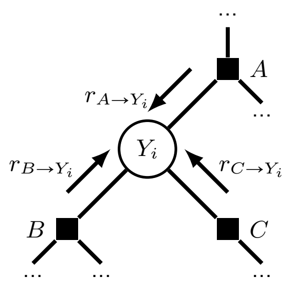
where $N(F) = \{i \in V : (i,F) \in \mathcal{E}\}$ denotes the set of variables adjacent to F.



Inference result

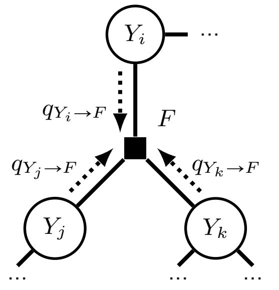
Partition function is evaluated at the (root) node i

$$Z = \sum_{y_i \in \mathcal{Y}_i} \prod_{F \in M(i)} r_{F \to Y_i}(y_i) \; .$$



The **marginal distribution** for each factor can be computed as

$$\mu_F(\mathbf{y}_F) = \frac{1}{Z} \exp(-E_F(\mathbf{y}_F)) \prod_{i \in N(F)} q_{Y_i \to F}(y_i) \; .$$



Sum-product and Max-sum comparison *

- Sum-product algorithm

$$q_{Y_i \to F}(y_i) = \prod_{F' \in M(i) \setminus \{F\}} r_{F' \to Y_i}(y_i)$$

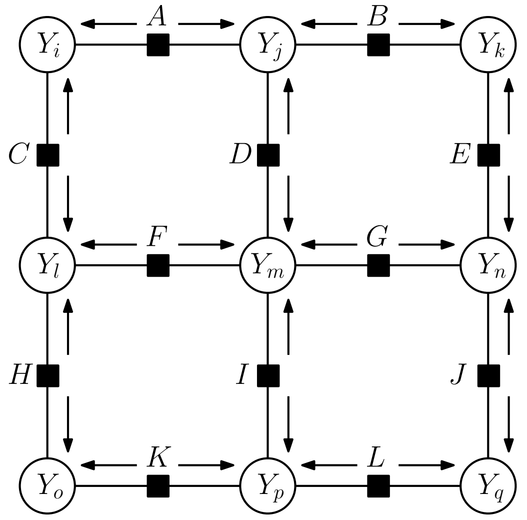
$$r_{F \to Y_i}(y_i) = \sum_{\substack{y'_F \in \mathcal{Y}_F, \\ y'_i = y_i}} \left(\exp(-E_F(y'_F)) \prod_{l \in N(F) \setminus \{i\}} q_{Y_l \to F}(y'_l) \right)$$

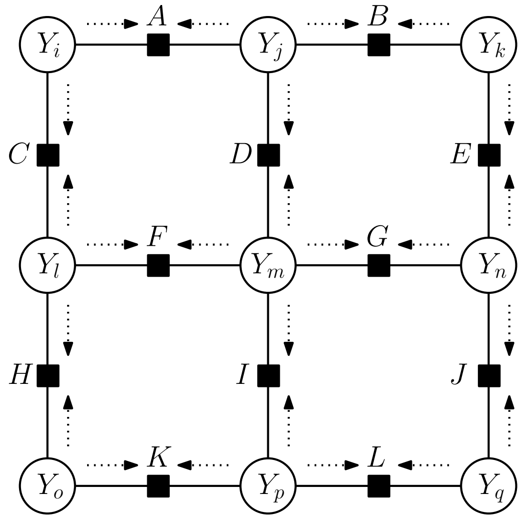
- Max-sum algorithm

$$q_{Y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to Y_i}(y_i)$$

$$r_{F \to Y_i}(y_i) = \max_{\substack{y'_F \in \mathcal{Y}_F, \\ y'_i = y_i}} \left(-E_F(y'_F) + \sum_{l \in N(F) \setminus \{i\}} q_{Y_l \to F}(y'_l) \right)$$

Loopy belief propagation





Loopy belief propagation is very popular, but has some problems:

- It might not converge (e.g., it can oscillate).
- Even if it does, the computed probabilities are only *approximate*.
- If there is a single cycle only in the graph, then it converges.

