M2 Data & Knowledge: Data Stream Mining

Jesse Read

Classification in Data Streams

Outline

Classification

We want a model h, which can take inputs in $\mathcal{X}$ and provide a suitable output in $\mathcal{Y}$ (under some suitable loss metric).

$\mathbf{x} =$ 

Binary classification

$$\mathcal{Y} = \{\texttt{non\_beach}, \texttt{beach}\}$$
$$\hat{y} = h(\mathbf{x}), \quad \text{where } \hat{y} \in \mathcal{Y}$$

e.g., $\hat{y} = \texttt{beach}$.

Classification

We want a model h, which can take inputs in $\mathcal{X}$ and provide a suitable output in $\mathcal{Y}$ (under some suitable loss metric).

$\mathbf{x} =$ 

Multi-*class* classification

$$\mathcal{Y} = \{\texttt{beach}, \texttt{people}, \texttt{foliage}, \texttt{sunset}, \texttt{urban}\}$$

$$\hat{y} = h(\mathbf{x}), \quad \text{where } \hat{y} \in \mathcal{Y}$$

e.g., $\hat{y} = \texttt{beach}$.

Classification

We want a model h, which can take inputs in $\mathcal{X}$ and provide a suitable output in $\mathcal{Y}$ (under some suitable loss metric).

$\mathbf{x} =$ 

Multi-label classification

$$\mathcal{Y} = \{\texttt{beach}, \texttt{people}, \texttt{foliage}, \texttt{sunset}, \texttt{urban}\}$$

$$\hat{y} = h(\mathbf{x}), \quad \text{where } \hat{y} \subseteq \mathcal{Y}$$

e.g., $\hat{y} = \{\texttt{beach}, \texttt{foliage}\} \Leftrightarrow \hat{\mathbf{y}} = [1, 0, 1, 0, 0]$ where $\hat{\mathbf{y}} \in \{0, 1\}^5$.
i.e., multiple labels per instance.

Single-label vs. Multi-label Data

Single-label Problem $Y \in \{0, 1\}$

X_1	X_2	X_3	X_4	X_5	Y
1	0.1	3	A	NO	0
0	0.9	1	C	YES	1
0	0.0	1	A	NO	0
1	0.8	2	B	YES	1
1	0.0	2	B	YES	0
0	0.0	3	A	YES	?

Multi-label Problem $Y \subseteq \{\lambda_1, \ldots, \lambda_L\}$

X_1	X_2	X_3	X_4	X_5	Y
1	0.1	3	A	NO	$\{\lambda_2, \lambda_3\}$
0	0.9	1	C	YES	$\{\lambda_1\}$
0	0.0	1	A	NO	$\{\lambda_2\}$
1	0.8	2	B	YES	$\{\lambda_1, \lambda_4\}$
1	0.0	2	B	YES	$\{\lambda_4\}$
0	0.0	3	A	YES	?

Single-label vs. Multi-label Data

Single-label Problem $Y \in \{0, 1\}$

X_1	X_2	X_3	X_4	X_5	Y
1	0.1	3	A	NO	0
0	0.9	1	C	YES	1
0	0.0	1	A	NO	0
1	0.8	2	B	YES	1
1	0.0	2	B	YES	0
0	0.0	3	A	YES	?

Multi-label Problem $[Y_1, \ldots, Y_L] \in \{0, 1\}^L$

X_1	X_2	X_3	X_4	X_5	Y_1	Y_2	Y_3	Y_4
1	0.1	3	A	NO	0	1	1	0
0	0.9	1	C	YES	1	0	0	0
0	0.0	1	A	NO	0	1	0	0
1	0.8	2	B	YES	1	0	0	1
1	0.0	2	B	YES	0	0	0	1
0	0.0	3	A	YES	?	?	?	?

Multi-output Data

We can generalized to multi-label multi-class (multi-output) classification:

X_1	X_2	X_3	X_4	X_5	type	gender	group
x_1	x_2	x_3	x_4	x_5	1	M	2
x_1	x_2	x_3	x_4	x_5	4	F	2
x_1	x_2	x_3	x_4	x_5	2	?	1
x_1	x_2	x_3	x_4	x_5	3	M	1
x_1	x_2	x_3	x_4	x_5	?	?	?

Multi-output Data

We can generalized to multi-label multi-class (multi-output) classification:

X_1	X_2	X_3	X_4	X_5	type	gender	group
x_1	x_2	x_3	x_4	x_5	1	M	2
x_1	x_2	x_3	x_4	x_5	4	F	2
x_1	x_2	x_3	x_4	x_5	2	?	1
x_1	x_2	x_3	x_4	x_5	3	M	1
x_1	x_2	x_3	x_4	x_5	?	?	?

Multi-output regression:

X_1	X_2	X_3	X_4	X_5	amount	age	percent
x_1	x_2	x_3	x_4	x_5	37.00	25	0.88
x_1	x_2	x_3	x_4	x_5	-22.88	22	0.22
x_1	x_2	x_3	x_4	x_5	19.21	12	0.25
x_1	x_2	x_3	x_4	x_5	88.23	11	0.77
x_1	x_2	x_3	x_4	x_5	?	?	?

Applications

Text Categorization, . . .

Email, Bookmarks, References, Encyclopedia, . . .

Applications

Labelling Images . . .





e.g., associated with concept labels,

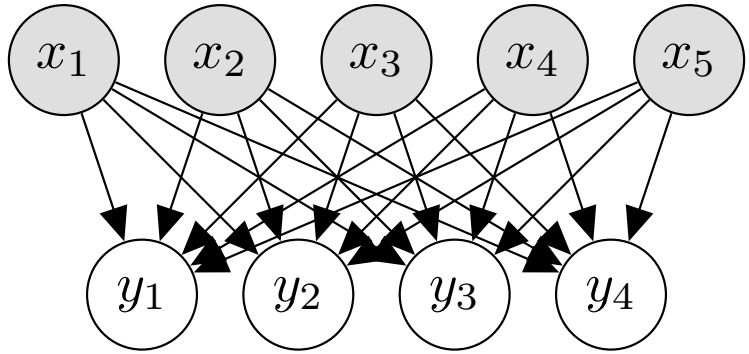
$\subseteq \{\texttt{beach}, \texttt{sunset}, \texttt{foliage}, \texttt{field}, \texttt{mountain}, \texttt{urban}\}$

Applications

Labelling Audio . . .

e.g., associated with emotions, concepts, genres;
$\subseteq \{$ `amazed-surprised`, `happy-pleased`, `relaxing-calm`, `quiet-still`, `sad-lonely`, `angry-aggressive` $\}$

Individual Classifiers



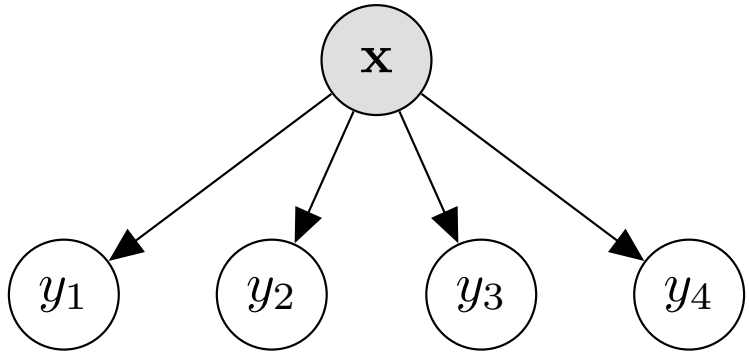
$$\hat{y}_j = h_j(\mathbf{x}) = \underset{y_j \in \{0,1\}}{\operatorname{argmax}} P(y_j|\mathbf{x}) \quad \triangleright \text{ for } j = 1, \ldots, L$$

and

$$\begin{aligned}\hat{\mathbf{y}} = \mathbf{h}(\mathbf{x}) &= [\hat{y}_1, \ldots, \hat{y}_4] \\ &= \left[\underset{y_1 \in \{0,1\}}{\operatorname{argmax}} P(y_1|\mathbf{x}), \cdots, \underset{y_4 \in \{0,1\}}{\operatorname{argmax}} P(y_4|\mathbf{x})\right] \\ &= \left[h_1(\mathbf{x}), \cdots, h_4(\mathbf{x})\right]\end{aligned}$$

Also known as the binary relevance method (BR) when $y_j \in \{0, 1\}$.

Individual Classifiers



$$\hat{y}_j = h_j(\mathbf{x}) = \underset{y_j \in \{0,1\}}{\operatorname{argmax}} P(y_j|\mathbf{x}) \quad \triangleright \text{ for } j = 1, \ldots, L$$

and

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{h}(\mathbf{x}) = [\hat{y}_1, \ldots, \hat{y}_4] \\ &= \left[\underset{y_1 \in \{0,1\}}{\operatorname{argmax}} P(y_1|\mathbf{x}), \cdots, \underset{y_4 \in \{0,1\}}{\operatorname{argmax}} P(y_4|\mathbf{x}) \right] \\ &= \left[h_1(\mathbf{x}), \cdots, h_4(\mathbf{x}) \right]\end{aligned}$$

Also known as the binary relevance method (BR) when $y_j \in \{0, 1\}$.

