HUMAN CAPITAL
NATIONAL COHESION STRATEGY

EUROPEAN UNION
EUROPEAN
SOCIAL FUND

Binary Classification, Multi-label Classification and Ranking: A Decision-theoretic Approach

Krzysztof Dembczyński and Wojciech Kotłowski

Intelligent Decision Support Systems Laboratory (IDSS)
Poznań University of Technology, Poland

PAN Summer School, 29.06-04.07.2014

Agenda

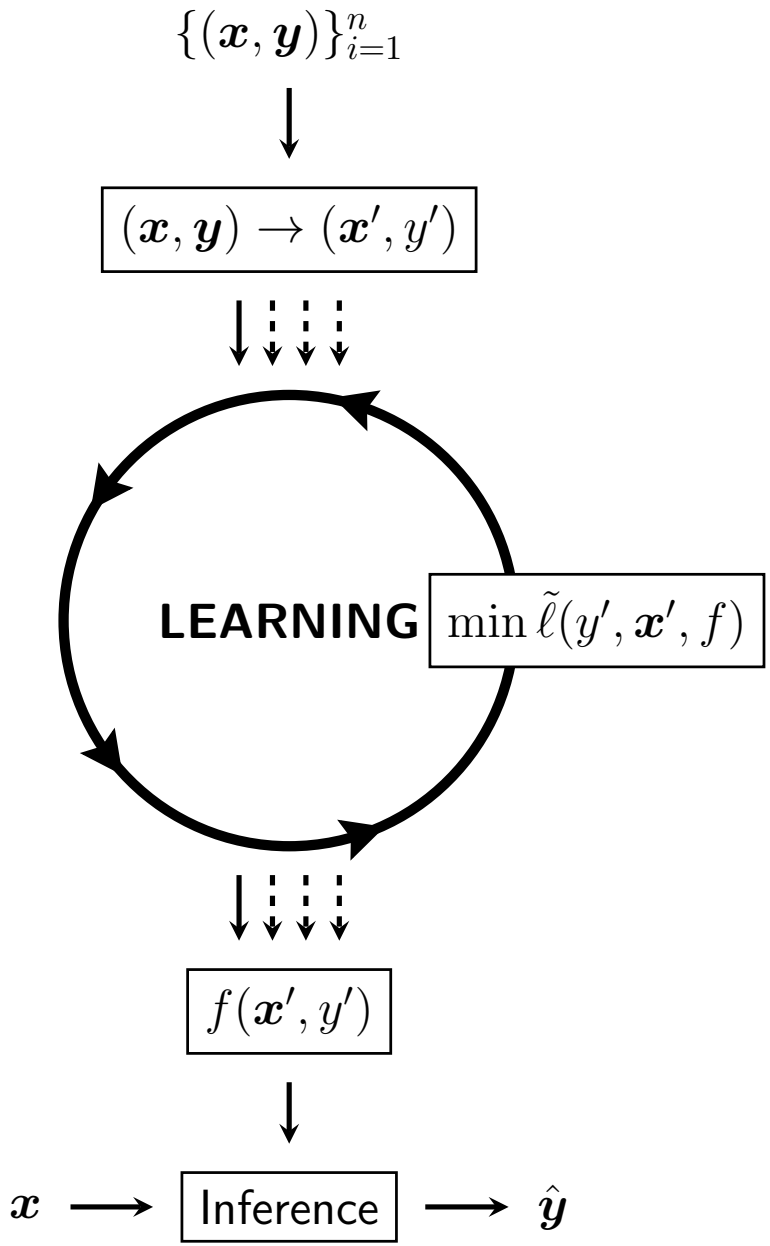
1. Binary Classification
2. Bipartite Ranking
3. Multi-Label Classification
4. **Reductions in Multi-Label Classification**
5. Conditional Ranking

The project is co-financed by the European Union from resources of the European Social Found

HUMAN CAPITAL
NATIONAL COHESION STRATEGY

EUROPEAN UNION
EUROPEAN
SOCIAL FUND

Reduction



- **Reduce** the original problem into problems of simpler type, for which efficient algorithmic solutions are available.
- Reduction to one or a sequence of problems.
- Plug-in rule classifiers.

Reduction

- We would like to find a reduction algorithm that works for **any task loss**.
- Ideally, the reduction should be **consistent** and **efficient** in training and in inference.

Reduction

- Binary relevance (BR)
- Label powerset (LP)
- Probabilistic classifier chains (PCC)
- Filter tree (FT)
- Plug-in rule classifiers for the F-measure (LFP and EFP)
- Principal label space transformation (PLST)

Outline

Outline

Probabilistic classifier chains

- Probabilistic classifier chains (PCCs)[1] similarly to CRFs estimate the joint conditional distribution $P(\boldsymbol{y} \,|\, \boldsymbol{x})$.
- Their idea is to repeatedly apply the **product rule of probability**:

$$P(\boldsymbol{y} \,|\, \boldsymbol{x}) = \prod_{i=1}^{m} P(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) \,.$$

[1] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning Journal*, 85:333–359, 2011

K. Dembczyński, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286. Omnipress, 2010

Probabilistic classifier chains

- Probabilistic classifier chains (PCCs)[1] similarly to CRFs estimate the joint conditional distribution $P(\boldsymbol{y} \,|\, \boldsymbol{x})$.
- Their idea is to repeatedly apply the **product rule of probability**:

$$P(\boldsymbol{y} \,|\, \boldsymbol{x}) = \prod_{i=1}^{m} P(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) \,.$$

- **Example**:

$$P(y_1, y_2 \,|\, \boldsymbol{x}) = \frac{P(y_1, \boldsymbol{x})}{P(\boldsymbol{x})} \frac{P(y_1, y_2, \boldsymbol{x})}{P(y_1, \boldsymbol{x})} = P(y_1 \,|\, \boldsymbol{x}) P(y_2 \,|\, y_1, \boldsymbol{x}) \,.$$

---

[1] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning Journal*, 85:333–359, 2011

K. Dembczyński, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286. Omnipress, 2010

Probabilistic classifier chains

- PCCs follow a reduction to a sequence of subproblems:

$$(\boldsymbol{x}, \boldsymbol{y}) \longrightarrow (\boldsymbol{x}' = (\boldsymbol{x}, y_1, \ldots, y_{i-1}), y = y_i), \quad i = 1, \ldots, m$$

- Learning of PCCs relies on constructing probabilistic classifiers for estimating

$$P(y_i | \boldsymbol{x}, y_1, \ldots, y_{i-1}) \,,$$

  independently for each $i = 1, \ldots, m$.
- Let us denote these estimates by

$$Q(y_i | \boldsymbol{x}, y_1, \ldots, y_{i-1}) \,.$$

- The final model is:

$$Q(\boldsymbol{y} \,|\, \boldsymbol{x}) = \prod_{i=1}^{m} Q(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) \,.$$

Probabilistic classifier chains

- We can use scoring functions of the form $f_i(\boldsymbol{x}', y_i)$ and train logistic regression to get $Q(y_i|\boldsymbol{x}, y_1, \ldots, y_{i-1})$.
- By using the linear models, the overall scoring function takes the form:

$$f(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} f_i(\boldsymbol{x}, y_i) + \sum_{y_k, y_l} f_{k,l}(y_k, y_l)$$

- Theoretically the order of labels does not matter, but practically it may.

Probabilistic classifier chains

- PCCs enable estimation of probability of any label vector $\boldsymbol{y}$.
- To get such an estimate it is enough to compute:

$$Q(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{i=1}^{m} Q(y_i | \boldsymbol{x}, y_1, \ldots, y_{i-1})$$

- There is, however, a problem how to compute the optimal decision $\boldsymbol{h}(\boldsymbol{x})$ (with respect to Q) for a given loss function.

Probabilistic classifier chains

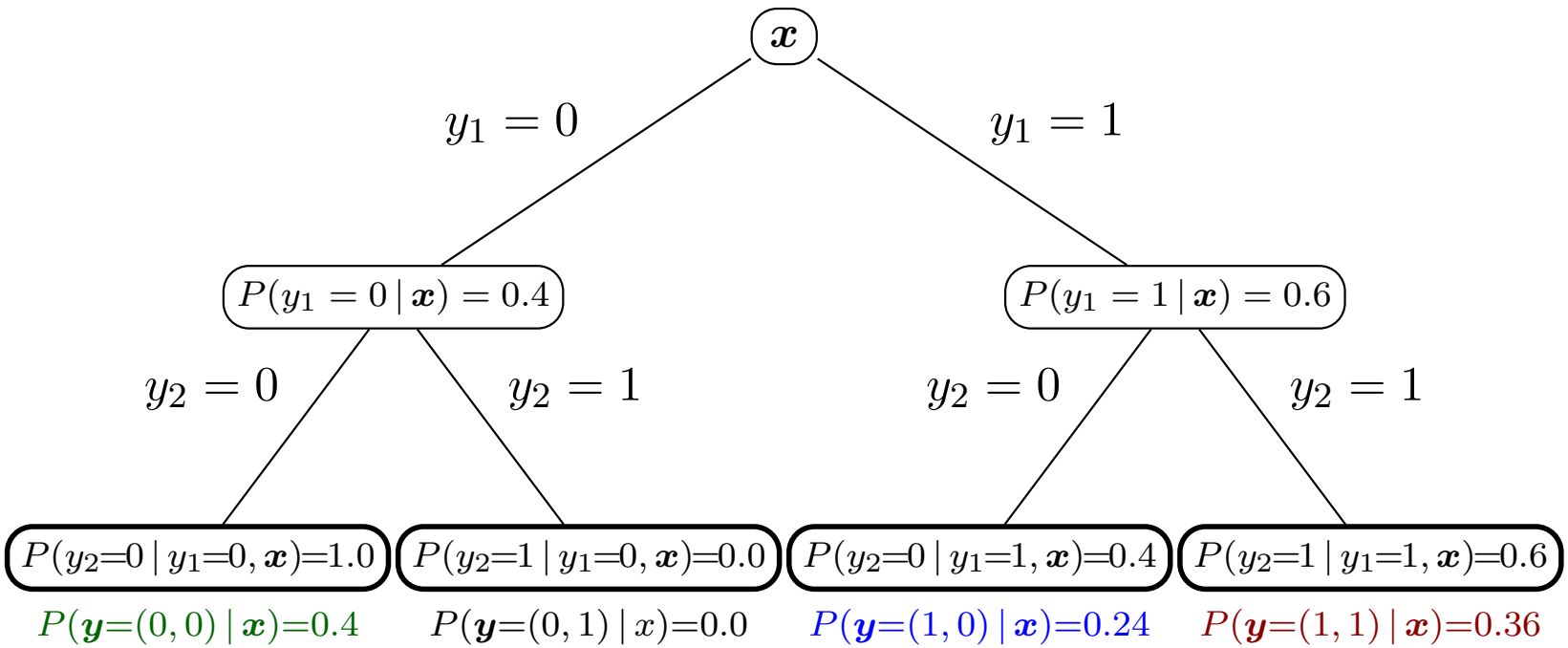
- Inference in PCCs:
  - Greedy search,
  - Advanced search techniques: beam search, uniform-cost search,
  - Exhaustive search,
  - Sampling + inference.

Greedy search

- Greedy search follows the chain by using predictions from previous steps as inputs in the consecutive steps:
  - $f_1 : \boldsymbol{x} \mapsto \hat{y}_1$
  - $f_2 : \boldsymbol{x}, \hat{y}_1 \mapsto \hat{y}_2$
  - $f_3 : \boldsymbol{x}, \hat{y}_1, y_2 \mapsto \hat{y}_3$
  - $\ldots$
  - $f_m : \boldsymbol{x}, \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{m-1} \mapsto \hat{y}_m$
- Greedy search is fast ($O(m)$).
- Does not require probabilistic classifiers.
- The resulting $\hat{\boldsymbol{y}}$ is neither the joint nor the marginal mode.
- Optimal if labels are independent or the probability of the joint mode > 0.5.

Greedy search

- Greedy search fails for the joint mode and the marginal mode:



Advanced search techniques

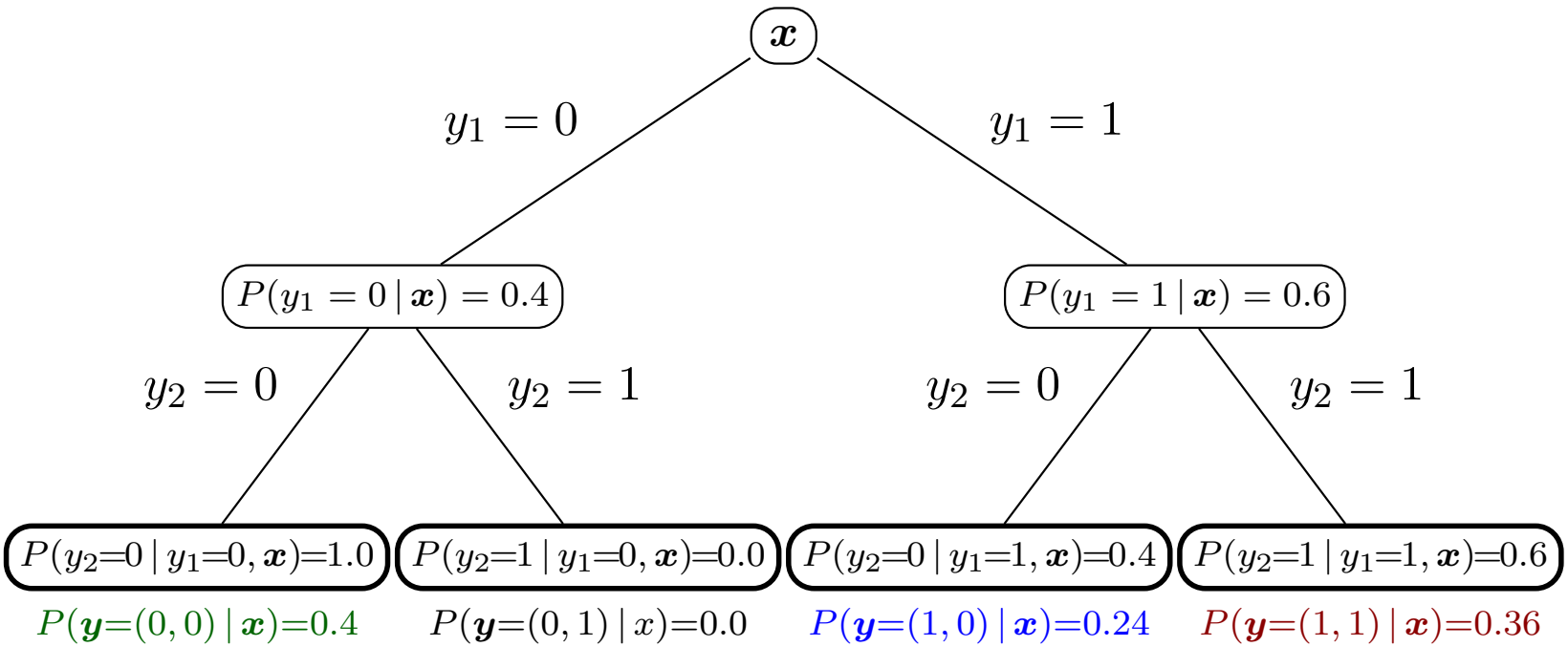
- Advanced search techniques: beam search,[2] a variant of uniform-cost search.[3]
- Finding the joint mode relies on finding the most probable path in the tree.
- The use of a priority queue and a cut point gives a fast algorithm with provable guarantees.

[2] A. Kumar, S. Vembu, A.K. Menon, and C. Elkan. Beam search algorithms for multilabel learning. In *Machine Learning*, 2013

[3] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. An analysis of chaining in multi-label classification. In *ECAI*, 2012

Advanced search techniques

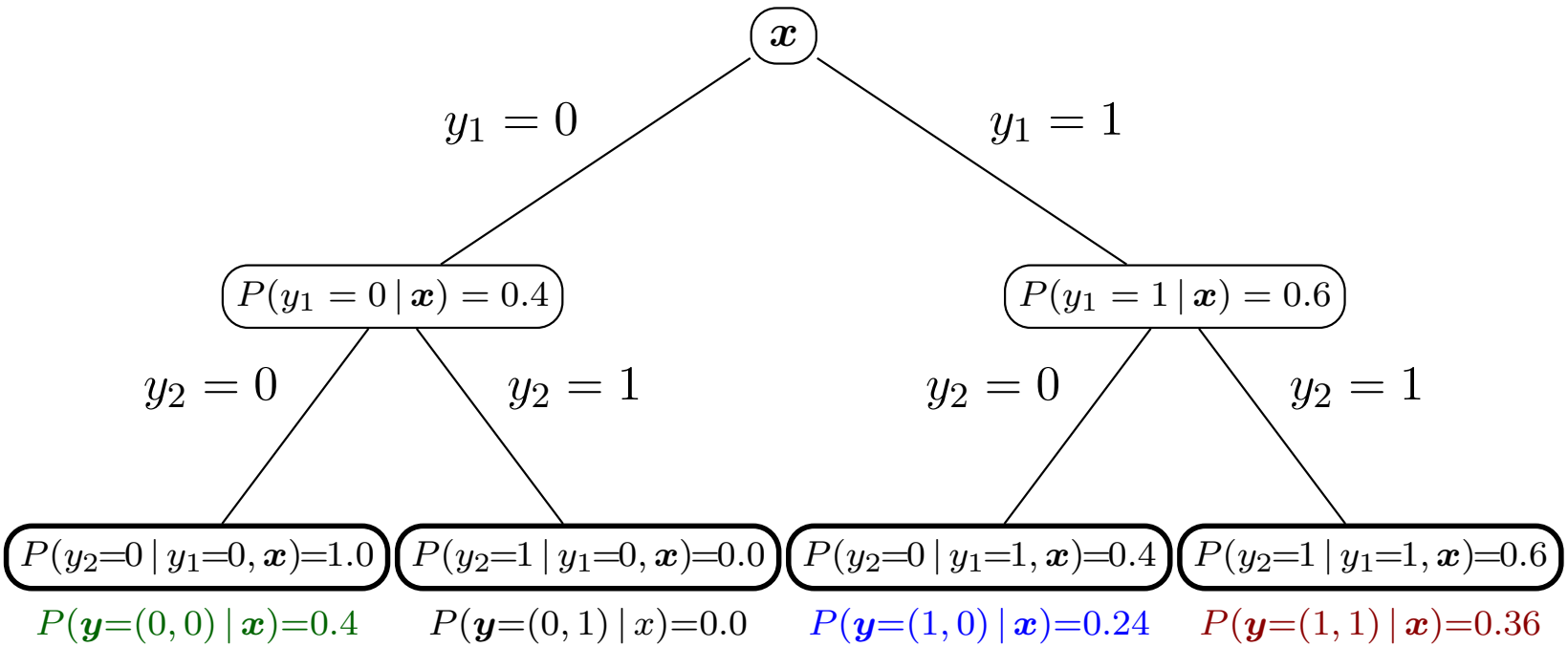
- Uniform-cost search



- Priority list $\mathcal{Q}$:

