Computer Vision II

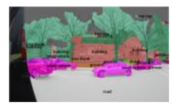
- Last time: Computer vision tasks as massive search problems

$$f(x) = y^* = argmax_{y \in \mathcal{Y}}\ g(x, y)$$

- Detection: $f: \mathcal{X} \rightarrow \mathcal{Y}$ $\mathcal{Y}$ = all possible positions (and scales) of object

- Foreground/background segmentation: $f: \mathcal{X} \rightarrow \mathcal{Y}$ $\mathcal{Y}$ = all possible 0/1 labelings of image $\{0,1\}^n$

- Labeling: $f: \mathcal{X} \rightarrow \mathcal{Y}$ $\mathcal{Y}$ = all possible labelings of image $\{1, \ldots, L\}^n$



- Pose estimation: $f: \mathcal{X} \rightarrow \mathcal{Y}$ $\mathcal{Y}$ = all possible poses (u, v, θ, s) of image $\{1, \ldots, P\}^K$

$$\boxed{f(x) = y^* = argmax_{y \in \mathcal{Y}}\, g(x,y)}$$

- Detection: $f: \mathcal{X} \rightarrow \mathcal{Y}$ $\mathcal{Y}$ = all possible positions (and scales) of object

$$\boxed{g(x,y) = w.\varphi(x,y)}$$

- Foreground/background segmentation: $f: \mathcal{X} \rightarrow \mathcal{Y}$ $\mathcal{Y}$ = all possible 0/1 labelings of image $\{0,1\}^n$
- Labeling: $f: \mathcal{X} \rightarrow \mathcal{Y}$ $\mathcal{Y}$ = all possible labelings of image $\{1, \ldots, L\}^n$
- Pose estimation: $f: \mathcal{X} \rightarrow \mathcal{Y}$ $\mathcal{Y}$ = all possible poses (u, v, θ, s) of image $\{1, \ldots, P\}^K$

Energy models:

$$g(x,y) = \sum_{i=1}^{n} g_i(x, y_i) + \sum_{i,j \in \mathcal{N}(i)} g_{i,j}(y_i, y_j, x)$$

Probabilistic models:

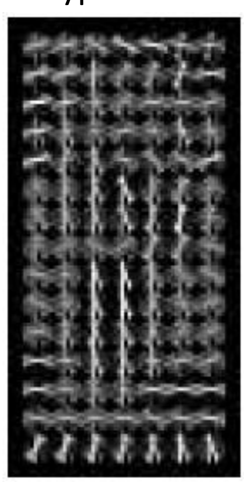
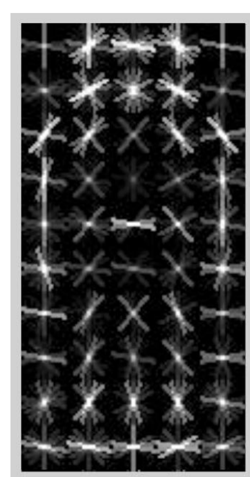
$$p(y|x) = \frac{1}{Z}\exp\big(g(x,y)\big) = \frac{1}{Z}\prod \psi_i(x, y_i) \prod_{i,j \text{ linked}} \psi_{ij}(y_i, y_j)$$

Reminder of key results

- Exact algorithms on tree-structured graphs
  - Message passing
  - Max-product: compute y^*
  - Sum-product: estimate marginals $p(y_i|x)$

- Today:
  - Details of max/sum for tree-structured models
    - Detection with parts
    - Pose estimation
  - Efficient search for detection (in some special cases)
- Next:
  - Details of general cases for segmentation and labeling

Richer description is needed to capture the variation in appearance of typical visual classes

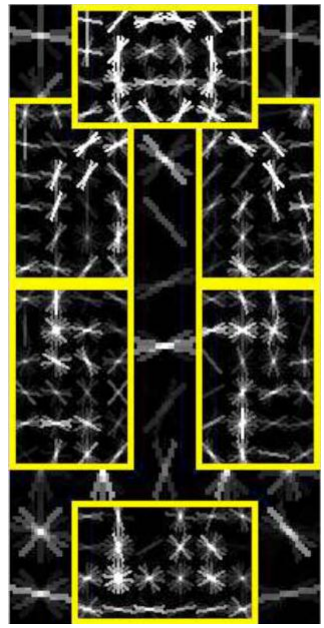
$$g(x, y) = \quad w \quad \cdot \quad \varphi(x, y)$$

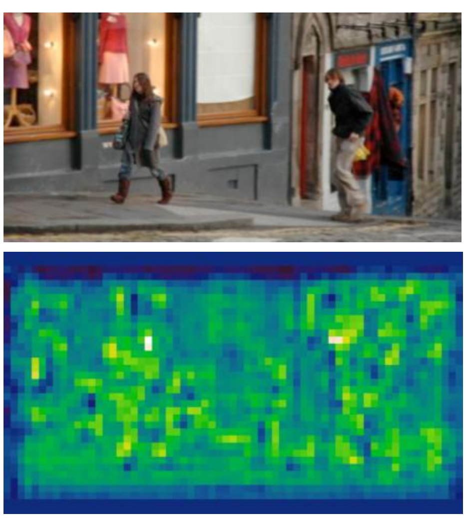
Representation by parts:

$$y = y_1, \ldots, y_K$$

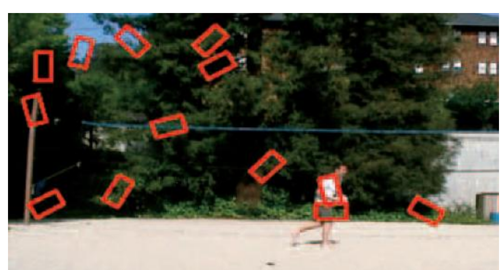
Possible (bad) model:
Find each part independently

$$y_i^* = \text{argmax}_{y_i} g_i(x, y_i)$$





Example from P. Felsenzwalb

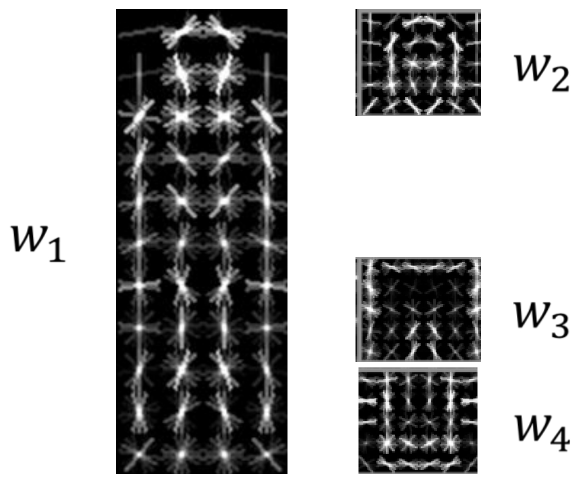


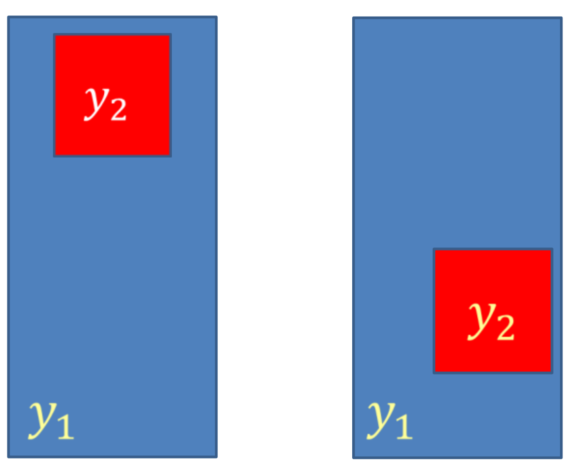
Example from D. Ramanan

- For each individual part:

$$g_i(x, y_i) = w_i \cdot \varphi(x, y_i)$$

Feature at y_i (e.g., HoG)



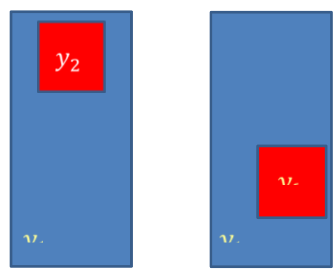


Likely Unlikely

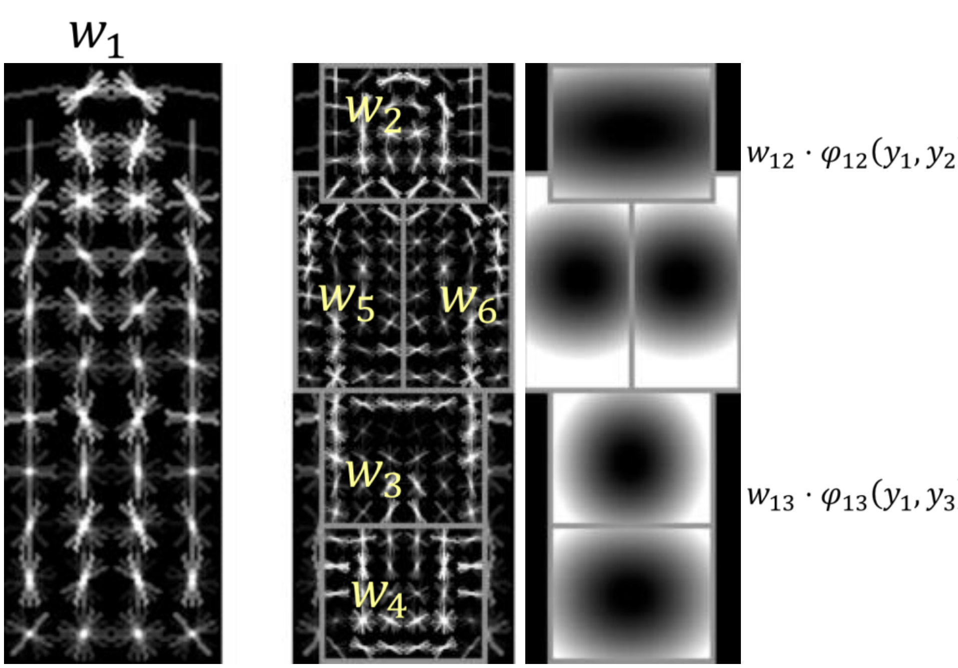
$$g_{ij}(y_i, y_j) = w_{ij} \cdot \varphi_{ij}(y_i, y_j)$$

