Extreme Multi-label Classification for Information Retrieval XMLC4IR Tutorial at ECIR 2018

Rohit Babbar[1] and Krzysztof Dembczyński[2]

[1]Aalto University, Finland,
[2]Poznań University of Technology, Poland

European Conference on Information Retrieval
Grenoble, France, March 26, 2018

- **Rohit Babbar**:
  - **Affiliation**: University Aalto
  - **Previous affiliations**: Max-Planck Institute for Intelligent Systems, Université Grenoble Alpes
  - **Main interests**: extreme classification, multi-label classification, multi-class classification, text classification

- **Krzysztof Dembczyński**:
  - **Affiliation**: Poznan University of Technology
  - **Previous affiliations**: Marburg University
  - **Main interests**: extreme multi-machine learning, multi-label classification, label tree algorithms, learning theory

Agenda

1. Extreme classification: applications and challenges
2. Algorithms
   - Label embeddings
   - Smart 1-vs-All approaches
   - Tree-based methods
   - Label filtering/maximum inner product search
3. Live demonstration

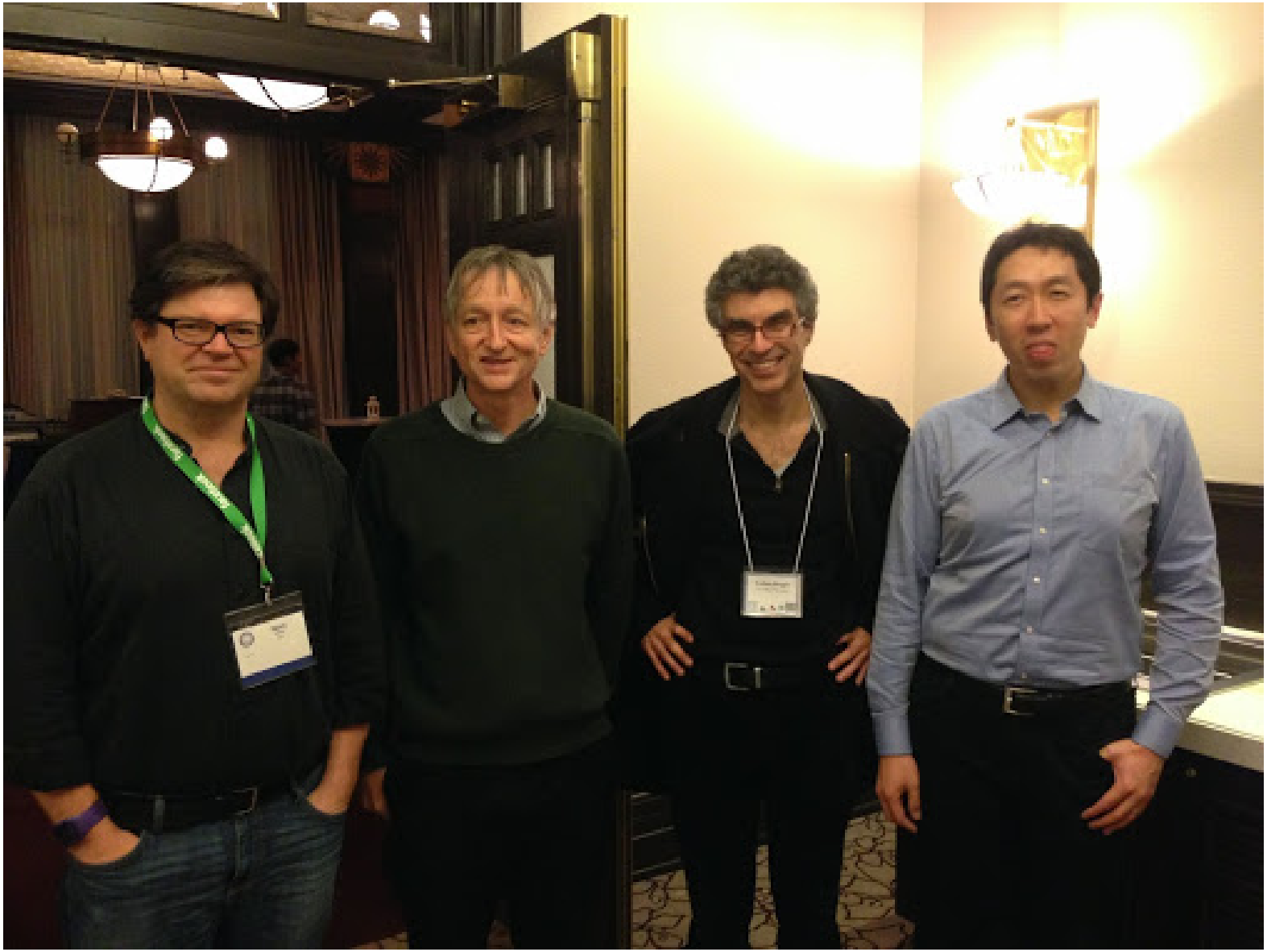
Webpage: `http://www.cs.put.poznan.pl/kdembczynski/xmlc-tutorial-ecir-2018/`

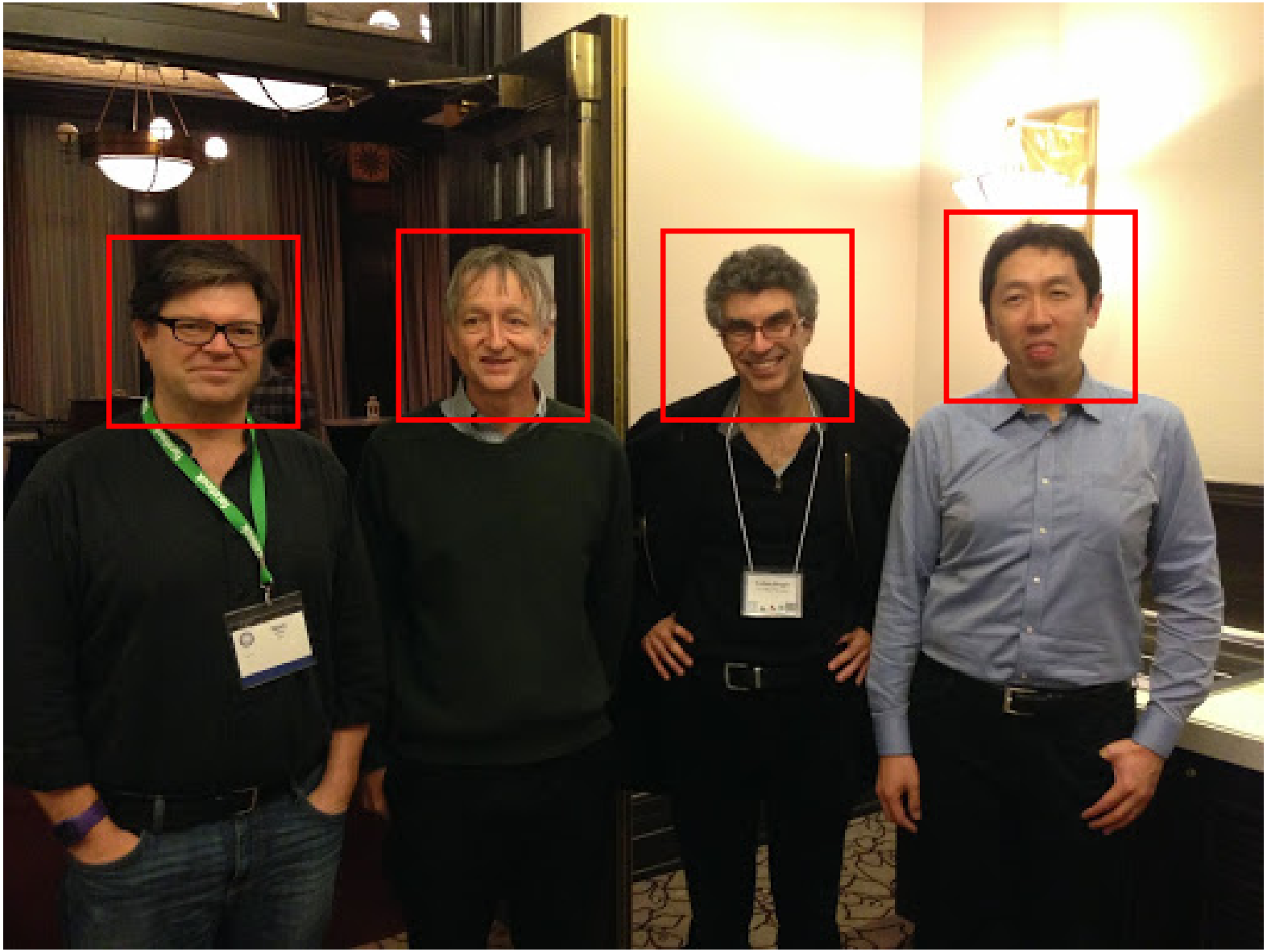
Agenda

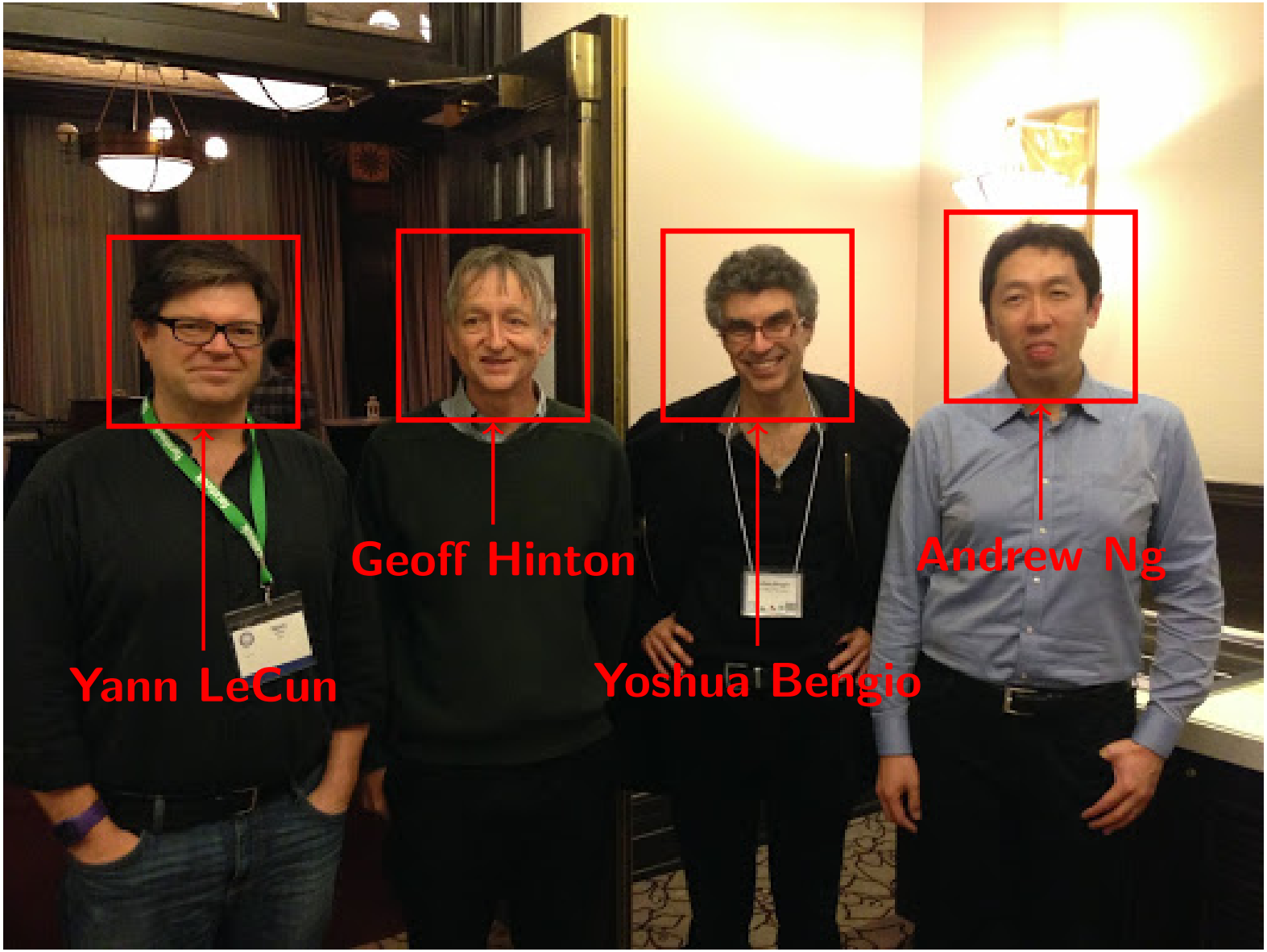
1. **Extreme classification: applications and challenges**
2. Algorithms
   - Label embeddings
   - Smart 1-vs-All approaches
   - Tree-based methods
   - Label filtering/maximum inner product search
3. Live demonstration

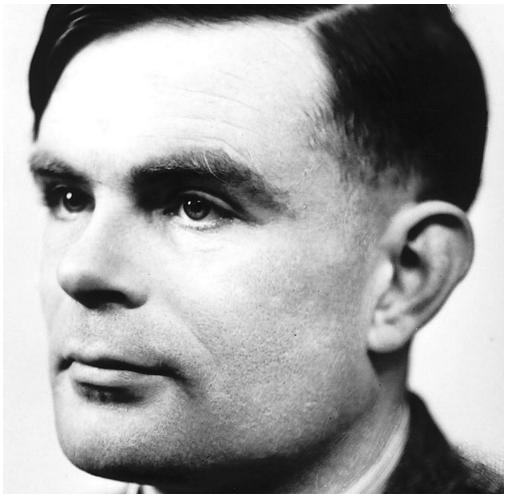
Webpage: `http://www.cs.put.poznan.pl/kdembczynski/xmlc-tutorial-ecir-2018/`

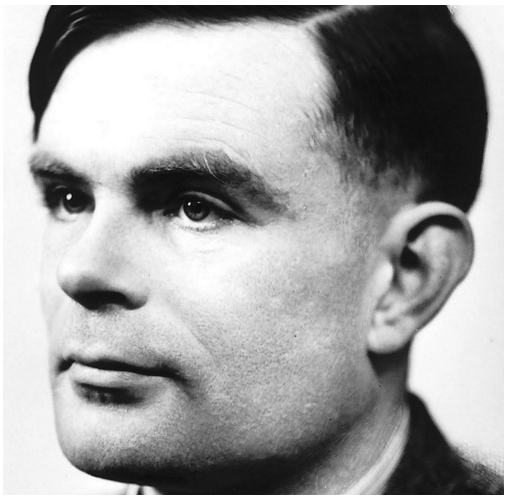
Extreme multi-label classification is a problem of **labeling** an item with a **small** set of tags out of an **extremely large** number of potential tags.



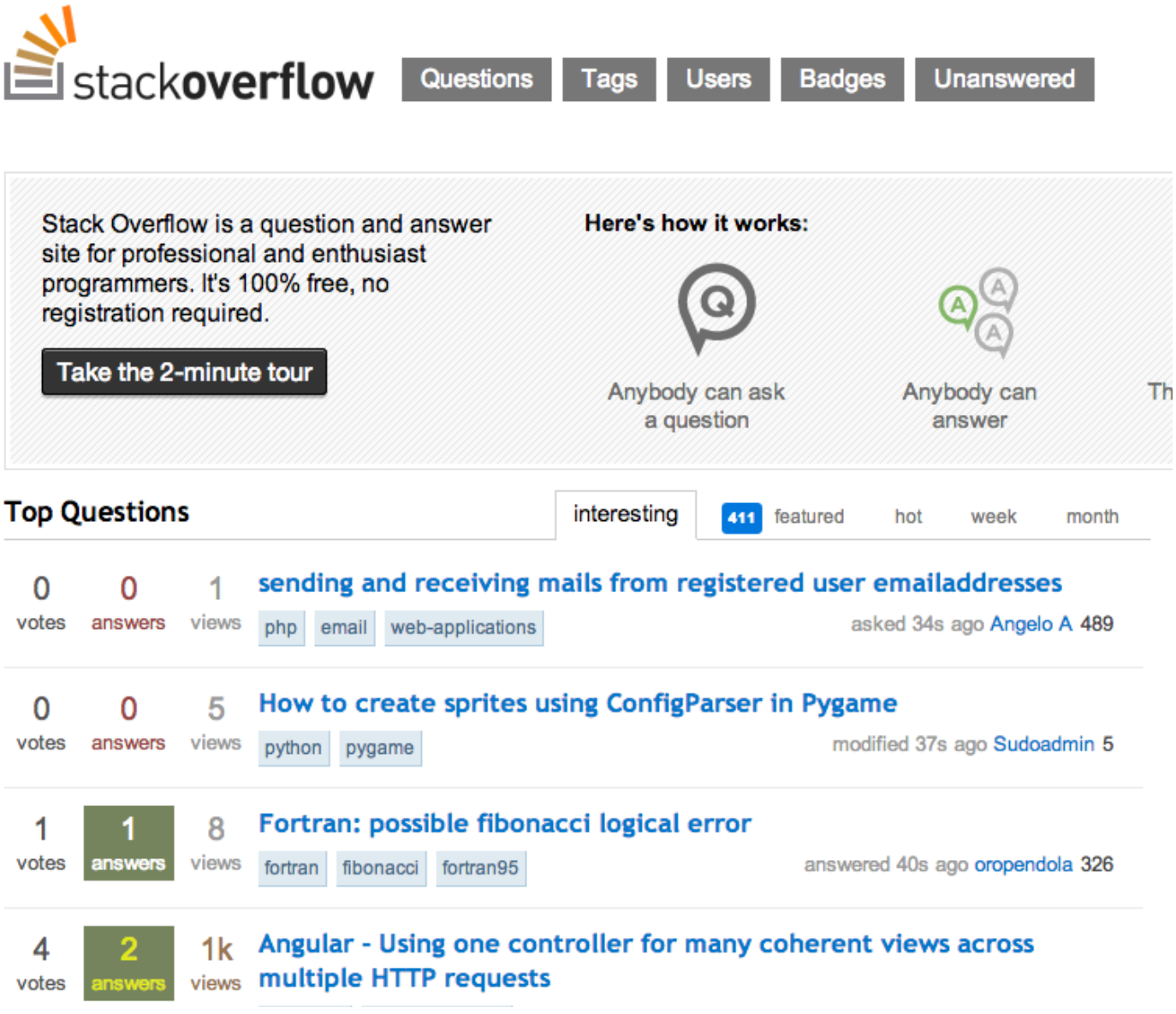



Yann LeCun
Geoff Hinton
Yoshua Bengio
Andrew Ng

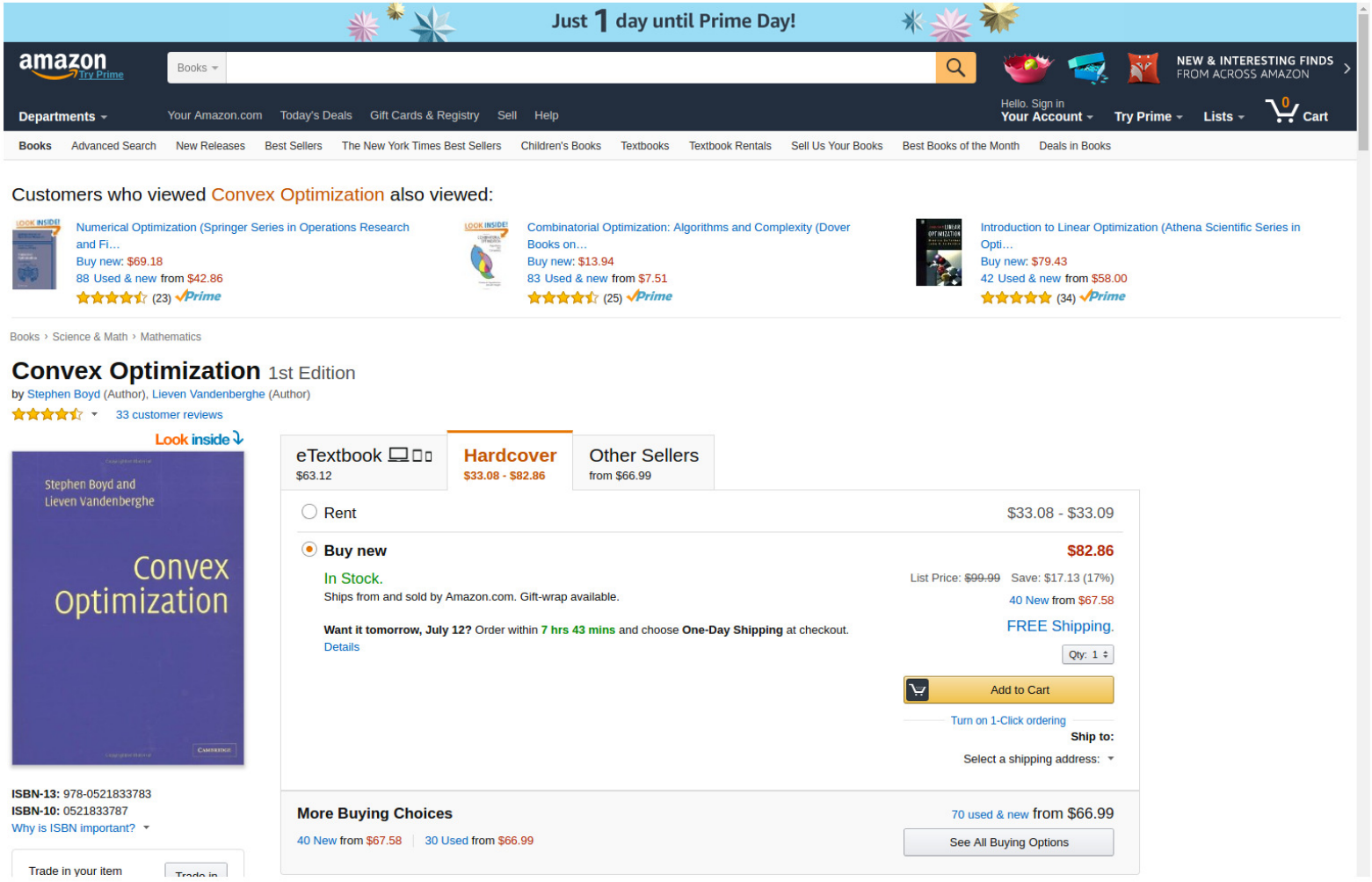




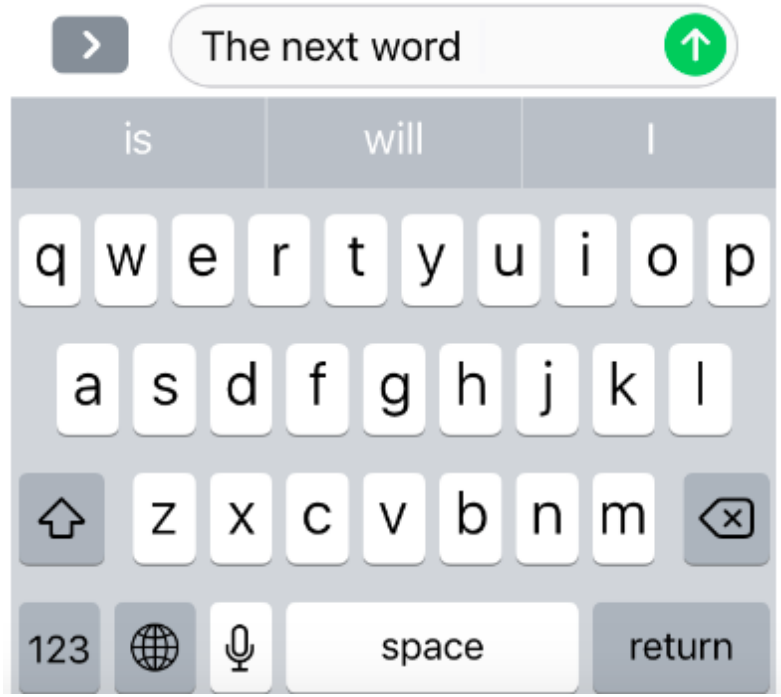
Alan Turing, 1912 births, 1954 deaths
20th-century mathematicians, 20th-century philosophers
Academics of the University of Manchester Institute of Science and Technology
Alumni of King's College, Cambridge Artificial intelligence researchers
Atheist philosophers, Bayesian statisticians, British cryptographers, British logicians
British long-distance runners, British male athletes, British people of World War II
Computability theorists, Computer designers, English atheists
English computer scientists, English inventors, English logicians
English long-distance runners, English mathematicians
English people of Scottish descent, English philosophers, Former Protestants
Fellows of the Royal Society, Gay men
Government Communications Headquarters people, History of artificial intelligence
Inventors who committed suicide, LGBT scientists
LGBT scientists from the United Kingdom, Male long-distance runners
Mathematicians who committed suicide, Officers of the Order of the British Empire
People associated with Bletchley Park, People educated at Sherborne School
People from Maida Vale, People from Wilmslow
People prosecuted under anti-homosexuality laws, Philosophers of mind
Philosophers who committed suicide, Princeton University alumni, 1930-39
Programmers who committed suicide, People who have received posthumous pardons
Recipients of British royal pardons, Academics of the University of Manchester
Suicides by cyanide poisoning, Suicides in England, Theoretical computer scientists



New question $\Rightarrow$ Assignment/recommendation of users



Selected item $\Rightarrow$ Recommendation of top 3 items

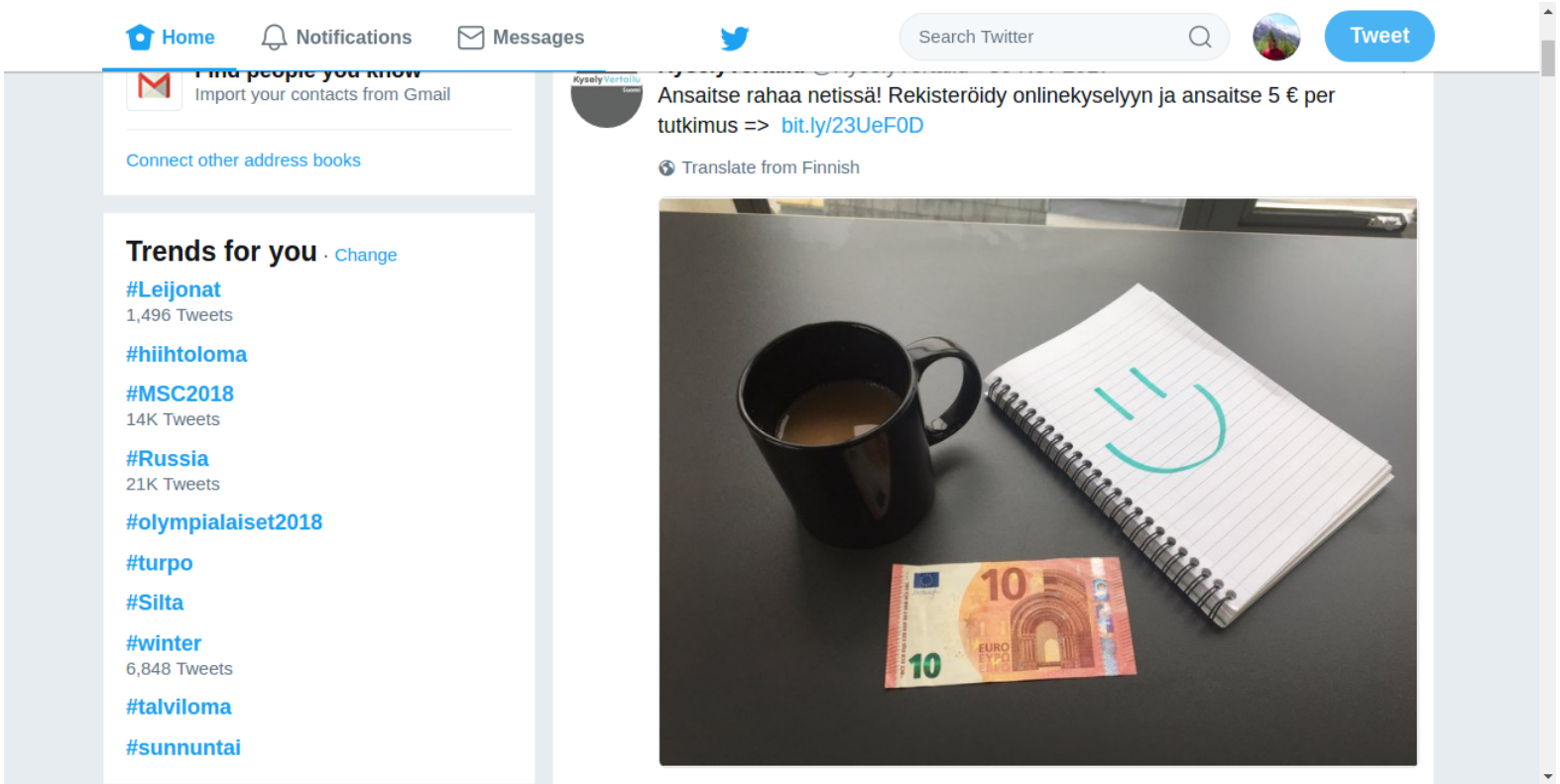


Sequence of words $\Rightarrow$ Recommendation of the next word

Possible bid phrases:

- Zurich car insurance
- Car insurance
- Auto insurance
- Vehicle insurance
- Electric car insurance

On-line ad $\Rightarrow$ Recommendation of queries to an advertiser



Suggestion of top Twitter Trends

Setting

- **Multi-class classification**:

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d \xrightarrow{h(\boldsymbol{x})} y \in \{1, \ldots, m\}$$

	x_1	x_2	$\ldots$	x_d	y
$\boldsymbol{x}$	4.0	2.5		-1.5	5

Setting

- **Multi-class classification**:

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d \xrightarrow{h(\boldsymbol{x})} y \in \{1, \ldots, m\}$$

	x_1	x_2	$\ldots$	x_d	y
$\boldsymbol{x}$	4.0	2.5		-1.5	5

- **Multi-label classification**:

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

	x_1	x_2	$\ldots$	x_d	y_1	y_2	$\ldots$	y_m
$\boldsymbol{x}$	4.0	2.5		-1.5	1	1		0

Extreme classification

Extreme classification $\Rightarrow$ a **large** number of **labels** m $(\geq 10^5)$

Extreme classification

Extreme classification $\Rightarrow$ a **large** number of **labels** m $(\geq 10^5)$

- **Predictive performance**:

Extreme classification

Extreme classification $\Rightarrow$ a **large** number of **labels** m $(\geq 10^5)$

- **Predictive performance**:
  - Performance measures: Hamming loss, prec@k, NDCG@k, Macro F

Extreme classification

Extreme classification $\Rightarrow$ a **large** number of **labels** m $(\geq 10^5)$

- **Predictive performance**:
  - Performance measures: Hamming loss, prec@k, NDCG@k, Macro F
  - Learning theory for large m

Extreme classification

Extreme classification $\Rightarrow$ a **large** number of **labels** m $(\geq 10^5)$

- **Predictive performance**:
  - Performance measures: Hamming loss, prec@k, NDCG@k, Macro F
  - Learning theory for large m
  - Training and prediction under limited time and space budged

Extreme classification

Extreme classification $\Rightarrow$ a **large** number of **labels** m $(\geq 10^5)$

- **Predictive performance**:
  - Performance measures: Hamming loss, prec@k, NDCG@k, Macro F
  - Learning theory for large m
  - Training and prediction under limited time and space budged
  - Learning with missing labels and positive-unlabeled learning

Extreme classification

Extreme classification $\Rightarrow$ a **large** number of **labels** m $(\geq 10^5)$

- **Predictive performance**:
  - Performance measures: Hamming loss, prec@k, NDCG@k, Macro F
  - Learning theory for large m
  - Training and prediction under limited time and space budged
  - Learning with missing labels and positive-unlabeled learning
  - Long-tail label distributions and zero-shot learning

Extreme classification

Extreme classification $\Rightarrow$ a **large** number of **labels** m $(\geq 10^5)$

- **Predictive performance**:
  - Performance measures: Hamming loss, prec@k, NDCG@k, Macro F
  - Learning theory for large m
  - Training and prediction under limited time and space budged
  - Learning with missing labels and positive-unlabeled learning
  - Long-tail label distributions and zero-shot learning
- **Computational complexity**:

Extreme classification

Extreme classification $\Rightarrow$ a **large** number of **labels** m $(\geq 10^5)$

- **Predictive performance**:
  - Performance measures: Hamming loss, prec@k, NDCG@k, Macro F
  - Learning theory for large m
  - Training and prediction under limited time and space budged
  - Learning with missing labels and positive-unlabeled learning
  - Long-tail label distributions and zero-shot learning
- **Computational complexity**:
  - time vs. space

Extreme classification

Extreme classification $\Rightarrow$ a **large** number of **labels** m $(\geq 10^5)$

- **Predictive performance**:
  - Performance measures: Hamming loss, prec@k, NDCG@k, Macro F
  - Learning theory for large m
  - Training and prediction under limited time and space budged
  - Learning with missing labels and positive-unlabeled learning
  - Long-tail label distributions and zero-shot learning
- **Computational complexity**:
  - time vs. space
  - #examples vs. #features vs. #labels

Extreme classification

Extreme classification $\Rightarrow$ a **large** number of **labels** m $(\geq 10^5)$

- **Predictive performance**:
  - Performance measures: Hamming loss, prec@k, NDCG@k, Macro F
  - Learning theory for large m
  - Training and prediction under limited time and space budged
  - Learning with missing labels and positive-unlabeled learning
  - Long-tail label distributions and zero-shot learning
- **Computational complexity**:
  - time vs. space
  - #examples vs. #features vs. #labels
  - training vs. validation vs. prediction

Computational challenges: naive solution

- Size of the problem:

Computational challenges: naive solution

- Size of the problem:
  - # examples: $n > 10^6$

Computational challenges: naive solution

- Size of the problem:
  - # examples: $n > 10^6$
  - # features: $d > 10^6$

Computational challenges: naive solution

- Size of the problem:
  - # examples: $n > 10^6$
  - # features: $d > 10^6$
  - # labels: $m > 10^5$

Computational challenges: naive solution

- Size of the problem:
  - # examples: $n > 10^6$
  - # features: $d > 10^6$
  - # labels: $m > 10^5$
- Naive solution: A dense linear model for each label (1-vs-All):

$$\hat{\boldsymbol{y}} = \mathbf{W}^\top \boldsymbol{x}$$

Computational challenges: naive solution

- Size of the problem:
  - # examples: $n > 10^6$
  - # features: $d > 10^6$
  - # labels: $m > 10^5$
- Naive solution: A dense linear model for each label (1-vs-All):

$$\hat{\boldsymbol{y}} = \mathbf{W}^\top \boldsymbol{x}$$

  - Train time complexity:

Computational challenges: naive solution

- Size of the problem:
  - # examples: $n > 10^6$
  - # features: $d > 10^6$
  - # labels: $m > 10^5$
- Naive solution: A dense linear model for each label (1-vs-All):

$$\hat{\boldsymbol{y}} = \mathbf{W}^\top \boldsymbol{x}$$

  - Train time complexity: $> 10^{17}$

Computational challenges: naive solution

- Size of the problem:
  - # examples: $n > 10^6$
  - # features: $d > 10^6$
  - # labels: $m > 10^5$
- Naive solution: A dense linear model for each label (1-vs-All):

$$\hat{\boldsymbol{y}} = \mathbf{W}^\top \boldsymbol{x}$$

  - Train time complexity: $> 10^{17}$
  - Space complexity:

Computational challenges: naive solution

- Size of the problem:
  - # examples: $n > 10^6$
  - # features: $d > 10^6$
  - # labels: $m > 10^5$
- Naive solution: A dense linear model for each label (1-vs-All):

$$\hat{\boldsymbol{y}} = \mathbf{W}^\top \boldsymbol{x}$$

  - Train time complexity: $> 10^{17}$
  - Space complexity: $> 10^{11}$

Computational challenges: naive solution

- Size of the problem:
  - # examples: $n > 10^6$
  - # features: $d > 10^6$
  - # labels: $m > 10^5$
- Naive solution: A dense linear model for each label (1-vs-All):

$$\hat{\boldsymbol{y}} = \mathbf{W}^\top \boldsymbol{x}$$

  - Train time complexity: $> 10^{17}$
  - Space complexity: $> 10^{11}$
  - Test time complexity:

Computational challenges: naive solution

- Size of the problem:
  - # examples: $n > 10^6$
  - # features: $d > 10^6$
  - # labels: $m > 10^5$
- Naive solution: A dense linear model for each label (1-vs-All):

$$\hat{\boldsymbol{y}} = \mathbf{W}^\top \boldsymbol{x}$$

  - Train time complexity: $> 10^{17}$
  - Space complexity: $> 10^{11}$
  - Test time complexity: $> 10^{11}$

Computational challenges: naive solution

- **It does not have to be so hard**:

Computational challenges: naive solution

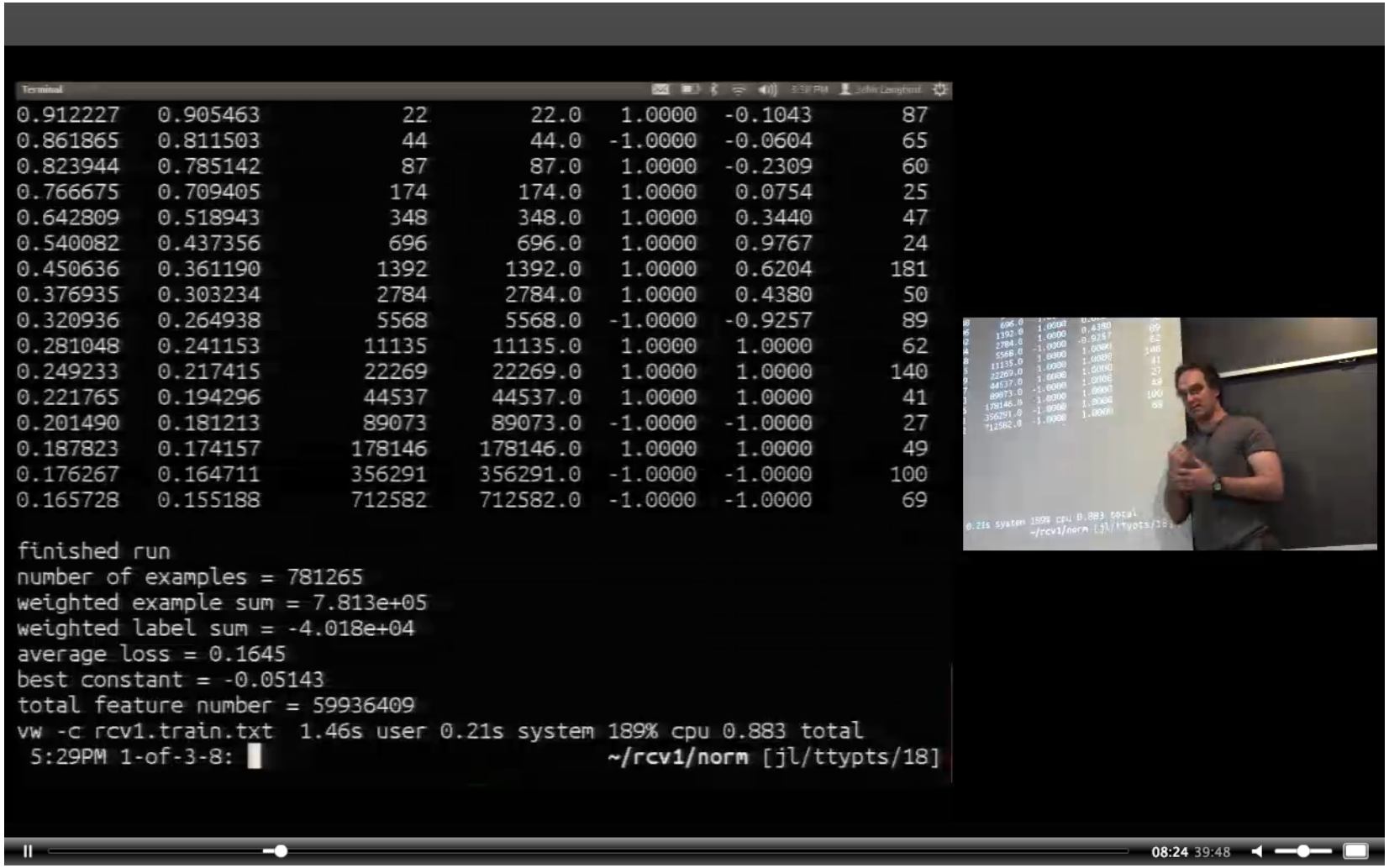
- **It does not have to be so hard**:
  - High performance computing resources available

Computational challenges: naive solution

- **It does not have to be so hard**:
  - High performance computing resources available
  - Large data $\longrightarrow$ sparse data (sparse features and labels)

Computational challenges: naive solution

- **It does not have to be so hard**:
  - High performance computing resources available
  - Large data $\longrightarrow$ sparse data (sparse features and labels)
  - Fast learning algorithms for standard learning problems exist



Vowpal Wabbit[1] at a lecture of John Langford[2]

[1] Vowpal Wabbit, `http://hunch.net/~vw`

[2] `http://cilvr.cs.nyu.edu/doku.php?id=courses:bigdata:slides:start`

Fast binary classification

- Data set: **RCV1**
- Predicted category: CCAT
- # training examples: 781 265
- # features: 60M
- Size: 1.1 GB
- Command line: `time vw -sgd rcv1.train.txt -c`
- Learning time: 1-3 secs on a laptop

Extreme classification with fast binary classifiers

- Running Vowpal Wabbit for 10^6 labels will still require:

$$1 \times 10^6\text{-}3 \times 10^6 \text{ secs} \approx 11.6\text{-}34.7 \text{ days}$$

Extreme classification with fast binary classifiers

- Running Vowpal Wabbit for 10^6 labels will still require:

$$1 \times 10^6\text{-}3 \times 10^6 \text{ secs} \approx 11.6\text{-}34.7 \text{ days}$$

- **Can we further reduce computational costs?**

Extreme classification with fast binary classifiers

- Running Vowpal Wabbit for 10^6 labels will still require:

$$1 \times 10^6\text{-}3 \times 10^6 \text{ secs} \approx 11.6\text{-}34.7 \text{ days}$$

- **Can we further reduce computational costs?**
  - Smart 1-vs-All approaches

Extreme classification with fast binary classifiers

- Running Vowpal Wabbit for 10^6 labels will still require:

$$1 \times 10^6\text{-}3 \times 10^6 \text{ secs} \approx 11.6\text{-}34.7 \text{ days}$$

- **Can we further reduce computational costs?**
  - Smart 1-vs-All approaches
  - Embeddings

Extreme classification with fast binary classifiers

- Running Vowpal Wabbit for 10^6 labels will still require:

$$1 \times 10^6\text{-}3 \times 10^6 \text{ secs} \approx 11.6\text{-}34.7 \text{ days}$$

- **Can we further reduce computational costs?**
  - Smart 1-vs-All approaches
  - Embeddings
  - Tree-based methods

Extreme classification with fast binary classifiers

- Running Vowpal Wabbit for 10^6 labels will still require:

$$1 \times 10^6\text{-}3 \times 10^6 \text{ secs} \approx 11.6\text{-}34.7 \text{ days}$$

- **Can we further reduce computational costs?**
  - Smart 1-vs-All approaches
  - Embeddings
  - Tree-based methods
  - Label filtering/maximum inner product search (MIPS)

Statistical challenges

- Learning theory for an extremely large number of labels:

Statistical challenges

- Learning theory for an extremely large number of labels:
  - **Statistical guarantees** for the **error** rate that **do not depend**, or depend very weakly (sublinearly), on the **total number of labels**.

Statistical challenges

- Learning theory for an extremely large number of labels:
  - **Statistical guarantees** for the **error** rate that **do not depend**, or depend very weakly (sublinearly), on the **total number of labels**.
  - The **bound** on the error rate could be expressed in terms of the average number of **positive labels** (which is certainly much less than the total number of labels).

Statistical challenges

- Learning theory for an extremely large number of labels:
  - **Statistical guarantees** for the **error** rate that **do not depend**, or depend very weakly (sublinearly), on the **total number of labels**.
  - The **bound** on the error rate could be expressed in terms of the average number of **positive labels** (which is certainly much less than the total number of labels).
  - Particular performance guarantees depend on the considered **loss function**.

Statistical challenges

- Learning theory for an extremely large number of labels:
  - **Statistical guarantees** for the **error** rate that **do not depend**, or depend very weakly (sublinearly), on the **total number of labels**.
  - The **bound** on the error rate could be expressed in terms of the average number of **positive labels** (which is certainly much less than the total number of labels).
  - Particular performance guarantees depend on the considered **loss function**.
  - **Different theoretical settings**: statistical learning theory, learning reductions, online learning.

Statistical challenges

- Training and prediction under limited time and space budget:

Statistical challenges

- Training and prediction under limited time and space budget:
  - **Restricted** computational **resources** (time and space) for both **training** and **prediction**.

Statistical challenges

- Training and prediction under limited time and space budget:
  - **Restricted** computational **resources** (time and space) for both **training** and **prediction**.
  - A **trade-off** between computational (time and space) **complexity** and the **predictive performance**.

Statistical challenges

- Training and prediction under limited time and space budget:
  - **Restricted** computational **resources** (time and space) for both **training** and **prediction**.
  - A **trade-off** between computational (time and space) **complexity** and the **predictive performance**.
  - By imposing hard constraints on time and space budget, the challenge is then to **optimize** the **predictive performance** of an algorithm under these **constraints**.

Statistical challenges

- Unreliable learning information:

Statistical challenges

- Unreliable learning information:
  - We **cannot** expect that all labels will be properly **checked** and **assigned** to training examples.

Statistical challenges

- Unreliable learning information:
  - We **cannot** expect that all labels will be properly **checked** and **assigned** to training examples.
  - Therefore we often deal with a problem of learning with **missing labels** or learning from **positive and unlabeled examples**.

Statistical challenges

- Performance measures:

Statistical challenges

- Performance measures:
  - Typical performance measures such as **0/1** or **Hamming** loss do **not fit** well to the extreme setting.

Statistical challenges

- Performance measures:
  - Typical performance measures such as **0/1** or **Hamming** loss do **not fit** well to the extreme setting.
  - Other measures are often used such as **precision@k** or the **F-measure**.

Statistical challenges

- Performance measures:
  - Typical performance measures such as **0/1** or **Hamming** loss do **not fit** well to the extreme setting.
  - Other measures are often used such as **precision@k** or the **F-measure**.
  - However, it remains an **open question** how to **design loss functions** suitable for extreme classification.

Do we search in the right place?

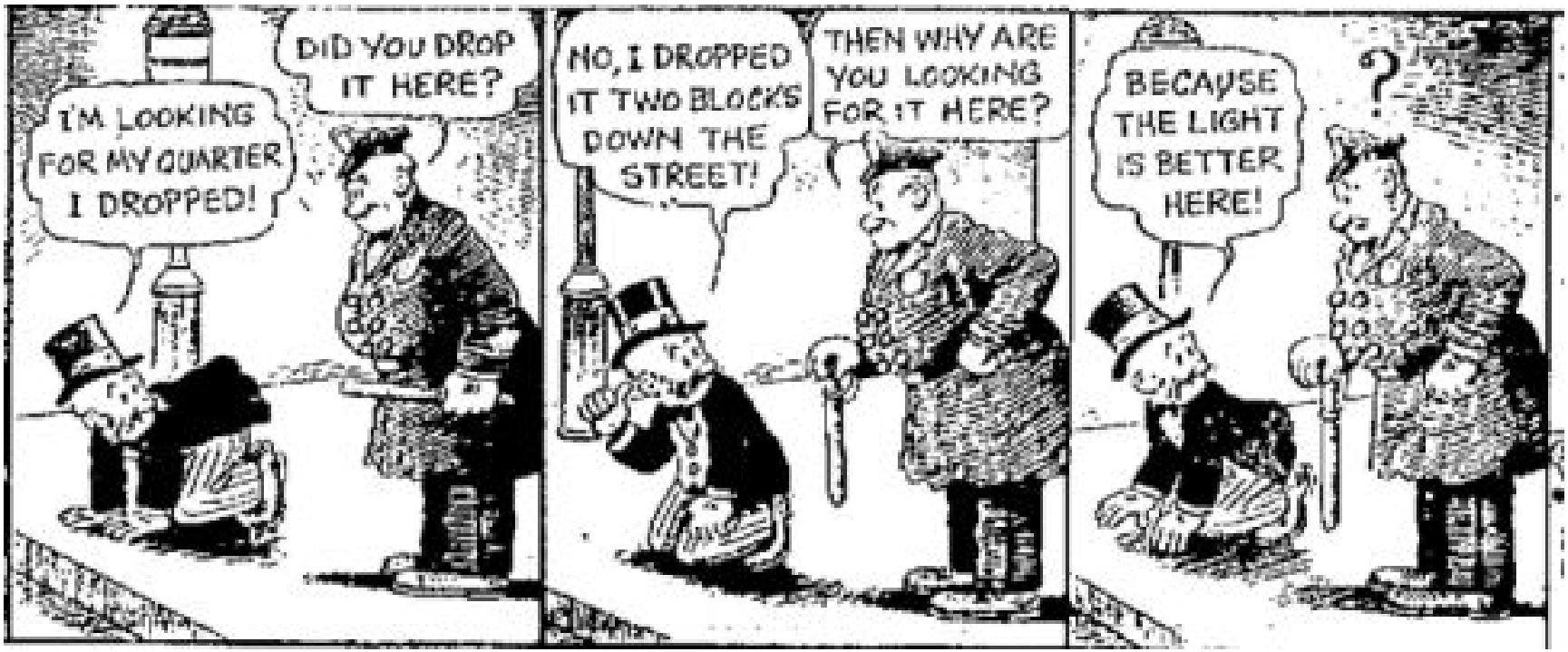


Figure: [3] A similar comics has been earlier used by Asela Gunawardana.[4]

[3] Source: Florence Morning News, Mutt and Jeff Comic Strip, Page 7, Florence, South Carolina,1942

[4] Asela Gunawardana, *Evaluating Machine Learned User Experiences*. Extreme Classification Workshop. NIPS 2015

Statistical challenges

- Long-tail label distributions and zero-shot learning:

Statistical challenges

- Long-tail label distributions and zero-shot learning:
  - A close relation to the problem of **estimating distributions over large alphabets**.

Statistical challenges

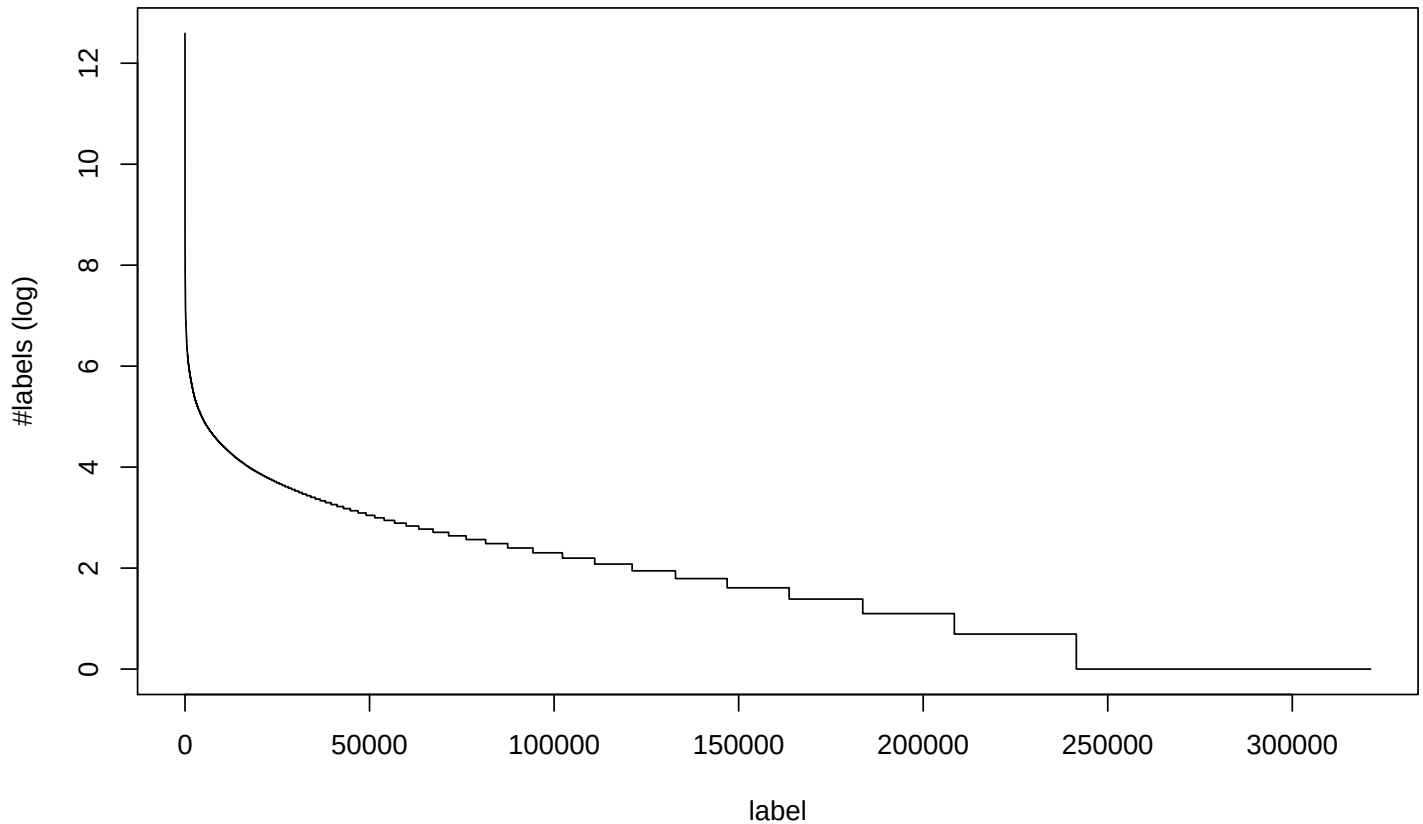
- Long-tail label distributions and zero-shot learning:
  - A close relation to the problem of **estimating distributions over large alphabets**.
  - The distribution of label frequencies is often characterized by a **long-tail** for which proper **smoothing** (like add-constant or Good-Turing estimates) or **calibration** techniques (like isotonic regression or domain adaptation) have to be used.

Statistical challenges

- Long-tail label distributions and zero-shot learning:
  - A close relation to the problem of **estimating distributions over large alphabets**.
  - The distribution of label frequencies is often characterized by a **long-tail** for which proper **smoothing** (like add-constant or Good-Turing estimates) or **calibration** techniques (like isotonic regression or domain adaptation) have to be used.
  - In practical applications, learning algorithms run in **rapidly changing environments**: **new labels** may appear during testing/prediction phase ($\Rightarrow$ **zero-shot learning**)

Statistical challenges

- Long-tail label distributions and zero-shot learning:
  - Frequency of labels in the WikiLSHTC dataset:[5]

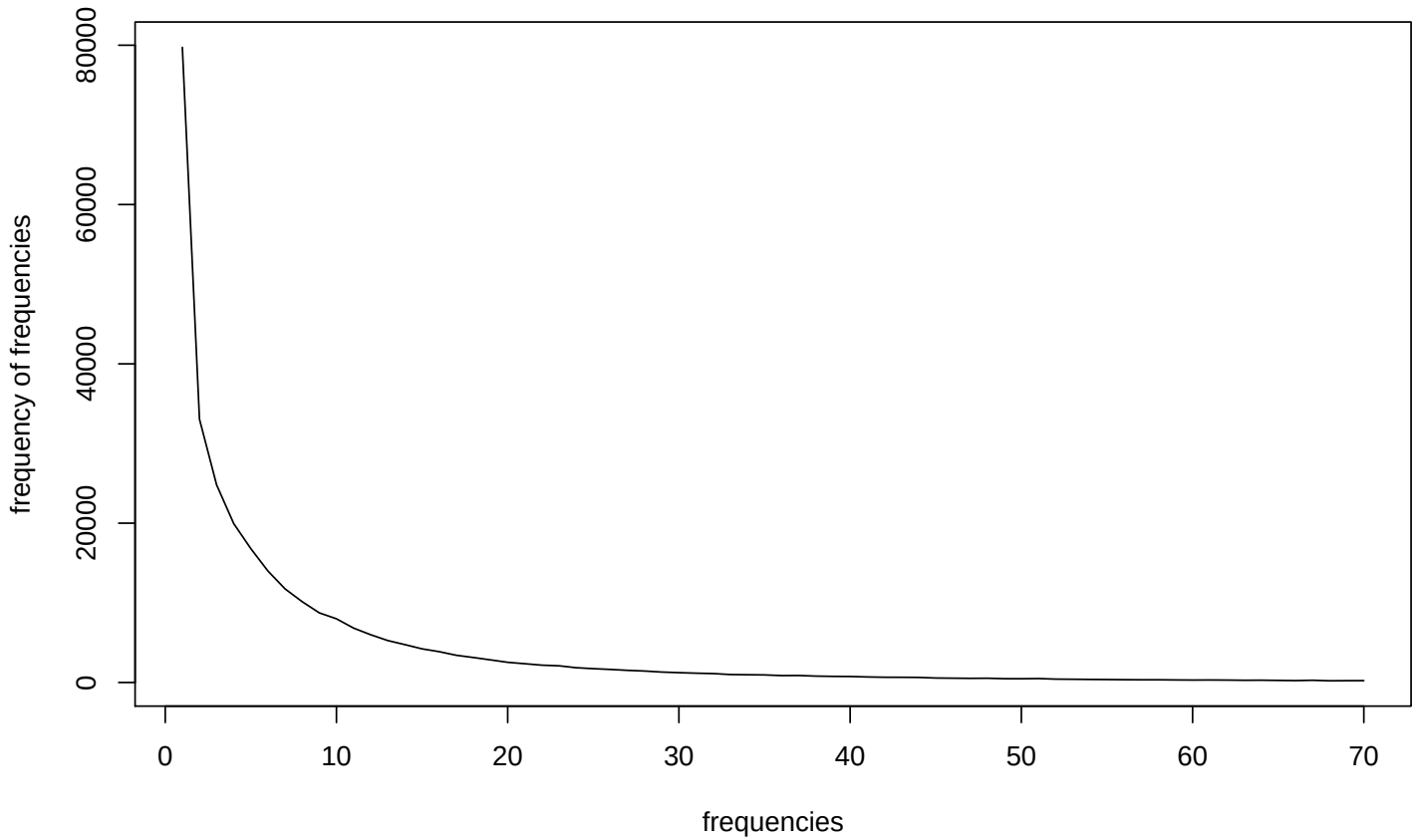


  - Many labels with only few examples ($\Rightarrow$ one-shot learning).

[5] `http://manikvarma.org/downloads/XC/XMLRepository.html`

Statistical challenges

- Long-tail label distributions and zero-shot learning:
  - Frequency of frequencies for the WikiLSHTC dataset:



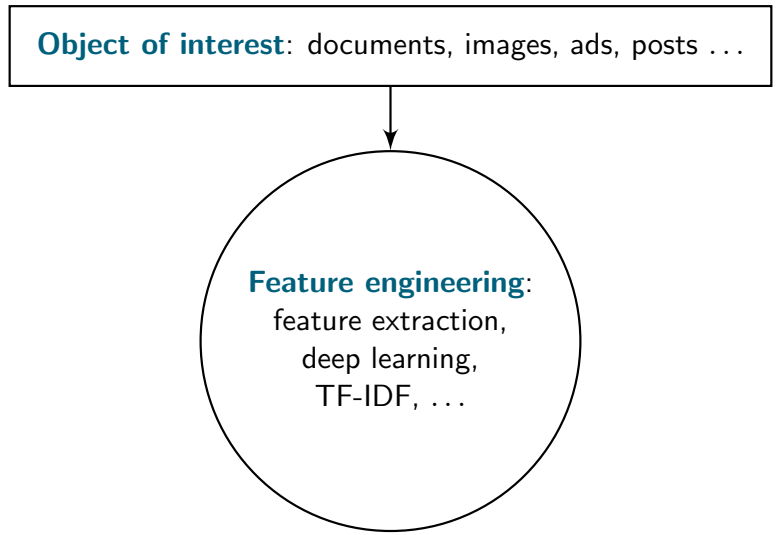
  - The missing mass obtained by the Good-Turing estimate: 0.014.

Formal setting

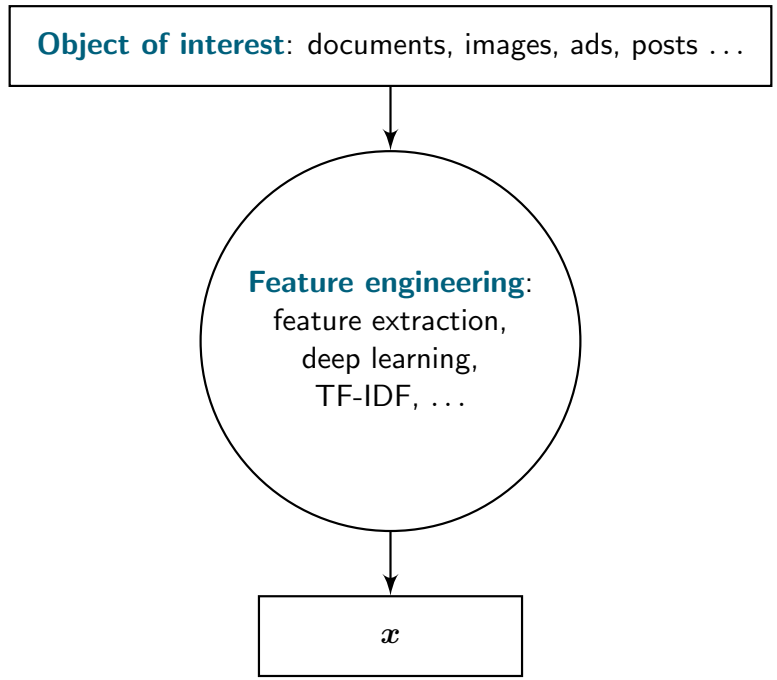
Formal setting

Object of interest: documents, images, ads, posts ...

