To Search or to Filter? Consistent Label Tree Classifiers for Efficient Classification

Arkadiusz Jachnik

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

PL-SIGML, Wroclaw, March 17-18, 2014

This is a join work with Krzysztof Dembczynski.

Multi-Class and Multi-Label Classification

- **Multi-class classification** (MCC): a classification task with **one target** variable belonging to more than two classes.
- **Multi-label classification** (MLC): variant of the classification problem where **none, one or more** class labels are assigned to single instances simultaneously.

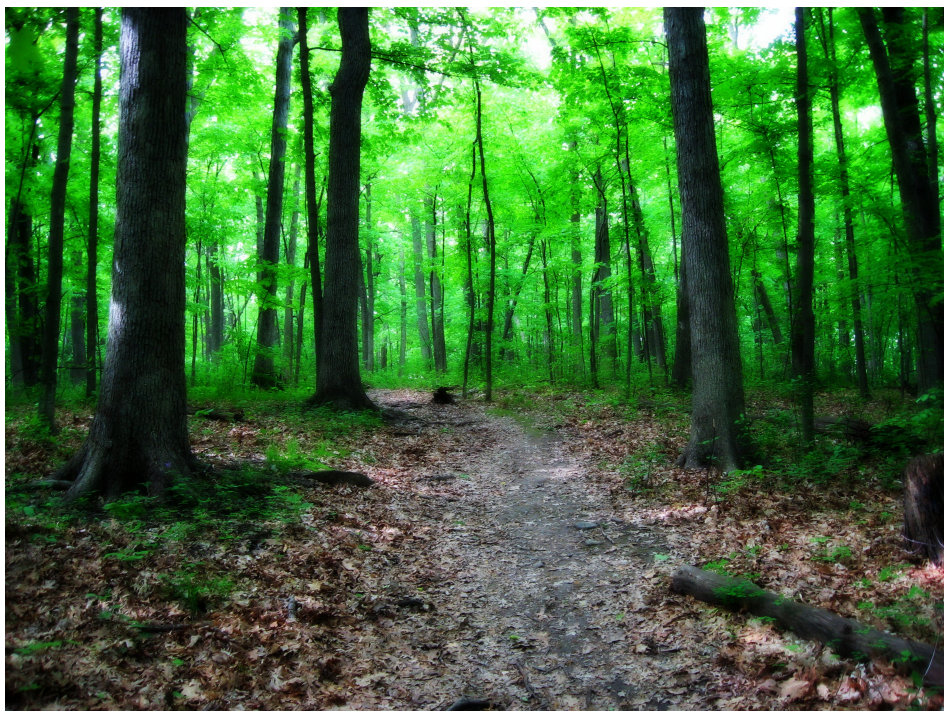
Example

Multi-class classification

Target: forest `OR` path `OR` dog `OR` ...

Multi-label classification

Target 1:	forest	yes/no
Target 2:	path	yes/no
Target 3:	dog	yes/no
...	...	...



MCC and MLC More Formally

- **Multi-class classification:** For a feature vector $\boldsymbol{x}$ predict accurately one response y using a function $h(\boldsymbol{x})$:

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_p) \xrightarrow{h(\boldsymbol{x})} y \in \{1, \ldots, k\}$$

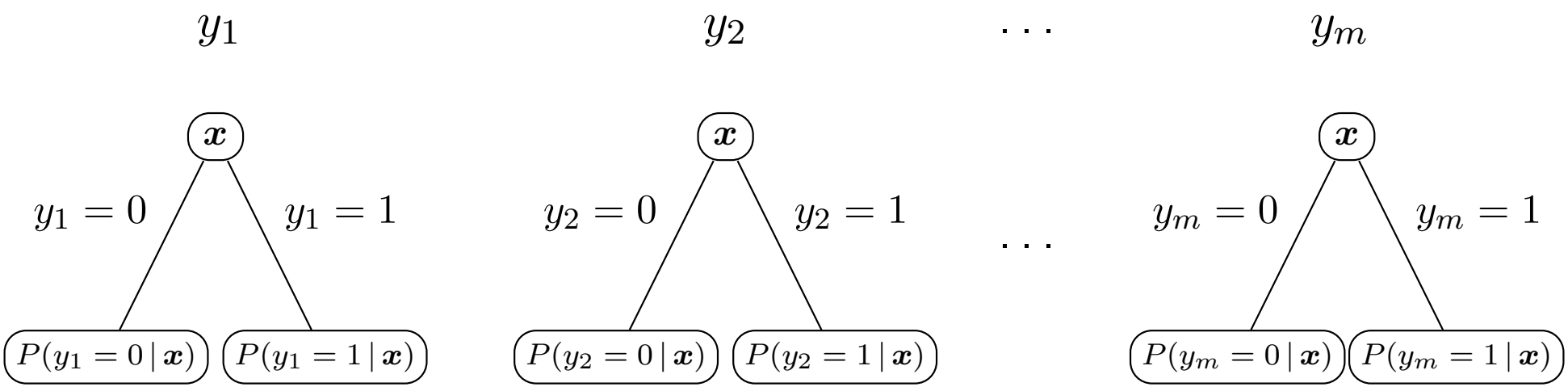
- **Multi-label classification:** For a feature vector $\boldsymbol{x}$ predict accurately a vector of responses $\boldsymbol{y}$ using a function $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x}))$:

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_p) \xrightarrow{\boldsymbol{h}(\boldsymbol{x})} \boldsymbol{y} = (y_1, y_2, \ldots, y_m) \in \{0, 1\}^m$$