Mean field approximation

For general (discrete) factor graph models, performing *probabilistic inference* is hard. Assume we are given an **intractable** distribution $p(\mathbf{y} \mid \mathbf{x})$. We consider an **approximate distribution** $q(\mathbf{y})$, which is tractable, for $p(\mathbf{y} \mid \mathbf{x})$.

One way of finding the best approximating distribution is to pose it as an **optimization problem** over probability distributions: given a distribution $p(\mathbf{y} \mid \mathbf{x})$ and a family Q of *tractable distributions* $q \in Q$ on $\mathcal{Y}$, we want to solve

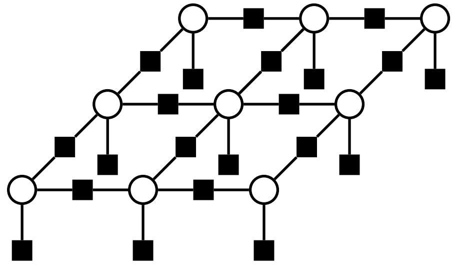
$$\begin{aligned} q^* &\in \operatorname*{argmin}_{q \in Q} D_{\mathrm{KL}}(q(\mathbf{y}) \| p(\mathbf{y} \mid \mathbf{x})) \\ &= \operatorname*{argmin}_{q \in Q} \Big\{ \underbrace{\sum_{\mathbf{y} \in \mathcal{Y}} q(\mathbf{y}) \log q(\mathbf{y})}_{-H(q)} - \sum_{\mathbf{y} \in \mathcal{Y}} q(y) \log p(\mathbf{y} \mid \mathbf{x}) \Big\} . \end{aligned}$$

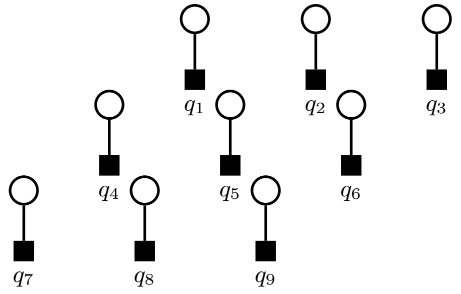
Naïve mean field

Take a set Q as the set of all distributions in the form:

$$q(\mathbf{y}) = \prod_{i \in \mathcal{V}} q_i(y_i) .$$

For example, in case of the following factor graph:


Original factor graph

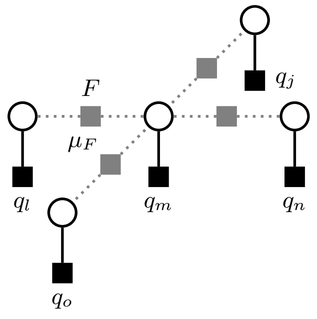


Mean field approximation

Naïve mean field: Optimization

Block coordinate ascent:

We hold all variables fixed except for a single block q_m, then we obtain a tractable concave maximization problem
→ closed-form update for each q_m.



The update equation for the `Naïve mean field method` is given by

$$q_i^*(y_i) = \frac{1}{Z_i(\mathbf{x}_F)} \exp\left(-\sum_{F \in M(i)} \sum_{\substack{\mathbf{y}'_F \in \mathcal{Y}_F, \\ y'_i = y_i}} \Big(\prod_{j \in N(F) \setminus \{i\}} q_j(y_j) \Big) E_F(\mathbf{y}_F; \mathbf{x}_F) \right) .$$

Gibbs sampling

Each step of the *Gibbs sampling* procedure involves replacing the value of one of the variables y_i by a value drawn from the distribution of that variable conditioned on the values of the remaining variables, that is