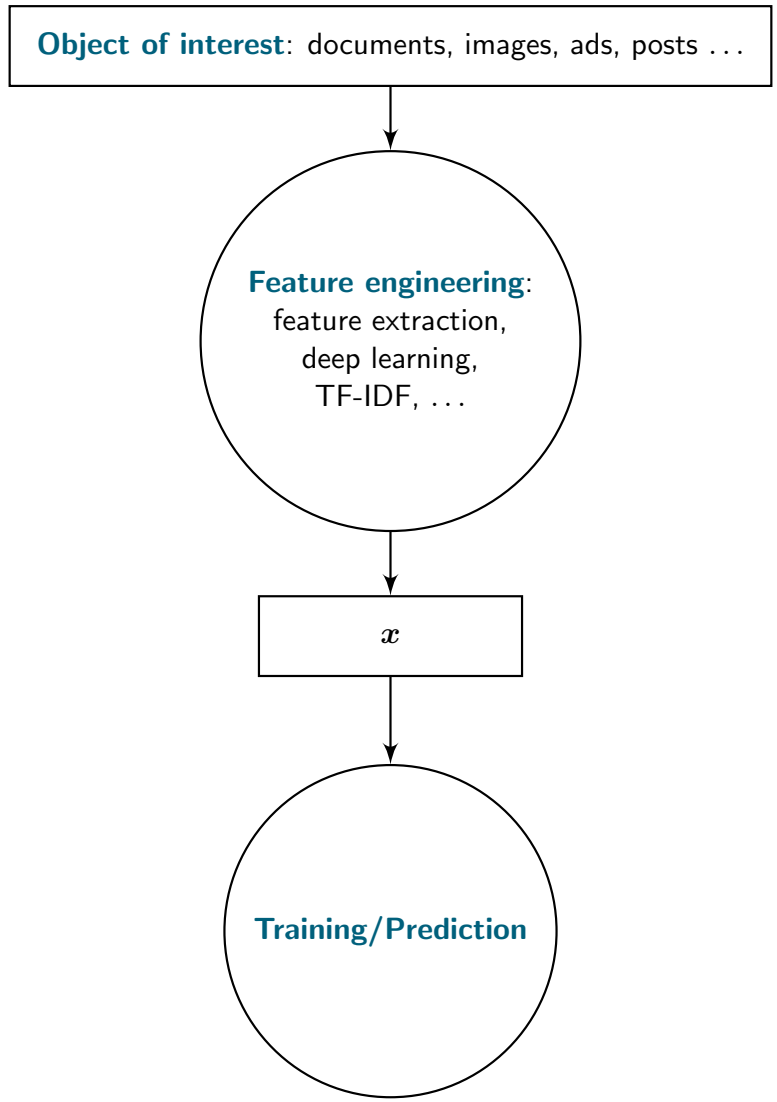
Formal setting



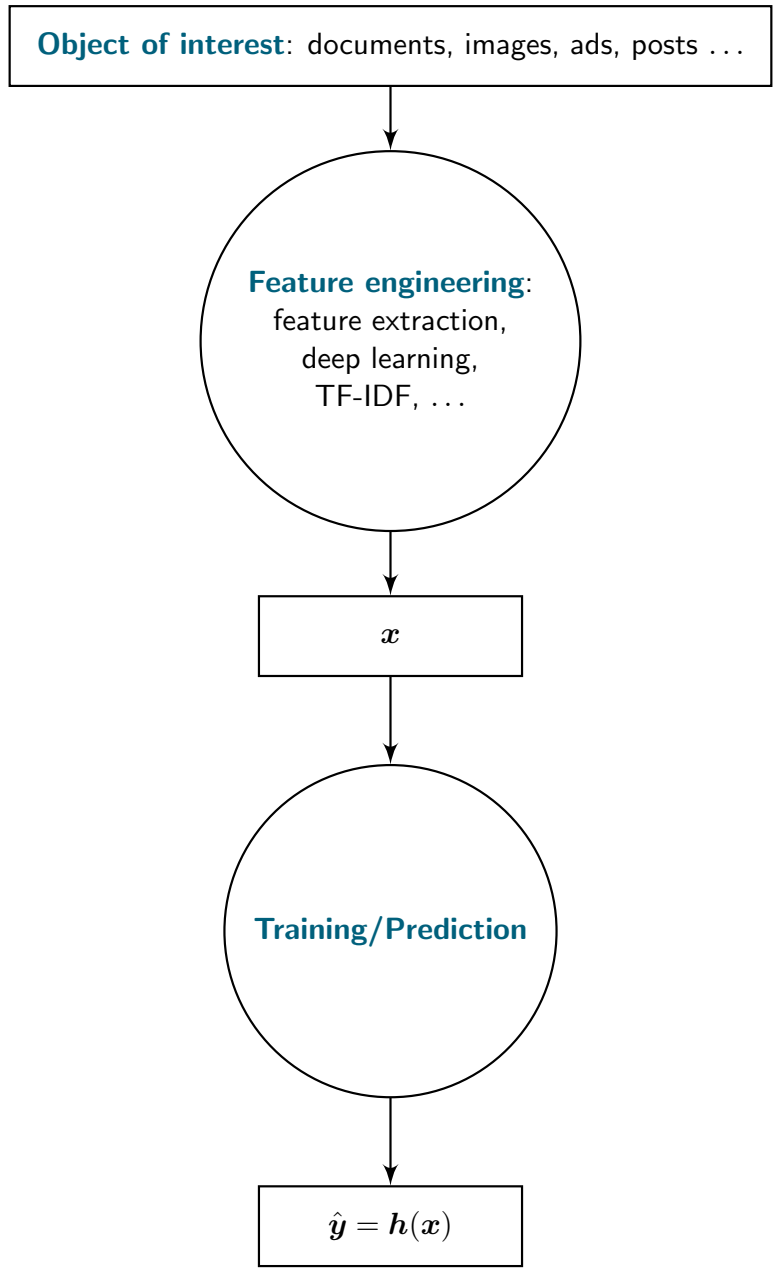
Formal setting



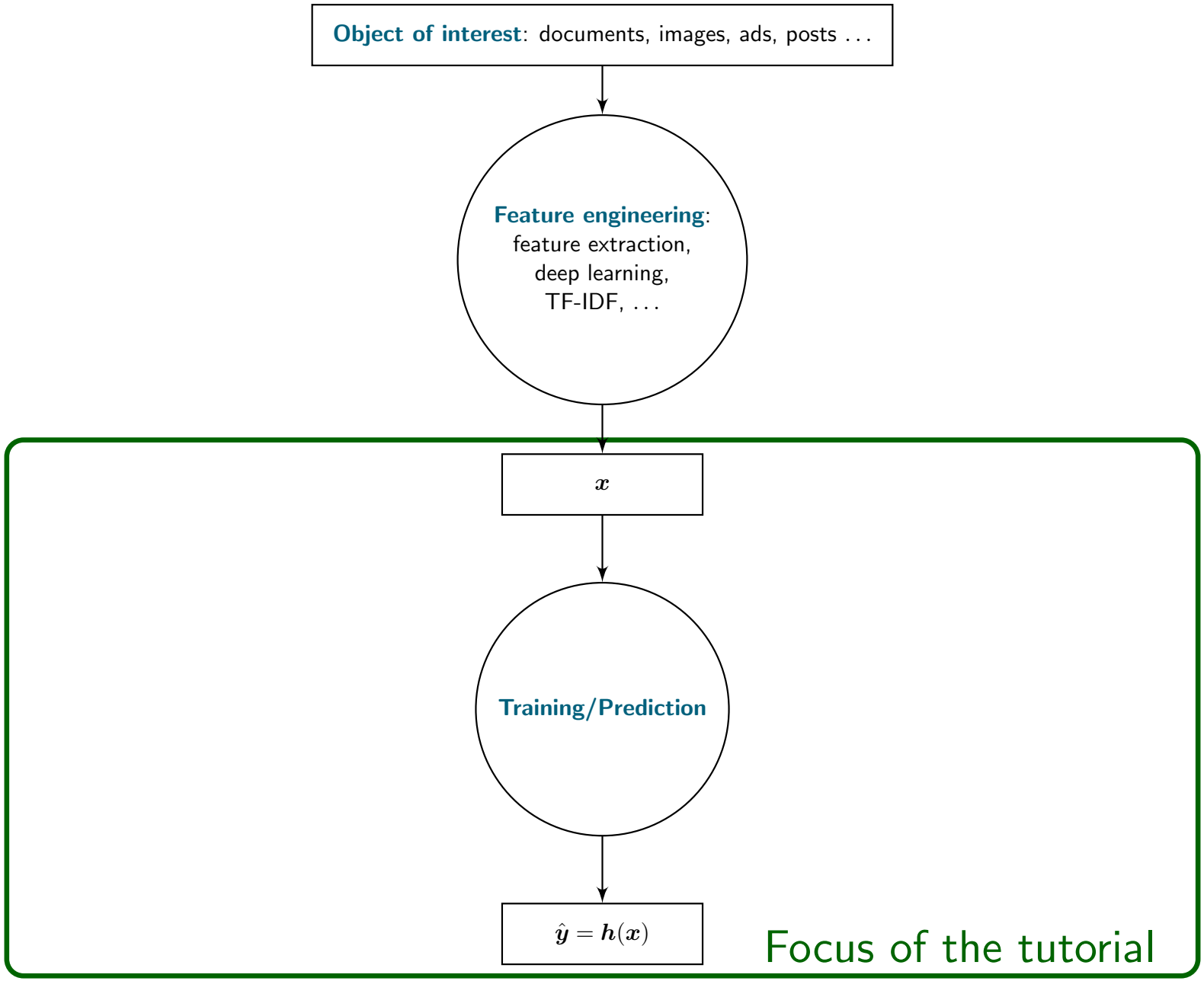
Formal setting



Formal setting



Formal setting

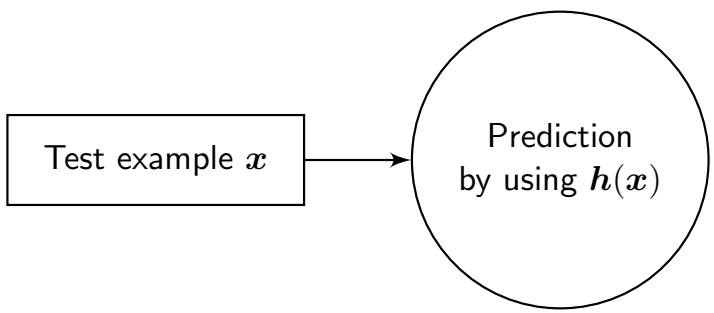


Formal setting

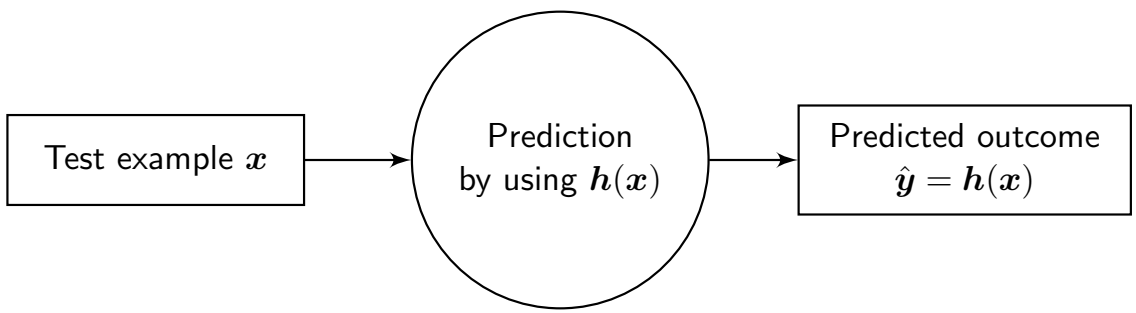
Formal setting

Test example $\boldsymbol{x}$

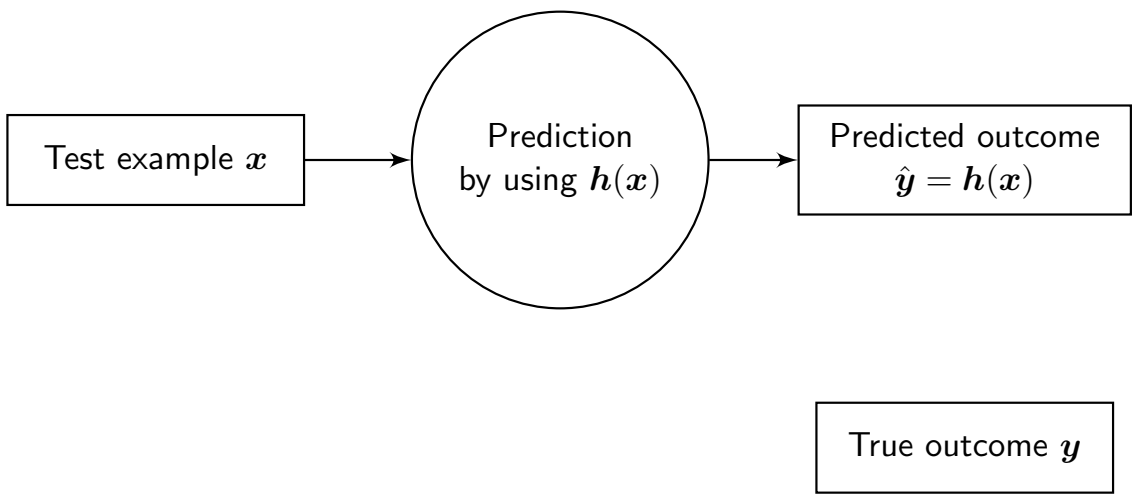
Formal setting



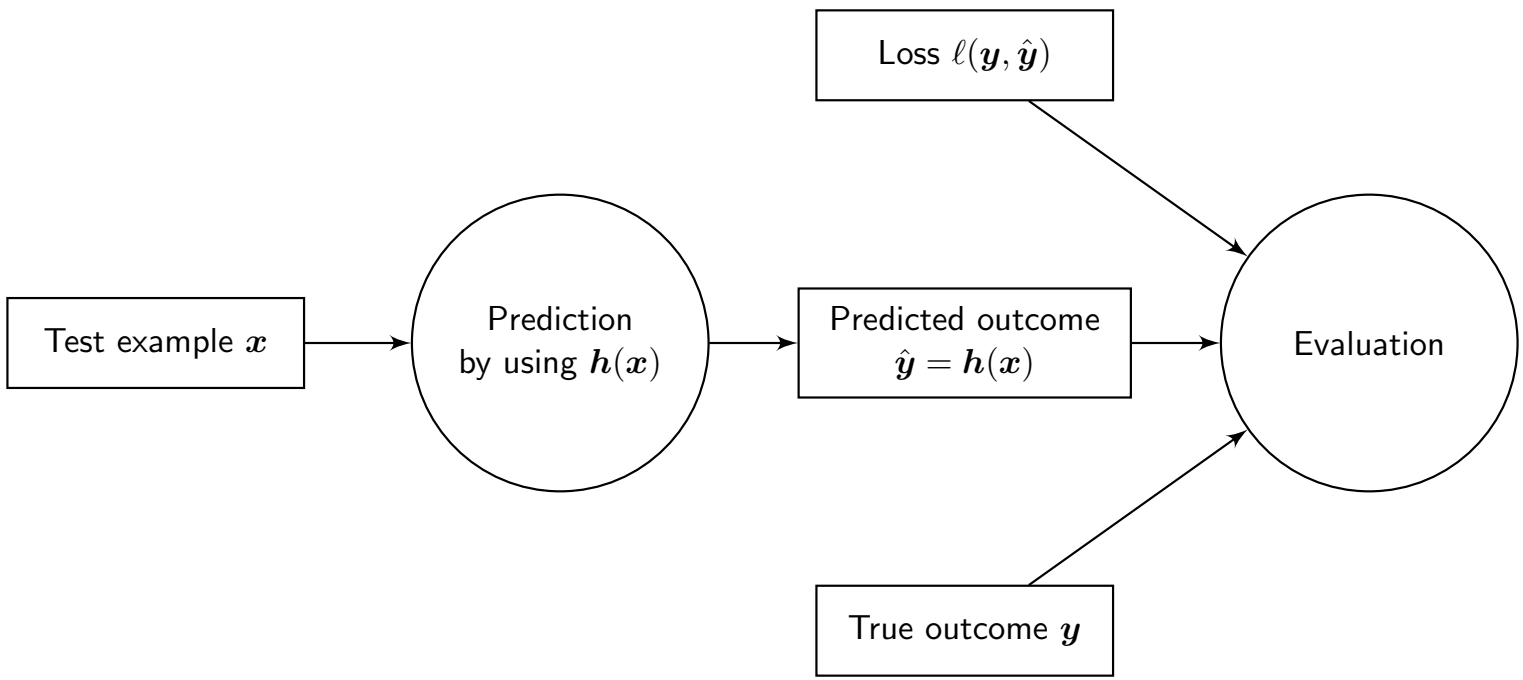
Formal setting



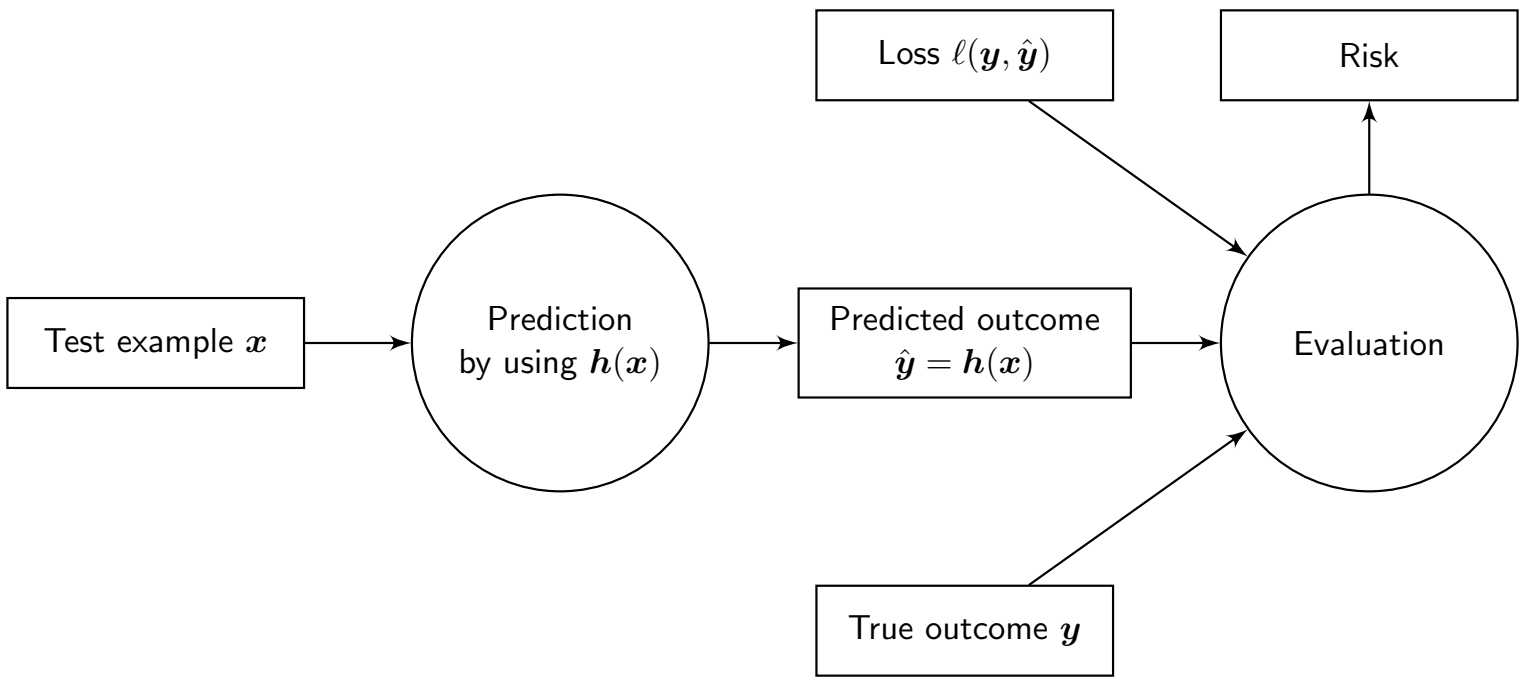
Formal setting



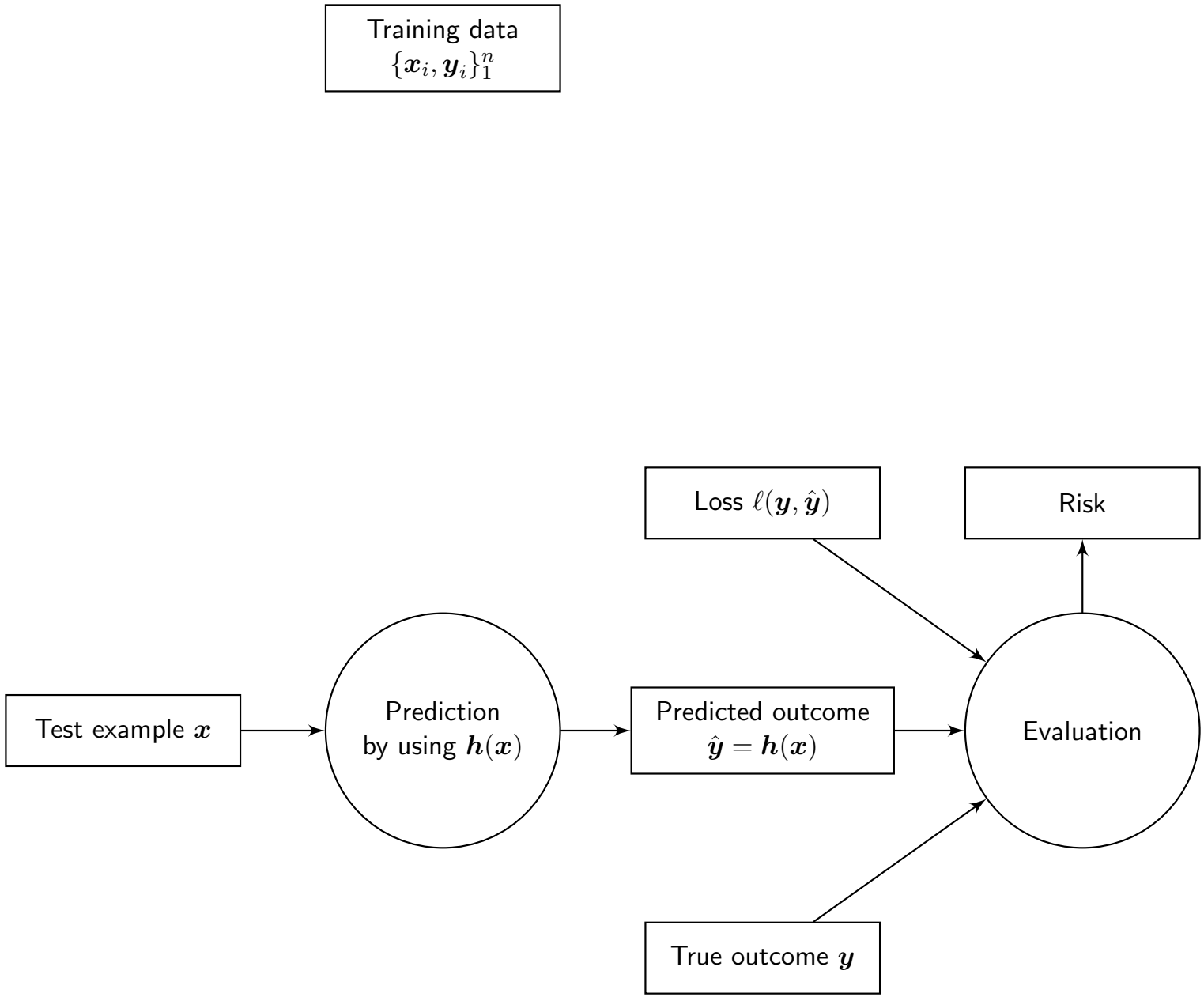
Formal setting



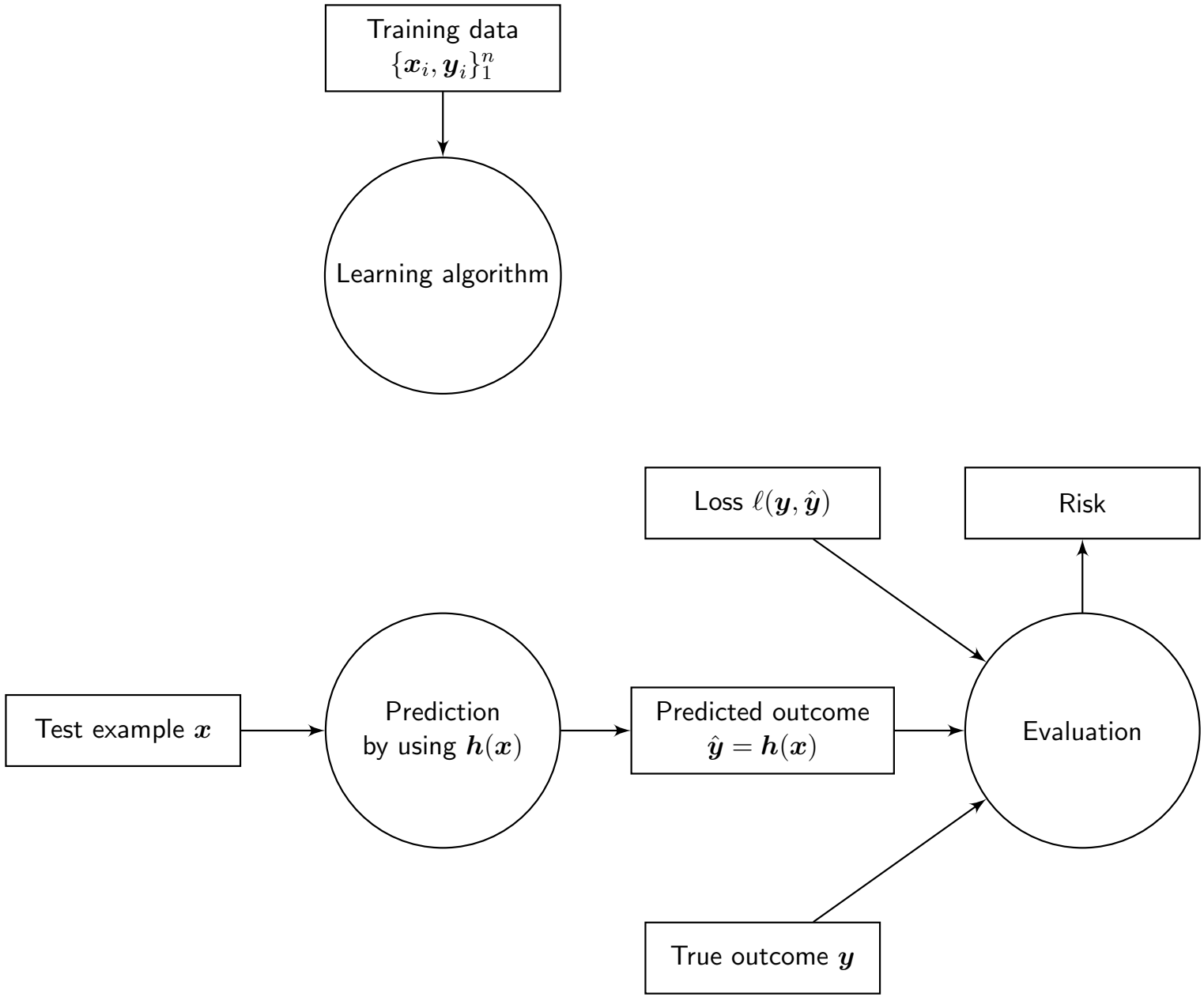
Formal setting



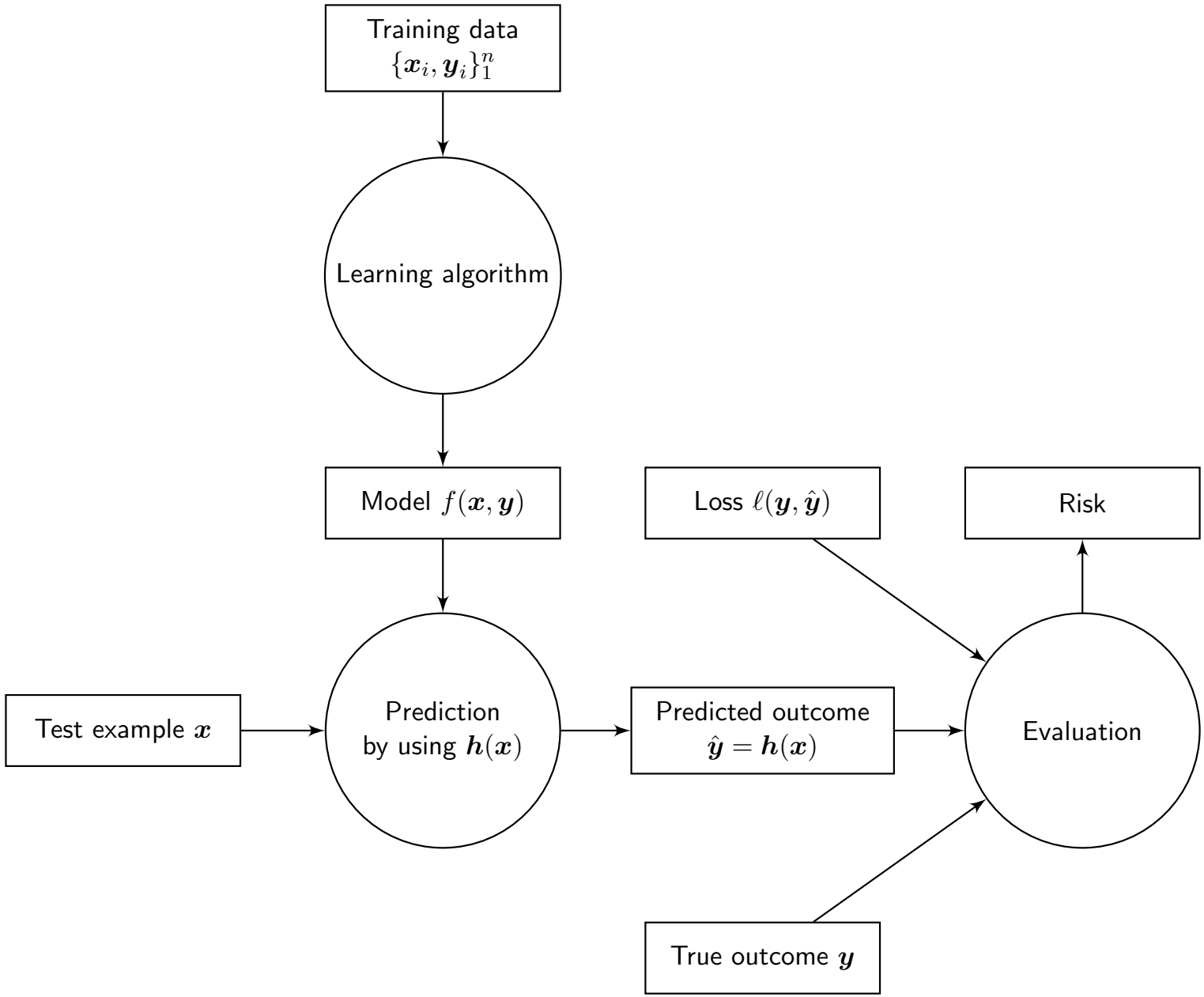
Formal setting



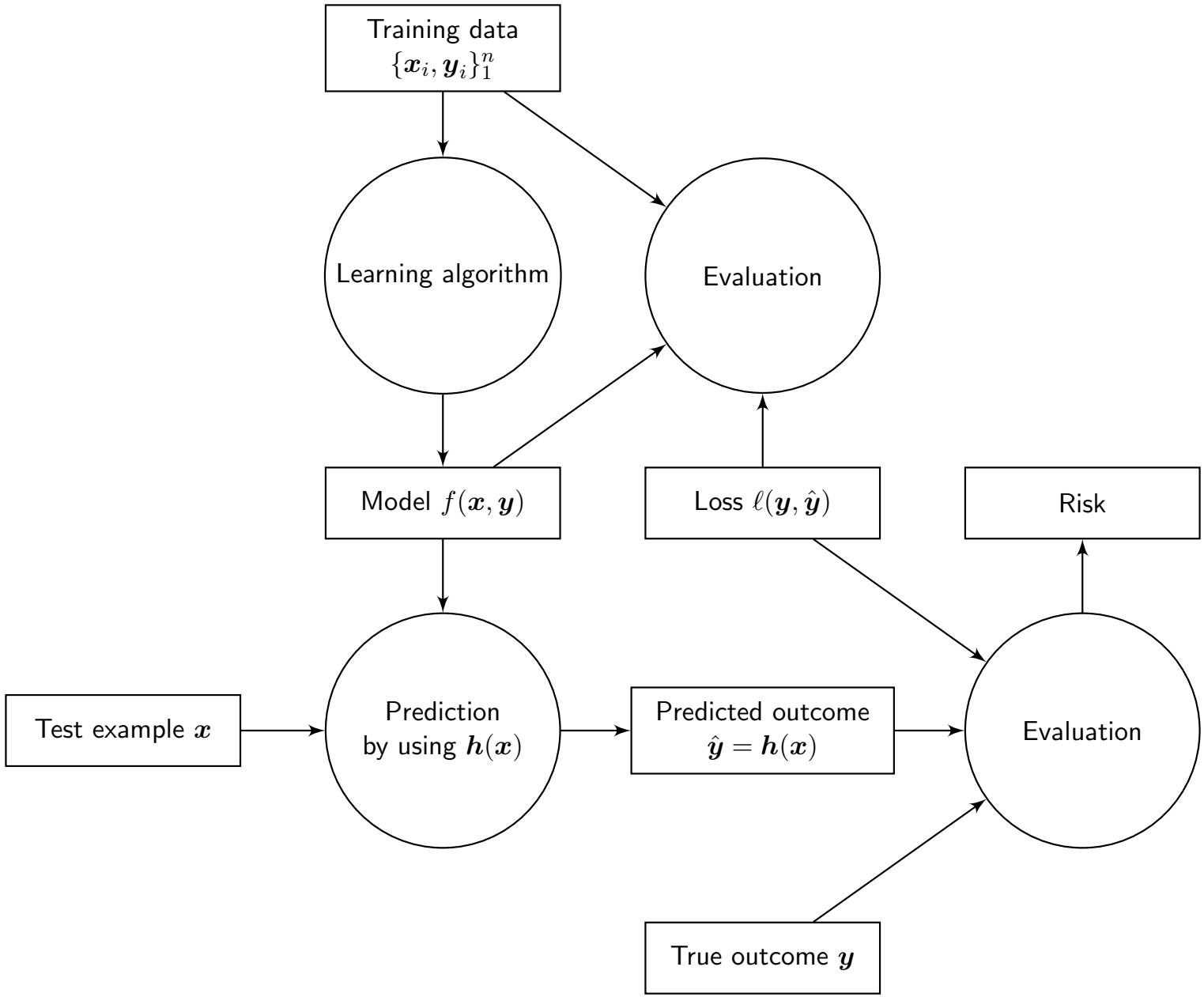
Formal setting



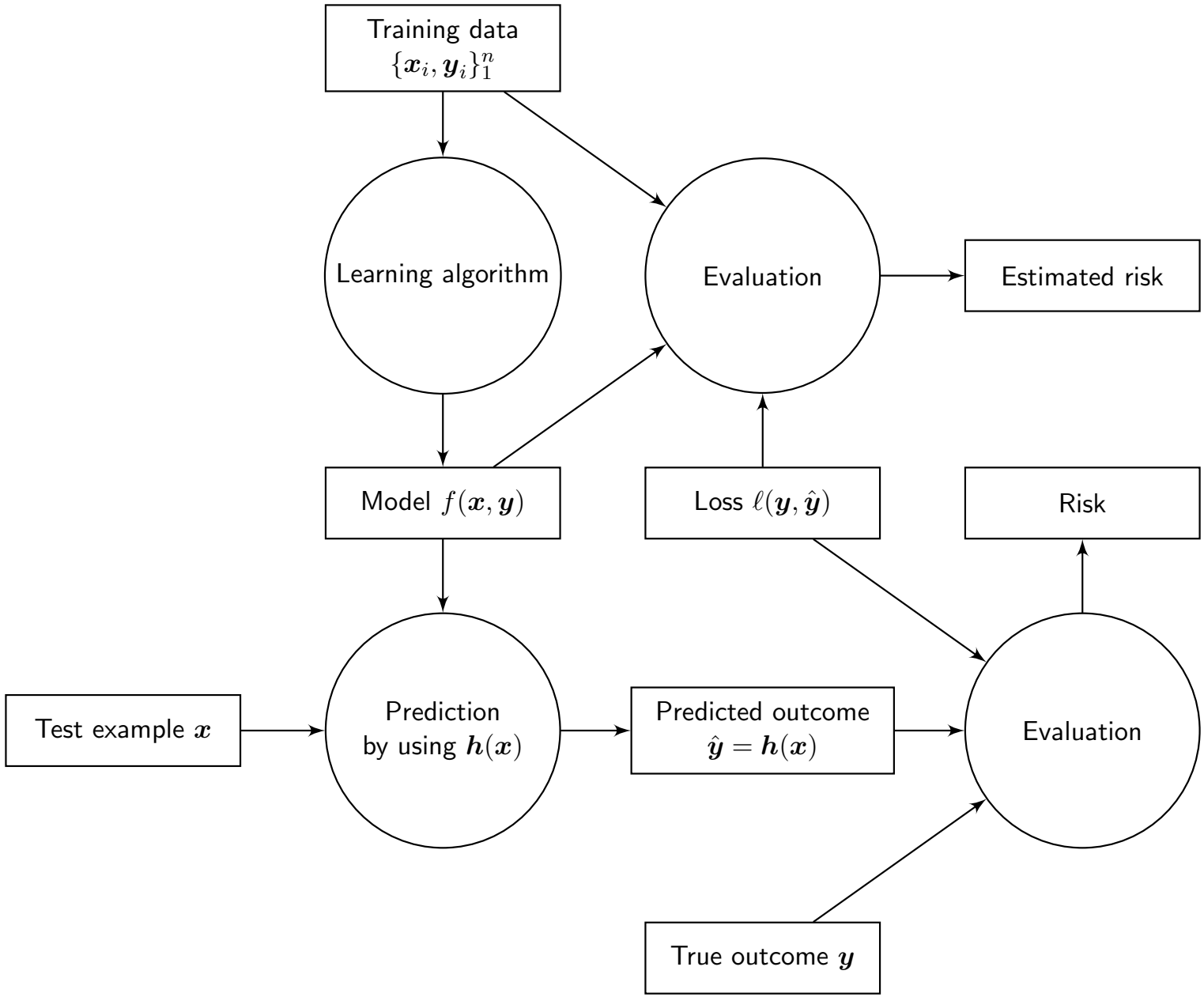
Formal setting



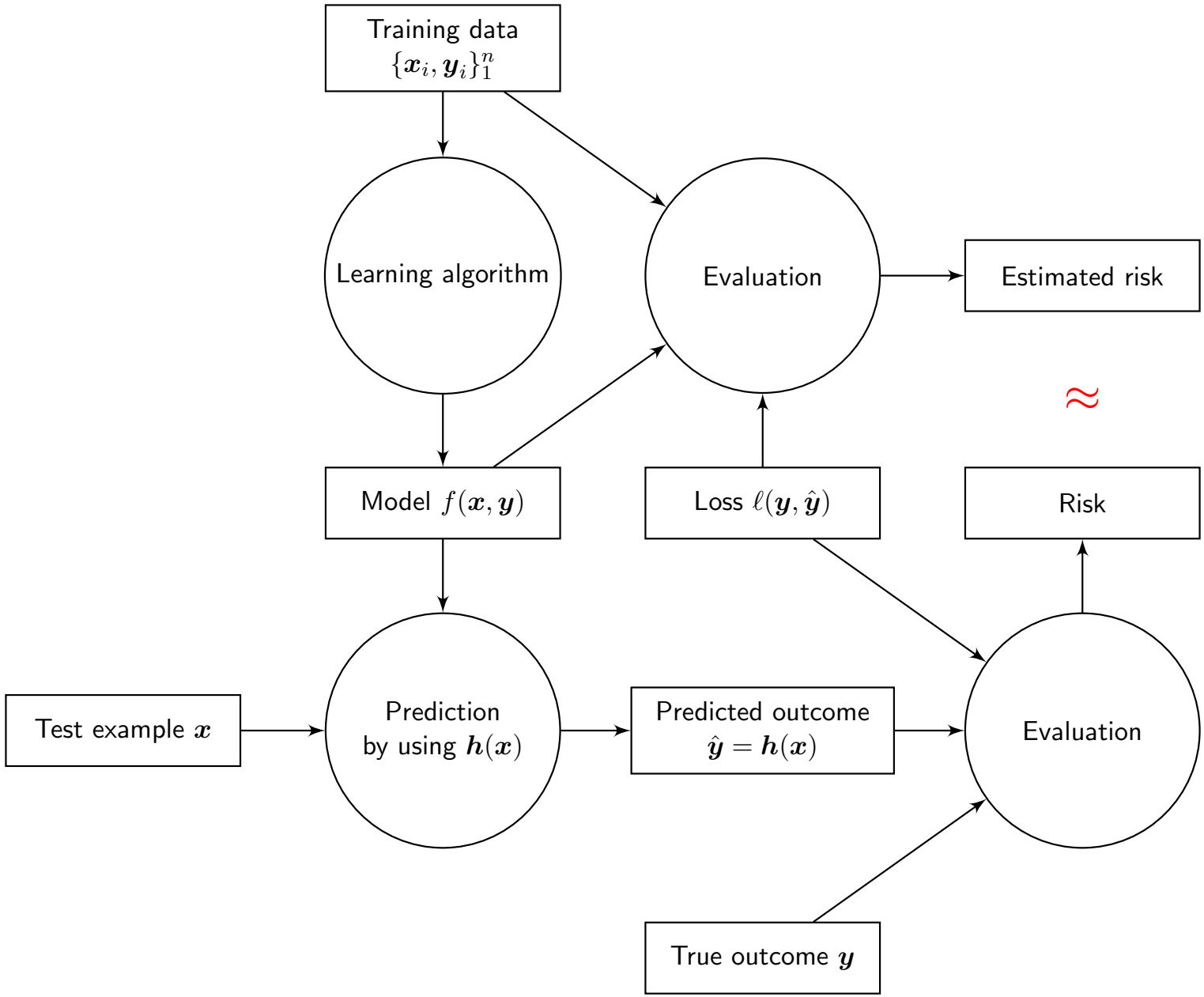
Formal setting



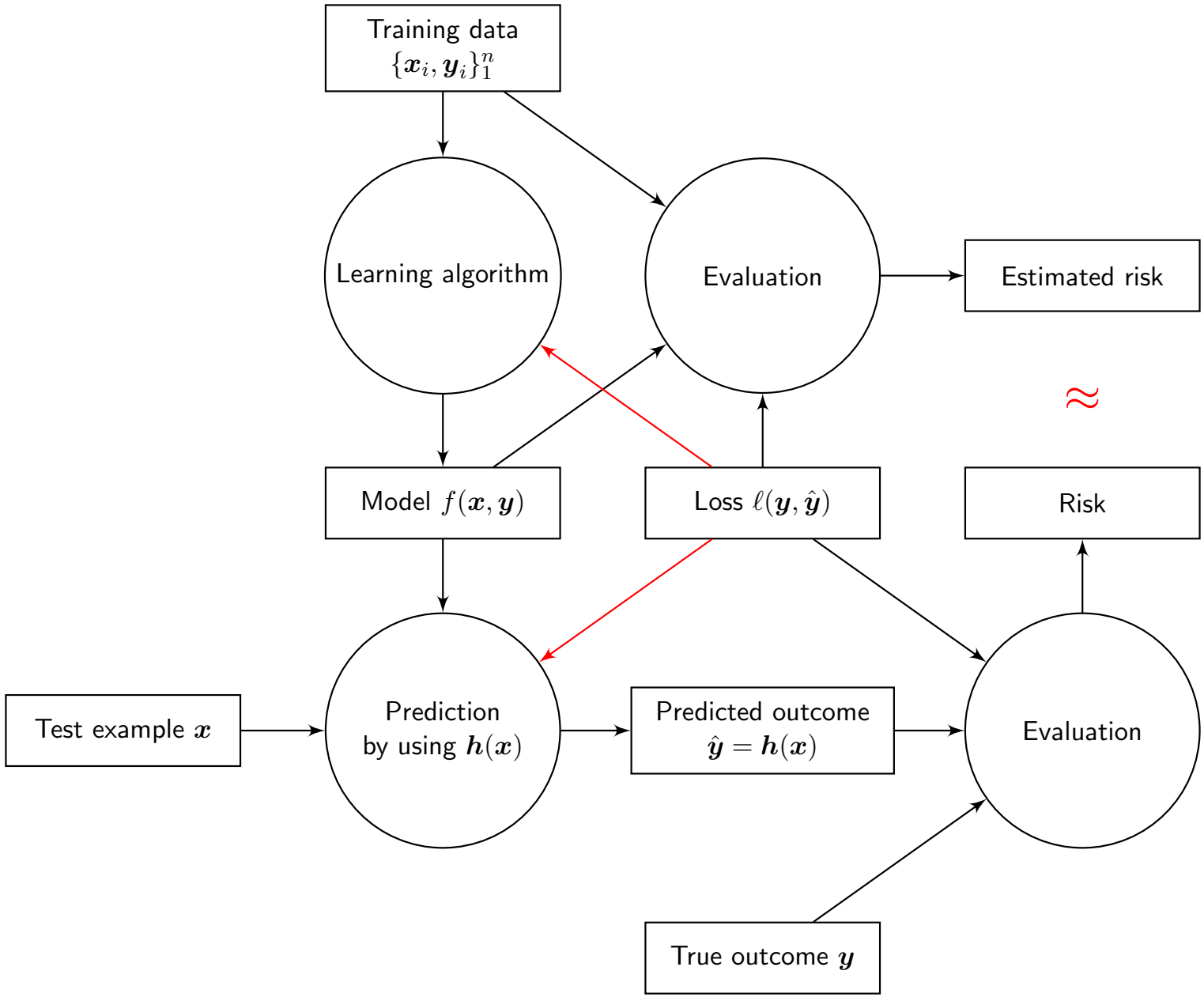
Formal setting



Formal setting



Formal setting



Formal setting

Formal setting

- **Input** $\boldsymbol{x} \in \mathcal{X}$ drawn from a distribution $\mathbf{P}(\boldsymbol{x})$.
  - usually a feature vector, $\mathcal{X} \subseteq \mathbb{R}^d$.