A simple transformation . . .

$\boldsymbol{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

. . . into L separate binary problems (one for each label)

$\boldsymbol{X}$	Y_1
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	1
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	0

$\boldsymbol{X}$	Y_2
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	1
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

$\boldsymbol{X}$	Y_3
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

$\boldsymbol{X}$	Y_4
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	1

- Training: Build L base classifiers $h_1, \ldots, h_L$.
- Prediction: Each classifier provides $\hat{y}_j = h_j(\mathbf{x})$

Why Not Independent Classifiers?

Label Dependence

$$P(\mathbf{y}|\mathbf{x}) \neq \prod_{j=1}^{L} P(y_j|\mathbf{x})$$

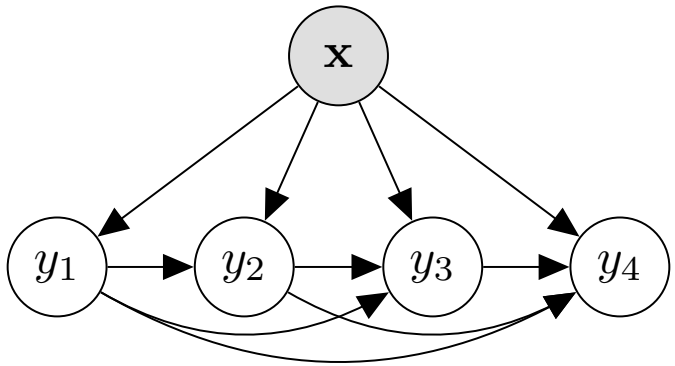
- Dependence is almost always an appropriate assumption
- Usually loss function is non-decomposable across labels, e.g., 0/1 loss (exact match), Jaccard index, rank loss, ...

Table: Exact Match (5 fold CV) Binary Relevance vs 'MCC'.

	L	BR	MCC
Music	6	0.30	**0.37**
Scene	6	0.54	**0.68**
Yeast	14	0.14	**0.23**
Genbase	27	0.94	**0.96**
Medical	45	0.58	**0.62**
Enron	53	0.07	**0.09**
Reuters	101	0.29	**0.37**

Classifier Chains

Classifier Chains, ; for modelling label dependence,



$$P(\mathbf{y}|\mathbf{x}) = P(y_1|\mathbf{x}) \prod_{j=2}^{L} P(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})$$

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \{0,1\}^L}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{x})$$

Make L binary problems (as in the binary relevance approach), but include previous predictions as feature attributes:

$\mathbf{X}$	Y_1
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	1
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	0

$\mathbf{X}$	Y_1	Y_2
$\mathbf{x}^{(1)}$	0	1
$\mathbf{x}^{(2)}$	1	0
$\mathbf{x}^{(3)}$	0	1
$\mathbf{x}^{(4)}$	1	0
$\mathbf{x}^{(5)}$	0	0

$\mathbf{X}$	Y_1	Y_2	Y_3
$\mathbf{x}^{(1)}$	0	1	1
$\mathbf{x}^{(2)}$	1	0	0
$\mathbf{x}^{(3)}$	0	1	0
$\mathbf{x}^{(4)}$	1	0	0
$\mathbf{x}^{(5)}$	0	0	0

$\mathbf{X}$	Y_1	Y_3	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

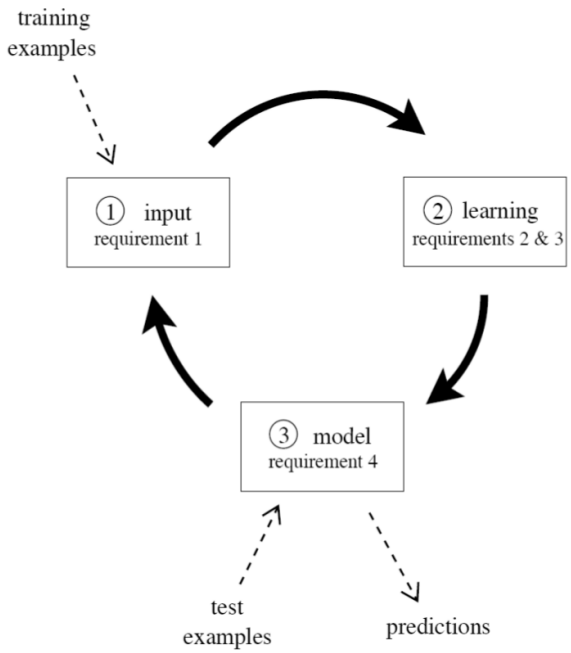
- Training: Build L base classifiers $h_1, \ldots, h_L$.
- Prediction: Each classifier provides $\hat{y}_j = h_j(\mathbf{x}, y_1, \ldots, y_{j-1})$
  Greedy inference: go down the chain once:

$$\hat{y}_j = h_j(\mathbf{x}, \hat{y}_1, \ldots, \hat{y}_{j-1})$$

Outline

Learning in Data Streams

- Sequence is potentially infinite
  $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_\infty$
- Need label for $\mathbf{x}_t$ now (before $\mathbf{x}_{t+1}$)
- Stream is one-way
- Expect high speed of arrival



At $t = 2$:

1	0.1	3	A	NO	1
0	0.9	1	C	YES	?

In a data stream, we assume that data arrives i.d.[1]

$$(\mathbf{x}_t, y_t) \sim p_t(\mathcal{X}, \mathcal{Y})$$

over time $t = 1, \ldots, \infty$ (p_t is the generating process). A model is given test instance $\mathbf{x}_t$ and is required to make a prediction at time t:

$$\hat{y}_t = h_t(\mathbf{x}_t)$$

- The computational time (training + testing) spent per instance must be less that the rate of arrival of new instances (i.e., the real clock time between time steps $t-1$ and t).
- A usual assumption: true label y_{t-1} available at time t (can use to update the model)
- Compare to standard batch setting, a dataset $\{\mathbf{x}_t, y_t\}_{t=1}^{N} \sim p(\mathcal{X}, \mathcal{Y})$, of fixed N, *fully observed prior to inducing the model.*

[1] Independently but not identically distributed

Why Classification for Streams?

- Why classify in a stream?
- Why learn in a stream?

Why Classification for Streams?

- Why classify in a stream?
- Why learn in a stream?

- Predictions required before all data has arrived
- Storage limitations (too much data to store)
- and/or random access expensive
- Concept drift

Examples

- Document classification
- Demand (and supply) prediction
- Call records
- Financial markets
- Anomaly and Intrusion detection
- Pollution and environmental monitoring
- Detection in sensor networks
- Click streams
- Online advertising and shopping
- News feeds
- ...
- Any dataset too large to load into memory

Demand Prediction

Outputs (labels) represent the demand at multiple points.

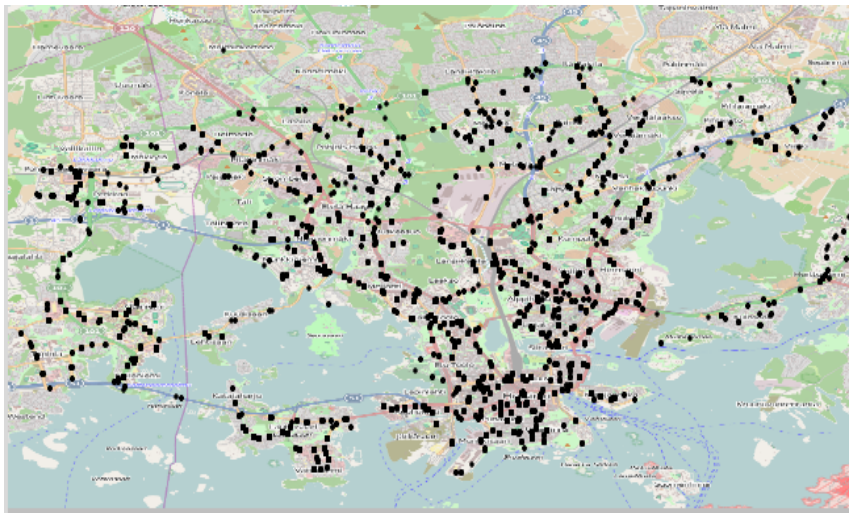


Figure: Stops in the greater Helsinki region.

Each stop is a label: We have L stops, wish to predict stops that will experience high demand.

Localization and Tracking

Outputs represent points in space which may contain an object ($y_j = 1$) or not ($y_j = 0$). Sensor readings $\mathbf{x}$.

