Multi-Output Chain Models and their Application in Data Streams

Jesse Read

1. Data Streams are Time Series

2. Multi-label Learning
   - Classifier Chains
   - Regressor Chains

3. Applications to Data Streams

Acknowledgements

Work with: Kostas Skiannis, Nikos Tziortziotis (*DaSciM team, École Polytechnique*), Jacob Montiel, Heitor Gomes, Albert Bifet, Rodrigo Mello (*Télécom ParisTech*), Luca Martino (*Univ. Rey Juan Carlos*), Jaakko Hollmèn (*Aalto University*), among others.

Outline

Data Streams

In a data stream,

$$(x_t, y_t) \sim P_t$$

over time $t = 1, \ldots, \infty$.

- P_t is the concept at time t.
- We observe x_t at time t.

Requirements for a model h_t in the data stream setting

- Prediction $\widehat{y}_t = h_t(x_t)$ required at real time t (immediately).
- Computational time (training + labelling) 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).

Streaming Classification

Common assumptions found in the literature

1. Speed and size of stream is fast enough to require instance-incremental learning
2. The true label y_{t-1} is available at time t, providing new training example (x_{t-1}, y_{t-1})
3. No temporal dependence
4. Concept drift will occur ($P_t \neq P_{t+1}$ for some t).

Streaming Classification

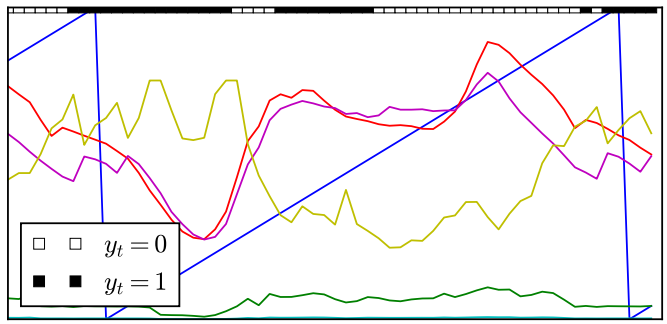
Common assumptions found in the literature

1. Speed and size of stream is fast enough to require instance-incremental learning
2. The true label y_{t-1} is available at time t, providing new training example (x_{t-1}, y_{t-1})
3. No temporal dependence
4. Concept drift will occur ($P_t \neq P_{t+1}$ for some t).

- Where do we get y_{t-1} from?
  - A human (then 1. and 2. are probably contradictory)
  - The present (i.e., the future wrt y_{t-1}) meaning it is a time series (contradicts 3.)
- Assumptions 3. and 4. are contradictory

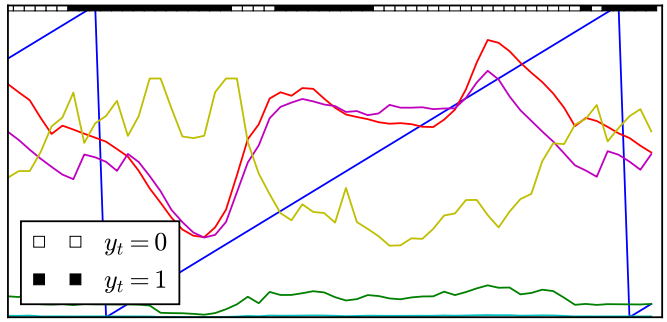
Data Streams as Time Series

Benchmark datasets often look like time series:

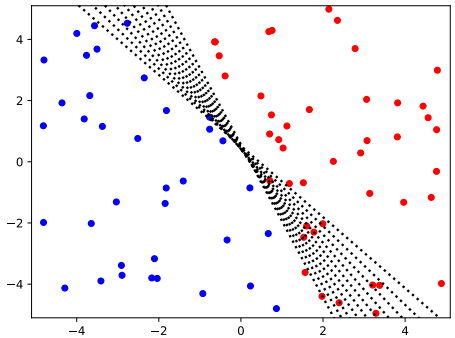


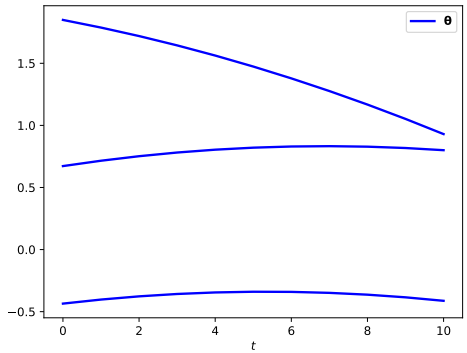
Data Streams as Time Series

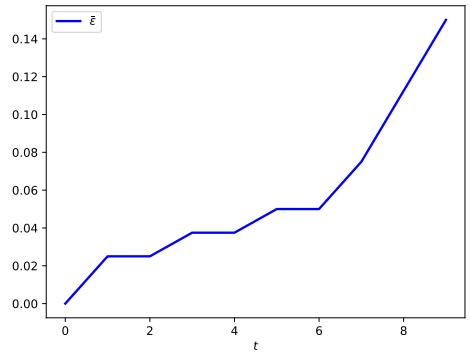
Benchmark datasets often look like time series:

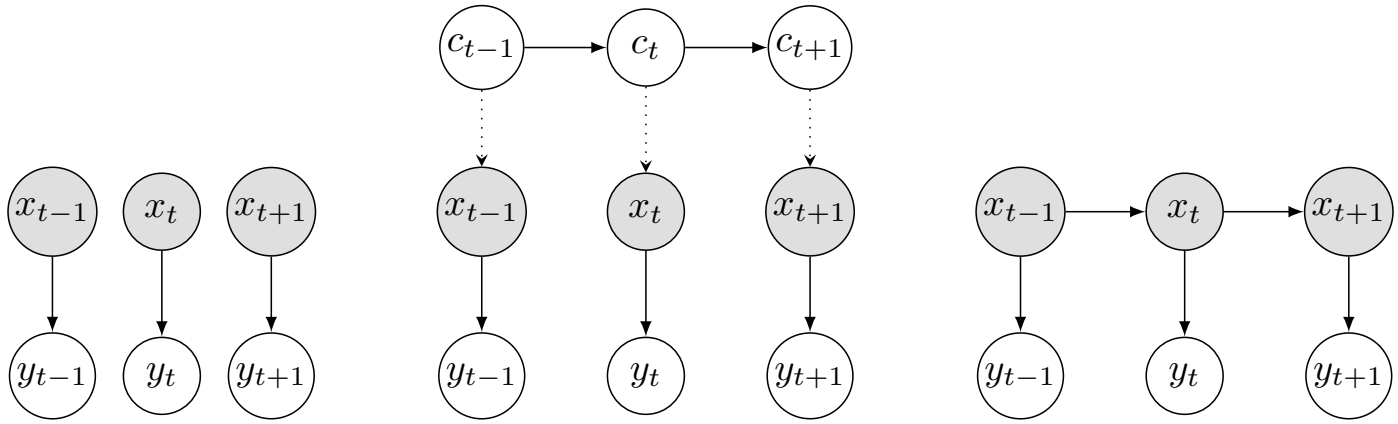


Even when points sampled iid wrt current concept, a time series forms in the coefficients, and/or in error signal:









Time series tasks:

- Filtering
- Forecasting
- State labelling/change point detection
- Event/anomaly detection
- ...

(no supervised streaming classification!)

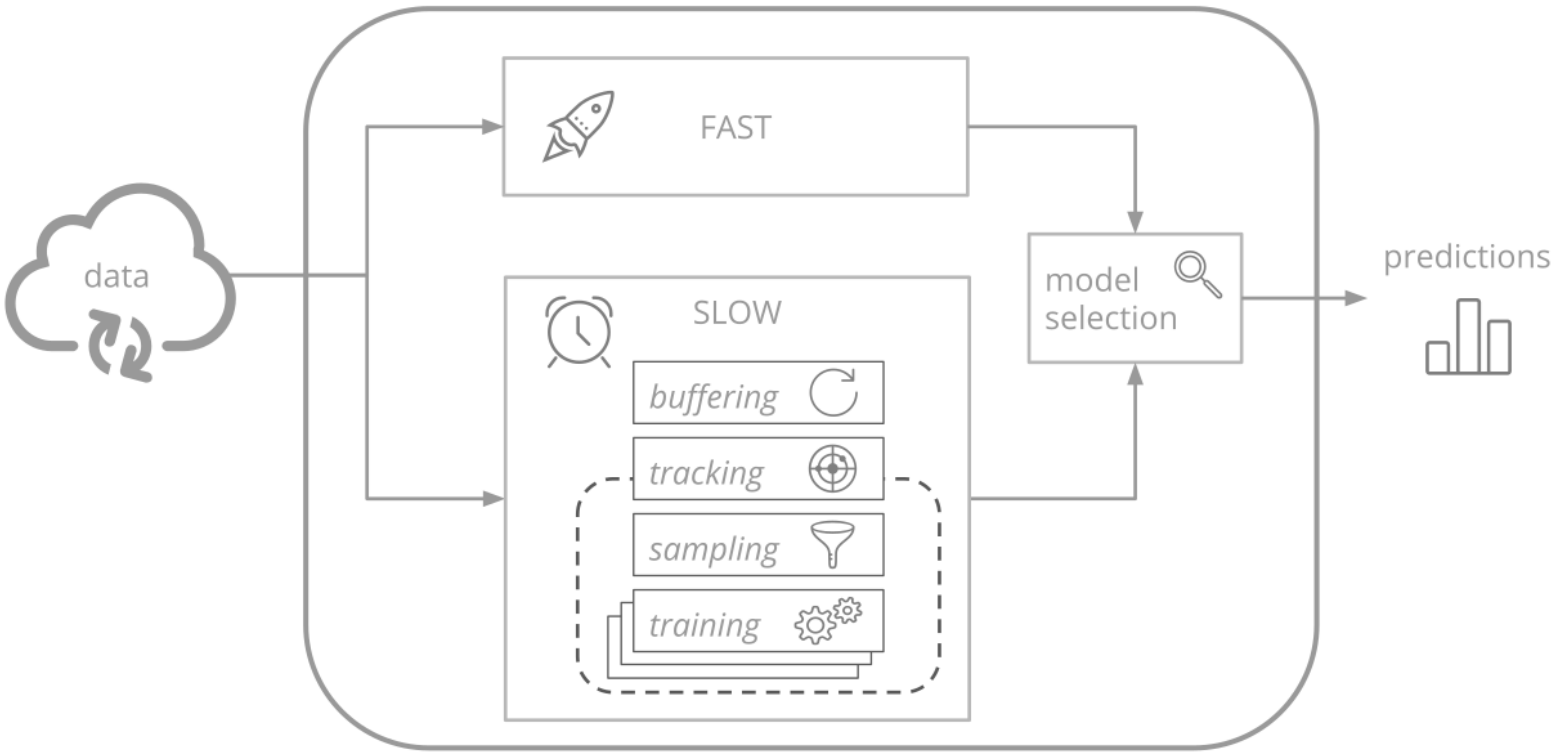
The data stream imposes constraints on the time series (prediction required *now*, finite resources for infinite data points, concept drift).

