Efficient Label Tree Structures for Top-k Classification

Arkadiusz Jachnik and Krzysztof Dembczyński

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

IFORS 2014, Barcelona, July 14, 2014

Introduction

Introduction

- Large-scale machine learning problems:

Introduction

- Large-scale machine learning problems:
  - Large number of training examples (in hundreds of millions),

Introduction

- Large-scale machine learning problems:
  - Large number of training examples (in hundreds of millions),
  - Large number of features (in hundreds of thousands),

Introduction

- Large-scale machine learning problems:
  - Large number of training examples (in hundreds of millions),
  - Large number of features (in hundreds of thousands),
  - **Large number of class labels (in tens or hundreds of thousands)**.

Introduction

- Large-scale machine learning problems:
  - Large number of training examples (in hundreds of millions),
  - Large number of features (in hundreds of thousands),
  - **Large number of class labels (in tens or hundreds of thousands)**.
- A need for efficient algorithms and data structures.

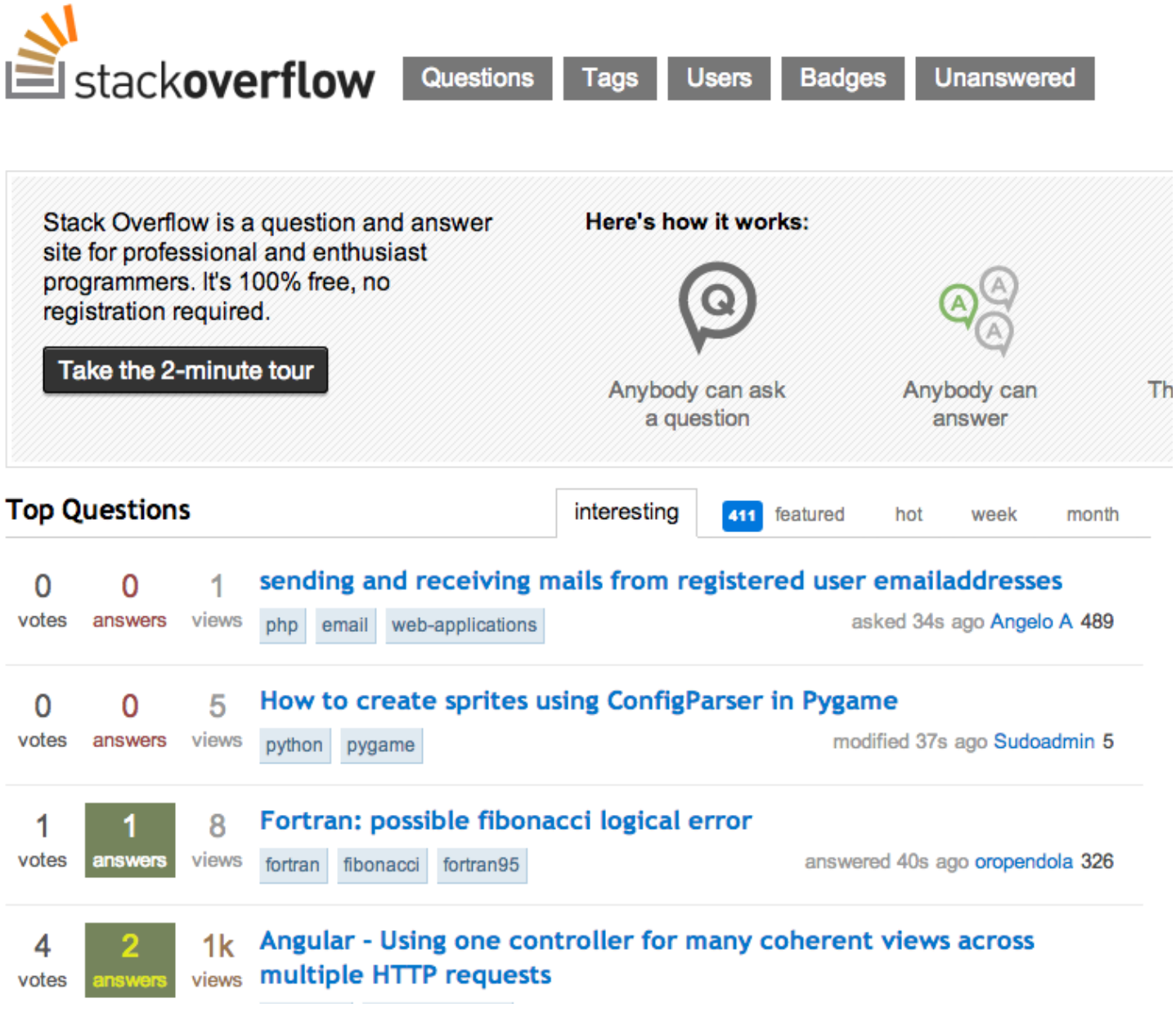
Introduction

- Large-scale machine learning problems:
  - Large number of training examples (in hundreds of millions),
  - Large number of features (in hundreds of thousands),
  - **Large number of class labels (in tens or hundreds of thousands)**.
- A need for efficient algorithms and data structures.
- **Label trees** for problems with a large number of class labels.

Introduction

- Large-scale machine learning problems:
  - Large number of training examples (in hundreds of millions),
  - Large number of features (in hundreds of thousands),
  - **Large number of class labels (in tens or hundreds of thousands)**.
- A need for efficient algorithms and data structures.
- **Label trees** for problems with a large number of class labels.
- Two settings: **multi-class** and multi-label classification.

Image annotation: cloud? sky? tree?



Document tagging

Supervised learning

- **Example** $\boldsymbol{x}$ is coming from an unknown input distribution $P(\boldsymbol{x})$.
- **True outcome** $y \in \mathcal{Y} = \{1, \ldots, m\}$ is generated from $P(y \,|\, \boldsymbol{x})$.
- **Predicted outcome** is given by $\hat{y} = h(\boldsymbol{x})$.
- The (**task**) **loss** of a single prediction is $\ell(y, \hat{y})$.

Supervised learning

- The overall goal is to minimize the **risk**:

$$L_\ell(h) = \mathbb{E}_{(\boldsymbol{x},y)}(\ell(y, h(\boldsymbol{x}))$$

- The optimal prediction function, the so-called **Bayes classifier**, is:

$$h^* = \arg\min_h L(h)$$

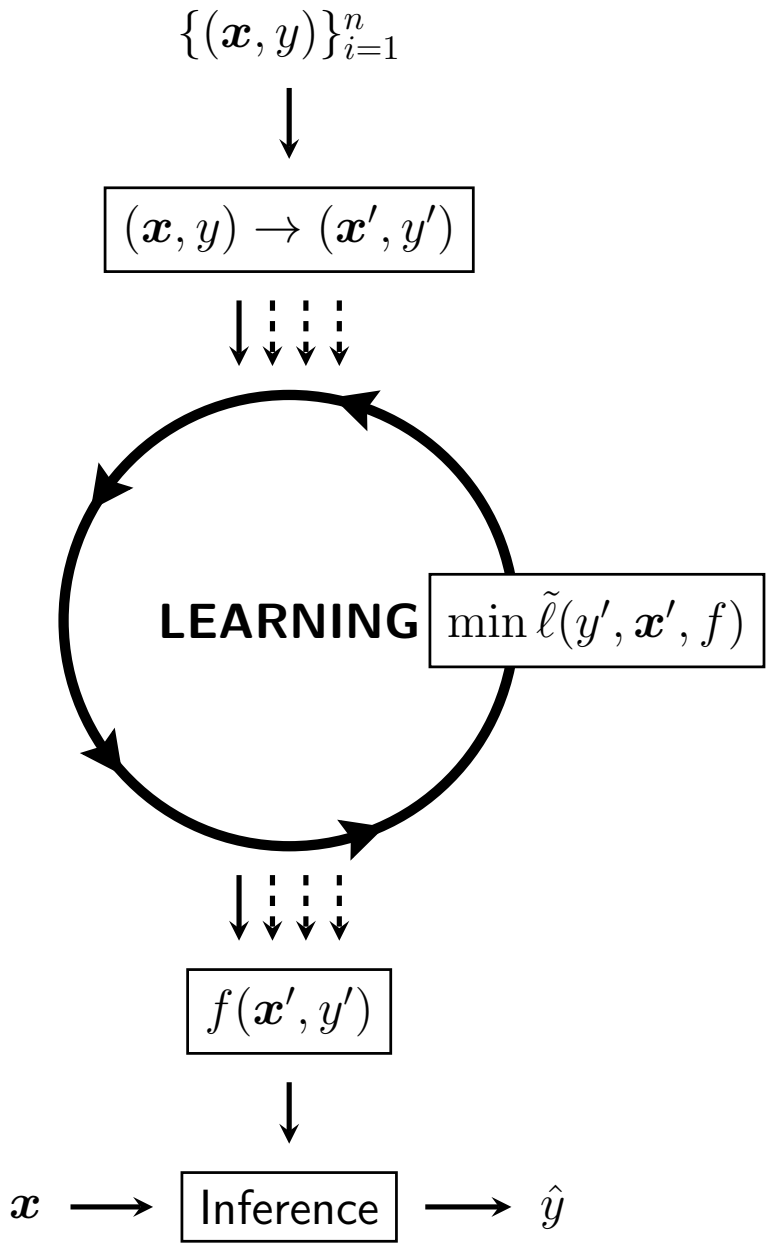
- The **regret** of a classifier h with respect to ℓ is defined as:

$$\mathrm{Reg}_\ell(h) = L_\ell(h) - L_\ell(h^*_\ell)$$

Supervised learning

- We use **training examples** $\{\boldsymbol{x}_i, y_i\}_1^n$ to find either:
  - A good approximation of h^*.
  - A scoring function $f(\boldsymbol{x}, y)$ being a good estimate of $P(y \mid \boldsymbol{x})$ (or a function of it).
- In the second case, we need to apply an **inference procedure** to approximate h^*.
- In general, training and/or inference are **hard** problems:
  - The loss function to be minimized is often neither convex nor differentiable.
- Two approaches to make the learning task easier: surrogate loss minimization and **reduction**.

Reduction



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

Statistical consistency

- We say that a surrogate (proxy) loss $\tilde{\ell}$ is **consistent** (**calibrated**) with the task loss ℓ when the following holds:

$$\mathrm{Reg}_{\tilde{\ell}}(h) \to 0 \Rightarrow \mathrm{Reg}_{\ell}(h) \to 0\,.$$

- The definition concerns both surrogate loss minimization and reduction algorithms:[1]
  - Surrogate loss minimization: $\tilde{\ell} =$ surrogate loss.
  - Reduction: $\tilde{\ell} =$ loss used in the reduced problem.

[1] P. Bartlett, M. Jordan, and J. McAuliffe. Convexity, classification and risk bounds. *JASA*, 101:138–156, 2006

A. Tewari and P.L. Bartlett. On the consistency of multiclass classification methods. *JMLR*, 8:1007–1025, 2007

A. Beygelzimer, J. Langford, and B. Zadrozny. Machine Learning Techniques—Reductions Between Prediction Quality Metrics. In *Performance Modeling and Engineering*, pages 3–28. Springer, 2008

Top-k classification

- Training data: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.
- **Predict** top k classes $\{y_j\}_{j=1}^k$ for a given $\boldsymbol{x}$.

	x_1	x_2	x_3	y
$\boldsymbol{x}_1$	5.0	4.5	1.2	1
$\boldsymbol{x}_2$	5.0	4.5	4.3	5
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$\boldsymbol{x}_n$	3.0	3.5	0.5	16
$\boldsymbol{x}$	4.0	2.5	0.8	?

Top-k classification

- Training data: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.
- **Predict** top k classes $\{y_j\}_{j=1}^k$ for a given $\boldsymbol{x}$.

	x_1	x_2	x_3	y
$\boldsymbol{x}_1$	5.0	4.5	1.2	1
$\boldsymbol{x}_2$	5.0	4.5	4.3	5
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$\boldsymbol{x}_n$	3.0	3.5	0.5	16
$\boldsymbol{x}$	4.0	2.5	0.8	$\{1, 3, 16\}$

Top-k classification

- In top-k classification we define the prediction function as:

$$h(\boldsymbol{x}) = \operatorname{top}_f(k)\,,$$

  where $\operatorname{top}_f(k)$ returns top k classes with the highest score f for a given $\boldsymbol{x}$.
- The learning problem is then defined as minimization of the $1\text{–}prec@k$ loss:

$$\ell_{@k}(y, \boldsymbol{x}, f) = [\![y \notin \operatorname{top}_f(k)]\!]\,.$$

Top-k classification

- The **conditional risk** for the $1\text{–}prec@k$ loss is:

$$L_{@k}(h \,|\, \boldsymbol{x}) = 1 - \sum_{y \in \text{top}_f(k)} P(y \,|\, \boldsymbol{x})$$

- Therefore, the **Bayes classifier** for the $1\text{–}prec@k$ loss predicts top k classes with largest conditional probabilities $P(y \,|\, \boldsymbol{x})$, i.e.:

$$h^*(\boldsymbol{x}) = \text{top}_P(k)\,.$$

- The **conditional regret** for the $1\text{–}prec@k$ is:

$$\text{Reg}_{@k}(h \,|\, \boldsymbol{x}) = \sum_{y \in \text{top}_P(k)} P(y \,|\, \boldsymbol{x}) - \sum_{y \in \text{top}_f(k)} P(y \,|\, \boldsymbol{x})\,.$$

Top-1 classification

- Top-1 classification corresponds to minimization of $0/1$ loss:

$$\ell_{0/1} = [\![y \neq h(\boldsymbol{x})]\!] \, .$$

- The simplest reduction, referred to as **1vsAll**, relies on solving m binary classification problems of the form:

$$(\boldsymbol{x}, y) \longrightarrow (\boldsymbol{x}, y' = [\![y = j]\!]) \quad j \in \mathcal{Y} \, .$$

- Traning and prediction is **linear** in m.
- **Consistent** if used with probabilistic classifiers.

