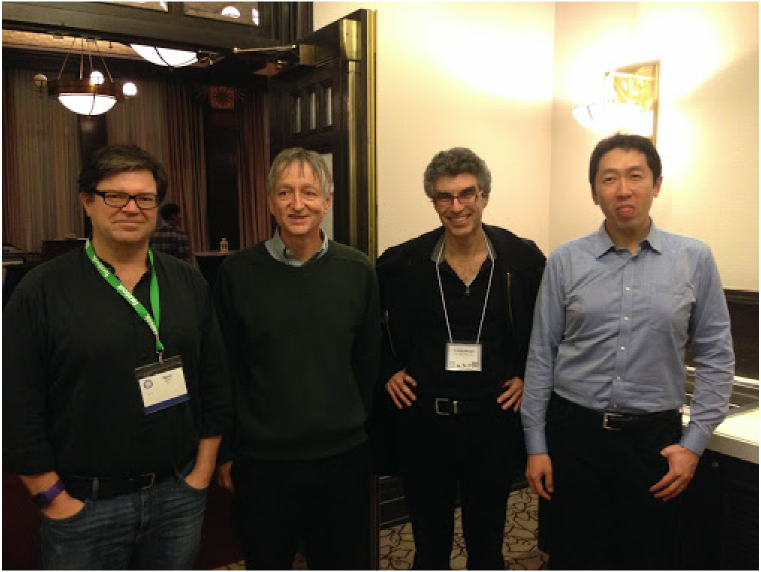
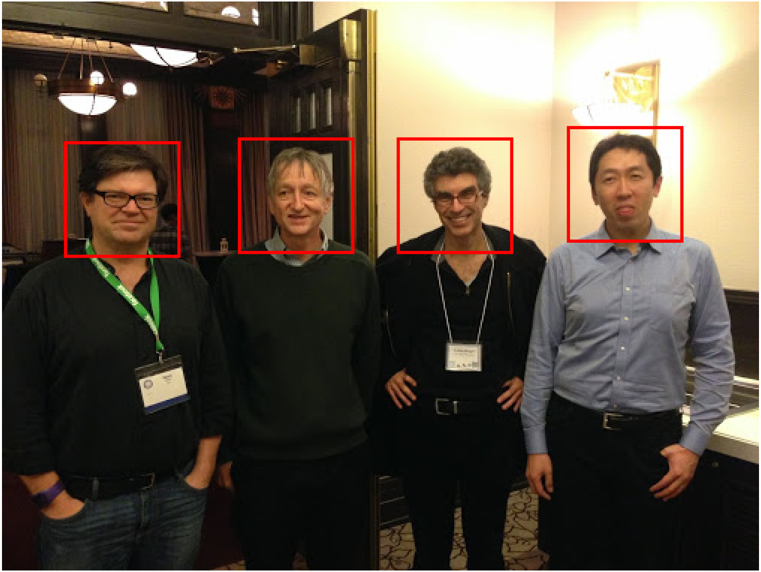
# Extreme Classification: Machine Learning with Millions of Labels