Dealing with Drift, Learning Efficiently

- We detect drift, to make iid stream. But
  - Perhaps not a single point (continuous drift, gradual drift . . .)
  - May be an anomaly.
  - . . . a change in state – this this drift?
  - When drift is confirmed – may be too late.
- Use a sliding window to make an iid stream. But
  - What size?
  - The right size can be too big.
- Recurrent neural network
  - Difficult to train in a streaming setting (especially cold start)
- The model adapts automatically
  - Particular model selection, parametrization (buffer size, learning rate, . . .) and maintenance
  - May forget useful concepts for the future
  - Instance-incremental is not always necessary, or possible (need labels). Batch-incremental models may perform well.

Fast and Slow Learning

- Batch and stream learning need not be mutually exclusive
- A framework for Fast and Slow learning.



Built into **Scikit MultiFlow** framework
`https://scikit-multiflow.github.io/`

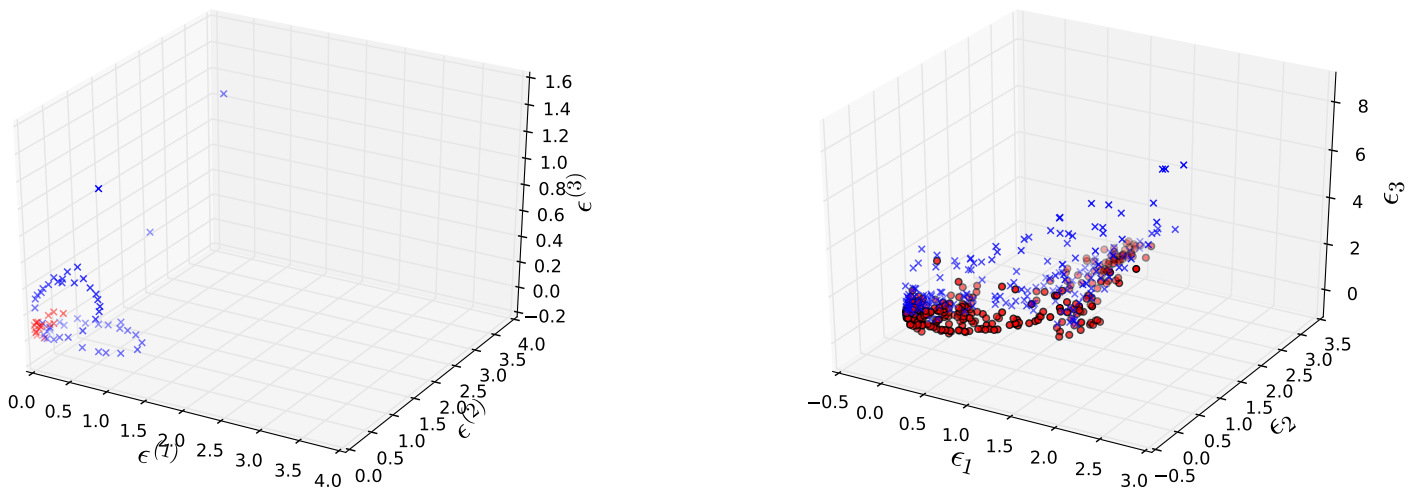
Montiel et al. 2018a

Data Streams as Time Series Problems

- Often better to model trajectory/behaviour of $\hat{\theta}_t$,

$$(x_t, y_t) \sim P_{\theta_t}$$
$$y_t = h(x_t) + \epsilon_t$$

  rather than detect drift retrospectively via ϵ_t
- Treat problem as time series forecasting (always have labels!)
- Error space is informative. In a multi-dimensional time series we require **multi-output models**.



The error of 3 one-step-ahead forecasting models (left: Synth, right: Elec), as a 3D space

Outline

Multi-label Learning

We want a model h to make predictions $\widehat{\mathbf{y}} = h(\mathbf{x})$. For example,

$\mathbf{x} =$ 

Given a set of L labels

$$\{\texttt{beach}, \texttt{people}, \texttt{foliage}, \texttt{sunset}, \texttt{urban}\}$$

our prediction is a subset:

$$\widehat{\mathbf{y}} = [1, 0, 1, 0, 0] \Leftrightarrow \{\text{beach}, \text{foliage}\}$$

i.e., multiple labels per instance, $\widehat{\mathbf{y}} \in \{0, 1\}^L$.

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 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 $[Y_1, \ldots, Y_L] \in \{0, 1\}^L$

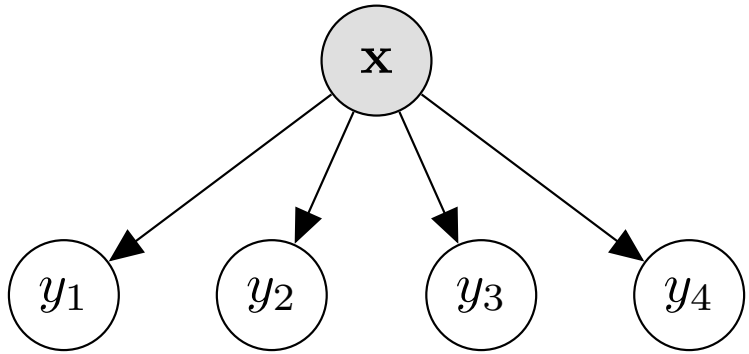
X_1	X_2	X_3	X_4	X_5	Y_1	Y_2	Y_3	Y_4
1	0.1	3	A	NO	0	1	1	0
0	0.9	1	C	YES	1	0	0	0
0	0.0	1	A	NO	0	1	0	0
1	0.8	2	B	YES	1	0	0	1
1	0.0	2	B	YES	0	0	0	1
0	0.0	3	A	YES	?	?	?	?

Applications

Titles of some multi-label papers

- *Machine learning for health informatics*
- *[. . .] Multi-label Sentiment Classification of Health Forums*
- *Using Multi-Label Classification for Improved Question Answering*
- *Predictive Skill Based Call Routing [. . .]*
- *An experimental design for identification of multiple load appliances*
- *[. . .] Methods for Prediagnosis of Cervical Cancer*
- *Probabilistic Expert Systems for Reasoning in Clinical Depressive Disorders*
- *Spectral features for audio based vehicle and engine classification*
- *Deep learning based multi-label classification for surgical tool presence detection in laparoscopic videos*
- *Ensemble-Based Location Tracking Using Passive RFID*
- *Multi-task network embedding*
- *Multi-Target Classification and Regression in Wineinformatics*

Individual Classifiers



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

and then,

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

where h_j can be any classifier $h : \mathcal{X} \to \{0, 1\}$

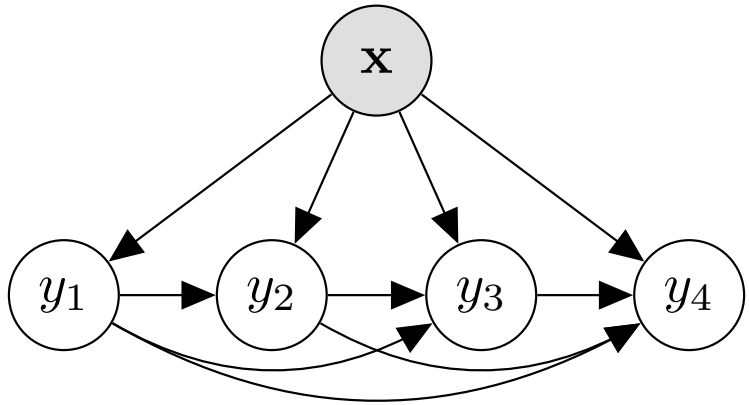
Why not Independent Classifiers?