Label tree structure

- Organize class labels into a tree to improve computational complexity:[2]
  - Binary coding of class labels.
  - Hierarchical clustering.
  - Tree structure learning.

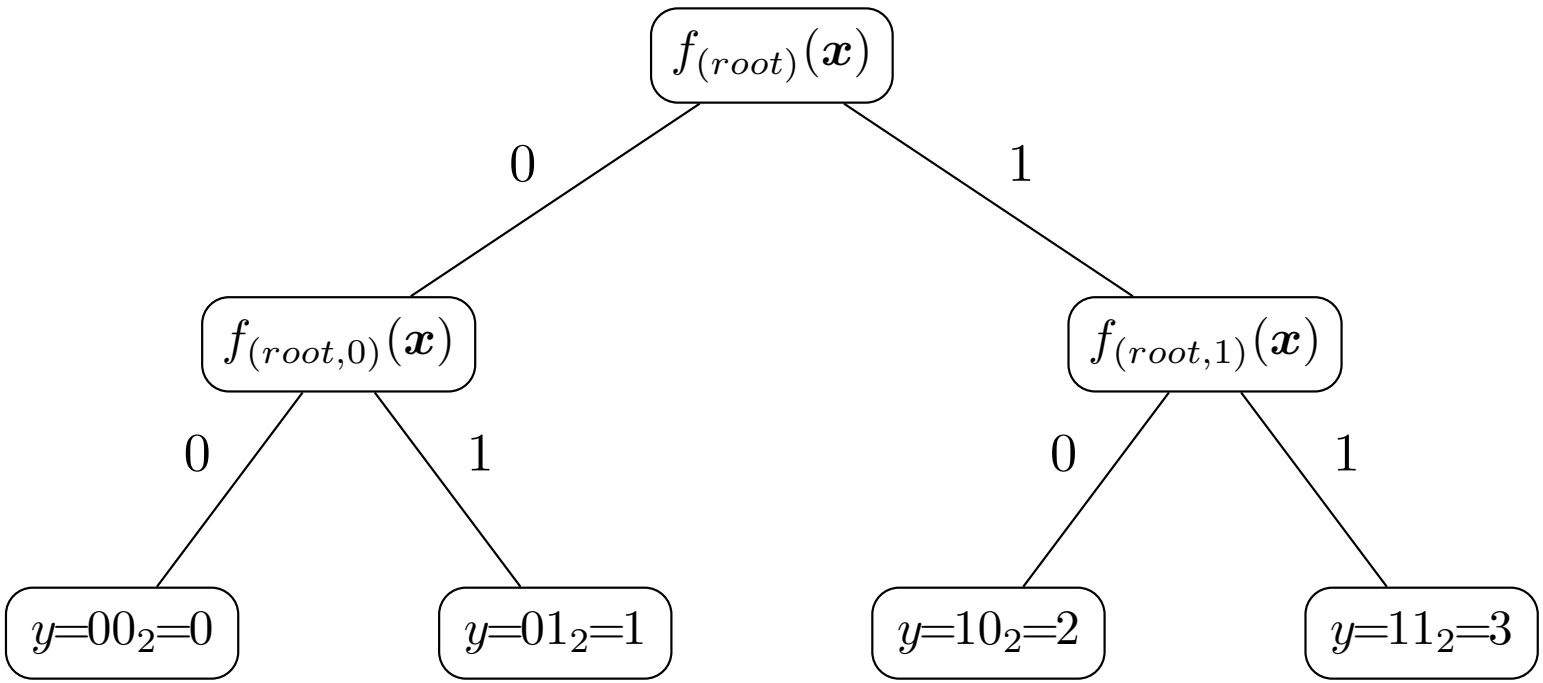
[2] A. Beygelzimer, J. Langford, Y. Lifshits, G. B. Sorkin, and A. L. Strehl. Conditional probability tree estimation analysis and algorithms. In *UAI*, pages 51–58, 2009

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

Label tree structure

- Assign each class an integer from 0 to $m-1$ and use a binary code of lenght t.
- Leafs correspond to class labels.
- Internal nodes (including the root) contain binary classifiers.



Label tree structure

- Different training schemes possible:
  - Train a classifier in each node.
  - Train a classifier on each level.
  - Train one global binary classifier (in several loops).
- Complexity of learning is in fact $O(\log m)$:
  - We need only $\log m$ copies of a given training example (on a path from the root to the corresponding leaf).
- Prediction can be done in $O(\log m)$.

Greedy prediction

- Greedy prediction follows one path from the root to the leaf:

$$\begin{aligned} f_{(root)} &: && \boldsymbol{x} \mapsto \hat{y}_1 \\ f_{(root,\hat{y}_1)} &: && \boldsymbol{x} \mapsto \hat{y}_2 \\ &\dots && \dots \\ f_{(root,\hat{y}_1,\dots\hat{y}_{t-1})} &: && \boldsymbol{x} \mapsto \hat{y}_t \end{aligned}$$

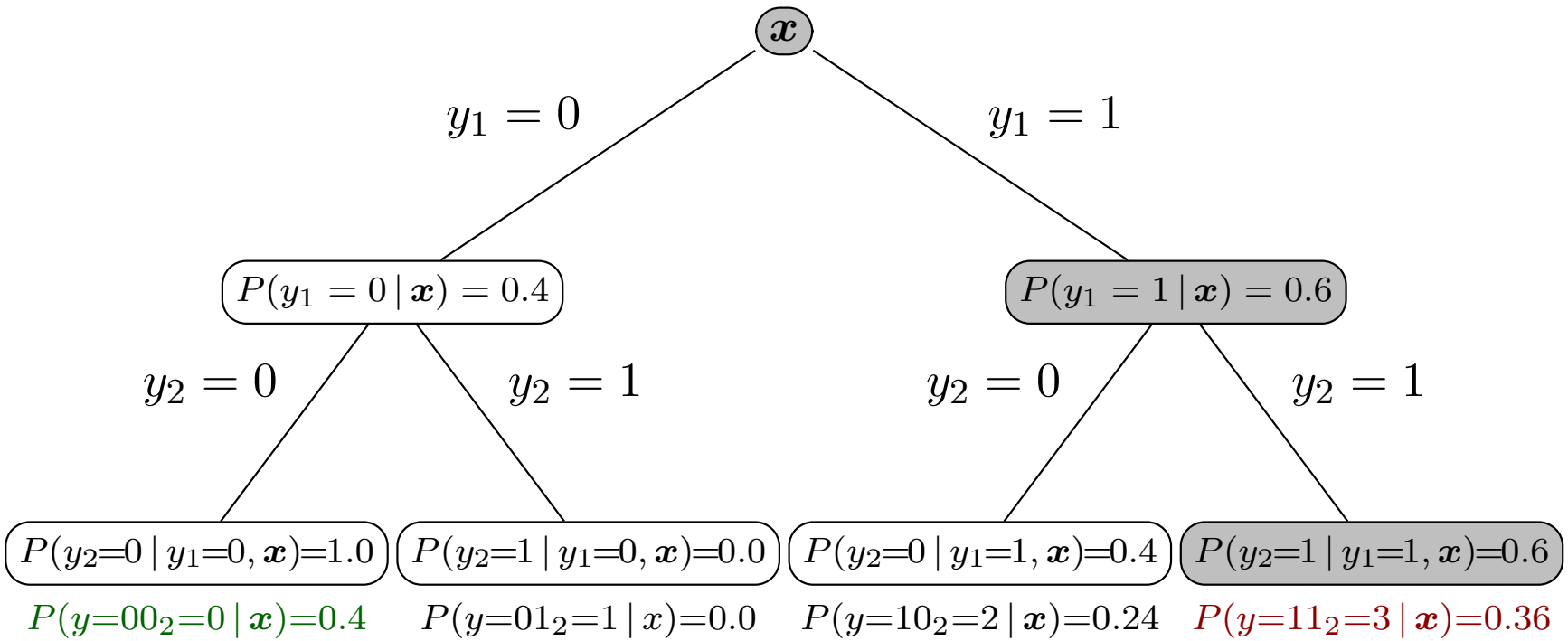
- The final prediction is

$$\hat{y} = \text{decode}(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_t)$$

- Greedy search is fast ($O(\log m)$).
- Any binary classifier can be used.
- However, this approach is **inconsistent**.

Greedy prediction

- Greedy prediction fails for finding the most probable class:



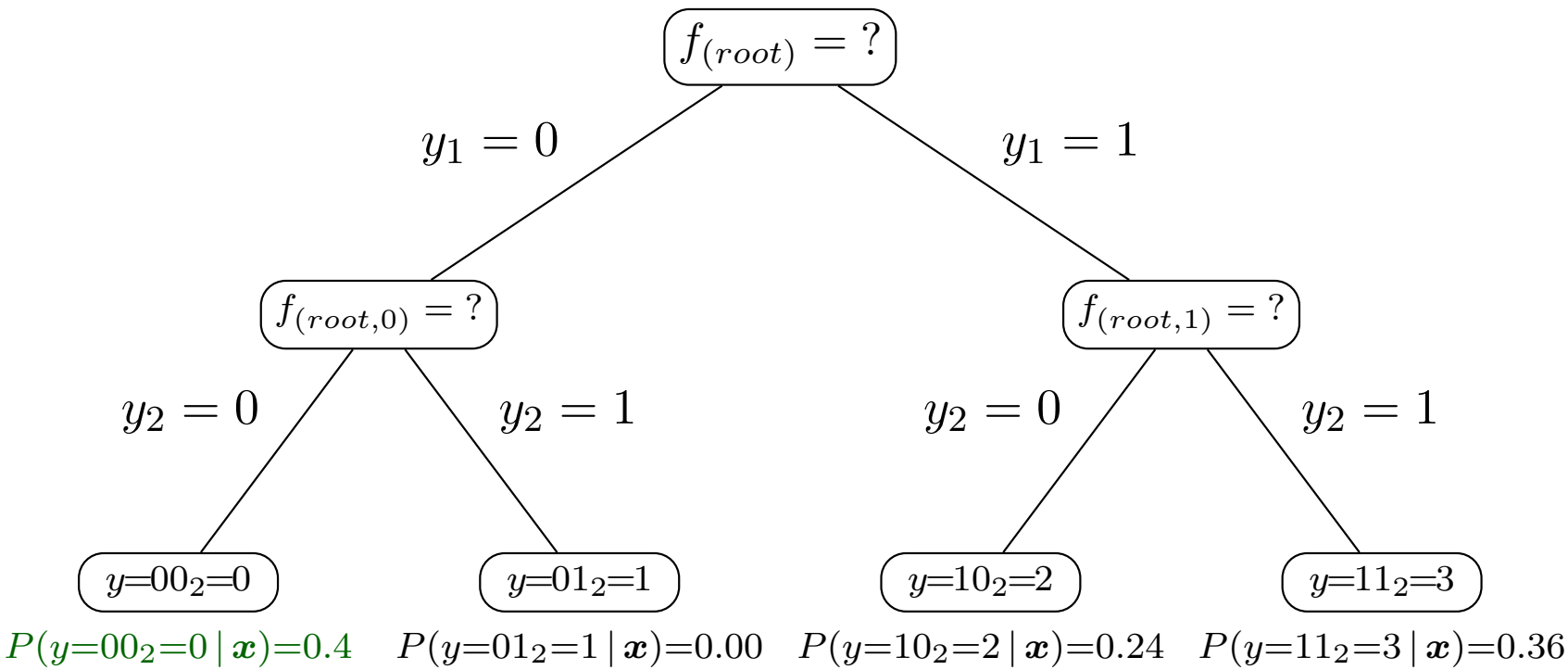
Filter trees

- Filter trees[3] (FT) use a **bottom-up** learning algorithm to train the label tree $\rightarrow$ a single **elimination tournament** on the set of class labels.
- FT implicitly transform the underlying distribution P over multi-class examples into a **specific distribution** P^{FT} over weighted binary examples.
- FT use the greedy prediction.
- FT are **consistent** for 0/1 loss and cost-sensitive classification.