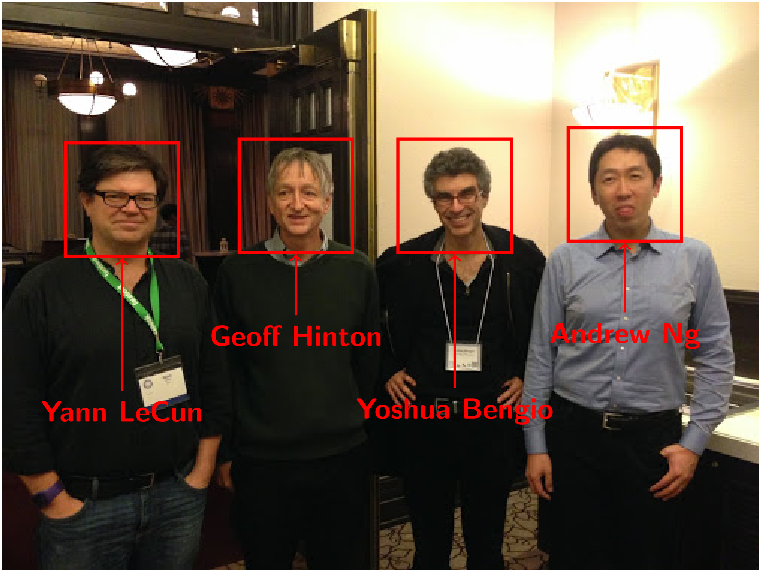
Krzysztof Dembczyński

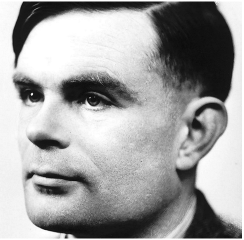
Intelligent Decision Support Systems Laboratory (IDSS)
Poznań University of Technology, Poland

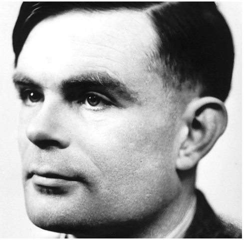
Uniwersytet Adama Mickiewicza
Poznań, May 24, 2017






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**:

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

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

# Setting

- **Multi-class classification**:

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

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

- **Multi-label classification**:

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

| | $x_1$ | $x_2$ | $\ldots$ | $x_p$ | $y_1$ | $y_2$ | $\ldots$ | $y_m$ |
|---|---|---|---|---|---|---|---|---|
| $\boldsymbol{x}$ | 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**:
  - Learning theory for large $m$

# Extreme classification

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

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

# Extreme classification

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

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

# Extreme classification

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

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

## 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.

# 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

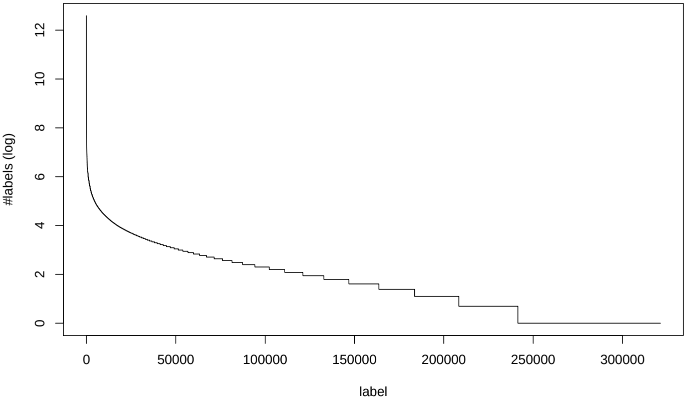
- Long-tail label distributions and zero-shot learning:
  - A close relation to the problem of **estimating distributions over large alphabets**.
  - The distribution of label frequencies is often characterized by a **long-tail** for which proper **smoothing** (like add-constant or Good-Turing estimates) 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:[1]

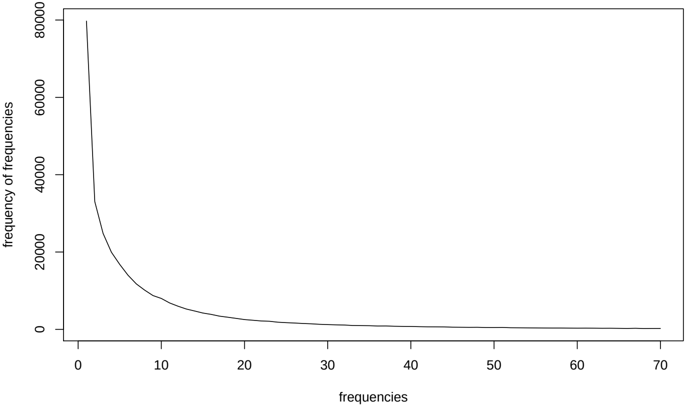



  - Many labels with only few examples (⇒ one-shot learning).

[1] http://research.microsoft.com/en-us/um/people/manik/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.