Figure: Modelled on a real-world scenario; a room with a single light source and a number of light sensors.

We are interested in predicting, for each label $[y_1, \ldots, y_L]$,

$y_j = 1$ • if j-th tile occupied

Localization and Tracking

Outputs represent points in space which may contain an object ($y_j = 1$) or not ($y_j = 0$). Sensor readings $\mathbf{x}$.

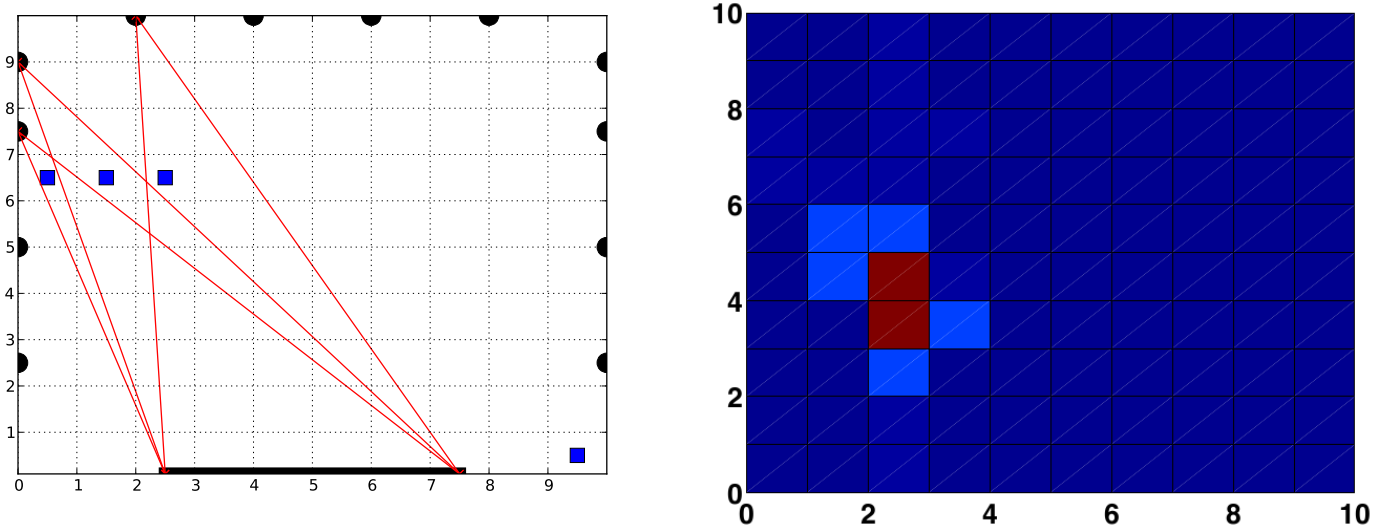


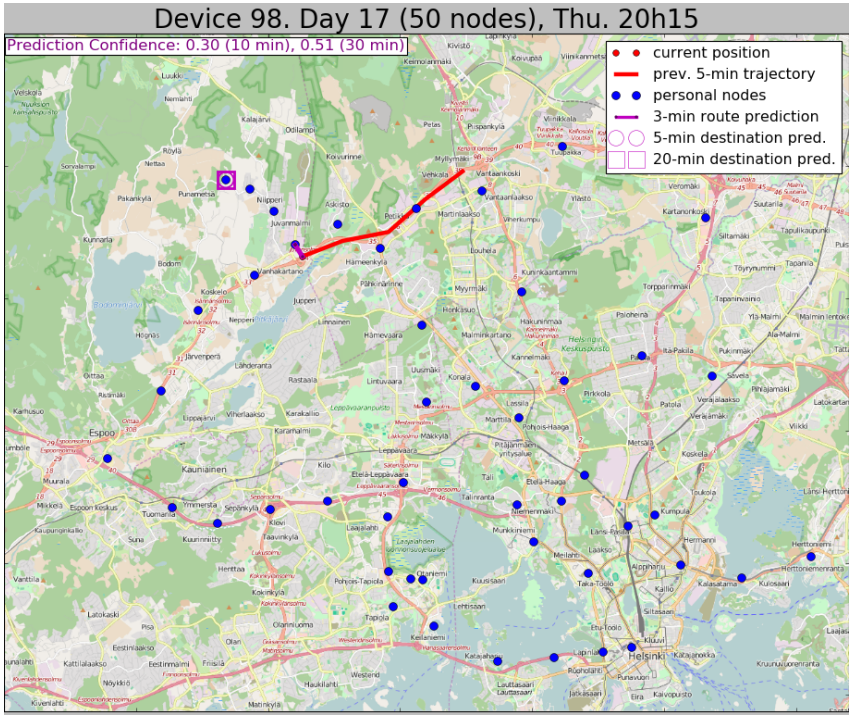
Figure: Modelled on a real-world scenario; a room with a single light source and a number of light sensors.

We are interested in predicting, for each label $[y_1, \ldots, y_L]$,

$y_j = 1$ • if j-th tile occupied

Route/Destination Forecasting

Personal nodes of a traveller and predicted trajectory



- Many gigabytes of information per traveller,
- Arrives in the form of a stream
- Model frequently needs to be updated

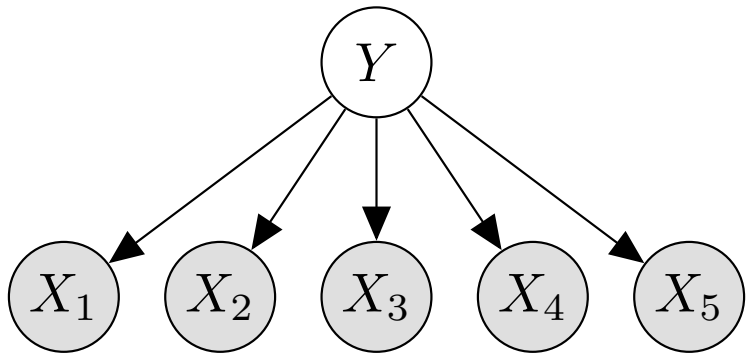
'Related' tasks: popularity prediction; predictive maintenance of aircraft.

Recap (Implications of Data Streams):

- Prediction of y_t must be made immediately (at time t)
- Update the model efficiently with training examples $(y_1, \mathbf{x}_1), (y_2, \mathbf{x}_2), \ldots$
  – limited time and memory (on a potentially infinite stream)
- Expect concept drift to occur
  - Feature attribute set $\boldsymbol{X}$ may evolve over time
  - Existing distribution $p_t(\mathcal{X}, \mathcal{Y})$ may change over time

Outline

Naive Bayes

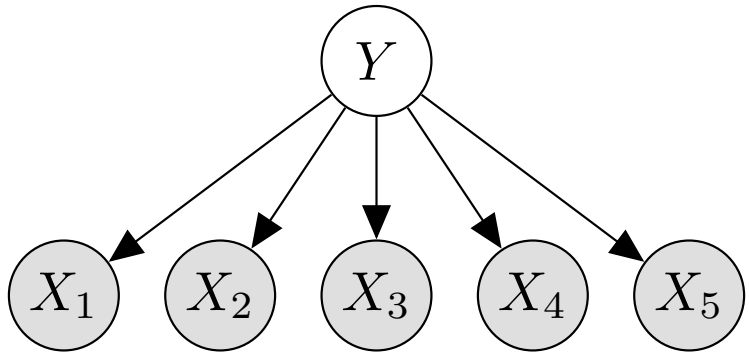


$$P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{P(\mathbf{x})}$$ • Bayes rule

where likelihood

$$P(\mathbf{x}|y) = \prod_{d=1}^{D} P(x_d|y)$$

Naive Bayes



$$\hat{y} = h(\mathbf{x}) = \underset{y \in \{0,1\}}{\operatorname{argmax}} P(y)P(\mathbf{x}|y) \quad \bullet \; P(y|\mathbf{x}) \propto P(\mathbf{x}|y)P(y)$$

$$= \underset{y \in \{0,1\}}{\operatorname{argmax}} P(y) \prod_{d=1}^{D} P(x_d|y)$$

- Estimate Ps from counting
- Adaptation to streams? Counts can be updated!

t	X_1	X_2	X_3	Y
1	1	0	1	1
2	1	0	0	1
3	1	1	1	0
4	1	1	0	0
5	0	0	1	0
6	0	0	0	0
7	0	1	1	?

t	X_1	X_2	X_3	Y
1	1	0	1	1
2	1	0	0	1
3	1	1	1	0
4	1	1	0	0
5	0	0	1	0
6	0	0	0	0
7	0	1	1	?