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

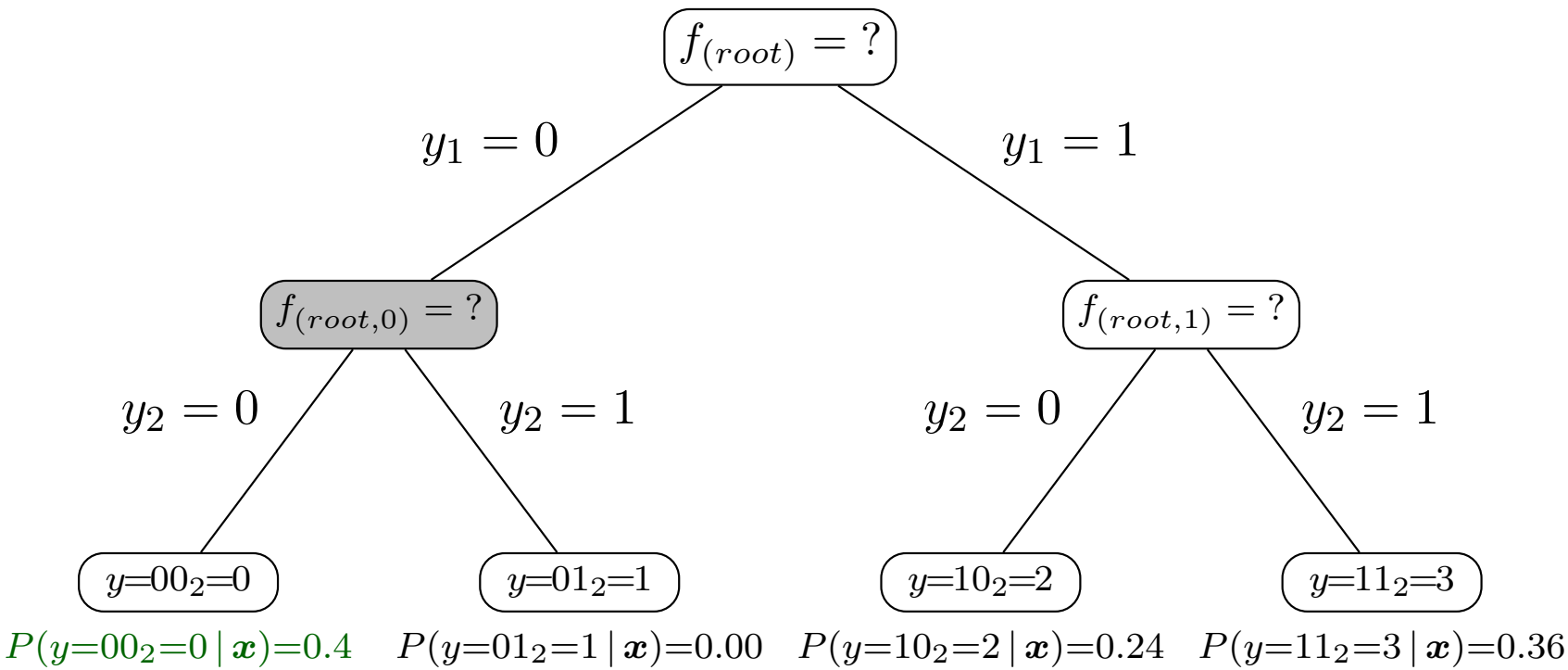
Filter trees

- FT **filter out** all examples that are misclassified by the lower-level classifiers.
- $f_{(root,y_1,\ldots y_i)}(\boldsymbol{x})$ predicts y_{i+1} given that all classifiers below predict the subsequent decisions correctly.



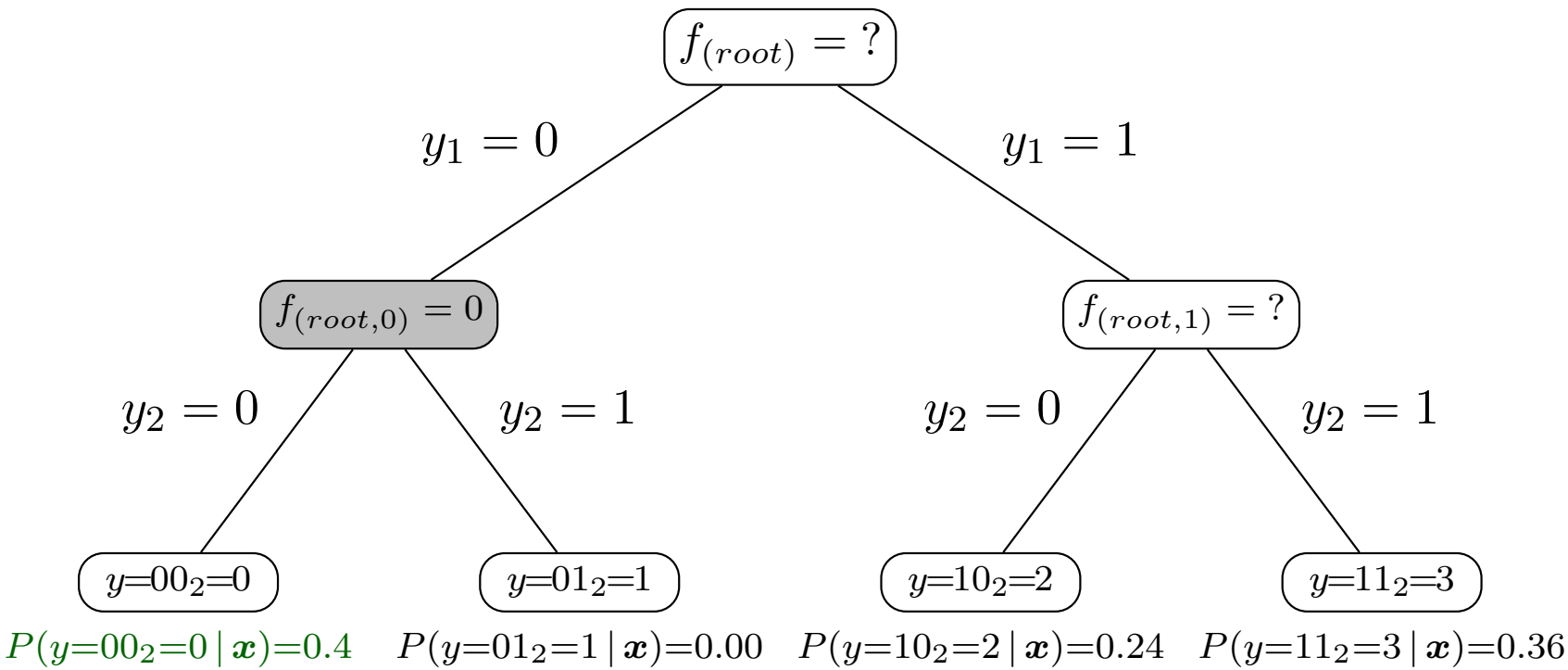
Filter trees

- FT **filter out** all examples that are misclassified by the lower-level classifiers.
- $f_{(root,y_1,\ldots y_i)}(\boldsymbol{x})$ predicts y_{i+1} given that all classifiers below predict the subsequent decisions correctly.



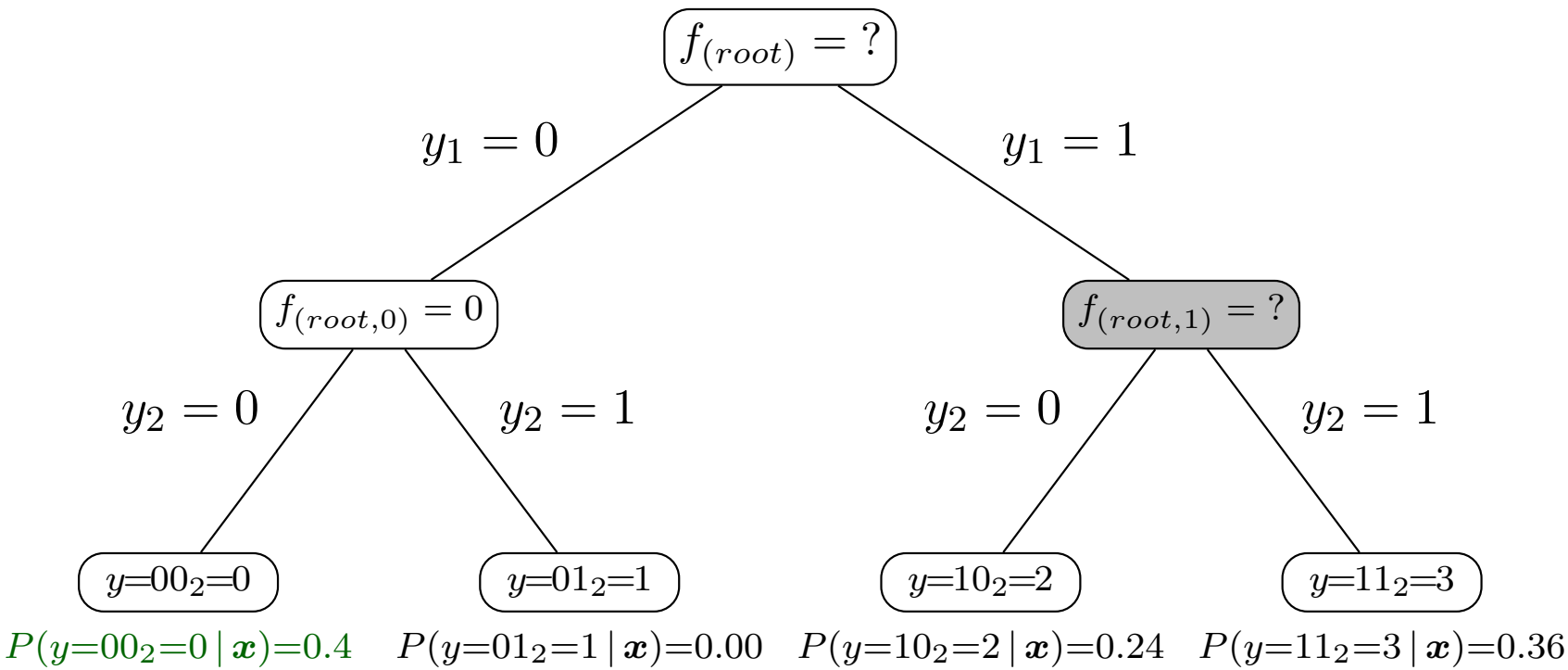
Filter trees

- FT **filter out** all examples that are misclassified by the lower-level classifiers.
- $f_{(root,y_1,\ldots y_i)}(\boldsymbol{x})$ predicts y_{i+1} given that all classifiers below predict the subsequent decisions correctly.



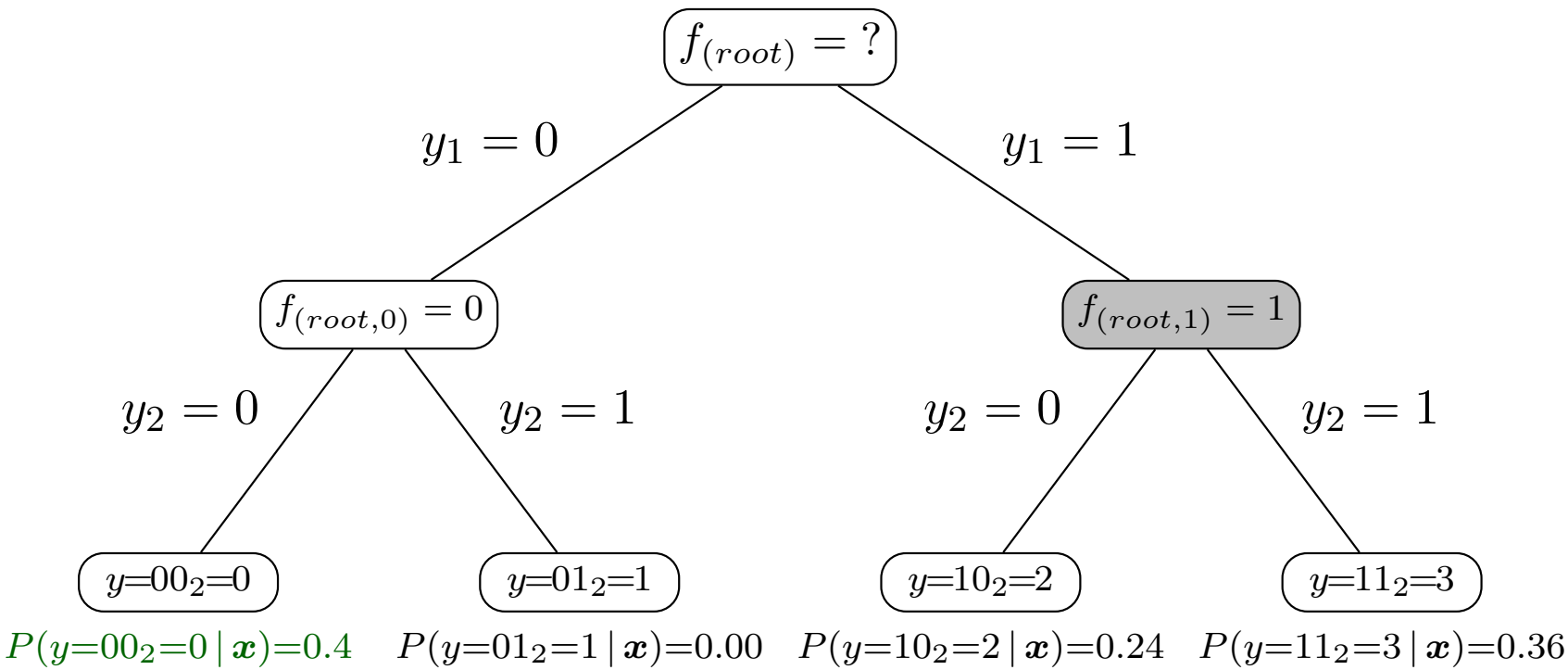
Filter trees

- FT **filter out** all examples that are misclassified by the lower-level classifiers.
- $f_{(root,y_1,\ldots y_i)}(\boldsymbol{x})$ predicts y_{i+1} given that all classifiers below predict the subsequent decisions correctly.