# 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**:
  - Large data $\longrightarrow$ sparse data (sparse features and labels)

# Computational challenges: naive solution

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

# Computational challenges: naive solution

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

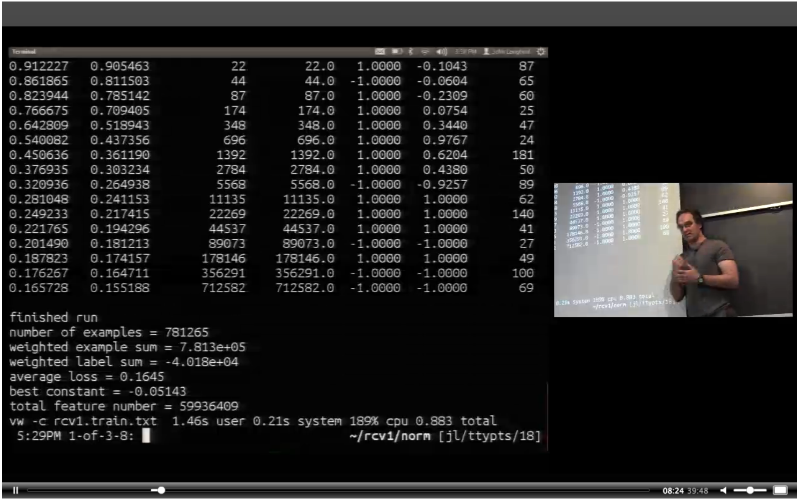



Figure: Vowpal Wabbit[2] at a lecture of John Langford[3]

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

[3] `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.

# Computational challenges

- **How can we reduce computational (time and space) costs of the naive solution?**

# Computational challenges

- **How can we reduce computational (time and space) costs of the naive solution?**
  - Linear models
  - Nearest neighbors
  - Hashing
  - Decision trees
  - Label trees

# Linear models

# Linear models

- Fast training by least squares:[4]

[4] 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:[4]

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

[4] 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:[4]

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

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

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

- Training time complexity:

---

[5] 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

[6] 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

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

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

[9] 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

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

# Linear models

- Training time complexity:
  - Stochastic gradient descent[5] or coordinate gradient descent[6]

---

[5] 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

[6] 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

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

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

[9] 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

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

## Linear models

- Training time complexity:
  - Stochastic gradient descent[5] or coordinate gradient descent[6]
  - Sparse feature vectors (e.g., sparse updates in SGD[7])

---

[5] 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

[6] 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

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

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

[9] 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

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

# Linear models

- Training time complexity:
  - Stochastic gradient descent[5] or coordinate gradient descent[6]
  - Sparse feature vectors (e.g., sparse updates in SGD[7])
  - Negative sampling.[8]

---

[5] 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

[6] 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

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

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

[9] 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

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

# Linear models

- Training time complexity:
  - Stochastic gradient descent[5] or coordinate gradient descent[6]
  - Sparse feature vectors (e.g., sparse updates in SGD[7])
  - Negative sampling.[8]
- Space complexity:

---

[5] 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

[6] 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

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

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

[9] 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

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

# Linear models

- Training time complexity:
  - Stochastic gradient descent[5] or coordinate gradient descent[6]
  - Sparse feature vectors (e.g., sparse updates in SGD[7])
  - Negative sampling.[8]
- Space complexity:
  - Proper regularization: $L_1$ vs $L_2$.

---

5 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

6 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

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

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

9 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

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

# Linear models

- Training time complexity:
  - Stochastic gradient descent[5] or coordinate gradient descent[6]
  - Sparse feature vectors (e.g., sparse updates in SGD[7])
  - Negative sampling.[8]
- Space complexity:
  - Proper regularization: $L_1$ vs $L_2$.
  - Feature hashing.[9]

---

5 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

6 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

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

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

9 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

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

# Linear models

- Training time complexity:
  - Stochastic gradient descent[5] or coordinate gradient descent[6]
  - Sparse feature vectors (e.g., sparse updates in SGD[7])
  - Negative sampling.[8]
- Space complexity:
  - Proper regularization: $L_1$ vs $L_2$.
  - Feature hashing.[9]
  - Removing small weights.[10]

---

[5] 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

[6] 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

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

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

[9] 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

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

# 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 $\mathbf{Y}$,[11] compressed sensing,[12] etc.
  - both spaces: CCA on both $\mathbf{X}$ and $\mathbf{Y}$,[13] etc.
  - matrix factorization of $\mathbf{W}$.[14]
  - A kind of **lossy compression**/**embedding** methods.