Feature vector describing the location of y_i with respect to y_j

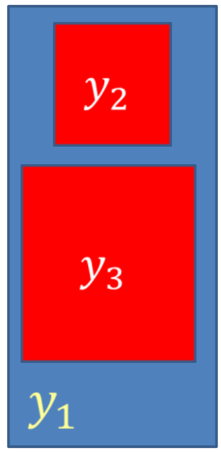
For example: $\varphi_{ij} = \begin{bmatrix} -(u_1 - u_2)^2 \\ -(v_1 - v_2)^2 \end{bmatrix}$



Likely Unlikely



$$
\begin{aligned}
g(x, y) \\
&= w_1^T \cdot \varphi_1(x, y_1) + w_2^T \cdot \varphi_2(x, y_2) + w_3^T \cdot \varphi_3(x, y_3) \\
&+ w_{12} \cdot \varphi_{12}(x, y_{12}) + w_{13} \cdot \varphi(x, y_{13})
\end{aligned}
$$



$$
\begin{aligned}
g(x, y) \\
&= w_1^T \varphi_1(x, y_1) + w_2^T \varphi_2(x, y_2) + w_3^T \varphi_3(x, y_3) \\
&+ w_{12}^T \varphi_{12}(y_1, y_2) + w_{13}^T \varphi_{13}(y_2, y_3)
\end{aligned}
$$

$$
\max_y g(x, y) = \max_{y_1} \max_{y_2} \max_{y_3}
$$

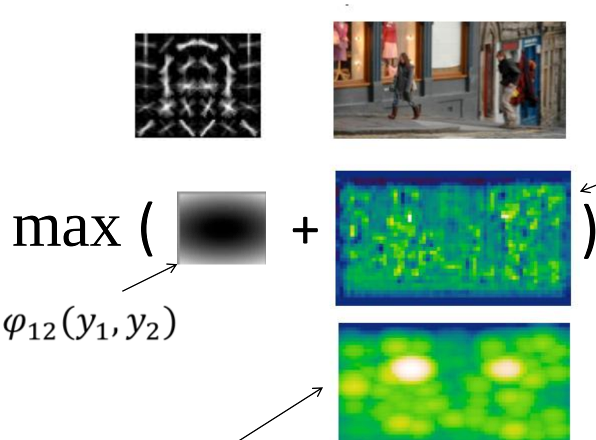
$$
w_1^T \varphi_1(x, y_1) + w_2^T \varphi_2(x, y_2) + w_3^T \varphi_3(x, y_3)
$$

$$
+ w_{12}^T \varphi_{12}(y_1, y_2) + w_{13}^T \varphi_{13}(y_1, y_3)
$$

Usual max sum trick: max(a+b, a+c) = a + max(b,c)

$$\max_{y_1}(w_1^T\varphi_1(x,y_1) + \max_{y_2}(w_2^T\,\varphi_2(x,y_2) + w_{12}^T\varphi_{12}(y_1,y_2)) + \max_{y_3}(w_3^T\varphi_3(x,y_3) + w_{13}^T\varphi_{13}(y_1,y_3)))$$

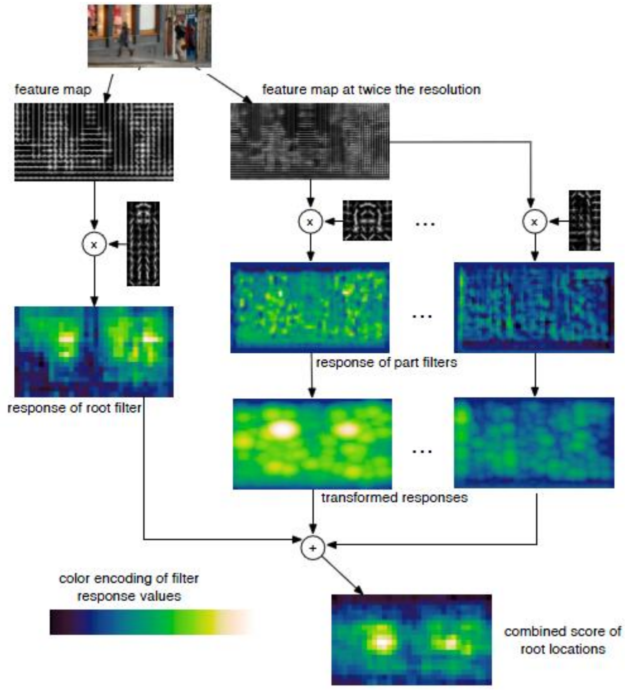
$$\max_{y_1}(w_1^T\varphi_1(x,y_1) + \max_{y_2}(w_2^T\,\varphi_2(x,y_2) + w_{12}^T\varphi_{12}(y_1,y_2)) + \max_{y_3}(w_3^T\varphi_3(x,y_3) + w_{13}^T\varphi_{13}(y_1,y_3)))$$