Models specified by $\boldsymbol{\theta}_t$:

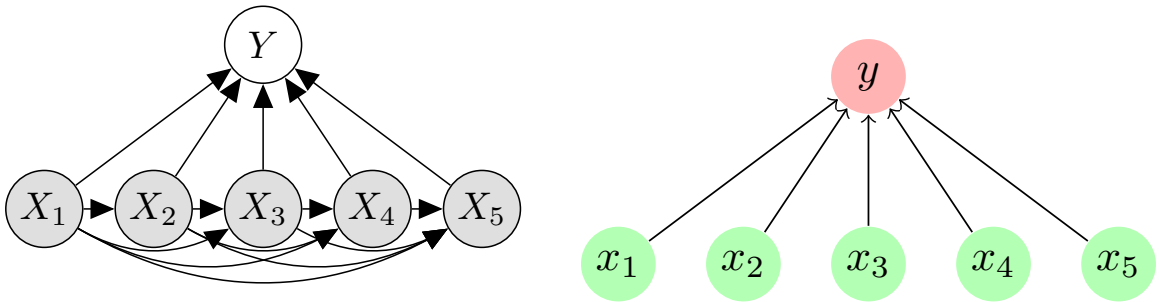
$P_t(X_1\|Y)$	$P_t(X_2\|Y)$	$P_t(X_3\|Y)$

$P_t(Y)$

- Naive Bayes can perform well
  (especially in text; has been very popular with spam filtering),
- but the independence assumption is harmful in many applications, and
- probabilities can be unrealistic
- It is often used as a baseline method.

Outline

Logistic Regression



$$p(y = 1|\mathbf{x}) \approx \sigma(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^\top \mathbf{x})}$$

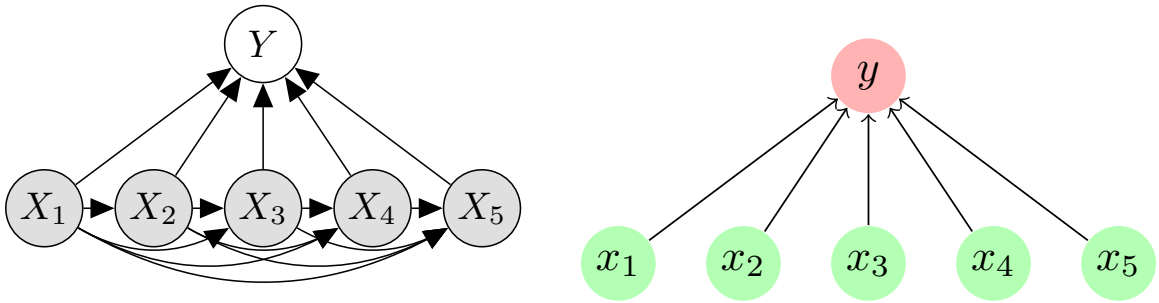
We take the gradient of the negative log likelihood (error)

$$E_t(\boldsymbol{\theta}) = -\log \prod_{i=1}^{N} \left[\sigma_i^{y_i} \cdot (1 - \sigma_i)^{1-y_i}\right]$$

wrt $\boldsymbol{\theta}$; giving $\nabla E_t(\boldsymbol{\theta})$, and we use it to update the weights $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha \nabla E_t(\boldsymbol{\theta}_t)$$

Logistic Regression



- Adaptation to streams? Update in one example at a time; i.e., (stochastic/*incremental*) gradient descent (SGD).

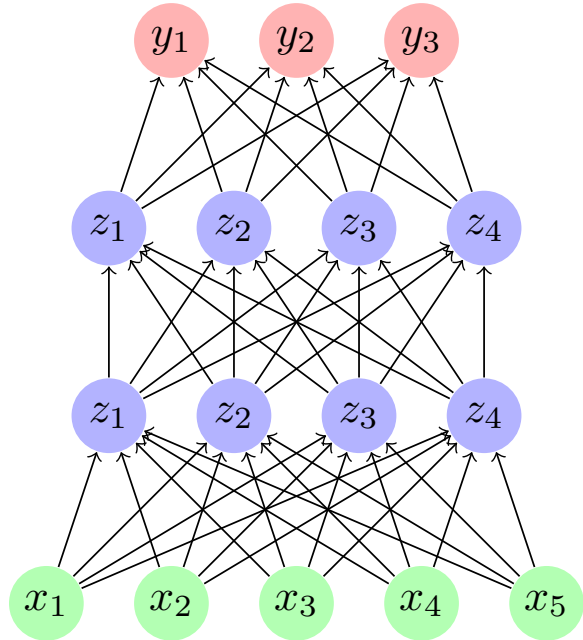
$$\nabla E_t(\boldsymbol{\theta}) = (\sigma_t - y_t)\mathbf{x}_t$$

$$\hat{y} = h(\mathbf{x}) = \underset{y \in \{0,1\}}{\operatorname{argmax}}\, p(y|\mathbf{x}) \quad \bullet \text{ classification}$$

$$p(y = 1|\mathbf{x}) = p(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^\top \mathbf{x})}$$

$$p(y = 0|\mathbf{x}) = 1 - p(\mathbf{x}; \boldsymbol{\theta})$$

Neural Networks and Deep Learning



- Simply back-propagate the weight-updates down the net

However

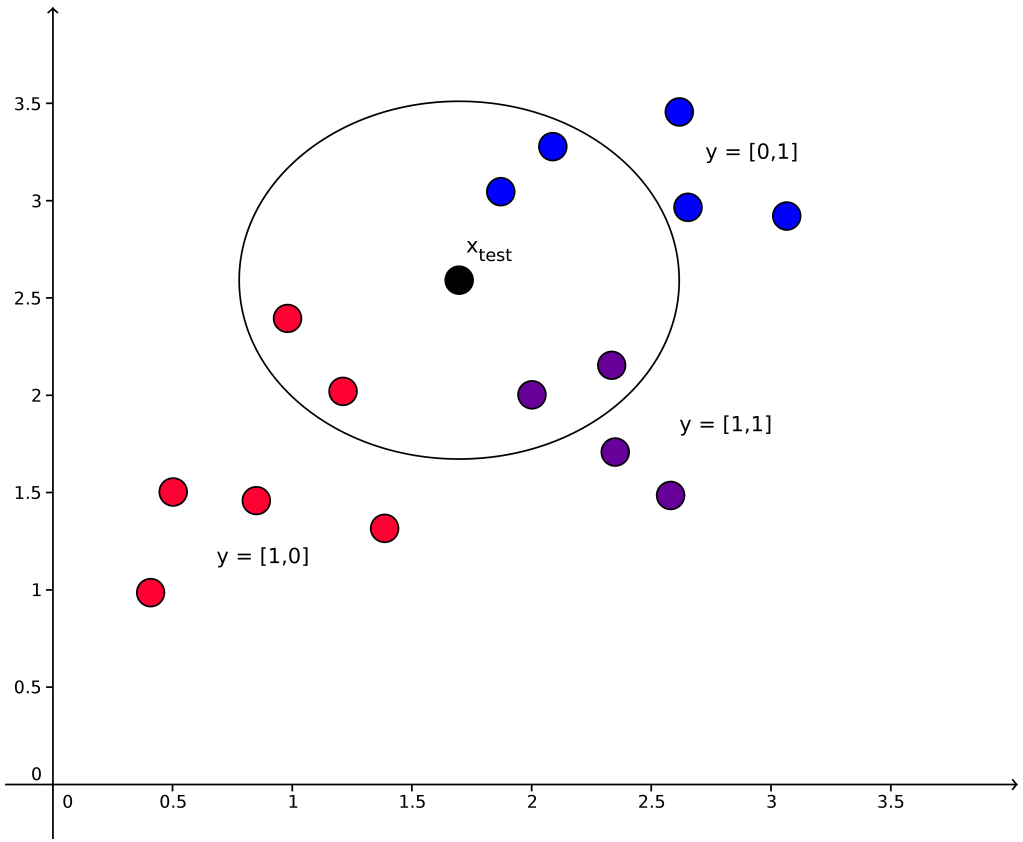
- Often need many training iterations for good performance
- Hyper-parameter tuning: how many nodes, layers (deep learning), learning rate, weight penalty, convolutions etc?
- Empirical comparisons in data streams show that SGD/MLPs struggle against other methods (in predictive performance).