---

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

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

[13] 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

[14] 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** test time complexity by using appropriate data structures:

# Computational challenges

- Prediction time is still **linear** in the number of labels!
- **Reduce** test time complexity by using appropriate data structures:
  - Hashing

# Computational challenges

- Prediction time is still **linear** in the number of labels!
- **Reduce** test time complexity by using appropriate data structures:
  - Hashing ($\longrightarrow$ clustering).

## Computational challenges

- Prediction time is still **linear** in the number of labels!
- **Reduce** test time complexity by using appropriate data structures:
  - Hashing ($\longrightarrow$ clustering).
  - Sorting $\longrightarrow$ **trees**

# Computational challenges

- Prediction time is still **linear** in the number of labels!
- **Reduce** test time complexity by using appropriate data structures:
  - Hashing ($\longrightarrow$ clustering).
  - Sorting $\longrightarrow$ **trees**
  - $\longrightarrow$ **decision trees**.
  - $\longrightarrow$ **label trees**.

# Test time complexity for linear models

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

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

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

# Test time complexity for linear models

- Exact solution: the threshold algorithm[15]
  - Requires efficient sorted and random access to the weights.
  - Based on a lower and upper bound on the result.
  - Sorting of feature weights over different models/labels.
  - Storing the sorted lists.
  - Optimal in terms of time complexity.

[15] 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, but not the same, to the nearest neighbor search under the square or cosine distance:

$$
\begin{aligned}
i^* &= \operatorname*{arg\,min}_{i \in \{1,\ldots,m\}} \|\boldsymbol{w}_i - \boldsymbol{x}\|_2^2 = \operatorname*{arg\,max}_{i \in \{1,\ldots,m\}} \boldsymbol{w}_i^\top \boldsymbol{x} - \frac{\|\boldsymbol{w}_i\|_2^2}{2} \\
i^* &= \operatorname*{arg\,max}_{i \in \{1,\ldots,m\}} \frac{\boldsymbol{w}_i^\top \boldsymbol{x}}{\|\boldsymbol{w}_i\|\|\boldsymbol{x}\|} = \operatorname*{arg\,max}_{i \in \{1,\ldots,m\}} \frac{\boldsymbol{w}_i^\top \boldsymbol{x}}{\|\boldsymbol{w}_i\|}
\end{aligned}
$$

- 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

# Test time complexity

- Generalization of MIPS
  - k-MIPS (for prec@k)
  - Inner products above a given threshold (for Hamming loss)

# Nearest neighbors

# Nearest neighbors

- In general, the space and time complexity is linear in $n$.

[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*, 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

# Nearest neighbors

- In general, the space and time complexity is linear in $n$.
- This also implies linear complexity in $m$.

[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*, 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

# Nearest neighbors

- In general, the space and time complexity is linear in $n$.
- This also implies linear complexity in $m$.
- For low-dimensional problems, efficient tree-based structures exist.[17]

[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*, 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

# Nearest neighbors

- In general, the space and time complexity is linear in $n$.
- This also implies linear complexity in $m$.
- For low-dimensional problems, efficient tree-based structures exist.[17]
- Approximate nearest neighbor search via locality-sensitive hashing.[18]

[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*, 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

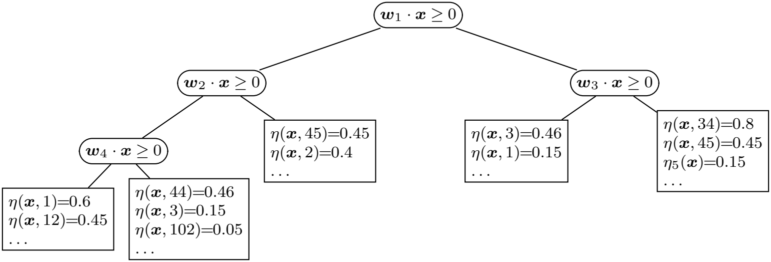
- Fast prediction: logarithmic in $n$
- Training can be expensive: computation of split criterion
- Two new algorithms: LomTree[19] and FastXML[20]