$w_2^T\varphi_2(x,y_2)$

$w_{12}^T\varphi_{12}(y_1,y_2)$

$w_2^T\varphi_2(x,y_2) + w_{12}^T\varphi_{12}(y_1,y_2)$



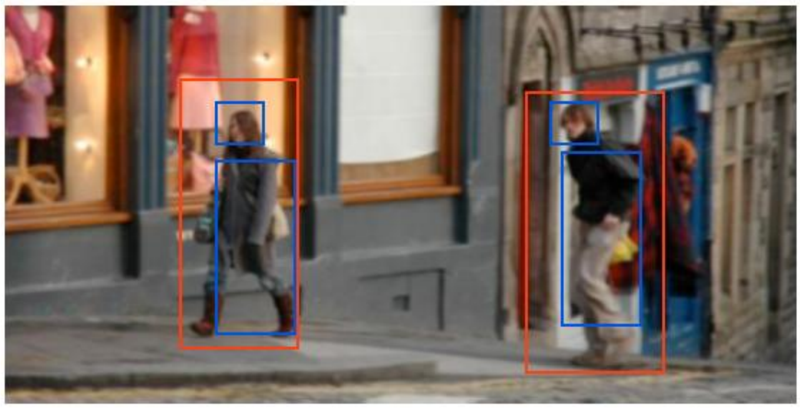
General case

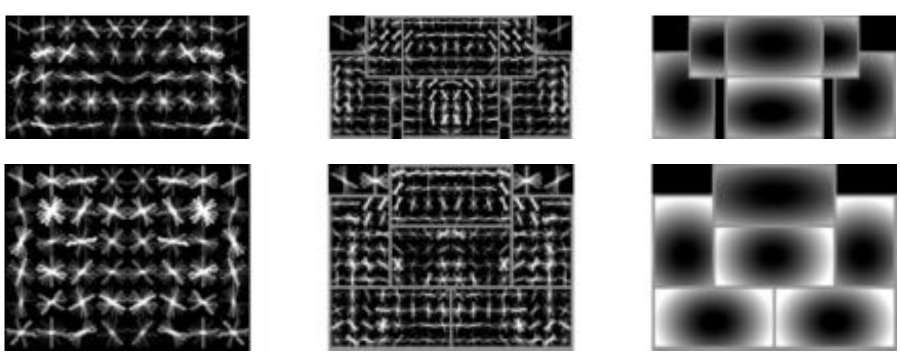
- Message passing (DP):

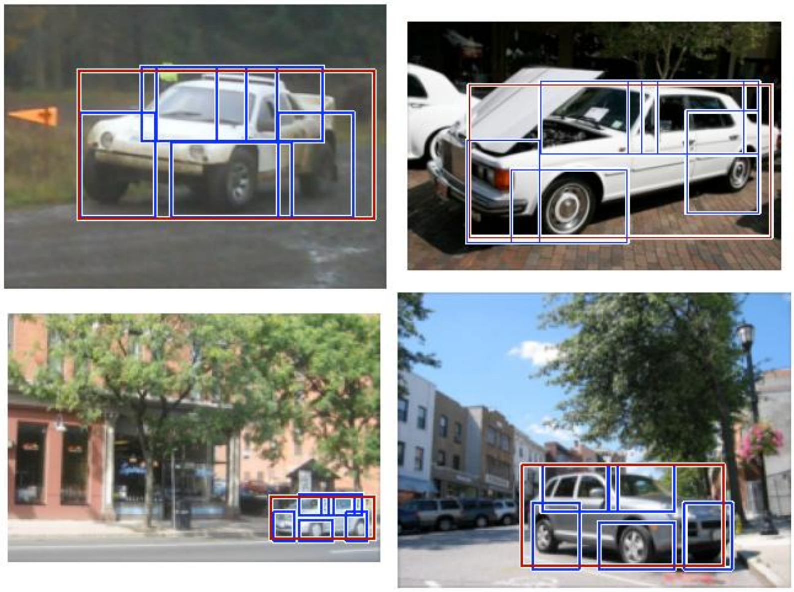
$$\text{score}(y_j)$$

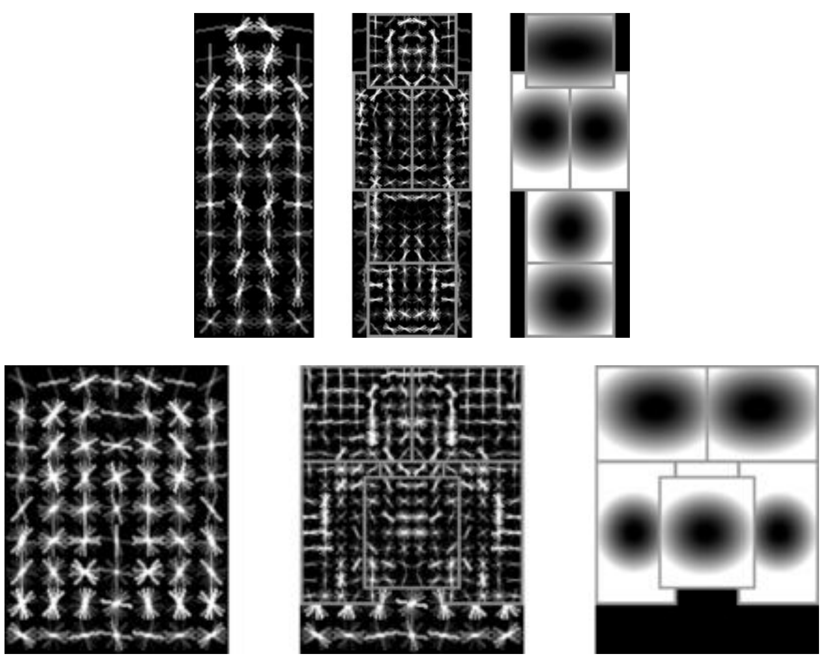
$$= w_j^T \varphi_j(x, y_j) + \sum_{k \text{ descendant}(j)} m_k(y_j)$$

