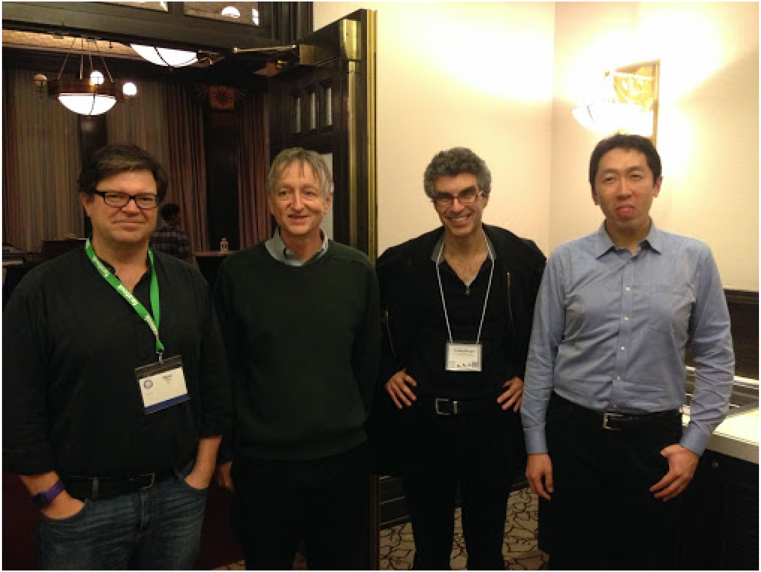
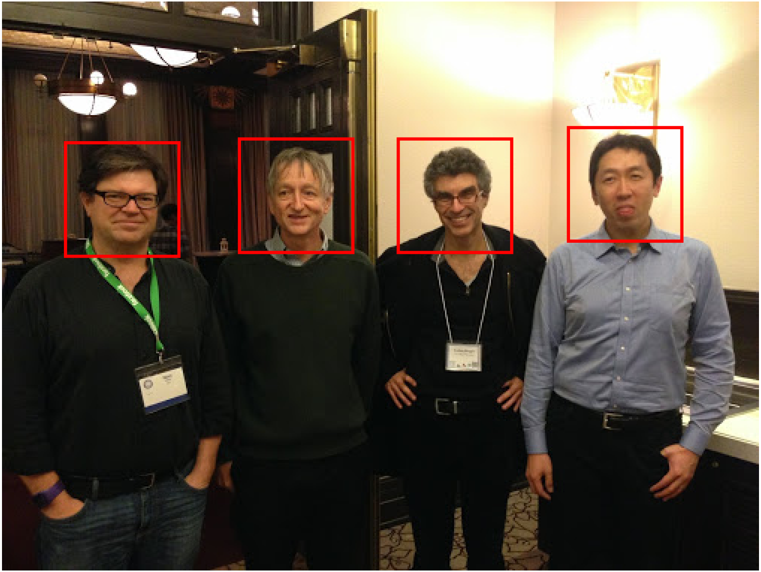
Learning with a large number of labels

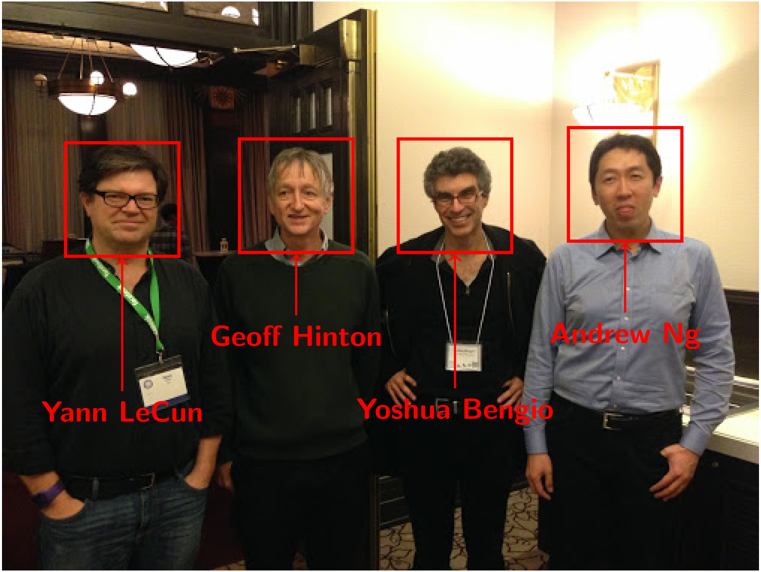
Krzysztof Dembczyński

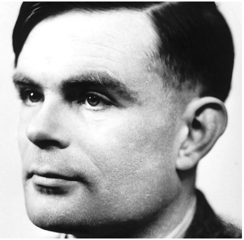
Intelligent Decision Support Systems Laboratory (IDSS)
Poznań University of Technology, Poland

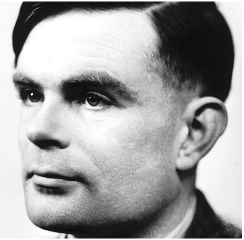
Pre-doc Summer School on Learning Systems
Monday, July 3, 2017
ETH Zürich, Switzerland






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

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
- **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
- **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
- **Computational complexity**:
  - time vs. space
  - #examples vs. #features vs. #labels

Extreme classification

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

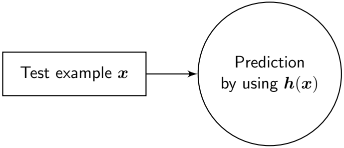
- **Predictive performance**:
  - Performance measures: Hamming loss, prec@k, NDCG@k, Macro F
- **Computational complexity**:
  - time vs. space
  - #examples vs. #features vs. #labels
  - training vs. validation vs. prediction