Advanced search techniques

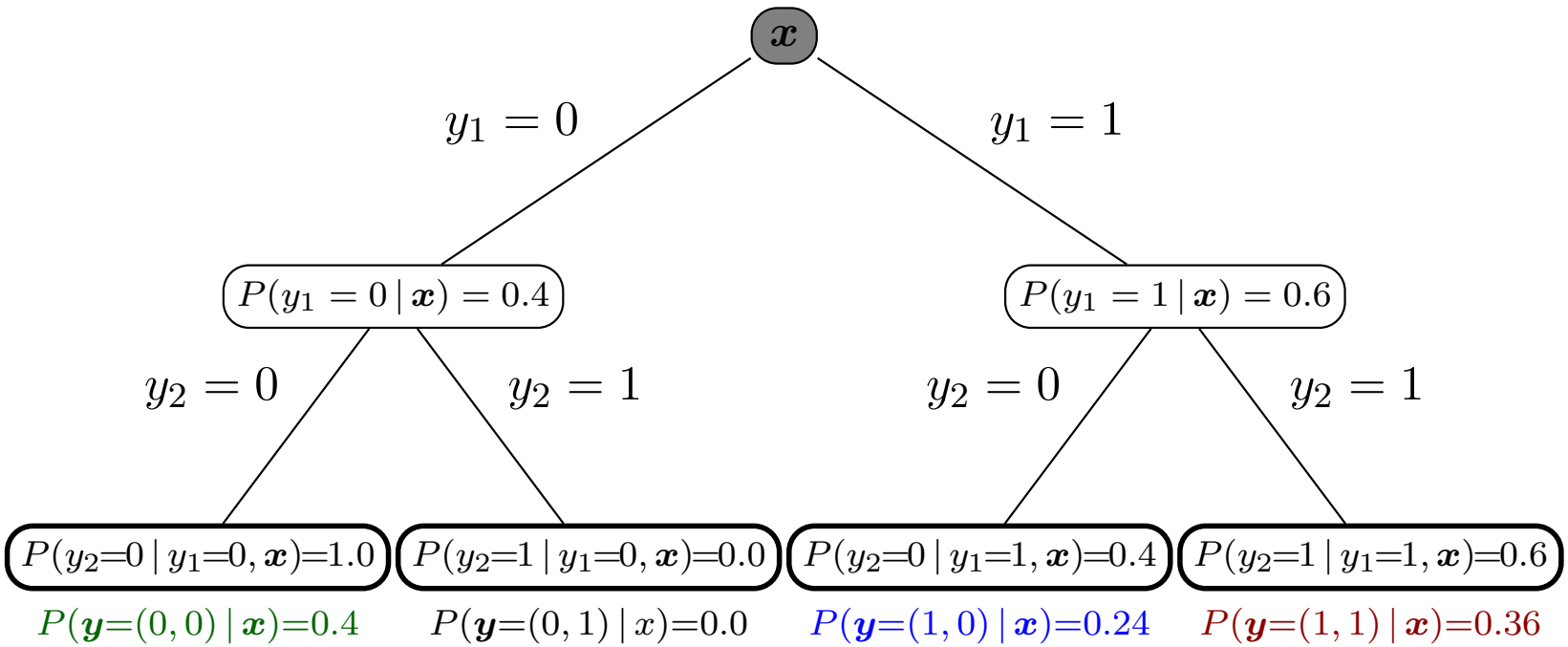
- Uniform-cost search



- Priority list $\mathcal{Q}$: root

Advanced search techniques

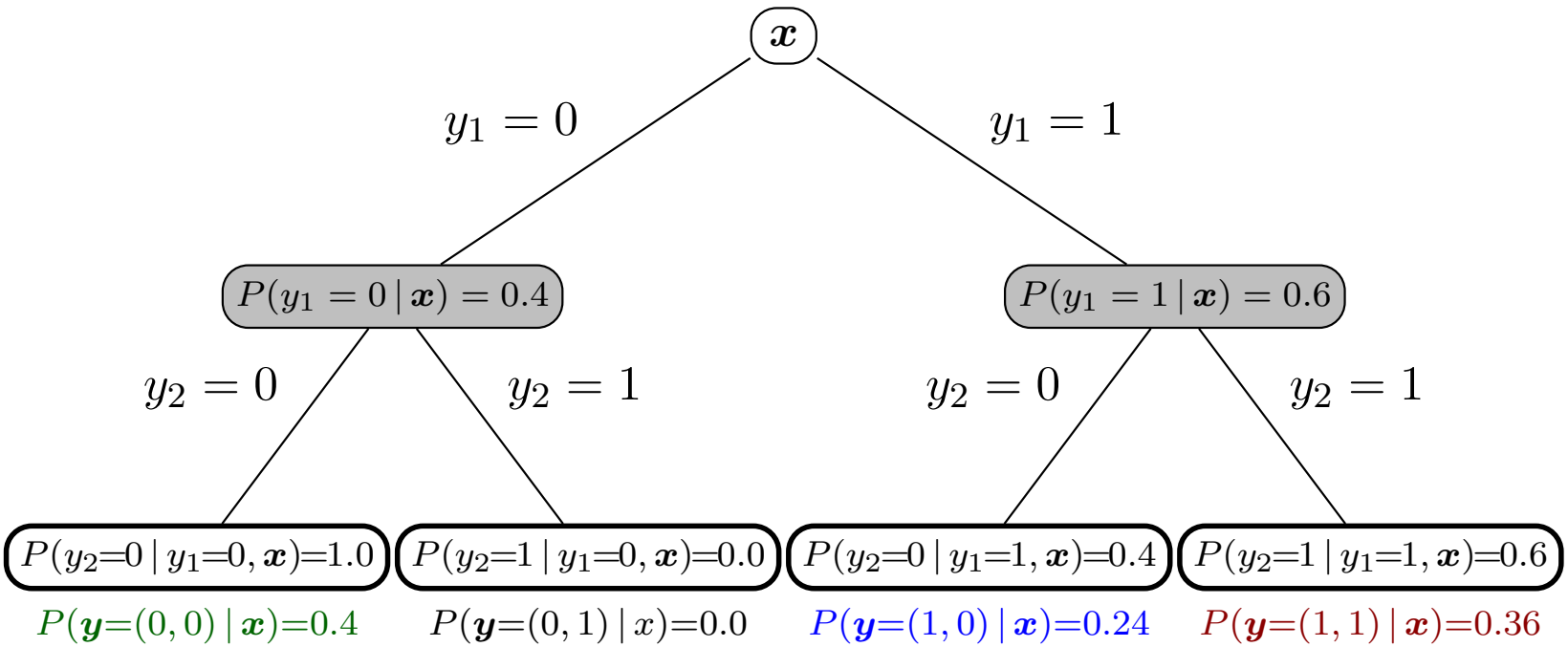
- Uniform-cost search



- Priority list $\mathcal{Q}$:

Advanced search techniques

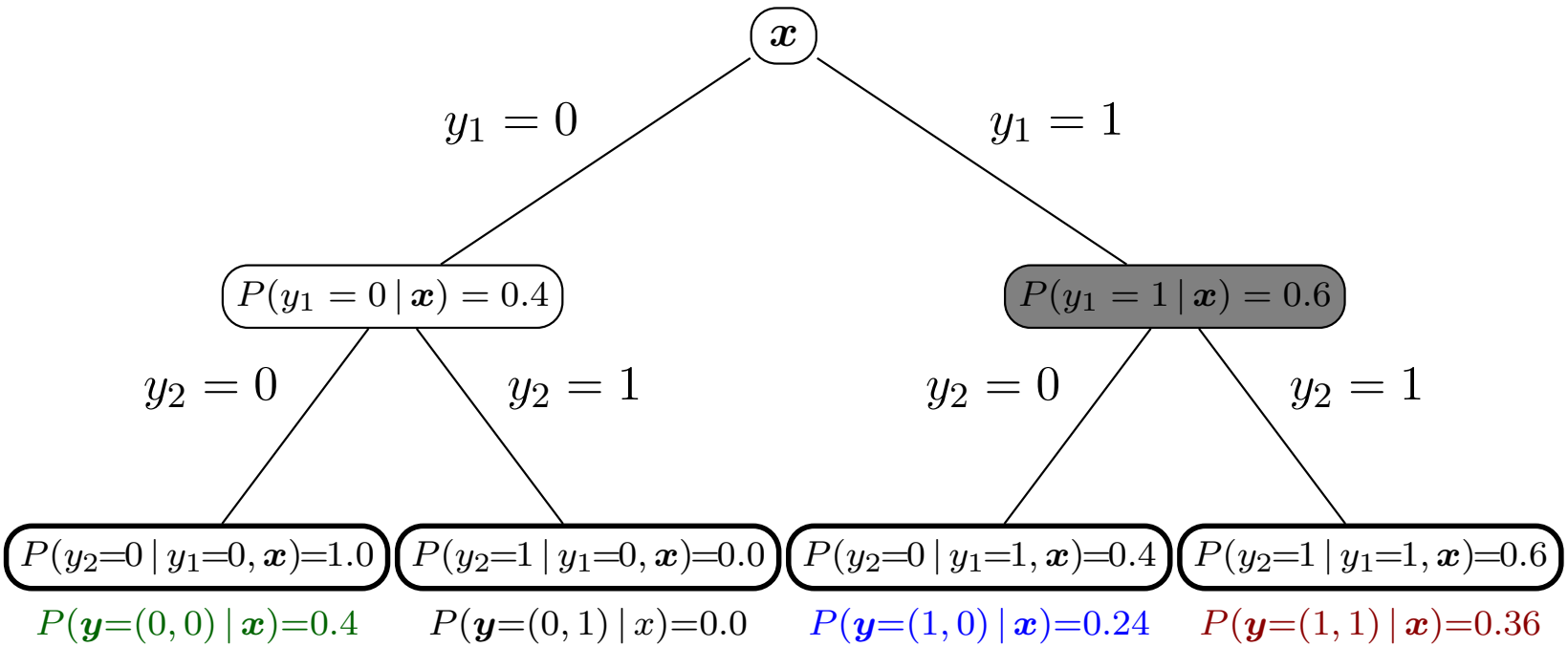
- Uniform-cost search



- Priority list $\mathcal{Q}$: [(1),0.6], [(0),0.4]

Advanced search techniques

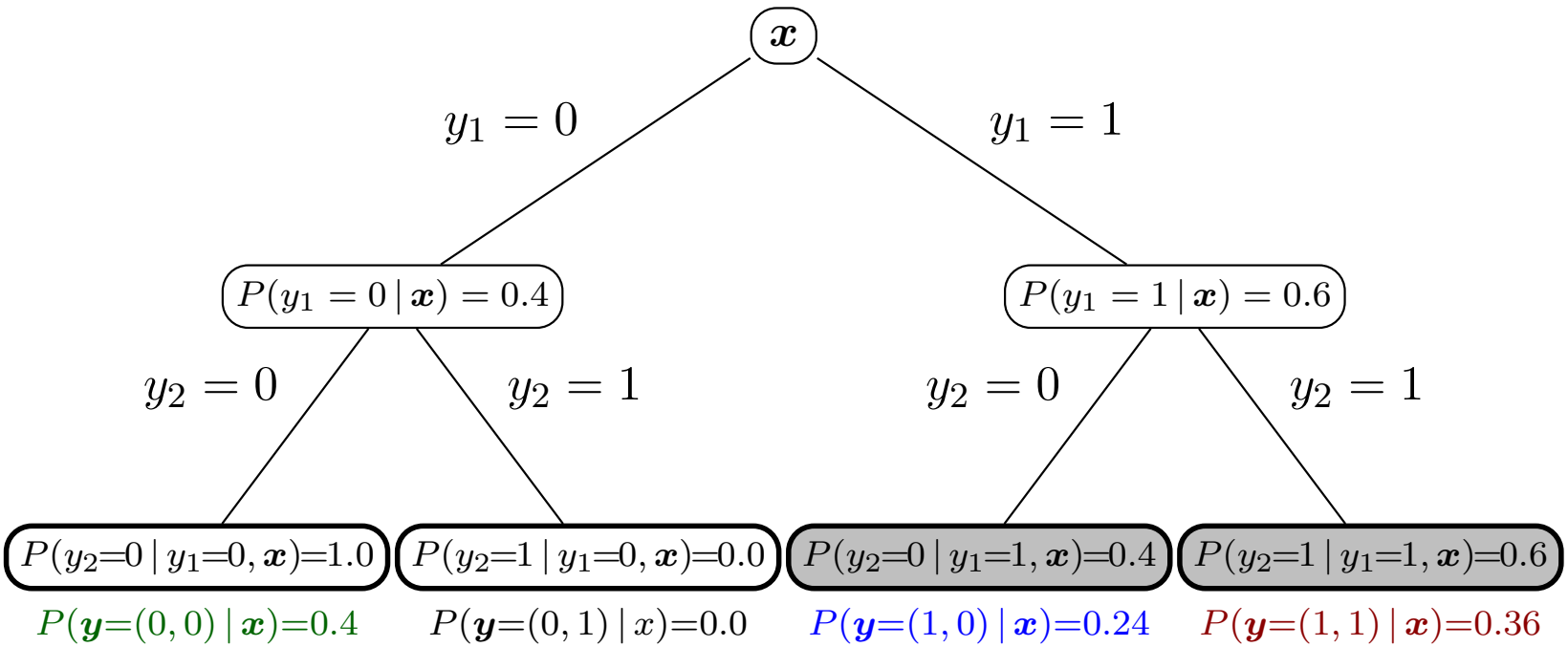
- Uniform-cost search



- Priority list $\mathcal{Q}$: [(0),0.4]

Advanced search techniques

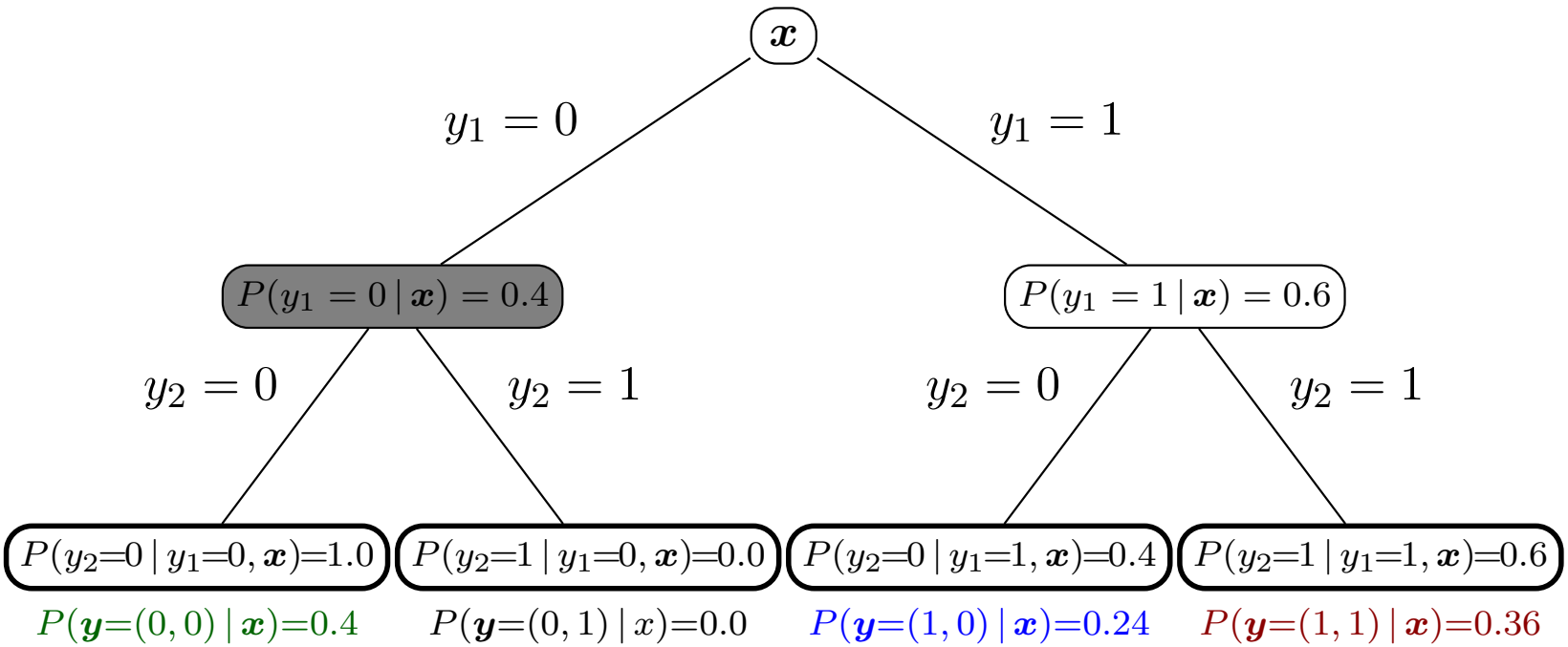
- Uniform-cost search



- Priority list $\mathcal{Q}$: [(0),0.4], [(1,1),0.36], [(1,0),0.24]