[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

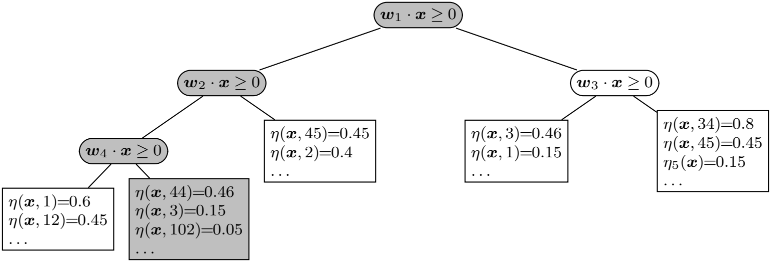
# FastXML

- 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.

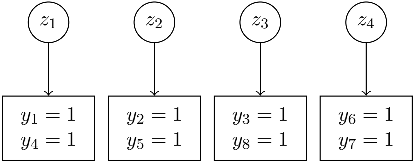


# Hashing

# Hashing

- Hash label indexes to integers in $\{1, \ldots, r\}$:

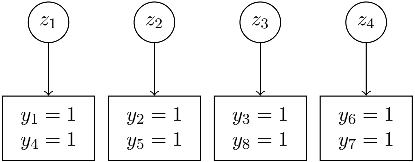
$$z_j = [\![hash(i) = j \wedge y_i = 1]\!], \quad j = 1, \ldots, r$$



## Hashing

- Hash label indexes to integers in $\{1, \ldots, r\}$:

$$z_j = [\![ hash(i) = j \wedge y_i = 1 ]\!], \quad j = 1, \ldots, r$$

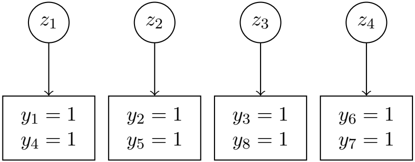



- Train $r$ binary models, one for each hash value.

## Hashing

- Hash label indexes to integers in $\{1, \ldots, r\}$:

$$z_j = [\![hash(i) = j \wedge y_i = 1]\!], \quad j = 1, \ldots, r$$

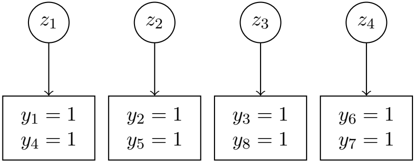



- Train $r$ binary models, one for each hash value.
- Decode original labels from hash values.

## Hashing

- Hash label indexes to integers in $\{1, \ldots, r\}$:

$$z_j = [\![hash(i) = j \wedge y_i = 1]\!], \quad j = 1, \ldots, r$$

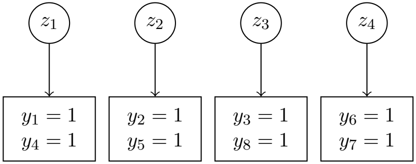



- Train $r$ binary models, one for each hash value.
- Decode original labels from hash values.
- Learning and prediction linear in $r$ instead of $m$.

# Hashing

- Hash label indexes to integers in $\{1, \ldots, r\}$:

$$z_j = [\![hash(i) = j \wedge y_i = 1]\!], \quad j = 1, \ldots, r$$

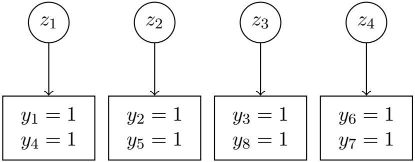



- Train $r$ binary models, one for each hash value.
- Decode original labels from hash values.
- Learning and prediction linear in $r$ instead of $m$.
- Clustering can be used to obtain good hash functions.