Supervised learning

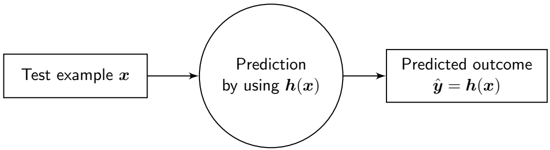
Supervised learning

Test example x

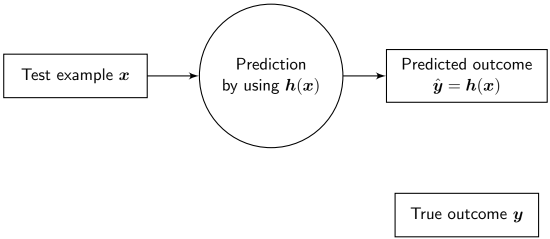
Supervised learning



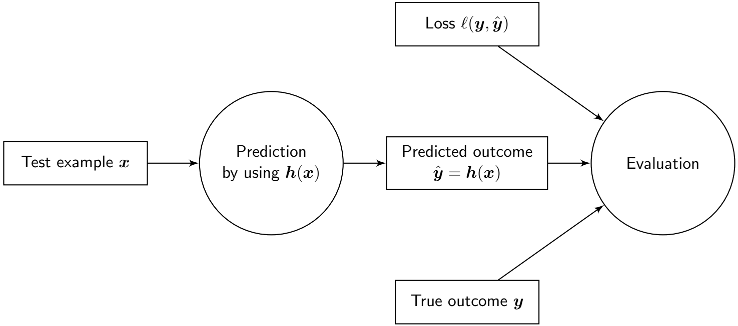
Supervised learning



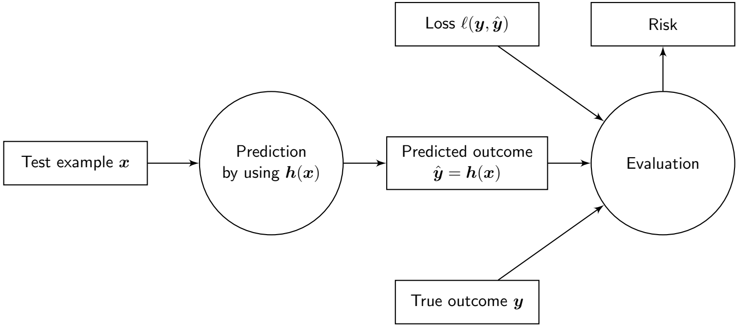
Supervised learning



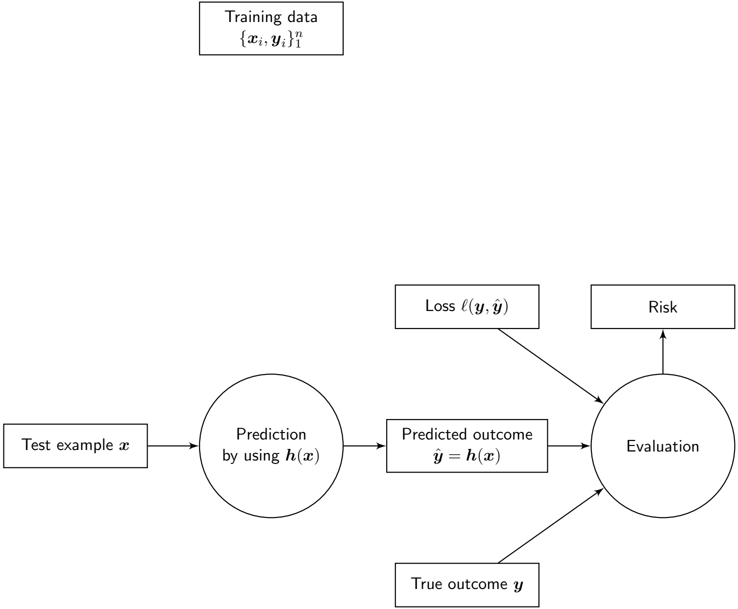
Supervised learning



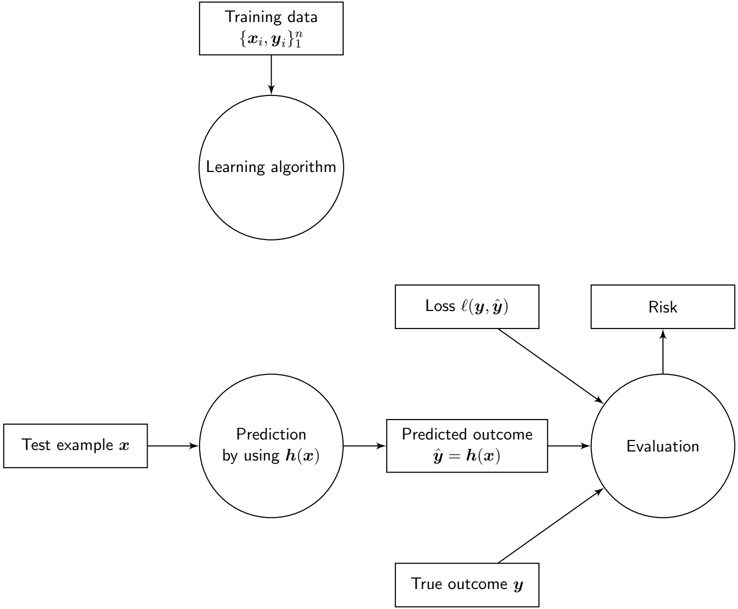
Supervised learning



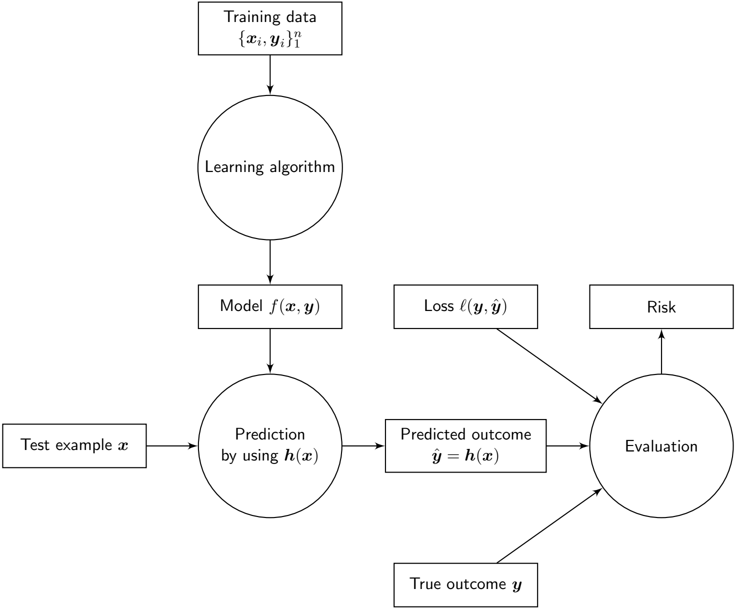
Supervised learning



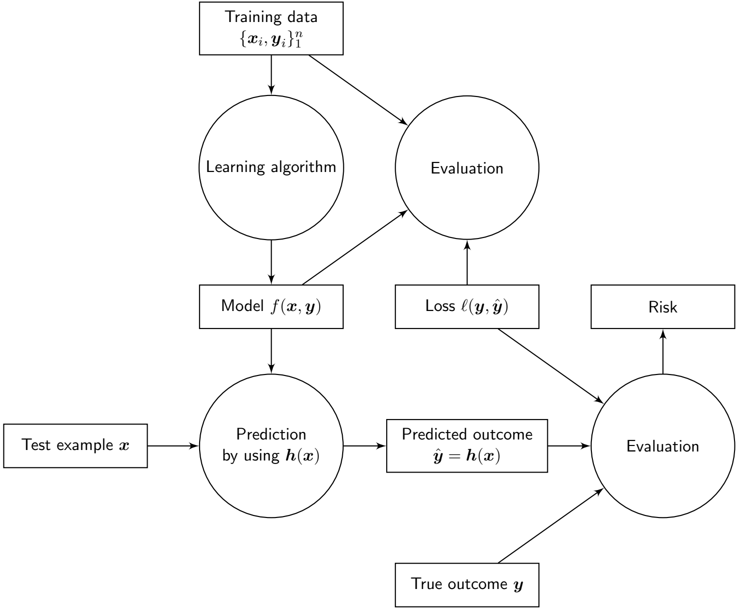
Supervised learning



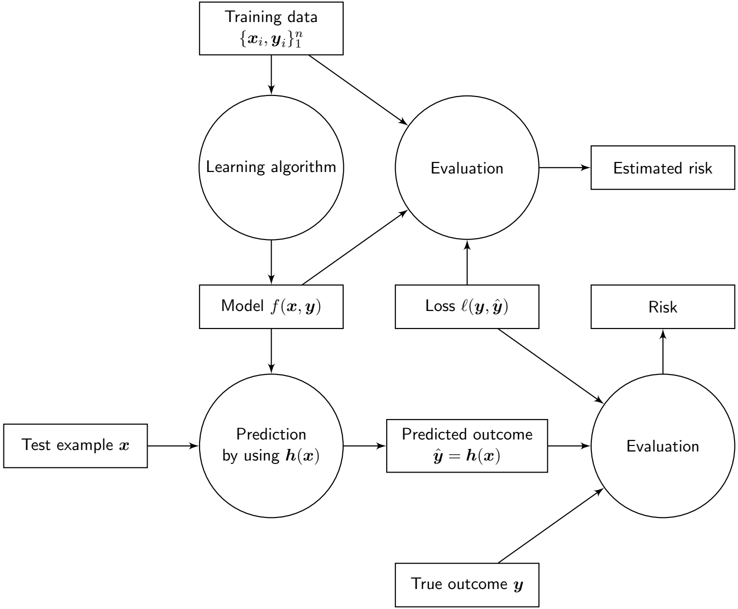
Supervised learning



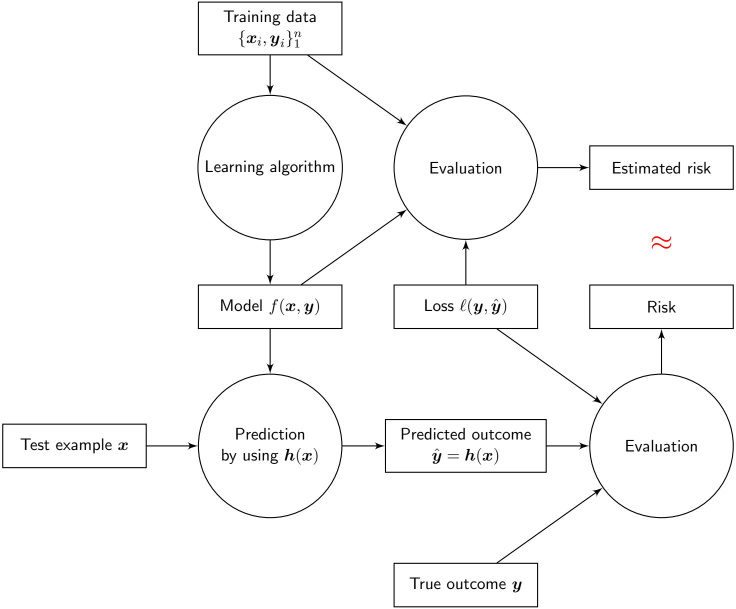
Supervised learning



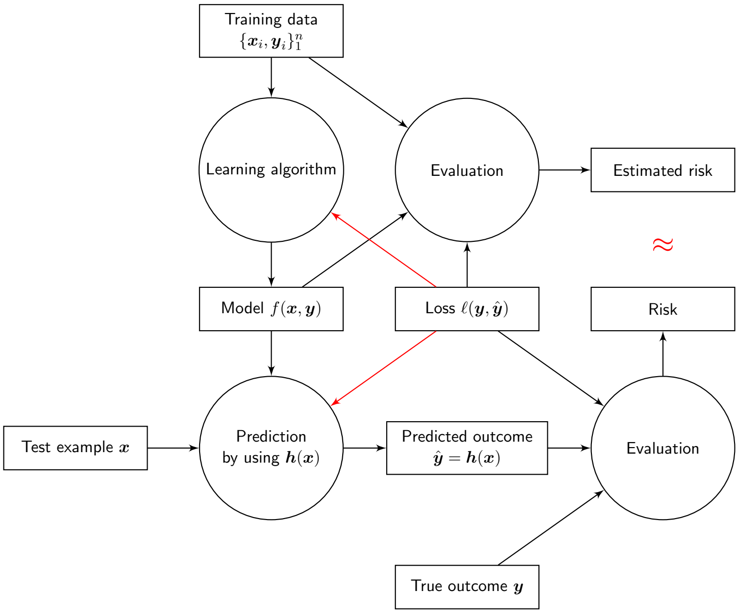
Supervised learning



Supervised learning



Supervised learning



Statistical learning framework

Statistical learning framework

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

Statistical learning framework

- **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 $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$.

Statistical learning framework

- **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 $\boldsymbol{x}$.

Statistical learning framework

- **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 $\boldsymbol{x}$.
- **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**.

Statistical learning framework

- **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 $\boldsymbol{x}$.
- **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.

Risk

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

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

Risk

- **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**:

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

[1] 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**.

[1] 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**:[1]

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

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

[1] 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

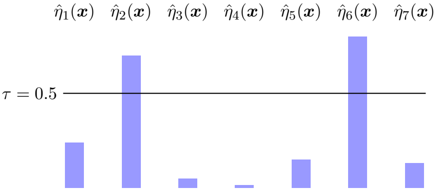
- **Hamming loss**:

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

- Sparse labels ⇒ Hamming loss of an **all-zero** classifier close to **0**.
- **The optimal strategy**:[1]

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

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