If we model together, we can achieve

- Better predictive performance (up to 20%)
- Better computational performance (up to orders of magnitude)
- Discover interesting relationships among labels (i.e., interpretability)
- Approach structured-output prediction tasks

Outline

Classifier Chains

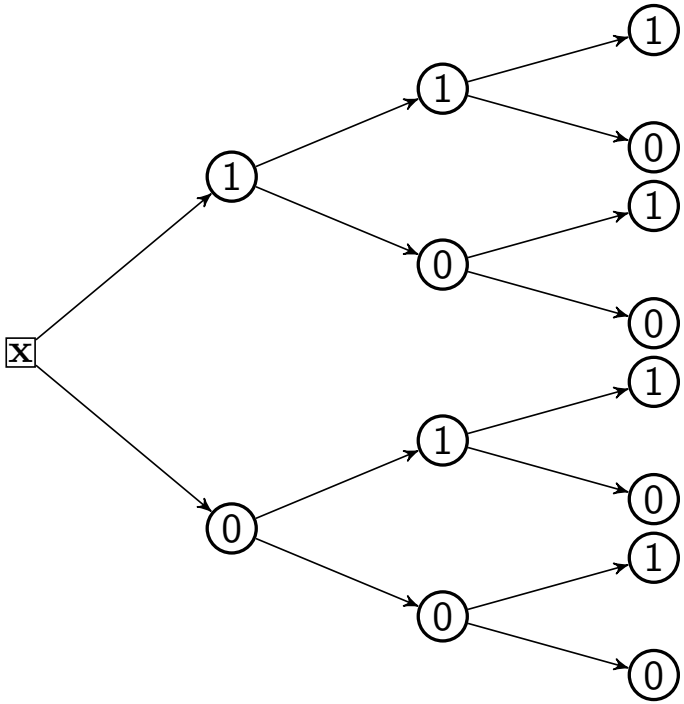


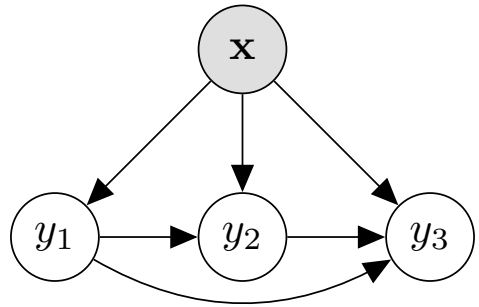
Given a query instance $\mathbf{x}$,

$$\begin{aligned}\widehat{\mathbf{y}} &= \underset{\mathbf{y}\in\{0,1\}^L}{\text{argmax}}\, P(\mathbf{y}|\mathbf{x}) \\ &= \underset{\mathbf{y}\in\{0,1\}^L}{\text{argmax}}\, P(y_1|\mathbf{x}) \prod_{j=2}^{L} P(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})\end{aligned}$$

i.e., inference becomes a search (for the best $\mathbf{y}^*$).

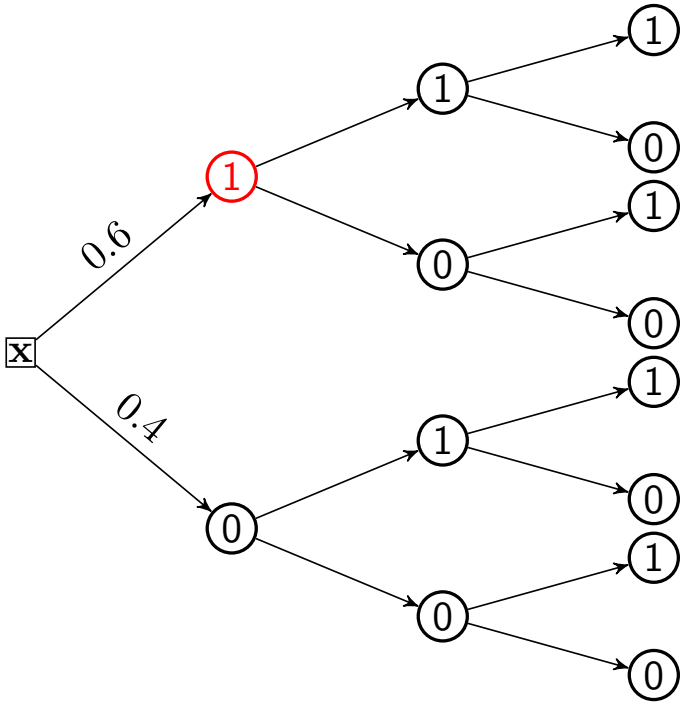
Greedy Search



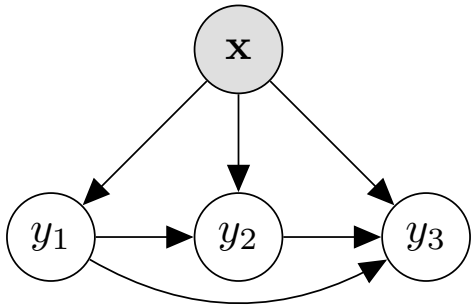


$$\widehat{\mathbf{y}} = \mathbf{h}(\mathbf{x}) = [?, ?, ?]$$

Greedy Search

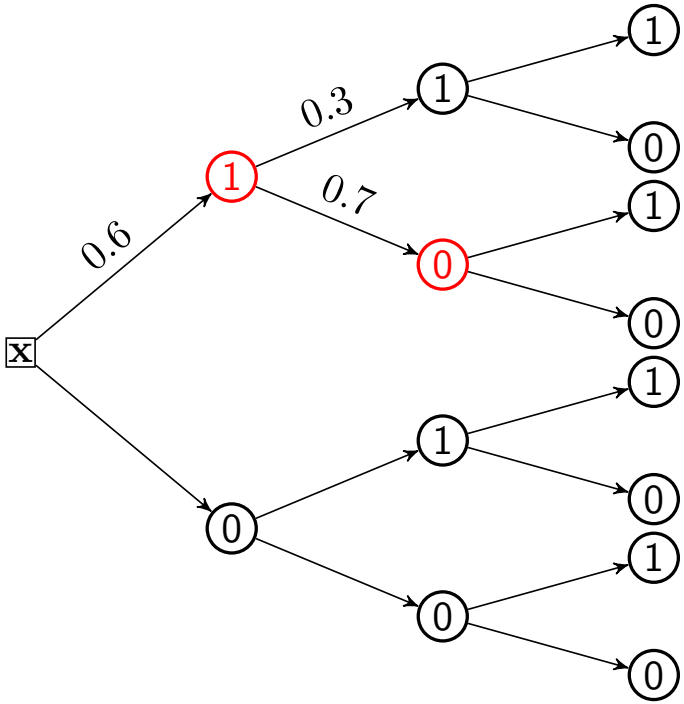


$\widehat{\mathbf{y}} = \mathbf{h}(\mathbf{x}) = [1, ?, ?]$

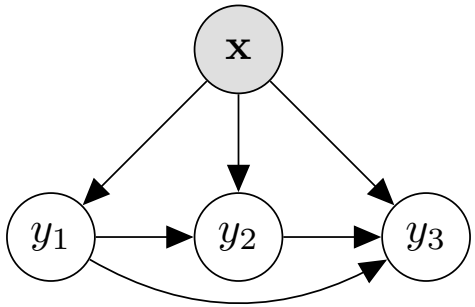


1. $\hat{y}_1 = h_1(\mathbf{x}) = \text{argmax}_{y_1} P(y_1|\mathbf{x}) = 1$

Greedy Search

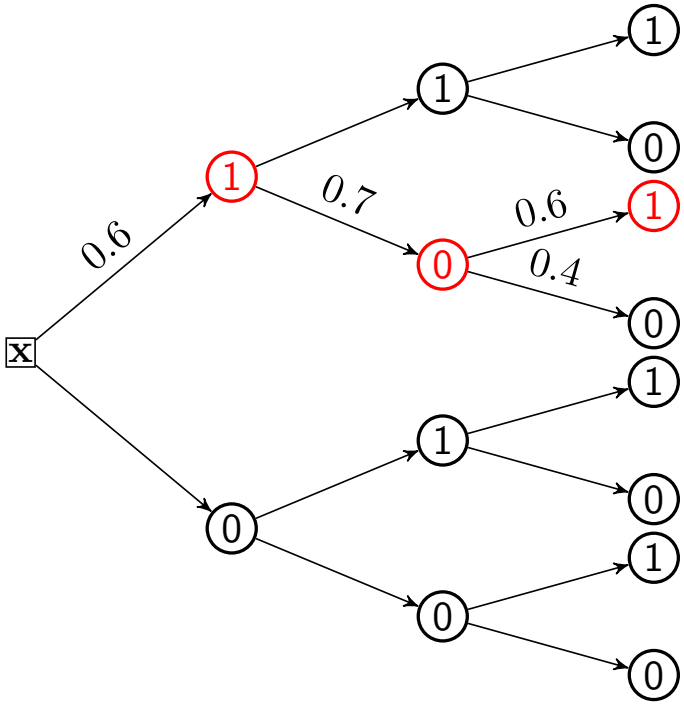


$\widehat{\mathbf{y}} = \mathbf{h}(\mathbf{x}) = [1, 0, ?]$

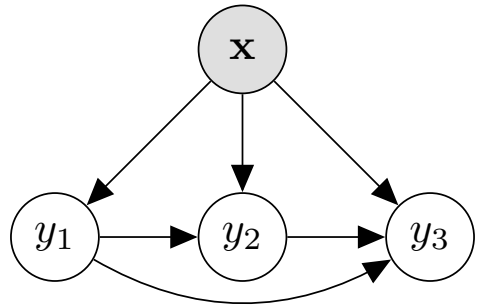


1. $\hat{y}_1 = h_1(\mathbf{x}) = \text{argmax}_{y_1} P(y_1|\mathbf{x}) = 1$
2. $\hat{y}_2 = h_2(\mathbf{x}, \hat{y}_1) = \ldots = 0$