Advanced search techniques

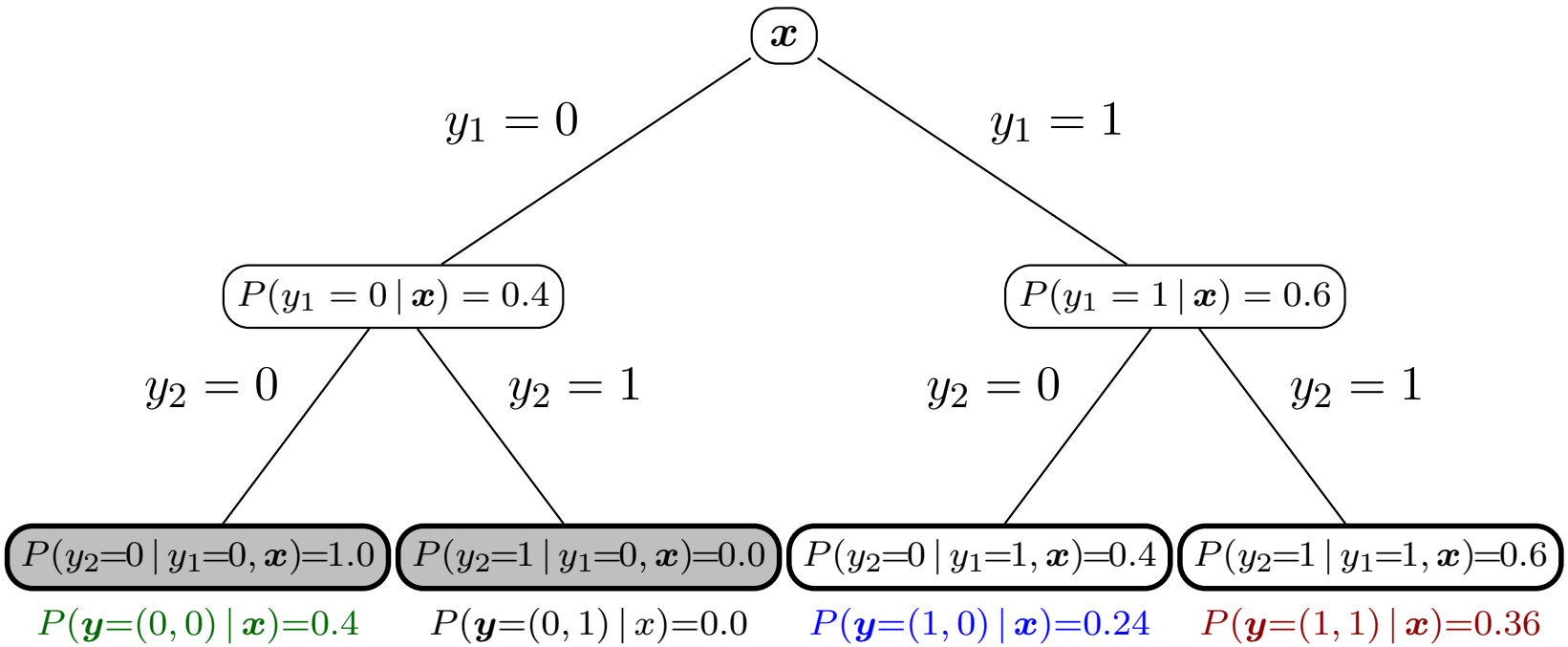
- Uniform-cost search



- Priority list $\mathcal{Q}$: [(1,1),0.36], [(1,0),0.24]

Advanced search techniques

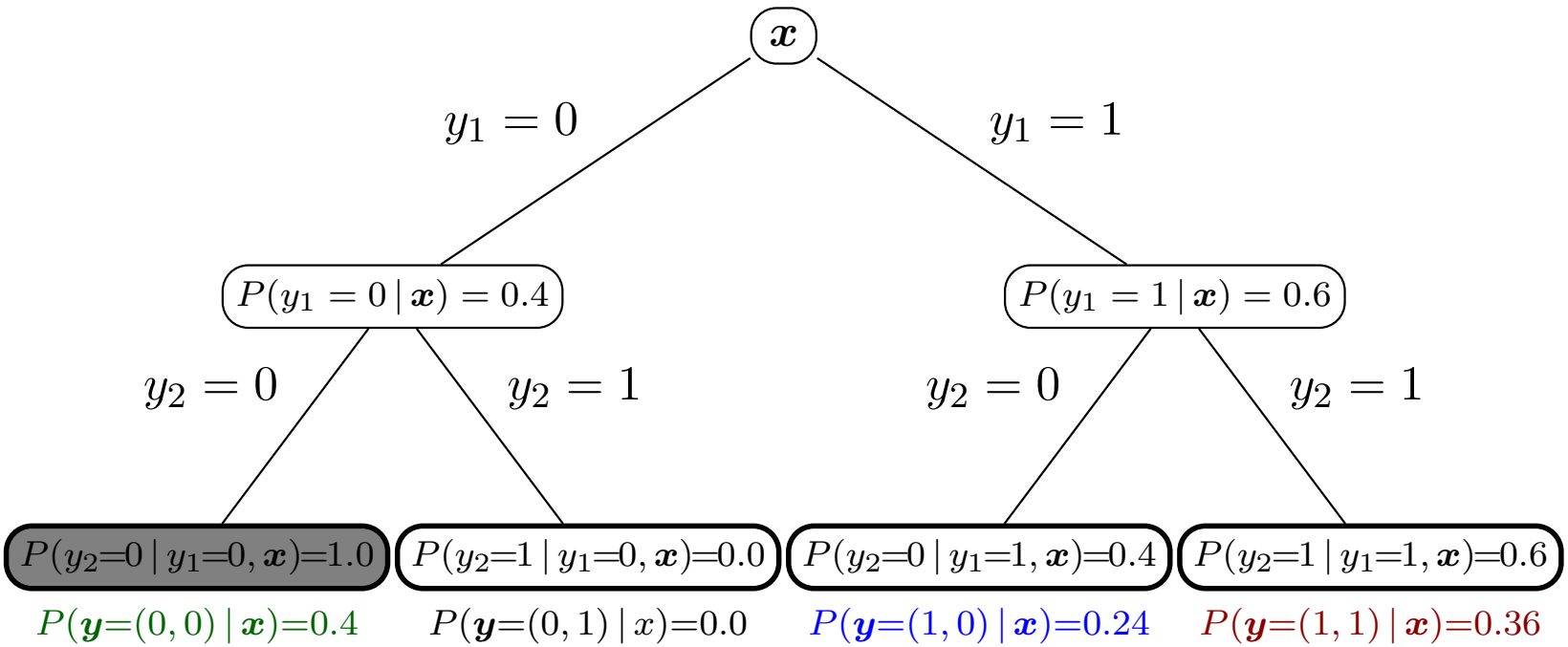
- Uniform-cost search



- Priority list $\mathcal{Q}$: [(0,0),0.4], [(1,1),0.36], [(1,0),0.24], [(0,1),0.0]

Advanced search techniques

- Uniform-cost search



- Priority list $\mathcal{Q}$: Solution is found

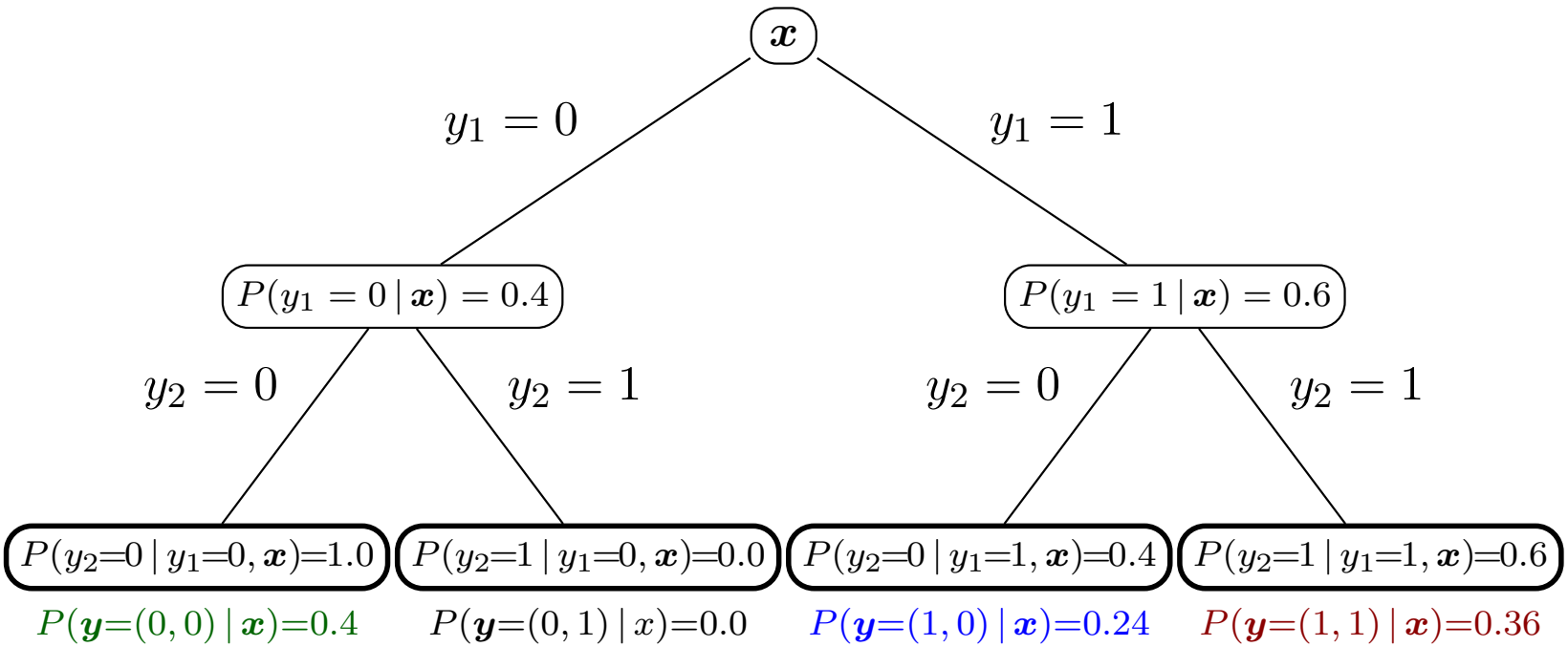
Advanced search techniques

- ϵ-approximation inference:[4]
  - Insert items to priority queue $\mathcal{Q}$ with partial probabilities $> \epsilon$.
  - If solution has not been found, then perform greedy search from nodes without survived children.

[4] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. An analysis of chaining in multi-label classification. In *ECAI*, 2012

ϵ-approximation inference

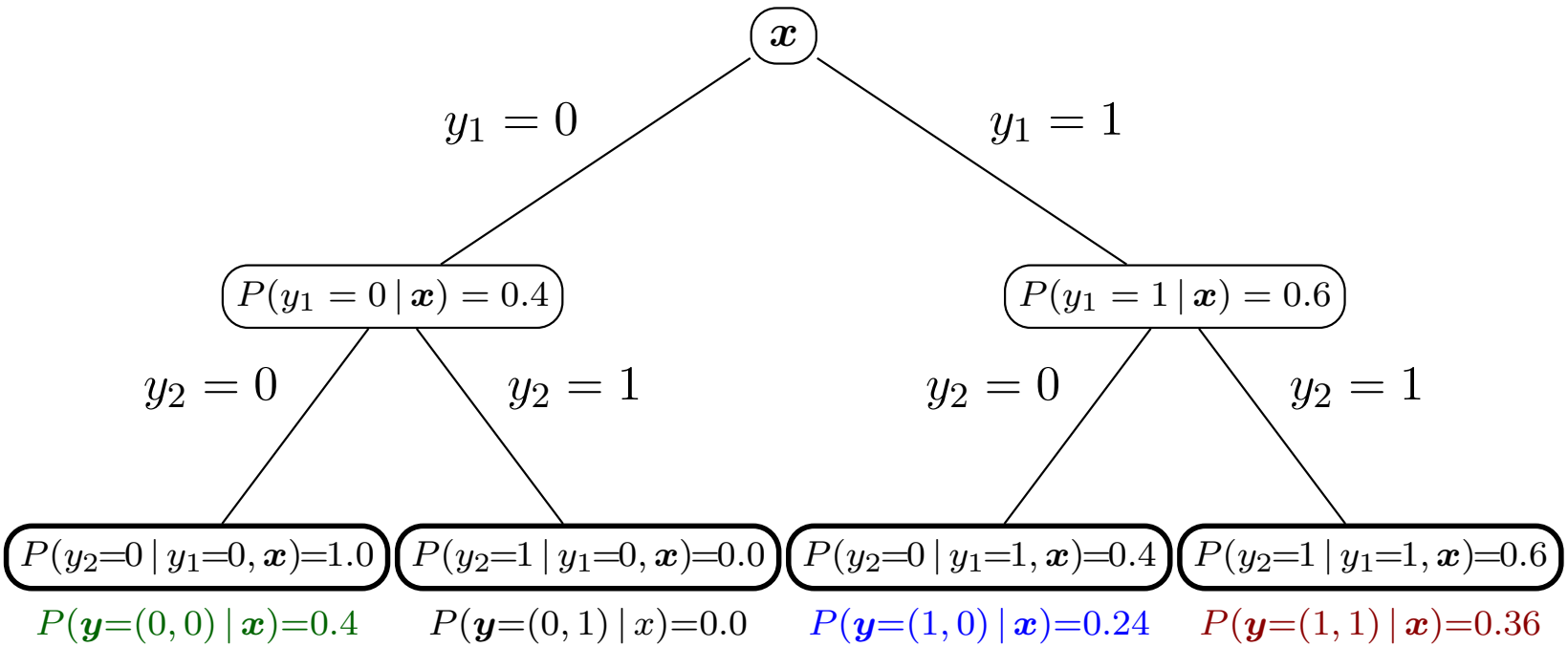
- $\epsilon = 0.5$



- Priority list $\mathcal{Q}$:

ϵ-approximation inference

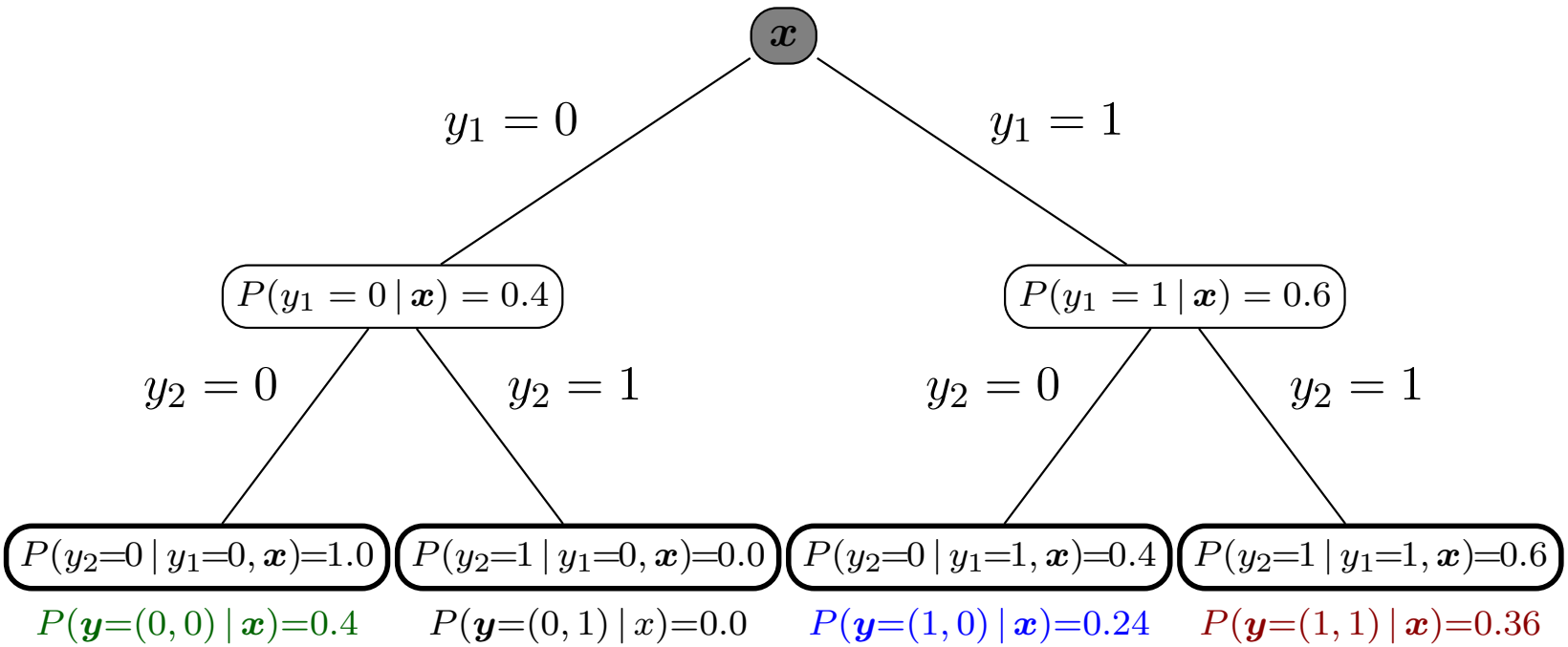
- $\epsilon = 0.5$



- Priority list $\mathcal{Q}$: root

ϵ-approximation inference

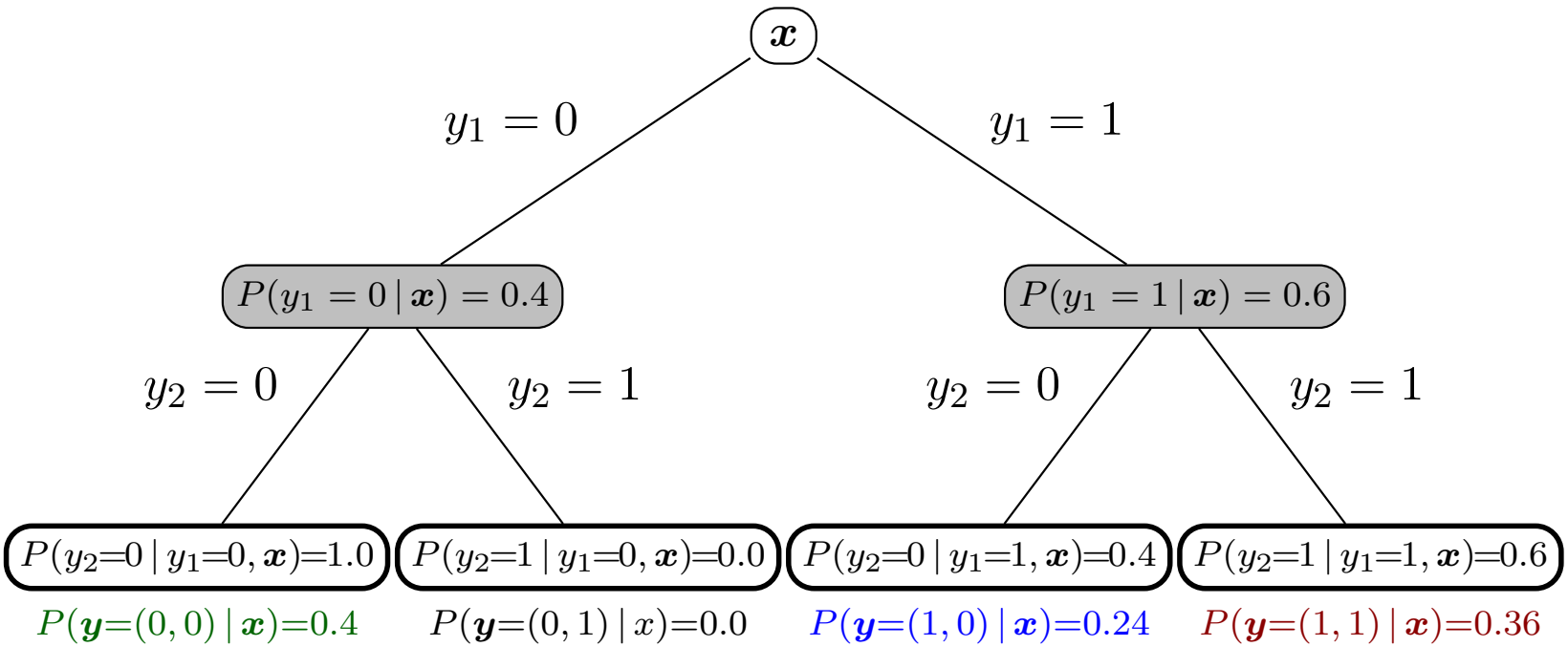
- $\epsilon = 0.5$



- Priority list $\mathcal{Q}$: $\epsilon = 0.5$

ϵ-approximation inference

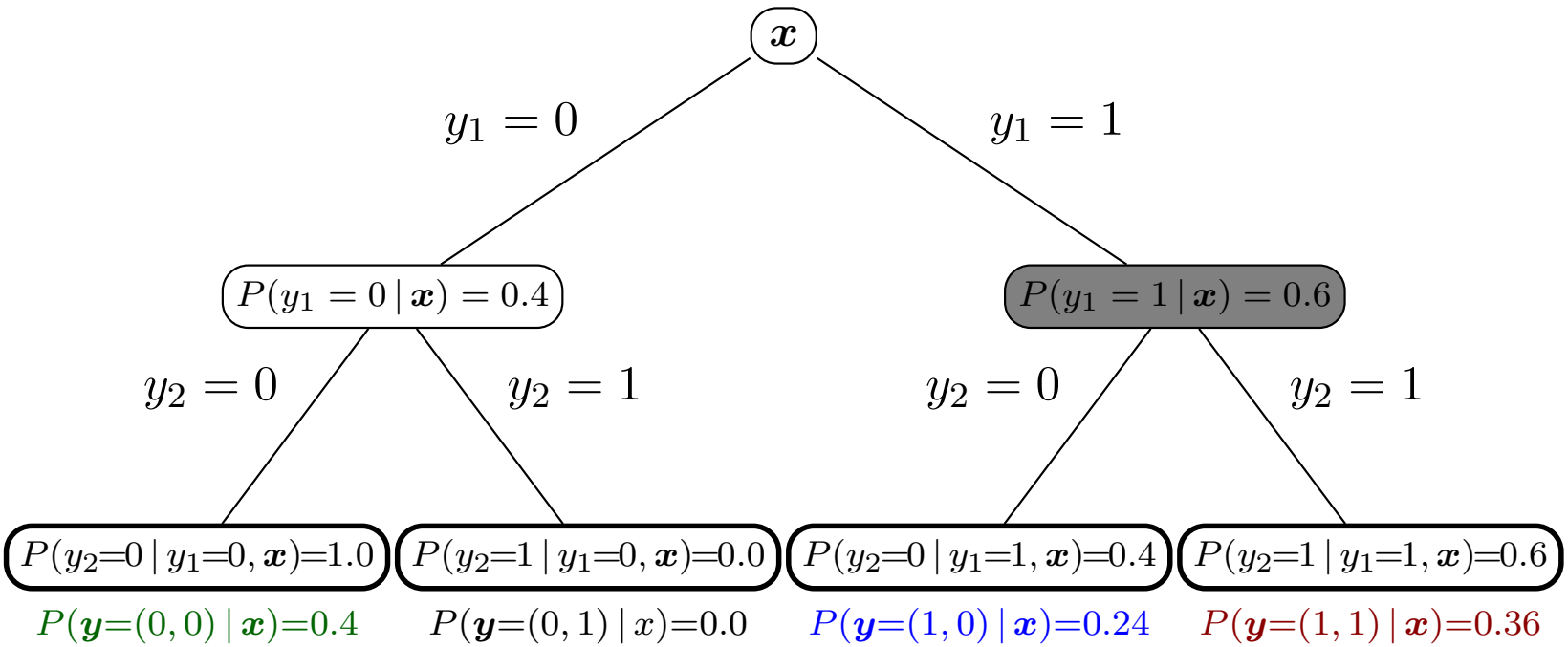
- $\epsilon = 0.5$



- Priority list $\mathcal{Q}$: [(1),0.6], $\epsilon = 0.5$, [(0),0.4]

ϵ-approximation inference

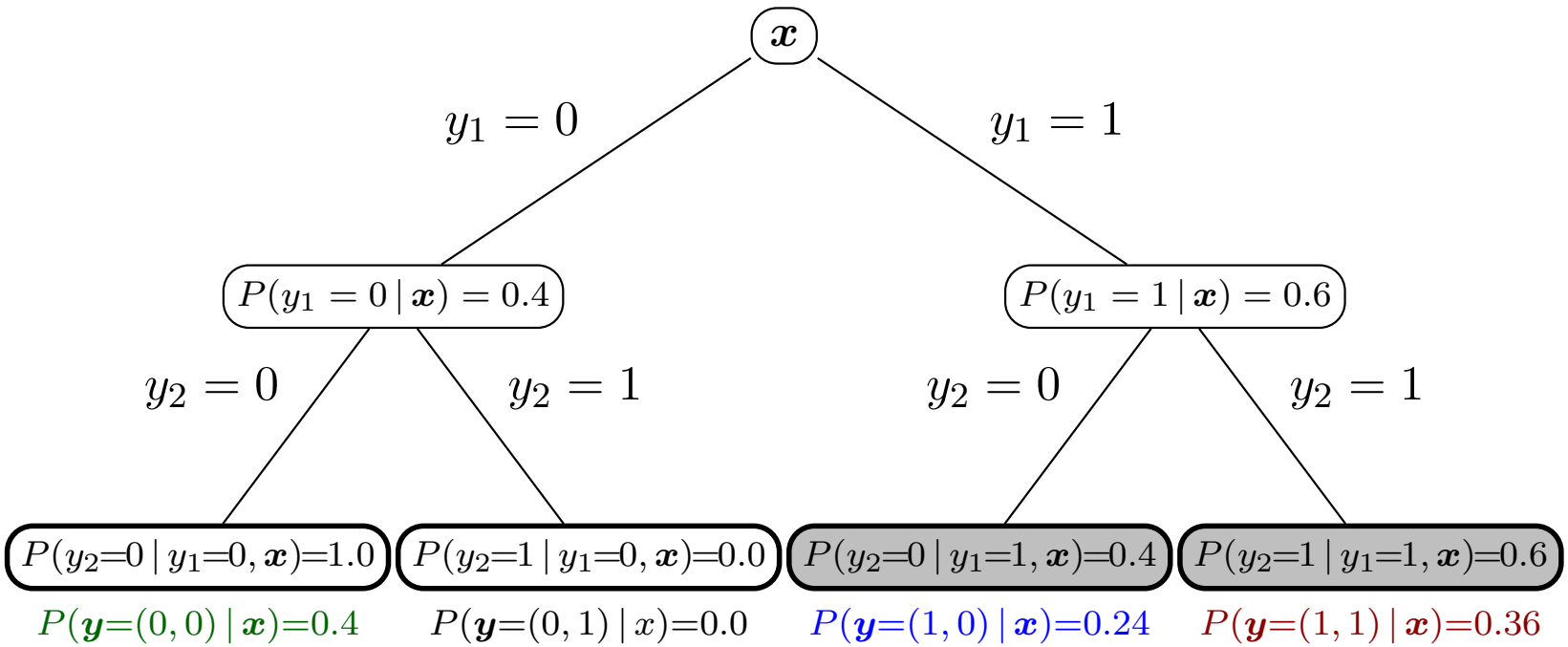
- $\epsilon = 0.5$



- Priority list $\mathcal{Q}$: $\epsilon = 0.5$, [(0),0.4]

ϵ-approximation inference

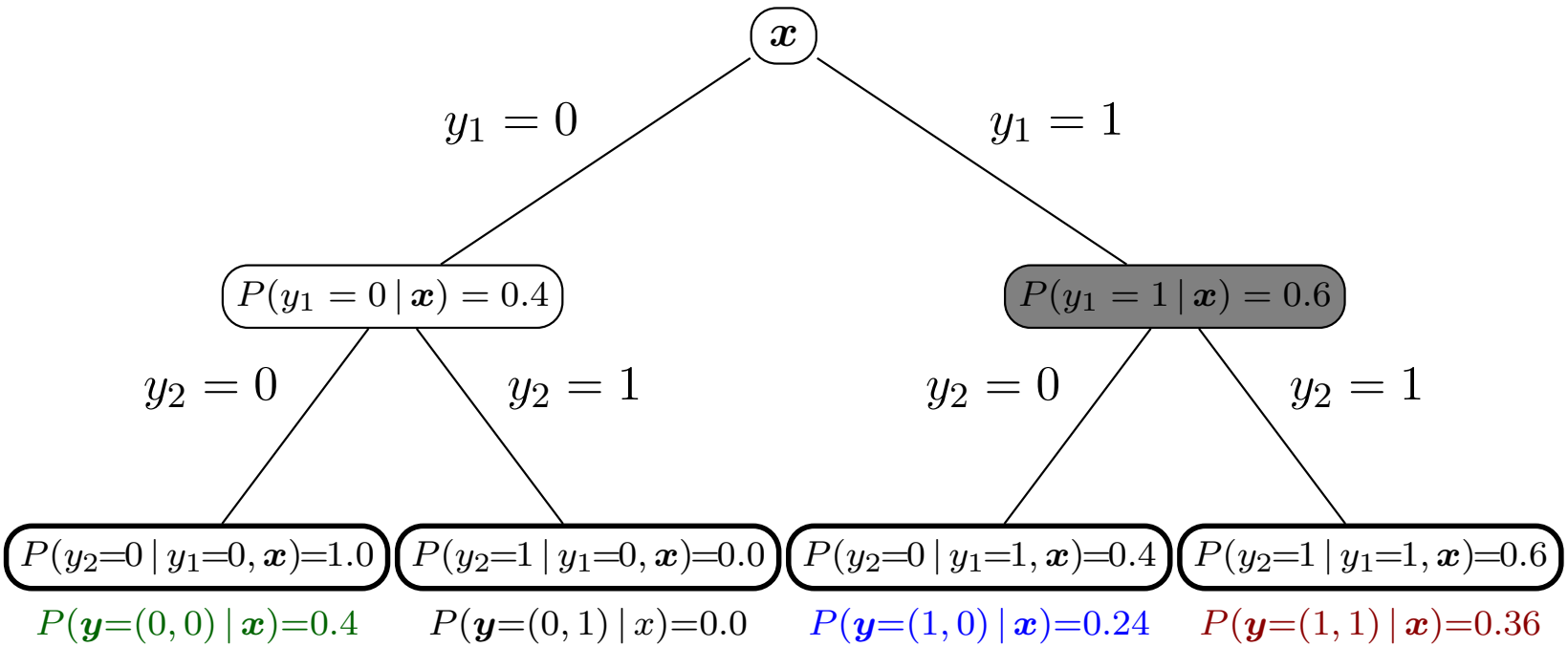
- $\epsilon = 0.5$



- Priority list $\mathcal{Q}$: $\epsilon = 0.5$, [(0),0.4], [(1,1),0.36], [(1,0),0.24]

ϵ-approximation inference

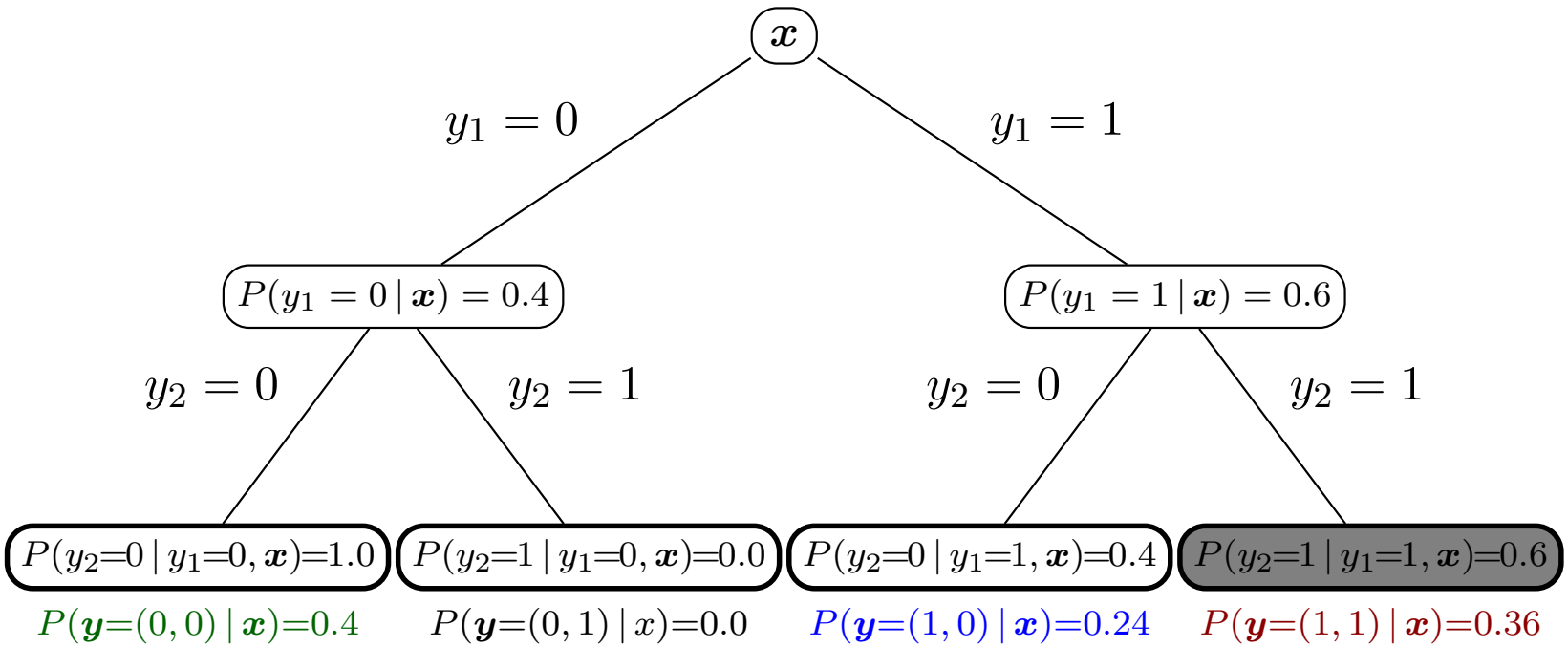
- $\epsilon = 0.5$



- Priority list $\mathcal{Q}$: Start the greedy search from (1).

ϵ-approximation inference

- $\epsilon = 0.5$



- Priority list $\mathcal{Q}$: Suboptimal solution (1,1) is found.

ϵ-approximation inference

- For $\epsilon = 0.5$, it is equivalent to greedy search.
- For $\epsilon = 0.0$, it is equivalent to uniform-cost search.
- For a given ϵ, the following guarantees can be given:

ϵ-approximation inference

- **Theorem**: Let $\epsilon = 2^{-c}$, where $1 \leq c \leq m$. The label vector $\hat{\boldsymbol{y}}$ will be returned in time $\mathcal{O}(m2^c)$ with a guarantee that:

$$Q(\boldsymbol{y}^* \,|\, \boldsymbol{x}) - Q(\hat{\boldsymbol{y}} \,|\, \boldsymbol{x}) \leq \epsilon - 2^{-m}$$

ϵ-approximation inference

- **Theorem**: Let $\epsilon = 2^{-c}$, where $1 \leq c \leq m$. The label vector $\hat{\boldsymbol{y}}$ will be returned in time $\mathcal{O}(m2^c)$ with a guarantee that:

$$Q(\boldsymbol{y}^* \,|\, \boldsymbol{x}) - Q(\hat{\boldsymbol{y}} \,|\, \boldsymbol{x}) \leq \epsilon - 2^{-m}$$

- **Proof**:
  - A leaf node popped from the list $\mathcal{Q}$ is the solution.
  - The optimal label vector $\boldsymbol{y}^*$ that has not be founded by the constraint uniform-cost search has $Q(\boldsymbol{y}^* \,|\, \boldsymbol{x}) < \epsilon$.
  - The greedy search will always find a solution with $Q(\boldsymbol{y} \,|\, \boldsymbol{x}) \geq 2^{-m}$.
  - Each element $\boldsymbol{v}$ of the list $\mathcal{Q}$ has $Q(\boldsymbol{v}) \geq \epsilon$.
  - The sum of the elements on the list $\mathcal{Q}$ is always ≤ 1.
  - Therefore the list $\mathcal{Q}$ will always contain less than ϵ^{-1} elements.
  - Since every element in the list $\mathcal{Q}$ can be replaced at most m times by one or two new elements, we have $\mathcal{O}(m2^c)$.
  - The greedy search performed in the second part of the algorithm has also the worst case complexity of $\mathcal{O}(m2^c)$.

ϵ-approximation inference

- The ϵ-approximate inference will always find the joint mode if its probability mass $\geq \epsilon$.
- In other words, the algorithm finds the solution in a linear time of $1/p_{\max}$, where $p_{\max}$ is the probability mass of the joint mode.
- For problems with low noise (high values of $p_{\max}$), this method should work very fast.
- Greedy search has very bad guarantees:

$$Q(\boldsymbol{y}^* \,|\, \boldsymbol{x}) - Q(\hat{\boldsymbol{y}} \,|\, \boldsymbol{x}) \leq 0.5 - 2^{-m}\,.$$

Regret bound for PCC

- We have shown that there exist fast and accurate inference methods for PCC,
- But we do not know what are the guarantees for learning PCC.