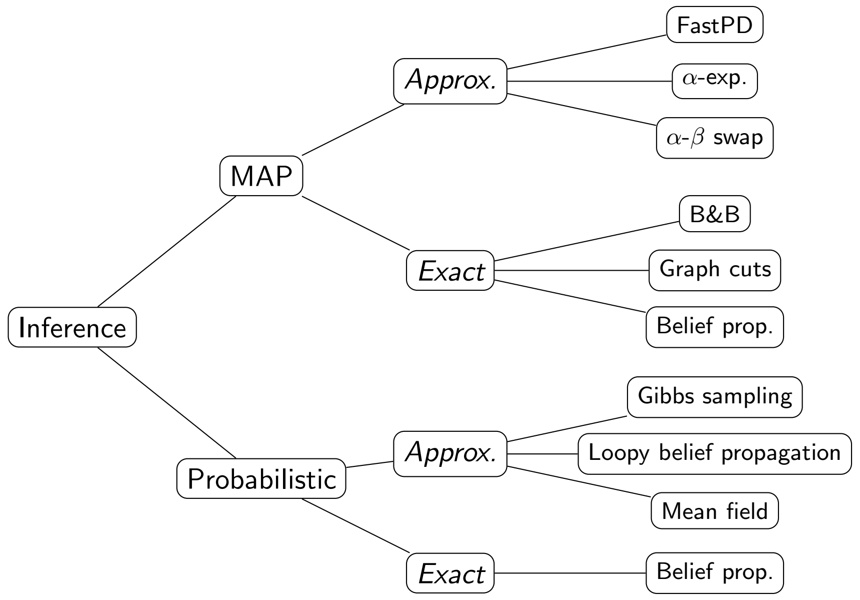
$$y_i^{(t+1)} \leftarrow y'_i \sim p(y_i \mid \mathbf{y}_{\setminus i}^{(t)}, \mathbf{x}) .$$

This requires only the unnormalized distribution $\tilde{p}$ and the normalization over a single variable:

$$p(y_i \mid \mathbf{y}_{\setminus i}^{(t)}, \mathbf{x}) = \frac{\prod_{F \in M(i)} \exp(-E_F(y_i, \mathbf{y}_{N(F)\setminus\{i\}}^{(t)}; \mathbf{x}_F))}{\sum_{y_i \in \mathcal{Y}_i} \prod_{F \in M(i)} \exp(-E_F(y_i, \mathbf{y}_{N(F)\setminus\{i\}}^{(t)}; \mathbf{x}_F))} .$$

The basic idea is that while sampling from $p(\mathbf{y} \mid \mathbf{x})$ is hard, sampling from the conditional distributions $p(y_i \mid \mathbf{y}_{\setminus i}, \mathbf{x})$ can be performed efficiently.

Overview


Inference
MAP
Approx.
FastPD
α-exp.
α-β swap
Exact
B&B
Graph cuts
Belief prop.
Probabilistic
Approx.
Gibbs sampling
Loopy belief propagation
Mean field
Exact
Belief prop.

Parameter learning

Parameter learning

Learning graphical models (from training data) is a way to find among a large class of possible models a single one that is *best* in some sense for the task at hand.

We assume a fixed underlying graphical model with **parameterized conditional probability distribution**

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \exp(-E(\mathbf{y}; \mathbf{x}, \mathbf{w})) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \exp(-\langle \mathbf{w}, \varphi(\mathbf{x}, \mathbf{y}) \rangle) ,$$

where $Z(\mathbf{x}, \mathbf{w}) = \sum_{\mathbf{y} \in \mathcal{Y}} \exp(-\langle \mathbf{w}, \varphi(\mathbf{x}, \mathbf{y}) \rangle)$. The only unknown quantity is the *parameter vector* $\mathbf{w}$, on which the energy $E(\mathbf{y}; \mathbf{x}, \mathbf{w})$ depends **linearly**.

Probabilistic parameter learning

We aim at identifying a weight vector $\mathbf{w}^*$ that makes $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$ as close to the **true conditional label distribution** $d(\mathbf{y} \mid \mathbf{x})$ as possible. The label distribution itself is unknown to us, but we have an *i.i.d.* sample set $\mathcal{D} = \{(\mathbf{x}^n, \mathbf{y}^n)\}_{n=1,\dots,N}$ from $d(\mathbf{x}, \mathbf{y})$ that we can use for learning.