# Hashing

- Hash label indexes to integers in $\{1, \ldots, r\}$:

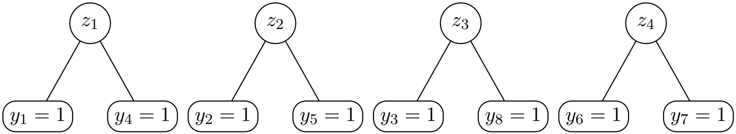
$$z_j = [\![hash(i) = j \wedge y_i = 1]\!], \quad j = 1, \ldots, r$$




- Train $r$ binary models, one for each hash value.
- Decode original labels from hash values.
- Learning and prediction linear in $r$ instead of $m$.
- Clustering can be used to obtain good hash functions.
- How to resolve conflicts?

# Hashing

- Resolving conflicts → Train a classifier for each original label:

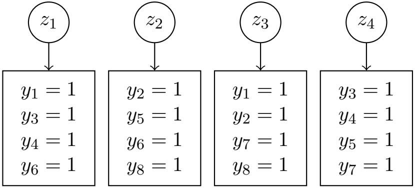



- Learning complexity increases, but prediction is sublinear in $m$.
- More levels → label trees

## Bloom filters

- Resolving conflicts → Use more than one hash function:[21]

$$z_j = [\![ \bigvee_{h=1}^{k} hash_h(i) = j \wedge y_i = 1 ]\!], \quad j = 1, \ldots, r$$

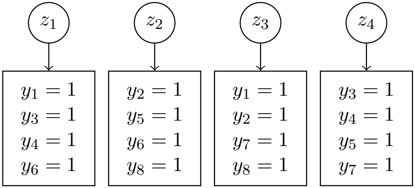



[21] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970
Moustapha Cissé, Nicolas Usunier, Thierry Artières, and Patrick Gallinari. Robust bloom filters for large multilabel classification tasks. In *NIPS*, pages 1851–1859, 2013

## Bloom filters

- Resolving conflicts → Use more than one hash function:[21]

$$z_j = [\![ \bigvee_{h=1}^{k} hash_h(i) = j \wedge y_i = 1 ]\!], \quad j = 1, \ldots, r$$



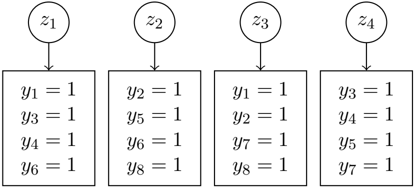

- With deterministic data only false positives appear.

[21] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970
Moustapha Cissé, Nicolas Usunier, Thierry Artières, and Patrick Gallinari. Robust bloom filters for large multilabel classification tasks. In *NIPS*, pages 1851–1859, 2013

# Bloom filters

- Resolving conflicts → Use more than one hash function:[21]

$$z_j = [\![\bigvee_{h=1}^{k} hash_h(i) = j \wedge y_i = 1]\!], \quad j = 1, \ldots, r$$



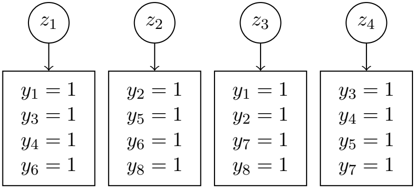

- With deterministic data only false positives appear.
- More hash functions → more combinations but also 1s in the filter.

[21] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970
Moustapha Cissé, Nicolas Usunier, Thierry Artières, and Patrick Gallinari. Robust bloom filters for large multilabel classification tasks. In *NIPS*, pages 1851–1859, 2013

# Bloom filters

- Resolving conflicts → Use more than one hash function:[21]

$$z_j = [\![ \bigvee_{h=1}^{k} hash_h(i) = j \wedge y_i = 1 ]\!], \quad j = 1, \ldots, r$$




- With deterministic data only false positives appear.
- More hash functions → more combinations but also 1s in the filter.
- Proper tuning of $r$ and $k$.