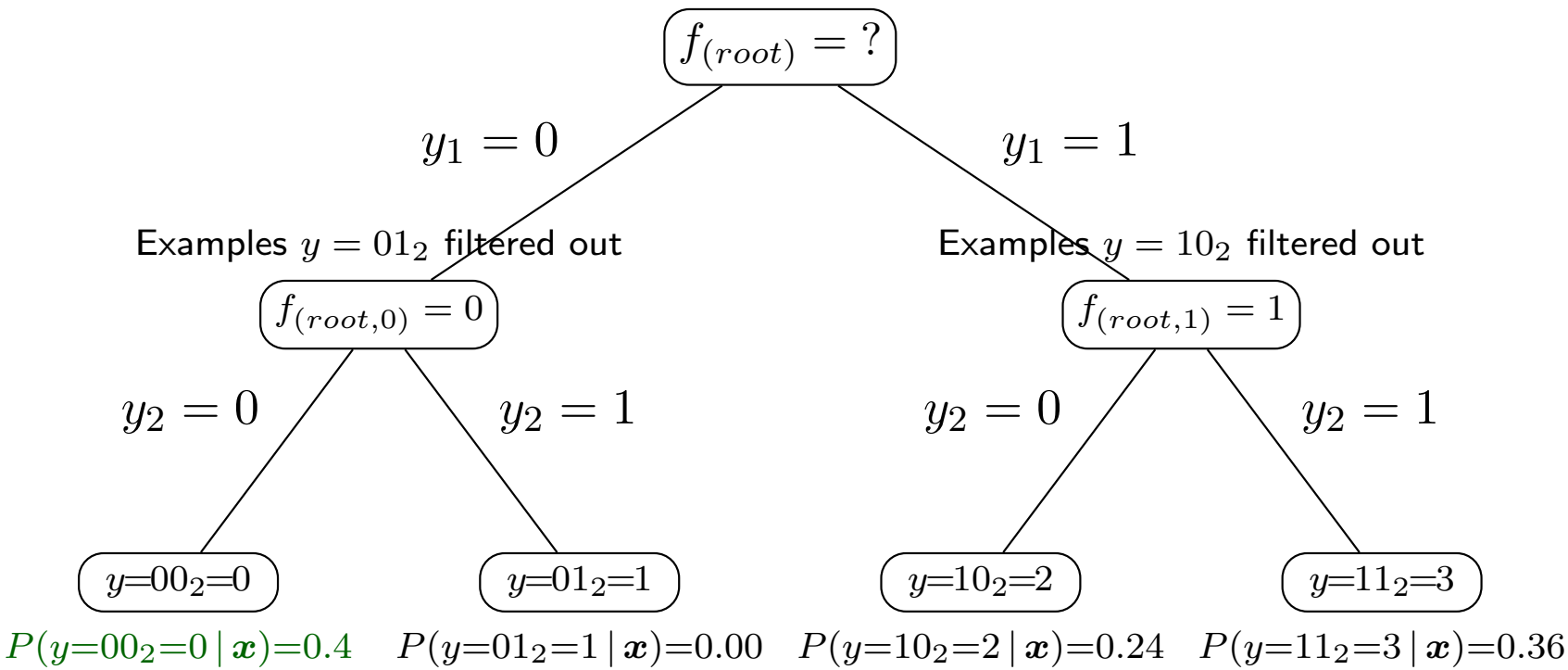
Filter trees

- FT **filter out** all examples that are misclassified by the lower-level classifiers.
- $f_{(root,y_1,\ldots y_i)}(\boldsymbol{x})$ predicts y_{i+1} given that all classifiers below predict the subsequent decisions correctly.



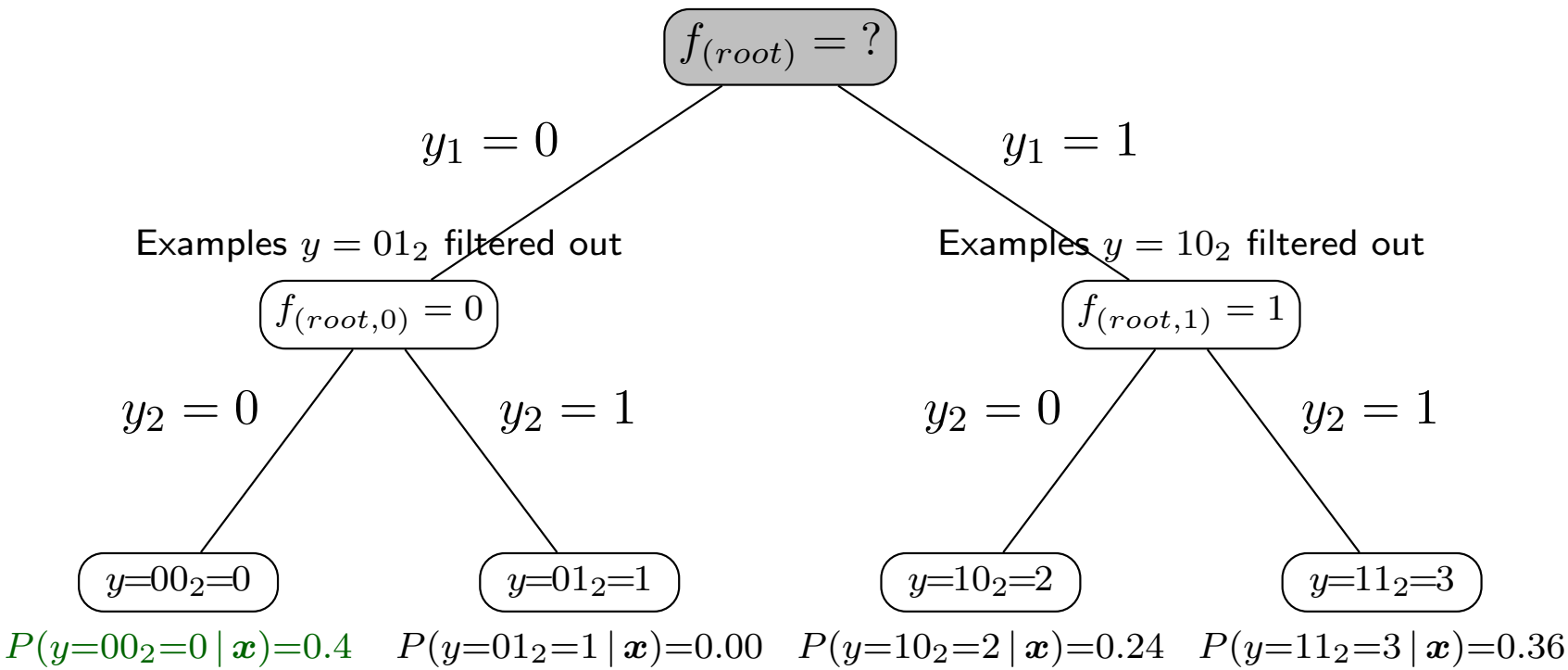
Filter trees

- FT **filter out** all examples that are misclassified by the lower-level classifiers.
- $f_{(root,y_1,\ldots y_i)}(\boldsymbol{x})$ predicts y_{i+1} given that all classifiers below predict the subsequent decisions correctly.



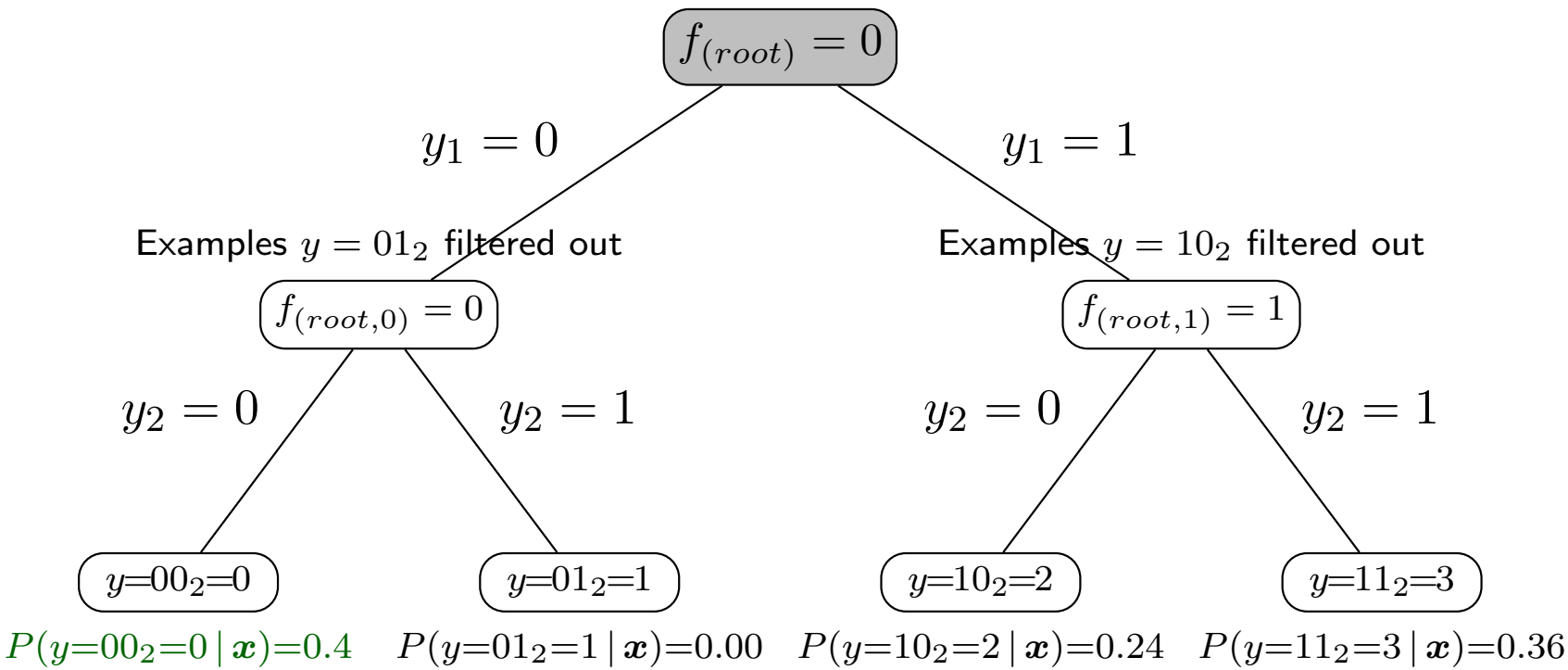
Filter trees

- FT **filter out** all examples that are misclassified by the lower-level classifiers.
- $f_{(root,y_1,\ldots y_i)}(\boldsymbol{x})$ predicts y_{i+1} given that all classifiers below predict the subsequent decisions correctly.



Filter trees

- FT **filter out** all examples that are misclassified by the lower-level classifiers.
- $f_{(root,y_1,\ldots y_i)}(\boldsymbol{x})$ predicts y_{i+1} given that all classifiers below predict the subsequent decisions correctly.



Filter trees

- FT have strong guarantees in the form of the regret bound:

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

- However, they cannot be easily adapted to the problem of top-k classification.

Probabilistic classifier trees

- Let the node classifiers $f_{(root,y_1,\ldots y_{i-1})}$ estimate $P(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1})$:

$$\begin{aligned} Q(y_i = 1 \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) &= f_{(root,y_1,\ldots y_{i-1})} \\ Q(y_i = 0 \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) &= 1 - f_{(root,y_1,\ldots y_{i-1})} \end{aligned}$$

- Then, we can easily estimate $P(y \,|\, \boldsymbol{x})$ by:[4]

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

- We refer to this approach as probabilistic classifier trees (PCT).

[4] 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

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

Advanced search techniques

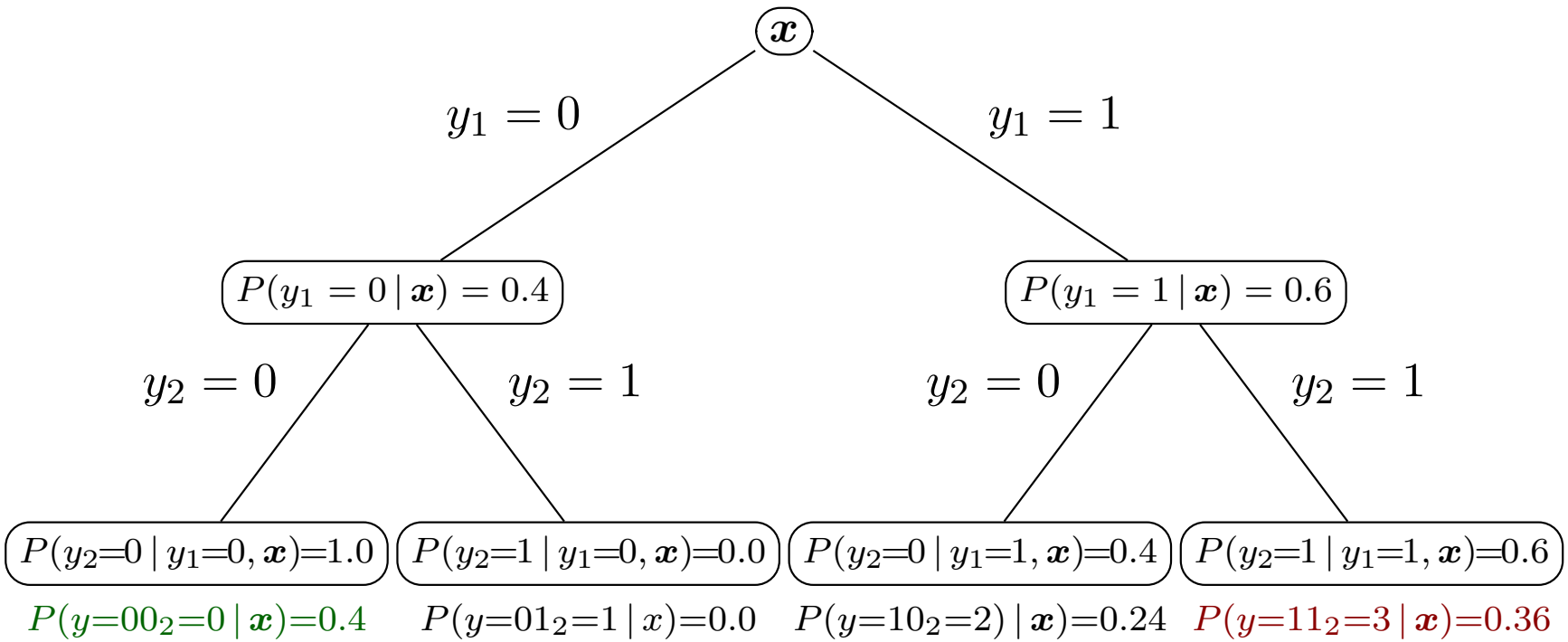
- Finding the most probable class label relies on finding the most probable path in the tree.
- Advanced search techniques: beam search,[5] uniform-cost search.[6]
- The use of a priority queue and a cut point gives a fast algorithm with provable guarantees.