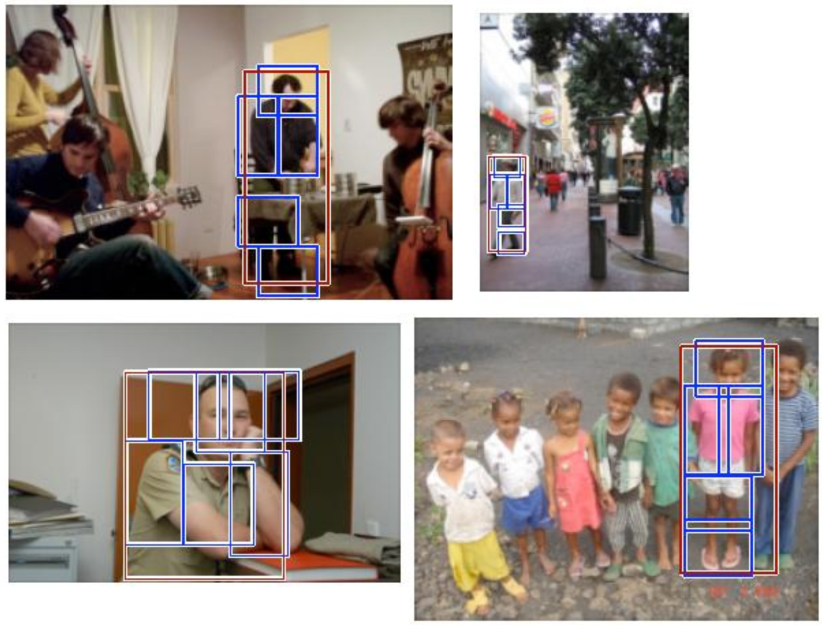
$$m_a(y_b) = \max_{y_a} \text{score}(y_a) + w_{ab}^T \varphi_{ab}(y_a, y_b)$$