Regret bound for PCC

- The typical approach for estimating probabilities of $\boldsymbol{y}$ is minimization of the logistic loss:

$$\ell_{\log}(\boldsymbol{y}, \boldsymbol{x}, f) = -\log Q(\boldsymbol{y} \,|\, \boldsymbol{x}) \,,$$

  where f is a model that delivers estimate $Q(\boldsymbol{y} \,|\, \boldsymbol{x})$ of $P(\boldsymbol{y} \,|\, \boldsymbol{x})$.
- By using the chain rule of probability, we get:

$$\begin{aligned} \ell_{\log}(\boldsymbol{y}, \boldsymbol{x}, f) &= -\log \prod_{i=1}^{m} Q(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) \\ &= -\sum_{i=1}^{m} \log Q(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) = -\sum_{i=1}^{m} \log Q_i(\boldsymbol{y}) \,, \end{aligned}$$

  where we use the notation $Q_i(\boldsymbol{y}) = Q(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1})$.
- This is a sum of univariate log losses on a path from the root to the leaf corresponding to $\boldsymbol{y}$.

Regret bound for PCC

- The conditional regret $\mathrm{Reg}_{\log}(f \,|\, \boldsymbol{x})$ for the logistic loss is defined as:

$$\begin{aligned}
\mathrm{Reg}_{\log}(f \,|\, \boldsymbol{x}) &= L_{\log}(f \,|\, \boldsymbol{x}) - L_{\log}(f^* \,|\, \boldsymbol{x}) \\
&= \sum_{\boldsymbol{y} \in \mathcal{Y}} P(\boldsymbol{y} \,|\, \boldsymbol{x}) (\log P(\boldsymbol{y} \,|\, \boldsymbol{x}) - \log Q(\boldsymbol{y} \,|\, \boldsymbol{x})) \\
&= D_{KL}(P \| Q)
\end{aligned}$$

  where $D_{KL}(P \| Q)$ is the Kullback-Leibler (KL) divergence.

Regret bound for PCC

- By using the chain rule of probability we can rewrite the regret to the following form:

$$\begin{aligned}
\mathrm{Reg}_{\log}(f \,|\, \boldsymbol{x}) &= \sum_{\boldsymbol{y} \in \mathcal{Y}} P(\boldsymbol{y} \,|\, \boldsymbol{x}) \sum_{i=1}^{m} \left(\log P_i(\boldsymbol{y}) - \log Q_i(\boldsymbol{y})\right) \\
&= \sum_{i=1}^{m} \sum_{\boldsymbol{y} \in \mathcal{Y}} \left(\log P_i(\boldsymbol{y}) - \log Q_i(\boldsymbol{y})\right) P(\boldsymbol{y} \,|\, \boldsymbol{x}) \\
&= \sum_{i=1}^{m} \sum_{(y_1, \ldots, y_i)} \left(\log P_i(\boldsymbol{y}) - \log Q_i(\boldsymbol{y})\right) P(y_1, \ldots, y_i \,|\, \boldsymbol{x}) \,,
\end{aligned}$$

  where we use the notation $P_i(\boldsymbol{y}) = P(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1})$, similarly as for Q.

Regret bound for PCC

- The last sum can be further reformulated to the following form:

$$\sum_{(y_1,\ldots,y_i)} \left(\log P_i(\boldsymbol{y}) - \log Q_i(\boldsymbol{y})\right) P(y_1,\ldots,y_i \,|\, \boldsymbol{x}) =$$
$$\sum_{(y_1,\ldots,y_{i-1})} P(y_1,\ldots,y_{i-1} \,|\, \boldsymbol{x}) \sum_{y_i} \left(\log P_i(\boldsymbol{y}) - \log Q_i(\boldsymbol{y})\right) P_i(\boldsymbol{y})\,.$$

- In turn, the last sum in this equation is nothing else than the regret or the Kullback-Leibler (KL) divergence $D_{KL}(P_i(\boldsymbol{y}) \| Q_i(\boldsymbol{y}))$ of the binary problem associated with node $(root, y_1, \ldots, y_{i-1})$ of the tree.

Regret bound for PCC

- Further, we can see that:

$$
\begin{aligned}
\mathrm{Reg}_{\log}(f \mid \boldsymbol{x}) &= \sum_{i=1}^{m} \sum_{(y_1,\ldots,y_{i-1})} P(y_1,\ldots,y_{i-1}) D_{KL}(P_i(\boldsymbol{y}) \| Q_i(\boldsymbol{y})) \\
&= \sum_{i=1}^{m} \mathbb{E}_{y_1,\ldots,y_{i-1}} \left[D_{KL}(P_i(\boldsymbol{y}) \| Q_i(\boldsymbol{y})) \right] \\
&= m\mathbb{E}_{\boldsymbol{y}} \left[\frac{1}{m} \sum_{i=1}^{m} D_{KL}\left(P_i(\boldsymbol{y}) \| Q_i(\boldsymbol{y})\right) \right] \\
&= m\overline{\mathrm{Reg}_{\log}}(f \mid \boldsymbol{x}) \,,
\end{aligned}
$$

where $\overline{\mathrm{Reg}_{\log}}(f \mid \boldsymbol{x})$ expresses the expected regret in the no-leaf nodes of the tree.

Regret bound for PCC

- **Theorem**: For all distributions and all PCCs trained with logistic regression f and used with the ϵ-approximate inference algorithm,

$$\mathrm{Reg}_{0/1}(\mathrm{PCC}_\epsilon(f)) \leq \sqrt{2m\overline{\mathrm{Reg}_{\log}}(f)} + \epsilon$$

Regret bound for PCC

- **Proof**:
  - The conditional regret of $\mathrm{PCC}_\epsilon(f)$ predicting $\hat{\boldsymbol{y}}_\epsilon$ is:

$$\mathrm{Reg}_{0/1}(\mathrm{PCC}_\epsilon(f) \,|\, \boldsymbol{x}) = P(\boldsymbol{y}^* \,|\, \boldsymbol{x}) - P(\hat{\boldsymbol{y}}_\epsilon \,|\, \boldsymbol{x}).$$

  - Let $Q(\boldsymbol{y} \,|\, \boldsymbol{x})$ be the estimate given by f.
  - If $\boldsymbol{y}^* = \hat{\boldsymbol{y}}$, then $\mathrm{Reg}_{0/1}(f \,|\, \boldsymbol{x}) = 0$. Otherwise, we have:

$$Q(\hat{\boldsymbol{y}}_\epsilon \,|\, \boldsymbol{x}) + \epsilon - Q(\boldsymbol{y}^* \,|\, \boldsymbol{x}) \geq 0\,.$$

  - Adding $\mathrm{Reg}_{0/1}(\mathrm{PCC}_\epsilon(f) \,|\, \boldsymbol{x})$ to both sides we get

$$\begin{aligned}
&\mathrm{Reg}_{0/1}(\mathrm{PCC}_\epsilon(f) \,|\, \boldsymbol{x}) \\
&\qquad \leq (P(\boldsymbol{y}^* \,|\, \boldsymbol{x}) - Q(\boldsymbol{y}^* \,|\, \boldsymbol{x})) + (Q(\hat{\boldsymbol{y}}_\epsilon \,|\, \boldsymbol{x}) + \epsilon - P(\hat{\boldsymbol{y}}_\epsilon \,|\, \boldsymbol{x})) \\
&\qquad \leq |P(\boldsymbol{y}^* \,|\, \boldsymbol{x}) - Q(\boldsymbol{y}^* \,|\, \boldsymbol{x})| + |P(\hat{\boldsymbol{y}}_\epsilon \,|\, \boldsymbol{x}) - Q(\hat{\boldsymbol{y}}_\epsilon \,|\, \boldsymbol{x})| + \epsilon\,.
\end{aligned}$$

  - We can now make use of Pinsker's inequality:

$$\frac{1}{2} \sum_{\boldsymbol{y} \in \mathcal{Y}} \big|P(\boldsymbol{y}) - Q(\boldsymbol{y})\big| \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(P \| Q)}$$

Regret bound for PCC

- **Proof**:
  - Since the KL divergence $D_{\mathrm{KL}}(P\|Q)$ is closely related to the regret of the log loss, we have:

$$|P(\boldsymbol{y}\,|\,\boldsymbol{x}) - Q(\boldsymbol{y}\,|\,\boldsymbol{x})| + |P(\hat{\boldsymbol{y}}_\epsilon\,|\,\boldsymbol{x}) - Q(\hat{\boldsymbol{y}}_\epsilon\,|\,\boldsymbol{x})| \leq \sqrt{2\mathrm{Reg}_{\log}(f\,|\,\boldsymbol{x})}\,.$$

  - Getting the results together:

$$\begin{aligned}\mathrm{Reg}_{0/1}(\mathrm{PCC}_\epsilon(f)\,|\,\boldsymbol{x}) &\leq \epsilon + \sqrt{2\mathrm{Reg}_{\log}(f\,|\,\boldsymbol{x})} \\ &= \epsilon + \sqrt{2m\overline{\mathrm{Reg}_{\log}}(f\,|\,\boldsymbol{x})}\,.\end{aligned}$$

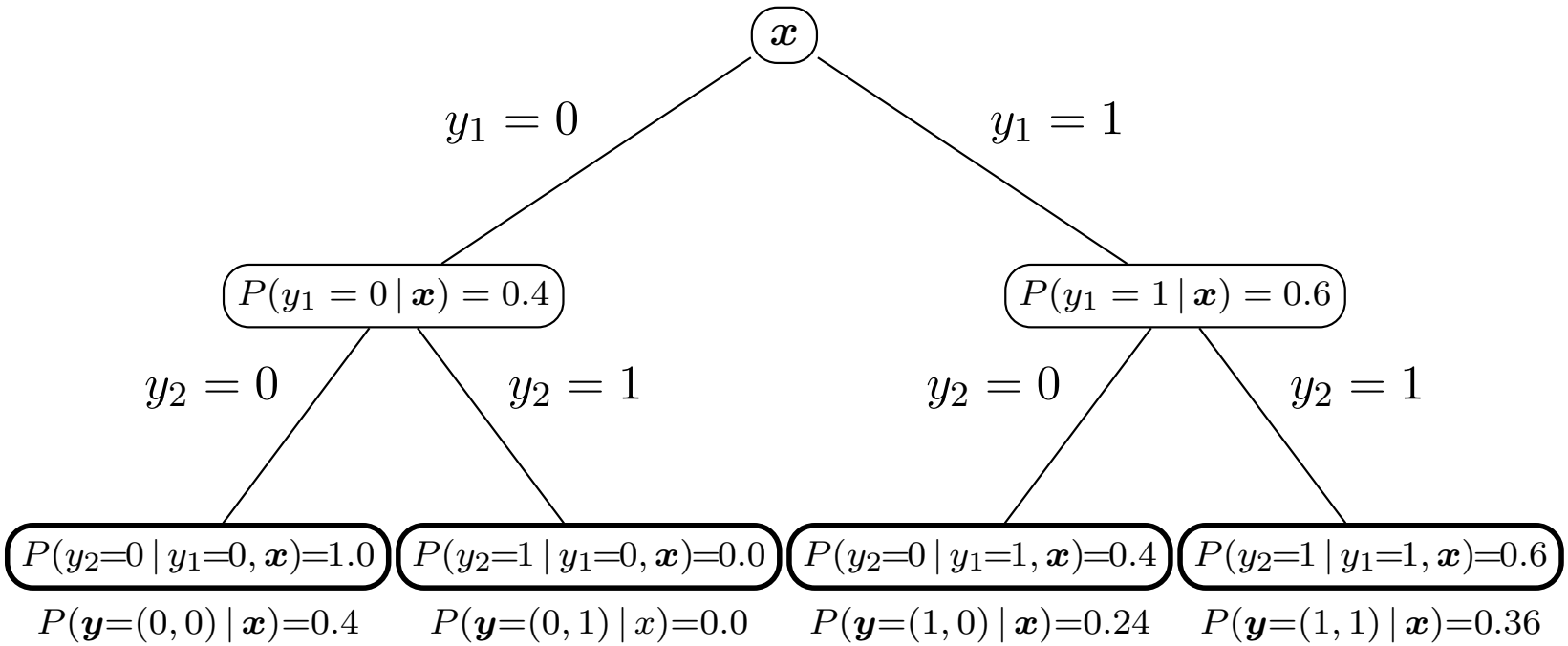
  - Finally, we prove the theorem by taking the expectation with respect to $\boldsymbol{x}$ on both sides, and applying Jensen's inequality $\mathbb{E}[f(A)] \leq f(\mathbb{E}[A])$ for the concave function $f(a) = \sqrt{a}$.

PCC for other losses

- **Exhaustive search**:
  - Compute the entire distribution $Q(\boldsymbol{y} \mid \boldsymbol{x})$ by traversing the probability tree.
  - Use an appropriate inference for a given loss ℓ on the estimated joint distribution:
$$\hat{y} = \arg\max_{\boldsymbol{h} \in \mathcal{Y}} \sum_{y \in \mathcal{Y}} Q(\boldsymbol{y} \mid \boldsymbol{x}) \ell(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x}))$$
  - This approach is extremely costly.
- **Ancestral sampling**:
  - Sampling can be easily performed by using the probability tree.
  - Make inference based on the empirical distribution.
  - Hamming loss: estimate marginal probabilities.
  - F-measure: estimate $P(\boldsymbol{y} = \mathbf{0} \mid \boldsymbol{x})$ and matrix Δ with elements
$$\Delta_{ik} = \sum_{\boldsymbol{y}: y_i = 1} \frac{2P(\boldsymbol{y})}{s_{\boldsymbol{y}} + k} \,.$$

Probabilistic classifier chains

- Exhaustive search and ancestral sampling:



- Sample: (1,1), (1,0), (0,0), (0,0), (1,1), (0,0), ...

Probabilistic classifier chains

Table : PCC vs. SSVMs on Hamming loss and PCC vs. BR on subset 0/1 loss.

Dataset	PCC	SSVM Best	PCC	BR
	Hamming loss		subset 0/1 loss	
Scene	0.104±.004	0.101±.003	0.385±.014	0.509±.014
Yeast	0.203±.005	0.202±.005	0.761±.014	0.842±.012
Synth1	0.067±.001	0.069±.001	0.239±.006	0.240±.006
Synth2	0.000±.000	0.058±.001	0.000±.000	0.832±.004
Reuters	0.060±.002	0.045±.001	0.598±.009	0.689±.008
Mediamill	0.172±.001	0.182±.001	0.885±.003	0.902±.003

Recurrent classifiers

- PCCs are similar to Maximum Entropy Markov Models (MEMMs)[5] introduced for sequence learning:
  - One logistic classifier that takes dependences up to the k-th degree.
  - Inference by dynamic programming.
- Searn[6] is another approach that is based on recurrent classifiers:
  - Linear inference.
  - Learning is performed in the iterative way to solve the egg and the chicken problem: output of the classifier is also used as input to the classifier.

[5] A. K. McCallum, D. Freitag, and F. (2000) Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML*, 2000

[6] H. Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning*, 75:297–325, 2009

Output search space

- More advanced search techniques.
- Popular topic in structured output prediction.
- Search techniques for different task losses.[7]

[7] J.R. Doppa, A. Fern, and P. Tadepalli. Structured prediction via output space search. *JMLR*, 15:1317–1350, 2014

PCC for multi-class classification

- PCC can be used for multi-class classification:
  - Map each class label to a label vector: binary coding, hierarchical clustering, ...
  - The same idea as in conditional probability trees (CPT).[8]
  - Label tree classifiers for efficient multi-class classification.[9]

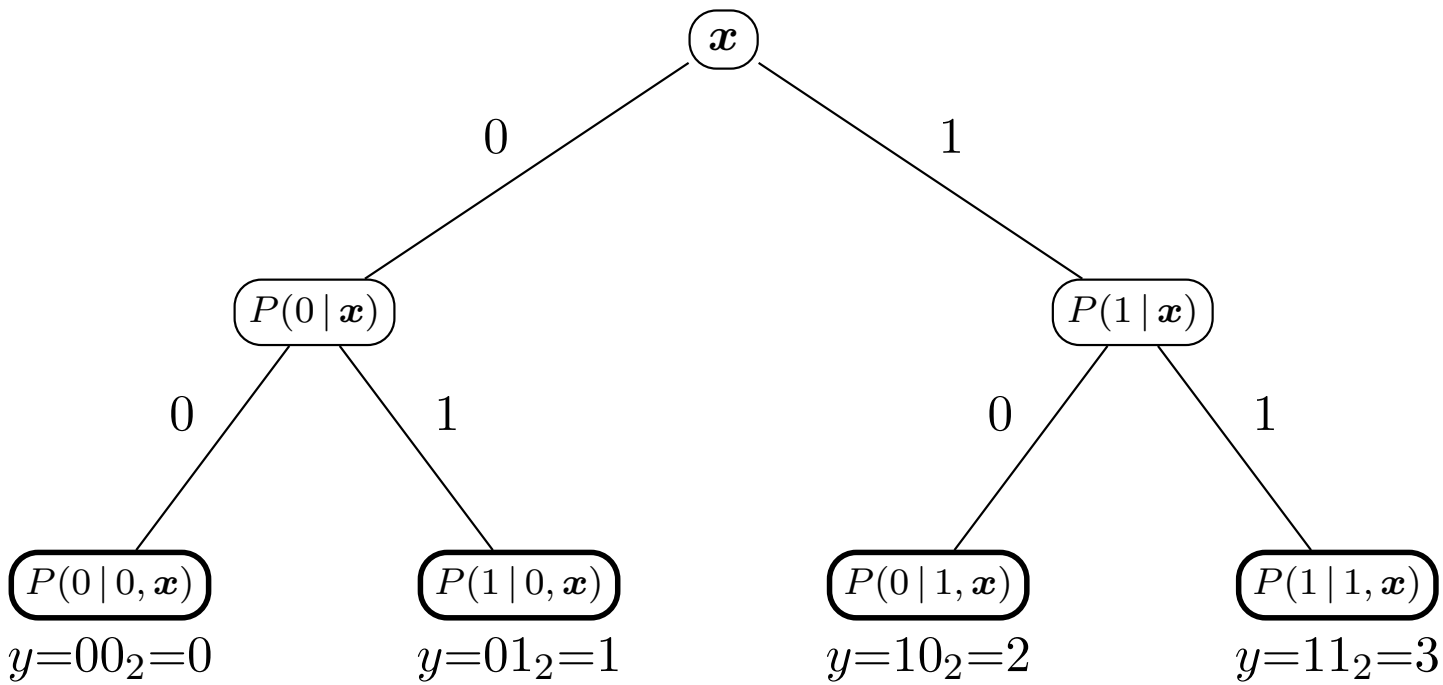
[8] A. Beygelzimer, J. Langford, Y. Lifshits, G. B. Sorkin, and A. L. Strehl. Conditional probability tree estimation analysis and algorithms. In *UAI*, pages 51–58, 2009

[9] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, pages 163–171. Curran Associates, Inc., 2010

J. Deng, S. Satheesh, A. C. Berg, and Fei Fei F. Li. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS*, pages 567–575. 2011

PCC for multi-class classification

- We assign each class an integer from 0 to $k-1$ and code it by its binary representation on m bits.
- Example: $k = 4$, $\mathcal{Y} = \{0, 1, 2, 3\}$.
- k leaves, one for each class.