[1] 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**:

$$\mathrm{prec@}k(\boldsymbol{y}, f, \boldsymbol{x}) = \frac{1}{k}\sum_{r=1}^{k} [\![y_{\sigma(r)} = 1]\!] \,,$$

where σ is a permutation of labels for $\boldsymbol{x}$ returned by ranker f.

Precision

- **Precision at position k**:

$$\mathrm{prec@}k(\boldsymbol{y}, f, \boldsymbol{x}) = \frac{1}{k} \sum_{r=1}^{k} [\![y_{\sigma(r)} = 1]\!] \,,$$

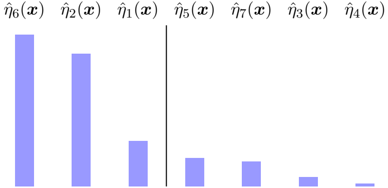
  where σ is a permutation of labels for $\boldsymbol{x}$ returned by ranker f.
- **The optimal strategy**: select top k labels according to $\eta_j(\boldsymbol{x})$.

Precision

- **Precision at position k**:

$$\text{prec@}k(\boldsymbol{y}, f, \boldsymbol{x}) = \frac{1}{k} \sum_{r=1}^{k} [\![y_{\sigma(r)} = 1]\!] \,,$$

  where σ is a permutation of labels for $\boldsymbol{x}$ returned by ranker f.
- **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

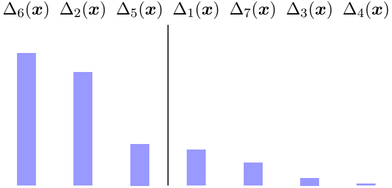
- **The optimal strategy**: rank labels according to the following marginal quantities:

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

Normalized Discounted Cumulative Gain

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

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



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.[2]

[2] 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.[2]
- **Thresholding** the conditional probabilities:

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

[2] 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.[2]
- **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.

[2] 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.[2]
- **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**: $F(\tau^*) = 2\tau^*$.

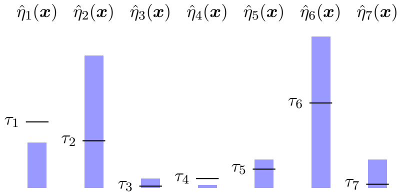
[2] O. Koyejo, N. Natarajan, P. Ravikumar, and I. Dhillon. Consistent multilabel classification. In *NIPS*, 2015

Macro-averaging of the F-measure

- Can be **solved** by **reduction** to m independent **binary** problems.[2]
- **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**: $F(\tau^*) = 2\tau^*$.



[2] 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

Predictive model

- From the above analysis we can conclude:

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

- We could use to this end the well-known **1-vs-All** approach.

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.