Greedy Search

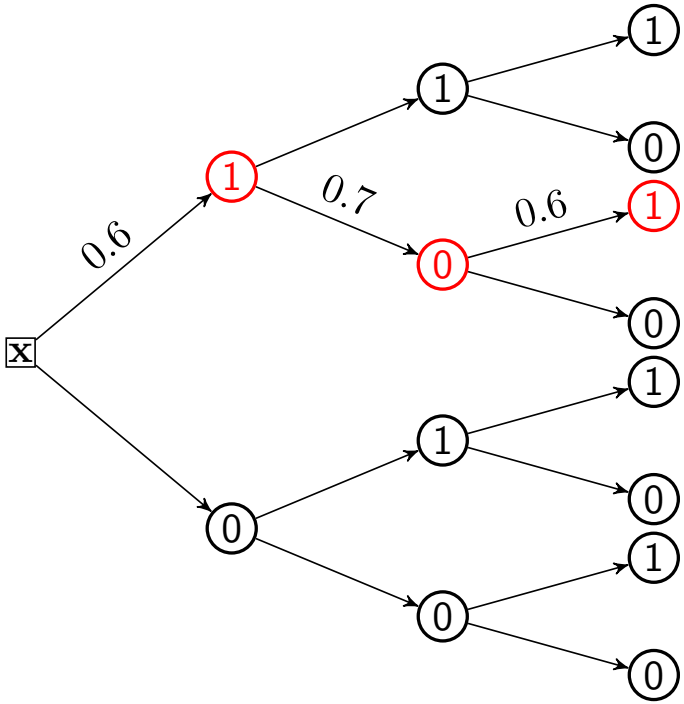


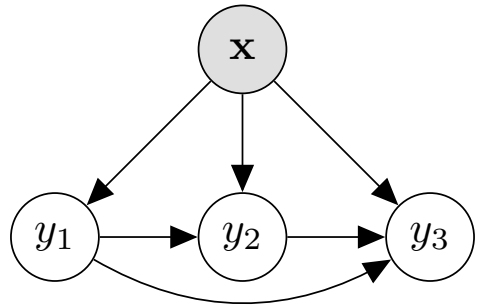
$\widehat{\mathbf{y}} = \mathbf{h}(\mathbf{x}) = [1, 0, 1]$



1. $\hat{y}_1 = h_1(\mathbf{x}) = \text{argmax}_{y_1} P(y_1|\mathbf{x}) = 1$
2. $\hat{y}_2 = h_2(\mathbf{x}, \hat{y}_1) = \ldots = 0$
3. $\hat{y}_3 = h_3(\mathbf{x}, \hat{y}_1, \hat{y}_2) = \ldots = 1$

Greedy Search





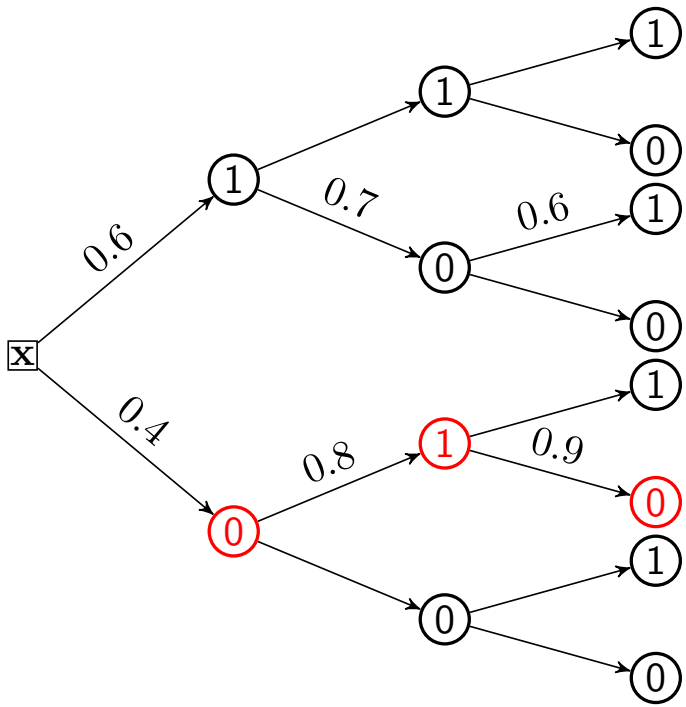
1. $\hat{y}_1 = h_1(\mathbf{x}) = \text{argmax}_{y_1} P(y_1|\mathbf{x}) = 1$
2. $\hat{y}_2 = h_2(\mathbf{x}, \hat{y}_1) = \ldots = 0$
3. $\hat{y}_3 = h_3(\mathbf{x}, \hat{y}_1, \hat{y}_2) = \ldots = 1$

$\widehat{\mathbf{y}} = \mathbf{h}(\mathbf{x}) = [1, 0, 1]$

- Again, any classifier can be used for h_j
- Fast, but susceptible to error proagation, since

$$\underset{\mathbf{y}\in\{0,1\}^L}{\text{argmax}}\, P(\mathbf{y}|\mathbf{x}) \neq [\underbrace{\underset{y_1\in\{0,1\}}{\text{argmax}}\, P(y_1|\mathbf{x})}_{\widehat{y}_1}, \ldots, \underbrace{\underset{y_3\in\{0,1\}}{\text{argmax}}\, P(y_1|\mathbf{x}, \widehat{y}_1, \widehat{y}_2)}_{\widehat{y}_3}]$$

Monte-Carlo Search



Generate samples $\{\mathbf{y}_t\}_{t=1}^{T} \sim P(\mathbf{y}|\mathbf{x})$:

$$y_1^{(t)} \sim P(y_1|\mathbf{x})$$
$$y_2^{(t)} \sim P(y_2|\mathbf{x}, y_1^{(t)})$$
$$y_3^{(t)} \sim P(y_3|\mathbf{x}, y_1^{(t)}, y_2^{(t)})$$

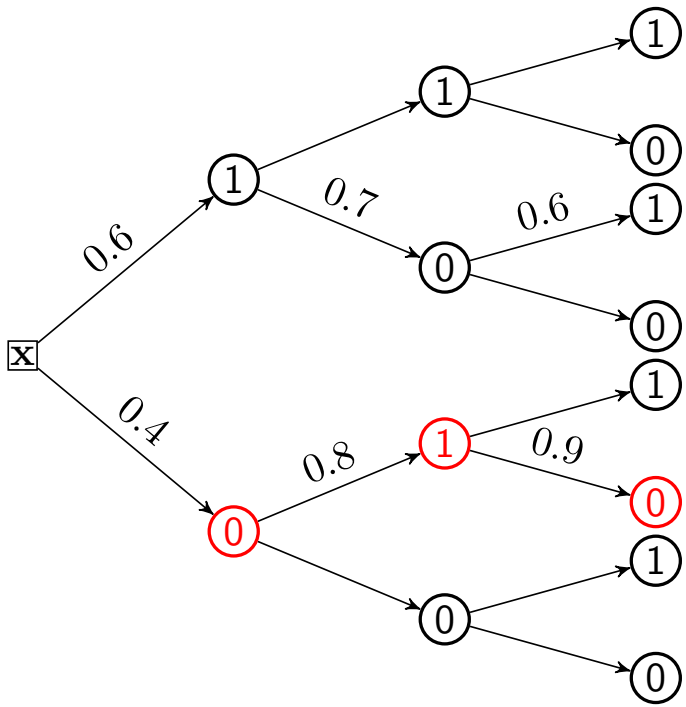
$\mathbf{y}_t = [y_1^{(t)}, y_2^{(t)}, y_3^{(t)}]$.

$$\text{return} \quad \underset{\mathbf{y} \in \{\mathbf{y}_t\}_{t=1}^{T}}{\text{argmax}} \; P(\mathbf{y}|\mathbf{x})$$

Read, Martino, and Luengo, Pat. Rec. 2014

[1] Survey: Mena et al., IJCAI 2015

Monte-Carlo Search



Generate samples $\{\mathbf{y}_t\}_{t=1}^{T} \sim P(\mathbf{y}|\mathbf{x})$:

$$y_1^{(t)} \sim P(y_1|\mathbf{x})$$
$$y_2^{(t)} \sim P(y_2|\mathbf{x}, y_1^{(t)})$$
$$y_3^{(t)} \sim P(y_3|\mathbf{x}, y_1^{(t)}, y_2^{(t)})$$

$\mathbf{y}_t = [y_1^{(t)}, y_2^{(t)}, y_3^{(t)}]$.

$$\text{return} \quad \underset{\mathbf{y} \in \{\mathbf{y}_t\}_{t=1}^{T}}{\text{argmax}} \; P(\mathbf{y}|\mathbf{x})$$

- Need probabilistic interpretation P
- Tractable, with similar accuracy to exhaustive search
- Can use other search algorithms[1], e.g., beam search, ϵ-approxmiate, A* search, ...