Outline

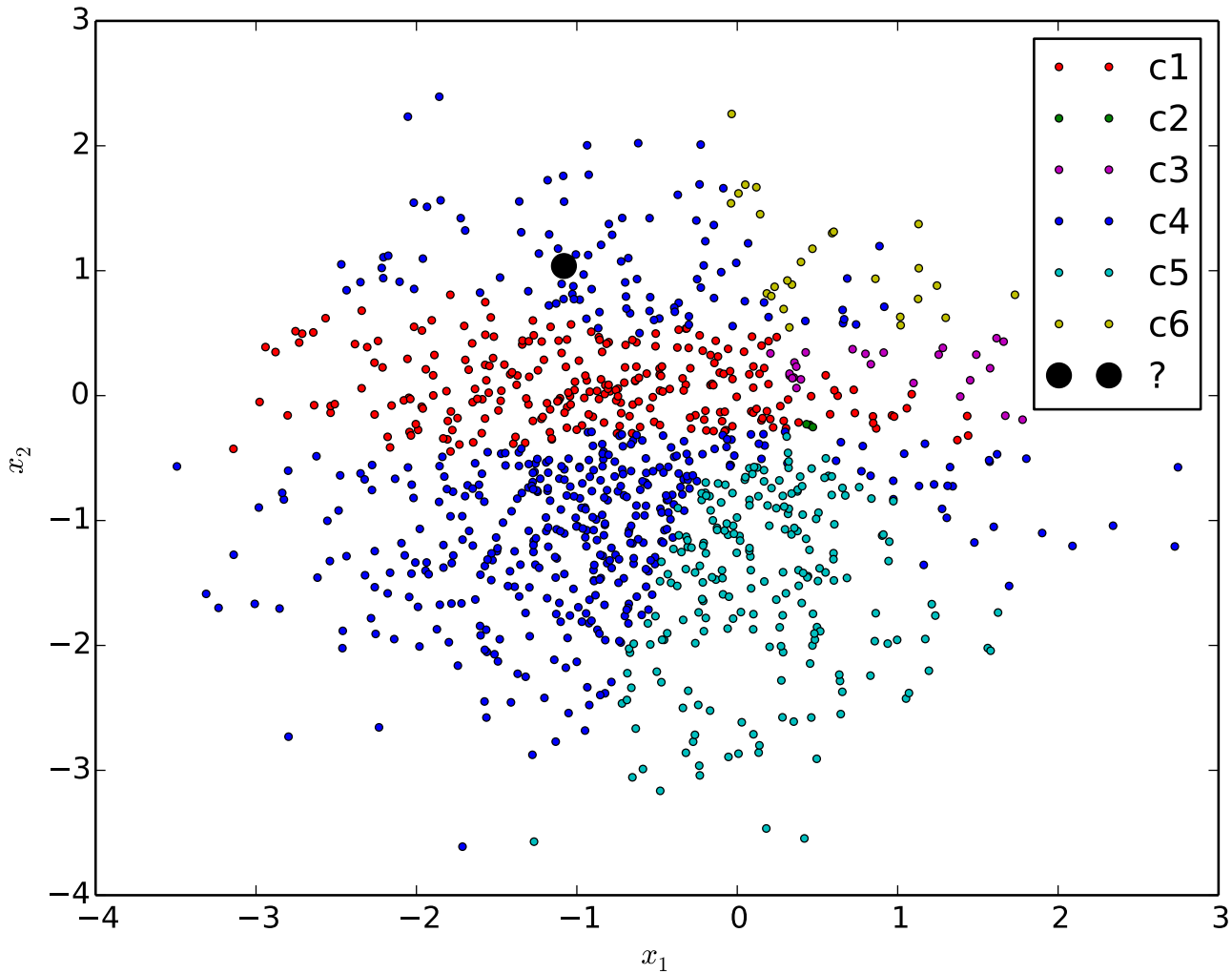
k-Nearest Neighbours (kNN)

$$p(y = 1|\mathbf{x}) \approx \frac{1}{k} \sum_{\mathbf{x}_i \in \mathrm{Ne}_k(\mathbf{x})} y_i$$

$$\hat{y} = \begin{cases} 1 & p(y = 1|\mathbf{x}) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

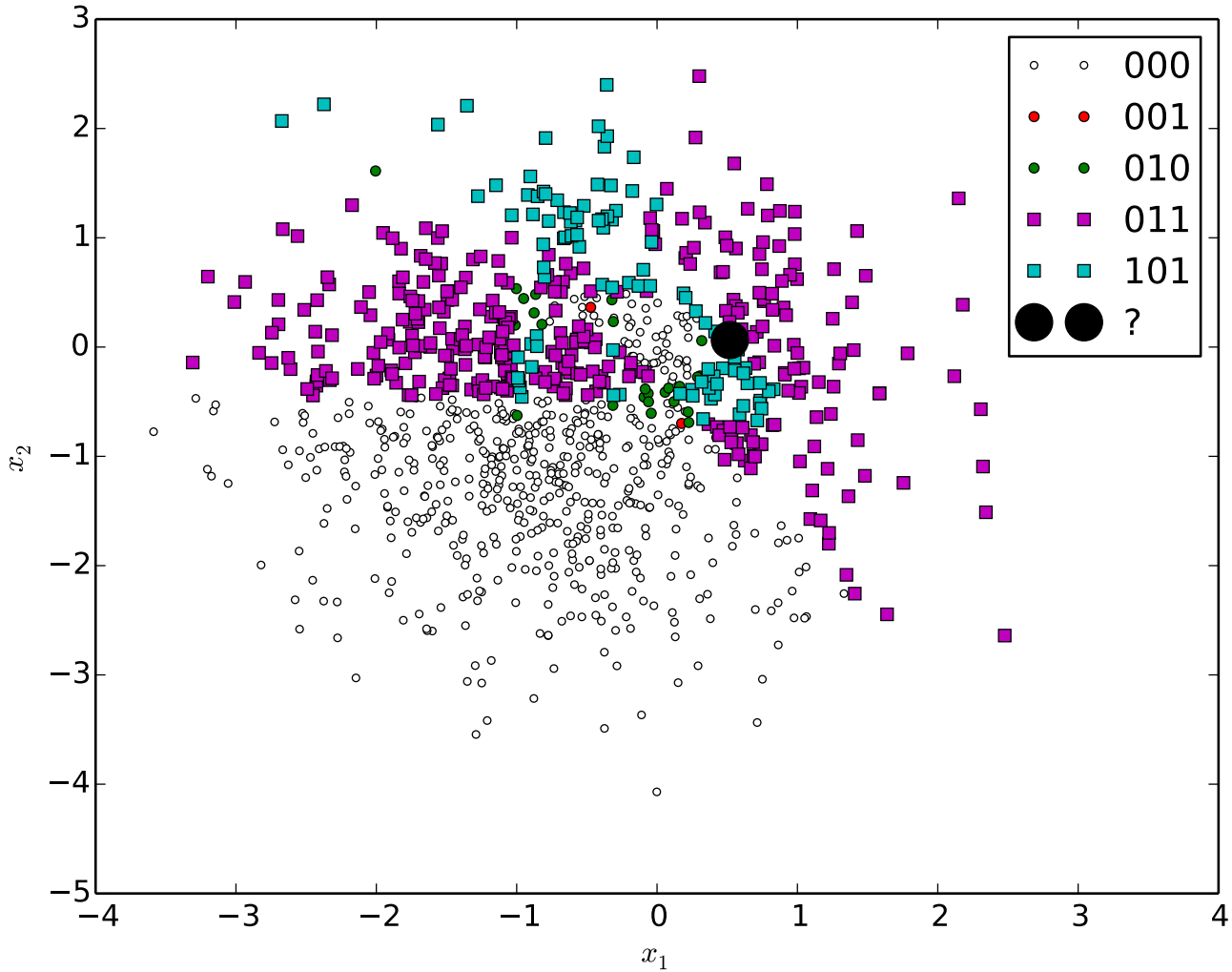


where $\mathrm{Ne}_k(\mathbf{x})$ contains the *k-closest* points to $\mathbf{x}$ (e.g., Euclidean distance) in the dataset.



Multi-label kNN

Adaptation to multi-label: simply consider $\hat{y}_j$ (per label)



kNN's adaptation to streams: A dynamic buffer / sliding window of N instances, e.g.,

$$\mathbf{X}_t = \{\mathbf{x}_{t-1000}, \mathbf{x}_{t-999}, \ldots, \mathbf{x}_{t-1}\}$$

where $N = 1000$.

- A non-linear and dynamic decision boundary; can be very effective in practice
- Learned *concept* is limited wrt N,
- Time complexity at t:

  $$O(NDK)$$

  (in the multi-label context, efficient wrt $L \equiv$ number of labels)
- Different windowing techniques can be used (we will see soon)
- Not suitable for all problems (esp. if high dimensionality)

Outline

Rules

For example,

$$\underbrace{X_4 > 1, X_2 \leq 0}_{\text{rule}} \mapsto \underbrace{[1, 0, 1, 0, 0]}_{\mathbf{y}}$$

Adaptation to streams? Can expand rules with the Hoeffding bound. For a variable (e.g., $G \in [0, R]$): The true mean G is *not* within

$$\epsilon = \sqrt{\frac{R^2 \ln\left(1/\delta\right)}{2n}}$$

with probability $1 - \delta$; over n examples.