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

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

Advanced search techniques

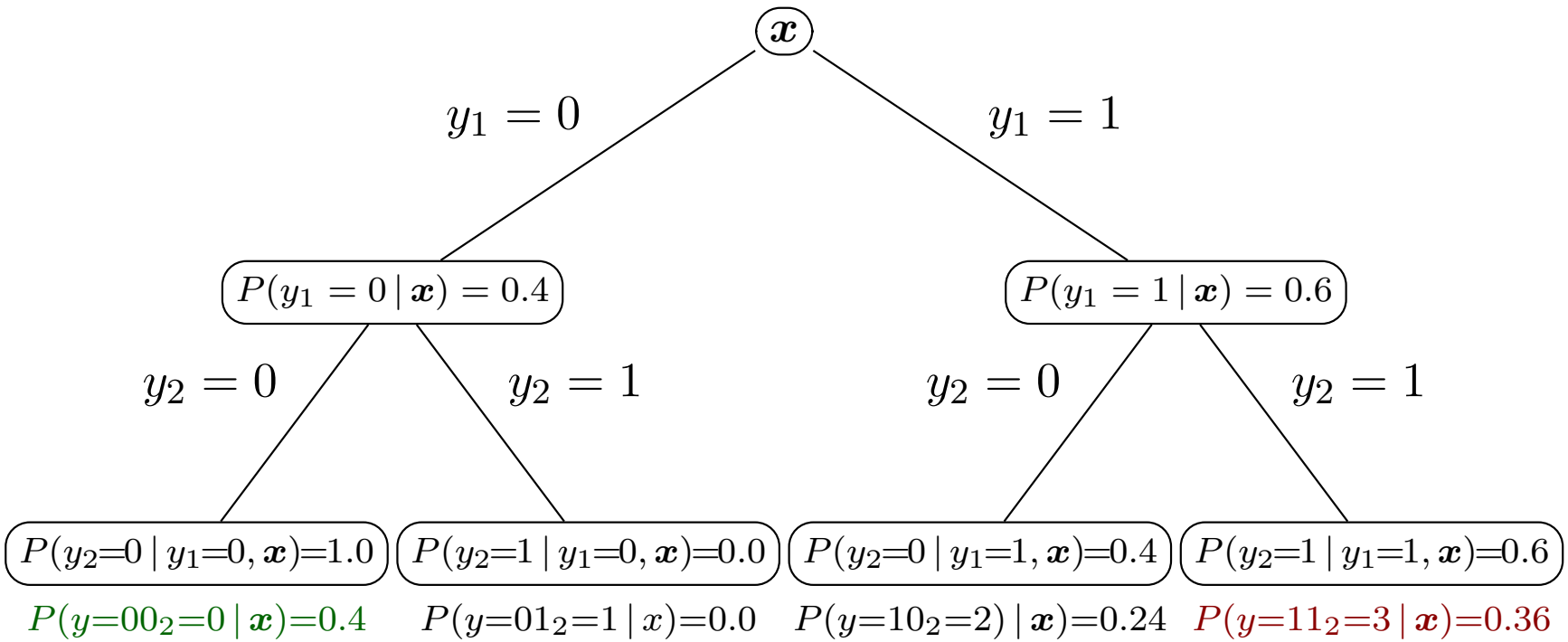
- Uniform-cost search



- Priority list $\mathcal{Q}$:

Advanced search techniques

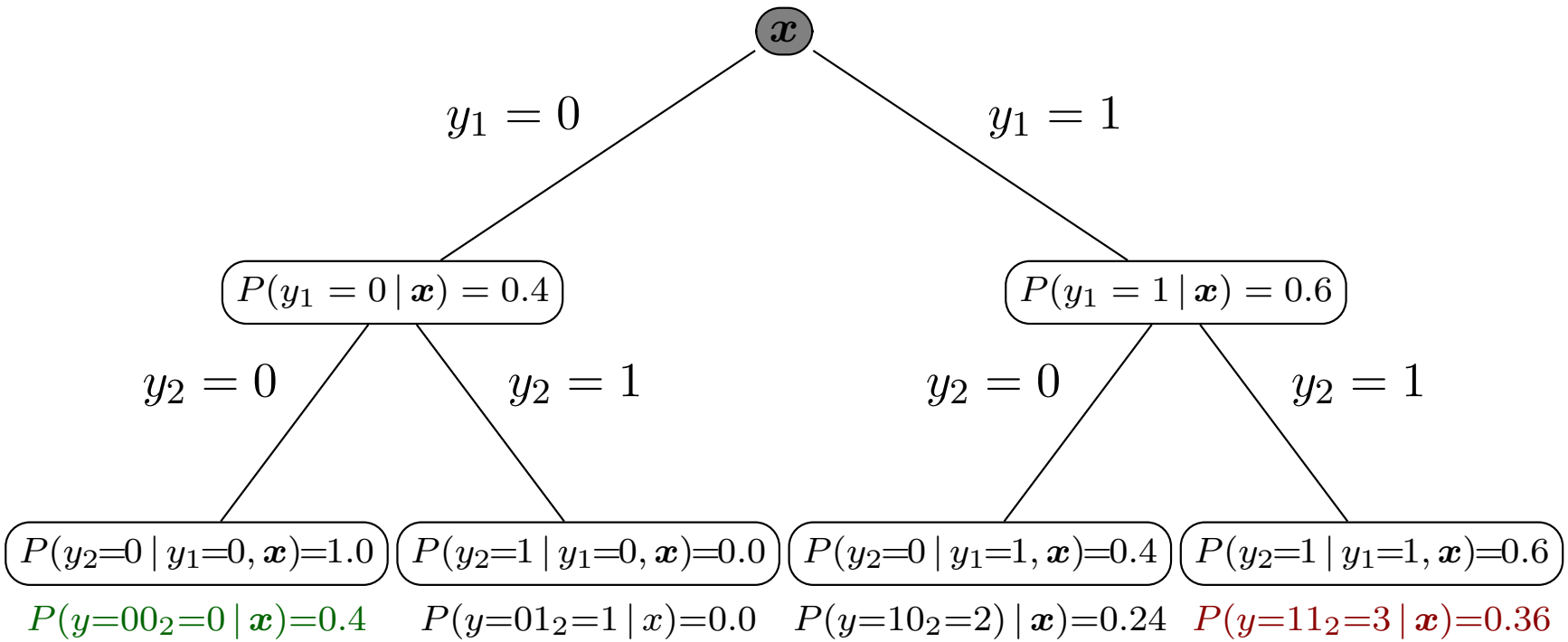
- Uniform-cost search



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

Advanced search techniques

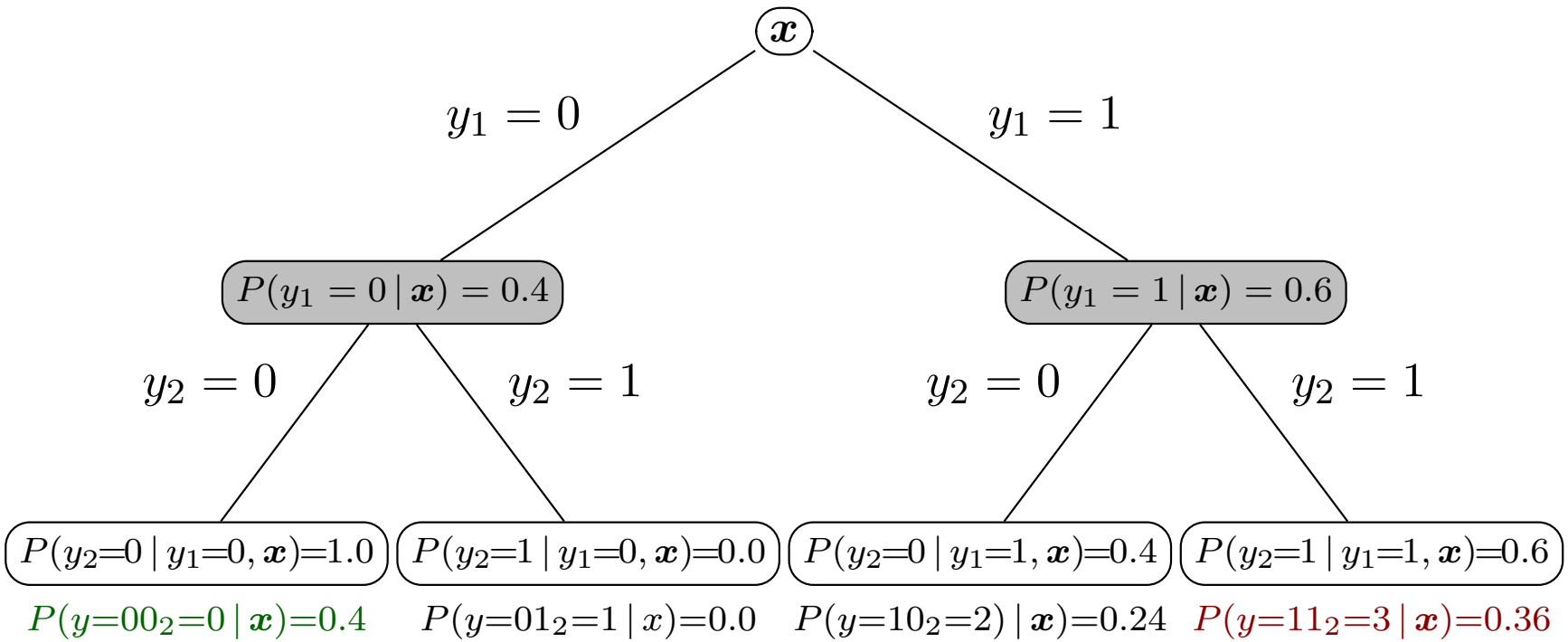
- Uniform-cost search



- Priority list $\mathcal{Q}$:

Advanced search techniques

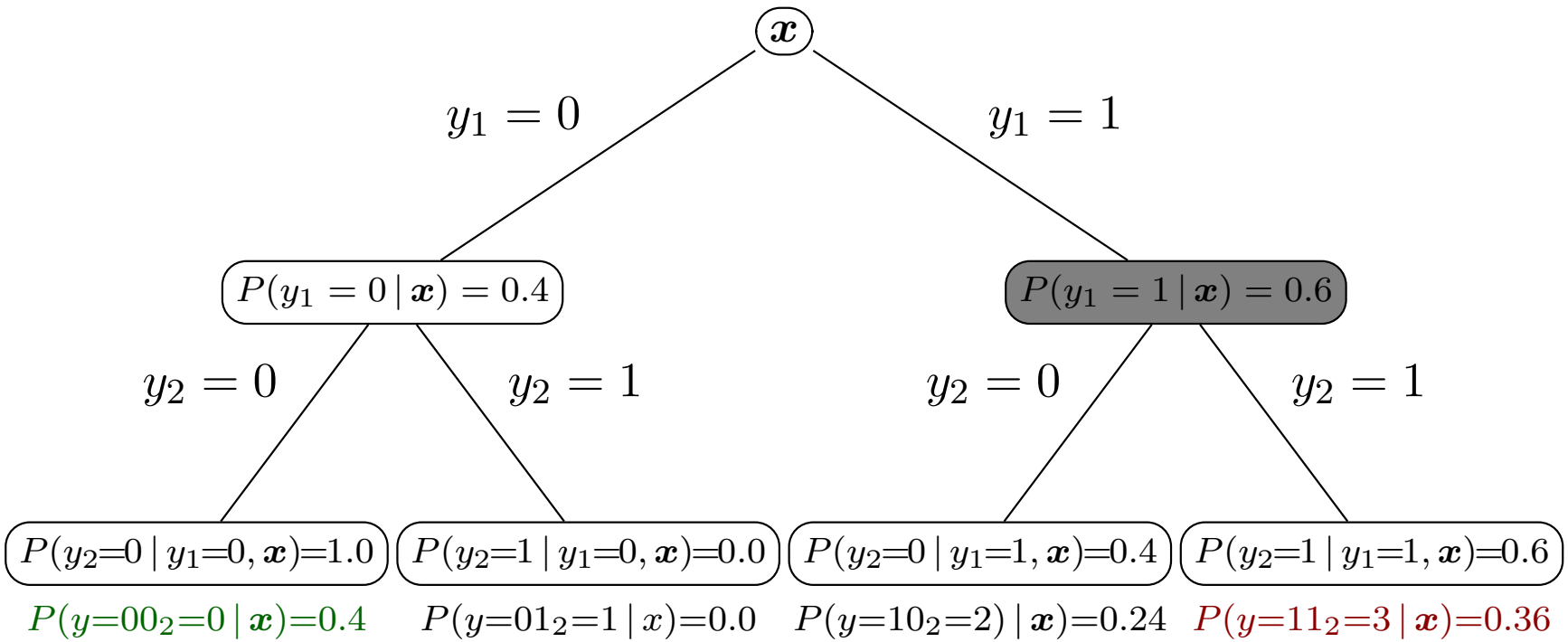
- Uniform-cost search



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

Advanced search techniques

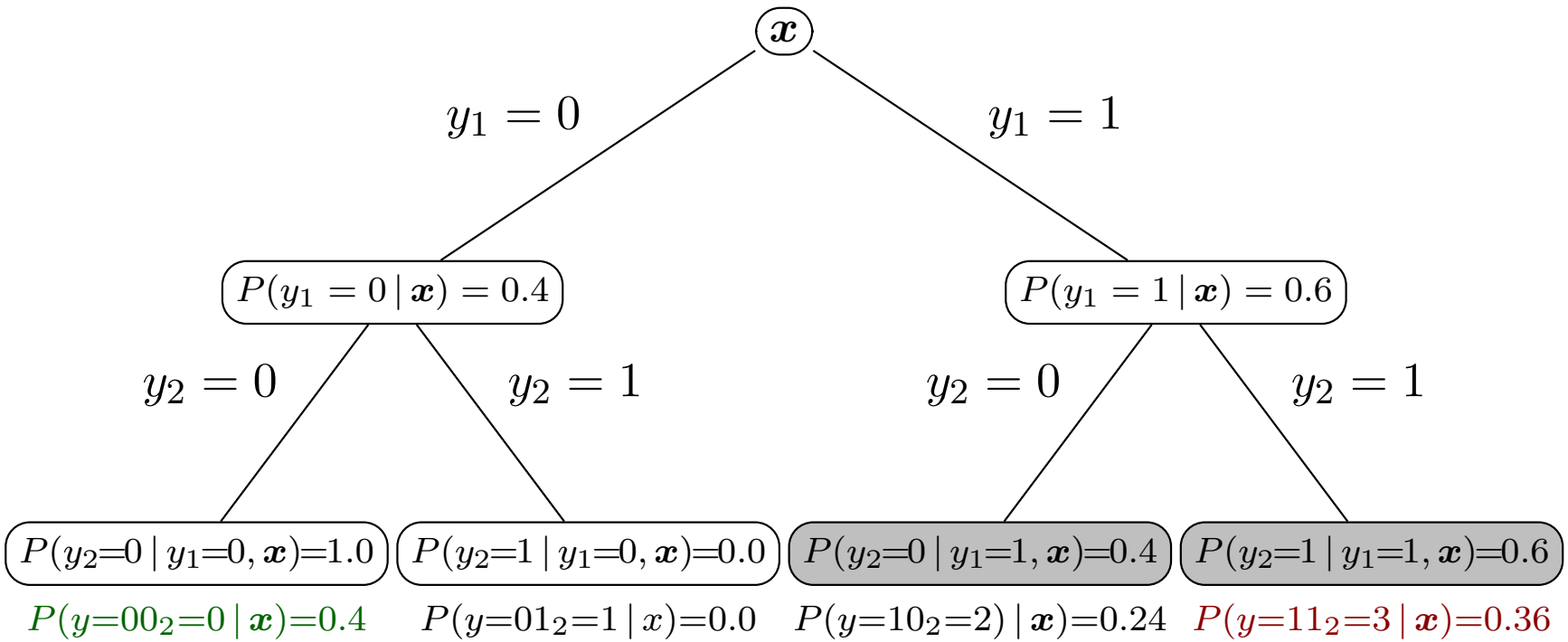
- Uniform-cost search



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

Advanced search techniques

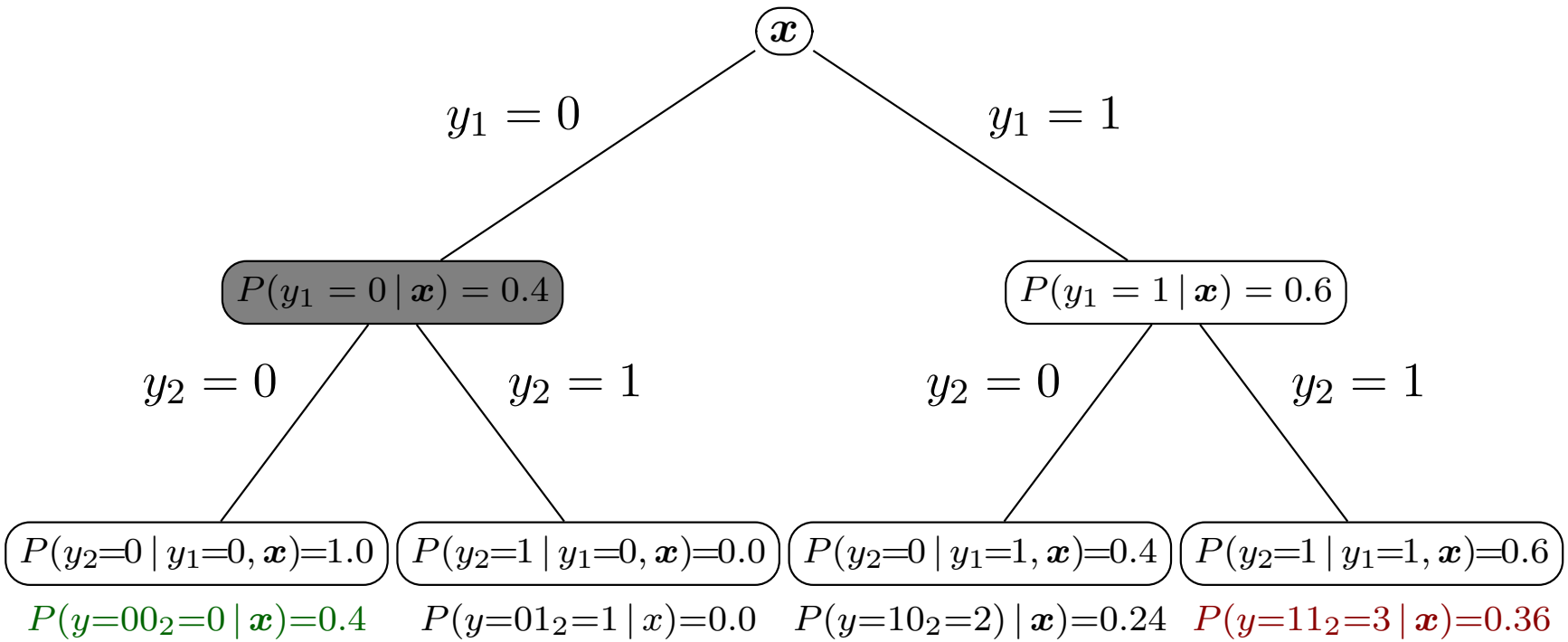
- Uniform-cost search



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

Advanced search techniques

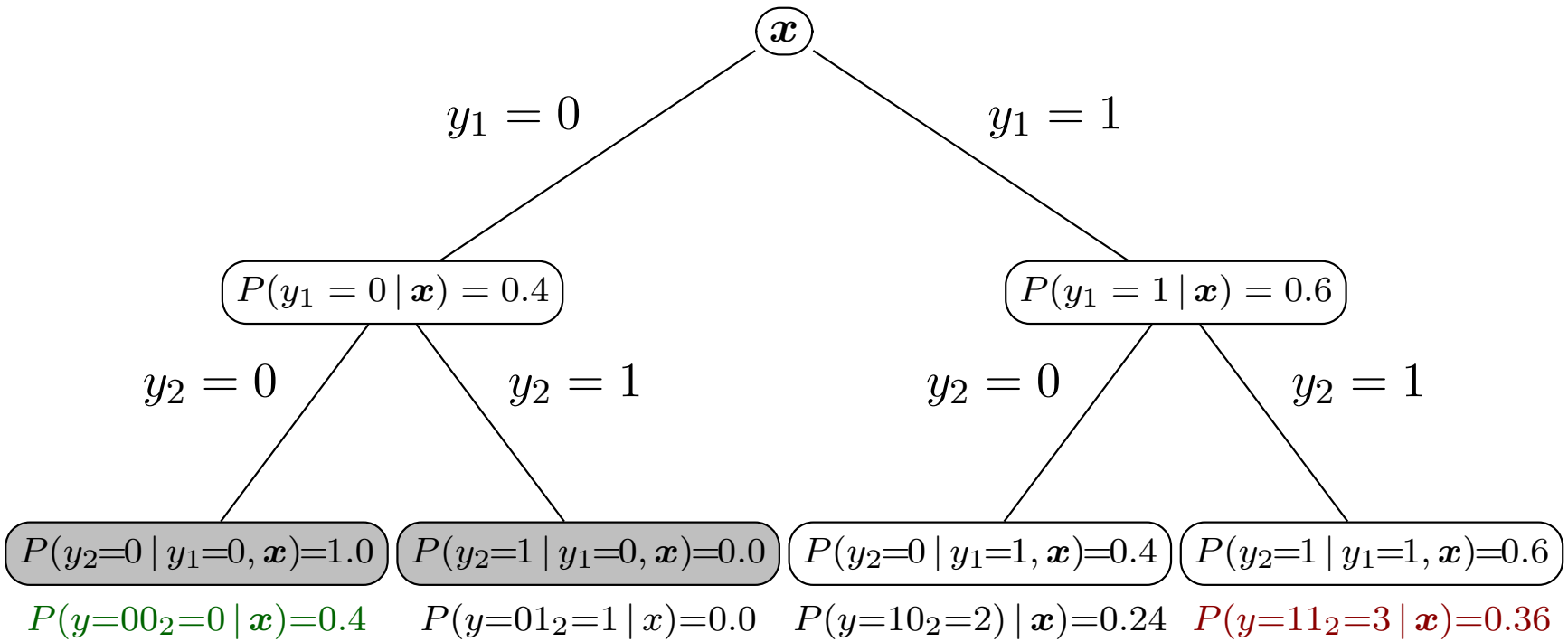
- Uniform-cost search



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

Advanced search techniques

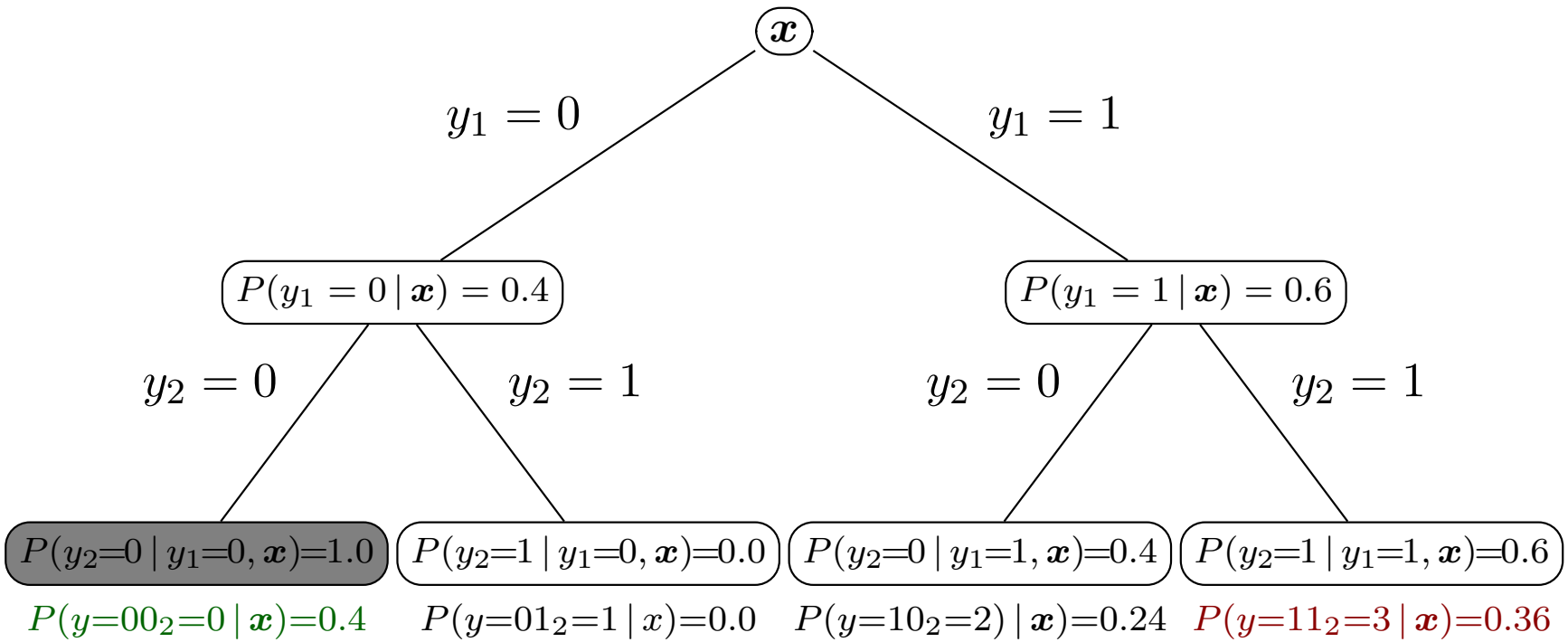
- Uniform-cost search



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

Advanced search techniques

- Uniform-cost search



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

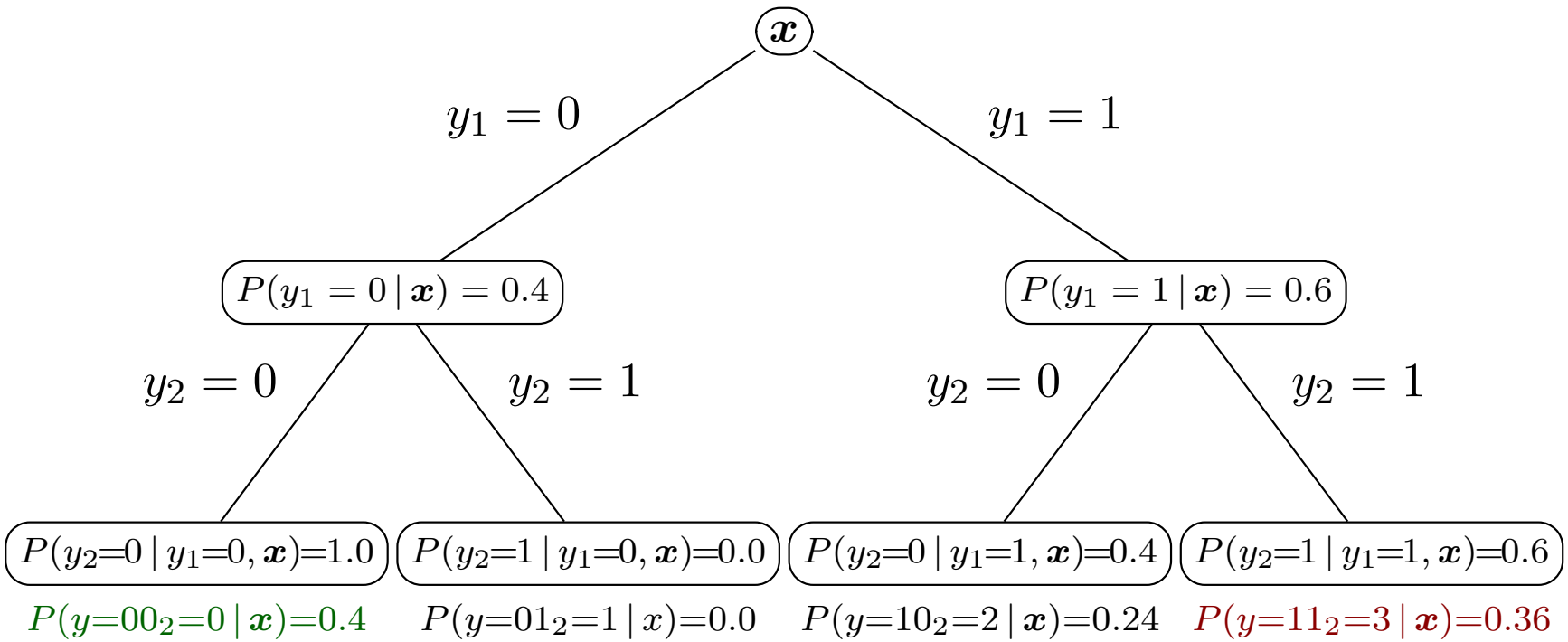
Advanced search techniques

- ϵ-approximation inference:[7]
  - 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.