Multi-Label Classification – Simple Approach

Binary relevance: Decomposes the problem to m independent binary classification problems:

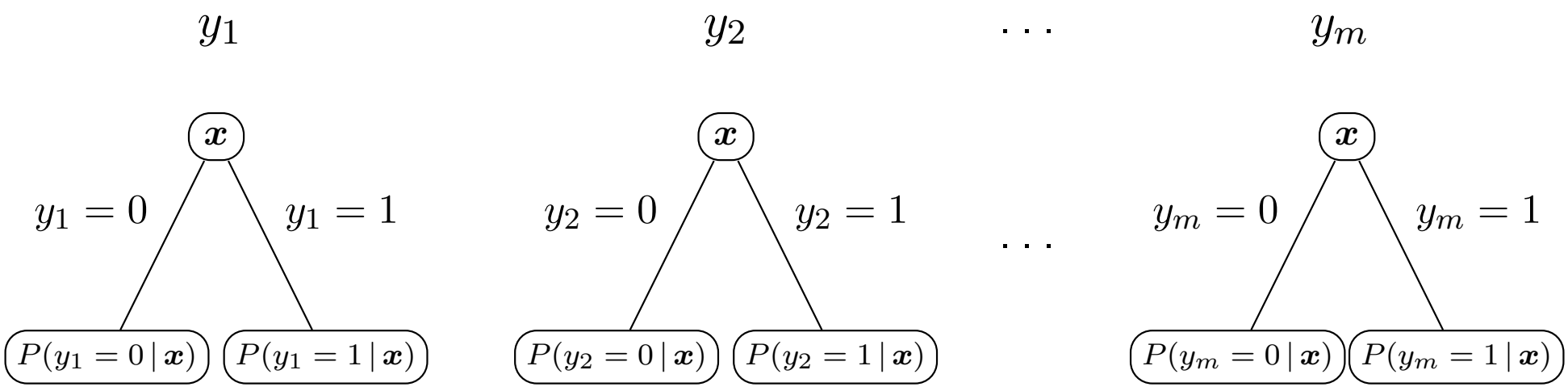
$$(\boldsymbol{x}, \boldsymbol{y}) \longrightarrow (\boldsymbol{x}, y = y_i), \quad i = 1, \ldots, m$$



Multi-Label Classification – Simple Approach

Binary relevance: Decomposes the problem to m independent binary classification problems:

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

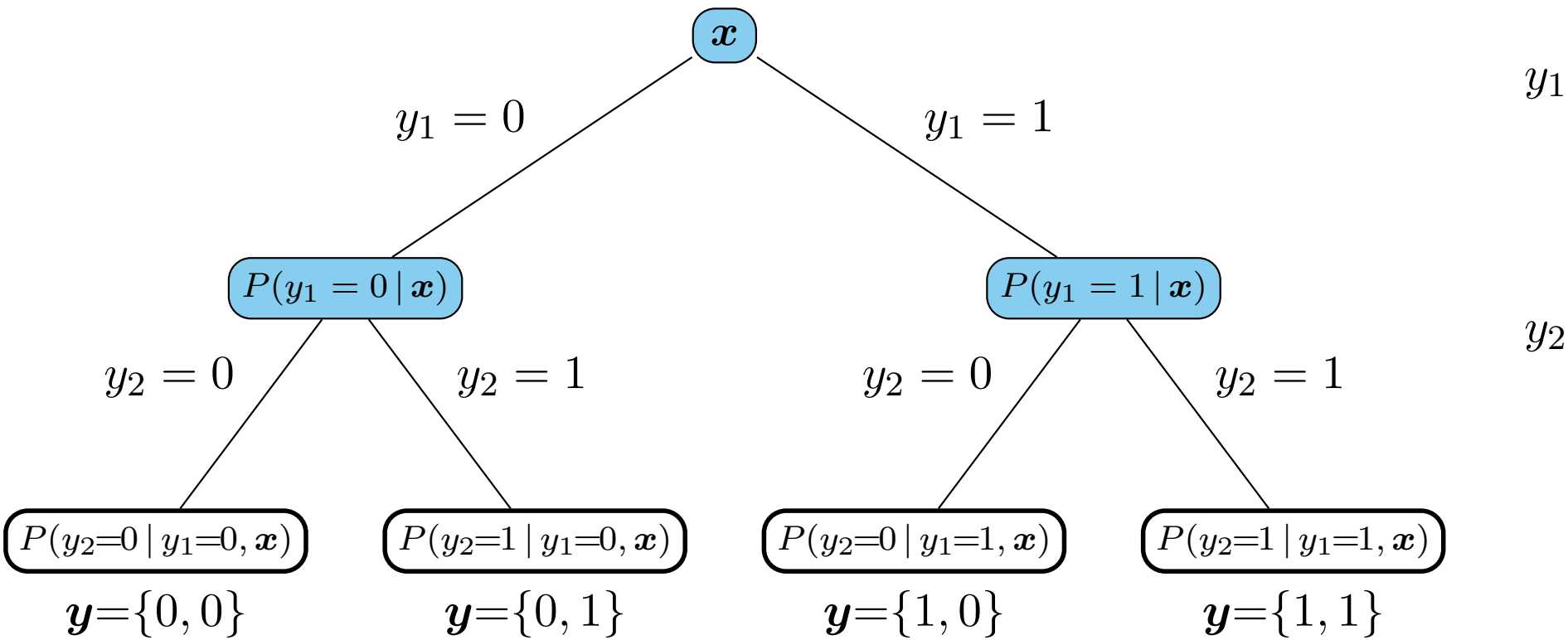


Challenges:

- How to model dependencies between labels?
- What about loss function defined over the binary vectors?