Consistent and efficient label tree classifiers

- PCC: fast learning but inference can be costly.
- Greedy search is the most efficient, but is not consistent.
- How to ensure a linear inference in m for any loss?

Outline

Filter trees

- Filter trees (FT)[10] have been originally introduced for cost-sensitive multi-class classification, but can be easily adapted to multi-label classification.

[10] A. Beygelzimer, J. Langford, and P. D. Ravikumar. Error-correcting tournaments. In *ALT*, pages 247–262, 2009

Filter trees

- Filter trees (FT)[10] have been originally introduced for cost-sensitive multi-class classification, but can be easily adapted to multi-label classification.
- They use a **bottom-up** learning algorithm to train the label tree.
- Based on a single **elimination tournament** on the set of classes/label combinations.

[10] A. Beygelzimer, J. Langford, and P. D. Ravikumar. Error-correcting tournaments. In *ALT*, pages 247–262, 2009

Filter trees

- FT are trained to predict y_{i+1} based on previous labels.
- FT implicitly transforms the underlying distribution P over multi-class/multi-label examples into a **specific distribution** P^{FT} over weighted binary examples.
- The inference procedure of FT is straight-forward and uses the **greedy search**.
- FT are **consistent** for any cost function.

Filter trees

- Filter tree training:

Input: training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, importance-weighted binary learner $Learn$
for each non-leaf node $\boldsymbol{v} = (root, y_1, \ldots, y_{i-1})$ in the order from leaves to root **do**
  $S_{\boldsymbol{v}} = \emptyset$
  for each traning example $(\boldsymbol{x}, \boldsymbol{y})$ **do**
    Let $\boldsymbol{y}_l$ and $\boldsymbol{y}_r$ be the two label vectors input to $\boldsymbol{v}$
    $y_i \leftarrow \arg\min_{l,r}\{\ell(\boldsymbol{y}, \boldsymbol{y}_l), \ell(\boldsymbol{y}, \boldsymbol{y}_r)\}$
    $w = |\ell(\boldsymbol{y}, \boldsymbol{y}_l) - \ell(\boldsymbol{y}, \boldsymbol{y}_r)|$
    $S_{\boldsymbol{v}} \leftarrow S_{\boldsymbol{v}} \cup (\boldsymbol{x}, y_i, w)$
  end for
  $f_{\boldsymbol{v}} = Learn(S_{\boldsymbol{v}})$
end for
return $f = \{f_{\boldsymbol{v}}\}$

Filter trees

- Different training schemes possible:
  - Train a classifier in each node,
  - Train a classifier on each level,
  - Train one global binary classifier (in several loops).
- The tree in multi-label classification is given naturally, but the order of labels may influence the performance.
- In general case, training can be costly ($O(2^m)$), but efficient variants for multi-label classification exist.[11]
- Prediction is always linear in the number of labels ($O(m)$).

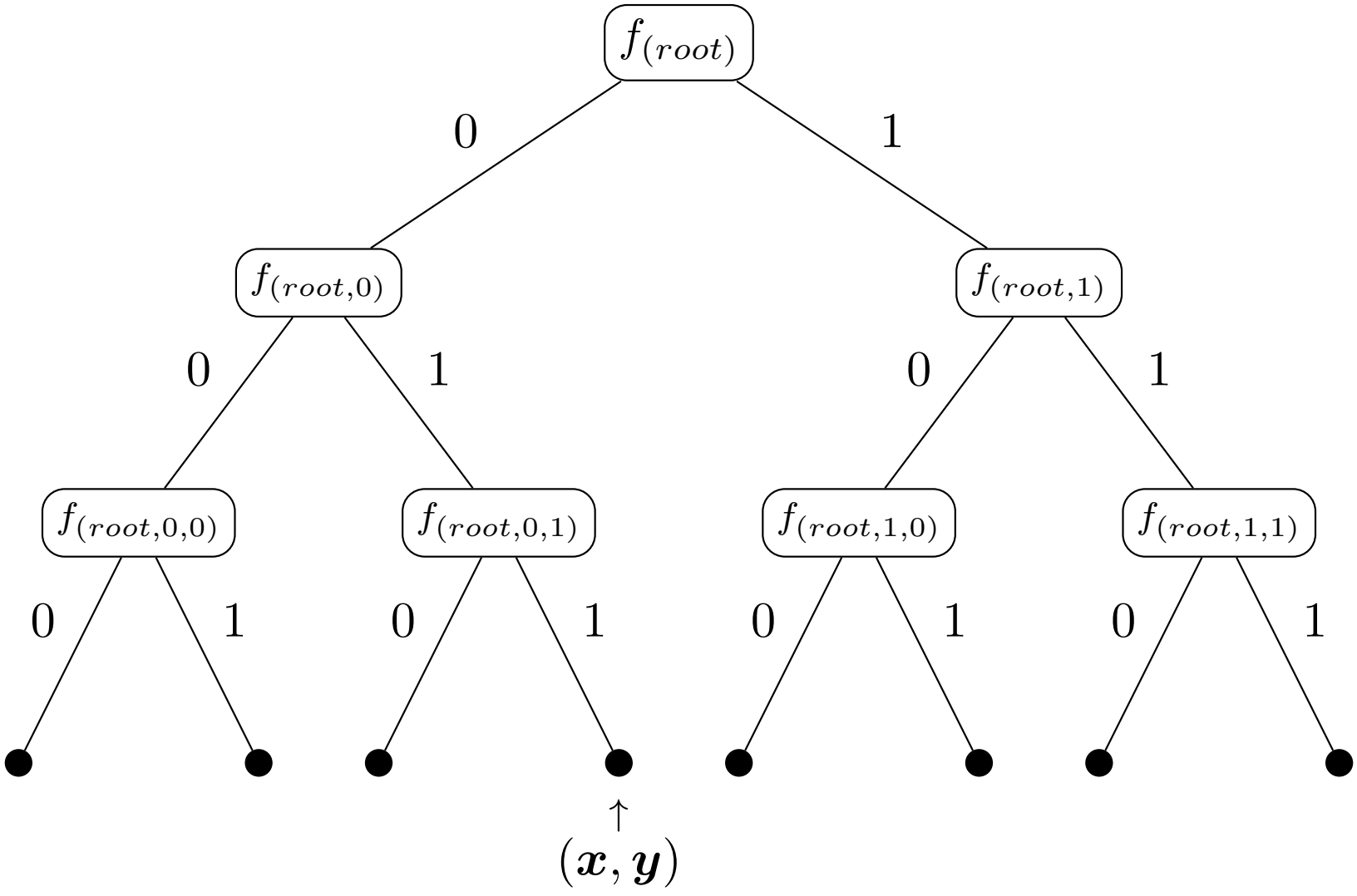
[11] Chun-Liang Li and Hsuan-Tien Lin. Condensed filter tree for cost-sensitive multi-label classification. In *ICML*, pages 423–431, 2014

Filter trees

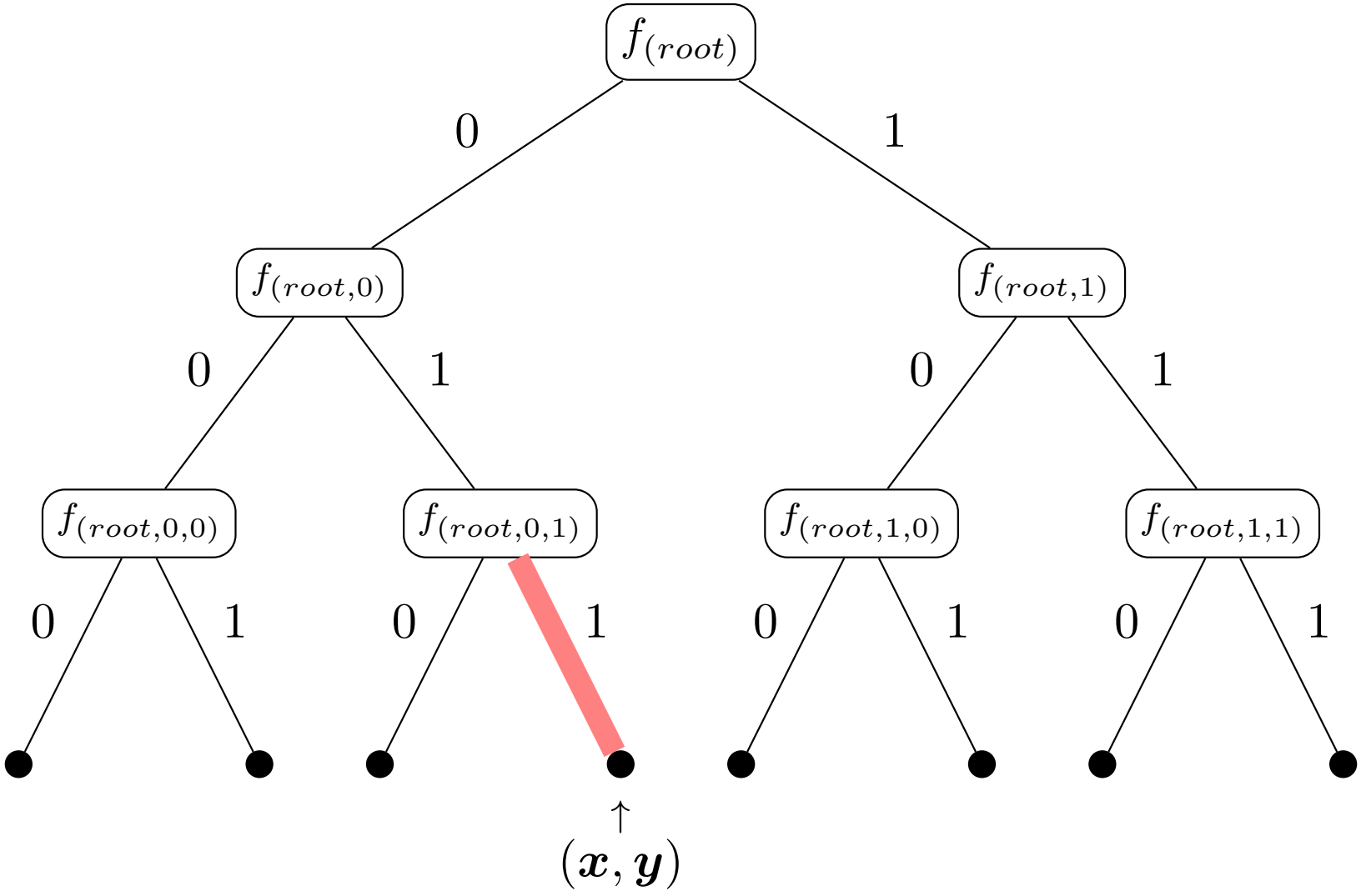
- Filter trees for the subset 0/1 loss **filter out** all examples that are misclassified by the lower-level classifiers.
- Therefore, training in this case is also linear in the number of labels ($O(m)$).
- $f_{(root,y_1,\ldots y_i)}(\boldsymbol{x})$ predicts y_{i+1} given that all classifiers below predict the subsequent labels correctly:

$$f_{(root,y_1,\ldots y_i)} : \boldsymbol{x} \mapsto (y_{i+1} \,|\, y_{j+1} = f_{(root,y_1,\ldots y_j)} : j = i+1, \ldots, m-1)$$