Formal setting

- **Input** $\boldsymbol{x} \in \mathcal{X}$ drawn from a distribution $\mathbf{P}(\boldsymbol{x})$.
  - usually a feature vector, $\mathcal{X} \subseteq \mathbb{R}^d$.
- **Outcome** $\boldsymbol{y} \in \mathcal{Y}$ drawn from a distribution $\mathbf{P}(\boldsymbol{y} \mid \boldsymbol{x})$.
  - a vector of labels $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$.

Formal setting

- **Input** $\boldsymbol{x} \in \mathcal{X}$ drawn from a distribution $\mathbf{P}(\boldsymbol{x})$.
  - usually a feature vector, $\mathcal{X} \subseteq \mathbb{R}^d$.
- **Outcome** $\boldsymbol{y} \in \mathcal{Y}$ drawn from a distribution $\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})$.
  - a vector of labels $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$.
- **Prediction** $\hat{\boldsymbol{y}} = \boldsymbol{h}(\boldsymbol{x})$ by means of **prediction function** $\boldsymbol{h} \colon \mathcal{X} \to \mathcal{Y}$.
  - $\boldsymbol{h}$ returns prediction $\hat{\boldsymbol{y}} = \boldsymbol{h}(\boldsymbol{x})$ for every input $\boldsymbol{x}$.

Formal setting

- **Input** $\boldsymbol{x} \in \mathcal{X}$ drawn from a distribution $\mathbf{P}(\boldsymbol{x})$.
  - usually a feature vector, $\mathcal{X} \subseteq \mathbb{R}^d$.
- **Outcome** $\boldsymbol{y} \in \mathcal{Y}$ drawn from a distribution $\mathbf{P}(\boldsymbol{y} \mid \boldsymbol{x})$.
  - a vector of labels $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$.
- **Prediction** $\hat{\boldsymbol{y}} = \boldsymbol{h}(\boldsymbol{x})$ by means of **prediction function** $\boldsymbol{h} \colon \mathcal{X} \to \mathcal{Y}$.
  - $\boldsymbol{h}$ returns prediction $\hat{\boldsymbol{y}} = \boldsymbol{h}(\boldsymbol{x})$ for every input $\boldsymbol{x}$.
- **Loss** of our prediction: $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}})$.
  - $\ell \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a task-specific **loss function**.

Formal setting

- **Input** $\boldsymbol{x} \in \mathcal{X}$ drawn from a distribution $\mathbf{P}(\boldsymbol{x})$.
  - usually a feature vector, $\mathcal{X} \subseteq \mathbb{R}^d$.
- **Outcome** $\boldsymbol{y} \in \mathcal{Y}$ drawn from a distribution $\mathbf{P}(\boldsymbol{y} \mid \boldsymbol{x})$.
  - a vector of labels $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$.
- **Prediction** $\hat{\boldsymbol{y}} = \boldsymbol{h}(\boldsymbol{x})$ by means of **prediction function** $\boldsymbol{h}\colon \mathcal{X} \to \mathcal{Y}$.
  - $\boldsymbol{h}$ returns prediction $\hat{\boldsymbol{y}} = \boldsymbol{h}(\boldsymbol{x})$ for every input $\boldsymbol{x}$.
- **Loss** of our prediction: $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}})$.
  - $\ell\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a task-specific **loss function**.
- **Goal**: find a prediction function with small loss.

Formal setting

- **Goal**: minimize the **expected** loss over all examples (**risk**):

$$L_\ell(h) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathbf{P}}\left[\ell(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x}))\right].$$

Formal setting

- **Goal**: minimize the **expected** loss over all examples (**risk**):

$$L_\ell(h) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathbf{P}}\left[\ell(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x}))\right].$$

- The **optimal** prediction function over all possible functions expressed conditionally for a given $\boldsymbol{x}$:

$$\boldsymbol{h}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{h}} L_\ell(\boldsymbol{h}|\boldsymbol{x}),$$

(so called **Bayes prediction function**).

Hamming loss

- **Hamming loss**:

$$\ell_H(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{1}{m} \sum_{j=1}^{m} [\![y_j \neq h_j(\boldsymbol{x})]\!] \,,$$

[6] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On loss minimization and label dependence in multi-label classification. *Machine Learning*, 88:5–45, 2012

Hamming loss

- **Hamming loss**:

$$\ell_H(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{1}{m} \sum_{j=1}^{m} [\![y_j \neq h_j(\boldsymbol{x})]\!] ,$$

- Sparse labels $\Rightarrow$ Hamming loss of an **all-zero** classifier close to **0**.

[6] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On loss minimization and label dependence in multi-label classification. *Machine Learning*, 88:5–45, 2012

Hamming loss

- **Hamming loss**:

$$\ell_H(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{1}{m} \sum_{j=1}^{m} [\![y_j \neq h_j(\boldsymbol{x})]\!] ,$$

- Sparse labels $\Rightarrow$ Hamming loss of an **all-zero** classifier close to **0**.
- **The optimal strategy**:[6]

$$h_j^*(\boldsymbol{x}) = [\![\eta_j(\boldsymbol{x}) > 0.5]\!] ,$$

where $\eta_j(\boldsymbol{x}) = \mathbf{P}(y_j = 1 \,|\, \boldsymbol{x})$.

[6] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On loss minimization and label dependence in multi-label classification. *Machine Learning*, 88:5–45, 2012

Hamming loss

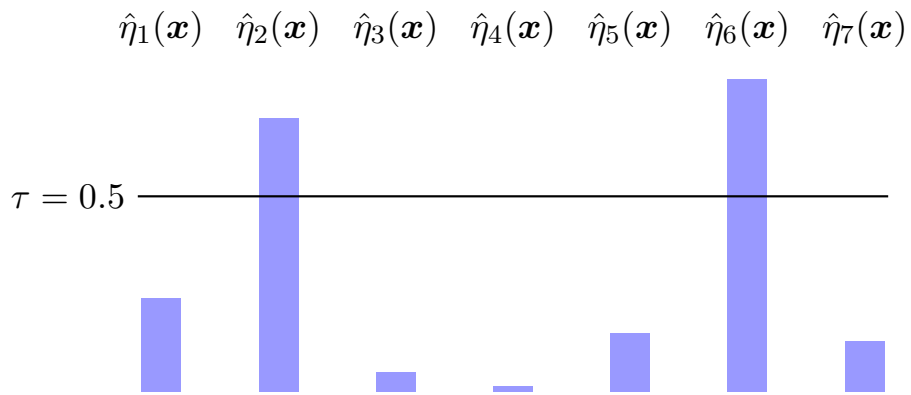
- **Hamming loss**:

$$\ell_H(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{1}{m} \sum_{j=1}^{m} [\![y_j \neq h_j(\boldsymbol{x})]\!] ,$$

- Sparse labels $\Rightarrow$ Hamming loss of an **all-zero** classifier close to **0**.
- **The optimal strategy**:[6]

$$h_j^*(\boldsymbol{x}) = [\![\eta_j(\boldsymbol{x}) > 0.5]\!] ,$$

where $\eta_j(\boldsymbol{x}) = \mathbf{P}(y_j = 1 \,|\, \boldsymbol{x})$.



[6] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On loss minimization and label dependence in multi-label classification. *Machine Learning*, 88:5–45, 2012

Precision

- **Precision at position k**:

$$\text{prec@}k(\boldsymbol{y}, \boldsymbol{h}, \boldsymbol{x}) = \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} [\![y_j = 1]\!] ,$$

  where $\hat{\mathcal{Y}}_k$ is a set of k labels predicted by $\boldsymbol{h}$.

Precision

- **Precision at position k**:

$$\mathrm{prec@}k(\boldsymbol{y}, \boldsymbol{h}, \boldsymbol{x}) = \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} [\![y_j = 1]\!] ,$$

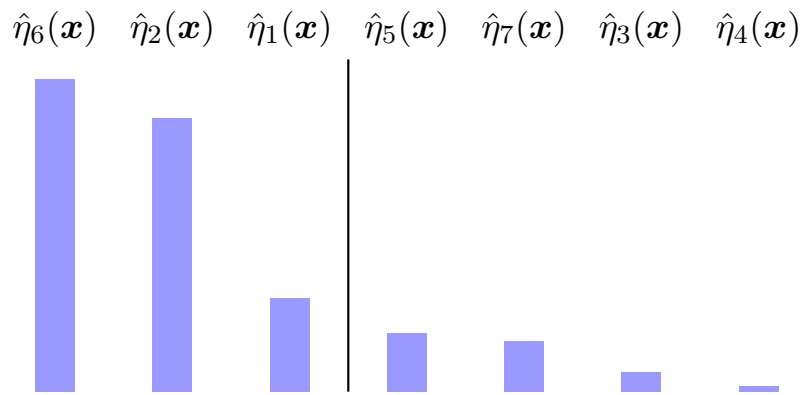
  where $\hat{\mathcal{Y}}_k$ is a set of k labels predicted by $\boldsymbol{h}$.
- **The optimal strategy**: select top k labels according to $\eta_j(\boldsymbol{x})$.

Precision

- **Precision at position k**:

$$\mathrm{prec@}k(\boldsymbol{y}, \boldsymbol{h}, \boldsymbol{x}) = \frac{1}{k} \sum_{j \in \hat{\mathcal{Y}}_k} [\![y_j = 1]\!] ,$$

  where $\hat{\mathcal{Y}}_k$ is a set of k labels predicted by $\boldsymbol{h}$.
- **The optimal strategy**: select top k labels according to $\eta_j(\boldsymbol{x})$.



Normalized Discounted Cumulative Gain

- **Normalized Discounted Cumulative Gain at position k**:

$$\mathrm{NDCG@}k(\boldsymbol{y}, f, \boldsymbol{x}) = N_k(\boldsymbol{y}) \sum_{r=1}^{k} \frac{y_{\sigma(r)}}{\log(1+r)},$$

where σ is a permutation of labels for $\boldsymbol{x}$ returned by ranker f, and $N_k(\boldsymbol{y})$ normalizes NDCG@k to the interval $[0, 1]$:

$$N_k(\boldsymbol{y}) = \left(\sum_{r=1}^{\max(k, \sum_{i=1}^{m} y_i)} \frac{1}{\log(1+r)} \right)^{-1}$$

Normalized Discounted Cumulative Gain

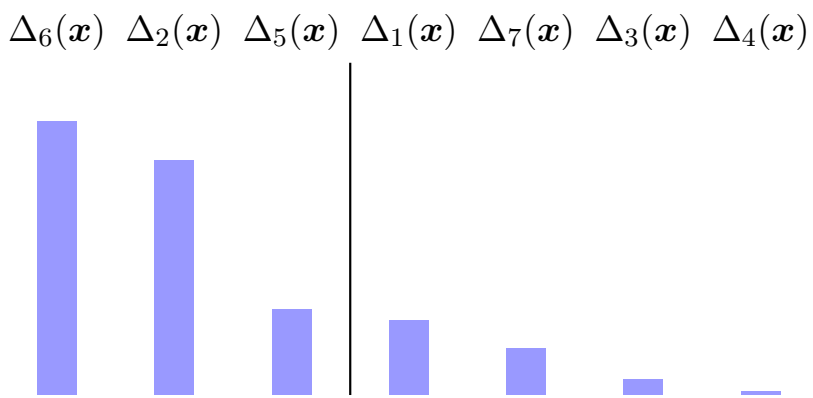
- **The optimal strategy**: rank labels according to the following marginal quantities:

$$\Delta_j(\boldsymbol{x}) = \sum_{\boldsymbol{y}:y_j=1} N_k(\boldsymbol{y})\mathbf{P}(\boldsymbol{y} \mid \boldsymbol{x})$$

Normalized Discounted Cumulative Gain

- **The optimal strategy**: rank labels according to the following marginal quantities:

$$\Delta_j(\boldsymbol{x}) = \sum_{\boldsymbol{y}:y_j=1} N_k(\boldsymbol{y})\mathbf{P}(\boldsymbol{y} \mid \boldsymbol{x})$$



Macro-averaging of the F-measure

- The **macro** F-measure (F-score):

$$F_M(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) = \frac{1}{m}\sum_{j=1}^{m} F(\boldsymbol{y}_{\cdot j}, \hat{\boldsymbol{y}}_{\cdot j}) = \frac{1}{m}\sum_{j=1}^{m} \frac{2\sum_{i=1}^{n} y_{ij}\hat{y}_{ij}}{\sum_{i=1}^{n} y_{ij} + \sum_{i=1}^{n} \hat{y}_{ij}} \,.$$

True labels

y_{11}	y_{12}	y_{13}	y_{14}
y_{21}	y_{22}	y_{23}	y_{24}
y_{31}	y_{32}	y_{33}	y_{34}
y_{41}	y_{42}	y_{43}	y_{44}
y_{51}	y_{52}	y_{53}	y_{54}
y_{61}	y_{62}	y_{63}	y_{64}

Predicted labels

$\hat{y}_{11}$	$\hat{y}_{12}$	$\hat{y}_{13}$	$\hat{y}_{14}$
$\hat{y}_{21}$	$\hat{y}_{22}$	$\hat{y}_{23}$	$\hat{y}_{24}$
$\hat{y}_{31}$	$\hat{y}_{32}$	$\hat{y}_{33}$	$\hat{y}_{34}$
$\hat{y}_{41}$	$\hat{y}_{42}$	$\hat{y}_{43}$	$\hat{y}_{44}$
$\hat{y}_{51}$	$\hat{y}_{52}$	$\hat{y}_{53}$	$\hat{y}_{54}$
$\hat{y}_{61}$	$\hat{y}_{62}$	$\hat{y}_{63}$	$\hat{y}_{64}$

Macro-averaging of the F-measure

- The **macro** F-measure (F-score):

$$F_M(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) = \frac{1}{m}\sum_{j=1}^{m} F(\boldsymbol{y}_{\cdot j}, \hat{\boldsymbol{y}}_{\cdot j}) = \frac{1}{m}\sum_{j=1}^{m} \frac{2\sum_{i=1}^{n} y_{ij}\hat{y}_{ij}}{\sum_{i=1}^{n} y_{ij} + \sum_{i=1}^{n} \hat{y}_{ij}} \,.$$

True labels

y_{11}	y_{12}	y_{13}	y_{14}
y_{21}	y_{22}	y_{23}	y_{24}
y_{31}	y_{32}	y_{33}	y_{34}
y_{41}	y_{42}	y_{43}	y_{44}
y_{51}	y_{52}	y_{53}	y_{54}
y_{61}	y_{62}	y_{63}	y_{64}

Predicted labels

$\hat{y}_{11}$	$\hat{y}_{12}$	$\hat{y}_{13}$	$\hat{y}_{14}$
$\hat{y}_{21}$	$\hat{y}_{22}$	$\hat{y}_{23}$	$\hat{y}_{24}$
$\hat{y}_{31}$	$\hat{y}_{32}$	$\hat{y}_{33}$	$\hat{y}_{34}$
$\hat{y}_{41}$	$\hat{y}_{42}$	$\hat{y}_{43}$	$\hat{y}_{44}$
$\hat{y}_{51}$	$\hat{y}_{52}$	$\hat{y}_{53}$	$\hat{y}_{54}$
$\hat{y}_{61}$	$\hat{y}_{62}$	$\hat{y}_{63}$	$\hat{y}_{64}$

Macro-averaging of the F-measure

- The **macro** F-measure (F-score):

$$F_M(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) = \frac{1}{m}\sum_{j=1}^{m} F(\boldsymbol{y}_{\cdot j}, \hat{\boldsymbol{y}}_{\cdot j}) = \frac{1}{m}\sum_{j=1}^{m} \frac{2\sum_{i=1}^{n} y_{ij}\hat{y}_{ij}}{\sum_{i=1}^{n} y_{ij} + \sum_{i=1}^{n} \hat{y}_{ij}} \,.$$

True labels

y_{11}	y_{12}	y_{13}	y_{14}
y_{21}	y_{22}	y_{23}	y_{24}
y_{31}	y_{32}	y_{33}	y_{34}
y_{41}	y_{42}	y_{43}	y_{44}
y_{51}	y_{52}	y_{53}	y_{54}
y_{61}	y_{62}	y_{63}	y_{64}

Predicted labels

$\hat{y}_{11}$	$\hat{y}_{12}$	$\hat{y}_{13}$	$\hat{y}_{14}$
$\hat{y}_{21}$	$\hat{y}_{22}$	$\hat{y}_{23}$	$\hat{y}_{24}$
$\hat{y}_{31}$	$\hat{y}_{32}$	$\hat{y}_{33}$	$\hat{y}_{34}$
$\hat{y}_{41}$	$\hat{y}_{42}$	$\hat{y}_{43}$	$\hat{y}_{44}$
$\hat{y}_{51}$	$\hat{y}_{52}$	$\hat{y}_{53}$	$\hat{y}_{54}$
$\hat{y}_{61}$	$\hat{y}_{62}$	$\hat{y}_{63}$	$\hat{y}_{64}$

Macro-averaging of the F-measure

- The **macro** F-measure (F-score):

$$F_M(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) = \frac{1}{m}\sum_{j=1}^{m} F(\boldsymbol{y}_{\cdot j}, \hat{\boldsymbol{y}}_{\cdot j}) = \frac{1}{m}\sum_{j=1}^{m} \frac{2\sum_{i=1}^{n} y_{ij}\hat{y}_{ij}}{\sum_{i=1}^{n} y_{ij} + \sum_{i=1}^{n} \hat{y}_{ij}} \,.$$

True labels

y_{11}	y_{12}	y_{13}	y_{14}
y_{21}	y_{22}	y_{23}	y_{24}
y_{31}	y_{32}	y_{33}	y_{34}
y_{41}	y_{42}	y_{43}	y_{44}
y_{51}	y_{52}	y_{53}	y_{54}
y_{61}	y_{62}	y_{63}	y_{64}

Predicted labels

$\hat{y}_{11}$	$\hat{y}_{12}$	$\hat{y}_{13}$	$\hat{y}_{14}$
$\hat{y}_{21}$	$\hat{y}_{22}$	$\hat{y}_{23}$	$\hat{y}_{24}$
$\hat{y}_{31}$	$\hat{y}_{32}$	$\hat{y}_{33}$	$\hat{y}_{34}$
$\hat{y}_{41}$	$\hat{y}_{42}$	$\hat{y}_{43}$	$\hat{y}_{44}$
$\hat{y}_{51}$	$\hat{y}_{52}$	$\hat{y}_{53}$	$\hat{y}_{54}$
$\hat{y}_{61}$	$\hat{y}_{62}$	$\hat{y}_{63}$	$\hat{y}_{64}$

Macro-averaging of the F-measure

- The **macro** F-measure (F-score):

$$F_M(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) = \frac{1}{m}\sum_{j=1}^{m} F(\boldsymbol{y}_{\cdot j}, \hat{\boldsymbol{y}}_{\cdot j}) = \frac{1}{m}\sum_{j=1}^{m} \frac{2\sum_{i=1}^{n} y_{ij}\hat{y}_{ij}}{\sum_{i=1}^{n} y_{ij} + \sum_{i=1}^{n} \hat{y}_{ij}} \,.$$

True labels

y_{11}	y_{12}	y_{13}	y_{14}
y_{21}	y_{22}	y_{23}	y_{24}
y_{31}	y_{32}	y_{33}	y_{34}
y_{41}	y_{42}	y_{43}	y_{44}
y_{51}	y_{52}	y_{53}	y_{54}
y_{61}	y_{62}	y_{63}	y_{64}

Predicted labels

$\hat{y}_{11}$	$\hat{y}_{12}$	$\hat{y}_{13}$	$\hat{y}_{14}$
$\hat{y}_{21}$	$\hat{y}_{22}$	$\hat{y}_{23}$	$\hat{y}_{24}$
$\hat{y}_{31}$	$\hat{y}_{32}$	$\hat{y}_{33}$	$\hat{y}_{34}$
$\hat{y}_{41}$	$\hat{y}_{42}$	$\hat{y}_{43}$	$\hat{y}_{44}$
$\hat{y}_{51}$	$\hat{y}_{52}$	$\hat{y}_{53}$	$\hat{y}_{54}$
$\hat{y}_{61}$	$\hat{y}_{62}$	$\hat{y}_{63}$	$\hat{y}_{64}$

Macro-averaging of the F-measure

- Can be **solved** by **reduction** to m independent **binary** problems.[7]

[7] O. Koyejo, N. Natarajan, P. Ravikumar, and I. Dhillon. Consistent multilabel classification. In *NIPS*, 2015

Macro-averaging of the F-measure

- Can be **solved** by **reduction** to m independent **binary** problems.[7]
- **Thresholding** the conditional probabilities:

$$F(\tau) = \frac{2 \int_{\mathcal{X}} \eta(\boldsymbol{x}) [\![\eta(\boldsymbol{x}) \geq \tau]\!] \, \mathrm{d}\mu(\boldsymbol{x})}{\int_{\mathcal{X}} \eta(\boldsymbol{x}) \, \mathrm{d}\mu(\boldsymbol{x}) + \int_{\mathcal{X}} [\![\eta(\boldsymbol{x}) \geq \tau]\!] \, \mathrm{d}\mu(\boldsymbol{x})}.$$

[7] O. Koyejo, N. Natarajan, P. Ravikumar, and I. Dhillon. Consistent multilabel classification. In *NIPS*, 2015

Macro-averaging of the F-measure

- Can be **solved** by **reduction** to m independent **binary** problems.[7]
- **Thresholding** the conditional probabilities:

$$F(\tau) = \frac{2 \int_{\mathcal{X}} \eta(\boldsymbol{x}) [\![\eta(\boldsymbol{x}) \geq \tau]\!] \, \mathrm{d}\mu(\boldsymbol{x})}{\int_{\mathcal{X}} \eta(\boldsymbol{x}) \, \mathrm{d}\mu(\boldsymbol{x}) + \int_{\mathcal{X}} [\![\eta(\boldsymbol{x}) \geq \tau]\!] \, \mathrm{d}\mu(\boldsymbol{x})}.$$

- The **optimal F-measure** is $F(\tau^*)$: no binary classifier can be better.

[7] O. Koyejo, N. Natarajan, P. Ravikumar, and I. Dhillon. Consistent multilabel classification. In *NIPS*, 2015

Macro-averaging of the F-measure

- Can be **solved** by **reduction** to m independent **binary** problems.[7]
- **Thresholding** the conditional probabilities:

$$F(\tau) = \frac{2 \int_{\mathcal{X}} \eta(\boldsymbol{x}) [\![\eta(\boldsymbol{x}) \geq \tau]\!] \, \mathrm{d}\mu(\boldsymbol{x})}{\int_{\mathcal{X}} \eta(\boldsymbol{x}) \, \mathrm{d}\mu(\boldsymbol{x}) + \int_{\mathcal{X}} [\![\eta(\boldsymbol{x}) \geq \tau]\!] \, \mathrm{d}\mu(\boldsymbol{x})}.$$

- The **optimal F-measure** is $F(\tau^*)$: no binary classifier can be better.
- The optimal solution **satisfies** the following **condition**: $F(\tau^*) = 2\tau^*$.

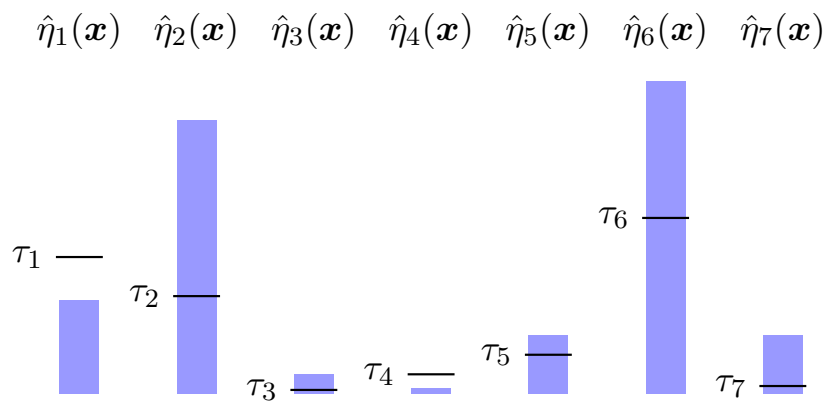
[7] O. Koyejo, N. Natarajan, P. Ravikumar, and I. Dhillon. Consistent multilabel classification. In *NIPS*, 2015

Macro-averaging of the F-measure

- Can be **solved** by **reduction** to m independent **binary** problems.[7]
- **Thresholding** the conditional probabilities:

$$F(\tau) = \frac{2 \int_{\mathcal{X}} \eta(\boldsymbol{x}) [\![\eta(\boldsymbol{x}) \geq \tau]\!] \, \mathrm{d}\mu(\boldsymbol{x})}{\int_{\mathcal{X}} \eta(\boldsymbol{x}) \, \mathrm{d}\mu(\boldsymbol{x}) + \int_{\mathcal{X}} [\![\eta(\boldsymbol{x}) \geq \tau]\!] \, \mathrm{d}\mu(\boldsymbol{x})}.$$

- The **optimal F-measure** is $F(\tau^*)$: no binary classifier can be better.
- The optimal solution **satisfies** the following **condition**: $F(\tau^*) = 2\tau^*$.



[7] O. Koyejo, N. Natarajan, P. Ravikumar, and I. Dhillon. Consistent multilabel classification. In *NIPS*, 2015

Predictive model

- From the above analysis we can conclude:

Predictive model

- From the above analysis we can conclude:

  We need to train models that accurately estimate marginal probabilities or other related marginal quantities

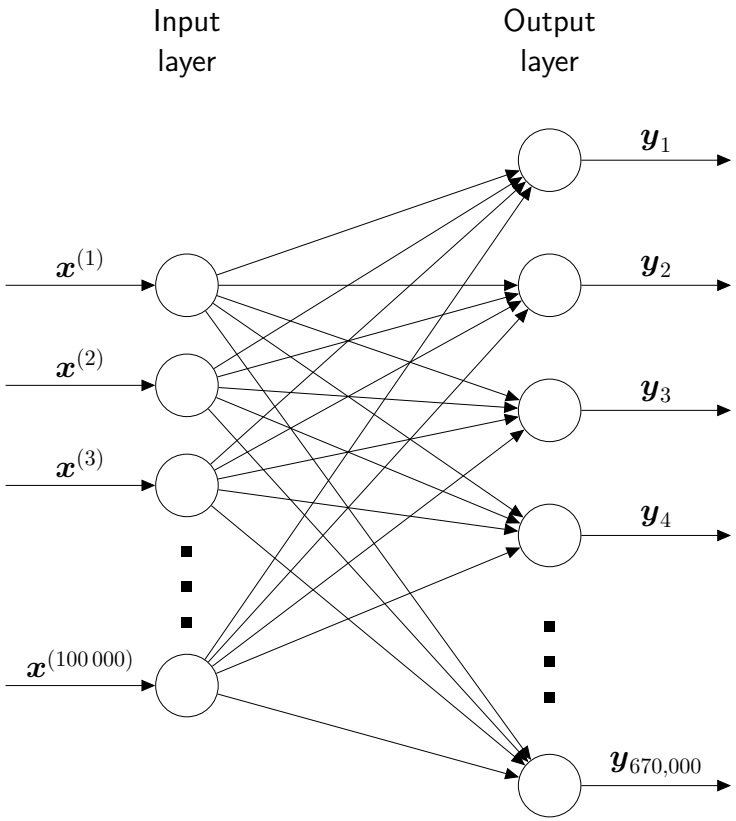
Agenda

1. Extreme classification: applications and challenges
2. **Algorithms**
3. Live demonstration

Label embedding methods

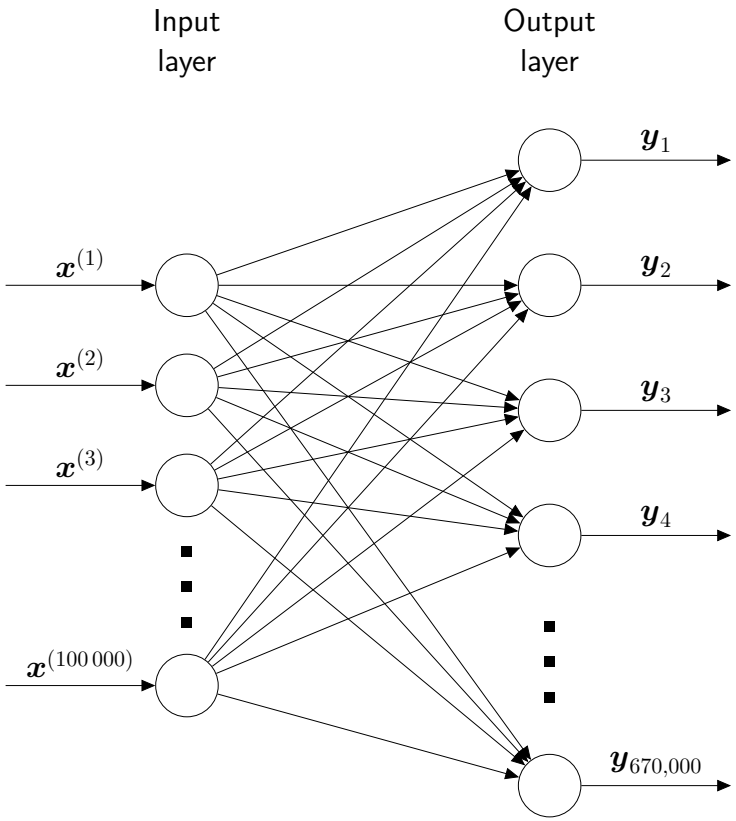
Low-dimensional Label Embeddings

- Shallow Networks - SVM
  - Direct mapping of input to output

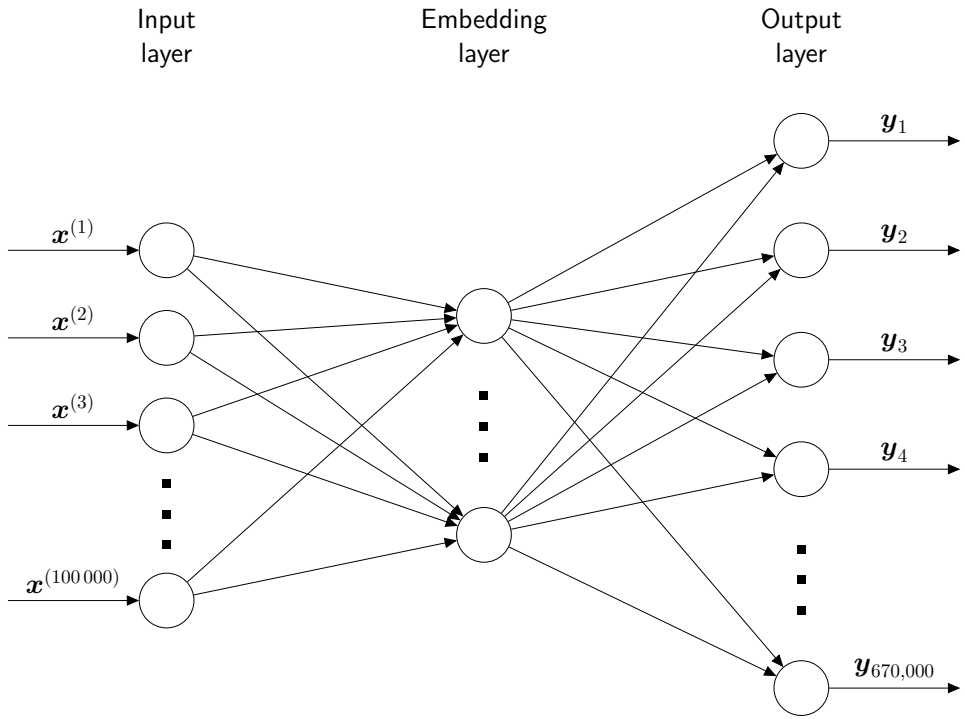


Low-dimensional Label Embeddings

- Shallow Networks - SVM
  - Direct mapping of input to output



- Label embedding
  - Mapping input to output via embedding layer



Low-dimensional Label Embeddings

Let $\boldsymbol{Y}$ be the label matrix of dimensionality $m \times n$ such that each row denotes the labels in an instance

- Label embedding methods - **Assume a low rank structure in the label matrix $\boldsymbol{Y}$** i.e. the m columns of the $\boldsymbol{Y}$ can be effectively represented by $\hat{m} \ll m$ columns

Low-dimensional Label Embeddings

Let $\boldsymbol{Y}$ be the label matrix of dimensionality $m \times n$ such that each row denotes the labels in an instance

- Label embedding methods - **Assume a low rank structure in the label matrix $\boldsymbol{Y}$** i.e. the m columns of the $\boldsymbol{Y}$ can be effectively represented by $\hat{m} \ll m$ columns

$$\begin{bmatrix} \text{——} \boldsymbol{y}_1 \text{——} \\ \text{——} \boldsymbol{y}_2 \text{——} \\ \vdots \\ \text{——} \boldsymbol{y}_n \text{——} \end{bmatrix}_{n \times m} \xrightarrow{\boldsymbol{U}} \begin{bmatrix} \text{——} \boldsymbol{z}_1 \text{——} \\ \text{——} \boldsymbol{z}_2 \text{——} \\ \vdots \\ \text{——} \boldsymbol{z}_n \text{——} \end{bmatrix}_{n \times \hat{m}}$$

Low-dimensional Label Embeddings

Let $\boldsymbol{Y}$ be the label matrix of dimensionality $m \times n$ such that each row denotes the labels in an instance

- Label embedding methods - **Assume a low rank structure in the label matrix $\boldsymbol{Y}$** i.e. the m columns of the $\boldsymbol{Y}$ can be effectively represented by $\hat{m} \ll m$ columns

$$\begin{bmatrix} \text{——} \boldsymbol{y}_1 \text{——} \\ \text{——} \boldsymbol{y}_2 \text{——} \\ \vdots \\ \text{——} \boldsymbol{y}_n \text{——} \end{bmatrix}_{n \times m} \xrightarrow{\boldsymbol{U}} \begin{bmatrix} \text{——} \boldsymbol{z}_1 \text{——} \\ \text{——} \boldsymbol{z}_2 \text{——} \\ \vdots \\ \text{——} \boldsymbol{z}_n \text{——} \end{bmatrix}_{n \times \hat{m}}$$