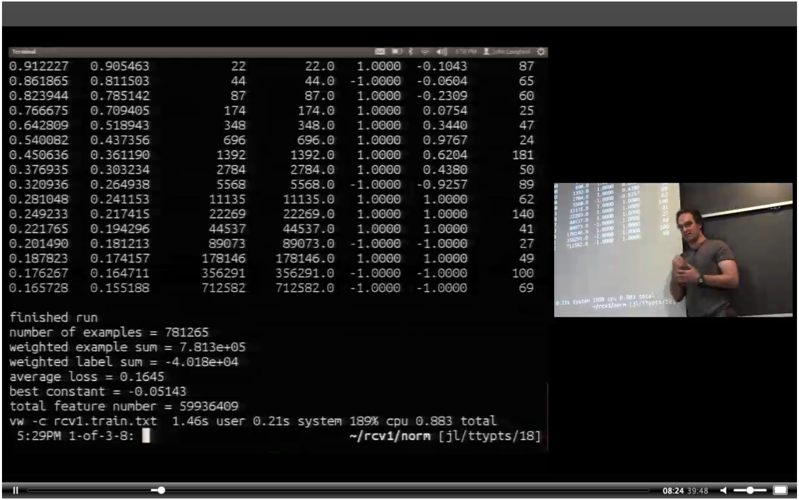


Figure: Vowpal Wabbit[3] at a lecture of John Langford[4]

3 Vowpal Wabbit, `http://hunch.net/~vw`

4 `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.

Reducing computational costs of the naive solution

- Linear models

Reducing computational costs of the naive solution

- Linear models
- Decision trees

Reducing computational costs of the naive solution

- Linear models
- Decision trees
- Label trees

Linear models

Linear models

- Fast training by least squares:[5]

[5] T. Hastie, R. Tibshirani, and J.H. Friedman. *Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, second edition, 2009

Linear models

- Fast training by least squares:[5]

$$\mathbf{W} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$$

[5] T. Hastie, R. Tibshirani, and J.H. Friedman. *Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, second edition, 2009

Linear models

- Fast training by least squares:[5]

$$\mathbf{W} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$$

- Works well in low dimensional feature spaces.
- Does not really improve space and test time complexity.

[5] T. Hastie, R. Tibshirani, and J.H. Friedman. *Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, second edition, 2009

Linear models

- Training time complexity:

[6] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *COMPSTAT*, pages 177–187, Paris, France, August 2010. Springer

[7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008

[8] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934, 2009

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008

[10] Rohit Babbar and Bernhard Schölkopf. Dismec - distributed sparse machines for extreme multi-label classification. *CoRR*, 2016

[11] K.Q. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120. ACM, 2009

Linear models

- Training time complexity:
  - Stochastic gradient descent[6] or coordinate gradient descent[7]

---

[6] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *COMPSTAT*, pages 177–187, Paris, France, August 2010. Springer

[7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008

[8] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934, 2009

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008

[10] Rohit Babbar and Bernhard Schölkopf. Dismec - distributed sparse machines for extreme multi-label classification. *CoRR*, 2016

[11] K.Q. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120. ACM, 2009

Linear models

- Training time complexity:
  - Stochastic gradient descent[6] or coordinate gradient descent[7]
  - Sparse feature vectors (e.g., sparse updates in SGD[8])

---

[6] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *COMPSTAT*, pages 177–187, Paris, France, August 2010. Springer

[7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008

[8] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934, 2009

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008

[10] Rohit Babbar and Bernhard Schölkopf. Dismec - distributed sparse machines for extreme multi-label classification. *CoRR*, 2016

[11] K.Q. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120. ACM, 2009

Linear models

- Training time complexity:
  - Stochastic gradient descent[6] or coordinate gradient descent[7]
  - Sparse feature vectors (e.g., sparse updates in SGD[8])
  - Negative sampling.[9]

---

[6] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *COMPSTAT*, pages 177–187, Paris, France, August 2010. Springer

[7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008

[8] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934, 2009

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008

[10] Rohit Babbar and Bernhard Schölkopf. Dismec - distributed sparse machines for extreme multi-label classification. *CoRR*, 2016

[11] K.Q. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120. ACM, 2009

Linear models

- Training time complexity:
  - Stochastic gradient descent[6] or coordinate gradient descent[7]
  - Sparse feature vectors (e.g., sparse updates in SGD[8])
  - Negative sampling.[9]
- Space complexity:

---

[6] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *COMPSTAT*, pages 177–187, Paris, France, August 2010. Springer

[7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008

[8] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934, 2009

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008

[10] Rohit Babbar and Bernhard Schölkopf. Dismec - distributed sparse machines for extreme multi-label classification. *CoRR*, 2016

[11] K.Q. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120. ACM, 2009

Linear models

- Training time complexity:
  - Stochastic gradient descent[6] or coordinate gradient descent[7]
  - Sparse feature vectors (e.g., sparse updates in SGD[8])
  - Negative sampling.[9]
- Space complexity:
  - Proper regularization: L_1 vs L_2.

---

[6] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *COMPSTAT*, pages 177–187, Paris, France, August 2010. Springer

[7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008

[8] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934, 2009

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008

[10] Rohit Babbar and Bernhard Schölkopf. Dismec - distributed sparse machines for extreme multi-label classification. *CoRR*, 2016

[11] K.Q. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120. ACM, 2009

Linear models

- Training time complexity:
  - Stochastic gradient descent[6] or coordinate gradient descent[7]
  - Sparse feature vectors (e.g., sparse updates in SGD[8])
  - Negative sampling.[9]
- Space complexity:
  - Proper regularization: L_1 vs L_2.
  - Removing small weights.[10]

---

[6] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *COMPSTAT*, pages 177–187, Paris, France, August 2010. Springer

[7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008

[8] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934, 2009

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008

[10] Rohit Babbar and Bernhard Schölkopf. Dismec - distributed sparse machines for extreme multi-label classification. *CoRR*, 2016

[11] K.Q. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120. ACM, 2009

Linear models

- Training time complexity:
  - Stochastic gradient descent[6] or coordinate gradient descent[7]
  - Sparse feature vectors (e.g., sparse updates in SGD[8])
  - Negative sampling.[9]
- Space complexity:
  - Proper regularization: L_1 vs L_2.
  - Removing small weights.[10]
  - Feature hashing.[11]

---

[6] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *COMPSTAT*, pages 177–187, Paris, France, August 2010. Springer

[7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008

[8] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934, 2009

[9] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008

[10] Rohit Babbar and Bernhard Schölkopf. Dismec - distributed sparse machines for extreme multi-label classification. *CoRR*, 2016

[11] K.Q. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120. ACM, 2009

Feature hashing in extreme classification

- Standard approach
  - A single slot for each weight and model.
  - Requires a lot of space.

