Artificial Intelligence for Time-Series and Sequential Decision Making

Jesse Read

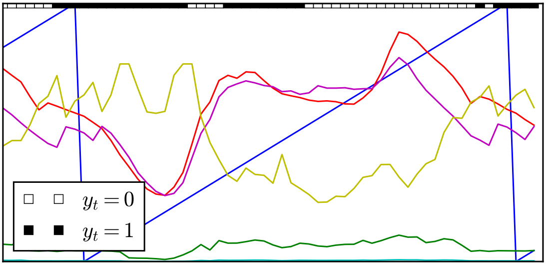
Nov. 6, 2018, Lyon.

Outline

Time Series

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots$$

generated by some process $\mathbf{x} \sim p(\mathcal{X})$ in the domain we are interested in. Measurements may be continuous, $\mathbf{x}_t \in \mathbb{R}^D$ or discrete, $\mathbf{x}_t \in \mathbb{N}_+^D$; across time t. May be associated with unobserved signal $\mathbf{y}_t$.



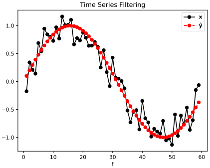
Time series $\mathbf{x}_t \in \mathbb{R}^5$ associated with state $y_t \in \{0, 1\}$.

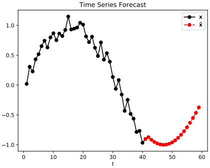
Examples of time series data:

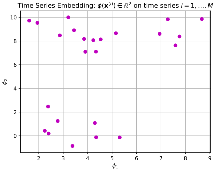
- Electricity demand for a city
- Sensor measurements on equipment in an aircraft
- Number of calls to an insurance service
- Light-sensor measurements (and movement through a room)
- Smartphone GPS and signal strength measurements of urban travellers (and their predicted trajectory)
- EEG and ECG signals obtained during sleep
- Cellular growth in trees
- Environmental measurements (temperature, humidity)

Time Series Tasks

- Filtering (*estimate*) $\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}, \mathbf{y}_t$ from observations $\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t$
- Forecasting (*predict*) $\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots$ from time t.
- Embedding: Describe a time series $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ as a vector $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_N]$ of fixed length N.







- Clustering
- Classification
- Motif extraction
- Novelty/anomaly detection
- Query by content

Outline

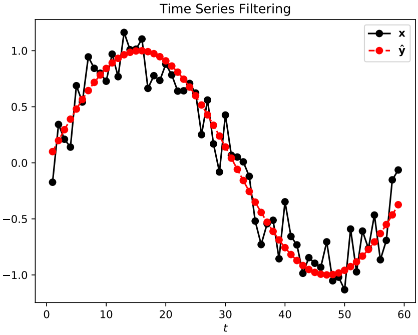
Filtering

Given observations (time series)

$$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots\}$$

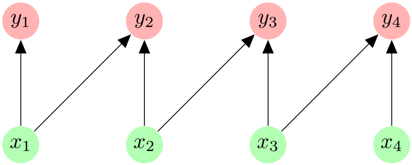
we want a model f to predict corresponding

$$\{\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_t, \ldots\}$$



Traditional Methods

- Finite impulse response (FIR) filter
- Moving average, exponential smoothing (low-pass filters)
- Kalman filter, particle filters
- ARIMA (Auto-Regressive Integrated Moving Average)



$$y_t = f(w_1 x_{t-0} + \cdots + w_k x_{t-k}) + \epsilon_t$$

with some weights $\mathbf{w} = [w_1, \ldots, w_k]$ (window size k). This is a convolution with kernel $\mathbf{w}$.