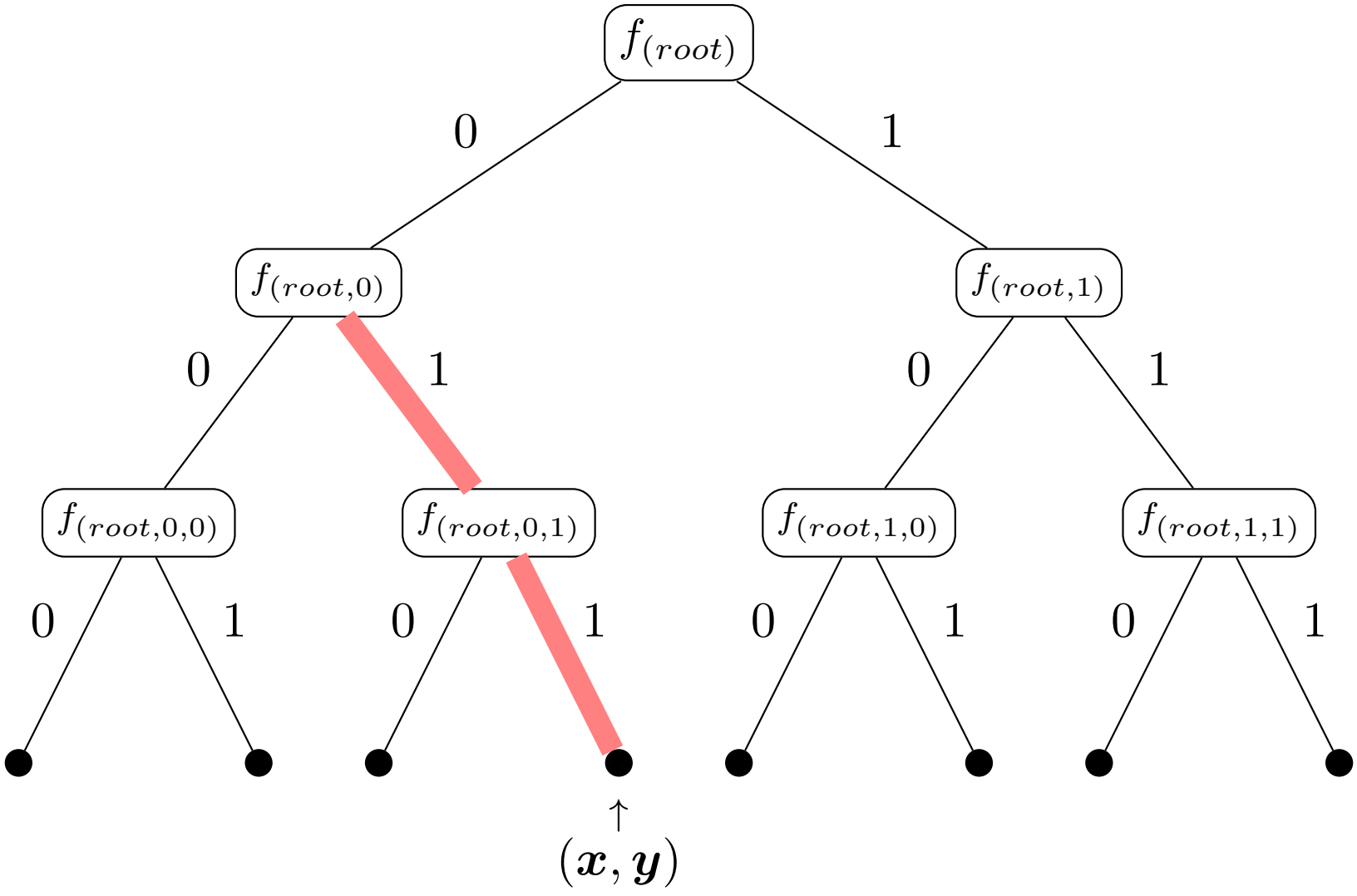
Filter trees : Example



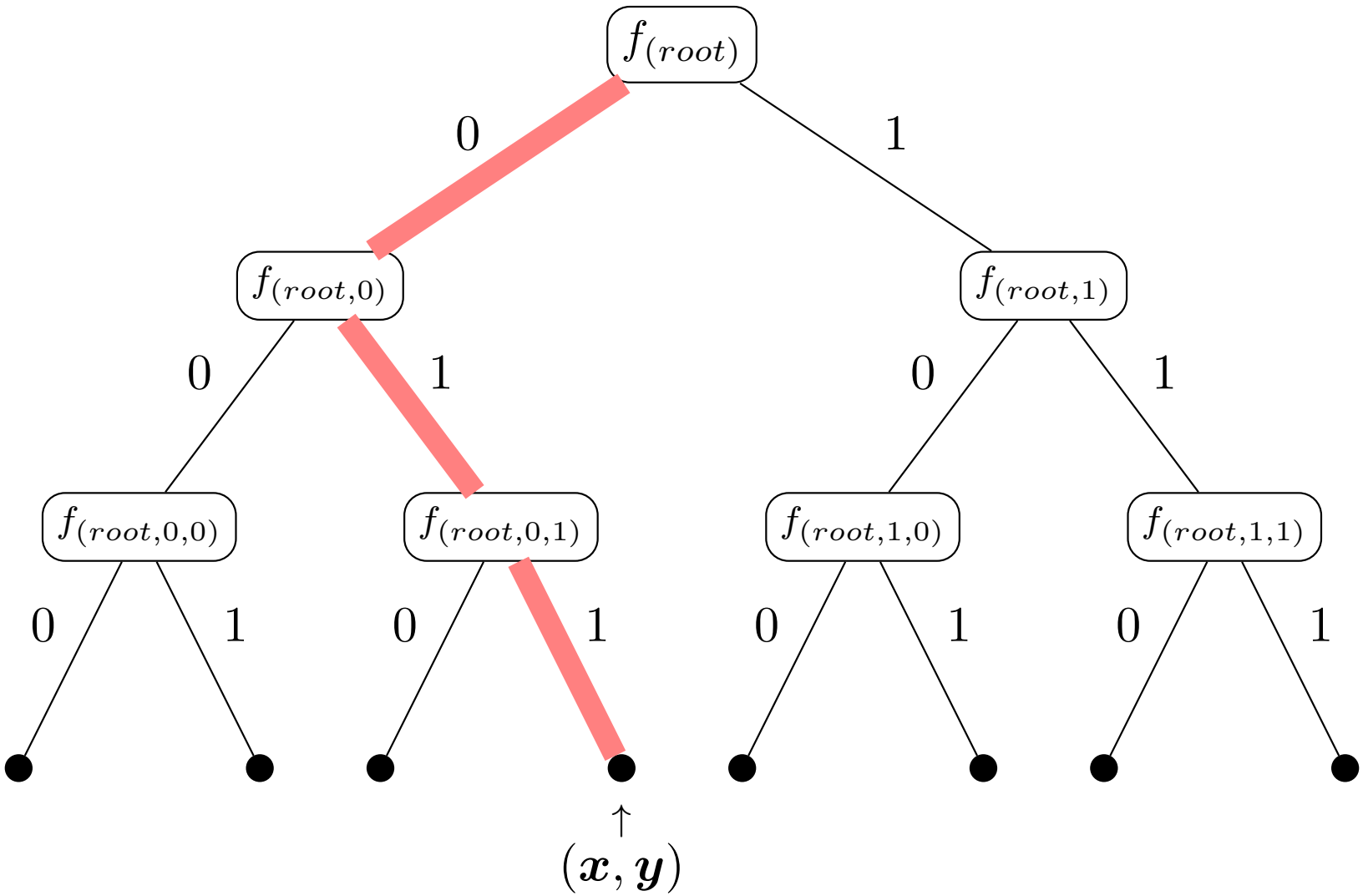
Filter trees : Example



Filter trees : Example

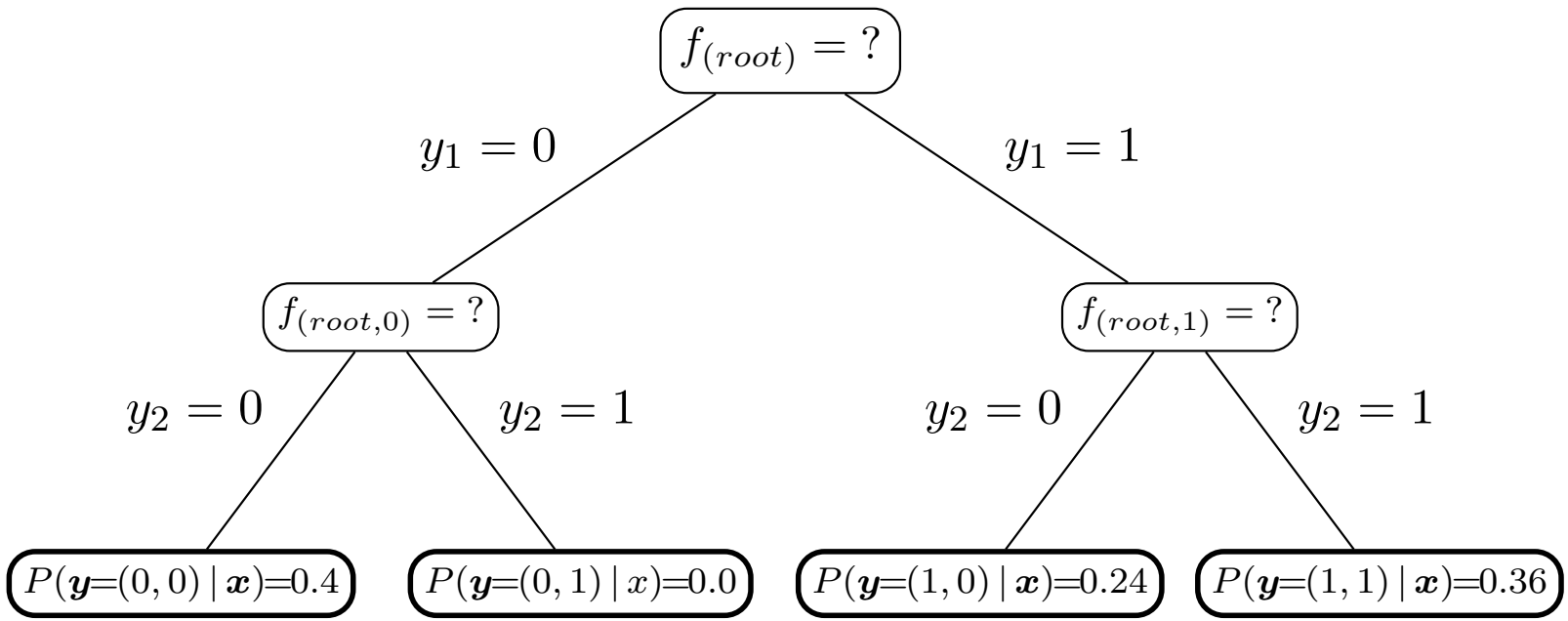


Filter trees : Example



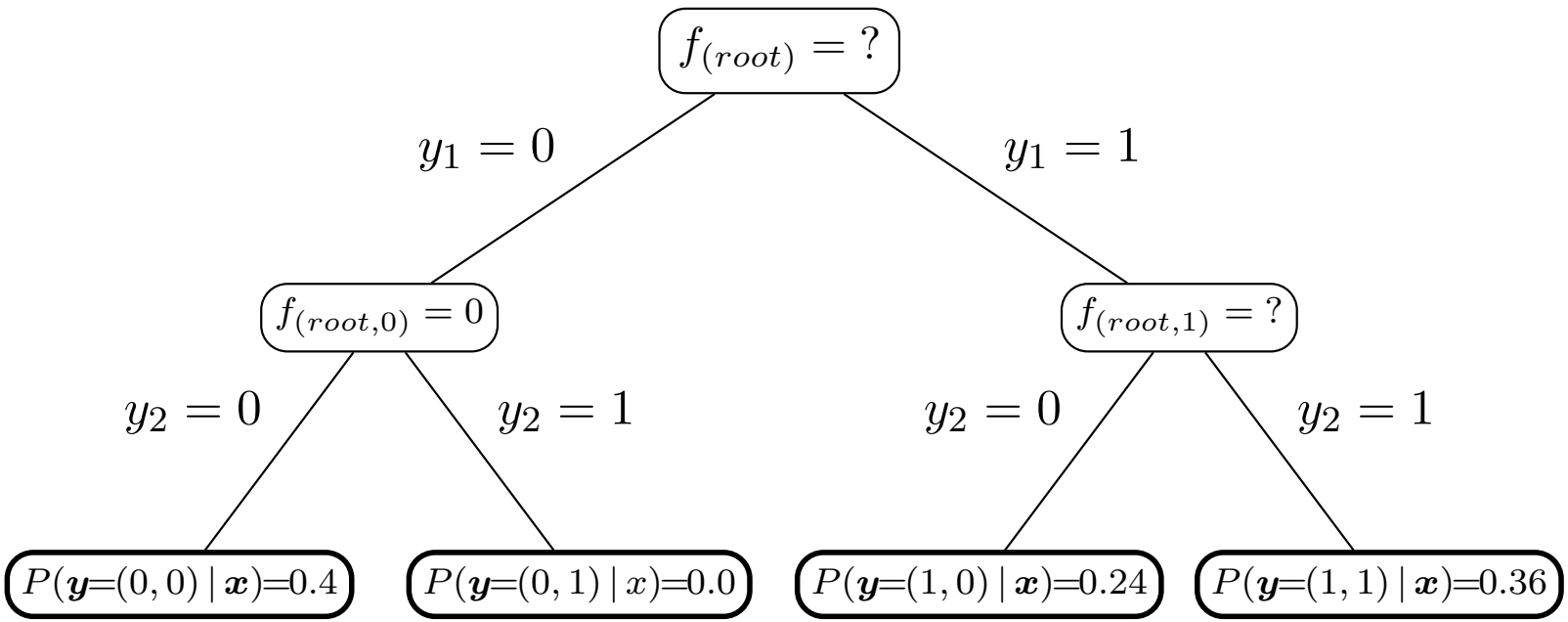
Filter trees: Consistency

- Consistency of FT for a single $\boldsymbol{x}$:



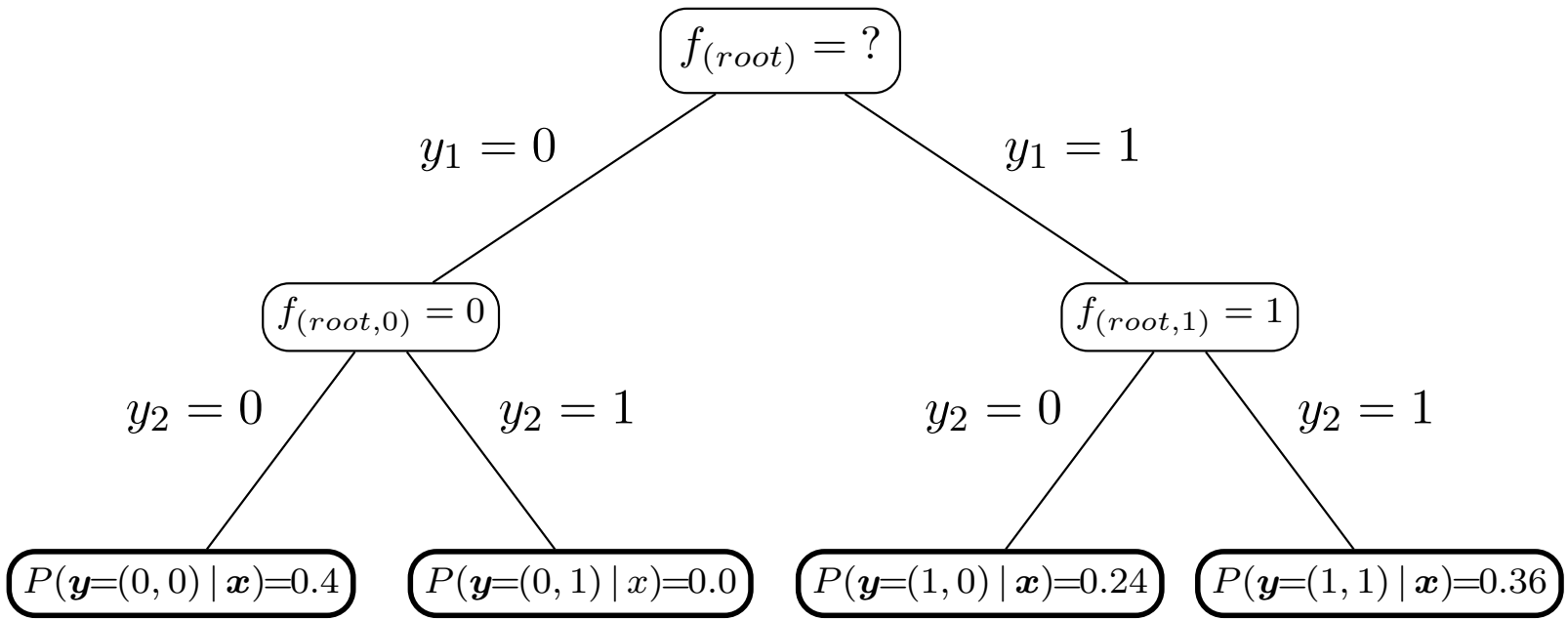
Filter trees: Consistency

- Consistency of FT for a single $\boldsymbol{x}$:



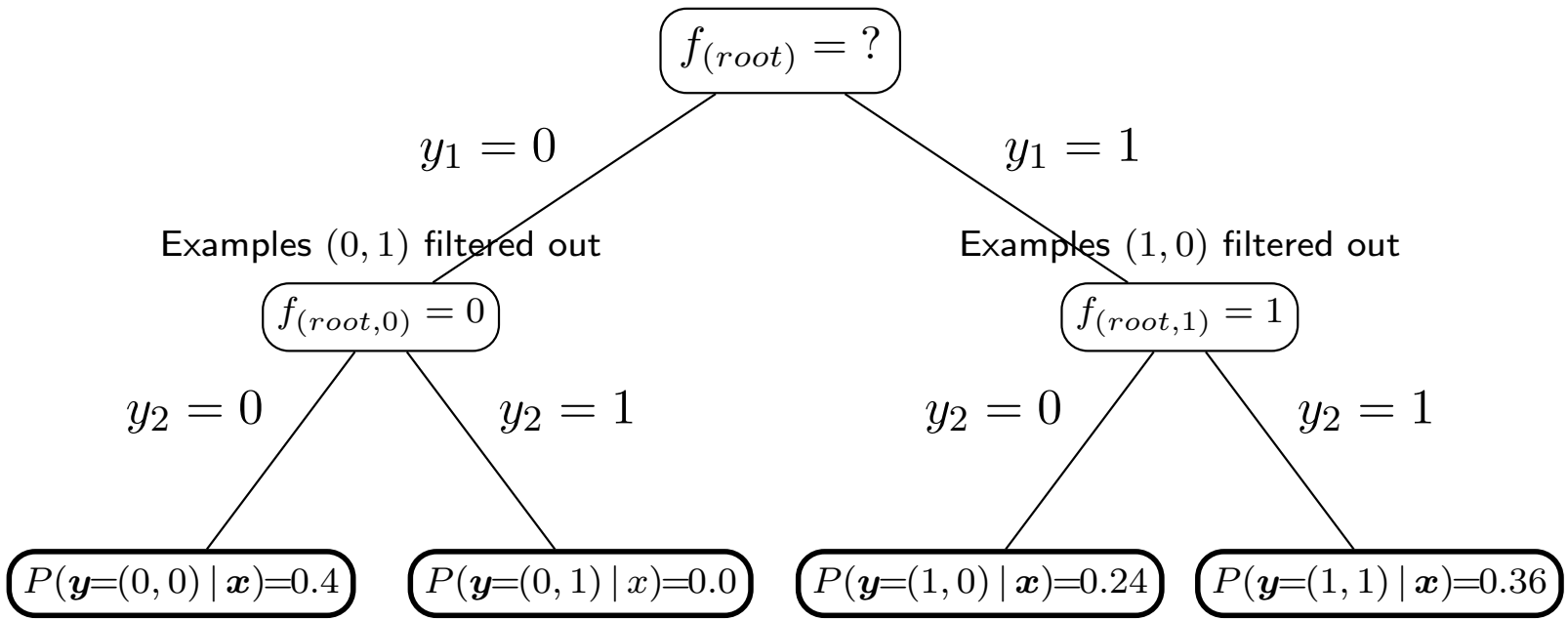
Filter trees: Consistency

- Consistency of FT for a single $\boldsymbol{x}$:



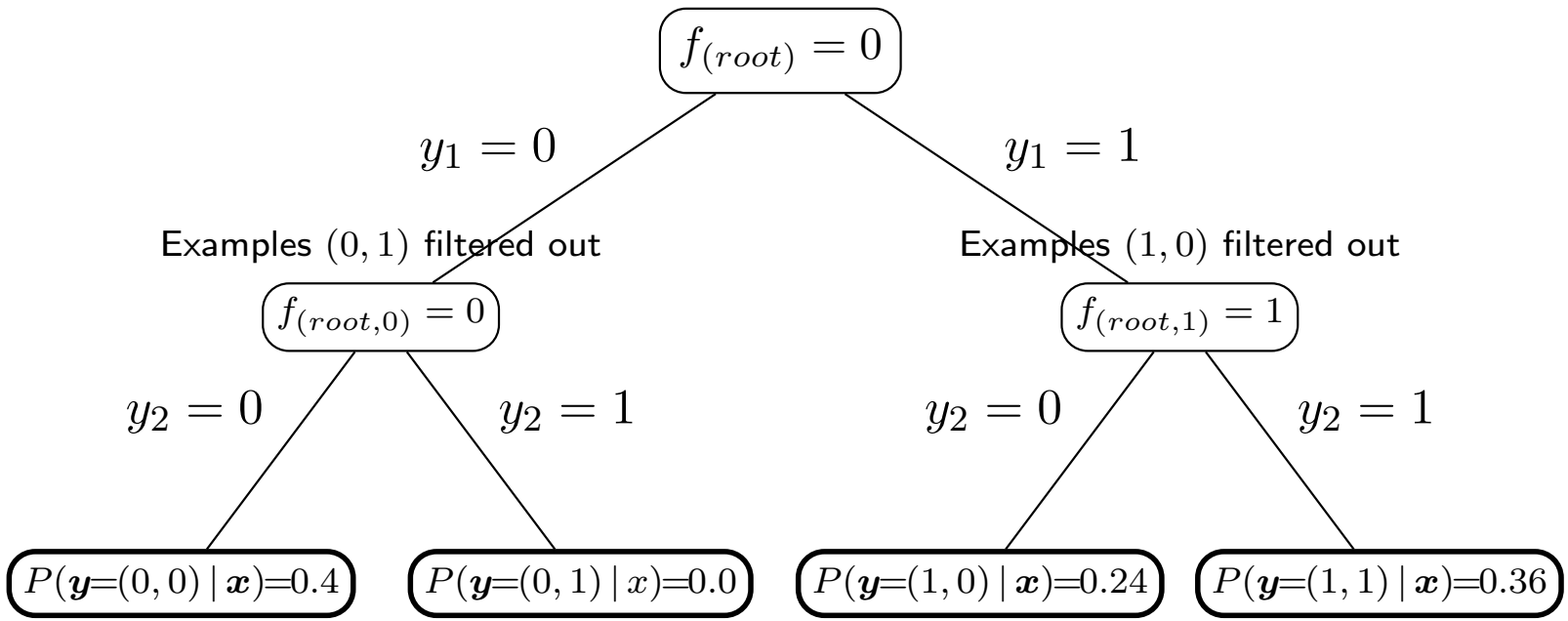
Filter trees: Consistency

- Consistency of FT for a single $\boldsymbol{x}$:



Filter trees: Consistency

- Consistency of FT for a single $\boldsymbol{x}$:



Regret bound for filter trees

- Let $f_{\boldsymbol{v}}$ be a classifier for the binary classification problem induced at node $\boldsymbol{v}$.
- The average binary regret is defined as:

$$\overline{\mathrm{Reg}}_{0/1}(f, P^{\mathrm{FT}}) = \frac{1}{\sum_{\boldsymbol{v}} W_{\boldsymbol{v}}} \sum_{\boldsymbol{v}} \mathrm{Reg}_{0/1}(f_{\boldsymbol{v}}, P^{\mathrm{FT}}_{\boldsymbol{v}}) W_{\boldsymbol{v}},$$

  where

$$W_{\boldsymbol{v}} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})} w_{\boldsymbol{v}}(\boldsymbol{x}, \boldsymbol{y}).$$

- **Theorem**:[12] For all distributions and all FT classifiers trained with a binary classifier f, and any cost-matrix-based task loss ℓ,

$$\mathrm{Reg}_{\ell}(\mathrm{FT}(f)) \leq \overline{\mathrm{Reg}}_{0/1}(f, P^{\mathrm{FT}}) \sum_{\boldsymbol{v}} W_{\boldsymbol{v}}\,.$$

[12] A. Beygelzimer, J. Langford, and P. D. Ravikumar. Error-correcting tournaments. In *ALT*, pages 247–262, 2009

Regret bound for filter trees

- For subset 0/1 loss, we have

$$\sum_{\boldsymbol{v}} w_{\boldsymbol{v}}(\boldsymbol{x}, \boldsymbol{y}) \leq m \,,$$

  since each training example $(\boldsymbol{x}, \boldsymbol{y})$ will appear in training at most once per level with importance weight 1.
- The regret bound has then the form:

$$\mathrm{Reg}_{\ell}(\mathrm{FT}(f)) \leq m \overline{\mathrm{Reg}}_{0/1}(f, P^{\mathrm{FT}}) \,.$$

Outline

Plug-in rule classifiers

- The general idea:
  - For a given task loss ℓ find its Bayes classifier.
  - Estimate all required parameters.
  - Plug the estimates into the Bayes classifier.
  - Perform inference.
- We will use this idea for the F-measure.