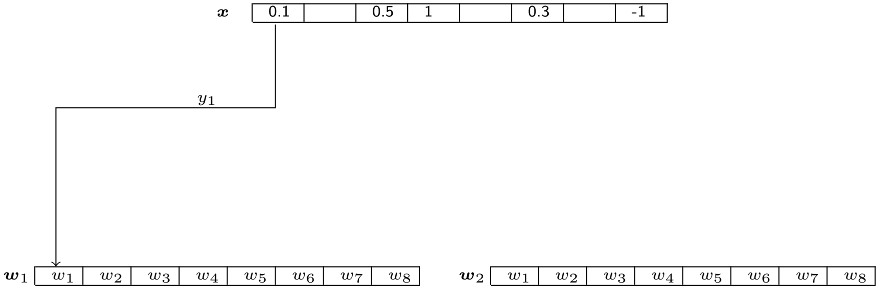
$\boldsymbol{x}$	0.1		0.5	1		0.3		-1

$\boldsymbol{w}_1$	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8

$\boldsymbol{w}_2$	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8

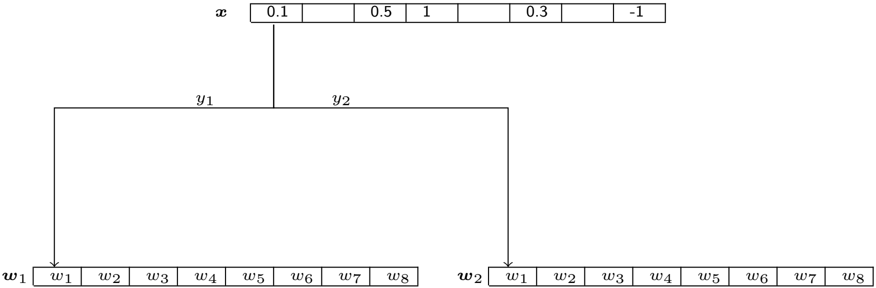
Feature hashing in extreme classification

- Standard approach
  - A single slot for each weight and model.
  - Requires a lot of space.



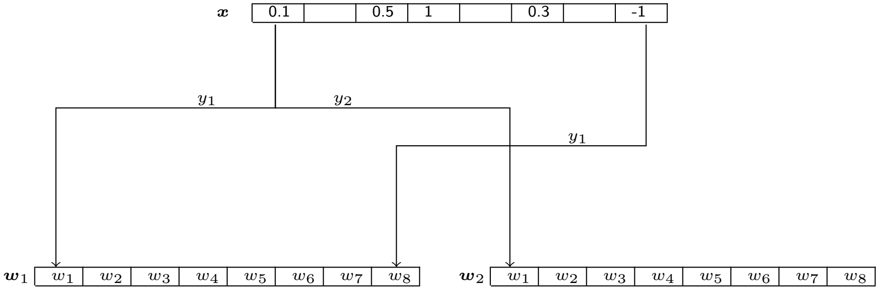
Feature hashing in extreme classification

- Standard approach
  - A single slot for each weight and model.
  - Requires a lot of space.



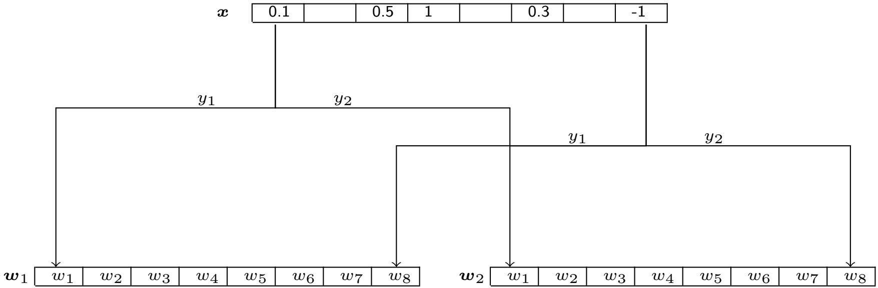
Feature hashing in extreme classification

- Standard approach
  - A single slot for each weight and model.
  - Requires a lot of space.



Feature hashing in extreme classification

- Standard approach
  - A single slot for each weight and model.
  - Requires a lot of space.



Feature hashing in extreme classification

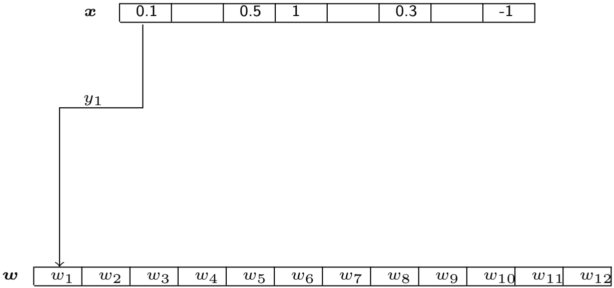
- Hashing to a common space
  - Hash the label and feature index using $h(j, v)$.
  - Hash a sign $\xi(j, v)$ to reduce the impact of conflicts.

$\boldsymbol{x}$	0.1		0.5	1		0.3		-1

$\boldsymbol{w}$	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}

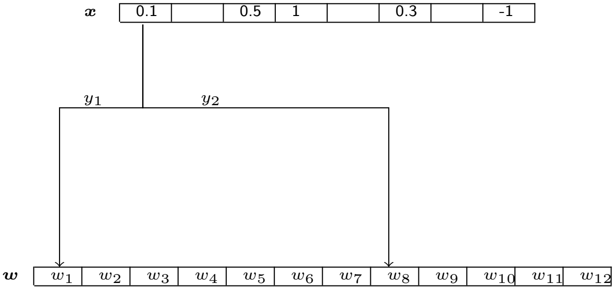
Feature hashing in extreme classification

- Hashing to a common space
  - Hash the label and feature index using $h(j, v)$.
  - Hash a sign $\xi(j, v)$ to reduce the impact of conflicts.



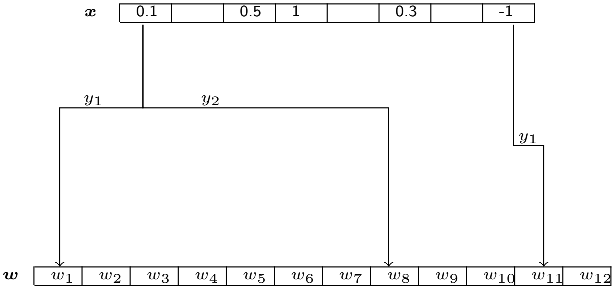
Feature hashing in extreme classification

- Hashing to a common space
  - Hash the label and feature index using $h(j, v)$.
  - Hash a sign $\xi(j, v)$ to reduce the impact of conflicts.



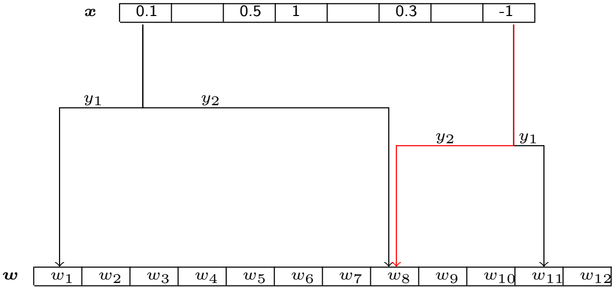
Feature hashing in extreme classification

- Hashing to a common space
  - Hash the label and feature index using $h(j, v)$.
  - Hash a sign $\xi(j, v)$ to reduce the impact of conflicts.



Feature hashing in extreme classification

- Hashing to a common space
  - Hash the label and feature index using $h(j, v)$.
  - Hash a sign $\xi(j, v)$ to reduce the impact of conflicts.



Linear models

- Low-dimensional representation of $\boldsymbol{x}$, $\mathbf{W}$, $\boldsymbol{y}$:

$$\boldsymbol{y} = \mathbf{U}^{\dagger}\mathbf{V}\boldsymbol{x}$$

  - feature space: PCA on $\mathbf{X}$.
  - label space: PCA no $\mathbf{Y}$,[12] compressed sensing,[13] etc.
  - both spaces: CCA on both $\mathbf{X}$ and $\mathbf{Y}$,[14] etc.
  - matrix factorization of $\mathbf{W}$.[15]
  - A kind of **lossy compression**/**embedding** methods.