- Robust and well-understood
- Need to be hand-crafted, calibration by domain expert
- else not suitable for multiple dimensions; complex problems

Machine Learning for Filtering

Given training data, we can design a machine learning approach (e.g., artificial neural networks, decision trees, ...), on

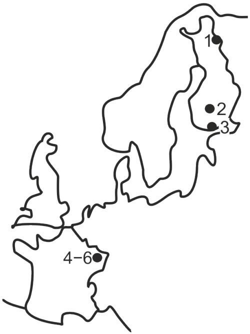
X_{t-4}	X_{t-3}	X_{t-2}	X_{t-1}	X_t	Y_t
1	A	2.3	1.8	-3	-24
A	2.3	1.8	-3	4	-28
2.3	1.8	-3	4	B	-32
1.8	-3	4	B	3	-43
...	...	...	...	...	...
T	39	3	4	0.1	?

i.e., model

$$y_t = f(x_{t-4}, \ldots, x_t; \theta) + \epsilon$$

The decision making and interpretation is relegated to the learner.

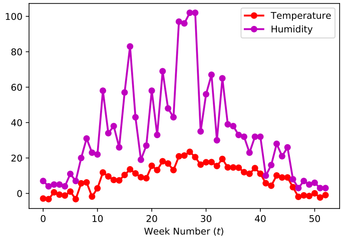
Example: Predicting Celular Growth in Scots Pine

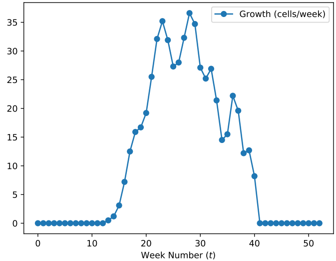


- 6 sites in Finland and France, of Scots pine
- Interested in modelling cellular growth under different latitude, altitude, . . .
- Models must be carefully crafted, parametrized, and adjusted by domain experts, *per site*.

Work with Liisa Kulmala et al. © University of Helsinki, and LERFOB, INRA, AgroParisTech.

Example: Predicting Celular Growth in Scots Pine

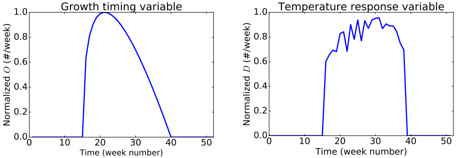




- Environmental measurements (temperature, humidity, . . .).
- Some cell-growth data (from micro-core samples and counts during growth season) over 3–4 years

Work with Liisa Kulmala et al. @ University of Helsinki, and LERFOB, INRA, AgroParisTech.

- Domain experts were using numerous functions, e.g., growth timing variable (left) and heat sum (right),

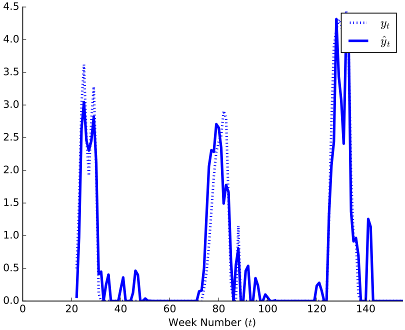


e.g., where τ_t = temperature and week t,

$$z_t = 1_{\tau_t > c} \left[\frac{1}{1 + \exp(-\beta \tau_t)} \right]$$

and c, β are per-site parameters.

- Assembled into a differential equation
- About 4-5 parameters to be hand-selected *per site*



- Data-driven model to parametrize and combine expert-inspired functions, for each site
- Achieved accuracy to within a fraction of a cell per week
- Using decision tree learners, interpretation was possible (e.g., how far back to take into account temperature measurements)

Outline

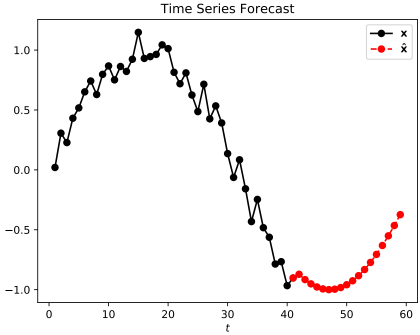
Time-Series Forecasting (Prediction)

Given

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$$

we want a model f to predict

$$\hat{\mathbf{x}}_{t+1}, \hat{\mathbf{x}}_{t+2}, \ldots, \hat{\mathbf{x}}_{t+\ell}$$



Traditional Methods

- Naive Forecasting (rain today = rain tomorrow),

$$\hat{x}_{t+1} = x_t$$

  Often effective.