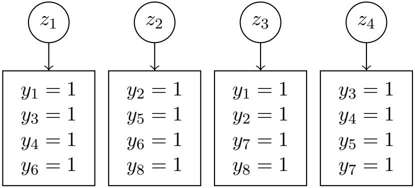
---

[21] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970
Moustapha Cissé, Nicolas Usunier, Thierry Artières, and Patrick Gallinari. Robust bloom filters for large multilabel classification tasks. In *NIPS*, pages 1851–1859, 2013

# Bloom filters

- Resolving conflicts → Use more than one hash function:[21]

$$z_j = [\![ \bigvee_{h=1}^{k} hash_h(i) = j \wedge y_i = 1 ]\!], \quad j = 1, \ldots, r$$



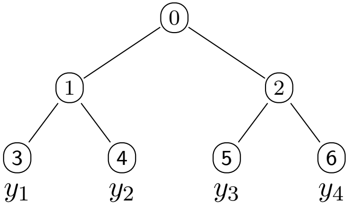

- With deterministic data only false positives appear.
- More hash functions → more combinations but also 1s in the filter.
- Proper tuning of $r$ and $k$.
- Hash functions can be obtained by (non-disjoint) clustering.

[21] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970
Moustapha Cissé, Nicolas Usunier, Thierry Artières, and Patrick Gallinari. Robust bloom filters for large multilabel classification tasks. In *NIPS*, pages 1851–1859, 2013

# Label trees

# Label trees

- Organize classifiers in a tree structure (one leaf $\Leftrightarrow$ one label).[22]

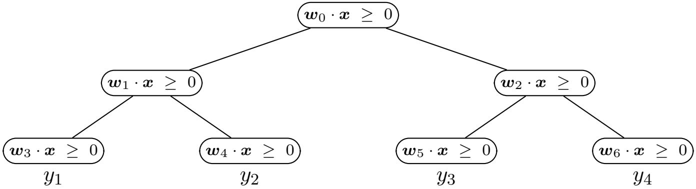



- Structure of the tree can be given or trained.
- Different training and test procedures for multi-class and multi-label classification.

[22] 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)[23]

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

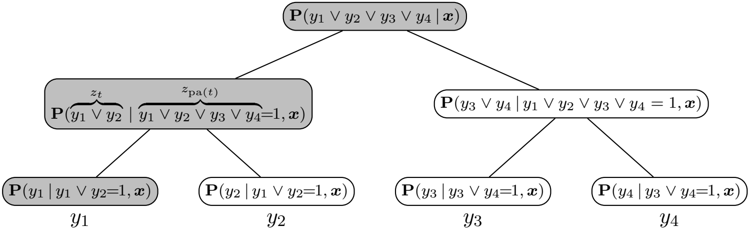



- **Probabilistic classifiers** in **all** nodes of the tree.
- **Internal** node classifier decides whether to **go down the tree**.
- A test example may follow **many paths** from the root to leaves.

[23] 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 $\mathbf{P}(y_i = 1 \,|\, \boldsymbol{x})$.



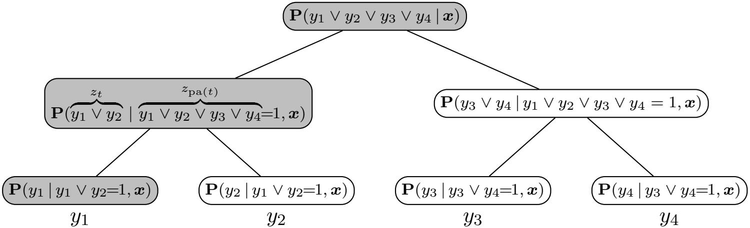

- Using the **chain rule** of probability

$$\mathbf{P}(y_i = 1 \,|\, \boldsymbol{x}) = \eta(\boldsymbol{x}, i) = \prod_{t \in \mathrm{Path}(i)} \eta_T(\boldsymbol{x}, t)\,,$$

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

# Probabilistic label trees

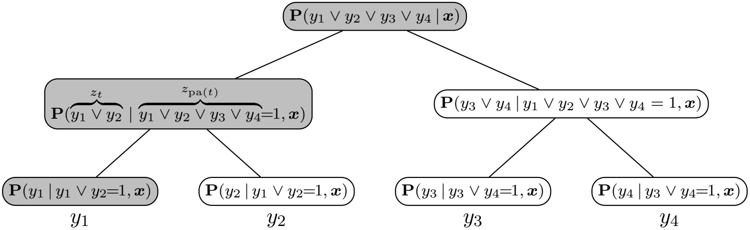
- Class probability estimators in nodes for estimating $\mathbf{P}(y_i = 1 \,|\, \boldsymbol{x})$.