---

[12] F. Tai and H.-T. Lin. Multi-label classification with principal label space transformation. In *Neural Computat.*, volume 9, pages 2508–2542, 2012

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

[14] Yao-Nan Chen and Hsuan-Tien Lin. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*, pages 1529–1537. Curran Associates, Inc., 2012

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

Computational challenges

- Prediction time is still **linear** in the number of labels!

Computational challenges

- Prediction time is still **linear** in the number of labels!
- **Reduce** the test time complexity by:
  - Maximum inner product search over linear models,
  - Decision trees,
  - Label trees.

Test time complexity for linear models

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

$$j^* = \arg\max_{j \in \{1,\dots,m\}} \boldsymbol{w}_j^\top \boldsymbol{x}\,,$$

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

MIPS vs. nearest neighbors

- MIPS is similar, but not the same, to the nearest neighbor search under the square or cosine distance:

$$j^* = \arg\min_{j \in \{1,\dots,m\}} \|\boldsymbol{w}_j - \boldsymbol{x}\|_2^2 = \arg\max_{j \in \{1,\dots,m\}} \boldsymbol{w}_j^\top \boldsymbol{x} - \frac{\|\boldsymbol{w}_j\|_2^2}{2}$$

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

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

[17] 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

[18] 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

MIPS vs. nearest neighbors

- MIPS is similar, but not the same, to the nearest neighbor search under the square or cosine distance:

$$j^* = \arg\min_{j \in \{1,\dots,m\}} \|\boldsymbol{w}_j - \boldsymbol{x}\|_2^2 = \arg\max_{j \in \{1,\dots,m\}} \boldsymbol{w}_j^\top \boldsymbol{x} - \frac{\|\boldsymbol{w}_j\|_2^2}{2}$$

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

- Some tricks are used to treat MIPS as nearest neighbor search.[16]

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

[17] 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

[18] 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

MIPS vs. nearest neighbors

- MIPS is similar, but not the same, to the nearest neighbor search under the square or cosine distance:

$$
\begin{aligned}
j^* &= \arg\min_{j \in \{1,\ldots,m\}} \|\boldsymbol{w}_j - \boldsymbol{x}\|_2^2 = \arg\max_{j \in \{1,\ldots,m\}} \boldsymbol{w}_j^\top \boldsymbol{x} - \frac{\|\boldsymbol{w}_j\|_2^2}{2} \\
j^* &= \arg\max_{j \in \{1,\ldots,m\}} \frac{\boldsymbol{w}_j^\top \boldsymbol{x}}{\|\boldsymbol{w}_j\|\|\boldsymbol{x}\|} = \arg\max_{j \in \{1,\ldots,m\}} \frac{\boldsymbol{w}_j^\top \boldsymbol{x}}{\|\boldsymbol{w}_j\|}
\end{aligned}
$$

- Some tricks are used to treat MIPS as nearest neighbor search.[16]
  - For low-dimensional problems, efficient tree-based structures exist.[17]

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

[17] 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

[18] 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

MIPS vs. nearest neighbors

- MIPS is similar, but not the same, to the nearest neighbor search under the square or cosine distance:

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

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

- Some tricks are used to treat MIPS as nearest neighbor search.[16]
  - For low-dimensional problems, efficient tree-based structures exist.[17]
  - Approximate nearest neighbor search via locality-sensitive hashing.[18]

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

[17] 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

[18] 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

Decision trees

Decision trees

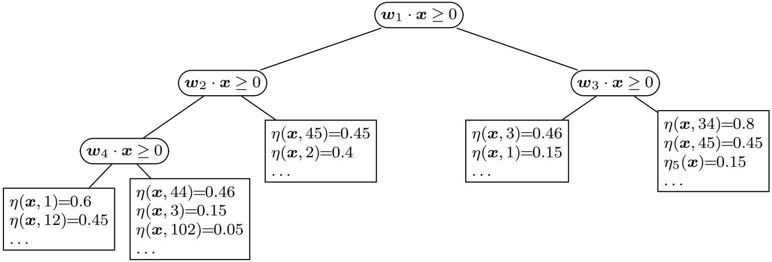
- Fast prediction: logarithmic in n
- Training can be expensive: computation of split criterion
- Two new algorithms: LomTree[19] and **FastXML**[20]

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

[20] 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

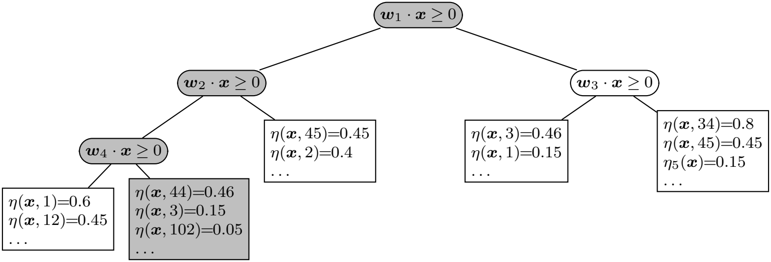
FastXML

- 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:

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

Optimization in FastXML

- In each internal node FastXML solves:

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

linear split

partitioning of training examples

label ranking in positive and negative partition

Optimization in FastXML

- In each internal node FastXML solves:

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

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

linear split

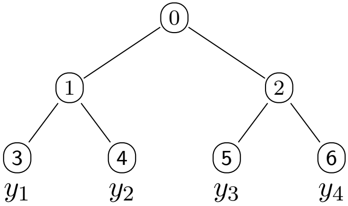
partitioning of training examples

label ranking in positive and negative partition

Label trees

Label trees

- Organize classifiers in a tree structure (one leaf $\Leftrightarrow$ one label).[21]

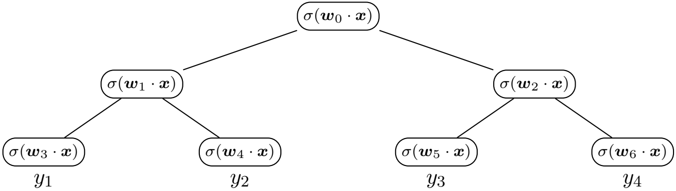


- Structure of the tree can be given or trained.
- Different training and test procedures for multi-class and multi-label classification.

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

Probabilistic label trees (PLT)[22]

- PLT are based on b-ary **label trees**.

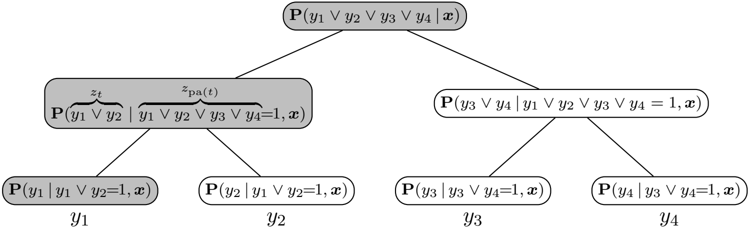


- **Probabilistic classifiers** in **all** nodes of the tree.
- **Internal** node classifier decides whether to **go down the tree**.
- A test example may follow **many paths** from the root to leaves.
- **Batch** and **online** learning possible.

[22] 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

- Class probability estimators in nodes for estimating $\mathbf{P}(y_j = 1 \mid \boldsymbol{x})$.