- Moving average (mean of last k observations)

$$\hat{x}_{t+1} = \mathbf{w}^\top \mathbf{x}$$

  on window $\mathbf{x} = [x_{t-(k-1)}, \ldots, x_t]$, $\mathbf{w} = [\frac{1}{k}, \ldots, \frac{1}{k}]$.

- Auto-regressive linear fit on previous k points, and extrapolate.

Machine Learning for Forecasting

Formulating a data-driven supervised learning problem:

$X_{t-\ell}$	$X_{\dots}$	X_{t-2}	X_{t-1}	X_t	X_{t+1}
1	A	2.3	1.8	-3	4
A	2.3	1.8	-3	4	B
2.3	1.8	-3	4	B	3
1.8	-3	4	B	3	-7
...	...	...	...	...	...
T	39	3	4	0.1	?

i.e., model

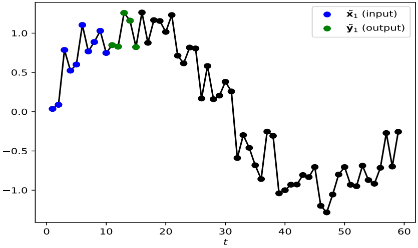
$$\hat{x}_{t+1} = f(x_{t-4}, \dots, x_t; \theta)$$

(we can plug in $\hat{x}_{t+1}$ and propagate); or estimate a window directly:

$$\hat{x}_{t+1}, \dots, \hat{x}_{t+k} = f(x_{t-4}, \dots, x_t)$$

Machine Learning for Forecasting

Formulating a data-driven supervised learning problem:



i.e., model

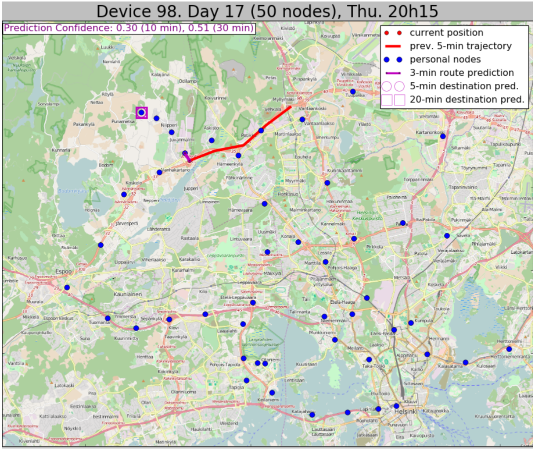
$$\hat{x}_{t+1} = f(x_{t-4}, \ldots, x_t; \theta)$$

(we can plug in $\hat{x}_{t+1}$ and propagate); or estimate a window directly:

$$\hat{x}_{t+1}, \ldots, \hat{x}_{t+k} = f(x_{t-4}, \ldots, x_t)$$

Example: Trajectory Estimation

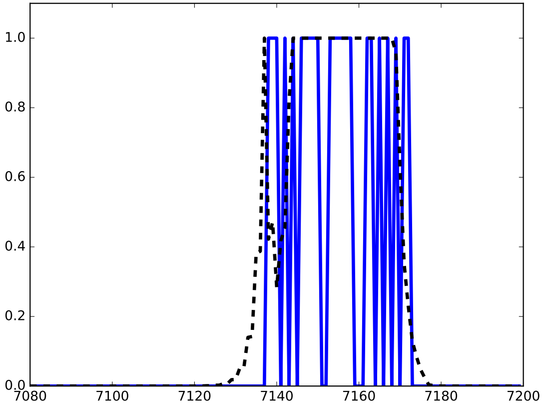
- Collected data of travellers[1]: GPS coordinates, signal strength, battery level, current time, . . .
- Predict future trajectory from current trajectory



[1] All participants volunteered to install App; share data
Work with Jaakko Hollmèn et al. @ Aalto University

Example: Predictive Maintenance of Aircraft

- Sensor readings from aircraft and textual description of observations
- Predict warnings/required replacement of components