We measure the dissimilarity by making use of **Kullback-Leibler (KL) divergence**:

$$\mathsf{KL}(d \| p) = \sum_{\mathbf{y} \in \mathcal{Y}} d(\mathbf{y} \mid \mathbf{x}) \log \frac{d(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})} \; .$$

We obtain a **total measure** of how much p differs from d by their **expected dissimilarity** over all $\mathbf{x} \in \mathcal{X}$:

$$\mathsf{KL}_{\mathsf{tot}}(d \| p) = \sum_{\mathbf{x} \in \mathcal{X}} d(\mathbf{x}) \sum_{\mathbf{y} \in \mathcal{Y}} d(\mathbf{y} \mid \mathbf{x}) \log \frac{d(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})} \; .$$

Regularized maximum conditional likelihood training

Let $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \exp(-\langle \mathbf{w}, \varphi(\mathbf{x}, \mathbf{y}) \rangle)$ be a *probability distribution parameterized by* $\mathbf{w} \in \mathbb{R}^D$, and let $\mathcal{D} = \{(\mathbf{x}^n, \mathbf{y}^n)\}_{n=1,\dots,N}$ be a set of *i.i.d. training examples*. For any $\lambda > 0$, **regularized maximum conditional likelihood** training chooses the parameter as

$$\begin{aligned} \mathbf{w}^* &\in \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} L(\mathbf{w}) \\ &= \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \lambda \|\mathbf{w}\|^2 + \sum_{n=1}^{N} \langle \mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}^n) \rangle + \sum_{n=1}^{N} \log Z(\mathbf{x}^n, \mathbf{w}) \; . \end{aligned}$$

Stochastic gradient descent

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = 2\lambda\mathbf{w} + \sum_{n=1}^{N} \left(\varphi(\mathbf{x}^n, \mathbf{y}^n) - \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x}^n,\mathbf{w})}[\varphi(\mathbf{x}^n, \mathbf{y})]\right) \ .$$

If the training set $\mathcal{D}$ is too large, one may *randomly select* only **one** sample and calculate the gradient

$$\tilde{\nabla}_{\mathbf{w}}^{(\mathbf{x}^n,\mathbf{y}^n)} L(\mathbf{w}) = 2\lambda\mathbf{w} + \varphi(\mathbf{x}^n, \mathbf{y}^n) - \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x}^n,\mathbf{w})}[\varphi(\mathbf{x}^n, \mathbf{y})] \ .$$

Note that line search is not possible, therefore, we need for an extra parameter, referred to as *step-size* η_t for each iteration.

Using of the output structure

Assume the set of factors $\mathcal{F}$ in a graphical model representation, such that $\varphi(\mathbf{x}, \mathbf{y})$ decomposes as $\varphi(\mathbf{x}, \mathbf{y}) = [\varphi_F(\mathbf{x}_F, \mathbf{y}_F)]_{F\in\mathcal{F}}$. Thus

$$\mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x},\mathbf{w})}[\varphi(\mathbf{x}, \mathbf{y})] = [\mathbb{E}_{\mathbf{y}_F\sim p(\mathbf{y}_F|\mathbf{x}_F,\mathbf{w})}[\varphi_F(\mathbf{x}_F, \mathbf{y}_F)]]_{F\in\mathcal{F}} \ ,$$

where

$$\mathbb{E}_{\mathbf{y}_F\sim p(\mathbf{y}_F|\mathbf{x}_F,\mathbf{w}_F)}[\varphi_F(\mathbf{x}_F, \mathbf{y}_F)] = \sum_{\mathbf{y}_F\in\mathcal{Y}_F} p(\mathbf{y}_F \mid \mathbf{x}_F, \mathbf{w}_F)\varphi_F(\mathbf{x}_F, \mathbf{y}_F) \, .$$

Factor marginals $\mu_F = p(\mathbf{y}_F \mid \mathbf{x}_F, \mathbf{w}_F)$ are generally (much) easier to calculate than the complete conditional distribution $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$.