- Using the **chain rule** of probability

$$\mathbf{P}(y_j = 1 \mid \boldsymbol{x}) = \eta_j(\boldsymbol{x}) = \prod_{t \in \mathrm{Path}(j)} \eta(\boldsymbol{x}, t)\,,$$

where $\eta(\boldsymbol{x}, t) = \begin{cases} \mathbf{P}(z_t = 1 \mid \boldsymbol{x}) & \text{if } t \text{ is root,} \\ \mathbf{P}(z_t = 1 \mid z_{\mathrm{pa}(t)} = 1, \boldsymbol{x}) & \text{otherwise.} \end{cases}$

Probabilistic label trees

- Chain rule of probability:

$$\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1 \,|\, \boldsymbol{x}) \times \mathbf{P}(y_1 \vee y_2 = 1 \,|\, y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x}) =$$

Probabilistic label trees

- Chain rule of probability:

$$\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1 \,|\, \boldsymbol{x}) \times \mathbf{P}(y_1 \vee y_2 = 1 \,|\, y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x}) =$$

$$\frac{\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} \times \frac{\mathbf{P}(y_1 \vee y_2 = 1, y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})}{\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})} =$$

Probabilistic label trees

- Chain rule of probability:

$$\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1 \,|\, \boldsymbol{x}) \times \mathbf{P}(y_1 \vee y_2 = 1 \,|\, y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x}) =$$

$$\frac{\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} \times \frac{\mathbf{P}(y_1 \vee y_2 = 1, y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})}{\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})} =$$

$$\frac{\mathbf{P}(y_1 \vee y_2 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} = \mathbf{P}(y_1 \vee y_2 = 1 \,|\, \boldsymbol{x})$$

- Chain rule of probability:

$$\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1 \,|\, \boldsymbol{x}) \times \mathbf{P}(y_1 \vee y_2 = 1 \,|\, y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x}) =$$
$$\frac{\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} \times \frac{\mathbf{P}(y_1 \vee y_2 = 1, y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})}{\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})} =$$
$$\frac{\mathbf{P}(y_1 \vee y_2 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} = \mathbf{P}(y_1 \vee y_2 = 1 \,|\, \boldsymbol{x})$$

$$\mathbf{P}(y_1 \vee y_2 = 1 \,|\, \boldsymbol{x}) \times \mathbf{P}(y_1 = 1 \,| y_1 \vee y_2 = 1 \boldsymbol{x}) =$$

Probabilistic label trees

- Chain rule of probability:

$$\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1 \,|\, \boldsymbol{x}) \times \mathbf{P}(y_1 \vee y_2 = 1 \,|\, y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x}) =$$

$$\frac{\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} \times \frac{\mathbf{P}(y_1 \vee y_2 = 1, y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})}{\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})} =$$

$$\frac{\mathbf{P}(y_1 \vee y_2 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} = \mathbf{P}(y_1 \vee y_2 = 1 \,|\, \boldsymbol{x})$$

$$\mathbf{P}(y_1 \vee y_2 = 1 \,|\, \boldsymbol{x}) \times \mathbf{P}(y_1 = 1 \,| y_1 \vee y_2 = 1 \boldsymbol{x}) =$$

$$\frac{\mathbf{P}(y_1 \vee y_2 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} \times \frac{\mathbf{P}(y_1 = 1, y_1 \vee y_2 = 1, \boldsymbol{x})}{\mathbf{P}(y_1 \vee y_2 = 1, \boldsymbol{x})} =$$

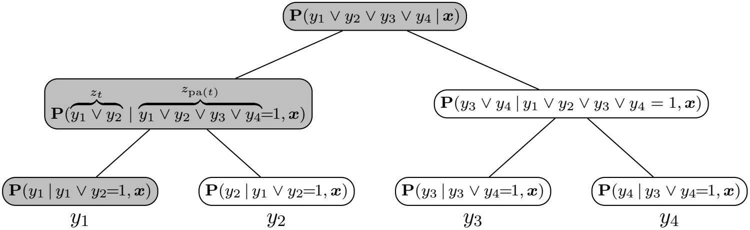
- Chain rule of probability:

$$\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1 \,|\, \boldsymbol{x}) \times \mathbf{P}(y_1 \vee y_2 = 1 \,|\, y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x}) =$$
$$\frac{\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} \times \frac{\mathbf{P}(y_1 \vee y_2 = 1, y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})}{\mathbf{P}(y_1 \vee y_2 \vee y_3 \vee y_4 = 1, \boldsymbol{x})} =$$
$$\frac{\mathbf{P}(y_1 \vee y_2 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} = \mathbf{P}(y_1 \vee y_2 = 1 \,|\, \boldsymbol{x})$$

$$\mathbf{P}(y_1 \vee y_2 = 1 \,|\, \boldsymbol{x}) \times \mathbf{P}(y_1 = 1 \,| y_1 \vee y_2 = 1 \boldsymbol{x}) =$$
$$\frac{\mathbf{P}(y_1 \vee y_2 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} \times \frac{\mathbf{P}(y_1 = 1, y_1 \vee y_2 = 1, \boldsymbol{x})}{\mathbf{P}(y_1 \vee y_2 = 1, \boldsymbol{x})} =$$
$$\frac{\mathbf{P}(y_1 = 1, \boldsymbol{x})}{\mathbf{P}(\boldsymbol{x})} = \mathbf{P}(y_1 = 1 \,|\, \boldsymbol{x})$$

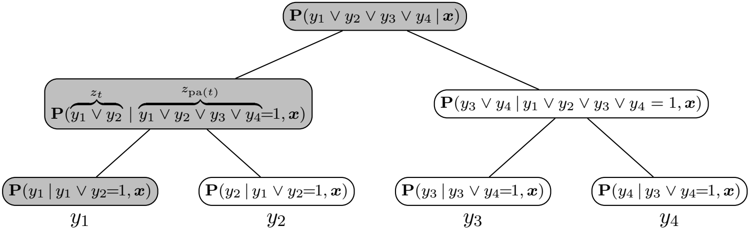
Probabilistic label trees

- Class probability estimators in nodes for estimating $\mathbf{P}(y_i = 1 \mid \boldsymbol{x})$.



Probabilistic label trees

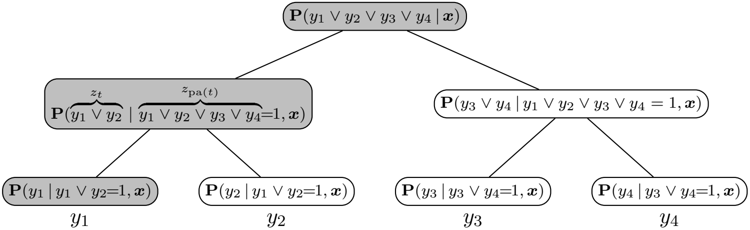
- Class probability estimators in nodes for estimating $\mathbf{P}(y_i = 1 \mid \boldsymbol{x})$.



- Training: reduced complexity by the **conditions** used in the **nodes**.

Probabilistic label trees

- Class probability estimators in nodes for estimating $\mathbf{P}(y_i = 1 \,|\, \boldsymbol{x})$.



- Training: reduced complexity by the **conditions** used in the **nodes**.
- Prediction: **priority queue search** or **branch and bound**.