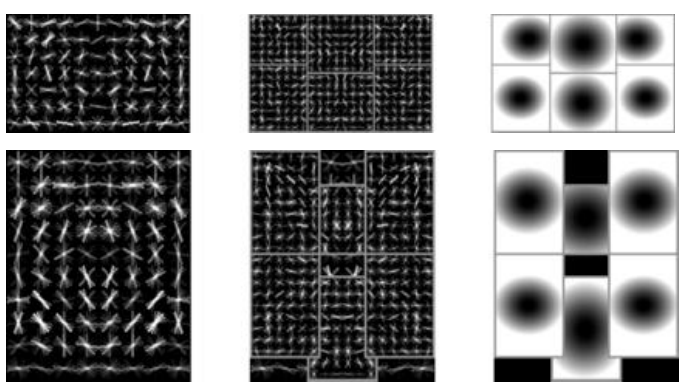


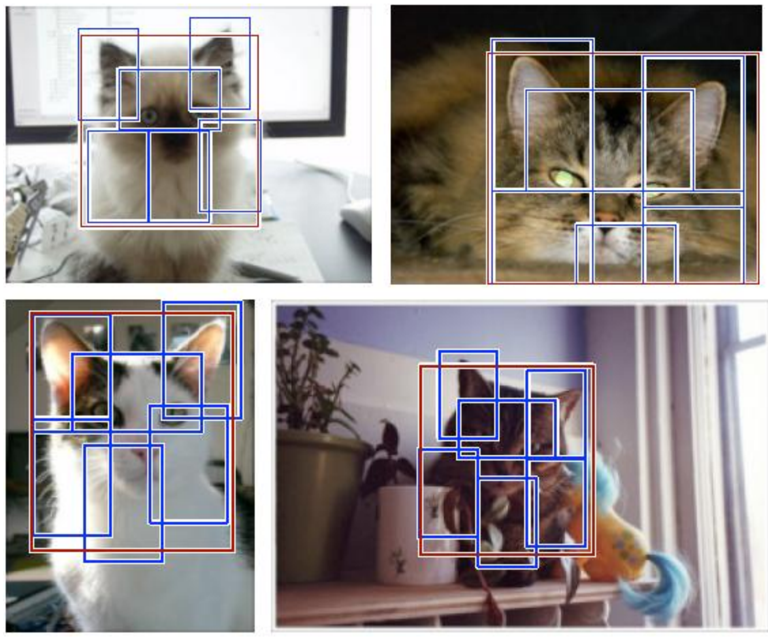


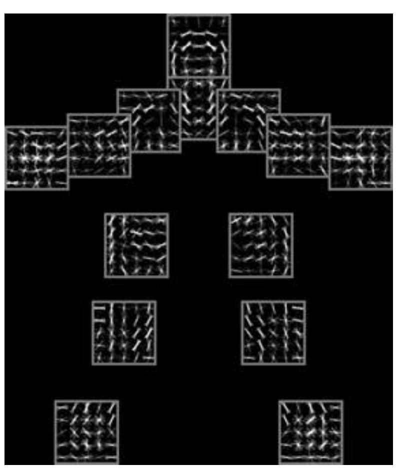


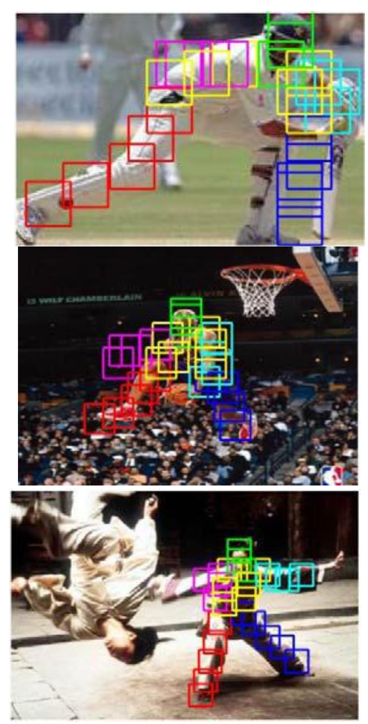












Estimating the marginals

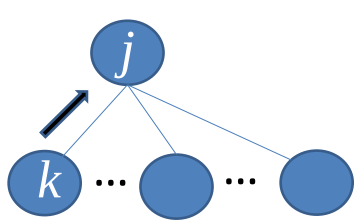
- $p(y|x) = \frac{1}{Z}\exp\big(g(x,y)\big) = \frac{1}{Z}\prod \psi_i(x, y_i) \prod_{i,j \text{ linked}} \psi_{ij}(y_i, y_j)$

Estimating the marginals

- Propagate partial sums from leaves

$$P(y_j | y_i, x) = P(y_j | y_i) m_j(y_j)$$

$$m_j(y_j) \propto e^{w_j^T \varphi_j(x, y_j)} \prod_{k \text{ child of } j} \sum_{y_k} P(y_k | y_j, x)$$

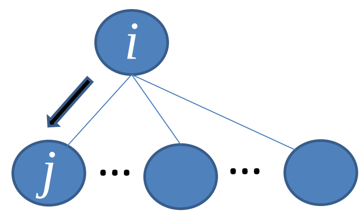


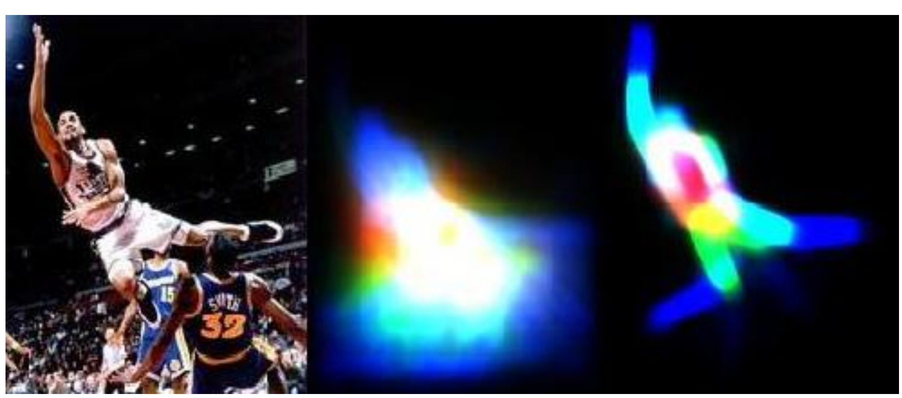
Estimating the marginals

- Propagate partial sums from root

$$P(y_j, y_i|x) = P(y_j|y_i, x)P(y_i|x)$$

$$P(y_j|x) = \sum_{y_i} P(y_j, y_i|x)$$





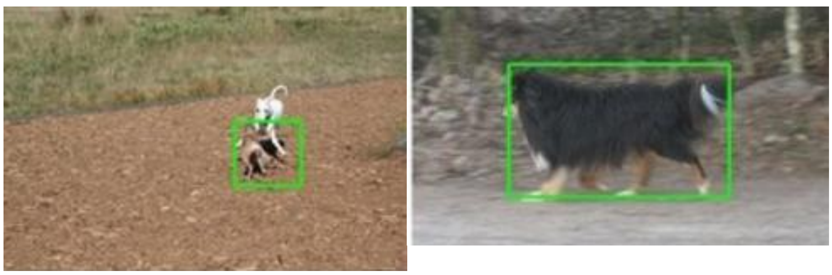
Example from Deva Ramanan

Parenthesis: Efficiency issues in search for a detection in an image

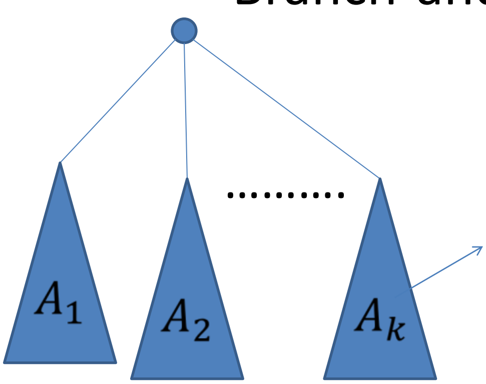
- Detection: $f\colon \mathcal{X} \rightarrow \mathcal{Y}$ $\mathcal{Y}$ = all possible positions (and scales) of object

$$g(x, y) = w.\varphi(x, y)$$

- Need to evaluate all possible boxes = all possible positions and sizes = N^4
- Is it possible to do this more efficiently?
- Yes, for *some special cases*