Two-stage learning

The idea here is to split learning of energy functions into two steps:

1. learning of unary energies via *classifiers*, and
2. learning of their importance and the weighting factors of pairwise (and higher-order) energies.

$$E(\mathbf{y};\mathbf{x}) = \sum_{i\in\mathcal{V}} w_i E_i(y_i;x_i) + \sum_{(i,j)\in\mathcal{E}'} w_{ij} E_{ij}(y_i,y_j)\ .$$

As an advantage, it results in a *faster* learning method. However, if local classifiers for E_i perform badly, then CRF learning **cannot** fix it.

Piecewise learning

Assume a set of factors $\mathcal{F}$ in a *factor graph model*, such that the vector $\varphi(\mathbf{x},\mathbf{y}) = [\varphi_F(\mathbf{x}_F,\mathbf{y}_F)]_{F\in\mathcal{F}}$.

We now **approximate** $p(\mathbf{y}\mid\mathbf{x},\mathbf{w})$ by a distribution that is a product over the factors:

$$p_{\mathsf{PW}}(\mathbf{y}\mid\mathbf{x},\mathbf{w}) := \prod_{F\in\mathcal{F}} p_F(\mathbf{y}_F\mid\mathbf{x}_F,\mathbf{w}_F) = \prod_{F\in\mathcal{F}} \frac{\exp(-\langle\mathbf{w}_F,\varphi_F(\mathbf{x}_F,\mathbf{y}_F)\rangle)}{Z_F(\mathbf{x}_F,\mathbf{w}_F)}\ .$$

Piecewise training chooses the parameters $\mathbf{w}^* = [\mathbf{w}_F^*]_{F\in\mathcal{F}}$ as

$$\mathbf{w}_F^* \in \operatorname*{argmin}_{\mathbf{w}_F\in\mathbb{R}} \lambda\|\mathbf{w}_F\|^2 + \sum_{n=1}^{N}\langle\mathbf{w}_F,\varphi_F(\mathbf{x}_F^n,\mathbf{y}_F^n)\rangle + \sum_{n=1}^{N}\log Z_F(\mathbf{x}_F^n,\mathbf{w}_F)\ .$$

One can perform gradient-based training for each factor as long as the individual factors remain small.

Loss-minimizing parameter learning

Let $\mathcal{D} = \{(\mathbf{x}^1, \mathbf{y}^1), \dots, (\mathbf{x}^N, \mathbf{y}^N)\} \subseteq \mathcal{X} \times \mathcal{Y}$ be *i.i.d.* samples from the (unknown) *true data distribution* $d(\mathbf{x}, \mathbf{y})$ and $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ be a *loss function*. The task is to find a weight vector $\mathbf{w}$ that leads to **minimal expected loss**

$$\mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim d(\mathbf{x},\mathbf{y})}[\Delta(\mathbf{y}, f(\mathbf{x}))]$$

for a *prediction function* $f(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} g(\mathbf{x}, \mathbf{y}; \mathbf{w})$, where $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is an **auxiliary function** that is parameterized by $\mathbf{w} \in \mathbb{R}^D$.

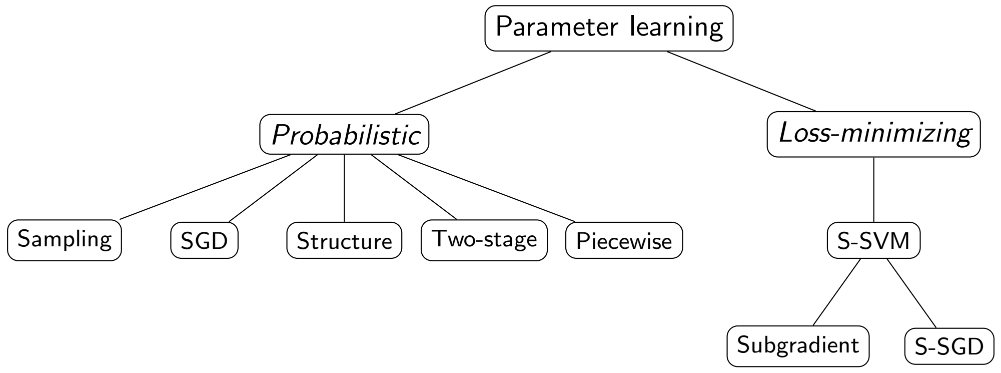
Let $g(\mathbf{x}, \mathbf{y}; \mathbf{w}) = -\langle \mathbf{w}, \varphi(\mathbf{x}, \mathbf{y}) \rangle$ be an *auxiliary function* parameterized by $\mathbf{w} \in \mathbb{R}^D$. For any $C > 0$, **structured support vector machine** (S-SVM) training chooses the parameter