Read, Martino, and Luengo, Pat. Rec. 2014

[1] Survey: Mena et al., IJCAI 2015

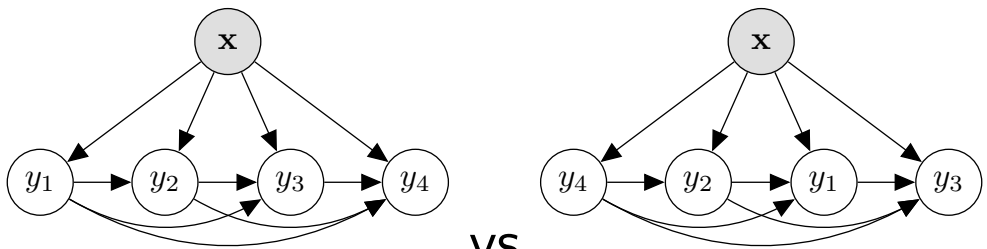
Results

Table: Average predictive performance (exact match, 5 fold CV). Independent Classifiers (IC) vs Classifier Chains with Monte-Carlo search (MCC). Base classifier logistic regression.

	L	IC	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**

On the Order of Labels in the Chain

If we permute the order of labels $Y_{\pi_1}, \ldots, Y_{\pi_L}$ as presented to the model (some permutation $\boldsymbol{\pi}$) – will predictive performance differ?

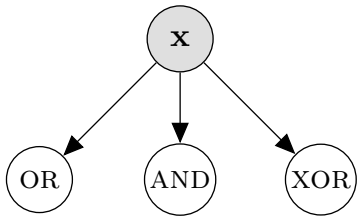


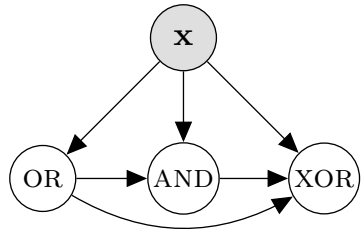
$\boldsymbol{\pi} = [1, 2, 3, 4]$ vs $\boldsymbol{\pi} = [4, 2, 1, 3]$

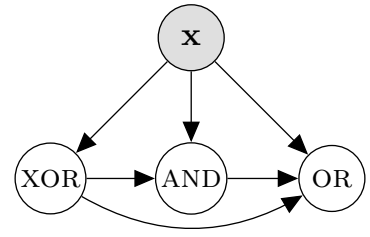
- Not in theory if given true P and optimal inference
- In practice – yes.

Which order is best?

- Existing hierarchy may not be useful:
  - Only models positive dependence (if human-defined)
  - Unlikely to be optimal for your classifiers,
  - May not even be better than a random structure.
- Based on label dependence? It *depends*; consider:







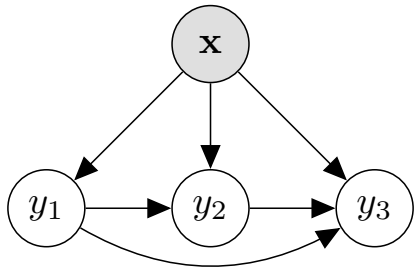
Metric	IC	CC_1	$CC_2^\dagger$
Hamming score	0.83	1.00	1.00
Exact match	0.50	1.00	1.00

Logistic reg. †Depends on base classifier/inference

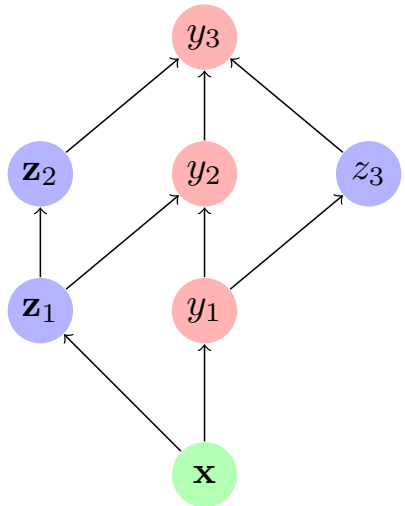
- Hill-climbing in the label-order space. Combinatorial complexity, but
  - Many local maxima.
  - Can be made more efficient with annealed proposal function (freeze the chain from left to right)
  - Having more than one order is useful: consider $\pi(1, \ldots, L|\mathbf{x})$

Classifier Chains: Why it Works

As a probabilistic graphical model:

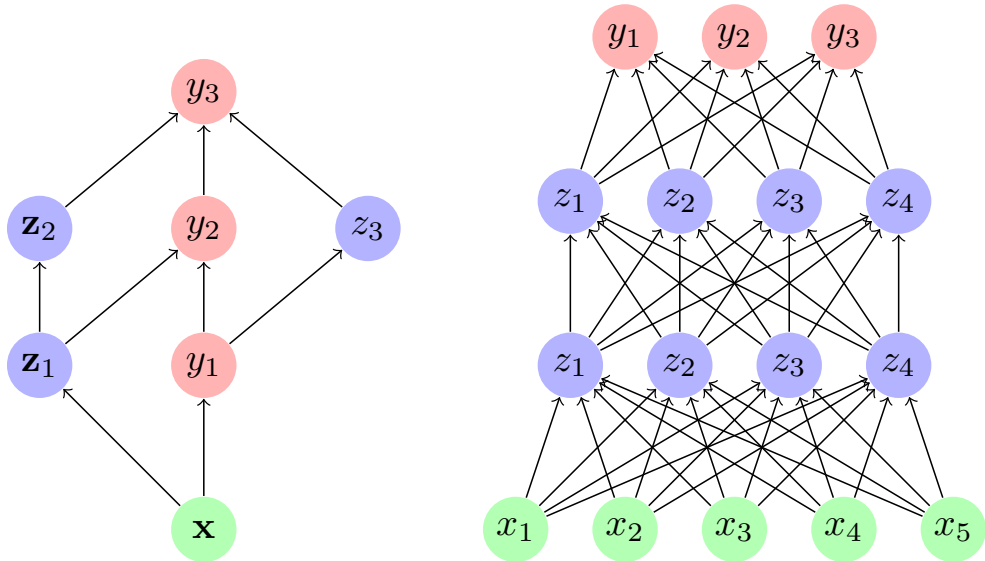


vs as a neural network (z_j nodes are delay nodes; carry forward value):



it's deep in the label space!

Advantages vs standard neural network?

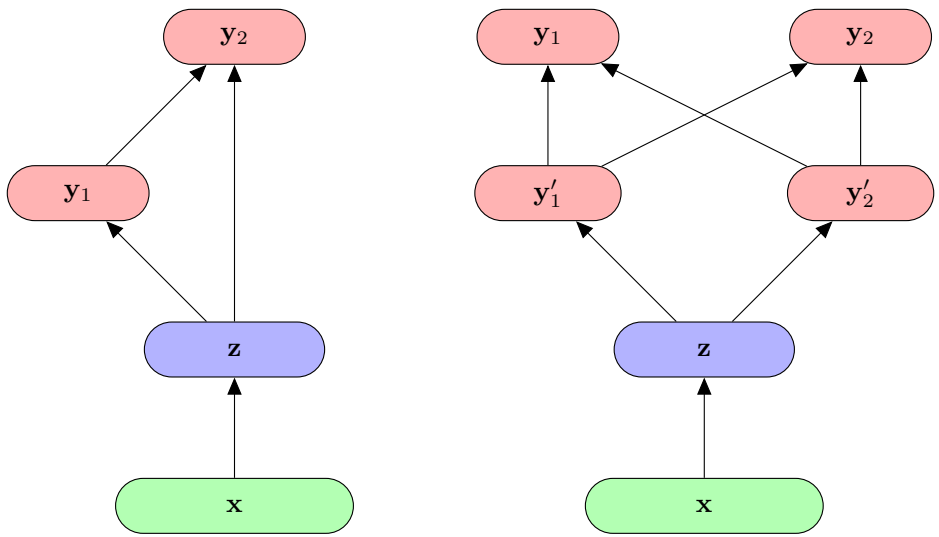


- Just apply 'off-the-shelf' [deep] neural net?
  - Dependence is modelled via the hidden layer(s)
  - Well-established, popular, competitive
- Classifier chains:
  - The 'hidden' nodes come 'for free' (faster training, not as much data required for a good model)
  - A form of transfer learning.
  - Can share techniques from multi-label learning/neural nets.

Deep in the Label Space

If using labels as inputs helps in predicting other labels. . . More labels are better? Where can we get more labels from?

- Use 'meta labels' (build labels from the label space)
- Use the same labels multiple times
  (remark: incorrect predictions can still be good representations!)



Results

Table: Exact Match, base classifier = logistic regression, except $\mathtt{BR}_{RF}$ (random forest)

Dataset	BR	$\mathtt{BR}_{RF}$	CC	...	CCSL	...	DNN
Logical	0.52 9	1.00 2	0.64 8	...	1.00 2	...	0.83 6
Music	0.23 8	0.25 5	0.25 4	...	0.26 1	...	0.25 3
Scene	0.47 8	0.48 7	0.55 5	...	0.58 1	...	0.56 2
Yeast	0.14 6	0.10 9	0.18 3	...	0.18 5	...	0.12 7
Medical	0.45 7	0.68 4	0.46 6	...	0.68 2	...	0.62 5
Enron	0.11 7	0.12 6	0.12 5	...	0.13 2	...	0.09 8
Reuters	0.45 7	0.47 4	0.47 3	...	0.47 2	...	0.38 8
Ohsumed	0.15 4	0.17 2	0.15 3	...	0.15 6	...	0.21 1
MediaMill	0.09 8	0.12 2	0.12 3	...	0.11 6	...	0.05 9
Bibtex	0.10 5	0.10 7	0.11 4	...	0.16 3	...	0.07 8
Corel5k	0.01 7	0.01 5	0.01 4	...	0.02 1	...	0.01 7
avg rank	6.95	4.82	4.36	...	2.91	...	5.82

CCSL ('Cascaded Synthetic Labels') performs well vs independent-classifier baseline, random-forest baseline, CC, and 'deep neural network' (DNN – MLP with two hidden layers).