Rules

For example,

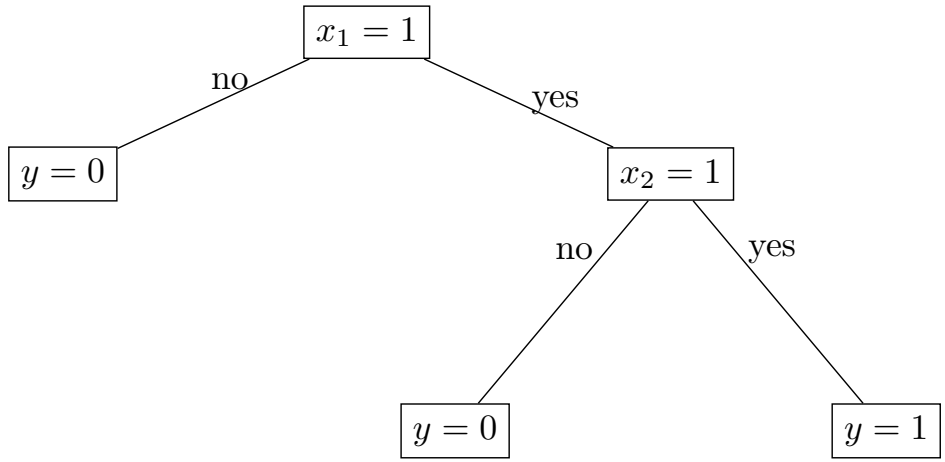
$$\underbrace{X_4 > 1, X_2 \leq 0}_{\text{rule}} \mapsto \underbrace{[1, 0, 1, 0, 0]}_{\mathbf{y}}$$

Adaptation to streams? Can expand rules with the Hoeffding bound. For a variable (e.g., $G \in [0, R]$): The true mean G is *not* within

$$\epsilon = \sqrt{\frac{R^2 \ln\left(1/\delta\right)}{2n}}$$

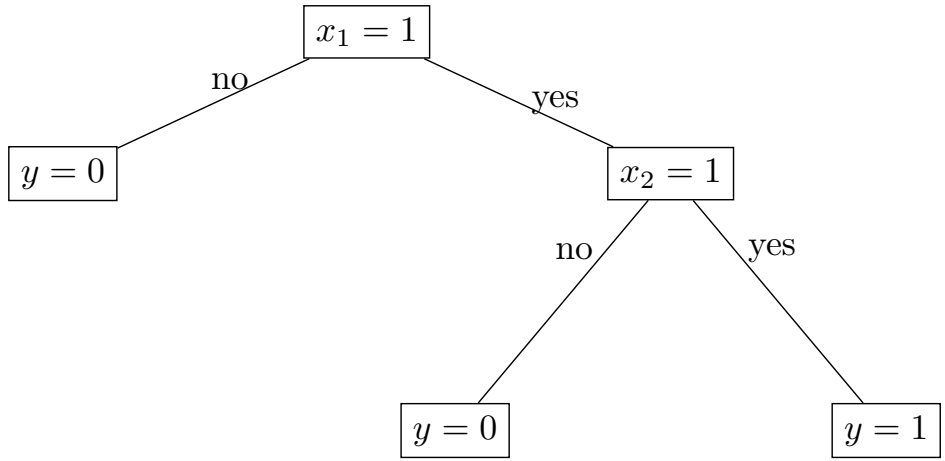
with probability $1 - \delta$; over n examples. This gives us an indication of when to expand a rule.

Decision Trees



X_1	X_2	X_3	Y
1	0	1	1
1	0	0	1
1	1	1	0
1	1	0	0
0	0	1	0
0	0	0	0
0	1	1	?

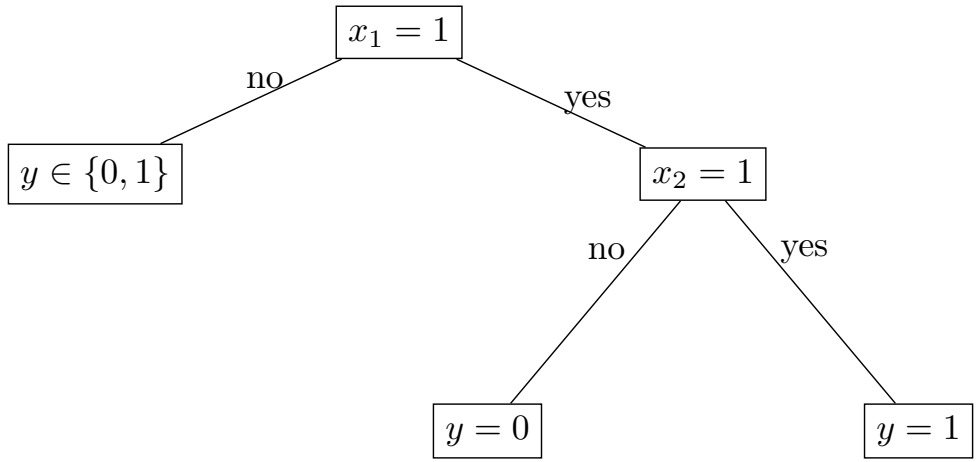
Decision Trees



Induce($\mathcal{S}$) returns Tree T:

1. if stop($\mathcal{S}$), return Leaf($\mathcal{S}$)
2. best $=$ argmax$_{j \in \{1,\ldots,D\}}$ $G(\mathcal{S}, X_j)$
3. Create decision node that tests x_{best}
4. For $v \in \{0, 1\}$:
   - $\mathcal{S}_v = \{(\mathbf{x}_{\neg \text{best}}, y) | x_{\text{best}} = v\}$
     - (Induce sub-datasets $\mathcal{S}_v$ based on $(x_{\text{best}} = v)$)
   - $T_v = \text{Induce}(\mathcal{S}_v)$
   - attach T_v to the corresponding branch
5. return Tree (root x_{best}, branches on T_0, T_1)

Decision Trees



$$H(\mathcal{S}) = -\sum_{y \in \{0,1\}} \Big(P(Y = y) \log_2 P(Y = y) \Big) \quad \bullet \text{ entropy}$$

$$G(\mathcal{S}, X) = H(\mathcal{S}) - \sum_{v \in \mathcal{X}} \frac{|\mathcal{S}_v|}{|\mathcal{S}|} \cdot H(\mathcal{S}_v) \quad \bullet \text{ information gain}$$

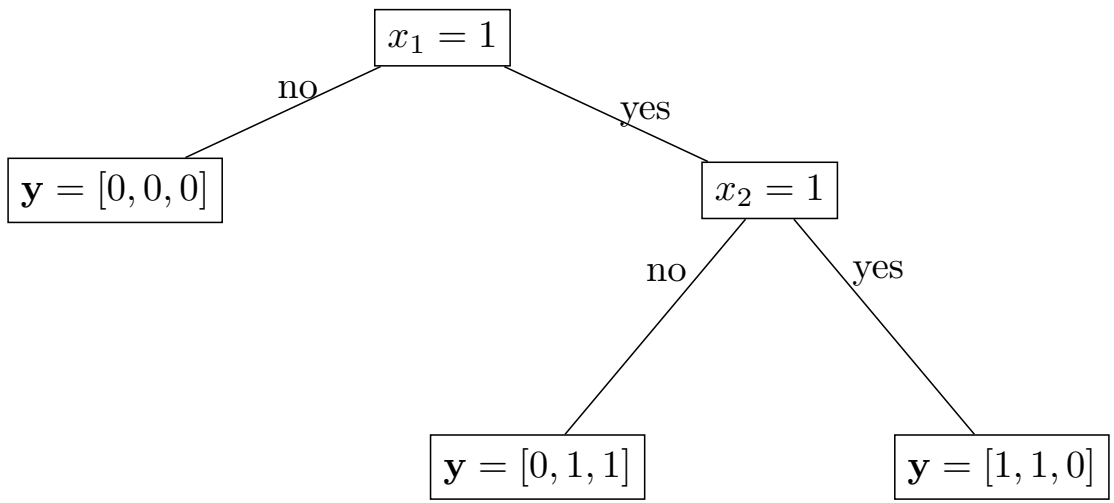
$$= H(\mathcal{S}) - \sum_{v \in \mathcal{X}} \frac{|\{(\mathbf{x}, y) \in \mathcal{S} | x = v\}|}{|\mathcal{S}|} \cdot H(\{(\mathbf{x}, y) \in \mathcal{S} | x = v\})$$

Multi-label Decision Trees

Using multi-label entropy,

$$H_{\mathsf{ML}}(\mathcal{S}) = -\sum_{j=1}^{L} \sum_{y \in \{0,1\}} P(Y_j = y) \log_2 P(Y_j = y)$$

$$= -\sum_{j=1}^{L} \Big[P(y_j) \log_2 P(y_j) + [1 - P(y_j)] \log_2 [1 - P(y_j)] \Big]$$



Note that now, $(\mathbf{y}, \mathbf{x}) \in \mathcal{S}$, i.e., multi-label vector assignments $\mathbf{y}$ are filtering down to the leaves.