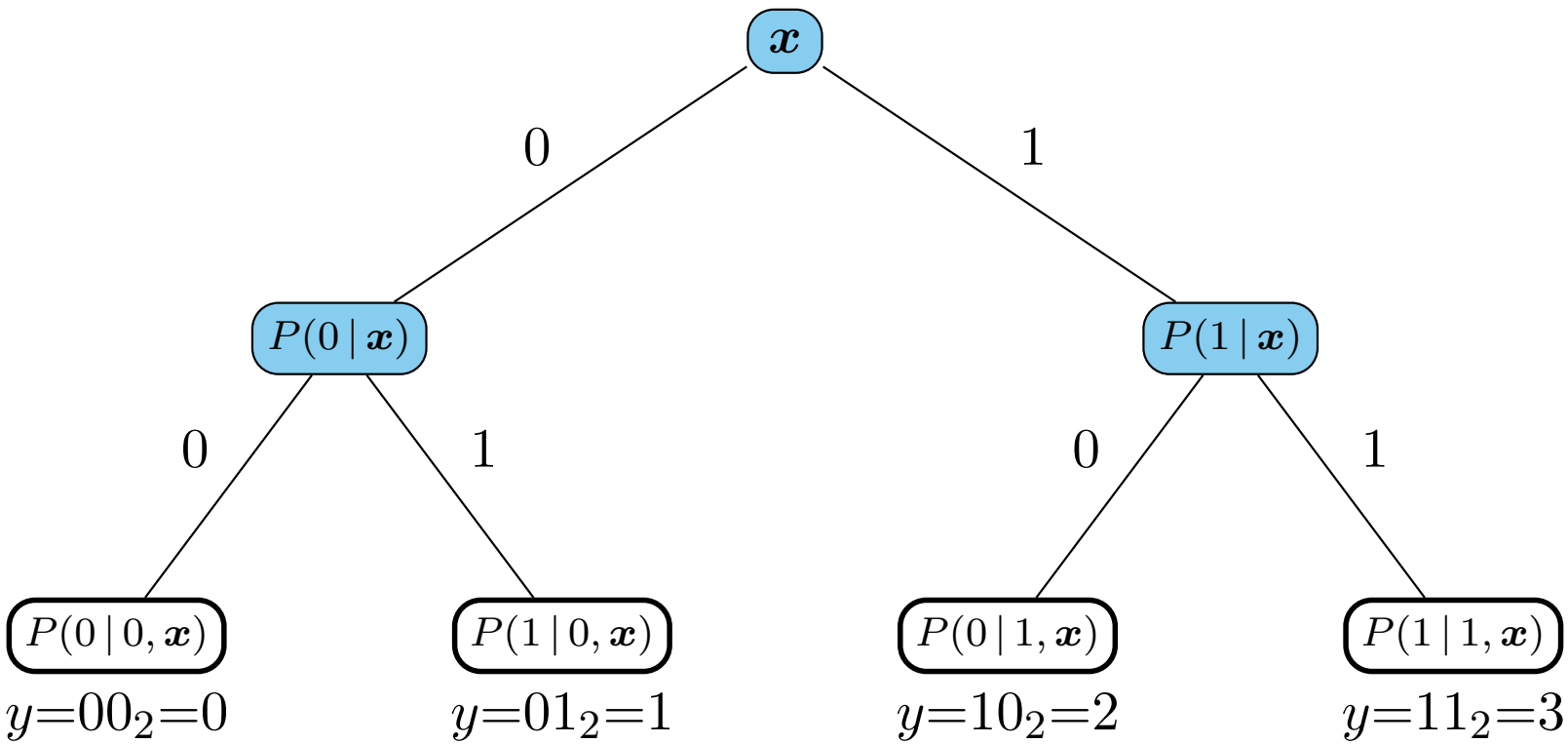
Label Tree Classifiers for MLC

- Example: $m = 2$, $\mathcal{Y} = \{0, 1\}^2$
- $2^m - 1$ classifiers on the tree (m levels); m-classifier trick with m classifiers (one per level)
- 2^m leaves of the all possible vectors of responses



Label Tree Classifiers for MCC

- We assign each class an integer from 0 to $k-1$ and code it by its binary representation on m bits.
- Example: $k = 4$, $\mathcal{Y} = \{0, 1, 2, 3\}$
- $k-1$ **classifiers** on the tree; trick with $\log k$ classifiers (one per level)
- k leaves, one for each class



Classifier Regret

- The prediction accuracy of $\boldsymbol{h}$ is measured in terms of its **risk**, that is, its expected loss

$$L(\boldsymbol{h}, P) = \mathbb{E}\left[\ell(\boldsymbol{Y}, \boldsymbol{h}(\boldsymbol{X}))\right] = \int \ell(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x}))\, dP(\boldsymbol{x}, \boldsymbol{y})\,,$$

  where ℓ is a **loss function**.
- Here, we consider the 0/1 loss:

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

Classifier Regret

- The optimal **Bayes classifier** minimizes the risk:

$$\boldsymbol{h}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{h}\in\mathcal{Y}} \int \sum_{\boldsymbol{y}\in\mathcal{Y}} \ell_{0/1}(\boldsymbol{y}, \boldsymbol{h}) P(\boldsymbol{y} \mid \boldsymbol{x})\, dP(\boldsymbol{x})\,.$$

- The Bayes classifier for the 0/1 loss has the form of a joint mode:

$$\boldsymbol{h}^*(\boldsymbol{x}) = \arg\max_{\boldsymbol{y}} P(\boldsymbol{y} \mid \boldsymbol{x})$$

- We note that $\boldsymbol{h}^*$ is in general not unique. However, the risk of $\boldsymbol{h}^*$, denoted $L^*(P)$, is unique, and is called the **Bayes risk**.
- The **regret** of $\boldsymbol{h}$ on $P(\boldsymbol{X}, \boldsymbol{Y})$ is defined as:

$$\mathrm{reg}_{0/1}(\boldsymbol{h}, P) = L_{0/1}(\boldsymbol{h}, P) - L^*_{0/1}(P)$$

- The goal is to train a classifier $\boldsymbol{h}$ with a small regret, ideally equal to zero.