Plug-in rule classifiers for the F-measure

- Label independence:

[13] N. Ye, K. Chai, W. Lee, and H. Chieu. Optimizing F-measures: a tale of two approaches. In *ICML*, 2012

Plug-in rule classifiers for the F-measure

- Label independence:
  - We need only to estimate marginal probabilities $\eta_i = P(y_i = 1 \,|\, \boldsymbol{x})$.

[13] N. Ye, K. Chai, W. Lee, and H. Chieu. Optimizing F-measures: a tale of two approaches. In *ICML*, 2012

Plug-in rule classifiers for the F-measure

- Label independence:
  - We need only to estimate marginal probabilities $\eta_i = P(y_i = 1 \,|\, \boldsymbol{x})$.
  - Reduction to binary probability estimation methods (BR with probabilistic classifiers) can be used to this end.

[13] N. Ye, K. Chai, W. Lee, and H. Chieu. Optimizing F-measures: a tale of two approaches. In *ICML*, 2012

Plug-in rule classifiers for the F-measure

- Label independence:
  - We need only to estimate marginal probabilities $\eta_i = P(y_i = 1 \,|\, \boldsymbol{x})$.
  - Reduction to binary probability estimation methods (BR with probabilistic classifiers) can be used to this end.
  - Apply the dynamic programming inference[13] on the estimates.

[13] N. Ye, K. Chai, W. Lee, and H. Chieu. Optimizing F-measures: a tale of two approaches. In *ICML*, 2012

Plug-in rule classifiers for the F-measure

- Label independence:
  - We need only to estimate marginal probabilities $\eta_i = P(y_i = 1 \,|\, \boldsymbol{x})$.
  - Reduction to binary probability estimation methods (BR with probabilistic classifiers) can be used to this end.
  - Apply the dynamic programming inference[13] on the estimates.
  - This approach is referred to as LFP.

[13] N. Ye, K. Chai, W. Lee, and H. Chieu. Optimizing F-measures: a tale of two approaches. In *ICML*, 2012

Plug-in rule classifiers for the F-measure

- General case:

[14] K. Dembczynski, A. Jachnik, W. Kotlowski, W. Waegeman, and E. Hüllermeier. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *ICML*, 2013

Plug-in rule classifiers for the F-measure

- General case:
  - We need to estimate $P(\boldsymbol{y} = \mathbf{0} \,|\, \boldsymbol{x})$ and matrix P with elements

$$p_{is} = P(y_i = 1, s_{\boldsymbol{y}} = s), \quad i, s \in \{1, \ldots, m\}\,.$$

[14] K. Dembczynski, A. Jachnik, W. Kotlowski, W. Waegeman, and E. Hüllermeier. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *ICML*, 2013

Plug-in rule classifiers for the F-measure

- General case:
  - We need to estimate $P(\boldsymbol{y} = \mathbf{0} \,|\, \boldsymbol{x})$ and matrix P with elements

$$p_{is} = P(y_i = 1, s_{\boldsymbol{y}} = s), \quad i, s \in \{1, \ldots, m\}\,.$$

  - The matrix P can be estimated by using a simple reduction to m multinomial regression problems with at most $m + 1$ classes.

[14] K. Dembczynski, A. Jachnik, W. Kotlowski, W. Waegeman, and E. Hüllermeier. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *ICML*, 2013

Plug-in rule classifiers for the F-measure

- General case:
  - We need to estimate $P(\boldsymbol{y} = \mathbf{0} \,|\, \boldsymbol{x})$ and matrix P with elements

$$p_{is} = P(y_i = 1, s_{\boldsymbol{y}} = s), \quad i, s \in \{1, \ldots, m\}\,.$$

  - The matrix P can be estimated by using a simple reduction to m multinomial regression problems with at most $m + 1$ classes.
  - The scheme of the reduction for the i-th subproblem is the following:

$$(\boldsymbol{x}, \boldsymbol{y}) \to (\boldsymbol{x}, y = [\![y_i = 1]\!] \cdot s_{\boldsymbol{y}}), \quad y \in \{0, \ldots, m\}\,.$$

[14] K. Dembczynski, A. Jachnik, W. Kotlowski, W. Waegeman, and E. Hüllermeier. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *ICML*, 2013

Plug-in rule classifiers for the F-measure

- General case:
  - We need to estimate $P(\boldsymbol{y} = \mathbf{0} \,|\, \boldsymbol{x})$ and matrix P with elements

$$p_{is} = P(y_i = 1, s_{\boldsymbol{y}} = s), \quad i, s \in \{1, \ldots, m\}\,.$$

  - The matrix P can be estimated by using a simple reduction to m multinomial regression problems with at most $m + 1$ classes.
  - The scheme of the reduction for the i-th subproblem is the following:

$$(\boldsymbol{x}, \boldsymbol{y}) \to (\boldsymbol{x}, y = [\![y_i = 1]\!] \cdot s_{\boldsymbol{y}}), \quad y \in \{0, \ldots, m\}\,.$$

  - A similar reduction can be performed for estimating $P(\boldsymbol{y} = \mathbf{0} \,|\, \boldsymbol{x})$:

$$(\boldsymbol{x}, \boldsymbol{y}) \to (\boldsymbol{x}, y = [\![\boldsymbol{y} = \mathbf{0}]\!])\,.$$

[14] K. Dembczynski, A. Jachnik, W. Kotlowski, W. Waegeman, and E. Hüllermeier. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *ICML*, 2013

Plug-in rule classifiers for the F-measure

- General case:
  - We need to estimate $P(\boldsymbol{y} = \mathbf{0} \,|\, \boldsymbol{x})$ and matrix P with elements

$$p_{is} = P(y_i = 1, s_{\boldsymbol{y}} = s), \quad i, s \in \{1, \ldots, m\}\,.$$

  - The matrix P can be estimated by using a simple reduction to m multinomial regression problems with at most $m + 1$ classes.
  - The scheme of the reduction for the i-th subproblem is the following:

$$(\boldsymbol{x}, \boldsymbol{y}) \to (\boldsymbol{x}, y = [\![y_i = 1]\!] \cdot s_{\boldsymbol{y}}), \quad y \in \{0, \ldots, m\}\,.$$

  - A similar reduction can be performed for estimating $P(\boldsymbol{y} = \mathbf{0} \,|\, \boldsymbol{x})$:

$$(\boldsymbol{x}, \boldsymbol{y}) \to (\boldsymbol{x}, y = [\![\boldsymbol{y} = \mathbf{0}]\!])\,.$$

  - This approach, referred to as EFP, is **consistent**, but on finite training sets the estimate of matrix P may not correspond to any valid distribution.[14]

[14] K. Dembczynski, A. Jachnik, W. Kotlowski, W. Waegeman, and E. Hüllermeier. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *ICML*, 2013

SSVMs for F-measure-based loss

- SSVMs can be used to minimize F-measure-based loss.
- Rescale the margin by $\ell_F(\boldsymbol{y}, \boldsymbol{y}')$.
- Two algorithms:[15]

RML:
No label interactions:

$$f(\boldsymbol{y}, \boldsymbol{x}) = \sum_{i=1}^{m} f_i(y_i, \boldsymbol{x})$$

Quadratic learning and linear prediction

SML:
Submodular interactions:

$$f(\boldsymbol{y}, \boldsymbol{x}) = \sum_{i=1}^{m} f_i(y_i, \boldsymbol{x}) + \sum_{y_k, y_l} f_{k,l}(y_k, y_l)$$

More complex (graph-cut and approximate algorithms)

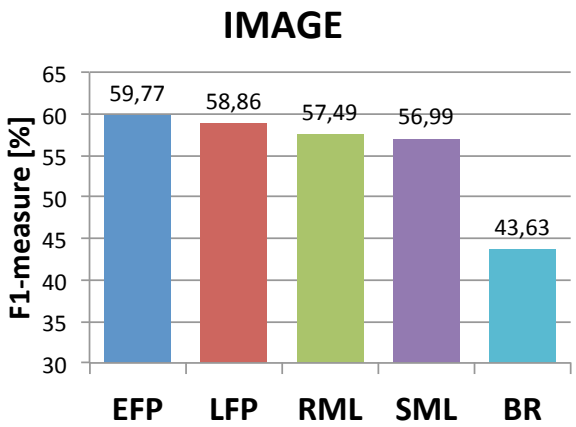
- Both are **inconsistent**.[16]

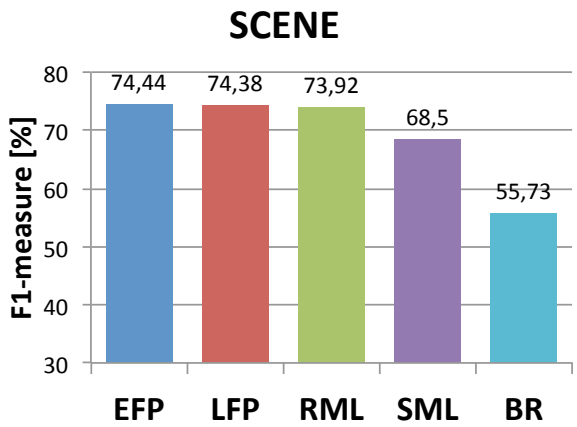
---

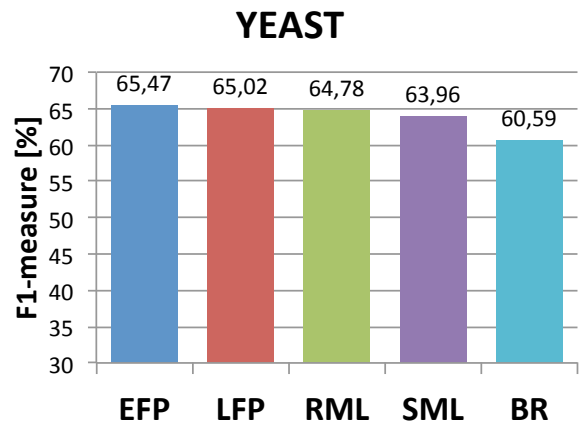
[15] J. Petterson and T. S. Caetano. Reverse multi-label learning. In *NIPS*, pages 1912–1920, 2010
J. Petterson and T. S. Caetano. Submodular multi-label learning. In *NIPS*, pages 1512–1520, 2011

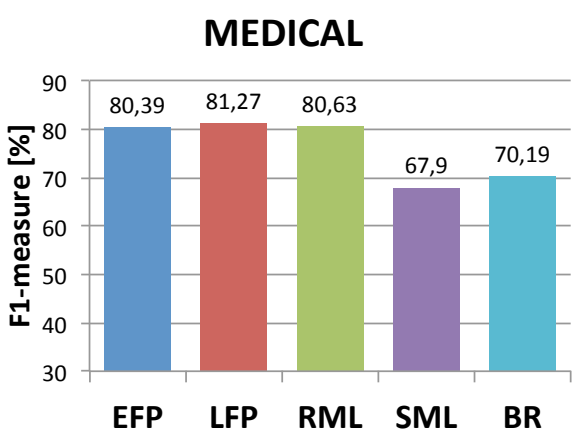
[16] K. Dembczynski, A. Jachnik, W. Kotlowski, W. Waegeman, and E. Hüllermeier. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *ICML*, 2013

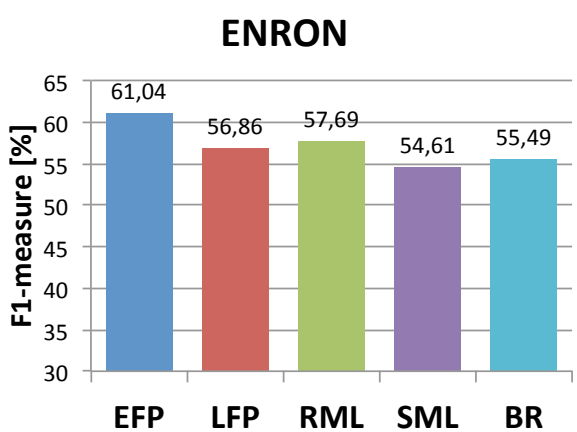
Experimental results

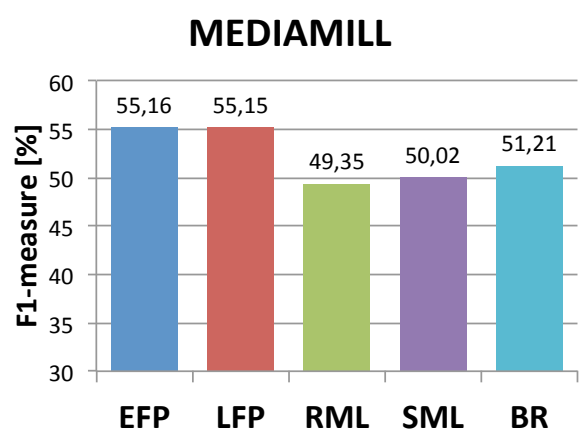












Outline

Binary relevance

- We already know that the simple BR can perform well for Hamming loss.
- It can also be useful in solving the problem of the F-measure maximization.
- We can also show a stronger result concerning the regret bound for Hamming loss:

$$\mathrm{Reg}_H(\mathrm{BR}(\boldsymbol{f})) \leq \sum_{i=1}^{m} \mathrm{Reg}_{0/1}(f_i) \leq \sum_{i=1}^{m} \psi^{-1}(\mathrm{Reg}_\ell(f_i))\,,$$

  where $\boldsymbol{f} = (f_1, \ldots, f_m)$ are prediction functions for corresponding labels, and ℓ is a surrogate loss for binary classification.
- Complexity of BR is $O(m)$.

Binary relevance

- Can we reduce the linear complexity and obtain good results with respect to Hamming loss?

Beyond binary relevance

- Several ideas exist:
  - Compressed sensing.[17]
  - Bloom filters.[18]
  - Principal label space transformation (PLST).[19]

[17] D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *NIPS*, 2009

[18] Moustapha Cissé, Nicolas Usunier, Thierry Artières, and Patrick Gallinari. Robust bloom filters for large multilabel classification tasks. In *NIPS*, pages 1851–1859, 2013

[19] Farbound Tai and Hsuan-Tien Lin. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542, 2012

PLST

- The general idea:
  - Perform PCA on label vectors.
  - Take k principal components.
  - Learn k regression functions.
  - Decode output of regression to the original space.

PLST

- PLST training:

1: **Input:** training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, parameter k, regression learning algorithm $Learn$
2: Let Y be the matrix of labels for all instances
3: $\boldsymbol{o} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{y}_i$
4: Let matrix Z consists of columns $\boldsymbol{y}_i - \boldsymbol{o}$.
5: Perform SVD on Z to obtain: $\mathrm{Z} = \mathrm{U}\Sigma\mathrm{V}^T$
6: Take $\mathrm{U}_k^T = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)^T$ that corresponds to k largest singular values.
7: **for** each training example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ **do**
8: Encode $\boldsymbol{p}_i = \mathrm{U}_k^T(\boldsymbol{y}_i - \boldsymbol{o})$
9: **end for**
10: **for** $j = 1$ to k **do**
11: $r_j = Learn(\{\boldsymbol{x}_i, p_{ij}\}_{i=1}^n)$
12: **end for**
13: **return** $\boldsymbol{r} = \{r_1, \ldots, r_k\}$

PLST

- PLST testing:

1: **Input:** test example $\boldsymbol{x}$, matrix U_k, regression models $\boldsymbol{r} = \{r_1, \ldots, r_k\}$
2: **for** $j = 1$ to k **do**
3: $\quad p_j = r_j(\boldsymbol{x})$
4: **end for**
5: $\tilde{\boldsymbol{y}} = \boldsymbol{o} + \sum_{j=1}^{k} p_j \boldsymbol{u}_j = \boldsymbol{o} + \mathrm{U}_k \boldsymbol{p}$
6: **return** $\tilde{\boldsymbol{y}} = \mathrm{round}(\tilde{\boldsymbol{y}})$

PLST

- It can be shown that PLST bounds the Hamming loss:

$$\ell_H(\boldsymbol{y}, \hat{\boldsymbol{y}}) \leq \frac{4}{m}(\|\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{p}\|^2 + \|\boldsymbol{y} - \boldsymbol{o} - \mathrm{U}_k\boldsymbol{p}\|^2)$$

- PLST speeds up BR, but also it may perform better.

Outline

Summary

- Reduction algorithms:
  - Probabilistic classifier chains,
  - Filter trees,
  - Plug-in rule classifiers,
  - Principal label space transformation,
  - ...

Open challenges

- Learning and inference algorithms for any task loss and output structure.
- Consistency of the algorithms.
- Large-scale datasets: number of instances, features, and labels.

Conclusions

- Take-away message:
  - Different reduction algorithms.
  - Consistent reduction for different task losses.
  - Efficiency in training and inference.
  - Reducing the label space for Hamming loss.
  - Extending the label space for non-decomposable losses.
- For more check:

`http://www.cs.put.poznan.pl/kdembczynski`

Thank you for your attention!

The project is co-financed by the European Union from resources of the European Social Found.

HUMAN CAPITAL
NATIONAL COHESION STRATEGY

EUROPEAN UNION
EUROPEAN
SOCIAL FUND