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

ϵ-approximation inference

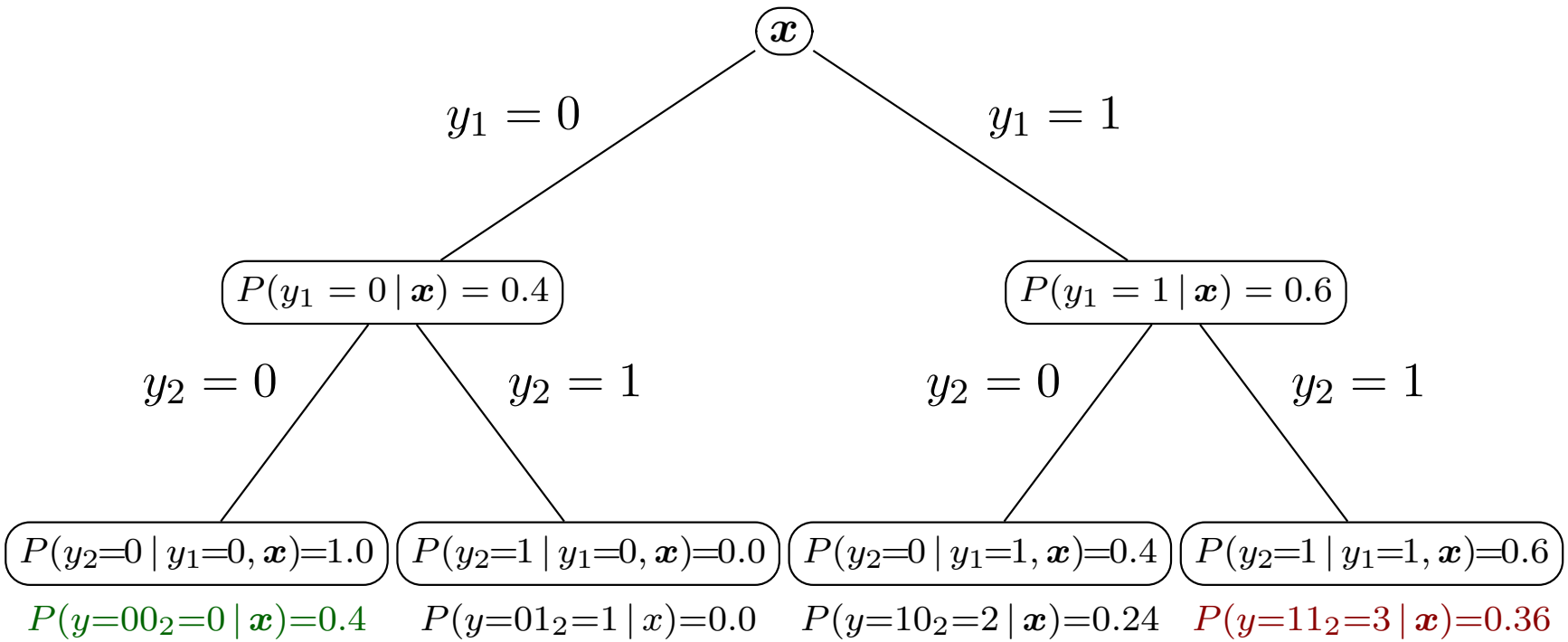
- $\epsilon = 0.5$



- Priority list $\mathcal{Q}$:

ϵ-approximation inference

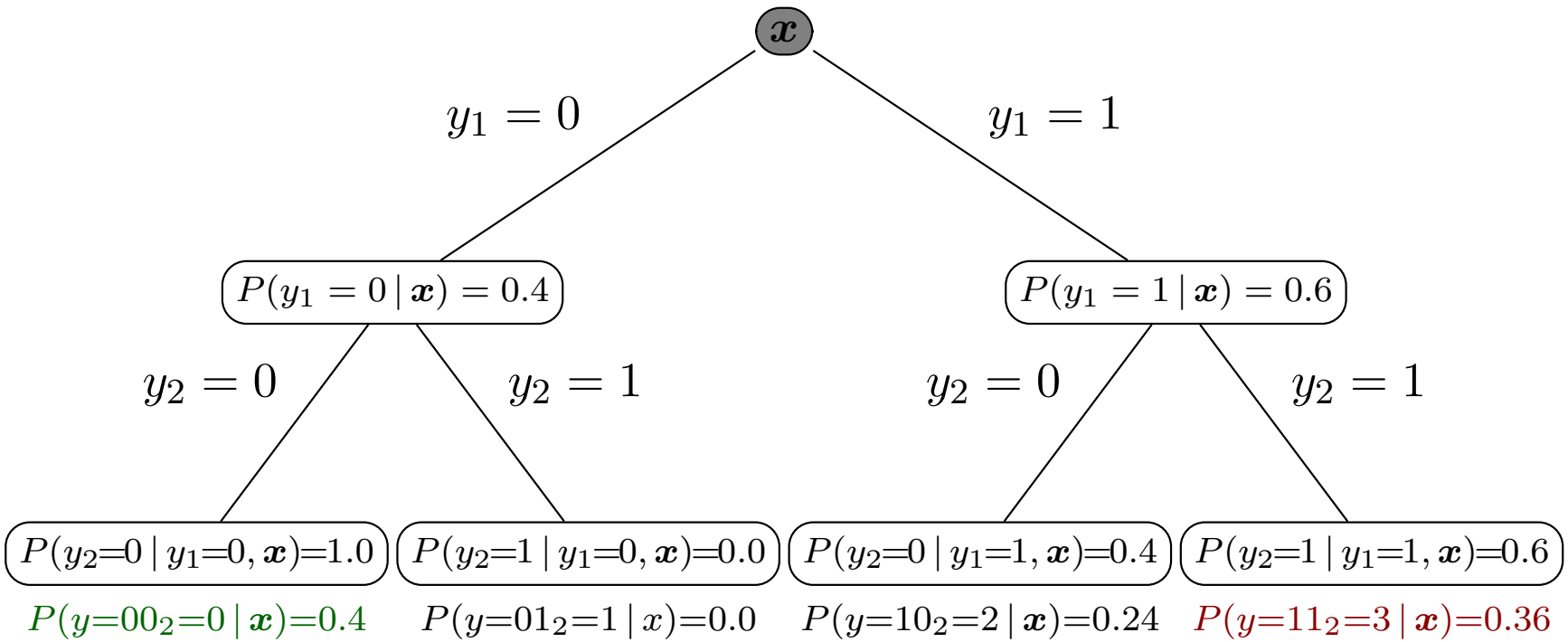
- $\epsilon = 0.5$



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

ϵ-approximation inference

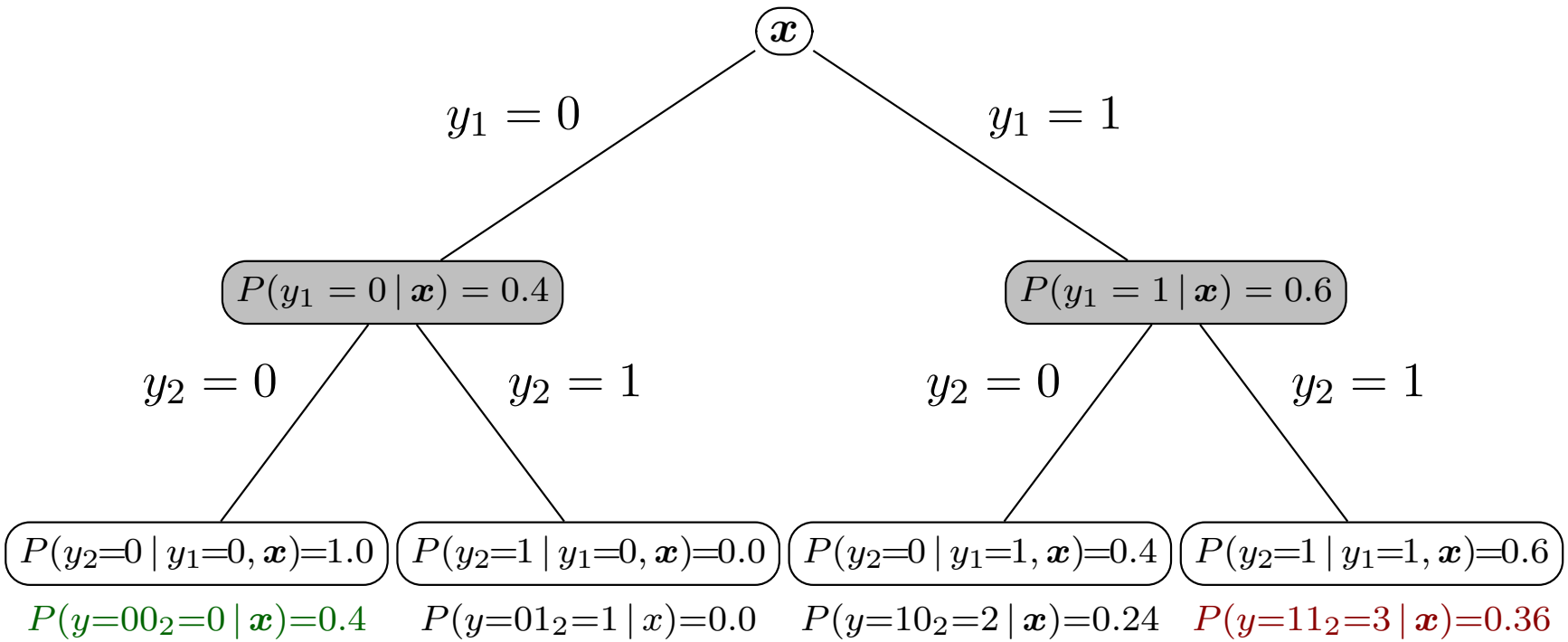
- $\epsilon = 0.5$



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

ϵ-approximation inference

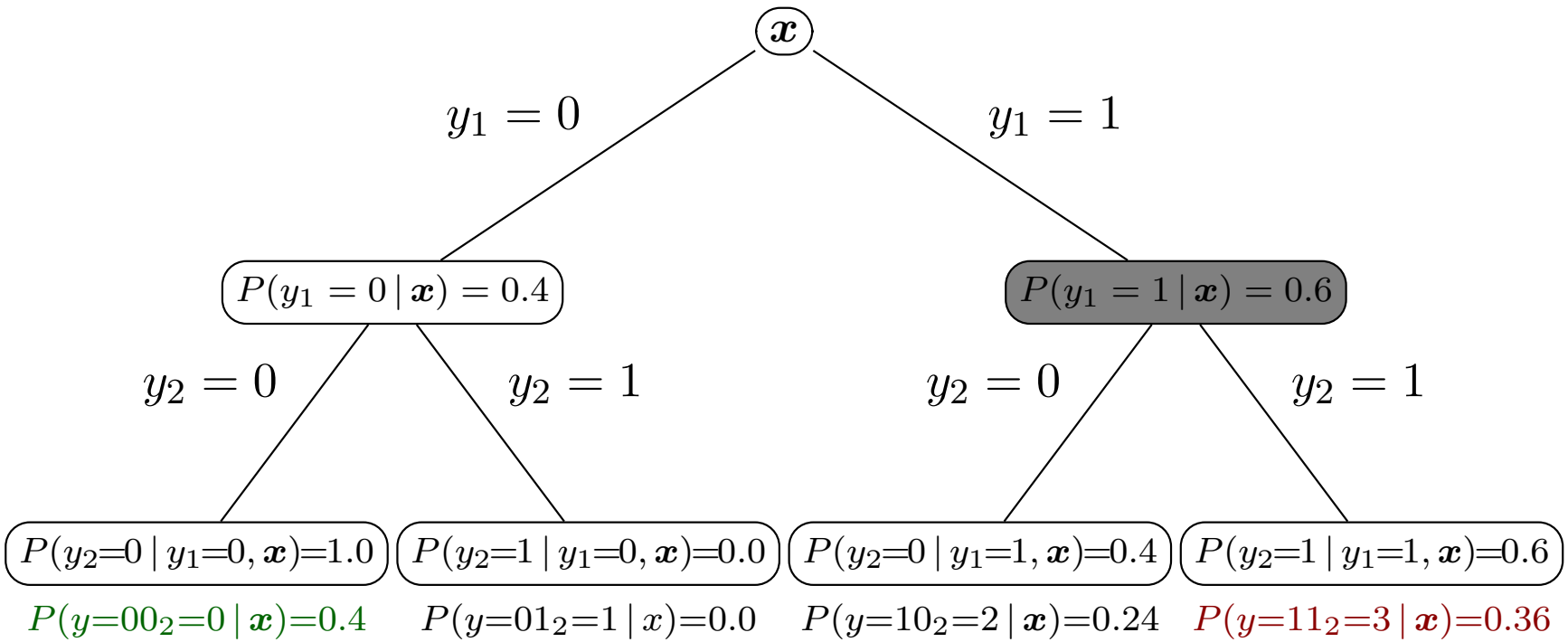
- $\epsilon = 0.5$



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

ϵ-approximation inference

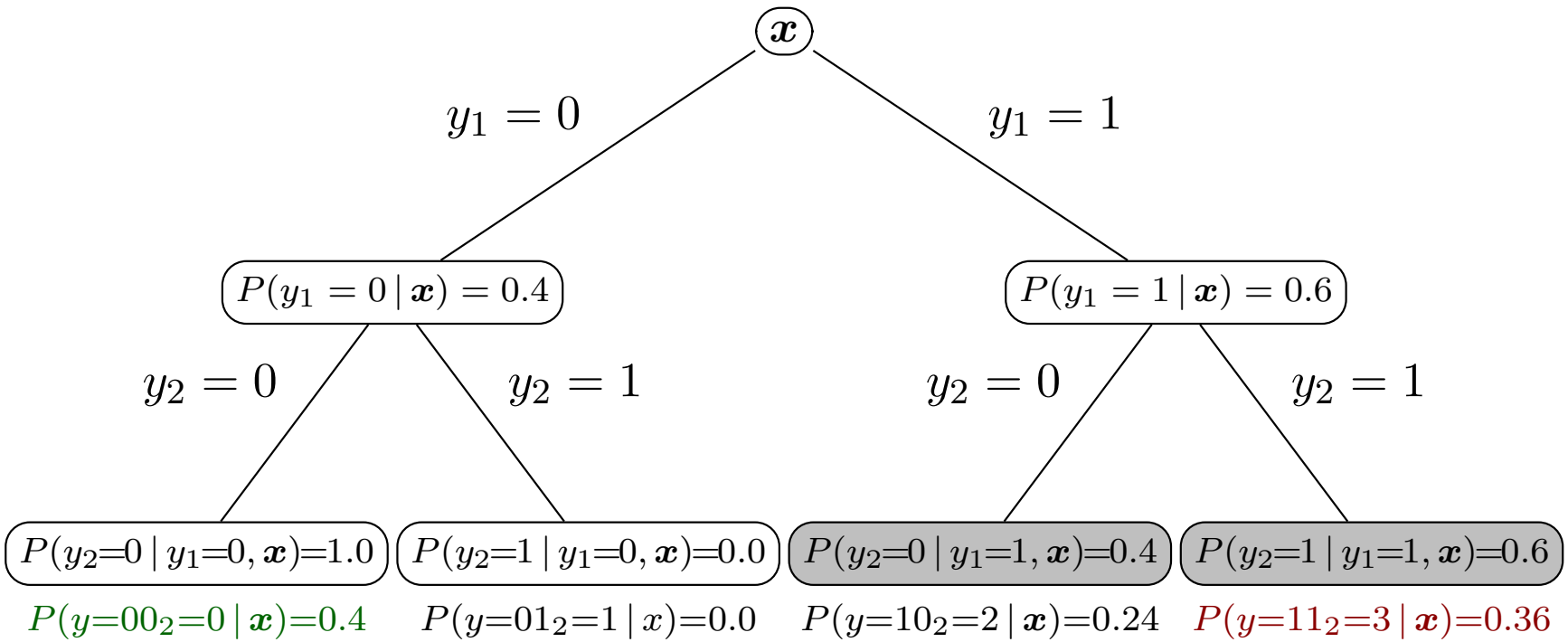
- $\epsilon = 0.5$



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

ϵ-approximation inference

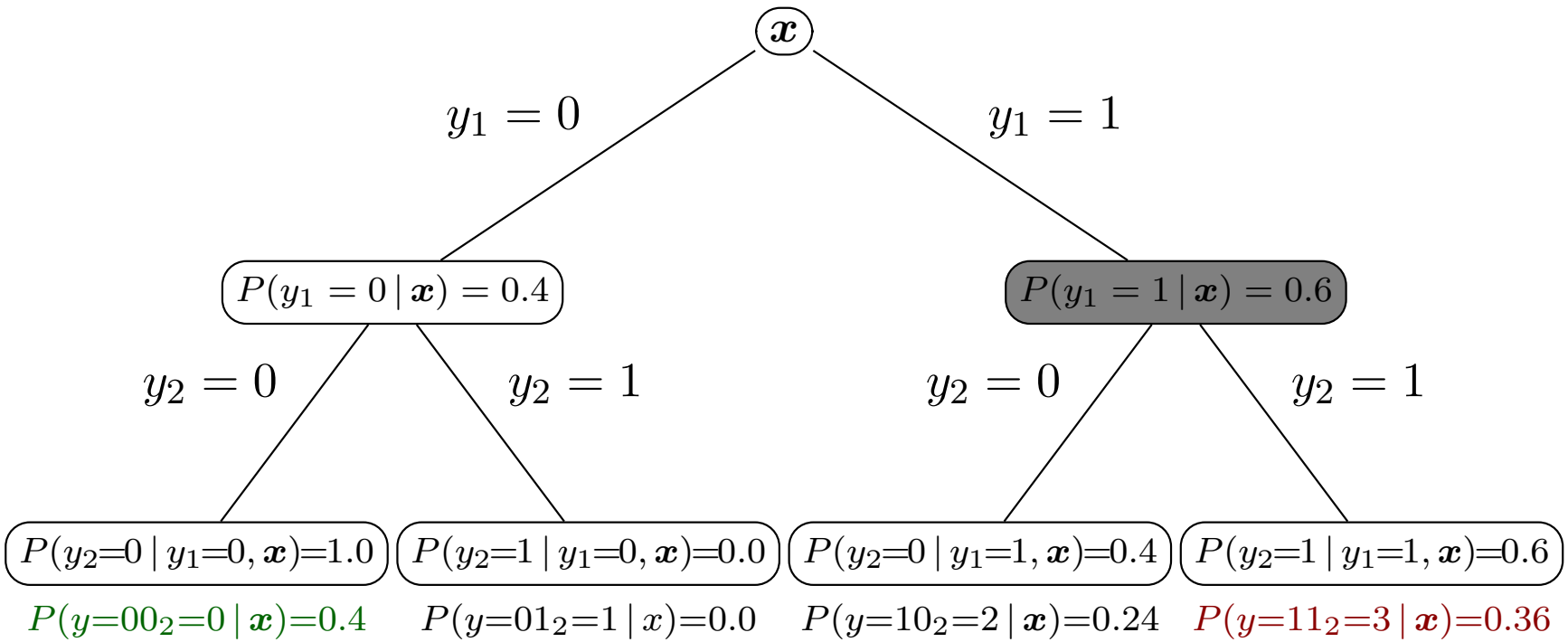
- $\epsilon = 0.5$



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

ϵ-approximation inference

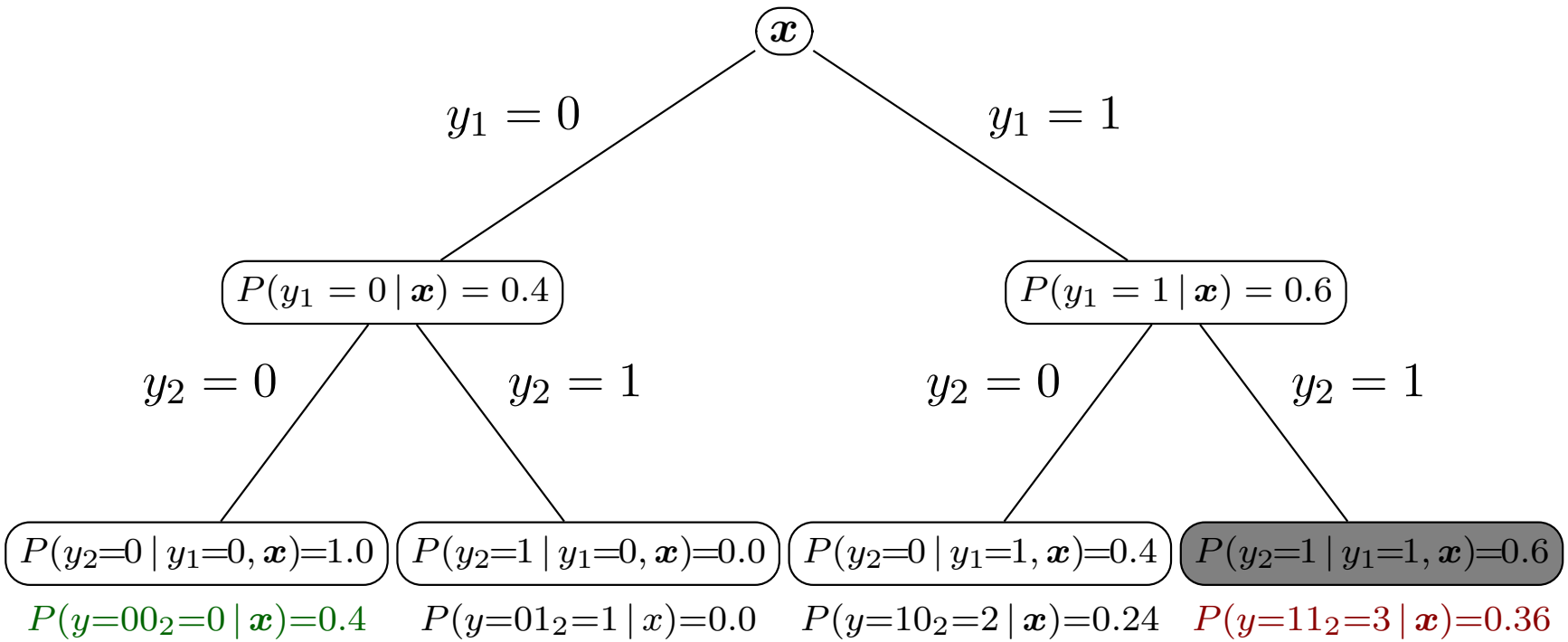
- $\epsilon = 0.5$



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

ϵ-approximation inference

- $\epsilon = 0.5$



- Priority list $\mathcal{Q}$: Suboptimal solution 11_2 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 guarantee can be given:

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 guarantee can be given:

  Theorem: Let $\epsilon = 2^{-c}$, where $1 \leq c \leq t$. The class label $\hat{y}$ will be returned in time $\mathcal{O}(t2^c)$ with a guarantee that:

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

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 prediction has very bad guarantees:

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

Regret bound for PCT

- The overall guarantees of PCT can be formalized as follows:

Regret bound for PCT

- The overall guarantees of PCT can be formalized as follows:

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

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

PCT for top-k classification

- Training of PCT does not change for top-k classification.
- The ϵ-approximate inference can be easily extended to the case of searching top-k class labels.
- The overall guarantees will slighlty change by a factor of k.

Experimental results

Top-1 multi-class classification

		0/1[%]	t_{train}	t_{test}	A	min	max	1q	2q	3q
ILSVR2010 $m = 1000$	1vsA	**91.80**	13435	581.43	1000	1000	1000	1000	1000	1000
	FT	95.77	**1051**	**24.50**	**10**	10	10	10	10	10
	$\text{PCT}_{\epsilon=0.5}$	96.92	1148	27.18	**10**	10	10	10	10	10
	$\text{PCT}_{\epsilon=0.25}$	94.99	1148	43.60	22	10	30	21	21	21
	$\text{PCT}_{\epsilon=0.0}$	92.93	1148	69.56	55	10	192	38	52	69
	$\text{PCT}^{*}_{\epsilon=0.0}$	92.66	2379	109.55	94	17	237	75	91	111

Top-5 multi-class classification

		1-P@5	t_{train}	t_{test}	A	min	max	1q	2q	3q
ILSVR2010 $m = 1000$	1vsA	**80.25**	13435	581.43	1000	1000	1000	1000	1000	1000
	$\text{PCT}_{\epsilon=0.5}$	90.1	**1148**	**75.08**	**41**	39	55	42	42	42
	$\text{PCT}_{\epsilon=0.25}$	88.39	**1148**	89.05	55	39	90	52	55	59
	$\text{PCT}_{\epsilon=0.0}$	81.39	**1148**	124.06	100	15	251	79	97	118
	$\text{PCT}^{*}_{\epsilon=0.0}$	**80.25**	2379	106.75	88	14	233	70	85	103

* – SGD with 25 epochs (for other methods 5 epochs have been used).

FT vs. PCT

Filter Trees

- Logarithmic prediction time.
- Can be used with any binary classifier.
- Training is more demanding.
- Filtering may reduce the number of training examples in the top levels of the tree.
- Does not work for top-k classification.

Probabilistic Classifier Trees

- The prediction scales from logarithmic to linear time.
- Requires probabilistic classifiers.
- Training can be parallelized.
- The ϵ-approximate inference may not work properly for noisy problems.
- Can be adapted for top-k classification.

Conclusions

- Take-away message:
  - Label tree structures for efficient classification.
  - Standard approach is inconsistent.
  - FT are consistent for $0/1$ loss and cost-sensitive classification.
  - PCT are consistent for top-k classification.
- For more check:

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