Definition of Consistency

We say that the reduction algorithm is consistent if zero regret solution on the reduced problem implies zero regret on the original problem.

Definition of Consistency

We say that the reduction algorithm is consistent if zero regret solution on the reduced problem implies zero regret on the original problem.

- The Label Tree Classifiers ensure consistency when the perfect (zero regret) classifier in each node of the tree guarantees the perfect global classification of a test example.

Label Tree Classifiers – Conventional Training and Classification

Definition of Consistency

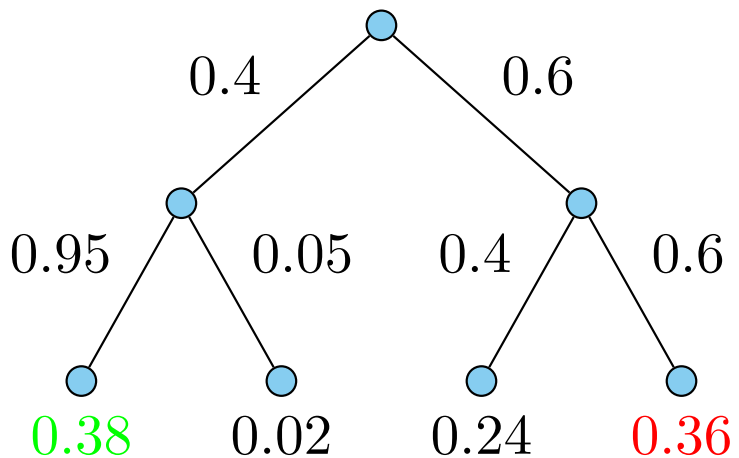
We say that the reduction algorithm is consistent if zero regret solution on the reduced problem implies zero regret on the original problem.

- The Label Tree Classifiers ensure consistency when the perfect (zero regret) classifier in each node of the tree guarantees the perfect global classification of a test example.
- The conventional training and classification procedures for Label Tree Classifiers do not ensure consistency!

Definition of Consistency

We say that the reduction algorithm is consistent if zero regret solution on the reduced problem implies zero regret on the original problem.

- The Label Tree Classifiers ensure consistency when the perfect (zero regret) classifier in each node of the tree guarantees the perfect global classification of a test example.
- The conventional training and classification procedures for Label Tree Classifiers do not ensure consistency!



Can Label Tree Classifiers be consistent?

Can Label Tree Classifiers be consistent?

Two approaches...

Filter Trees (FT) [1]