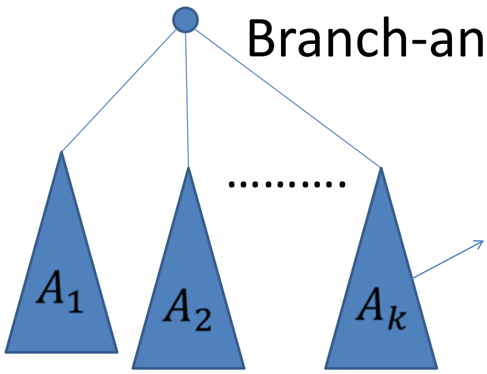
Branch-and-bound



$$\underline{g_k} < g_k(a) < \overline{g_k} \quad a \in A_k$$

- if $\overline{g_k} < \underline{g_l}$ for all l then there is no point in exploring A_k

Branch-and-bound

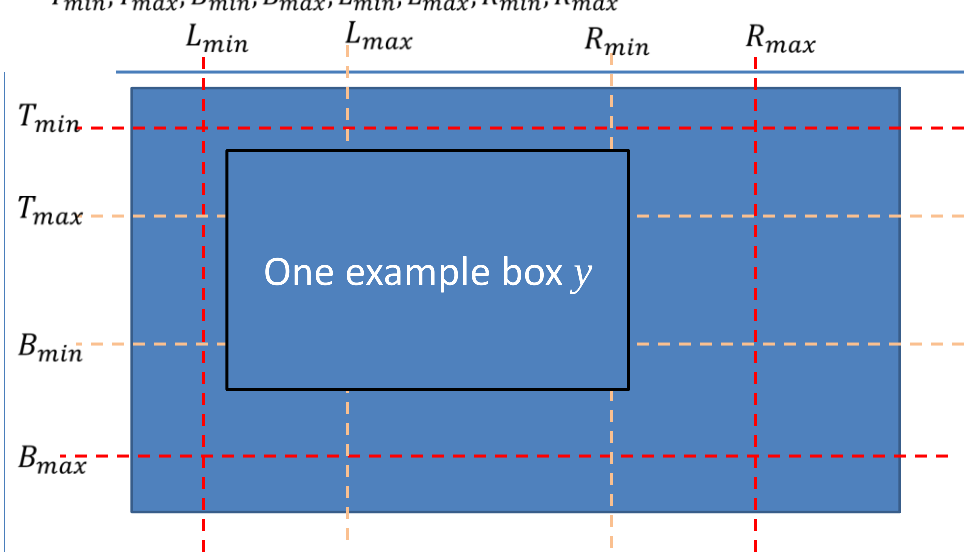


$\underline{g_k} < \max_{a \in A_k} g(a) < \overline{g_k}$

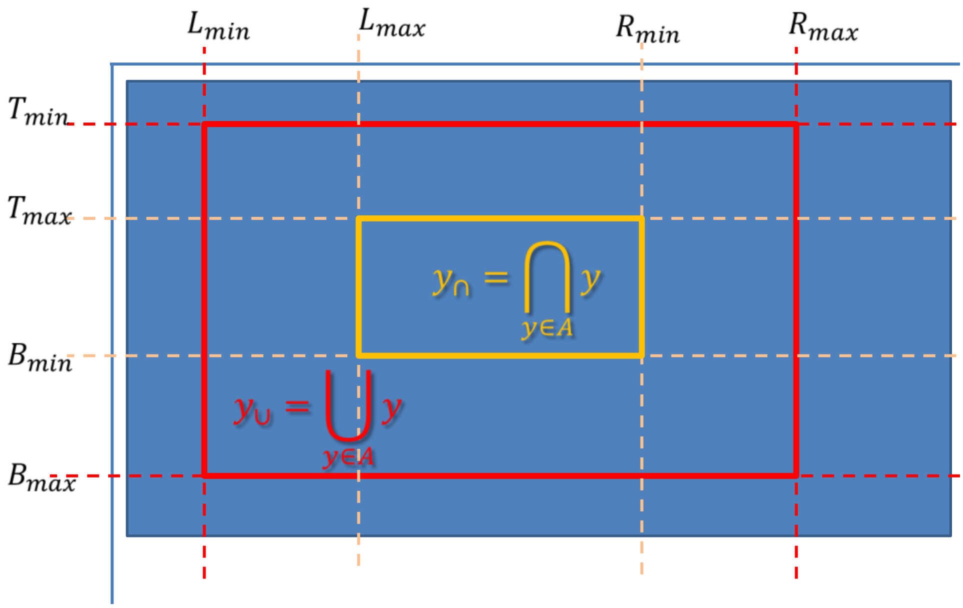
- if $\overline{g_k} < \underline{g_l}$ for all l then there is no point in exploring A_k
- If $\overline{g(\{y\})} = \underline{g(\{y\})} = g(\{y\})$ for all y
- Notes:
  - Worst case complexity remains the same
  - Lower is trivial: Pick any a in A_k