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

# 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**[24]
  - **Probabilistic classifier chains**[25]
  - **Hierarchical softmax**[26]
  - **Homer**[27]
  - **Nested dichotomies**[28]
  - **Multi-stage classification**[29]

---

[24] 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

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

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

[27] 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

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

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

# FastXML vs. PLT

| | FastXML | PLT |
|---|---|---|
| tree structure | ✓ | ✓ |
| structure learning | ✓ | $\times$ |
| number of trees | $\geq 1$ | $1$ |
| number of leaves | linear in # examples | $m$ |
| 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.[30]

[30] `http://research.microsoft.com/en-us/um/people/manik/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 |

## Summary and Take-away message

# New challenges

- Reduction of extreme classification to structured output prediction (log-time and log-space algorithms).
- Extreme zero-shot learning.
- Diverse predictions and performance measures.

## Do we search in the right place?

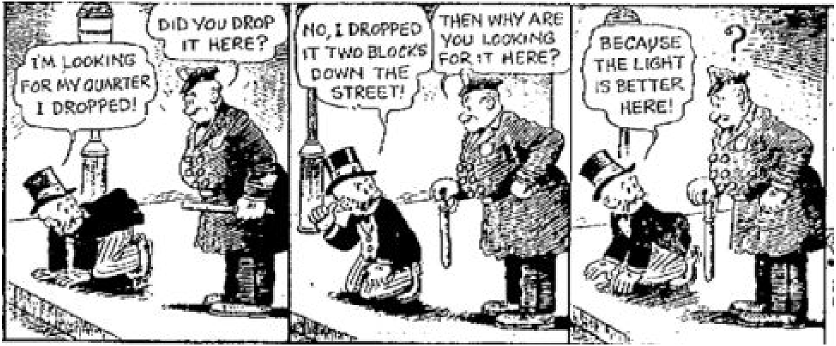



Figure: [31] A similar comics has been earlier used by Asela Gunawardana.[32]

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

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

# Conclusions

- **Take-away message**:

# Conclusions

- **Take-away message**:
  - **Extreme classification**: #examples, #features, **#labels**

# Conclusions

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

# Conclusions

- **Take-away message**:
  - **Extreme classification**: #examples, #features, **#labels**
  - **Complexity**: time vs. space, training vs. validation vs. **prediction**
  - **Statistical challenges**: Is learning possible in the extreme setting?

# Conclusions

- **Take-away message**:
  - **Extreme classification**: #examples, #features, **#labels**
  - **Complexity**: time vs. space, training vs. validation vs. **prediction**
  - **Statistical challenges**: Is learning possible in the extreme setting?
  - **Computational challenges**: compression, hashing/clustering, tree-based structures.

# Conclusions

- **Take-away message**:
  - **Extreme classification**: #examples, #features, **#labels**
  - **Complexity**: time vs. space, training vs. validation vs. **prediction**
  - **Statistical challenges**: Is learning possible in the extreme setting?
  - **Computational challenges**: compression, hashing/clustering, tree-based structures.
- For more check:

# Conclusions

- **Take-away message**:
  - **Extreme classification**: #examples, #features, **#labels**
  - **Complexity**: time vs. space, training vs. validation vs. **prediction**
  - **Statistical challenges**: Is learning possible in the extreme setting?
  - **Computational challenges**: compression, hashing/clustering, tree-based structures.
- For more check:
  - `http://www.cs.put.poznan.pl/kdembczynski`

# Conclusions

- **Take-away message**:
  - **Extreme classification**: #examples, #features, **#labels**
  - **Complexity**: time vs. space, training vs. validation vs. **prediction**
  - **Statistical challenges**: Is learning possible in the extreme setting?
  - **Computational challenges**: compression, hashing/clustering, tree-based structures.
- For more check:
  - `http://www.cs.put.poznan.pl/kdembczynski`
  - **Code**: `https://github.com/busarobi/XMLC`