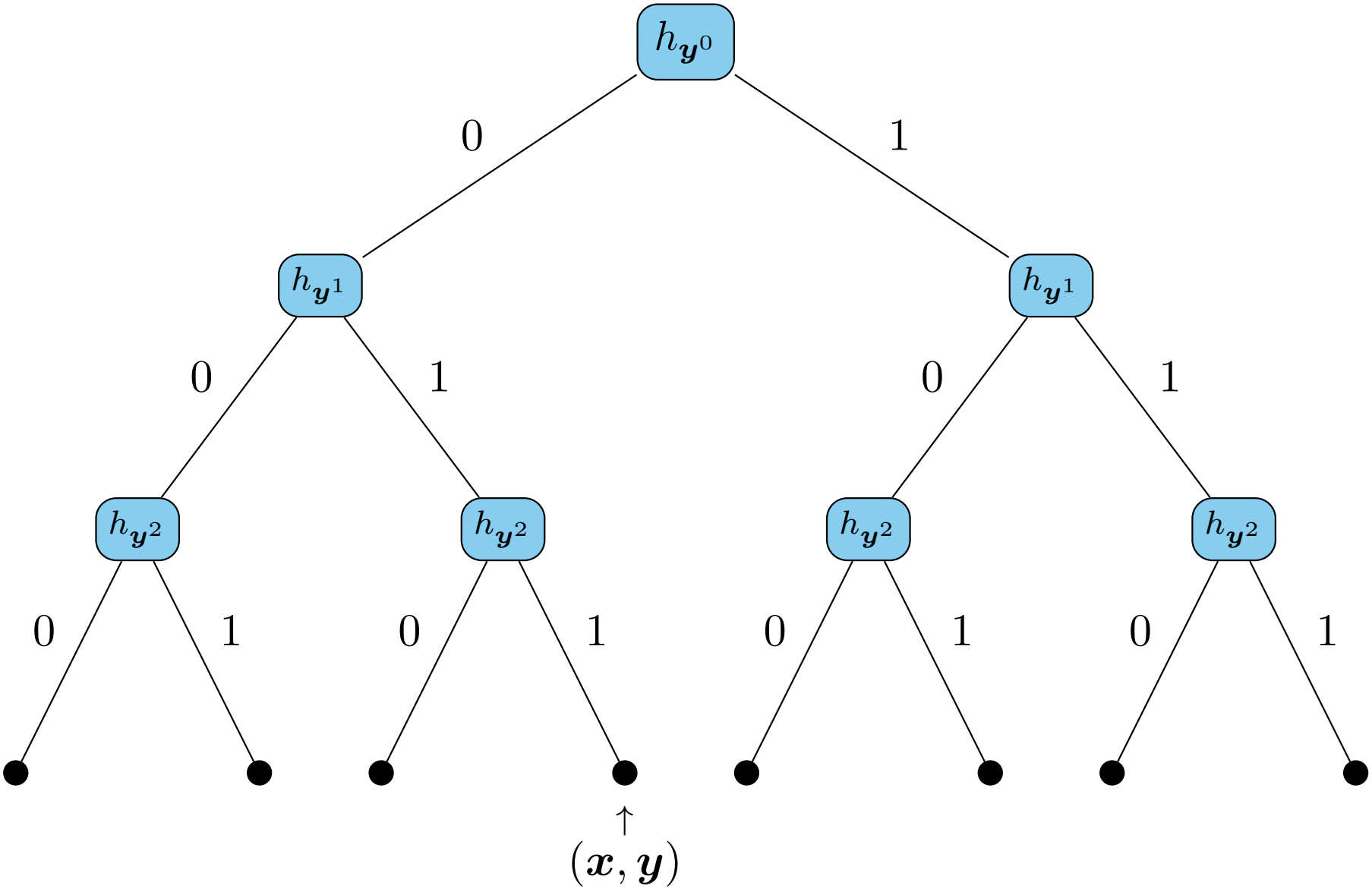
- **Bottom-up** learning algorithm to train binary classifiers of the tree.
- Single **elimination tournament** on the set of label combinations.
- The node classifiers $h_{\boldsymbol{y}^i}$ are trained as before to predict y_{i+1}, where $\boldsymbol{y}^i = (y_1, \ldots, y_i)$.
- FT implicitly transforms the underlying distribution P over multi-label examples into a **specific distribution** $P^{FT}_{\boldsymbol{y}^i}$ over binary examples at each node $\boldsymbol{y}^i$.
- This transformation for the 0/1 loss **filters out** all examples that are misclassified by the lower-level classifiers.
- $h_{\boldsymbol{y}^i}(\boldsymbol{x})$ predicts y_{i+1} given that all classifiers below predict the subsequent labels correctly:

$$h_{\boldsymbol{y}^i} : \boldsymbol{x} \mapsto (y_{i+1} \,|\, y_{j+1} = h_{\boldsymbol{y}^j}(\boldsymbol{x}) : j = i+1, \ldots, m)$$

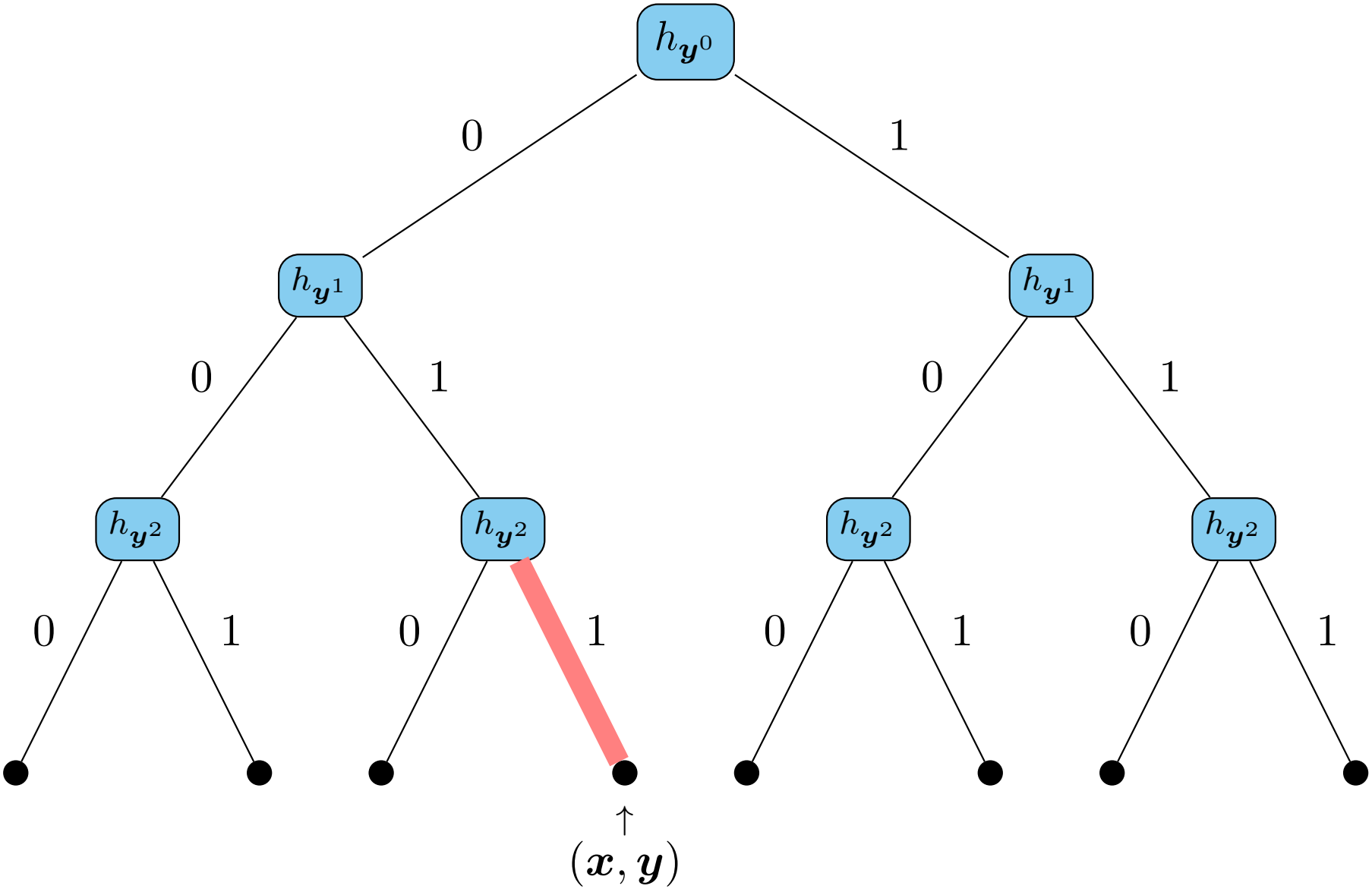
- The inference procedure of FT is straight-forward and uses the **greedy search** (which is sufficient for obtaining consistent predictions).