- Each label vector $\boldsymbol{y}_i \in \{0, 1\}^m$ is hence projected to $\boldsymbol{z}_i \in \mathbb{R}^{\hat{m}}$ by a projection matrix $\boldsymbol{U} \in \mathbb{R}^{m \times \hat{m}}$, such that $\boldsymbol{U}\boldsymbol{y}_i = \boldsymbol{z}_i$

Low-dimensional Label Embeddings

Let $\boldsymbol{Y}$ be the label matrix of dimensionality $m \times n$ such that each row denotes the labels in an instance

- Label embedding methods - **Assume a low rank structure in the label matrix $\boldsymbol{Y}$** i.e. the m columns of the $\boldsymbol{Y}$ can be effectively represented by $\hat{m} \ll m$ columns

$$\begin{bmatrix} \text{---} \boldsymbol{y}_1 \text{---} \\ \text{---} \boldsymbol{y}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{y}_n \text{---} \end{bmatrix}_{n \times m} \xRightarrow{\boldsymbol{U}} \begin{bmatrix} \text{---} \boldsymbol{z}_1 \text{---} \\ \text{---} \boldsymbol{z}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{z}_n \text{---} \end{bmatrix}_{n \times \hat{m}}$$

- Each label vector $\boldsymbol{y}_i \in \{0, 1\}^m$ is hence projected to $\boldsymbol{z}_i \in \mathbb{R}^{\hat{m}}$ by a projection matrix $\boldsymbol{U} \in \mathbb{R}^{m \times \hat{m}}$, such that $\boldsymbol{U}\boldsymbol{y}_i = \boldsymbol{z}_i$
- Regressors $\boldsymbol{V}$ are then trained to predict $\boldsymbol{z}_i = \boldsymbol{V}\boldsymbol{x}_i$

Low-dimensional Label Embeddings

- Otherwise, we may directly map the input $\boldsymbol{x}$ to its corresponding prediction $\hat{\boldsymbol{y}} = \boldsymbol{U}^\dagger \boldsymbol{V} \boldsymbol{x}$
- Here $\boldsymbol{U}^\dagger$ is the decompression matrix which lifts intermediary prediction $\boldsymbol{V}\boldsymbol{x}$ from the embedding space to original label space

Low-dimensional Label Embeddings

- Otherwise, we may directly map the input $\boldsymbol{x}$ to its corresponding prediction $\hat{\boldsymbol{y}} = \boldsymbol{U}^\dagger \boldsymbol{V} \boldsymbol{x}$
- Here $\boldsymbol{U}^\dagger$ is the decompression matrix which lifts intermediary prediction $\boldsymbol{V}\boldsymbol{x}$ from the embedding space to original label space

$$\begin{bmatrix} \text{---} \boldsymbol{x}_1 \text{---} \\ \text{---} \boldsymbol{x}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{x}_n \text{---} \end{bmatrix}_{n \times d} \overset{\boldsymbol{V}}{\Longrightarrow} \begin{bmatrix} \text{---} \boldsymbol{z}_1 \text{---} \\ \text{---} \boldsymbol{z}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{z}_n \text{---} \end{bmatrix}_{n \times \hat{m}} \overset{\boldsymbol{U}^\dagger}{\Longrightarrow} \begin{bmatrix} \text{---------} \boldsymbol{y}_1 \text{---------} \\ \text{---------} \boldsymbol{y}_2 \text{---------} \\ \vdots \\ \text{---------} \boldsymbol{y}_n \text{---------} \end{bmatrix}_{n \times m}$$

Label Embedding Methods

Various methods differ in terms of the embedding/compression and de-compression methods they employ

- Methods for label embedding
  - Compressed Sensing [8]
  - Label Embedding for Missing Labels (LEML) [9]
  - Sparse Local Embeddings for Extreme Classification (SLEEC) [10]

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

[9] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit S. Dhillon. Large-scale Multi-label Learning with Missing Labels. In *ICML*, 2014

[10] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *NIPS*, 2015

LEML - Global Embeddings

LEML learns global embeddings under the assumption :

- The label matrix is low rank

LEML - Global Embeddings

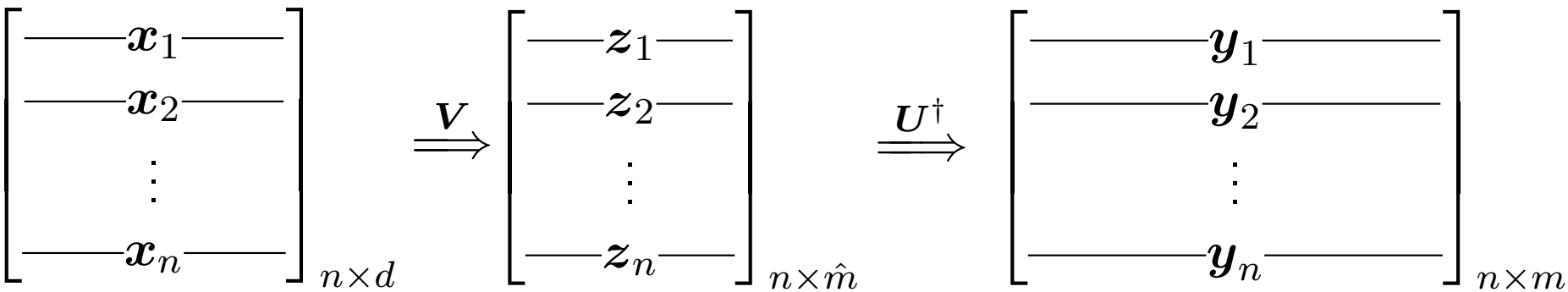
LEML learns global embeddings under the assumption :

- The label matrix is low rank
- Preserve global distances

LEML - Global Embeddings

LEML learns global embeddings under the assumption :

- The label matrix is low rank
- Preserve global distances



LEML - Global Embeddings

LEML Optimization - Direct mapping of input to output

$$\min_{\boldsymbol{V},\boldsymbol{U}^\dagger} \sum_{i=1}^{n} \ell(\boldsymbol{y}_i, \boldsymbol{U}^\dagger V \boldsymbol{x}_i) + \lambda ||\boldsymbol{V}||_F^2 + \lambda ||\boldsymbol{U}^\dagger||_F^2$$

$$\begin{bmatrix} \text{---} \boldsymbol{x}_1 \text{---} \\ \text{---} \boldsymbol{x}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{x}_n \text{---} \end{bmatrix}_{n \times d} \xRightarrow{\boldsymbol{V}} \begin{bmatrix} \text{---} \boldsymbol{z}_1 \text{---} \\ \text{---} \boldsymbol{z}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{z}_n \text{---} \end{bmatrix}_{n \times \hat{m}} \xRightarrow{\boldsymbol{U}^\dagger} \begin{bmatrix} \text{---------} \boldsymbol{y}_1 \text{---------} \\ \text{---------} \boldsymbol{y}_2 \text{---------} \\ \vdots \\ \text{---------} \boldsymbol{y}_n \text{---------} \end{bmatrix}_{n \times m}$$

LEML - Global Embeddings

LEML Optimization - Direct mapping of input to output

$$\min_{\boldsymbol{V},\boldsymbol{U}^\dagger} \sum_{i=1}^{n} \ell(\boldsymbol{y}_i, \boldsymbol{U}^\dagger V \boldsymbol{x}_i) + \lambda ||\boldsymbol{V}||_F^2 + \lambda ||\boldsymbol{U}^\dagger||_F^2$$

$$\begin{bmatrix} \text{---} \boldsymbol{x}_1 \text{---} \\ \text{---} \boldsymbol{x}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{x}_n \text{---} \end{bmatrix}_{n \times d} \xRightarrow{\boldsymbol{V}} \begin{bmatrix} \text{---} \boldsymbol{z}_1 \text{---} \\ \text{---} \boldsymbol{z}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{z}_n \text{---} \end{bmatrix}_{n \times \hat{m}} \xRightarrow{\boldsymbol{U}^\dagger} \begin{bmatrix} \text{---------} \boldsymbol{y}_1 \text{---------} \\ \text{---------} \boldsymbol{y}_2 \text{---------} \\ \vdots \\ \text{---------} \boldsymbol{y}_n \text{---------} \end{bmatrix}_{n \times m}$$

- The above optimization problem is non-convex

LEML - Global Embeddings

LEML Optimization - Direct mapping of input to output

$$\min_{\boldsymbol{V},\boldsymbol{U}^\dagger} \sum_{i=1}^{n} \ell(\boldsymbol{y}_i, \boldsymbol{U}^\dagger V \boldsymbol{x}_i) + \lambda ||\boldsymbol{V}||_F^2 + \lambda ||\boldsymbol{U}^\dagger||_F^2$$

$$\begin{bmatrix} \text{---} \boldsymbol{x}_1 \text{---} \\ \text{---} \boldsymbol{x}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{x}_n \text{---} \end{bmatrix}_{n \times d} \xRightarrow{\boldsymbol{V}} \begin{bmatrix} \text{---} \boldsymbol{z}_1 \text{---} \\ \text{---} \boldsymbol{z}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{z}_n \text{---} \end{bmatrix}_{n \times \hat{m}} \xRightarrow{\boldsymbol{U}^\dagger} \begin{bmatrix} \text{---------} \boldsymbol{y}_1 \text{---------} \\ \text{---------} \boldsymbol{y}_2 \text{---------} \\ \vdots \\ \text{---------} \boldsymbol{y}_n \text{---------} \end{bmatrix}_{n \times m}$$

- The above optimization problem is non-convex
- Fixing one of $\boldsymbol{U}^\dagger$ or $\boldsymbol{V}$ leads convex optimization

LEML - Global Embeddings

LEML Optimization - Direct mapping of input to output

$$\min_{\boldsymbol{V},\boldsymbol{U}^\dagger} \sum_{i=1}^{n} \ell(\boldsymbol{y}_i, \boldsymbol{U}^\dagger V \boldsymbol{x}_i) + \lambda ||\boldsymbol{V}||_F^2 + \lambda ||\boldsymbol{U}^\dagger||_F^2$$

$$\begin{bmatrix} \text{---} \boldsymbol{x}_1 \text{---} \\ \text{---} \boldsymbol{x}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{x}_n \text{---} \end{bmatrix}_{n \times d} \xRightarrow{\boldsymbol{V}} \begin{bmatrix} \text{---} \boldsymbol{z}_1 \text{---} \\ \text{---} \boldsymbol{z}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{z}_n \text{---} \end{bmatrix}_{n \times \hat{m}} \xRightarrow{\boldsymbol{U}^\dagger} \begin{bmatrix} \text{---} \boldsymbol{y}_1 \text{---} \\ \text{---} \boldsymbol{y}_2 \text{---} \\ \vdots \\ \text{---} \boldsymbol{y}_n \text{---} \end{bmatrix}_{n \times m}$$

- The above optimization problem is non-convex
- Fixing one of $\boldsymbol{U}^\dagger$ or $\boldsymbol{V}$ leads convex optimization
- Alternating minimization over $\boldsymbol{U}^\dagger$ and $\boldsymbol{V}$ is performed

LEML - Pros and Cons

- Advantages of label embedding
  - Exploit label correlations
  - Ease of implementation

LEML - Pros and Cons

- Advantages of label embedding
  - Exploit label correlations
  - Ease of implementation
- Disadvantages of label embedding
  - Assuption fails for long tail of labels
  - Prediction complexity - $\Omega(\hat{m}(m + \hat{d}))$

SLEEC - Sparse Local Embeddings for Extreme Multi-label Classification

- The embedding matrix $\boldsymbol{Z} \in \mathbb{R}^{\hat{m} \times n}$ into which the original labels are projected is learnt by

$$\min_{\boldsymbol{Z} \in \hat{m} \times n} ||P_{\Omega}(\boldsymbol{Y}^T \boldsymbol{Y}) - P_{\Omega}(\boldsymbol{Z}^T \boldsymbol{Z})||_F^2 + \lambda ||\boldsymbol{Z}||_1$$

  where Ω denotes the index set of nearest neighbors $(i, j) \in \Omega$ iff $j \in \mathcal{N}_i$, and $\mathcal{N}_i$ is the set of nearest neighbors of instance i.

SLEEC - Sparse Local Embeddings for Extreme Multi-label Classification

- The embedding matrix $\boldsymbol{Z} \in \mathbb{R}^{\hat{m} \times n}$ into which the original labels are projected is learnt by

$$\min_{\boldsymbol{Z} \in \hat{m} \times n} ||P_\Omega(\boldsymbol{Y}^T \boldsymbol{Y}) - P_\Omega(\boldsymbol{Z}^T \boldsymbol{Z})||_F^2 + \lambda ||\boldsymbol{Z}||_1$$

  where Ω denotes the index set of nearest neighbors $(i, j) \in \Omega$ iff $j \in \mathcal{N}_i$, and $\mathcal{N}_i$ is the set of nearest neighbors of instance i.

$$\begin{bmatrix} \text{——} \boldsymbol{y}_1 \text{——} \\ \text{——} \boldsymbol{y}_2 \text{——} \end{bmatrix}_{2 \times m} \xRightarrow{\boldsymbol{U}_1} \begin{bmatrix} \text{——} \boldsymbol{z}_1 \text{——} \\ \text{——} \boldsymbol{z}_2 \text{——} \end{bmatrix}_{2 \times \hat{m}}$$

$$\begin{bmatrix} \text{——} \boldsymbol{y}_{n-2} \text{——} \\ \text{——} \boldsymbol{y}_{n-1} \text{——} \\ \text{——} \boldsymbol{y}_n \text{——} \end{bmatrix}_{3 \times m} \xRightarrow{\boldsymbol{U}_k} \begin{bmatrix} \text{——} \boldsymbol{z}_{n-2} \text{——} \\ \text{——} \boldsymbol{z}_{n-1} \text{——} \\ \text{——} \boldsymbol{z}_n \text{——} \end{bmatrix}_{3 \times \hat{m}}$$

SLEEC - Sparse Local Embeddings for Extreme Multi-label Classification

- **Sparsity** in embedding is obtained by ℓ_1 regularization $||\boldsymbol{Z}||_1$

SLEEC - Sparse Local Embeddings for Extreme Multi-label Classification

- **Sparsity** in embedding is obtained by ℓ_1 regularization $||\boldsymbol{Z}||_1$
- **Locality** in the embedding is based on following a nearest neighbors approach and is related to $\mathcal{N}_i$ given by

$$\mathcal{N}_i = \underset{S,|S|\leq\alpha.n}{\arg\max} \sum_{j\in S} (\boldsymbol{Y}_i^T \boldsymbol{Y}_j)$$

$$\begin{bmatrix} \boldsymbol{Y}_1 \\ \boldsymbol{Y}_2 \end{bmatrix}_{2\times m} \xrightarrow{\boldsymbol{U}_1} \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{bmatrix}_{2\times \hat{m}}$$

$$\begin{bmatrix} \boldsymbol{Y}_{n-2} \\ \boldsymbol{Y}_{n-1} \\ \boldsymbol{Y}_n \end{bmatrix}_{3\times m} \xrightarrow{\boldsymbol{U}_k} \begin{bmatrix} \boldsymbol{z}_{n-2} \\ \boldsymbol{z}_{n-1} \\ \boldsymbol{z}_n \end{bmatrix}_{3\times \hat{m}}$$

SLEEC - Sparse Local Embeddings for Extreme Multi-label Classification

Overall optimization

- The overall SLEEC objective is therefore given by

$$\min_{\boldsymbol{V} \in \mathbb{R}^{\hat{m} \times d}} ||P_{\Omega}(\boldsymbol{Y}^T \boldsymbol{Y}) - P_{\Omega}(\boldsymbol{X}^T \boldsymbol{V}^T \boldsymbol{V} \boldsymbol{X})||_F^2 + \lambda ||\boldsymbol{V}||_F^2 + \mu ||\boldsymbol{V} \boldsymbol{X}||_1$$

- To optimize above is challenging due to non-convexity, non-differentiability and scale of the problem

SLEEC - Sparse Local Embeddings for Extreme Multi-label Classification

- Simplification
  - Finding the embedding matrix, without ℓ_1 regularization

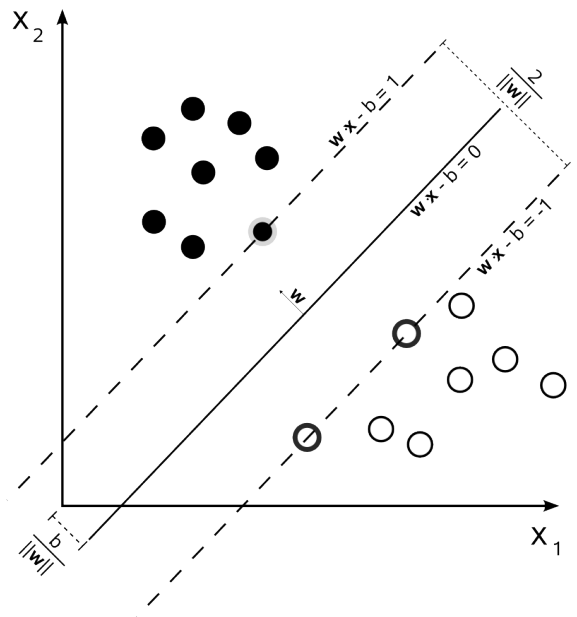
$$\min_{\boldsymbol{Z} \in \hat{m} \times n} ||P_\Omega(\boldsymbol{Y}^T \boldsymbol{Y}) - P_\Omega(\boldsymbol{Z}^T \boldsymbol{Z})||_F^2 \equiv \min_{\boldsymbol{M} \succeq 0, rank(\boldsymbol{M}) \leq \hat{m}} ||P_\Omega(\boldsymbol{Y}^T \boldsymbol{Y}) - P_\Omega(\boldsymbol{M})||$$

    where $\boldsymbol{M} = \boldsymbol{Z}^T \boldsymbol{Z}$
  - After learning $\boldsymbol{Z}$ in the above step,

$$\min_{\boldsymbol{V} \in \mathbb{R}^{\hat{m} \times d}} ||\boldsymbol{Z} - \boldsymbol{V}\boldsymbol{X}||_F^2 + \lambda ||\boldsymbol{V}||_F^2 + \mu ||\boldsymbol{V}\boldsymbol{X}||_1$$

Smart 1-vs-all approaches

Learning Linear Decision Boundaries



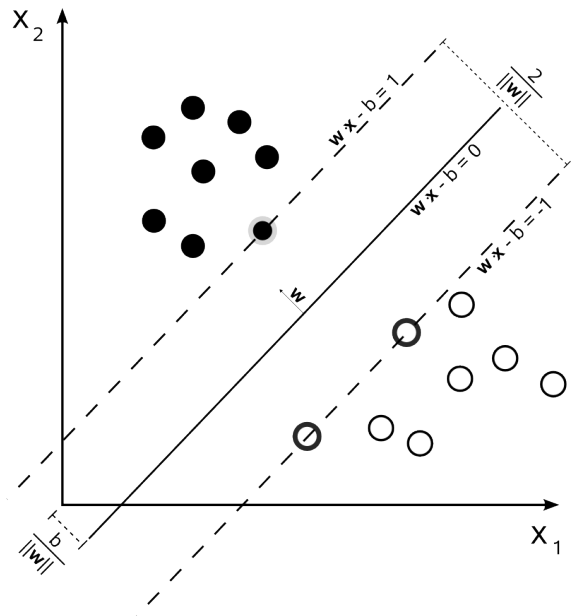
Learning process typically involves finding the optimal linear separator as follows:

- Template of the optimization problem being solved

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} R_{emp}(\boldsymbol{w}) + \lambda \; Reg(\boldsymbol{w})$$

  where $Reg(\boldsymbol{w})$ is the regularization term to avoid complex models and $R_{emp}(.)$ represents the empirical

Learning Linear Decision Boundaries



Learning process typically involves finding the optimal linear separator as follows:

- For Support Vector Machine - using squared hinge loss

$$\arg\min_{\boldsymbol{w}} \frac{\lambda}{2}||\boldsymbol{w}||^2 + \sum_{i=1}^{n}(\max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i))^2$$

Bag of Words Data Representation

Top-3 training instances and their corresponding labels from a Wikipedia Subset

```
1,2,3,4,5 1:0.1498 2:0.1216 3:0.0521 4:0.0637 5:0.0517 6:0.0891 7:0.0994 8:0.1944 9:0.3821 10:0.093
8 11:0.0361 12:0.186 13:0.0859 14:0.0641 15:0.0616 16:0.0821 17:0.1085 18:0.0636 19:0.0714 20:0.076
8 21:0.1532 22:0.0491 23:0.1017 24:0.1542 25:0.1136 26:0.0965 27:0.0473 28:0.1223 29:0.0936 30:0.09
29 31:0.1665 32:0.1444 33:0.0581 34:0.0622 35:0.0578 36:0.1462 37:0.0704 38:0.0623 39:0.2466 40:0.1
539 41:0.0992 42:0.071 43:0.1106 44:0.0849 45:0.1113 46:0.0725 47:0.1093 48:0.073 49:0.1415 50:0.09
23 51:0.0547 52:0.1022 53:0.1009 54:0.0431 55:0.0889 56:0.1311 57:0.0637 58:0.1108 59:0.0656 60:0.0
873 61:0.0813 62:0.0864 63:0.2023 64:0.2295 65:0.067 66:0.067 67:0.0614 68:0.0447 69:0.0878 70:0.06
8 71:0.0711 72:0.0755 73:0.0598 74:0.1457 75:0.058 76:0.0862 77:0.0748 78:0.0351 79:0.0552 477142:1

6,7,8,9 14:0.0562 27:0.0414 28:0.0633 42:0.0623 51:0.1006 54:0.0639 63:0.068 66:0.0587 71:0.0624 80
:0.0799 81:0.0477 82:0.1125 83:0.0356 84:0.0627 85:0.0989 86:0.0517 87:0.0395 88:0.3179 89:0.1795 9
0:0.0576 91:0.0916 92:0.0349 93:0.069 94:0.1272 95:0.0683 96:0.0836 97:0.0427 98:0.0502 99:0.046 10
0:0.1292 101:0.0429 102:0.048 103:0.0469 104:0.0839 105:0.0737 106:0.1424 107:0.0534 108:0.0818 109
:0.0804 110:0.0917 111:0.1455 112:0.1106 113:0.1168 114:0.0621 115:0.0995 116:0.0526 117:0.0891 118
:0.0714 119:0.0529 120:0.0555 121:0.0891 122:0.07 123:0.0525 124:0.1049 125:0.0255 126:0.0452 127:0
.0789 128:0.0503 129:0.0959 130:0.3668 131:0.0821 132:0.0715 133:0.2504 134:0.0647 135:0.0521 136:0
.0988 137:0.0579 138:0.0449 139:0.0682 140:0.0683 141:0.0694 142:0.0797 143:0.0615 144:0.0826 145:0
.1357 146:0.0608 147:0.0629 148:0.0724 149:0.0884 150:0.063 151:0.0537 152:0.0481 153:0.1023 154:0.
0436 155:0.0609 156:0.0338 157:0.0565 158:0.0628 159:0.1356 160:0.221 161:0.0468 162:0.0684 163:0.0
659 164:0.0483 165:0.0884 166:0.063 167:0.1083 168:0.0503 169:0.0853 170:0.0794 171:0.0364 172:0.07
26 173:0.0414 174:0.0744 175:0.1535 176:0.0482 177:0.0417 178:0.0629 179:0.083 180:0.0709 181:0.050
8 182:0.1009 477142:1
10,11,12,13,14,15,16 34:0.0992 125:0.0463 173:0.0752 183:0.0718 184:0.0924 185:0.219 186:0.0927 187
:0.1781 188:0.1625 189:0.135 190:0.0868 191:0.1272 192:0.103 193:0.1285 194:0.1504 195:0.1447 196:0
.1869 197:0.3767 198:0.1646 199:0.4598 200:0.2042 201:0.0736 202:0.1852 203:0.1223 204:0.0782 205:0
.2479 206:0.0893 207:0.1112 208:0.0861 209:0.1697 210:0.1213 211:0.2672 212:0.1393 477142:1
```

Computational Challenge - Big Data

	#Labels (m)	#Features (d)	#Instances (n)	Parameters ($m \times d$)	Training Data	Model size
DMOZ[11]	**12,294**	347,256	93,805	4,269,165,264	125MB	17 GB
Wikipedia	**325,056**	1,617,899	1,778,351	525,907,777,344	1GB	870GB
Delicious[12]	**205,443**	782,585	196,606	160,776,610,155	1GB	350GB
Amazon	**670,091**	135,909	490,449	91,071,397,719	600MB	250GB

- Naive application of binary one-vs-rest linear classification
  - Billions of parameters - Computational complexity
  - TeraBytes of disk space to store the model - Space Complexity

[11] From LSHTC challenge

[12] From XMC repository

Distribution of Learnt Weights

Weights ($\boldsymbol{W}_{d',m'}$) learnt by using One-vs-rest SVM for Wiki-31K dataset (101k dimensional data with 31k labels) from XMC repository

$$\min_{\boldsymbol{w}_{m'}} \left[||\boldsymbol{w}_{m'}||_2^2 + C \sum_{i=1}^{n} (\max(0, 1 - s_{m'_i} \boldsymbol{w}_{m'}^T \boldsymbol{x}_i))^2 \right]$$

$s_{m'_i} = +1$ if instance i has label m', -1 otherwise .

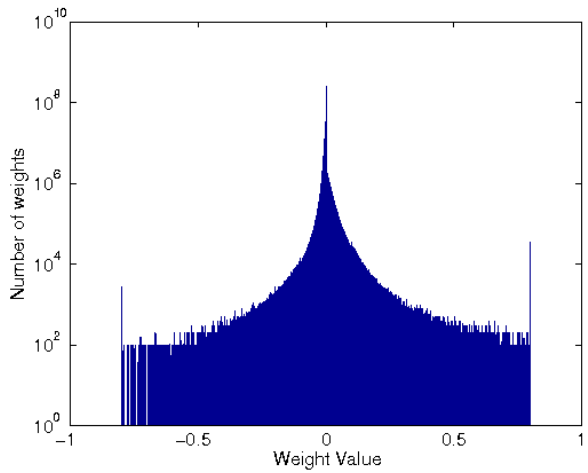


Figure: Distribution of learnt weights before pruning

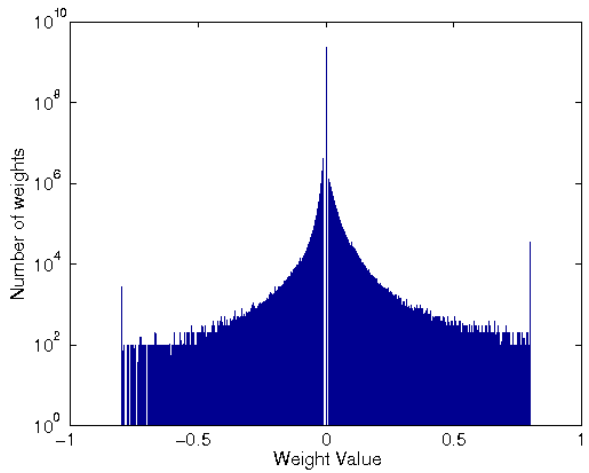


Figure: Distribution of learnt weights after pruning small weights

Distribution of Learnt Weights

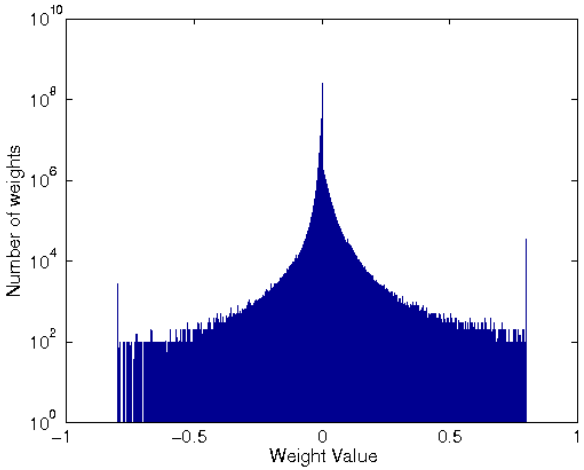


Figure: Distribution of learnt weights before pruning

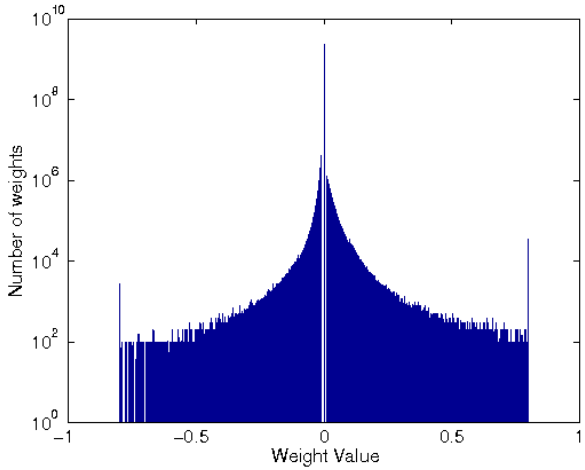


Figure: Distribution of learnt weights after pruning small weights

- Of the 3 Billion weights, 97% are s.t, $W_{d',m'} \leq |0.01|$, and hence non-discriminative
- Storing them leads to large model sizes but no benefit in classification

DiSMEC - Distributed Sparse Machines for XMC

Require: Training data $\mathcal{T} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1) \ldots (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, input dimensionality d, label set $\{1 \ldots m\}$, $B = \lfloor \frac{m}{1000} \rfloor + 1$ and Δ

Ensure: Learnt matrix $\boldsymbol{W}_{d,m}$ in sparse format

Load single copy of input vectors $\boldsymbol{X} = \{\boldsymbol{x}_1 \ldots \boldsymbol{x}_n\}$ in the main memory

Load binary sign vectors $\boldsymbol{s}_{m'} = \{+1, -1\}_{i=1}^{n}$ separately for each label

for $\{b = 0; b < B; b++\}$ **do** ▷ *1st parallelization*

    #pragma omp parallel for private(m') ▷ *2nd parallelization*

    for $\{m' = b \times 1000; m' \leq (b+1) \times 1000; m'++\}$ **do**

        Using $(\boldsymbol{X}, \boldsymbol{s}_{m'})$, train weight vector $\boldsymbol{w}_{m'}$ on a single core

        Prune ambiguous weights in $\boldsymbol{w}_{m'}$ ▷ Model reduction

    return $\boldsymbol{W}_{d,1000}$ ▷ *Learnt matrix for a batch on one node*

return $\boldsymbol{W}_{d,m}$ ▷ *Learnt matrix from all the nodes*

- Learns model for LSHTCWiki-325K in 6 hours on 400 cores
- Model size is 3GB due to pruning step

PDSparse - Primal Dual Sparsity for Extreme Classification [13]

Primal Dual Sparsity in Max-margin Multi-label SVM Formulation

- For prediction score $\boldsymbol{z} \in \mathbb{R}^m$ and true label $\boldsymbol{y} \in \{1, \ldots, m\}$, $L(\boldsymbol{z}, \boldsymbol{y}) = \max_{m_- \in \{\mathcal{N}(\boldsymbol{y})\}, m_+ \in \{\mathcal{P}(\boldsymbol{y})\}} (1 + z_{m_-} - z_{m_+})_+$, where $\mathcal{N}(\boldsymbol{y})$ and $\mathcal{P}(\boldsymbol{y})$ are respectively the sets of negative and positive labels for that instance

[13] Ian E.H. Yen, Xiangru Huang, Kai Zhong, Pradeep Ravikumar, and Inderjit S. Dhillon. PD-Sparse: A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification. In *ICML*, 2016

PDSparse - Primal Dual Sparsity for Extreme Classification [13]

Primal Dual Sparsity in Max-margin Multi-label SVM Formulation

- For prediction score $\boldsymbol{z} \in \mathbb{R}^m$ and true label $\boldsymbol{y} \in \{1, \ldots, m\}$, $L(\boldsymbol{z}, \boldsymbol{y}) = \max_{m_- \in \{\mathcal{N}(\boldsymbol{y})\}, m_+ \in \{\mathcal{P}(\boldsymbol{y})\}} (1 + z_{m_-} - z_{m_+})_+$, where $\mathcal{N}(\boldsymbol{y})$ and $\mathcal{P}(\boldsymbol{y})$ are respectively the sets of negative and positive labels for that instance
- $\boldsymbol{W}^* \in \mathbb{R}^{d \times m}$ obtained by $\sum_{i=1}^n L(\boldsymbol{W}^T \boldsymbol{x}_i, \boldsymbol{y}_i)$ is determined by the scores of *support labels* :

$$(m_-, m_+) = \underset{m_- \in \{\mathcal{N}(\boldsymbol{y})\}, m_+ \in \{\mathcal{P}(\boldsymbol{y})\}}{\arg\max} (1 + z_{m_-} - z_{m_+})_+$$

that attain the above maximum

[13] Ian E.H. Yen, Xiangru Huang, Kai Zhong, Pradeep Ravikumar, and Inderjit S. Dhillon. PD-Sparse: A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification. In *ICML*, 2016

PDSparse - Primal Dual Sparsity for Extreme Classification [13]

Primal Dual Sparsity in Max-margin Multi-label SVM Formulation

- For prediction score $\boldsymbol{z} \in \mathbb{R}^m$ and true label $\boldsymbol{y} \in \{1, \ldots, m\}$, $L(\boldsymbol{z}, \boldsymbol{y}) = \max_{m_- \in \{\mathcal{N}(\boldsymbol{y})\}, m_+ \in \{\mathcal{P}(\boldsymbol{y})\}} (1 + z_{m_-} - z_{m_+})_+$, where $\mathcal{N}(\boldsymbol{y})$ and $\mathcal{P}(\boldsymbol{y})$ are respectively the sets of negative and positive labels for that instance
- $\boldsymbol{W}^* \in \mathbb{R}^{d \times m}$ obtained by $\sum_{i=1}^{n} L(\boldsymbol{W}^T \boldsymbol{x}_i, \boldsymbol{y}_i)$ is determined by the scores of *support labels* :

$$(m_-, m_+) = \underset{m_- \in \{\mathcal{N}(\boldsymbol{y})\}, m_+ \in \{\mathcal{P}(\boldsymbol{y})\}}{\arg\max} (1 + z_{m_-} - z_{m_+})_+$$

  that attain the above maximum
- On the dual side, since the max is attained for very few label pairs (m_-, m_+), the dual variable is non-zero very infrequently i.e. $nnz(\boldsymbol{\alpha}_i) << m$

[13] Ian E.H. Yen, Xiangru Huang, Kai Zhong, Pradeep Ravikumar, and Inderjit S. Dhillon. PD-Sparse: A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification. In *ICML*, 2016

Primal Dual Sparsity in Max-margin Multi-label SVM Formulation

- On the primal side, this is achieved by ℓ_1 regularization

$$\boldsymbol{W}^* \in \arg\min_{\boldsymbol{W}} \lambda \sum_{m'=1}^{m} ||\boldsymbol{w}_{m'}||_1 + \sum_{i=1}^{n} L(\boldsymbol{W}^T \mathbf{x}_i, \boldsymbol{y}_i)$$

then it satisfies $d \times m_w = nnz(\boldsymbol{W}^*) \leq nnz(\boldsymbol{A}^*) = n \times m_a$, where m_w are average number of active labels per feature in the optimal weight matrix $\boldsymbol{W}^*$.