Probabilistic label trees

- The same idea under different names:
  - **Conditional probability trees**[23]
  - **Probabilistic classifier chains**[24]
  - **Hierarchical softmax**[25]
  - **Homer**[26]
  - **Nested dichotomies**[27]
  - **Multi-stage classification**[28]

---

[23] 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

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

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

[26] G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data*, 2008

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

[28] Marek Kurzynski. On the multistage bayes classifier. *Pattern Recognition*, 21(4):355–365, 1988

FastXML vs. PLT

	FastXML	PLT
tree structure	✓	✓
structure learning	✓	×
number of trees	≥ 1	1
number of leaves	$O(n)$	m
internal nodes models	linear	linear
leaves models	empirical distribution	linear
visited paths during prediction	1 per tree	several
sparse probability estimation	✓	✓

Experimental results

	#labels	#features	#test	#train	inst./lab.	lab./inst.
RCV1	2456	47236	155962	623847	1218.56	4.79
AmazonCat	13330	203882	306782	1186239	448.57	5.04
Wiki10	30938	101938	6616	14146	8.52	18.64
Delicious	205443	782585	100095	196606	72.29	75.54
WikiLSHTC	325056	1617899	587084	1778351	17.46	3.19
Amazon	670091	135909	153025	490449	3.99	5.45

Table: Datasets from the Extreme Classification repository.[29]

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

Experimental results

	PLT			FastXML		
	P@1	P@3	P@5	P@1	P@3	P@5
RCV1	90.46	72.4	51.86	**91.13**	**73.35**	**52.67**
AmazonCat	91.47	75.84	61.02	**92.95**	**77.5**	**62.51**
Wiki10	**84.34**	**72.34**	**62.72**	81.71	66.67	56.70
Delicious	**45.37**	**38.94**	35.88	42.81	38.76	**36.34**
WikiLSHTC	45.67	29.13	21.95	**49.35**	**32.69**	**24.03**
Amazon	**36.65**	**32.12**	**28.85**	34.24	29.3	26.12

Experimental results

	PLT					FastXML			
	train [min]	test [ms]	b	depth	#calls	train [min]	test [ms]	depth	#calls
RCV1	**64**	0.22	32	2,25	**43,46**	78	0.96	14.95	747
AmazonCat	**96**	0.17	16	3,43	**54,39**	561	1.14	17.44	871
Wiki10	290	2.66	32	2,98	**121,98**	**16**	3.00	10.83	541
Delicious	1327	32.97	2	17,69	11779,65	**458**	4.01	14.79	**739**
WikiLSHTC	**653**	3.00	32	3,66	**622,27**	724	1.17	18.01	900
Amazon	**54**	0.99	8	6,45	**374,30**	422	1.39	15.92	796

Challenges

Challenges

- Learning theory for an extremely large number of labels:

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**.

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).

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**.

Challenges

- Training and prediction under limited time and space budget:

Challenges

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

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**.

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**.

Challenges

- Unreliable learning information:

Challenges

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

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**.

Challenges

- Performance measures:

Challenges

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

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**.

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?

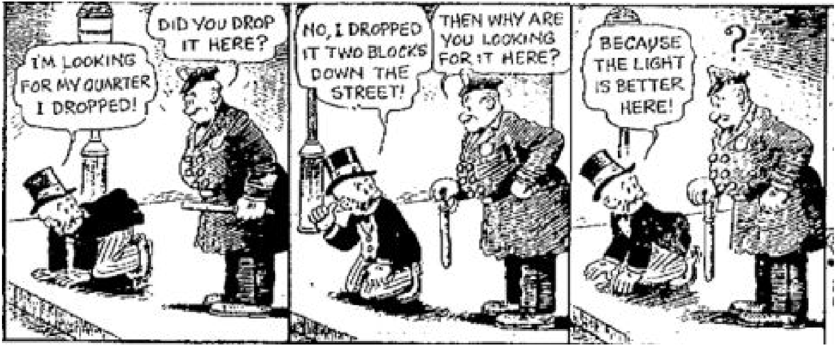


Figure: [30] A similar comics has been earlier used by Asela Gunawardana.[31]

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

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

Challenges

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

Challenges

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

Challenges

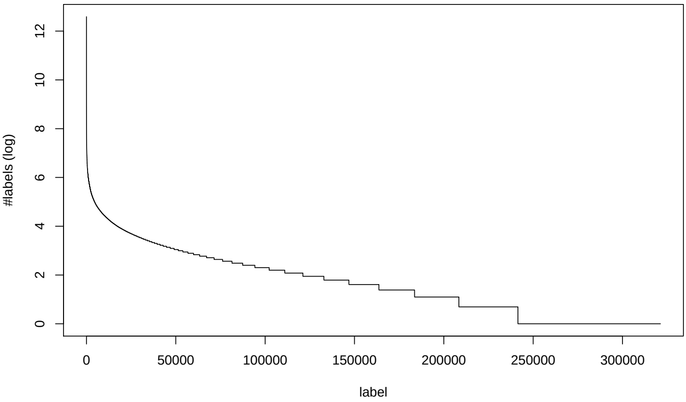
- Long-tail label distributions and zero-shot learning:
  - A close relation to the problem of **estimating distributions over large alphabets**.
  - The distribution of label frequencies is often characterized by a **long-tail** for which proper **smoothing** (like add-constant or Good-Turing estimates)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

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**)

Challenges

- Long-tail label distributions and zero-shot learning:
  - Frequency of labels in the WikiLSHTC dataset:[32]

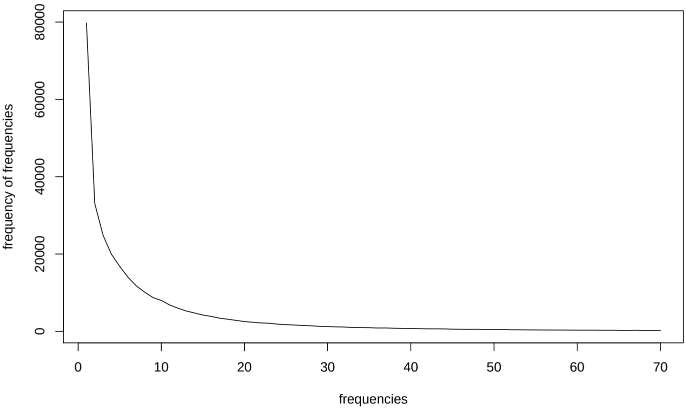


  - Many labels with only few examples ($\Rightarrow$ one-shot learning).

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

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.

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 challenges**: fast learning of linear models, compression, MIPS, decision trees, label trees.

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational challenges**: fast learning of linear models, compression, MIPS, decision trees, label trees.
- **Statistical challenges**: Is learning possible in the extreme setting?

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational challenges**: fast learning of linear models, compression, MIPS, decision trees, label trees.
- **Statistical challenges**: Is learning possible in the extreme setting?
- For more check:

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational challenges**: fast learning of linear models, compression, MIPS, decision trees, label trees.
- **Statistical challenges**: Is learning possible in the extreme setting?
- For more check:
  - `http://www.cs.put.poznan.pl/kdembczynski`

Take-away message

- **Extreme classification**: #examples, #features, **#labels**
- **Complexity**: time vs. space, training vs. validation vs. **prediction**
- **Computational challenges**: fast learning of linear models, compression, MIPS, decision trees, label trees.
- **Statistical challenges**: Is learning possible in the extreme setting?
- For more check:
  - `http://www.cs.put.poznan.pl/kdembczynski`
  - **Code**: `https://github.com/busarobi/XMLC`