[1] A. Beygelzimer, J. Langford, and P.D. Ravikumar, *Error-correcting tournaments*. In ALT, pp. 247262, 2009

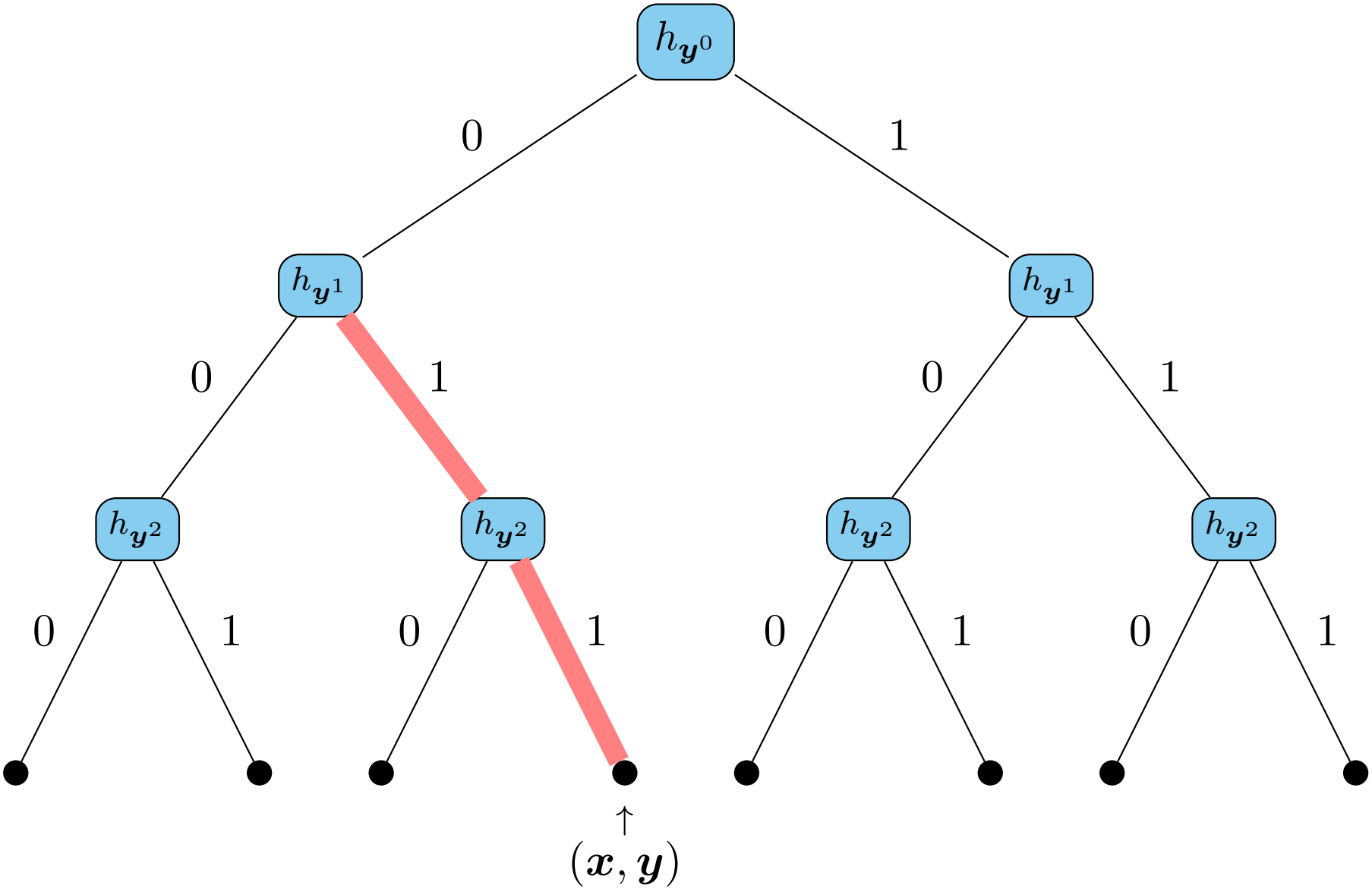
Filter Trees – Example



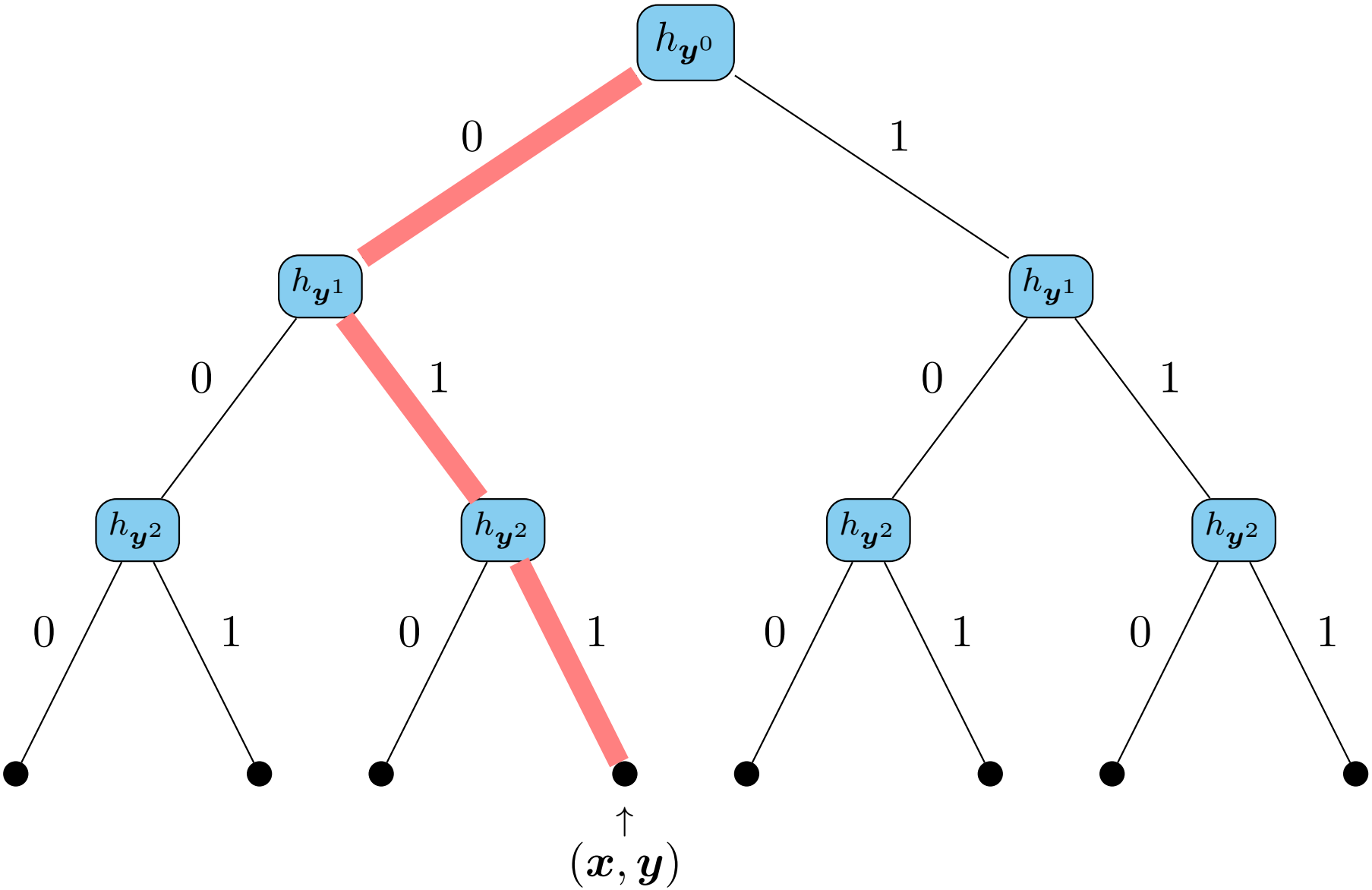
Filter Trees – Example



Filter Trees – Example



Filter Trees – Example



Filter Trees – Consistency

Theorem [2]

For all binary classifiers h and distributions P,

$$\mathrm{reg}_{0/1}(\boldsymbol{h}, P) \leq m\mathrm{reg}_{0/1}(h, P^{FT})$$

[2] A. Beygelzimer, J. Langford, and P.D. Ravikumar, *Error-correcting tournaments*. In ALT, pp. 247262, 2009

Probabilistic Classifier Trees (PCT)

- Probabilistic binary classifiers on the tree (e.g., logistic regression).
- Each node classifier $h_{\boldsymbol{y}^i}$ delivers estimates of conditional probability:

$$h_{\boldsymbol{y}^i} : \boldsymbol{x} \mapsto P(y_{i+1} = 1 | \boldsymbol{x}, \boldsymbol{y}^i)$$

  We denote these estimates by $Q(y_{i+1} = 1 | \boldsymbol{x}, \boldsymbol{y}^i)$.
- Training phase as in the conventional Label Tree Classifiers (can be easily parallelized).
- More complex inference procedure (the greedy approach does not guarantee to find the optimal solution).