PDSparse - Primal Dual Sparsity for Extreme Classification

Primal Dual Sparsity in Max-margin Multi-label SVM Formulation

- For ease of optimization, the following *strongly convex* elastic net regularized problem is solved

$$\arg\min_{\boldsymbol{W}} \sum_{m'=1}^{m} ||\boldsymbol{w}_{m'}||_2^2 + \lambda \sum_{m'=1}^{m} ||\boldsymbol{w}_{m'}||_1 + \sum_{i=1}^{n} L(\boldsymbol{W}^T \boldsymbol{x}_i, \boldsymbol{y}_i)$$

  which is empirically observed to give a similar sparsity level as the ℓ_1 regularized problem

PDSparse - Primal Dual Sparsity for Extreme Classification

Simultaneous primal dual updates

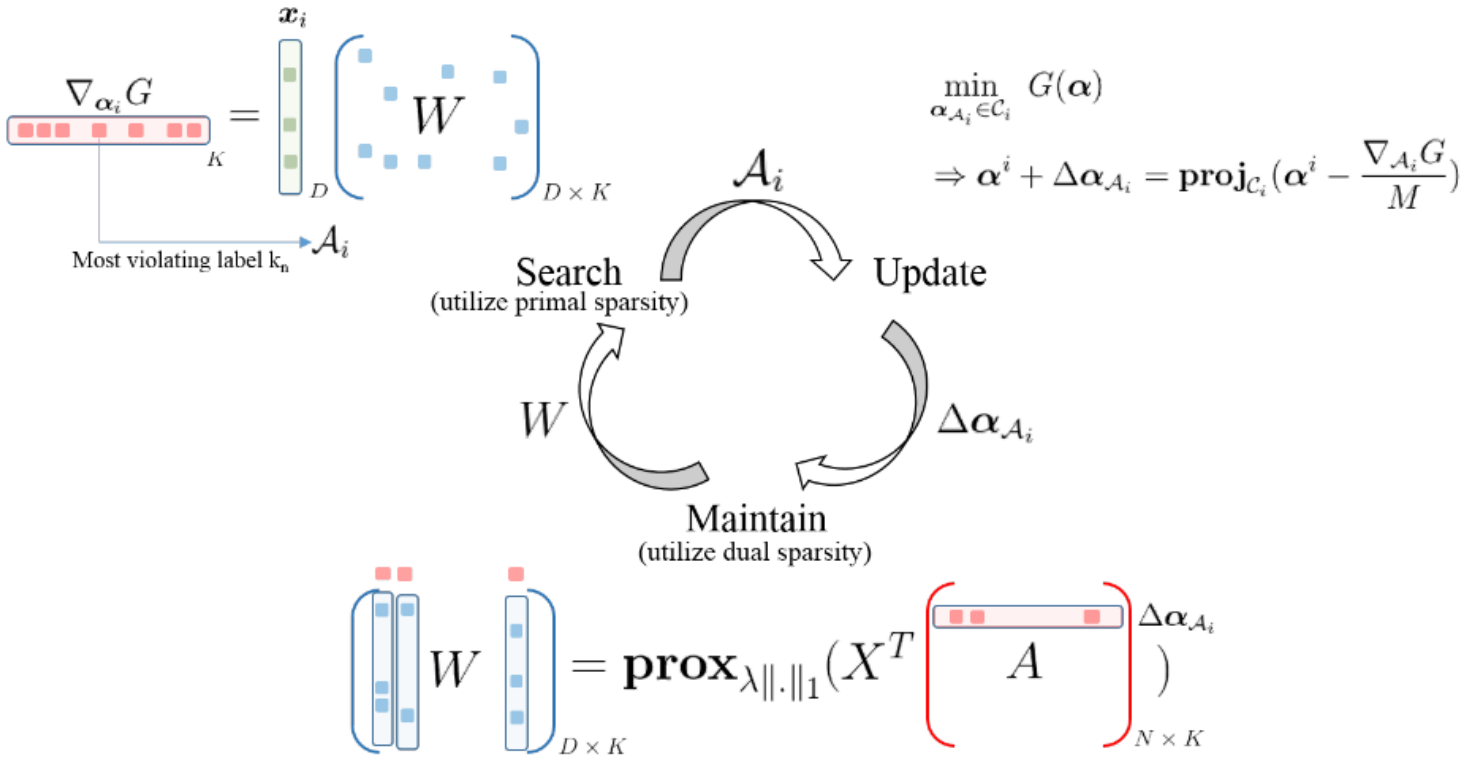


Figure: Simultaneous Primal Dual Updates in PDSparse

Dataset and Evaluation Metrics for Comparison

Datasets taken from Extreme Classification repository

Dataset	# Training	# Test	# Categories	# Features		
					APpL	ALpP
Amazon-13K	1,186,239	306,782	**13,330**	203,882	448.5	5.04
Amazon-14K	4,398,050	1,099,725	**14,588**	597,540	1330.1	3.53
Wikipedia-31K	14,146	6,616	**30,938**	101,938	8.5	18.6
Delicious-200K	196,606	100,095	**205,443**	1,123,497	72.3	75.5
WikiLSHTC-325K	1,778,351	587,084	**325,056**	1,617,899	17.4	3.2
Wikipedia-500K	1,813,391	783,743	**501,070**	2,381,304	24.7	4.7
Amazon-670K	490,499	153,025	**670,091**	135909	3.9	5.4

Table: Multi-label datasets taken from the Extreme Classification Repository. APpL and ALpP represent average points per label and average labels per point.

For true label vector $\boldsymbol{y}$, and predicted vector $\hat{\boldsymbol{y}}$:

$$\text{precision@k} := \frac{1}{k} \sum_{l \in rank_k(\hat{\boldsymbol{y}})} \boldsymbol{y}_l \quad ; \quad \text{nDCG@k} := \frac{\text{DCG@k}}{\sum_{l=1}^{\min(k, ||\boldsymbol{y}||_0)} \frac{1}{\log(l+1)}}$$

where $\text{DCG@k} := \frac{1}{k} \sum_{l \in rank_k(\hat{\boldsymbol{y}})} \frac{\boldsymbol{y}_l}{\log(l+1)}$, and $rank_k(\boldsymbol{y})$ returns the k largest indices of $\boldsymbol{y}$ ranked in descending order, and $||\boldsymbol{y}||_0$ returns the 0-norm of the true-label vector.

Comparison among state-of-the-art methods on Precision@k metric

Dataset	Proposed approach	Embedding based approaches			Tree based approaches			Sparsity inducing approaches	
	DiSMEC	SLEEC	LEML	RobustXML	Fast-XML	LPSR-NB	PLT	PD-Sparse	L1-SVM
Amazon-13K									
P@1	**93.4**	90.4	78.2	88.4	92.9	75.1	91.4	91.1	91.8
P@3	**79.1**	75.8	65.4	74.6	77.5	60.2	75.8	76.4	77.8
P@5	**64.1**	61.3	55.7	60.6	62.5	57.3	61.0	63.0.8	62.9
Amazon-14K									
P@1	**91.0**	80.3	75.2	83.2	90.3	74.2	86.4	88.4	88.2
P@3	**70.3**	67.2	62.5	66.4	70.1	55.7	65.2	68.1	67.6
P@5	**55.9**	50.6	40.8	52.3	55.4	44.3	50.7	50.5	51.2
Wikipedia-31K									
P@1	85.2	85.5	73.5	**85.5**	82.5	72.7	84.3	73.8	83.2
P@3	**74.6**	73.6	62.3	74.0	66.6	58.5	72.3	60.9	72.1
P@5	**65.9**	63.1	54.3	63.8	56.7	49.4	62.7	50.4	63.7
Delicious-200k									
P@1	45.5	**47.0**	40.3	45.0	42.8	18.6	45.3	41.2	42.1
P@3	38.7	**41.6**	37.7	40.0	38.7	15.4	38.9	35.3	34.8
P@5	35.5	**38.8**	36.6	38.0	36.3	14.0	35.8	31.2	30.4
WikiLSHTC-325K									
P@1	**64.4**	55.5	19.8	53.5	49.3	27.4	45.6	58.2	60.6
P@3	**42.5**	33.8	11.4	31.8	32.7	16.4	29.1	36.3	38.6
P@5	**31.5**	24.0	8.4	29.9	24.0	12.0	21.9	28.7	28.5
Wiki-500K									
P@1	**70.2**	48.2	41.3	-	54.1	38.2	51.5	-	65.3
P@3	**50.6**	29.4	30.1	-	35.5	29.3	35.7	-	46.1
P@5	**39.7**	21.2	19.8	-	26.2	18.7	27.7	-	35.3
Amazon-670K									
P@1	**44.7**	35.0	8.1	31.0	33.3	28.6	36.6	-	39.8
P@3	**39.7**	31.2	6.8	28.0	29.3	24.9	32.1	-	34.3
P@5	**36.1**	28.5	6.0	24.0	26.1	22.3	28.8	-	30.1

Table: Comparison of Precision@k for k=1,3 and 5

Comparison among methods on nDCG@k metric

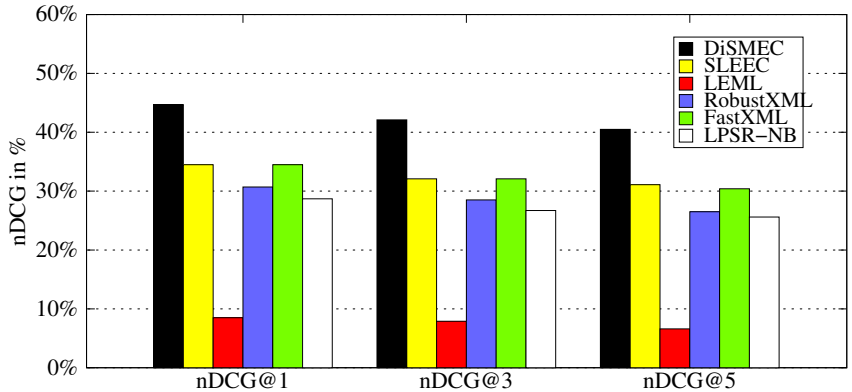


Figure: nDCG for Amazon-670K

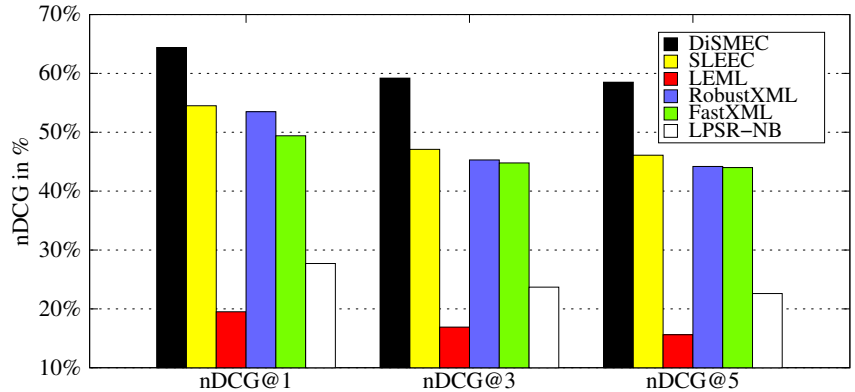


Figure: nDCG for WikiLSHTC-325K

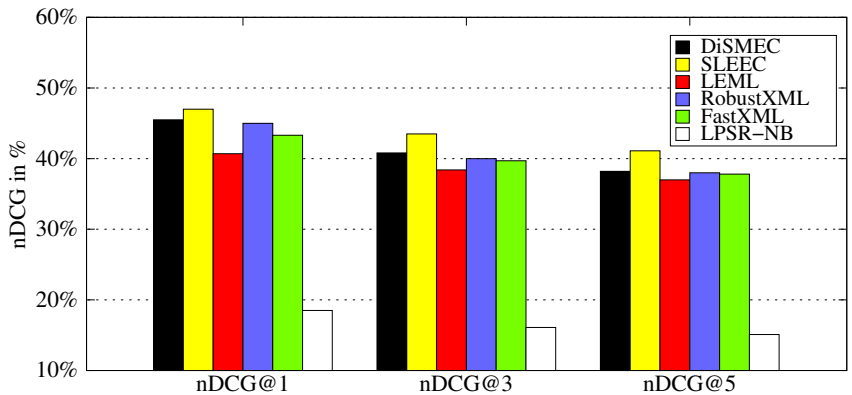


Figure: nDCG for Delicious-200K

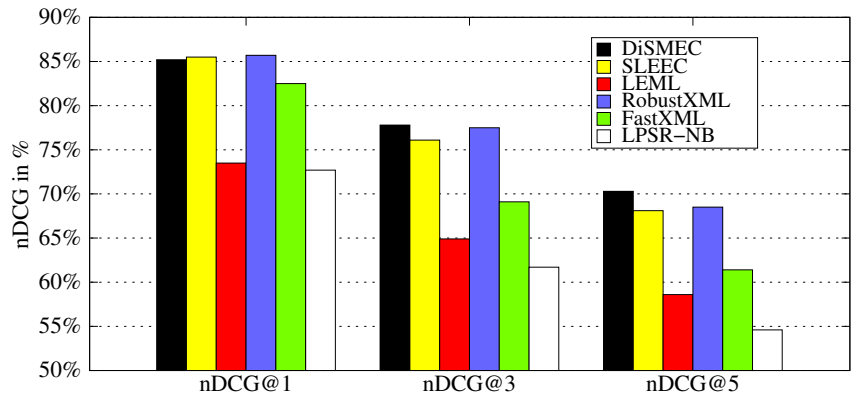


Figure: nDCG for Wiki-30K

Impact of Δ parameter

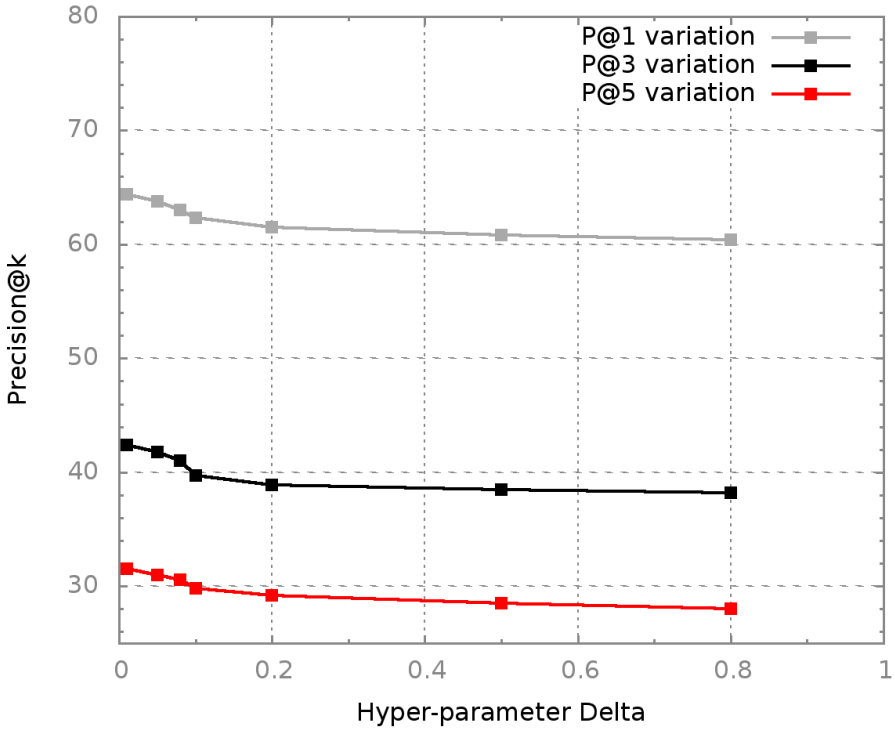


Figure: Variation of Precision@k for WikiLSHTC-325K with Δ

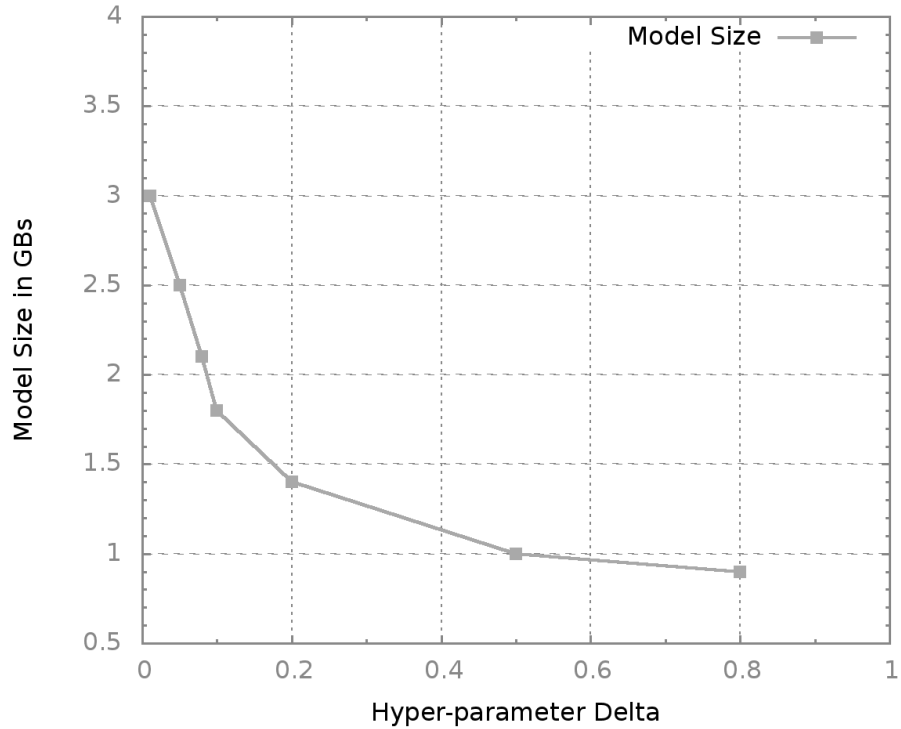


Figure: Variation of Model size for WikiLSHTC-325K with Δ

Label Embedding versus One-vs-Rest - What to choose when?

Algebraic graph theoretic view-point

- Let $A(G)$ be adjacency matrix for the label co-occurrence graph
- Let $D(G)$ be degree matrix representing degrees of labels

The Laplacian of the graph G, is given by $L(G) = D(G) - A(G)$.

Theorem

[14] *Given the training data $\mathcal{T}$, let L(G) be defined as above. Let $\lambda_1(G), \ldots, \lambda_L(G)$ be the eigen-values of $L(G)$, then :*
(a) $\lambda_\ell(G) \geq 0 \;\; \forall \ell$, and $\lambda_1(G) = 0$
(b) The multiplicity of 0 as an eigen value gives the number of connected components of G, $\implies \lambda_2(G) > 0$ if and only if G is connected.
(c) $\lambda_2(G) \leq \nu(G) \leq \eta(G)$, where $\lambda_2(G), \nu(G)$ and $\eta(G)$ are respectively the algebraic, edge and vertex connectivities.

[14] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973

Label Embedding vs One-vs-Rest - What to choose When?

Dataset	# Training	# Categories			Algebraic Connectivity, $\lambda_2(G)$	# Connected Components	Better Performance
			APpL	ALpP			
Mediamill	30993	**101**	1902.1	4.4	**0.46**	1	SLEEC
Bibtex	4880	**159**	111.7	2.4	**0.05**	1	SLEEC
Delicious-small	12,920	**983**	311.6	19.3	**0.3**	1	SLEEC
EUR-Lex	15,539	**3,993**	25.7	5.3	**0.22**	1	DiSMEC
Wikipedia-31K	14,146	**30,938**	8.5	18.6	**0.4**	1	Comparable
WikiLSHTC-325K	1,778,351	**325,056**	17.4	3.2	**0.002**	**740**	DiSMEC
Wiki-500K	1,813,391	**501,070**	24.7	4.7	**0.001**	**370**	DiSMEC
Amazon-670K	490,499	**670,091**	3.9	5.4	**0.0001**	**9,566**	DiSMEC

Table: Multi-label datasets from XMC repository. APpL and ALpP represent average points per label and average labels per point respectively

- For smaller datasets, the algebraic connectivity is much higher and label embedding based methods such as SLEEC work relatively well in this regime

Label Embedding vs One-vs-Rest - What to choose When?

Dataset	# Training	# Categories			Algebraic Connectivity, $\lambda_2(G)$	# Connected Components	Better Performance
			APpL	ALpP			
Mediamill	30993	**101**	1902.1	4.4	**0.46**	1	SLEEC
Bibtex	4880	**159**	111.7	2.4	**0.05**	1	SLEEC
Delicious-small	12,920	**983**	311.6	19.3	**0.3**	1	SLEEC
EUR-Lex	15,539	**3,993**	25.7	5.3	**0.22**	1	DiSMEC
Wikipedia-31K	14,146	**30,938**	8.5	18.6	**0.4**	1	Comparable
WikiLSHTC-325K	1,778,351	**325,056**	17.4	3.2	**0.002**	**740**	DiSMEC
Wiki-500K	1,813,391	**501,070**	24.7	4.7	**0.001**	**370**	DiSMEC
Amazon-670K	490,499	**670,091**	3.9	5.4	**0.0001**	**9,566**	DiSMEC

Table: Multi-label datasets from XMC repository. APpL and ALpP represent average points per label and average labels per point respectively

- For smaller datasets, the algebraic connectivity is much higher and label embedding based methods such as SLEEC work relatively well in this regime
- For much larger datasets, the algebraic connectivity is close to 0 and one-vs-rest approach such as DiSMEC works better in this regime

Tree-based models

Tree-based models

- Decision trees
- Label trees

Decision trees

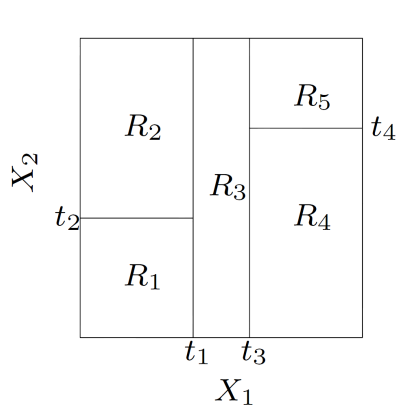
- Decision trees:

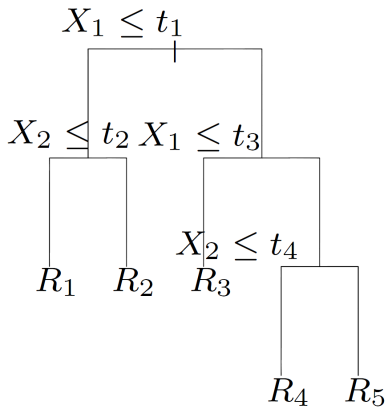
[15] Anna Choromanska and John Langford. Logarithmic time online multiclass prediction. In *NIPS 29*, 2015

[16] Yashoteja Prabhu and Manik Varma. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, pages 263–272. ACM, 2014

Decision trees

- Decision trees:
  - Partition of the feature space to small subregions:



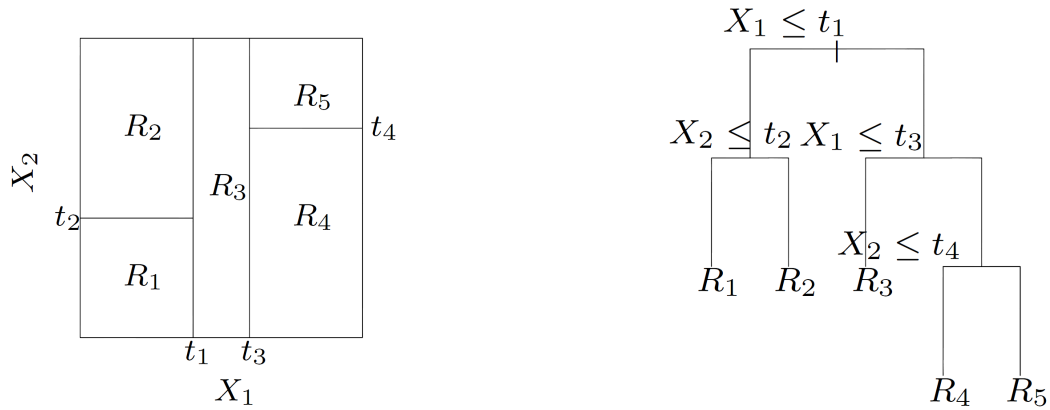


[15] Anna Choromanska and John Langford. Logarithmic time online multiclass prediction. In *NIPS 29*, 2015

[16] Yashoteja Prabhu and Manik Varma. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, pages 263–272. ACM, 2014

Decision trees

- Decision trees:
  - Partition of the feature space to small subregions:



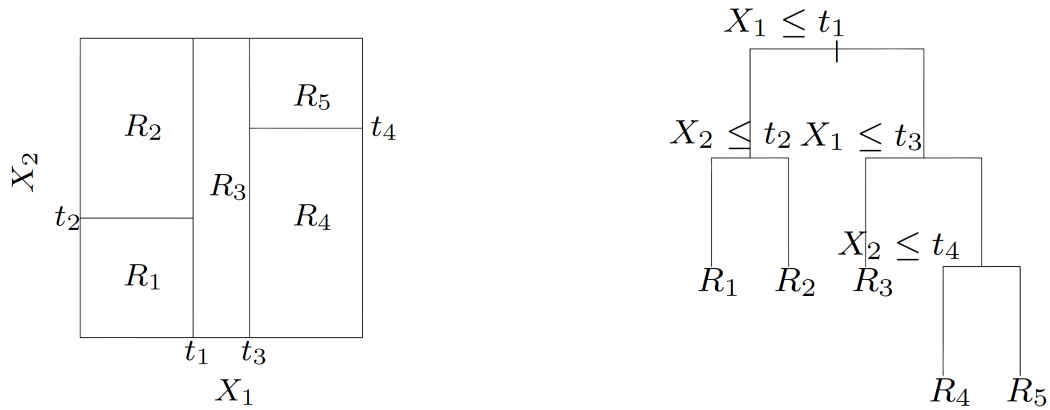
  - Fast prediction: logarithmic in n

[15] Anna Choromanska and John Langford. Logarithmic time online multiclass prediction. In *NIPS 29*, 2015

[16] Yashoteja Prabhu and Manik Varma. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, pages 263–272. ACM, 2014

Decision trees

- Decision trees:
  - Partition of the feature space to small subregions:



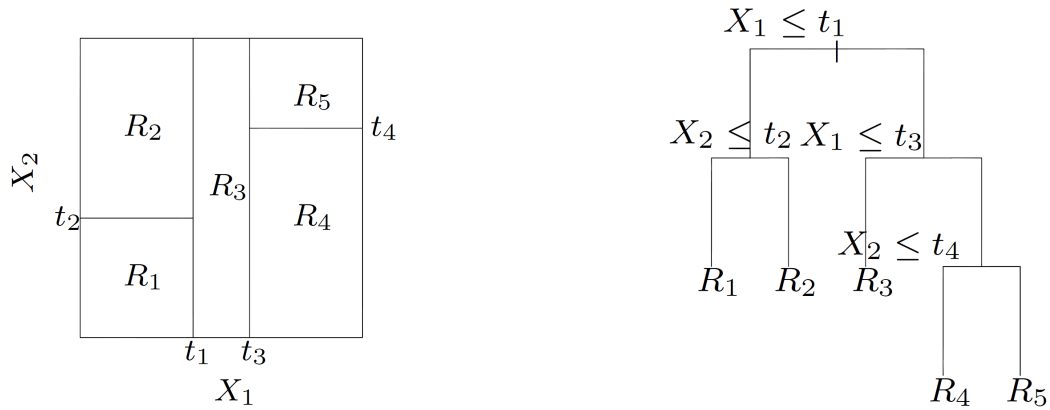
  - Fast prediction: logarithmic in n
  - Training can be expensive: computation of split criterion

[15] Anna Choromanska and John Langford. Logarithmic time online multiclass prediction. In *NIPS 29*, 2015

[16] Yashoteja Prabhu and Manik Varma. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, pages 263–272. ACM, 2014

Decision trees

- Decision trees:
  - Partition of the feature space to small subregions:



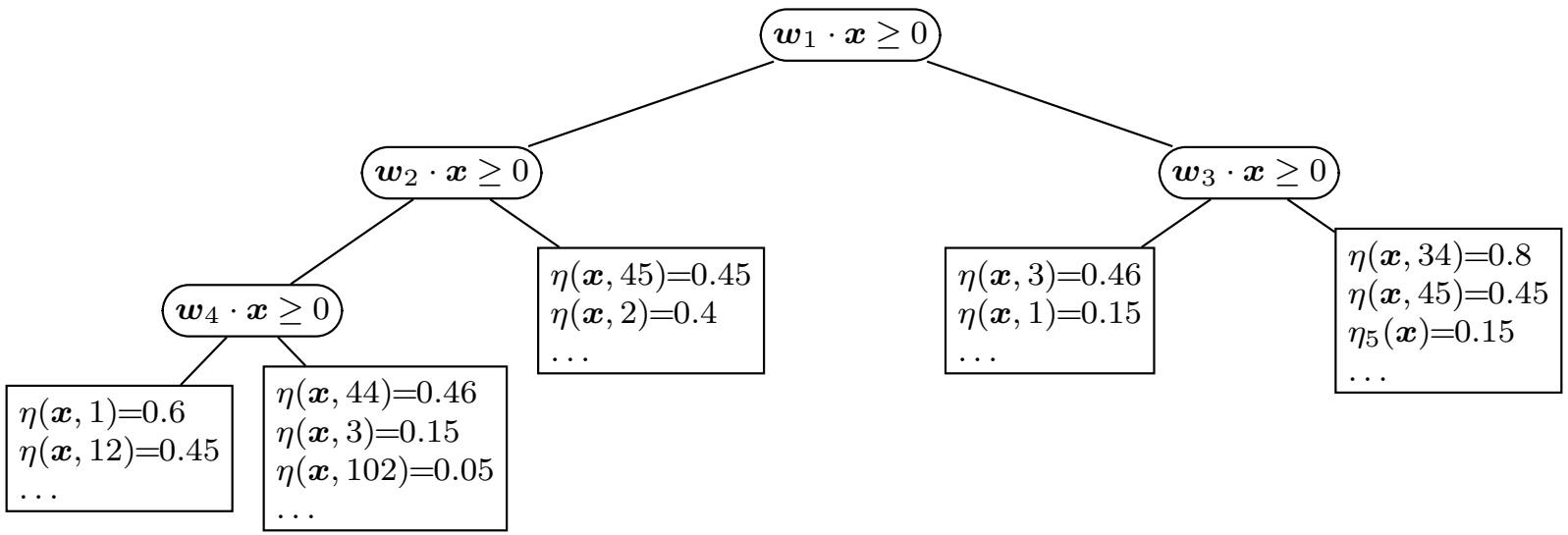
  - Fast prediction: logarithmic in n
  - Training can be expensive: computation of split criterion
  - Two new algorithms: LomTree[15] and FastXML[16]

[15] Anna Choromanska and John Langford. Logarithmic time online multiclass prediction. In *NIPS 29*, 2015

[16] Yashoteja Prabhu and Manik Varma. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, pages 263–272. ACM, 2014

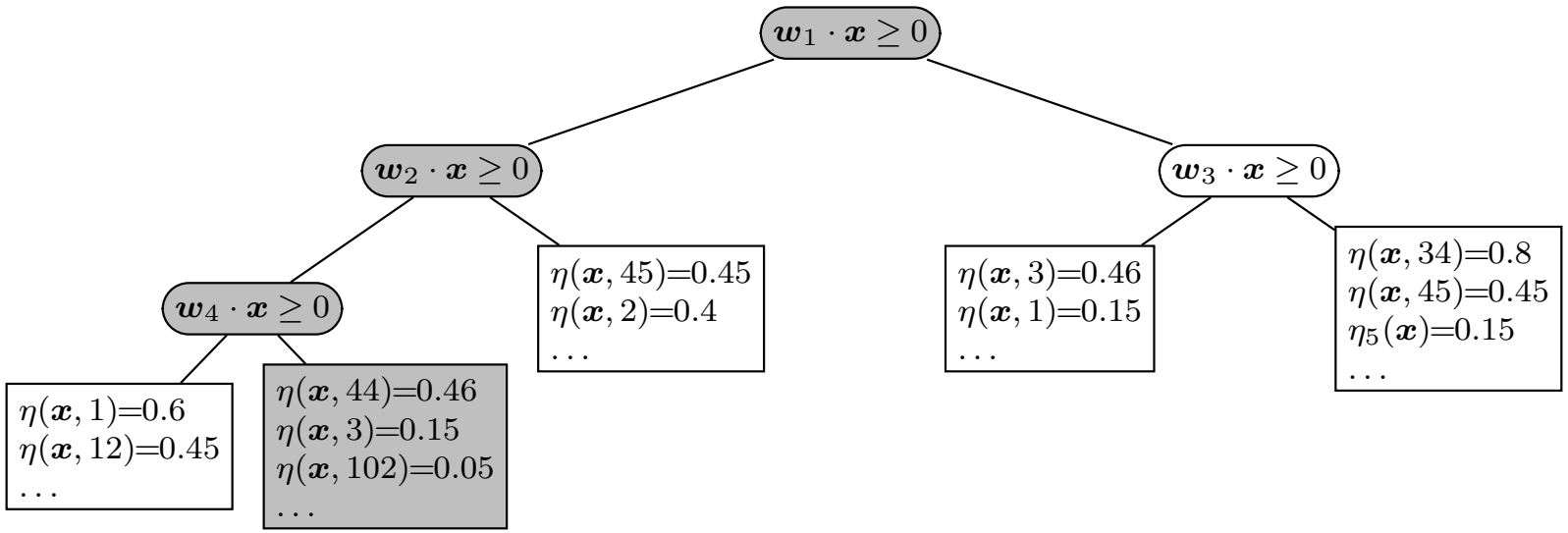
FastXML

- Uses an **ensemble** of standard decision trees
- **Sparse linear** classifiers trained in internal nodes
- Very **efficient** training procedure
- **Empirical distributions** in leaves
- A test example passes **one path** from the root to a leaf



FastXML

- Uses an **ensemble** of standard decision trees
- **Sparse linear** classifiers trained in internal nodes
- Very **efficient** training procedure
- **Empirical distributions** in leaves
- A test example passes **one path** from the root to a leaf



Optimization in FastXML

- In each internal node FastXML solves:

$$
\begin{aligned}
\min \quad & \|\boldsymbol{w}\|_1 + \sum_{i=1}^{n} C_\delta(\delta_i) \log(1 + \exp(-\delta_i \boldsymbol{w}^\top \boldsymbol{x}) \\
& -C_r \sum_{i=1}^{n} \frac{1}{2}(1 + \delta_i) \mathrm{NDCG@}m(\boldsymbol{r}^+, \boldsymbol{y}_i) \\
& -C_r \sum_{i=1}^{n} \frac{1}{2}(1 - \delta_i) \mathrm{NDCG@}m(\boldsymbol{r}^-, \boldsymbol{y}_i) \\
\text{w.r.t.} \quad & \boldsymbol{w} \in \mathbb{R}^d, \boldsymbol{\delta} \in \{-1, 1\}^n, \boldsymbol{r}^+, \boldsymbol{r}^- \in \Pi(1, m)
\end{aligned}
$$

Optimization in FastXML

- In each internal node FastXML solves:

$$\begin{aligned}
\min \quad & \|\boldsymbol{w}\|_1 + \sum_{i=1}^{n} C_\delta(\delta_i) \log(1 + \exp(-\delta_i \boldsymbol{w}^\top \boldsymbol{x}) \\
& -C_r \sum_{i=1}^{n} \frac{1}{2}(1 + \delta_i) \mathrm{NDCG@}m(\boldsymbol{r}^+, \boldsymbol{y}_i) \\
& -C_r \sum_{i=1}^{n} \frac{1}{2}(1 - \delta_i) \mathrm{NDCG@}m(\boldsymbol{r}^-, \boldsymbol{y}_i) \\
\text{w.r.t.} \quad & \boldsymbol{w} \in \mathbb{R}^d, \boldsymbol{\delta} \in \{-1, 1\}^n, \boldsymbol{r}^+, \boldsymbol{r}^- \in \Pi(1, m)
\end{aligned}$$

linear split

partitioning of training examples

label ranking in positive and negative partition

Optimization in FastXML

- In each internal node FastXML solves:

$$\begin{aligned}
\min \quad & \|\boldsymbol{w}\|_1 + \sum_{i=1}^{n} C_\delta(\delta_i) \log(1 + \exp(-\delta_i \boldsymbol{w}^\top \boldsymbol{x}) \\
& -C_r \sum_{i=1}^{n} \frac{1}{2}(1 + \delta_i) \mathrm{NDCG@}m(\boldsymbol{r}^+, \boldsymbol{y}_i) \\
& -C_r \sum_{i=1}^{n} \frac{1}{2}(1 - \delta_i) \mathrm{NDCG@}m(\boldsymbol{r}^-, \boldsymbol{y}_i) \\
\text{w.r.t.} \quad & \boldsymbol{w} \in \mathbb{R}^d, \boldsymbol{\delta} \in \{-1, 1\}^n, \boldsymbol{r}^+, \boldsymbol{r}^- \in \Pi(1, m)
\end{aligned}$$