Hoeffding [Incremental] Decision Trees

Hoeffding Tree Induction Algorithm:

1. Sort $(\mathbf{x}_t, y_t)$ into leaf ℓ
2. Update *sufficient statistics* (necessary to calculate $\bar{G}$) in ℓ
3. Increment n_ℓ (number of examples seen at ℓ)
4. If $n_\ell \mod n_{\min} = 0$:
   1. Compute $\bar{G}(X_j)$ for all attributes j
   2. X_a is the attribute with highest $\bar{G}$ and X_b second highest
   3. Compute $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n_\ell}}$
   4. if $\left[\bar{G}(X_a) - \bar{G}(X_b)\right] > \epsilon$ (wrt prop. $1 - \delta$),
      1. turn leaf ℓ into node splitting on X_a
      2. create leaves $\ell_1, \ldots, \ell_k$ for all values of $X_a \in \{v_1, \ldots, v_k\}$, with initialized sufficient statistics

Hoeffding [Incremental] Decision Trees

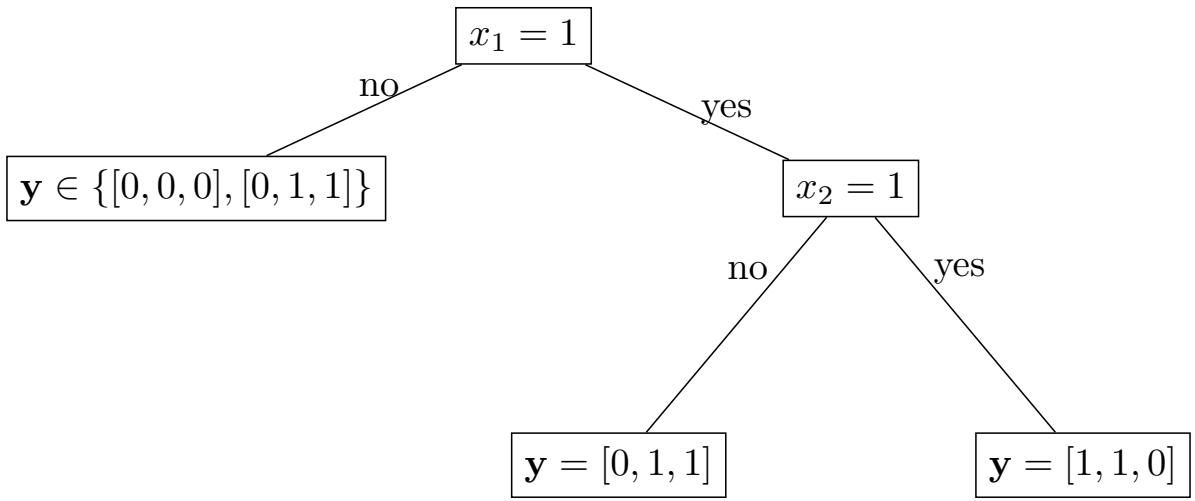
Hoeffding Tree Induction Algorithm:

1. Sort $(\mathbf{x}_t, y_t)$ into leaf ℓ
2. Update *sufficient statistics* (necessary to calculate $\bar{G}$) in ℓ
3. Increment n_ℓ (number of examples seen at ℓ)
4. If $n_\ell \mod n_{\min} = 0$:
   1. Compute $\bar{G}(X_j)$ for all attributes j
   2. X_a is the attribute with highest $\bar{G}$ and X_b second highest
   3. Compute $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n_\ell}}$
   4. if $\left[\bar{G}(X_a) - \bar{G}(X_b)\right] > \epsilon$ (wrt prop. $1-\delta$),
      1. turn leaf ℓ into node splitting on X_a
      2. create leaves $\ell_1, \ldots, \ell_k$ for all values of $X_a \in \{v_1, \ldots, v_k\}$, with initialized sufficient statistics

With high probability, constructs an identical model that a traditional (greedy) method would learn.

Multi-label Incremental Decision Trees

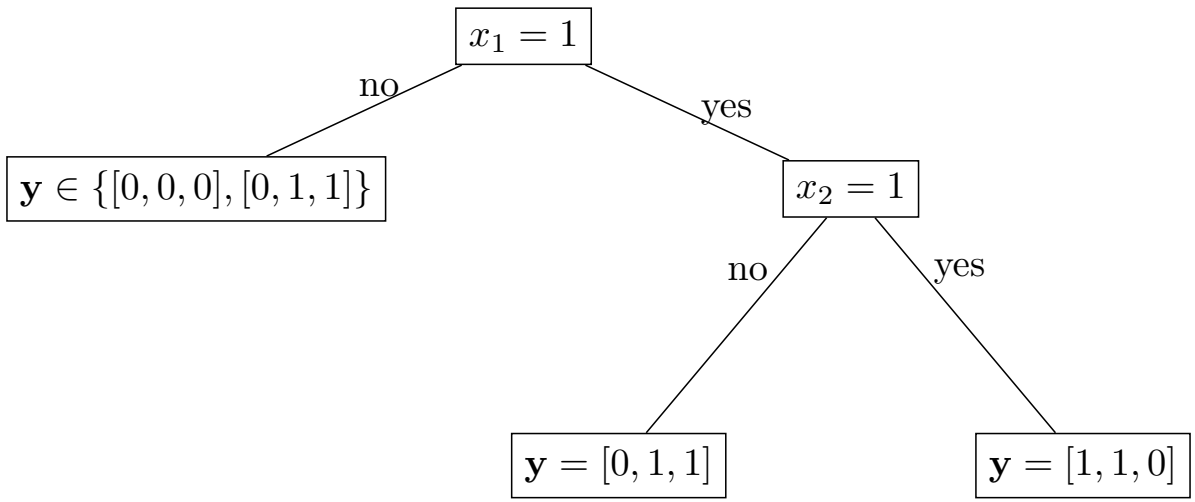
- Use Hoeffding tree algorithm with multi-label entropy
- Examples with *multiple labels* collect at the leaves (we can take the majority labelset, or merge . . .)



If $\{(\mathbf{x}_\ell, [1,1,0]), (\mathbf{x}_\ell, [1,0,0]), (\mathbf{x}_\ell, [0,1,0])\}$ at leaf ℓ, could return score $[0.67, 0.67, 0.0]$

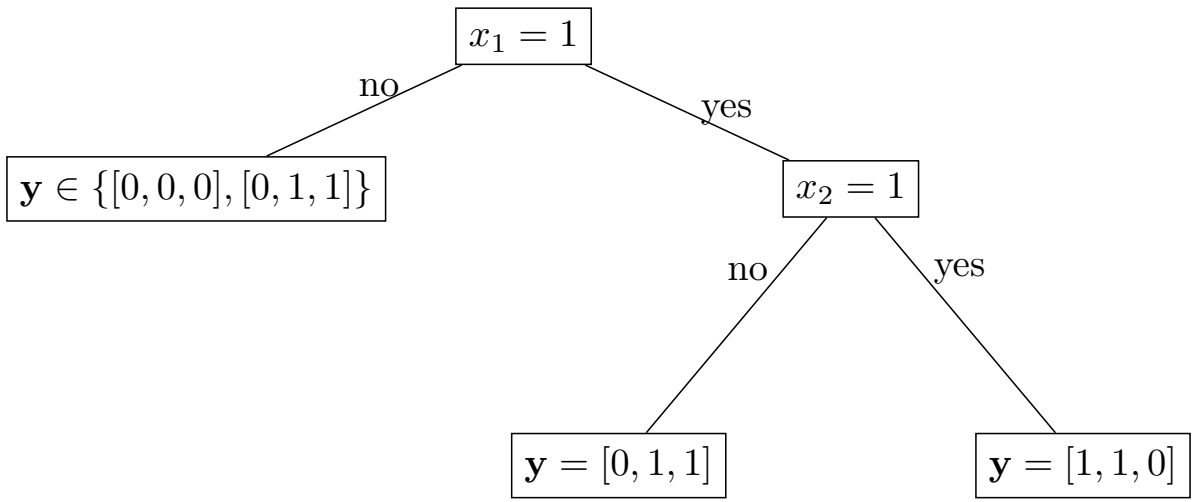
Multi-label Incremental Decision Trees

- A multi-label *incremental* model
- Very *fast* (*single* model for all labels), and usually very *competitive*,