Results

Cascaded Overlapped Blocks[2] (COB) vs MLP vs ADIOS[3]

Dataset	Model	Macro F_1	Micro F_1
`Delicious`	MLP	15.12	38.81
	ADIOS$_{RND}$	15.47	38.93
	ADIOS$_{MBC}$	**17.69**	**39.31**
	COB	15.91	
`MediaMill`	MLP	9.71	59.06
	ADIOS$_{RND}$	14.72	56.22
	ADIOS$_{MBC}$	16.01	**60.01**
	COB	**18.50**	58.74
`BioASQ`	MLP	14.86	42.40
	ADIOS$_{RND}$	15.91	43.11
	ADIOS$_{MBC}$	16.14	43.49
	COB	**16.95**	**44.53**
`NUS-WIDE`	MLP	14.00	42.09
	ADIOS$_{RND}$	11.41	41.87
	ADIOS$_{MBC}$	14.14	41.85
	COB	**14.61**	**47.21**

[2] Manuscript in preparation; based on Read and Hollmén, ArXiv 2017, Read and Hollmén 2014

[3] Cisse, Al-Shedivat, and Bengio, ICML 2016

Outline

Multi-Output Regression

What happens when the output space $\mathbf{y} \in \mathbb{R}^L$, e.g.,

X_1	X_2	X_3	X_4	X_5	Y_1	Y_2	Y_3
1	0.1	3	A	NO	37.00	25	0.88
0	0.9	1	C	YES	-22.88	22	0.22
0	0.0	1	A	NO	19.21	12	0.25
1	0.8	2	B	YES	88.23	11	0.77
1	0.0	2	B	YES	?	?	?

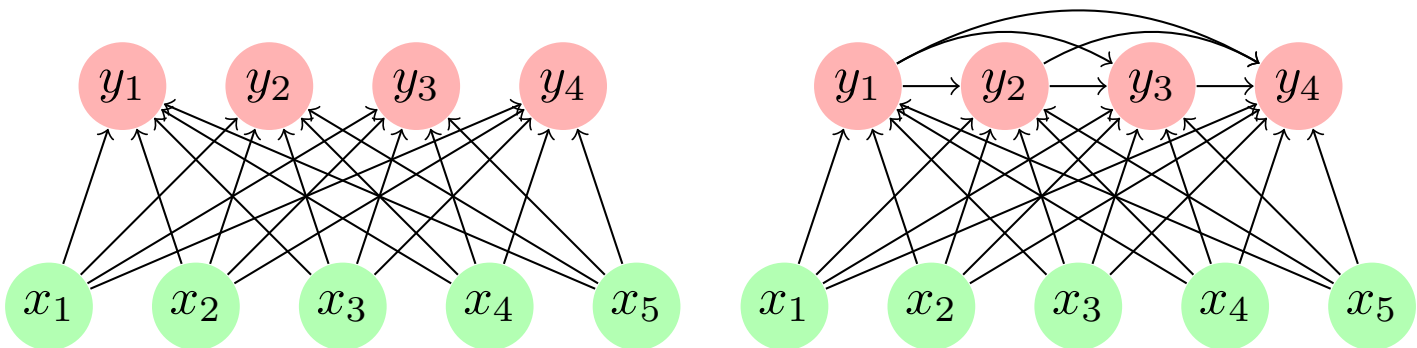
- Individual regressors – **directly applicable**.
- Chains with greedy inference – **directly applicable**.
- Chains with probabilistic inference – **not tractable**.
  Sampling? We need a model of $p(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})$.

Regression Chains

The chain model is directly applicable, i.e.,

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

for a set of regressors $[f_1, \ldots, f_L]$.



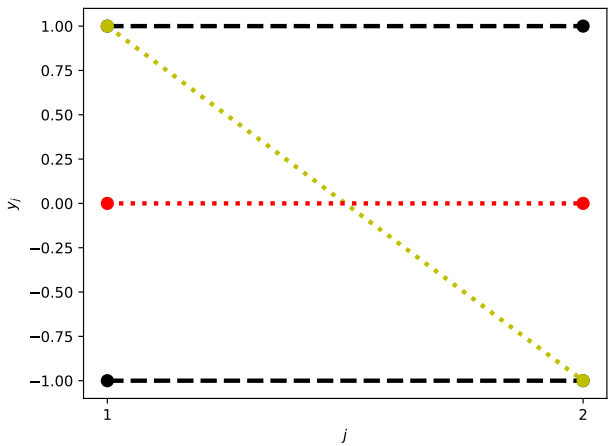
But (unlike in classifier chains):

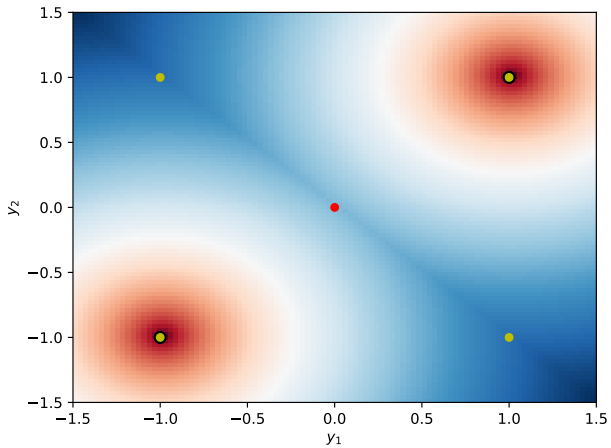
- Not necessarily a non-linearity in f, i.e., both models are equivalent (collapses into a single linear transformation)!
- What is our evaluation metric – MSE?

Toy problem:

$$x^{(i)} \sim \mathcal{N}(0, 0.1)$$

$$y_1^{(i)} \sim \mathcal{N}(\mu^{(i)}, 0.1) \quad \text{and} \quad y_2^{(i)} \sim \mathcal{N}(\mu^{(i)}, 0.1) \quad \text{where } \mu^{(i)} \sim \{-1, +1\}$$





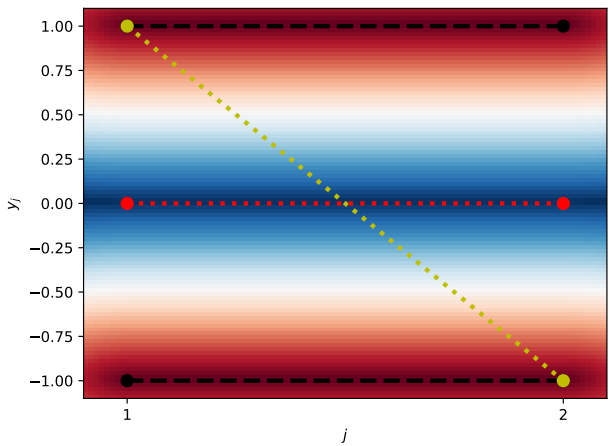
- In minimizing MSE ($\widehat{y}_j \approx \mathbb{E}[Y_j|\mathbf{x}]$; shown in red) the path cuts across the density (unknown to the learner).
- If minimizing MAE: all paths shown are equal on average.

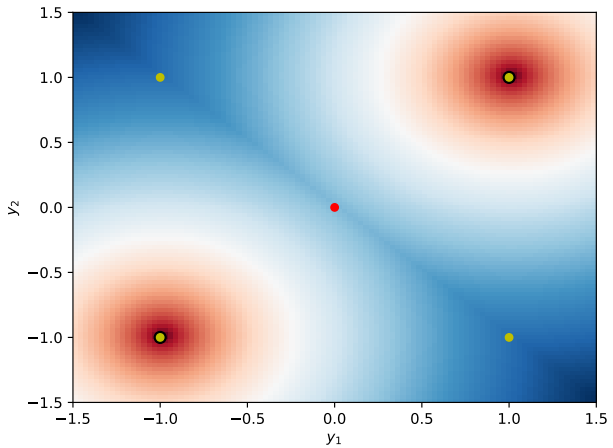
i.e., no need modelling labels together at all; except certain loss metrics, or we want interpretation (for that: we need $p(\mathbf{y}|\mathbf{x})$).

Toy problem:

$$x^{(i)} \sim \mathcal{N}(0, 0.1)$$

$$y_1^{(i)} \sim \mathcal{N}(\mu^{(i)}, 0.1) \quad \text{and} \quad y_2^{(i)} \sim \mathcal{N}(\mu^{(i)}, 0.1) \quad \text{where } \mu^{(i)} \sim \{-1, +1\}$$





- In minimizing MSE ($\widehat{y}_j \approx \mathbb{E}[Y_j|\mathbf{x}]$; shown in red) the path cuts across the density (unknown to the learner).
- If minimizing MAE: all paths shown are equal on average.

i.e., no need modelling labels together at all; except certain loss metrics, or we want interpretation (for that: we need $p(\mathbf{y}|\mathbf{x})$).