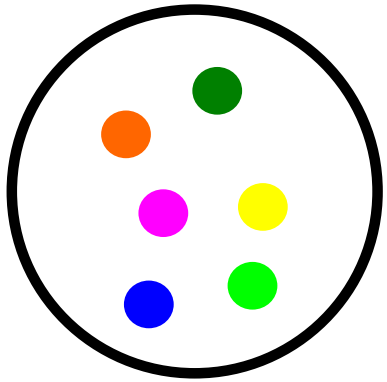
1. Bernoulli sampling of $\boldsymbol{\delta}$
2. Optimization of $\boldsymbol{r}^{\pm}$
3. Optimization of $\boldsymbol{\delta}$
4. Optimization of $\boldsymbol{w}$
5. Repeat 2-4

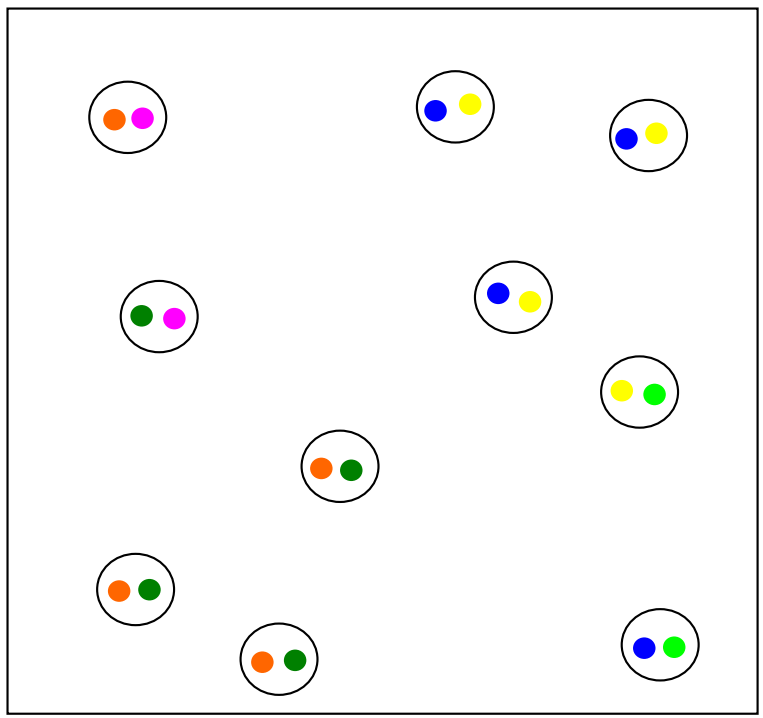
linear split

partitioning of training examples

label ranking in positive and negative partition

FastXML[17]

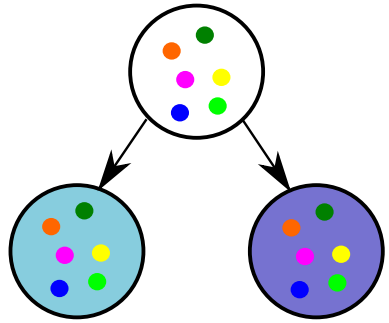


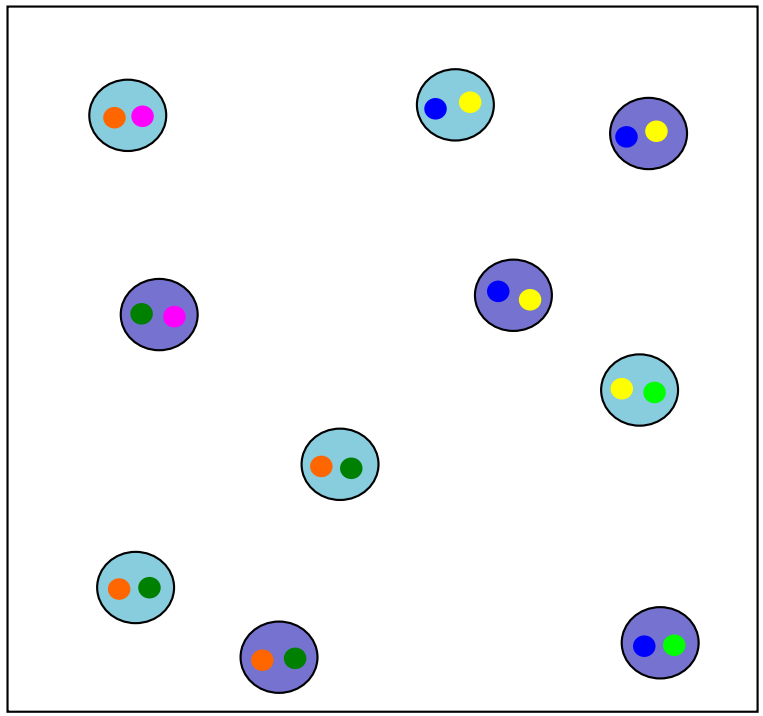


[17] https://www.youtube.com/watch?v=1X71fTx1LKA

FastXML[17]

Sampling of δ

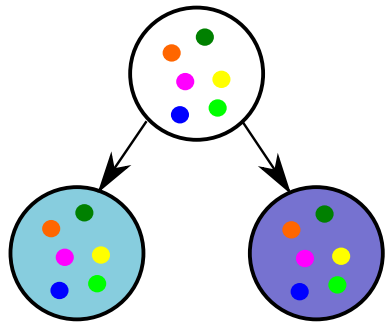


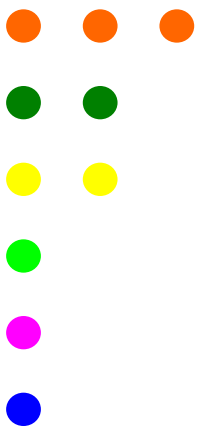


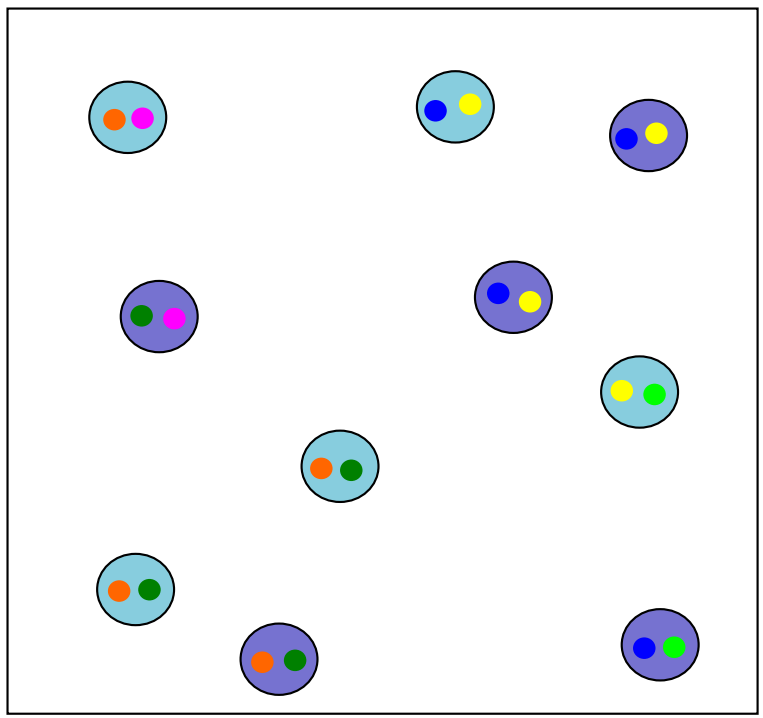
[17] https://www.youtube.com/watch?v=1X71fTx1LKA

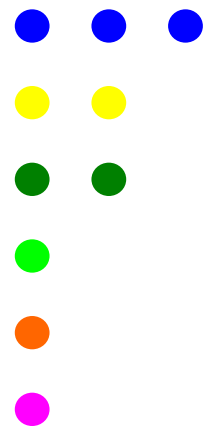
FastXML[17]

Optimization of $\boldsymbol{r}^{\pm}$





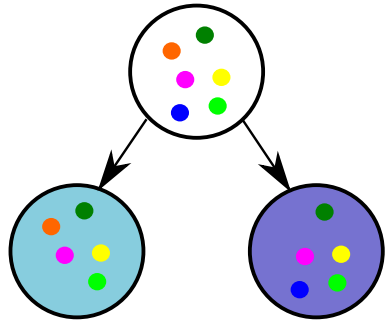


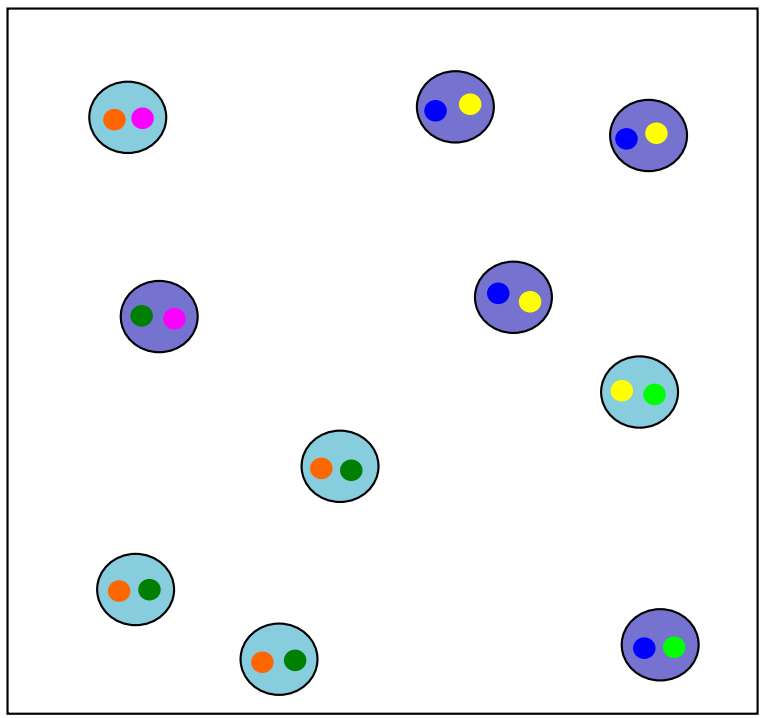


[17] https://www.youtube.com/watch?v=1X71fTx1LKA

FastXML[17]

Optimization of δ

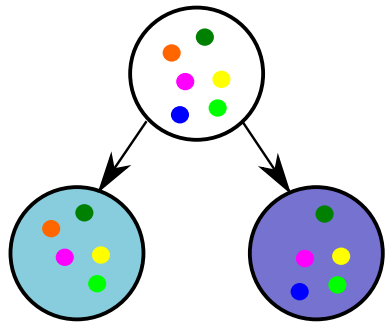


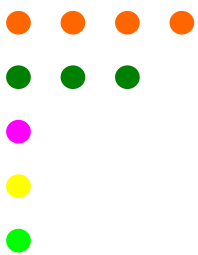


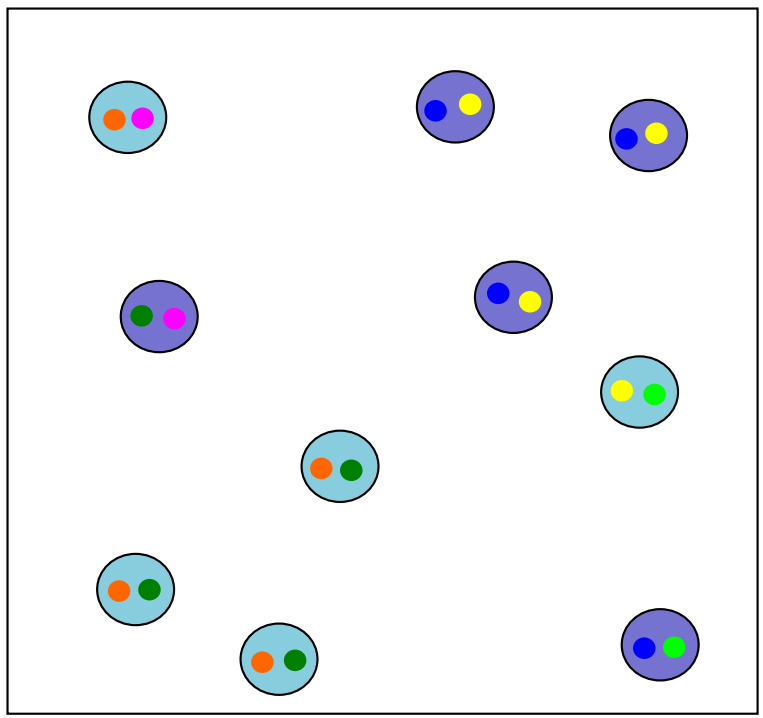
[17] https://www.youtube.com/watch?v=1X71fTx1LKA

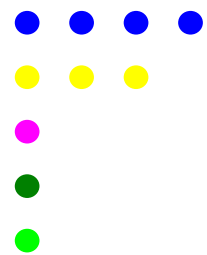
FastXML[17]

Optimization of $\boldsymbol{r}^{\pm}$





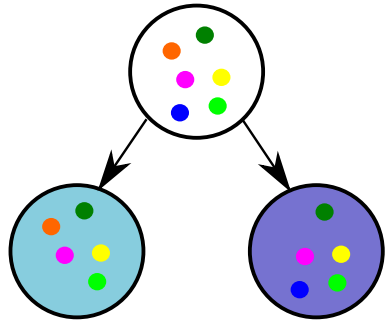


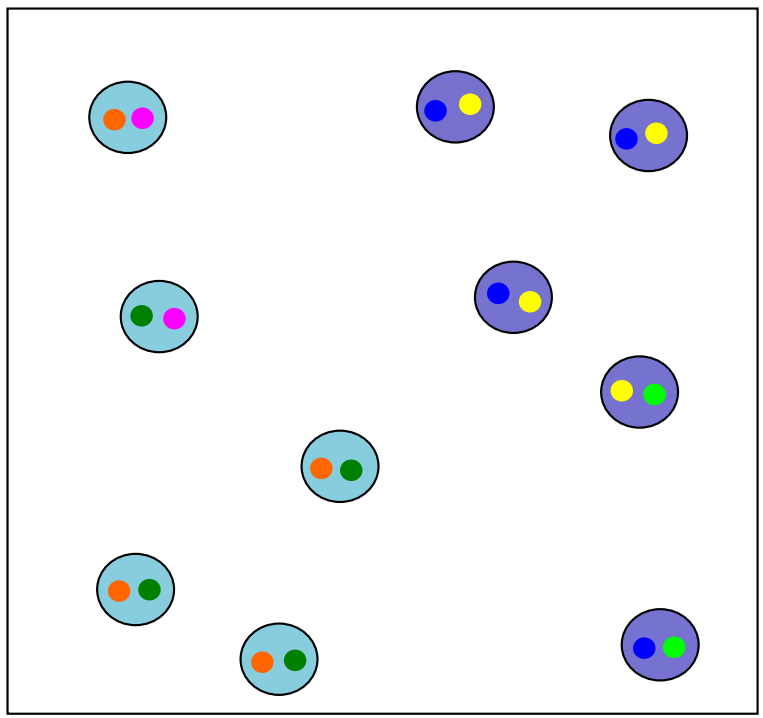


[17] https://www.youtube.com/watch?v=1X71fTx1LKA

FastXML[17]

Optimization of δ

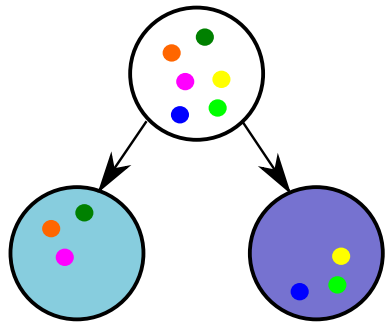


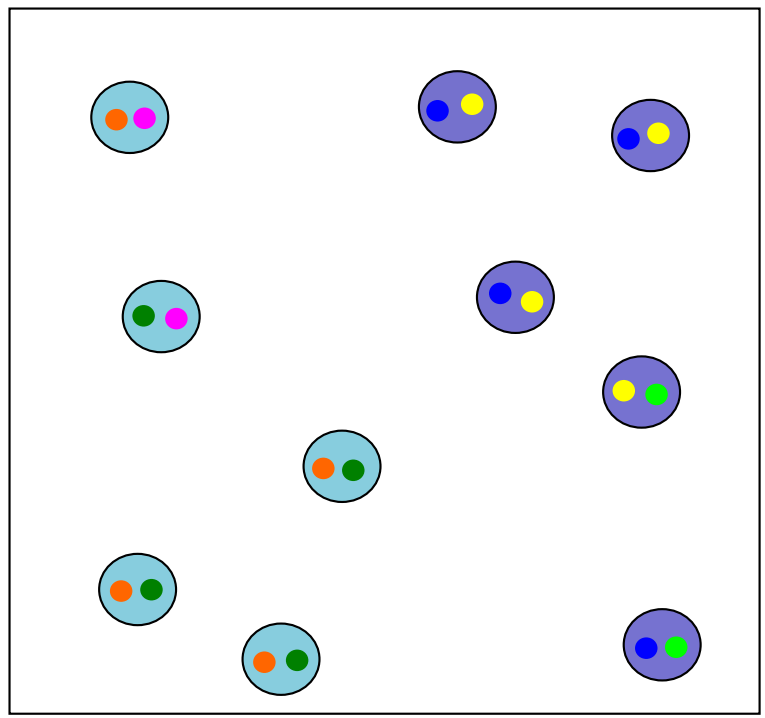


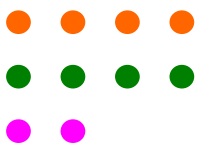
[17] https://www.youtube.com/watch?v=1X71fTx1LKA

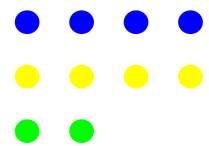
FastXML[17]

Optimization of $\boldsymbol{r}^{\pm}$





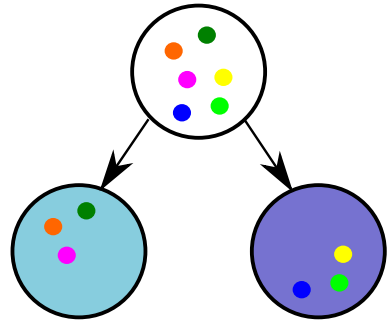


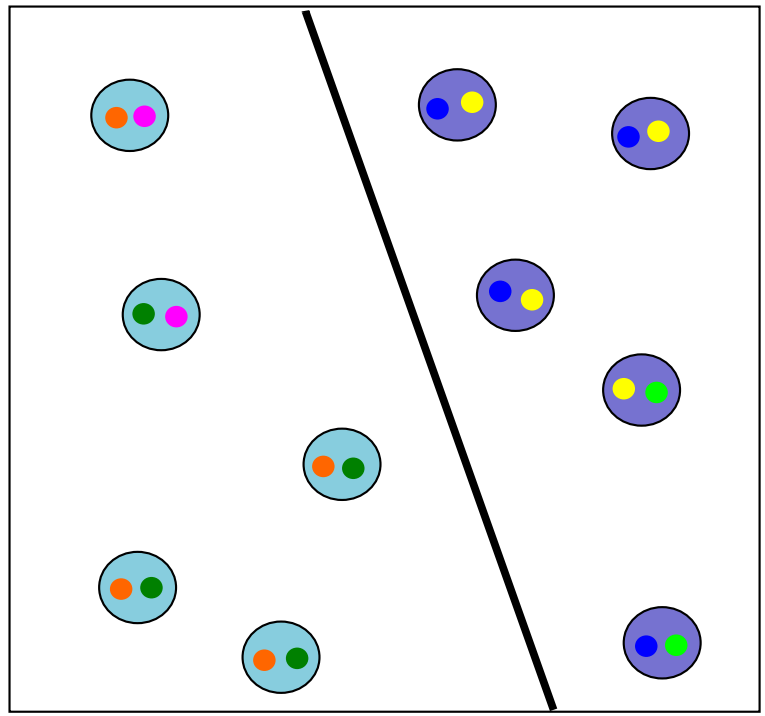


[17] https://www.youtube.com/watch?v=1X71fTx1LKA

FastXML[17]

Optimization of $\boldsymbol{w}$



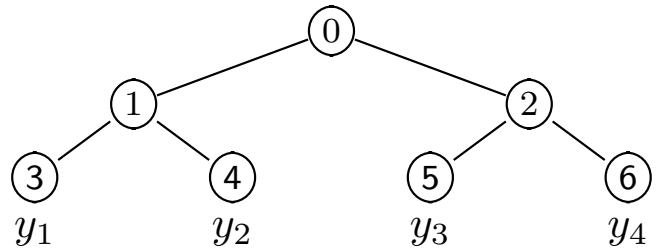


[17] https://www.youtube.com/watch?v=1X71fTx1LKA

Label trees

Label trees

- Label trees:
  - Organize classifiers in a tree structure (one leaf $\Leftrightarrow$ one label):



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

[19] J. Fox. *Applied regression analysis, linear models, and related methods*. Sage, 1997

E. Frank and S. Kramer. Ensembles of nested dichotomies for multi-class problems. In *ICML*, 2004

[20] 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

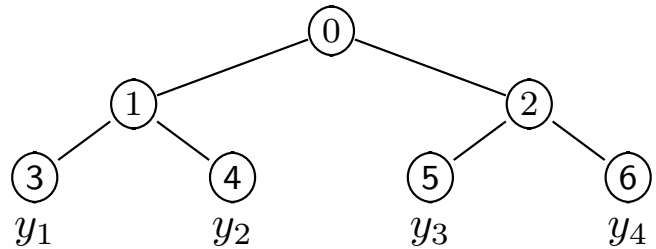
[21] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *AISTATS*, pages 246–252, 2005

[22] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016

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

Label trees

- Label trees:
  - Organize classifiers in a tree structure (one leaf $\Leftrightarrow$ one label):



  - Fast prediction: almost logarithmic in m

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

[19] J. Fox. *Applied regression analysis, linear models, and related methods*. Sage, 1997

E. Frank and S. Kramer. Ensembles of nested dichotomies for multi-class problems. In *ICML*, 2004

[20] 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

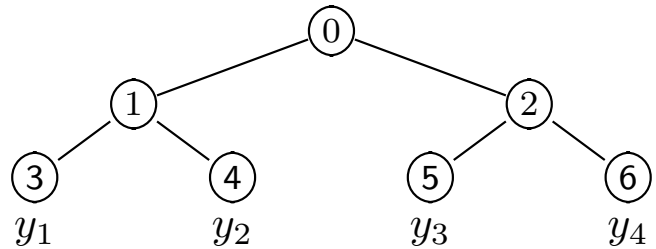
[21] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *AISTATS*, pages 246–252, 2005

[22] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016

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

Label trees

- Label trees:
  - Organize classifiers in a tree structure (one leaf $\Leftrightarrow$ one label):



  - Fast prediction: almost logarithmic in m
  - Algorithms: Label embedding trees,[18] Nested dichotomies,[19] Conditional probability trees,[20] Hierarchical softmax,[21] FastText,[22] Probabilistic classifier chains[23]

---

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

[19] J. Fox. *Applied regression analysis, linear models, and related methods*. Sage, 1997

E. Frank and S. Kramer. Ensembles of nested dichotomies for multi-class problems. In *ICML*, 2004

[20] 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

[21] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *AISTATS*, pages 246–252, 2005

[22] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016

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

Probabilistic classifier trees = Hierarchical softmax

Nested dichotomies, Hierarchical softmax, Conditional probability trees, Probabilistic classifier chains

$\Downarrow$

Probability classifier trees[24]

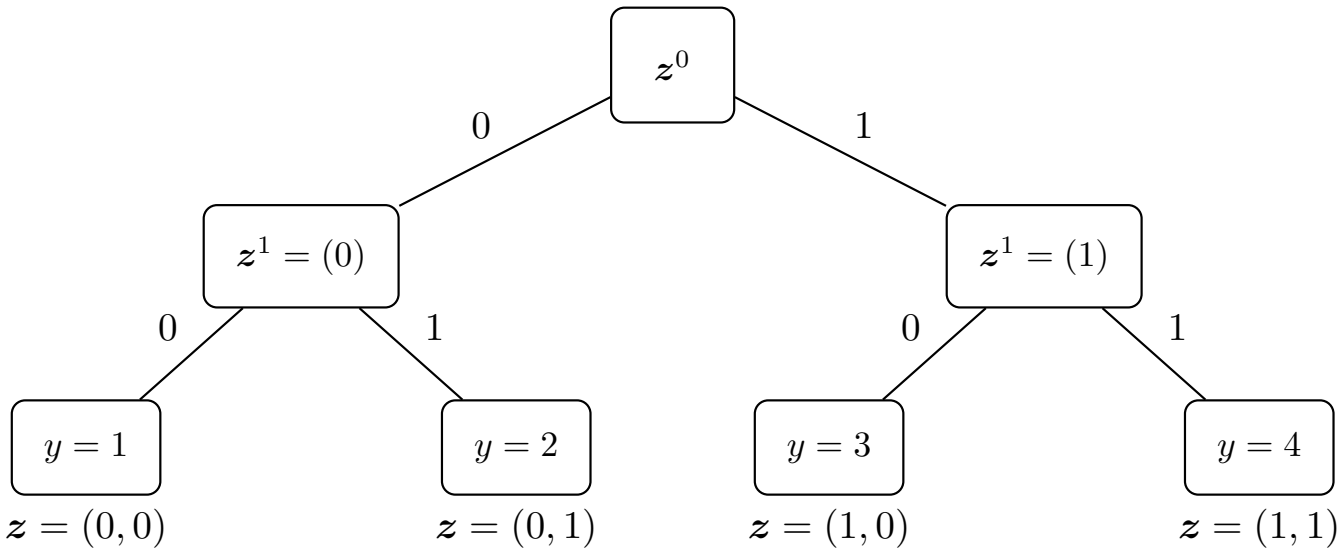
$\Downarrow$

Hierarchical softmax

[24] Krzysztof Dembczyński, Wojciech Kotłowski, Willem Waegeman, Róbert Busa-Fekete, and Eyke Hüllermeier. Consistency of probabilistic classifier trees. In *ECMLPKDD*. Springer, 2016

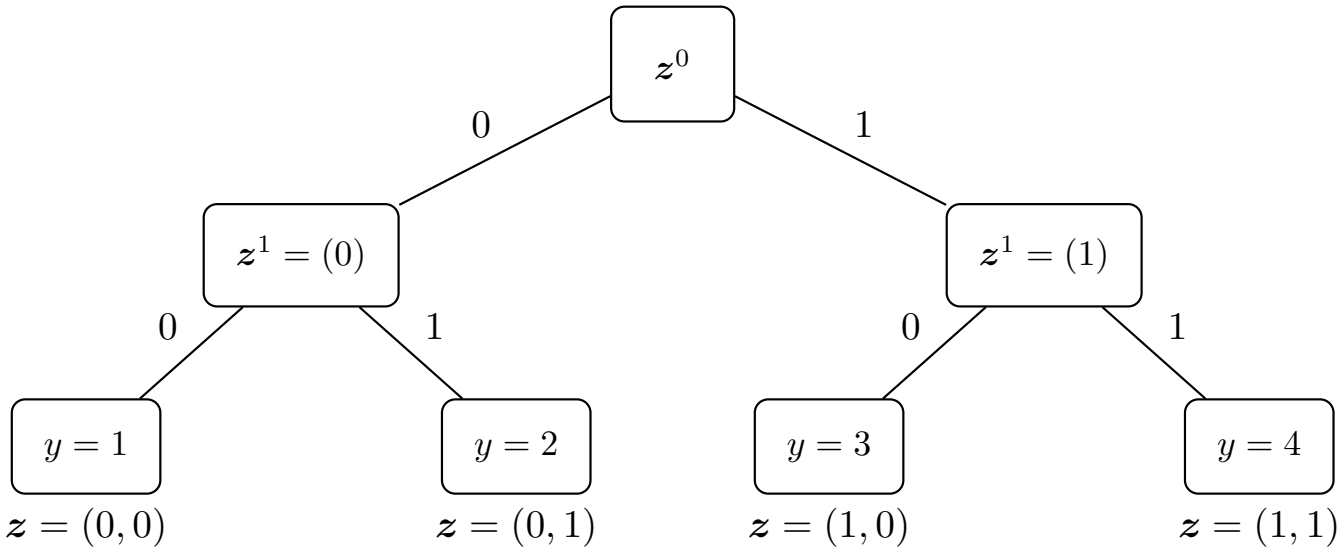
Hierarchical softmax (HSM)

- Encode the labels by a **prefix code** ($\Rightarrow$ tree structure)



Hierarchical softmax (HSM)

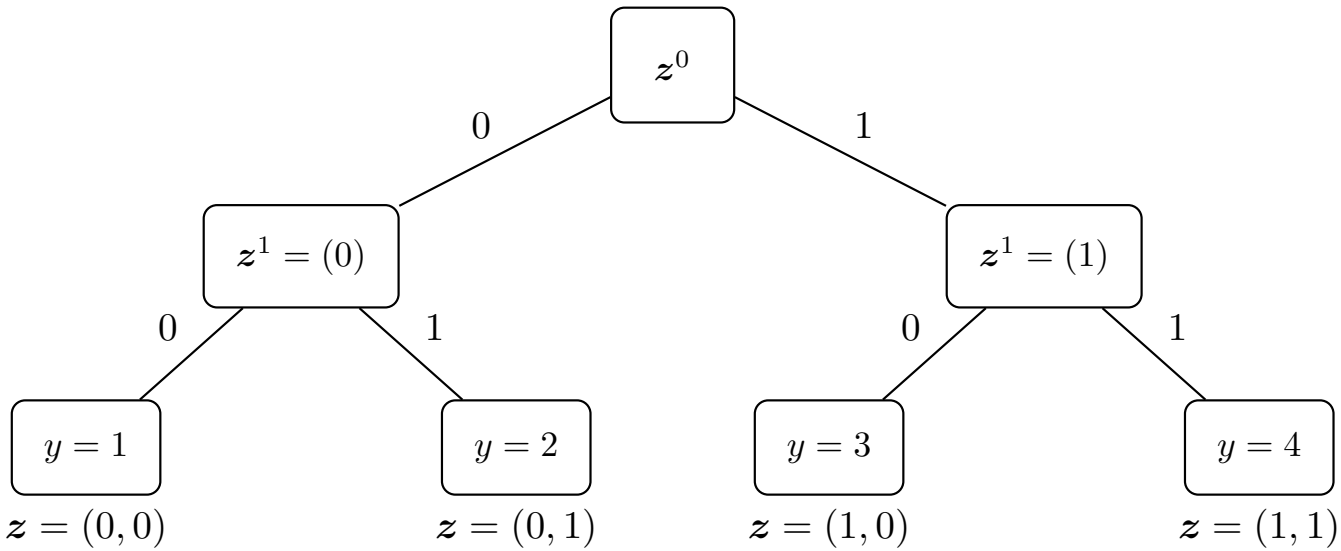
- Encode the labels by a **prefix code** ($\Rightarrow$ tree structure)



- Each label y **coded** by $\boldsymbol{z} = (z_1, \ldots, z_l) \in \mathcal{C}$

Hierarchical softmax (HSM)

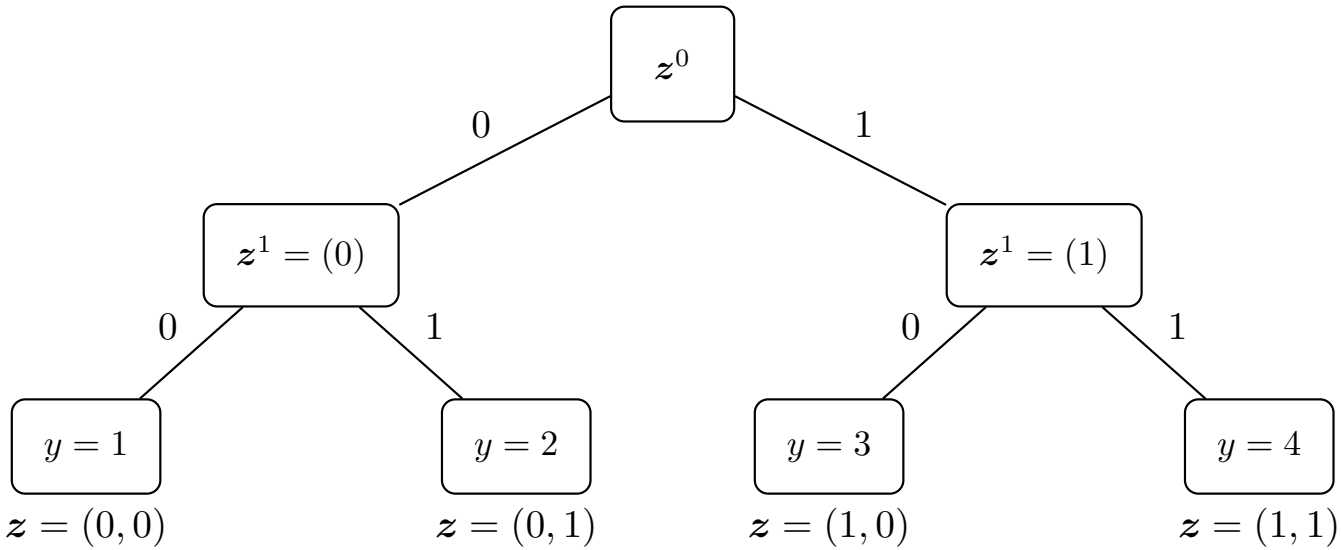
- Encode the labels by a **prefix code** ($\Rightarrow$ tree structure)



- Each label y **coded** by $\boldsymbol{z} = (z_1, \ldots, z_l) \in \mathcal{C}$
- An internal node identified by a **partial** code $\boldsymbol{z}^i = (z_1, \ldots, z_i)$

Hierarchical softmax (HSM)

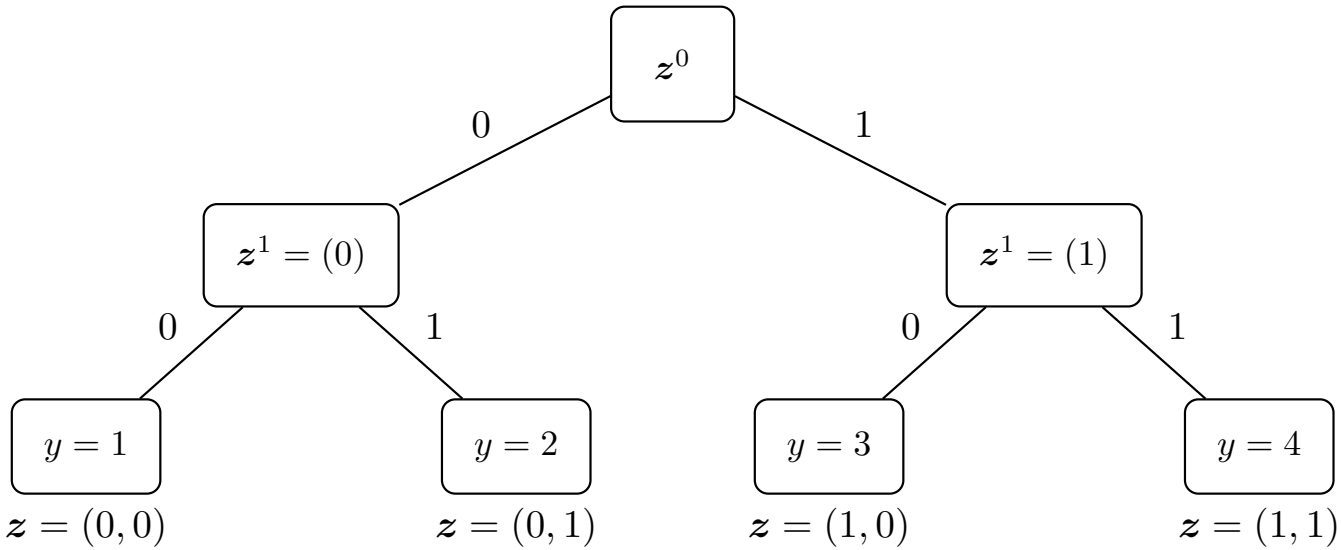
- Encode the labels by a **prefix code** ($\Rightarrow$ tree structure)