Branch-and-bound for detection

- *A* = set of boxes
- Each box parameterized by $[T, B, L, R]$
- Each set A parameterized by $T_{min}, T_{max}, B_{min}, B_{max}, L_{min}, L_{max}, R_{min}, R_{max}$

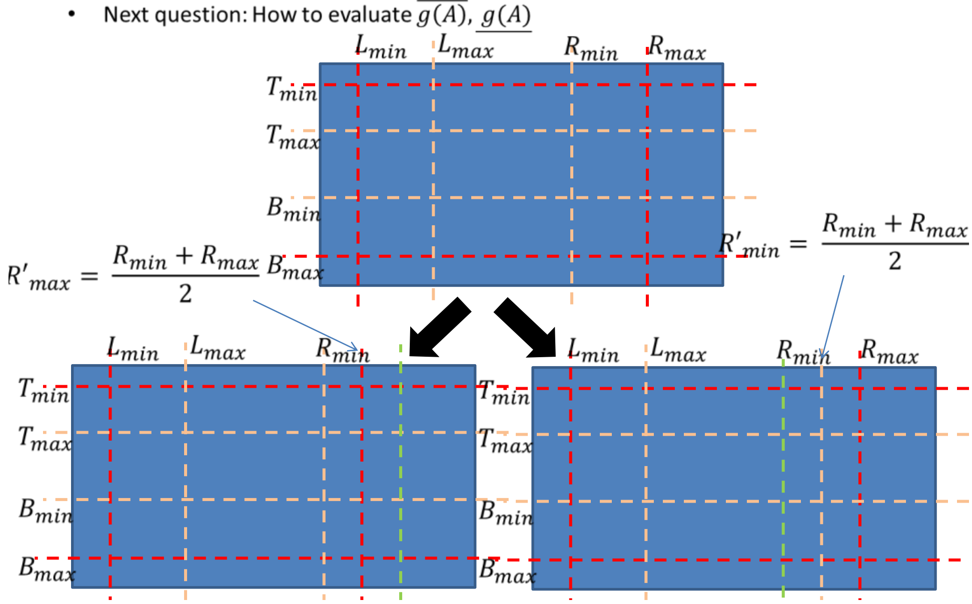


Branch-and-bound for detection



Subwindow search

- Depth first search: split current set of windows A by splitting one of the intervals $[T_{min}, T_{max}]$, $[B_{min}, B_{max}]$, $[L_{min}, L_{max}]$, $[R_{min}, R_{max}]$ in the middle
- Next question: How to evaluate $\overline{g(A)}$, $\underline{g(A)}$



Additive scores

- We consider scores of the form:

$$g(x, y) = \sum_{x_i \text{ occurs in } y} w(x_i)$$

Example BoW:

- $h(x_i) =$ number of times word x_i occurs in box y
- $w(x_i) =$ entry of weight vector w for word x_i

$$g(x, y) = \sum_{x_i} w(x_i) h(x_i) = \sum_{x_i \text{ occurs in } y} w(x_i)$$

Branch-and-bound for detection

- If score is additive, then simple upper bound:

- $\bar{g}(x, y) = \sum_{x_i \text{ occurs in } y_\cup} \max(w(x_i), 0) + \sum_{x_i \text{ occurs in } y_\cap} \min(w(x_i), 0)$

- $\bar{g}(x, y) = \sum_{x_i \text{ occurs in } y_\cup} w(x_i)^+ + \sum_{x_i \text{ occurs in } y_\cap} w(x_i)^-$

Efficient sliding windows (ESS)

- Branch:
  - Depth first search: split current set of windows A by splitting one of the intervals $[T_{min}, T_{max}]$, $[B_{min}, B_{max}]$, $[L_{min}, L_{max}]$, $[R_{min}, R_{max}$] in the middle
- Bound:
  $\bar{g}(x, y)$

$$= \sum_{x_i \text{ occurs in } y_\cup} w(x_i)^+ + \sum_{x_i \text{ occurs in } y_\cap} w(x_i)^-$$

Efficient sliding windows (ESS)

- Branch:
  - Depth first search: split current set of windows A by splitting one of the intervals $[T_{min}, T_{max}]$, $[B_{min}, B_{max}]$, $[L_{min}, L_{max}]$, $[R_{min}, R_{max}]$ in the middle
- Bound:

$$\bar{g}(x, y) = \sum_{x_i \text{ occurs in } y_\cup} w(x_i)^+ + \sum_{x_i \text{ occurs in } y_\cap} w(x_i)^-$$

- O(1) computation of bound
- Can be extended to non-linear operations on histogram representations: histogram intersection, χ^2 , pyramid kernels
- Later: BB idea can be applied to other problems, e.g., non-tree inference

Reminder of key results

- Exact algorithms on tree-structured graphs
  - Message passing
  - Max-product: compute $y*$
  - Sum-product: estimate marginals $p(y_i|x)$

- Today:
  - Details of max/sum for tree-structured models
    - Detection with parts
    - Pose estimation
  - Efficient search for detection (in some special cases)
- Next:
  - Details of general cases for segmentation and labeling