Multi-Output Regressor Chains

- As in classifier chains, we may factorize $p(\mathbf{y}|\mathbf{x})$ into $p(y_1|\mathbf{x}) \cdots p(y_L|\mathbf{x}, y_1, \ldots, y_{L-1})$, but have to model these!
- Tree search methods not applicable (there's no tree)
- Sampling? We need a model of each

$$y'_j \sim p(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})$$

Approaches:

- Discretize space
- Density estimation
- Bayesian regression
- Monte Carlo methods
- ...

Probabilistic Chains with Sequential Importance Sampling

Simplest formulation: draw samples and weight them. Under observation $\mathbf{x}$, where $w_0^{(m)} = 1$:

$$y_j^{(m)} \sim f(y_j | y_1^{(m)}, \ldots, y_{j-1}^{(m)})$$

$$w_j^{(m)} = w_{j-1}^{(m)} \cdot \frac{p(y_j^{(m)} | \mathbf{x}, y_1^{(m)}, \ldots, y_{j-1}^{(m)})}{f(y_j^{(m)} | y_1^{(m)}, \ldots, y_{j-1}^{(m)})}$$

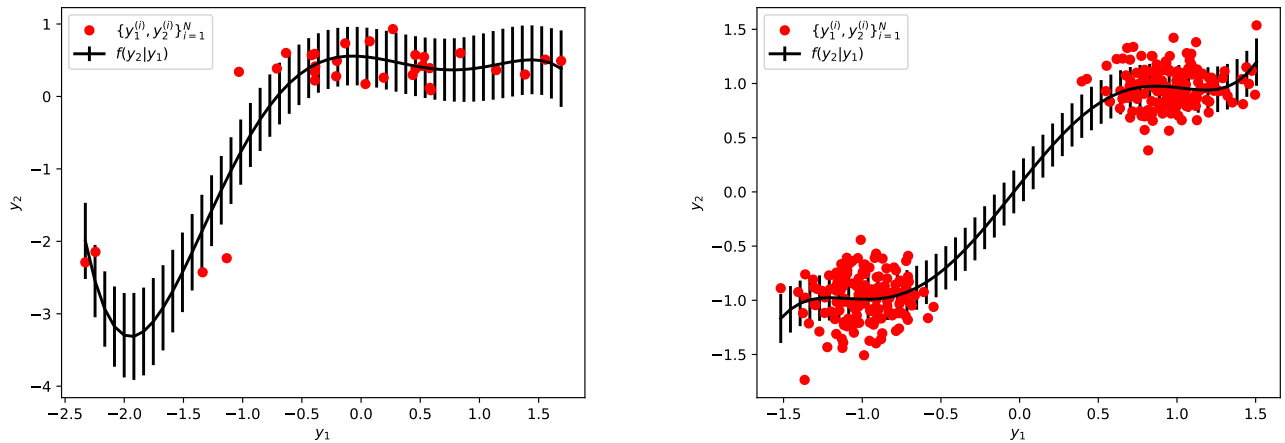
given appropriate f and p.

- We obtain M weighted trajectories

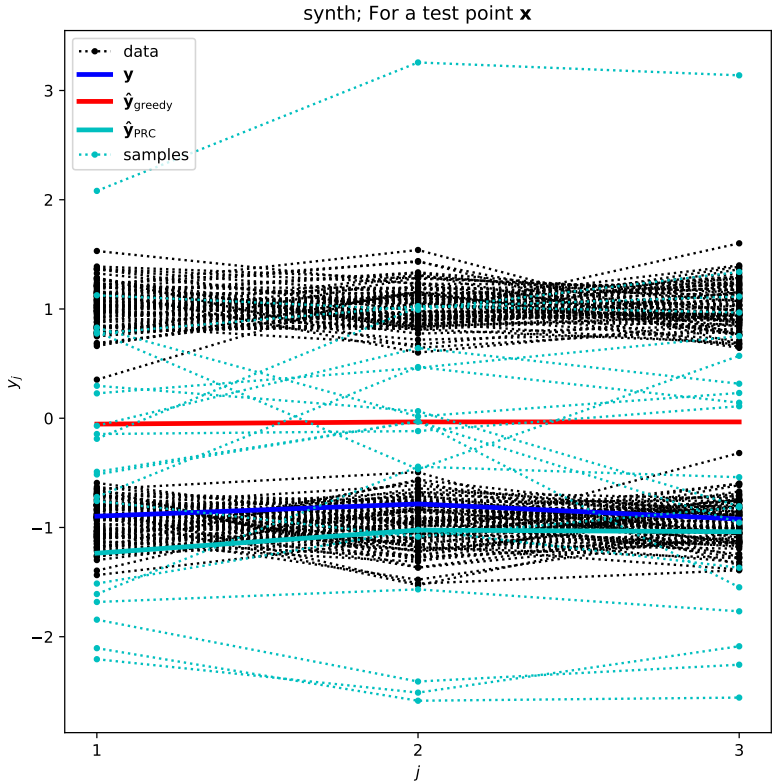
$$\{\mathbf{y}^{(m)}, w^{(m)}\}_{m=1}^{M}$$

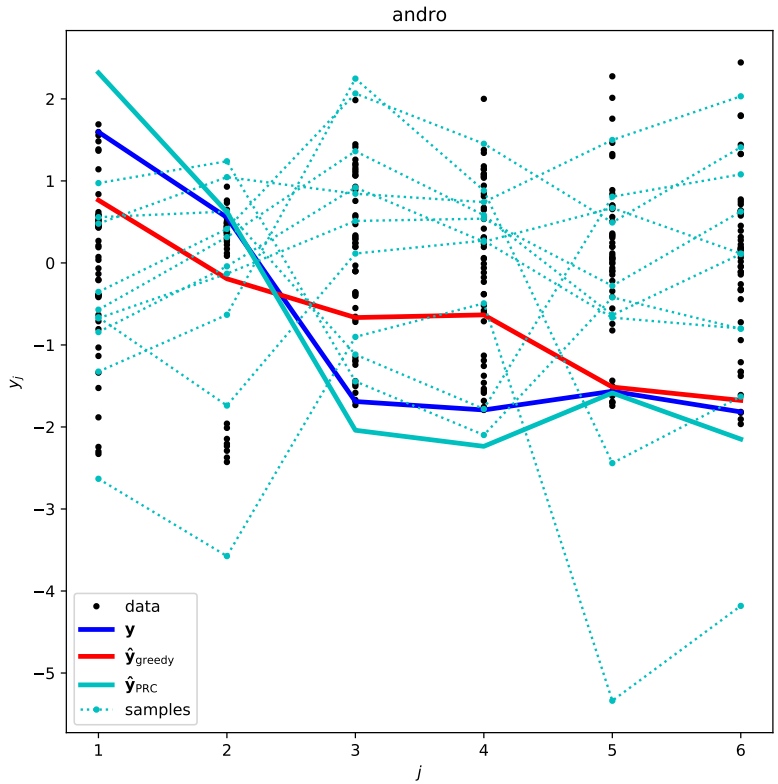
- It's (almost) a particle filter! We may also add:
  - Resampling step
  - Extra steps of MCMC or AIS schemes

Work with Luca Martino, manuscript in preparation



A Bayesian regression model suitable for drawing samples $y_j' \sim f(y_j|y_1, \ldots, y_{j-1})$ (variance shown as error bars). Shown for $y_2|y_1$ on two datasets.





The trajectory for a particular **x**; (all $\{\mathbf{x}^{(i)}$ shown in black).

Accuracy is not better on average compared to greedy chains, but predicted path *can be* a correct one; and we get a set of weighted samples.

Results

Dataset	L	IR_B	RC_B	RC_K	$MCRC_B$
synth	5	0.18	**0.18**	0.24	0.18
andro	6	0.69	0.61	**0.21**	0.52
atp1d	6	0.27	0.28	0.38	**0.27**
atp7d	6	0.38	0.39	0.53	**0.37**
edm	2	0.64	0.64	**0.49**	0.62
enb	2	0.20	**0.19**	0.21	0.20
jura	3	0.47	0.47	0.52	**0.46**
oes10	16	0.20	**0.20**	0.36	0.24
oes97	16	0.25	**0.25**	0.37	0.27
sf1	3	0.49	0.49	**0.46**	0.47
sf2	3	0.35	0.35	**0.34**	0.34
slump	3	0.47	0.47	0.51	**0.47**

MSE; Independent Regression, Regressor Chains, and sequential MC regressior chains; Bayesian or Kernel base regression.

Do Regressor Chains Work?