$$\mathbf{w}^* \in \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N} \sum_{n=1}^{N} \ell(\mathbf{x}^n, \mathbf{y}^n, \mathbf{w})$$

with $\ell(\mathbf{x}^n, \mathbf{y}^n, \mathbf{w}) = \max_{\mathbf{y} \in \mathcal{Y}}(\Delta(\mathbf{y}^n, \mathbf{y}) - \langle \mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}) \rangle + \langle \mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}^n) \rangle)$.

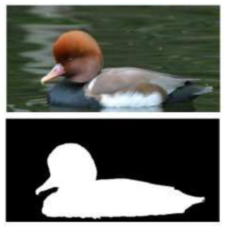
S-SVM learning ends up a **convex optimization** problem, but it is **non-differentiable**. Furthermore it requires repeated *argmax prediction*.

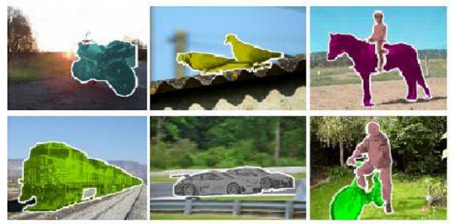
Overview

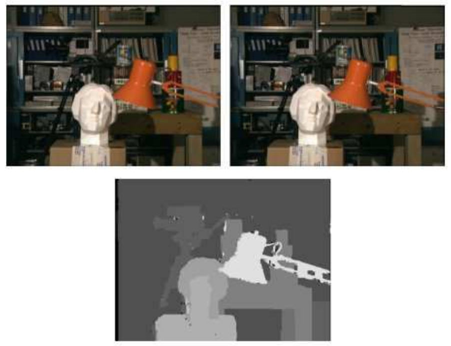


Computer vision

Overview



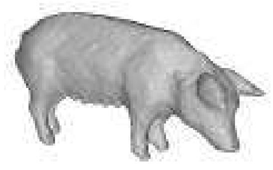




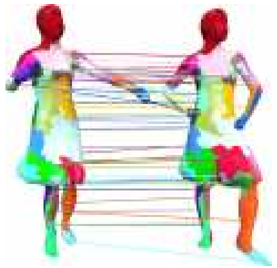
Announcement: Computer Vision Group

Inquiries for Bachelor and Master projects are always welcome!

We currently work on the following `research topics`:


3D reconstruction

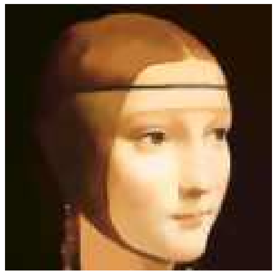
Optical flow

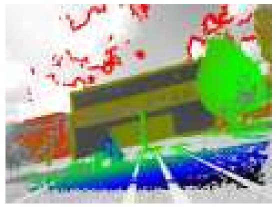

Shape analysis

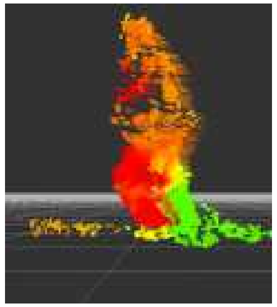
Robot vision

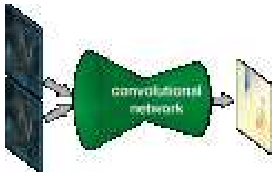
RGB-D vision

Image segmentation


Convex relaxation


Visual SLAM


Scene flow



Deep learning

Please complete the form: `https://vision.in.tum.de/application`