Multi-label Incremental Decision Trees

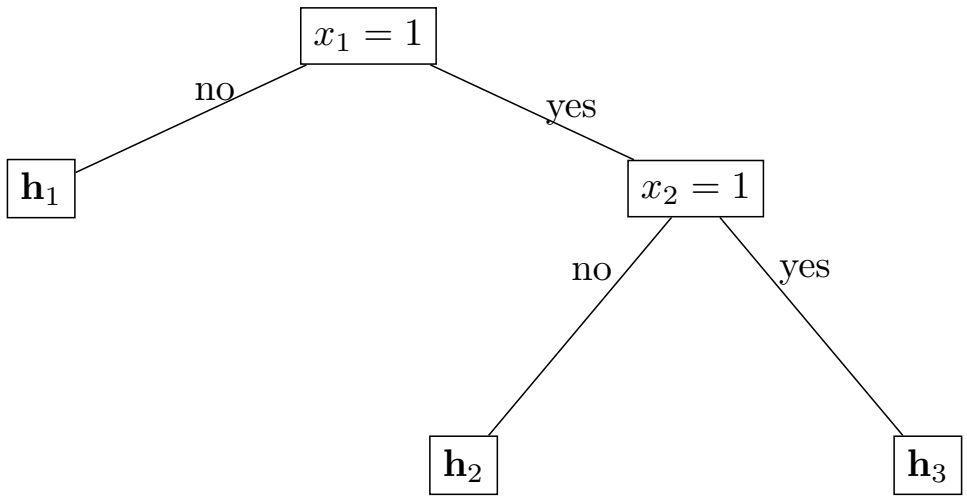
- A multi-label *incremental* model
- Very *fast* (*single* model for all labels), and usually very *competitive*,



- But Hoeffding bound is conservative, tree reluctant to grow, after many examples can remain as root node / stump / majority class classifier, . . .
- And when it does grow, it may soon become outdated by concept drift

Models at the Leaves

Place classifiers at the leaves of the tree



where each $\mathbf{h}_\ell : \mathcal{X}^{D^\ell} \to \mathcal{Y}^{L^\ell}$, e.g., a classifier dealing with a subset of the input space and a subset of the label space at each leaf ℓ (note: $D^\ell \leq D$, $L^\ell \leq L$). Naive Bayes can be a suitable choice (statistics are already available).

- Hoeffding tree: provably converges to batch tree with *sufficiently large data*,
- Decision trees powerful learners, with non-linear decision boundaries
- However Hoeffding trees are conservative, may grow slowly
  - Can use other models at the leaves
  - Can start from pre-grown tree (mix with batch-techniques)
  - Use in an ensemble
- Memory and time complexity is 'good', but not fixed like other methods

  it grows over time wrt branching factor!

Outline

Batch-Incremental / Batch Ensemble

Build regular models on batches of instances, of size N.

$$\underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_1, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_2, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_3, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_4}_{\text{build } \mathbf{h}_1}, \underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_5, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_6, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_7, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_8}_{\text{build } \mathbf{h}_2}, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_9, \begin{bmatrix}\mathbf{x}\\ \hat{\mathbf{y}}\end{bmatrix}_{10}$$

(e.g., $N = 4$)

- build model $\mathbf{h}_1$ on $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_4, \mathbf{y}_4)\}$
- build model $\mathbf{h}_2$ on $\{(\mathbf{x}_5, \mathbf{y}_5), \ldots, (\mathbf{x}_8, \mathbf{y}_8)\}$
- ...
- ensemble of M models: $\{\mathbf{h}_1, \ldots, \mathbf{h}_M\}$ (let $M = \lfloor t/N \rfloor$)
- predict via vote :

$$\hat{\mathbf{y}} = \sum_{m=1}^{M} \mathbf{h}_m(\mathbf{x})$$

Batch-Incremental / Batch Ensemble

Build regular models on batches of instances, of size N.

$$\underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_1, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_2, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_3, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_4}_{\text{build } \mathbf{h}_1}, \underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_5, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_6, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_7, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_8}_{\text{build } \mathbf{h}_2}, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_9, \begin{bmatrix}\mathbf{x}\\ \hat{\mathbf{y}}\end{bmatrix}_{10}$$

(e.g., $N = 4$)

- build model $\mathbf{h}_1$ on $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_4, \mathbf{y}_4)\}$
- build model $\mathbf{h}_2$ on $\{(\mathbf{x}_5, \mathbf{y}_5), \ldots, (\mathbf{x}_8, \mathbf{y}_8)\}$
- ...
- ensemble of M models: $\{\mathbf{h}_1, \ldots, \mathbf{h}_M\}$ (let $M = \lfloor t/N \rfloor$)
- predict via weighted vote (more recent = more relevant):

$$\hat{\mathbf{y}} = \sum_{m=1}^{M} \omega_m \cdot \mathbf{h}_m(\mathbf{x}) \quad \bullet \text{ where } \omega_m \propto m \text{ and } \sum_m \omega_m = 1$$

- Memory will run out! Unless ...
- Fix M wrt constraints (delete a model when adding a model),
- Which model to delete:
  - oldest?
  - weakest?
- What batch size (N) to use?
  - too small = models are insufficient
  - too large = slow to adapt (missing new instances!)
- Sliding window[2]: build new model every τ instances ($1 \leq \tau$)
  - Extreme case: model built *every instance* ($\tau = 1$)
  - But too many batches = too slow
- Sampling: Train m-th batch on

$$\mathbf{X}_m \sim \left\{ \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}_1, \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}_2, \ldots, \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}_t \right\}$$

  - which instances to sample?

[2] As opposed to tumbling windows

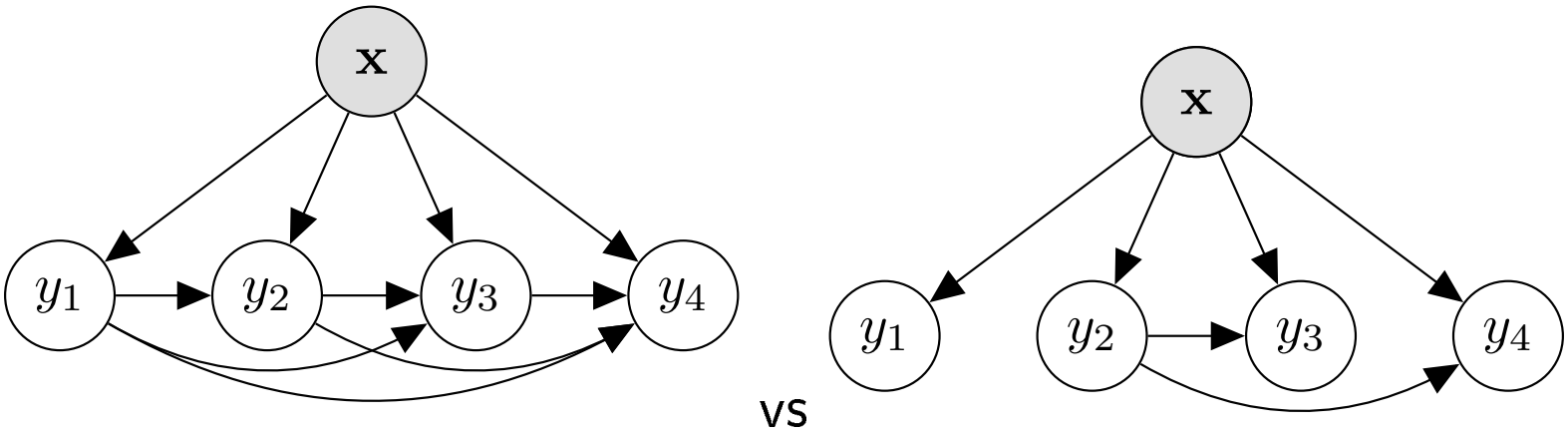
The batch-ensemble:

- A common approach to streams, often effective
- Open choice of base classifier (e.g., C4.5, SVM, ...)
- But careful consideration (of M, N, τ, etc.) needed
- Generally computationally expensive compared to other methods

Outline

Multi-label Implications

- Class imbalance (exacerbated)
- Overfitting (exacerbated)
- Multi-dimensional concept drift
- Labelled examples difficult to obtain (missing, weak, . . .)
- Dynamic label set
- Dynamic relationships among labels



$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x})$$

- A good structure is important wrt results on test data
- We need data to design it!
- Concept drift may change the distribution

Outline

Evaluating Streaming Classifiers

- Holdout
- Interleaved test-then-train
- Prequential (sliding window)
- Interleaved chunks

Summary

1. Introduction: Classification
2. Introduction: Data Streams
3. Naive Bayes
4. Neural Networks and SGD
5. k-Nearest Neighbours
6. Decision Trees
7. Batch-Ensemble Methods
8. Multi-label Considerations
9. Stream Evaluation

M2 Data & Knowledge: Data Stream Mining

Jesse Read

Classification in Data Streams