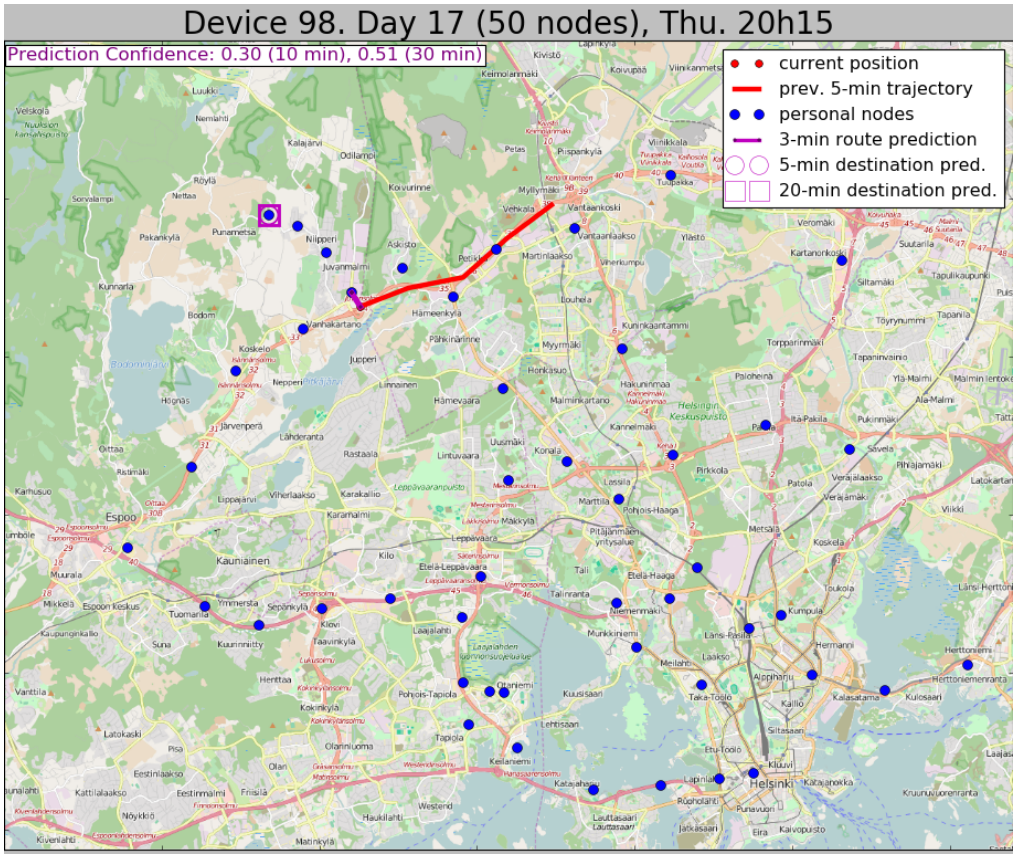
Not as clearly as classifier chains (wrt independent models), but yes if:

- In $\widehat{y}_2 = f_2(f_1(x), x)$, f_1 provides an improved (non-linear) representation. This can be seen as
  - A type of cascaded basis expansion
  - A particular form of deep learning
- In the non-isotropic case (x_j observed at step j): State space models, x_j can be used to *update* our estimate.
- In terms of interpretation, e.g.,
  - Anomaly detection
  - Missing-value imputation
  - ...
- If other label concepts arrive later, i.e., transfer learning (improvement in terms of computational time).

Outline

Route Forecasting

- Create ‘personal nodes’ for a traveller
- Model and predict routes using classifier chains
- An advantage with relatively little training data and vs other methods (e.g., HMM, RNN)

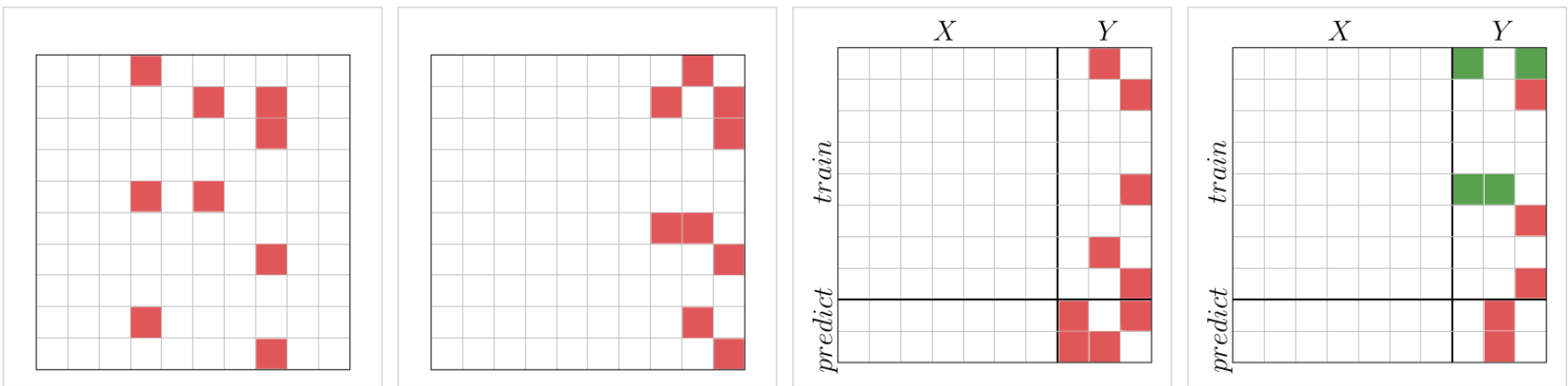


Personal nodes of a traveller and a predicted trajectory

Read, Martino, and Hollmén, Pat. Rec. 2017

Missing-Value Imputation

- Stream $\mathbf{x}_t | t = 1, \ldots$ but some values $x_j^{(t)}$ are missing.
- Turn the stream into multi-output samples, train, and predict (*impute*) missing values.
- Related to tasks in recommender systems

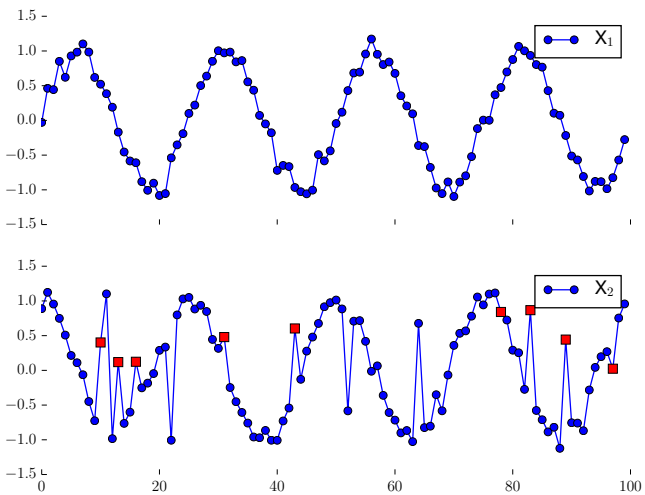


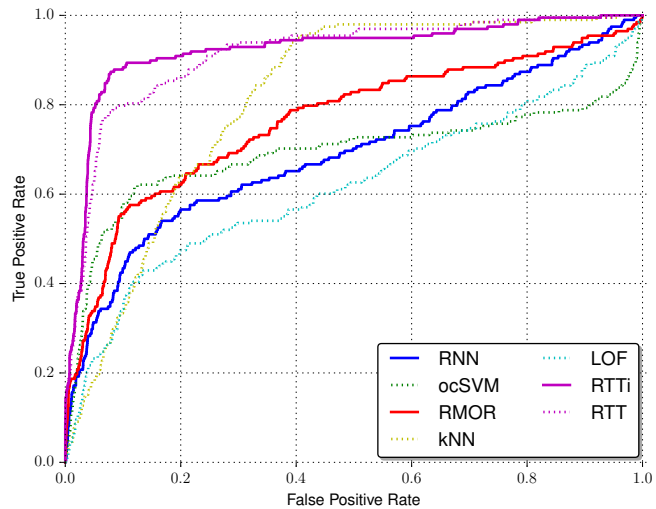
A set/stream of data transformed into a multi-output prediction problem.

Montiel et al., PAKDD 2018, and manuscript in preparation

Anomaly Detection and Interpretation

- Create 'random threads' (classifier/regressor chain cascades) through feature space and time (window)
- Monitor error spaces for anomalies
- Generate likely paths over the 'gap'
  (expand the number of samples if necessary)
- Impute this (treat it as a missing value) prior to using as a training example

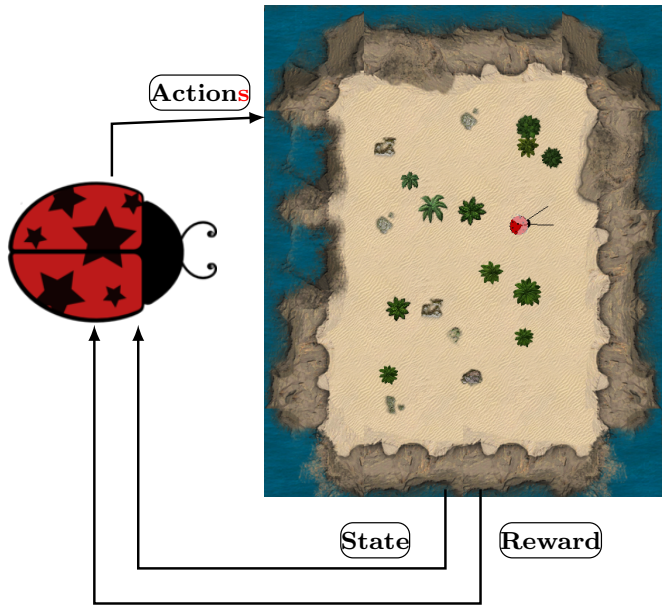




Song et al., ICDM demo 2018, and manuscript in preparation

Continual Learning

In reinforcement learning,



- Reward signal is sparse
- Self-train on own surrogate reward, then use it as a feature.
- Recall: Bad/outdated predictions are not useless representations
- i.e., build up representation; transfer learning.

Work in progress . . .

Summary

1. Data Streams are Time Series
2. Multi-label Learning
   - Classifier Chains
   - Regressor Chains
3. Applications to Data Streams

Multi-Output Chain Models and their Application in Data Streams

Jesse Read

http://www.lix.polytechnique.fr/~jread/
http://www.lix.polytechnique.fr/dascim/