- Each label y **coded** by $\boldsymbol{z} = (z_1, \ldots, z_l) \in \mathcal{C}$
- An internal node identified by a **partial** code $\boldsymbol{z}^i = (z_1, \ldots, z_i)$
- The code does **not** have to be binary $\Rightarrow$ b-ary trees

Hierarchical softmax (HSM)

- Encode the labels by a **prefix code** ($\Rightarrow$ tree structure)

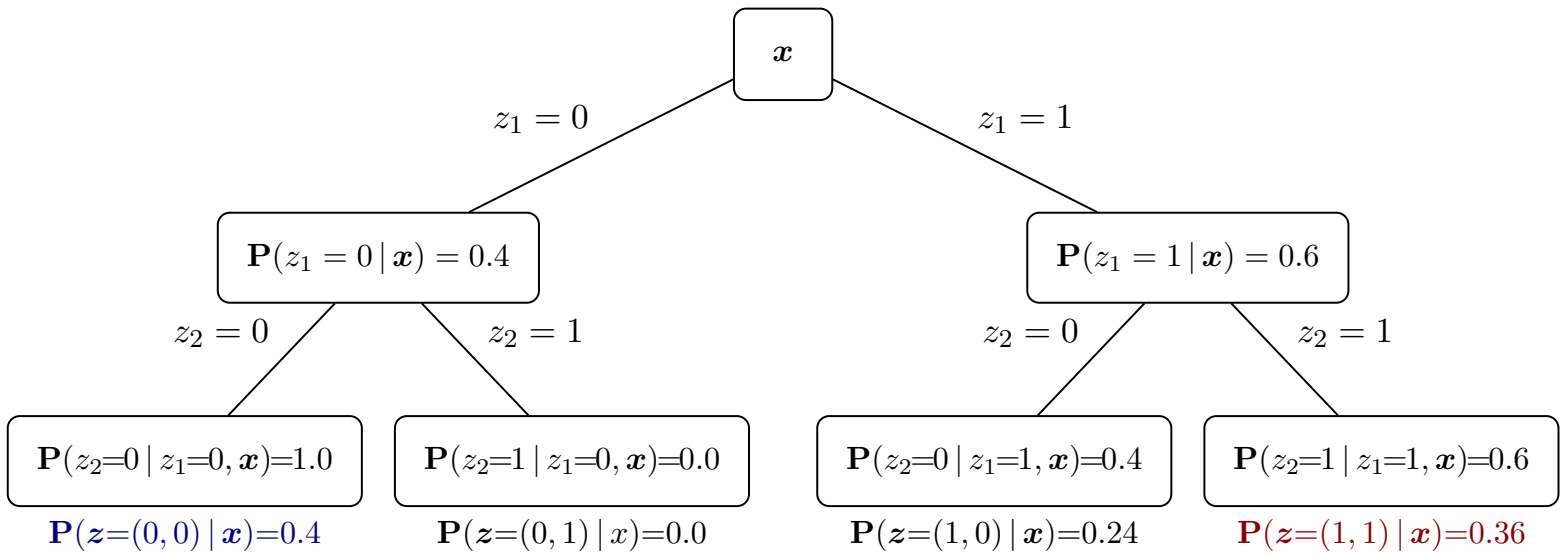


- Each label y **coded** by $\boldsymbol{z} = (z_1, \ldots, z_l) \in \mathcal{C}$
- An internal node identified by a **partial** code $\boldsymbol{z}^i = (z_1, \ldots, z_i)$
- The code does **not** have to be binary $\Rightarrow$ b-ary trees
- Different **structures** possible: random tree, Huffman tree, trained structure

Hierarchical softmax (HSM)

- HSM estimates $\mathbf{P}(y \mid \boldsymbol{x})$ by following a **path** from the root to a leaf:

$$\mathbf{P}(y \mid \boldsymbol{x}) = \mathbf{P}(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_{i=1}^{l} \mathbf{P}(z_i | \boldsymbol{z}^{i-1}, \boldsymbol{x})$$

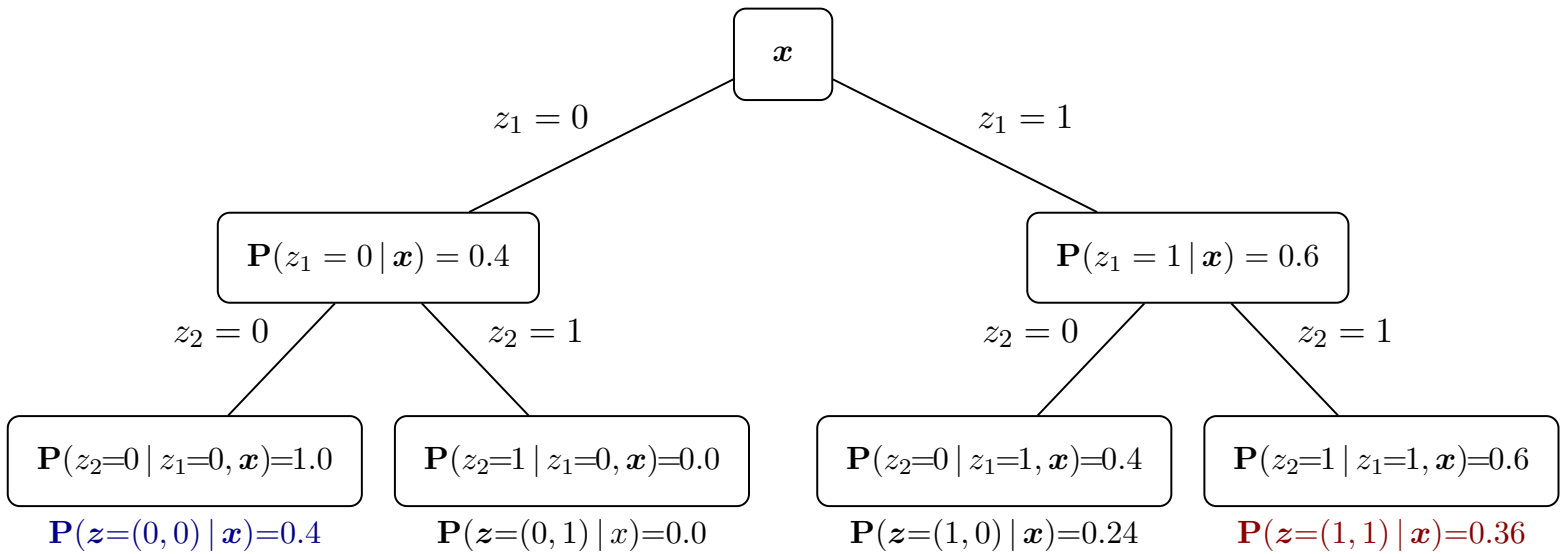


- Training: **separate** learning problems in the **internal** nodes
- Prediction: **depth first search**/**beam search**

Hierarchical softmax (HSM)

- HSM estimates $\mathbf{P}(y \mid \boldsymbol{x})$ by following a **path** from the root to a leaf:

$$\mathbf{P}(z_1 \mid \boldsymbol{x})\mathbf{P}(z_2 \mid z_1, \boldsymbol{x}) = \frac{\mathbf{P}(z_1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} \frac{\mathbf{P}(z_2, z_1, \boldsymbol{x})}{\mathbf{P}(z_1, \boldsymbol{x})} = \mathbf{P}(z_1, z_2 \mid \boldsymbol{x})$$



- Training: **separate** learning problems in the **internal** nodes
- Prediction: **depth first search**/**beam search**

HSM for MLC

- **Pick-one-label** heuristic used, for example, in FastText:

$$\eta'_j(\boldsymbol{x}) = \mathbf{P}'(y_j = 1 \,|\, \boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} y_j \frac{\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})}{\sum_{j'=1}^{m} y_{j'}}$$

HSM for MLC

- **Pick-one-label** heuristic used, for example, in FastText:

$$\eta'_j(\boldsymbol{x}) = \mathbf{P}'(y_j = 1 \,|\, \boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} y_j \frac{\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})}{\sum_{j'=1}^{m} y_{j'}}$$

- **Theorem**: inconsistent for label-wise logistic loss and precision@k

labels $\boldsymbol{y}$	probability $\mathbf{P}(\boldsymbol{y} \,\|\, \boldsymbol{x})$
$\{1\}$	0.15
$\{2\}$	0.10
$\{1, 2\}$	0.25
$\{3\}$	0.30
$\{4\}$	0.20

True marg. probs	P-o-l marg. probs
$\eta_1(\boldsymbol{x}) = 0.4$	$\eta'_3(\boldsymbol{x}) = 0.3$
$\eta_2(\boldsymbol{x}) = 0.35$	$\eta'_1(\boldsymbol{x}) = 0.275$
$\eta_3(\boldsymbol{x}) = 0.3$	$\eta'_2(\boldsymbol{x}) = 0.225$
$\eta_4(\boldsymbol{x}) = 0.2$	$\eta'_4(\boldsymbol{x}) = 0.2$

HSM for MLC

- **Pick-one-label** heuristic used, for example, in FastText:

$$\eta'_j(\boldsymbol{x}) = \mathbf{P}'(y_j = 1 \,|\, \boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} y_j \frac{\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})}{\sum_{j'=1}^{m} y_{j'}}$$

- **Theorem**: inconsistent for label-wise logistic loss and precision@k

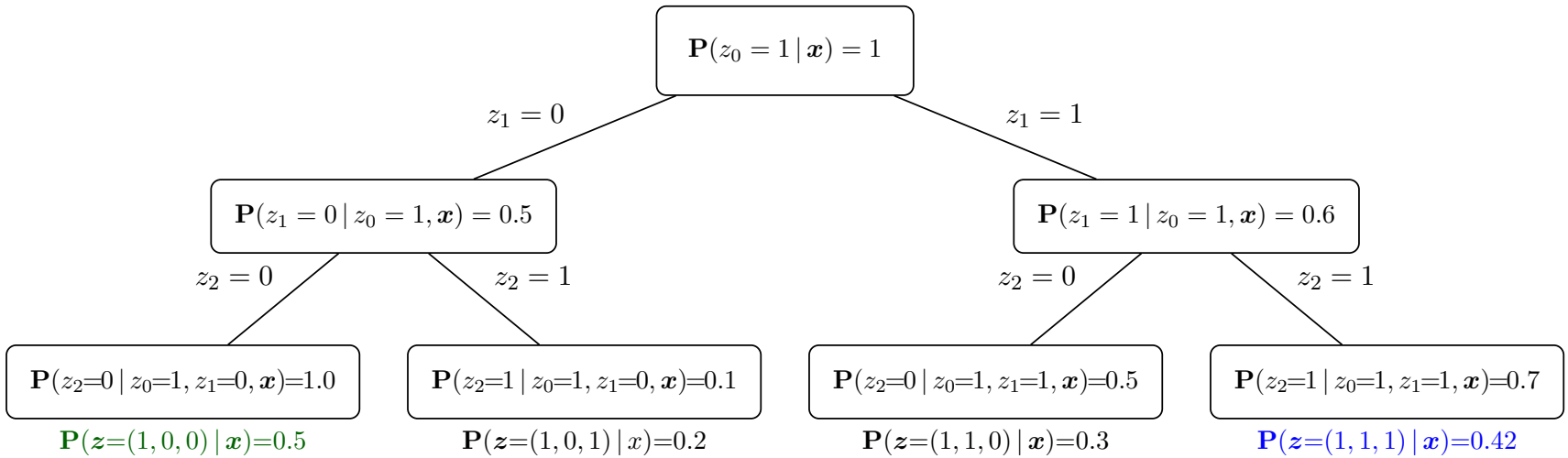
labels $\boldsymbol{y}$	probability $\mathbf{P}(\boldsymbol{y} \,\|\, \boldsymbol{x})$
$\{1\}$	0.15
$\{2\}$	0.10
$\{1,2\}$	0.25
$\{3\}$	0.30
$\{4\}$	0.20

True marg. probs	P-o-l marg. probs
$\eta_1(\boldsymbol{x}) = 0.4$	$\eta'_3(\boldsymbol{x}) = 0.3$
$\eta_2(\boldsymbol{x}) = 0.35$	$\eta'_1(\boldsymbol{x}) = 0.275$
$\eta_3(\boldsymbol{x}) = 0.3$	$\eta'_2(\boldsymbol{x}) = 0.225$
$\eta_4(\boldsymbol{x}) = 0.2$	$\eta'_4(\boldsymbol{x}) = 0.2$

- **Theorem**: consistent for precision@k for independent labels

Probabilistic label trees (PLT)[25]

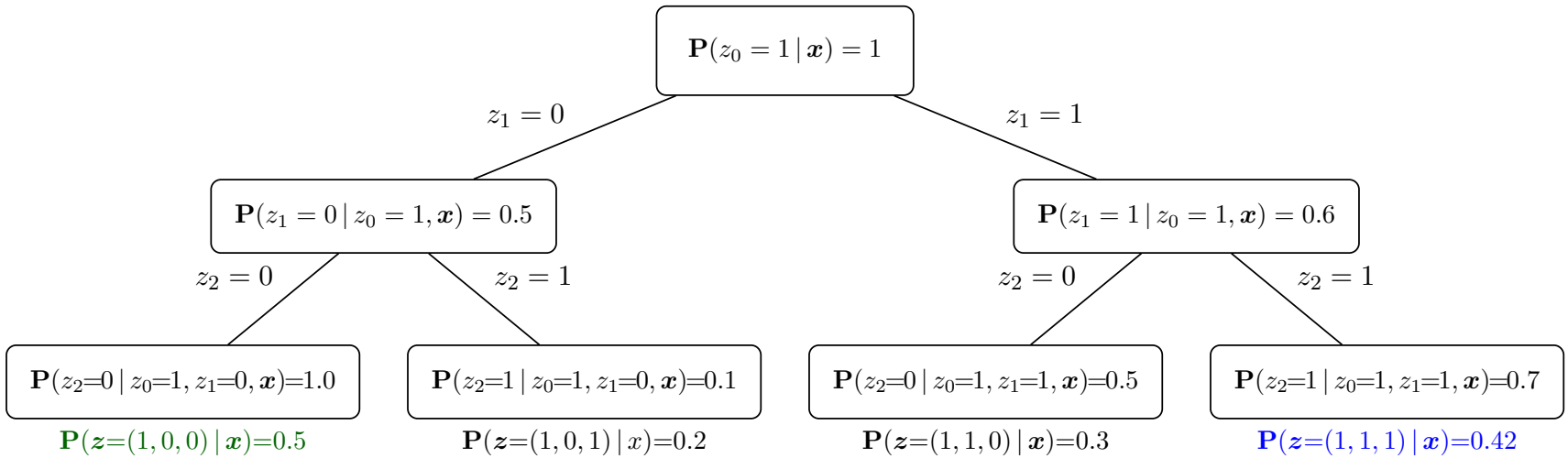
- **Similar** tree structure and encoding of $y_j = 1$ by $\boldsymbol{z} = (1, z_1, \ldots, z_l)$



[25] K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hüllermeier. Extreme F-measure maximization using sparse probability estimates. In *ICML*, 2016

Probabilistic label trees (PLT)[25]

- **Similar** tree structure and encoding of $y_j = 1$ by $\boldsymbol{z} = (1, z_1, \ldots, z_l)$



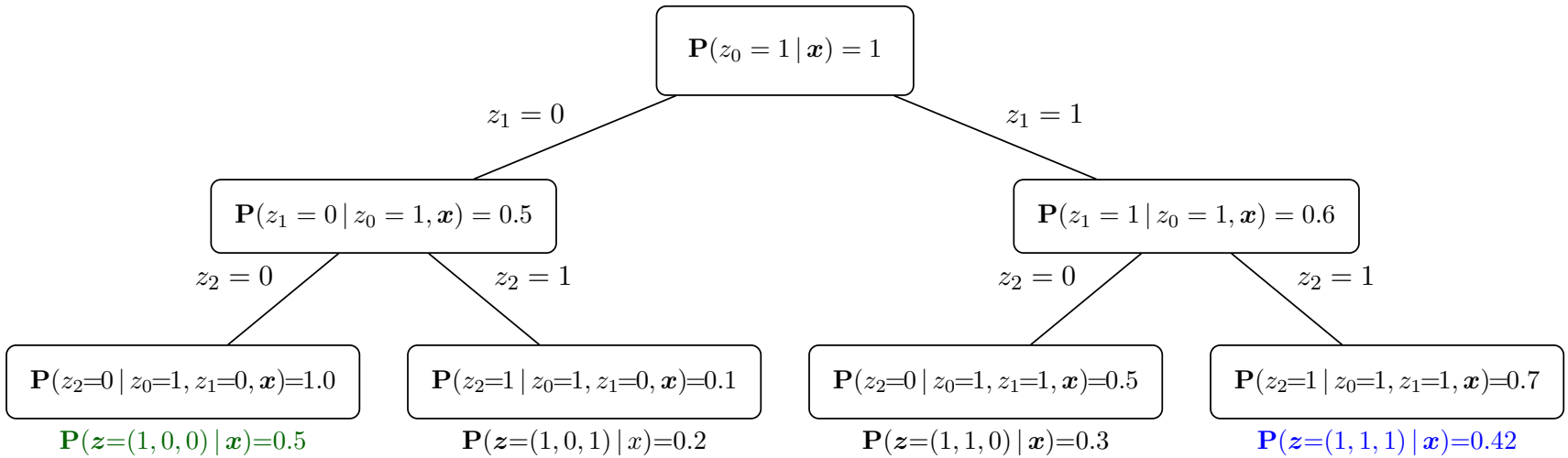
- **Marginal probabilities** $\eta_j(\boldsymbol{x})$ obtained by:

$$\eta_j(\boldsymbol{x}) = \mathbf{P}(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_{i=0}^{l} \mathbf{P}(z_i | \boldsymbol{z}^{i-1}, \boldsymbol{x})$$

[25] K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hüllermeier. Extreme F-measure maximization using sparse probability estimates. In *ICML*, 2016

Probabilistic label trees (PLT)[25]

- **Similar** tree structure and encoding of $y_j = 1$ by $\boldsymbol{z} = (1, z_1, \ldots, z_l)$



- **Marginal probabilities** $\eta_j(\boldsymbol{x})$ obtained by:

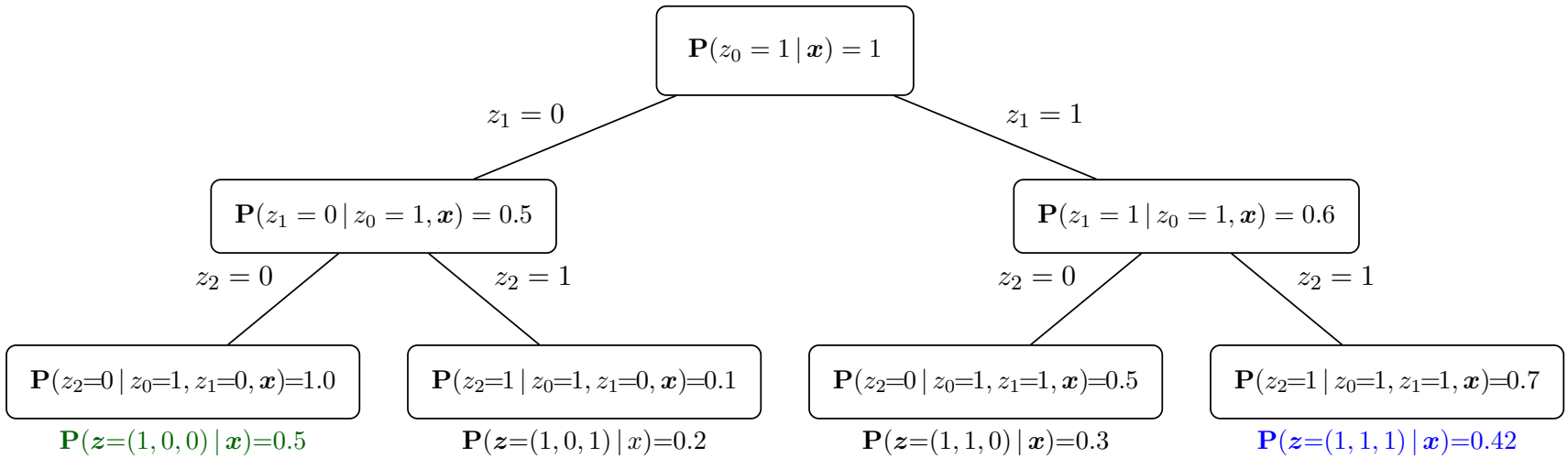
$$\eta_j(\boldsymbol{x}) = \mathbf{P}(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_{i=0}^{l} \mathbf{P}(z_i | \boldsymbol{z}^{i-1}, \boldsymbol{x})$$

- $\boldsymbol{z}^i \Leftrightarrow$ **at least one** positive label in the corresponding subtree

[25] K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hüllermeier. Extreme F-measure maximization using sparse probability estimates. In *ICML*, 2016

Probabilistic label trees (PLT)[25]

- **Similar** tree structure and encoding of $y_j = 1$ by $\boldsymbol{z} = (1, z_1, \ldots, z_l)$



- **Marginal probabilities** $\eta_j(\boldsymbol{x})$ obtained by:

$$\eta_j(\boldsymbol{x}) = \mathbf{P}(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_{i=0}^{l} \mathbf{P}(z_i | \boldsymbol{z}^{i-1}, \boldsymbol{x})$$

- $\boldsymbol{z}^i \Leftrightarrow$ **at least one** positive label in the corresponding subtree
- $\sum_{z_t} \mathbf{P}(z_t \mid \boldsymbol{z}^{t-1}) \geq 1 \Rightarrow$ separate classifiers in **all** nodes of the tree

[25] K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hüllermeier. Extreme F-measure maximization using sparse probability estimates. In *ICML*, 2016

Probabilistic label trees (PLT)

- **Training**:
  - independent training of all node classifiers
  - reduced complexity by the conditions used in the nodes
  - batch or online learning of node classifiers
  - sparse representation: small number of active features in lower nodes, feature hashing
  - dense representation: hidden representation of features (strong compression)

Probabilistic label trees (PLT)

- **Training**:
  - independent training of all node classifiers
  - reduced complexity by the conditions used in the nodes
  - batch or online learning of node classifiers
  - sparse representation: small number of active features in lower nodes, feature hashing
  - dense representation: hidden representation of features (strong compression)
- **Theoretical guarantees**
  - Regret bounds for the absolute error of marginal probability estimates and precision@k

Probabilistic label trees (PLT)

- **Training**:
  - independent training of all node classifiers
  - reduced complexity by the conditions used in the nodes
  - batch or online learning of node classifiers
  - sparse representation: small number of active features in lower nodes, feature hashing
  - dense representation: hidden representation of features (strong compression)
- **Theoretical guarantees**
  - Regret bounds for the absolute error of marginal probability estimates and precision@k
- **Tree structure**:
  - Random, complete-sorted tree, Huffman tree, trained (e.g., hierarchical clustering of labels), online training

Probabilistic label trees (PLT)

- **Training**:
  - independent training of all node classifiers
  - reduced complexity by the conditions used in the nodes
  - batch or online learning of node classifiers
  - sparse representation: small number of active features in lower nodes, feature hashing
  - dense representation: hidden representation of features (strong compression)
- **Theoretical guarantees**
  - Regret bounds for the absolute error of marginal probability estimates and precision@k
- **Tree structure**:
  - Random, complete-sorted tree, Huffman tree, trained (e.g., hierarchical clustering of labels), online training
- **Prediction**:
  - depth first search/beam search

Empirical studies

[26] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*. ACM, 2018

Empirical studies

- **HSM with pick-one-label**
  - online training
  - b-ary trees
  - Vowpal Wabbit: complete-sorted tree and feature hashing
  - FastText: Huffman tree and hidden representation of features

[26] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*. ACM, 2018

Empirical studies

- **HSM with pick-one-label**
  - online training
  - b-ary trees
  - Vowpal Wabbit: complete-sorted tree and feature hashing
  - FastText: Huffman tree and hidden representation of features
- **PLT**
  - online training
  - b-ary trees
  - Vowpal Wabbit: complete-sorted tree and feature hashing
  - FastText: Huffman tree and hidden representation of features

[26] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*. ACM, 2018

Empirical studies

- **HSM with pick-one-label**
  - online training
  - b-ary trees
  - Vowpal Wabbit: complete-sorted tree and feature hashing
  - FastText: Huffman tree and hidden representation of features
- **PLT**
  - online training
  - b-ary trees
  - Vowpal Wabbit: complete-sorted tree and feature hashing
  - FastText: Huffman tree and hidden representation of features
- **Parabel**[26]
  - PLT-like algorithm
  - Batch learning with squared hinge loss
  - Binary tree with higher arity of the last tree level
  - Tree structure obtained via 2-means++ algorithm

[26] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*. ACM, 2018

Empirical studies

- Results on the WikiLSHTC dataset
  - Stats: $m = 325056$, $d = 1617899$, $n_{\text{test}} = 587084$, $n_{\text{train}} = 1778351$
  - Results for precision@k:

Method	P@1
HSM-vw	36.90
PLT-vw	**41.63**
FastText	41.28
PLT-FastText	**41.58**
Parabel	**59.23**

Decision trees vs. Label trees

	Decision trees	Label trees
tree structure	✓	✓
structure learning	✓	✓
batch learning	✓	✓
online learning	✓	✓
number of trees	≥ 1	≥ 1
number of leaves	$O(n)$	m
internal node models	cuts/linear	linear/any model
prediction	empirical distribution	score along a path
visited paths during prediction	1	several
sparse probability estimation	✓	✓

Label filtering

Test time complexity for linear models

- Classification of a test example in case of linear models can be formulated as:

$$j^* = \underset{j \in \{1,\ldots,m\}}{\arg\max}\ \boldsymbol{w}_j^\top \boldsymbol{x}\,,$$

  i.e., as the problem of **maximum inner product search** (**MIPS**)

[27] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS '01*, pages 102–113. ACM, New York, NY, USA, 2001

Test time complexity for linear models

- Classification of a test example in case of linear models can be formulated as:

$$j^* = \underset{j \in \{1,\ldots,m\}}{\arg\max}\ \boldsymbol{w}_j^\top \boldsymbol{x}\,,$$

  i.e., as the problem of **maximum inner product search** (**MIPS**)
- Naive solution is $O(m)$

[27] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS '01*, pages 102–113. ACM, New York, NY, USA, 2001

Test time complexity for linear models

- Classification of a test example in case of linear models can be formulated as:

$$j^* = \underset{j \in \{1,\ldots,m\}}{\arg\max}\ \boldsymbol{w}_j^\top \boldsymbol{x}\,,$$

  i.e., as the problem of **maximum inner product search** (**MIPS**)
- Naive solution is $O(m)$
- Optimal exact algorithm: the threshold algorithm [27]

[27] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS '01*, pages 102–113. ACM, New York, NY, USA, 2001

Test time complexity for linear models

- Classification of a test example in case of linear models can be formulated as:

$$j^* = \underset{j \in \{1,\ldots,m\}}{\arg\max}\ \boldsymbol{w}_j^\top \boldsymbol{x}\,,$$

  i.e., as the problem of **maximum inner product search** (**MIPS**)
- Naive solution is $O(m)$
- Optimal exact algorithm: the threshold algorithm [27]
  - Does not perform well with a large number of features and labels

[27] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS '01*, pages 102–113. ACM, New York, NY, USA, 2001

Test time complexity for linear models

- Classification of a test example in case of linear models can be formulated as:

$$j^* = \underset{j \in \{1,\ldots,m\}}{\arg\max}\ \boldsymbol{w}_j^\top \boldsymbol{x}\,,$$

  i.e., as the problem of **maximum inner product search** (**MIPS**)
- Naive solution is $O(m)$
- Optimal exact algorithm: the threshold algorithm [27]
  - Does not perform well with a large number of features and labels
- There is a need for approximate solution ...

[27] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS '01*, pages 102–113. ACM, New York, NY, USA, 2001

MIPS vs. nearest neighbors

- MIPS is similar to the nearest neighbor search under the square or cosine distance:

$$j^* = \underset{j\in\{1,\ldots,m\}}{\arg\min} \|\boldsymbol{w}_j - \boldsymbol{x}\|_2^2 = \underset{j\in\{1,\ldots,m\}}{\arg\max} \boldsymbol{w}_j^\top \boldsymbol{x} - \frac{\|\boldsymbol{w}_j\|_2^2}{2}$$

$$j^* = \underset{j\in\{1,\ldots,m\}}{\arg\max} \frac{\boldsymbol{w}_j^\top \boldsymbol{x}}{\|\boldsymbol{w}_j\|\|\boldsymbol{x}\|} = \underset{j\in\{1,\ldots,m\}}{\arg\max} \frac{\boldsymbol{w}_j^\top \boldsymbol{x}}{\|\boldsymbol{w}_j\|}$$

[28] A. Shrivastava and P. Li. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (mips). In *UAI*, 2015

MIPS vs. nearest neighbors

- MIPS is similar to the nearest neighbor search under the square or cosine distance:

$$j^* = \underset{j\in\{1,\ldots,m\}}{\arg\min} \|\boldsymbol{w}_j - \boldsymbol{x}\|_2^2 = \underset{j\in\{1,\ldots,m\}}{\arg\max}\ \boldsymbol{w}_j^\top \boldsymbol{x} - \frac{\|\boldsymbol{w}_j\|_2^2}{2}$$

$$j^* = \underset{j\in\{1,\ldots,m\}}{\arg\max} \frac{\boldsymbol{w}_j^\top \boldsymbol{x}}{\|\boldsymbol{w}_j\|\|\boldsymbol{x}\|} = \underset{j\in\{1,\ldots,m\}}{\arg\max} \frac{\boldsymbol{w}_j^\top \boldsymbol{x}}{\|\boldsymbol{w}_j\|}$$

- Some tricks needs to be used to treat MIPS as nearest neighbor search[28]

[28] A. Shrivastava and P. Li. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (mips). In *UAI*, 2015

Nearest neighbors

- A well-studied problem with many existing algorithms:

[29] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977

[30] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM

[31] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734, 2017

[32] Alex Auvolat, Hugo Larochelle, Sarath Chandar, Pascal Vincent, and Yoshua Bengio. Clustering is efficient for approximate maximum inner product search. *CoRR*, abs/1507.05910, 2015

Nearest neighbors

- A well-studied problem with many existing algorithms:
  - For low-dimensional problems efficient tree-based structures exist[29]

[29] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977

[30] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM

[31] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734, 2017

[32] Alex Auvolat, Hugo Larochelle, Sarath Chandar, Pascal Vincent, and Yoshua Bengio. Clustering is efficient for approximate maximum inner product search. *CoRR*, abs/1507.05910, 2015

Nearest neighbors

- A well-studied problem with many existing algorithms:
  - For low-dimensional problems efficient tree-based structures exist[29]
  - Approximate nearest neighbor search via locality-sensitive hashing[30]

[29] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977

[30] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM

[31] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734, 2017

[32] Alex Auvolat, Hugo Larochelle, Sarath Chandar, Pascal Vincent, and Yoshua Bengio. Clustering is efficient for approximate maximum inner product search. *CoRR*, abs/1507.05910, 2015

Nearest neighbors

- A well-studied problem with many existing algorithms:
  - For low-dimensional problems efficient tree-based structures exist[29]
  - Approximate nearest neighbor search via locality-sensitive hashing[30]
  - Fast search via k-means[31] or hierarchical k-means[32]

[29] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977

[30] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM

[31] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734, 2017

[32] Alex Auvolat, Hugo Larochelle, Sarath Chandar, Pascal Vincent, and Yoshua Bengio. Clustering is efficient for approximate maximum inner product search. *CoRR*, abs/1507.05910, 2015

Agenda

1. Extreme classification: applications and challenges
2. Algorithms
3. **Live demonstration**

Live demonstration

Useful Links

Extreme Classification Repository [33]

- DataSets
  - Around a dozen datasets from Wikipedia, Amazon, Delicious
  - Datasets in raw and tf-idf format

[33] http://manikvarma.org/downloads/XC/XMLRepository.html

Useful Links

Extreme Classification Repository [33]

- DataSets
  - Around a dozen datasets from Wikipedia, Amazon, Delicious
  - Datasets in raw and tf-idf format
- Code
  - Around a dozen open source codes
  - State-of-the-art embedding, tree and linear methods

[33] http://manikvarma.org/downloads/XC/XMLRepository.html

Useful Links

Extreme Classification Repository [33]

- DataSets
  - Around a dozen datasets from Wikipedia, Amazon, Delicious
  - Datasets in raw and tf-idf format
- Code
  - Around a dozen open source codes
  - State-of-the-art embedding, tree and linear methods
- Comparisons
  - Comparison among state-of-the-art methods on various datasets
  - On Precision@k and nDCG@k and their propensity scored variants

[33] http://manikvarma.org/downloads/XC/XMLRepository.html

Useful Links

Extreme Classification Repository [33]

- DataSets
  - Around a dozen datasets from Wikipedia, Amazon, Delicious
  - Datasets in raw and tf-idf format
- Code
  - Around a dozen open source codes
  - State-of-the-art embedding, tree and linear methods
- Comparisons
  - Comparison among state-of-the-art methods on various datasets
  - On Precision@k and nDCG@k and their propensity scored variants
- Papers
  - Links to most of the recent papers in extreme classification

[33] http://manikvarma.org/downloads/XC/XMLRepository.html

Take-away message

Take-away message

- **Extreme classification**: #examples, #features, **#labels**

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational and stastistical challenges**

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational and stastistical challenges**
- **Different learning paradigms**:

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational and stastistical challenges**
- **Different learning paradigms**:
  - label embeddings,

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational and stastistical challenges**
- **Different learning paradigms**:
  - label embeddings,
  - smart 1-vs-All approaches,

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational and stastistical challenges**
- **Different learning paradigms**:
  - label embeddings,
  - smart 1-vs-All approaches,
  - tree-based methods,

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational and stastistical challenges**
- **Different learning paradigms**:
  - label embeddings,
  - smart 1-vs-All approaches,
  - tree-based methods,
  - label filtering/maximum inner product search.

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational and stastistical challenges**
- **Different learning paradigms**:
  - label embeddings,
  - smart 1-vs-All approaches,
  - tree-based methods,
  - label filtering/maximum inner product search.

`http://www.cs.put.poznan.pl/kdembczynski/xmlc-tutorial-ecir-2018/`

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational and stastistical challenges**
- **Different learning paradigms**:
  - label embeddings,
  - smart 1-vs-All approaches,
  - tree-based methods,
  - label filtering/maximum inner product search.

`http://www.cs.put.poznan.pl/kdembczynski/xmlc-tutorial-ecir-2018/`

Acknowledgments: Kalina Jasinska, Marek Wydmuch, Robert Busa-Fekete, Eyke Hüllermeier, Bernhard Schölkopf