Probabilistic Classifier Trees – ϵ-approximate inference

Input: $\boldsymbol{x}$, priority lists: $\mathcal{Q} \leftarrow \{\boldsymbol{y}_0\}$, $\mathcal{K} \leftarrow \{\}$, $\epsilon \leftarrow 2^{-c}$ with $c \leq m$
while $\mathcal{Q} \neq \emptyset$ **do**
  $\boldsymbol{v} \leftarrow$ pop first element in $\mathcal{Q}$
  if $\boldsymbol{v}$ is a leaf **then** delete all elements in $\mathcal{K}$ and **break**
  $\boldsymbol{v}_1 \leftarrow (\boldsymbol{v}, 1)$ (left child of $\boldsymbol{v}$) and $\boldsymbol{v}_0 \leftarrow (\boldsymbol{v}, 0)$ (right child of $\boldsymbol{v}$)
  compute $Q(\boldsymbol{v}_1 \,|\, \boldsymbol{x})$ and $Q(\boldsymbol{v}_0 \,|\, \boldsymbol{x})$ recursively from $Q(\boldsymbol{v} \,|\, \boldsymbol{x})$
  if $Q(\boldsymbol{v}_1 \,|\, \boldsymbol{x}) \geq \epsilon$ **then** add $\boldsymbol{v}_1$ to $\mathcal{Q}$ sorted in desc. order of Q
  if $Q(\boldsymbol{v}_0 \,|\, \boldsymbol{x}) \geq \epsilon$ **then** add $\boldsymbol{v}_0$ to $\mathcal{Q}$ sorted in desc. order of Q
  if $\boldsymbol{v}_1$ and $\boldsymbol{v}_0$ are not in $\mathcal{Q}$ **then** add $\boldsymbol{v}$ to $\mathcal{K}$ in desc. order of Q
end while
$\epsilon \leftarrow 0$
while $\mathcal{K} \neq \emptyset$ **do**
  $\boldsymbol{v}' \leftarrow$ pop first element in $\mathcal{K}$
  $\boldsymbol{v}' \leftarrow$ apply greedy search downward on $\boldsymbol{v}'$
  if $Q(\boldsymbol{v}' \,|\, \boldsymbol{x}) \geq \epsilon$ **then** $\boldsymbol{v} \leftarrow \boldsymbol{v}'$ and $\epsilon \leftarrow Q(\boldsymbol{v}' \,|\, \boldsymbol{x})$
end while
return $\boldsymbol{v} = (y_1, ..., y_m)$

Probabilistic Classifier Trees – Consistency

Theorem

For all binary probabilistic classifiers h and distributions the total regret of P, PCT with ϵ-approximate inference:

$$\mathrm{reg}_{0/1}(\boldsymbol{h}, P) \leq \sqrt{2m\overline{\mathrm{reg}}_{\log}(h, P)} + 2^{-c} - 2^{-m}$$

Experimental Results

		0/1[%]	t_{train}	t_{test}	A	min	max	1q	2q	3q	P@5
Yeast $m = 14$	BR	85.06	18	0.18	**14**	14	14	14	14	14	–
	FT	78.30	**16**	**0.17**	**14**	14	14	14	14	14	–
	PCT$_{\epsilon=0.5}$	79.17	21	0.19	**14**	14	14	14	14	14	–
	PCT$_{\epsilon=0.25}$	77.54	21	0.32	23	14	42	15	26	28	–
	PCT$_{\epsilon=0.0}$	**76.34**	21	0.37	25	14	61	17	23	30	–
Enron $m = 53$	BR	86.36	**66**	1.62	**53**	53	53	53	53	53	–
	FT	82.56	76	1.57	**53**	53	53	53	53	53	–
	PCT$_{\epsilon=0.5}$	83.77	92	**1.49**	**53**	53	53	53	53	53	–
	PCT$_{\epsilon=0.25}$	82.73	92	1.82	72	54	147	54	54	58	–
	PCT$_{\epsilon=0.0}$	**81.87**	92	3.12	144	53	888	73	106	175	–
Mediamill $m = 101$	BR	92.96	**4947**	**26.20**	**101**	101	101	101	101	101	–
	FT	**89.86**	7551	27.76	**101**	101	101	101	101	101	–
	PCT$_{\epsilon=0.5}$	90.33	8695	30.73	**101**	101	101	101	101	101	–
	PCT$_{\epsilon=0.25}$	90.12	8695	40.86	125	101	260	102	102	102	–
	PCT$_{\epsilon=0.0}$	90.07	8695	54.51	208	101	1756	149	182	228	–
ILSVR2010 $k = 1000$	1vsA	**91.80**	13435	581.43	1000	1000	1000	1000	1000	1000	19.75
	FT	95.77	**1051**	**24.50**	**10**	10	10	10	10	10	–
	PCT$_{\epsilon=0.5}$	96.92	1148	27.18	**10**	10	10	10	10	10	3.08
	PCT$_{\epsilon=0.25}$	94.99	1148	43.60	22	10	30	21	21	21	5.55
	PCT$_{\epsilon=0.0}$	92.93	1148	69.56	55	10	192	38	52	69	15.16
	PCT'$_{\epsilon=0.0}$	92.66	2379	109.55	94	17	237	75	91	111	19.75

Conclusions

Filter Trees

- Logarithmic prediction time in the number of classes or label combinations.
- FT can be used with any type of binary classifiers.
- To guarantee the consistency of the greedy prediction it requires more demanding training.
- Filtering may reduce the number of training examples in the top levels of the tree.
- There is no option to predict top classes with the highest conditional probabilities.

Probabilistic Classifier Trees

- The prediction can be longer than for FT.
- The complexity of the classification is connected with the noise of data.
- PCT requires the use of probabilistic classifiers.
- The learning is much simpler and can be easily parallelized.
- The probabilistic nature allows to deliver a list of top classes.

Thank you!

Questions?

For more check: `www.cs.put.poznan.pl/ajachnik`
or contact me: `ajachnik@cs.put.poznan.pl`

This project is partially supported by the Foundation of Polish Science under the Homing Plus programme, co-financed by the European Regional Development Fund.

INNOVATIVE ECONOMY
NATIONAL COHESION STRATEGY

FNP
Foundation for Polish Science

EUROPEAN UNION
EUROPEAN REGIONAL
DEVELOPMENT FUND