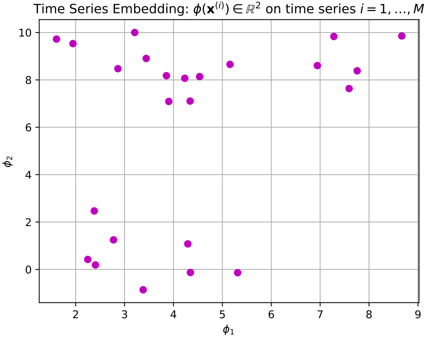


Work with DaSciM team @ École Polytechnique

Outline

Embedding Time Series

We seek to turn variable-length time series $\{\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_{T_i}^{(i)}\}_{i=1}^{M}$ into fixed-length vectors $\phi^{(i)} = [\phi_1, \ldots, \phi_D]$.



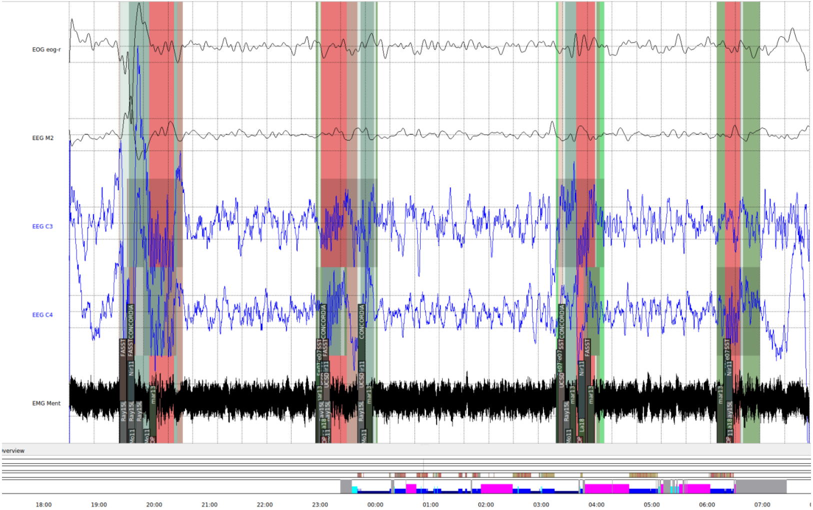
This lets us compare and cluster time series/look for anomalies, (and classify, if we have the label): measure similarity/distance between $\phi(\mathbf{x}^{(i)})$ and $\phi(\mathbf{x}^{(2)})$.

Example: Modelling and Treating Chronic Insomnia

- Goal: (semi-)automate clinical assessment; what kind of insomnia + treatment recommendation.
- Data from patients:
  - Psychological questionnaires (MMPI, CAS)
  - EEG and ECG data overnight
  - Some labels: follow-up tests/questionnaires and *biofeedback* results (some patients found success without pharmaceutical intervention, others not)

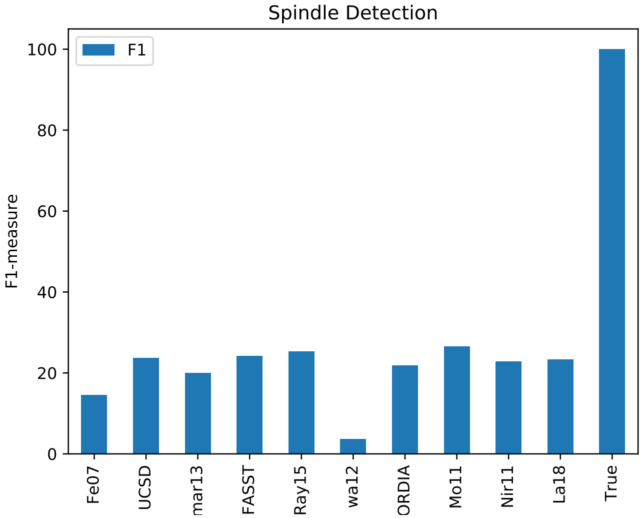
- Questionnaire data: can take 'standard' machine learning approach, $f : \mathcal{X} \to \mathcal{Y}$, and inspect feature importance, statistical correlation wrt to label variable (extent of insomnia, and improvement); cluster into groups, etc.
- Time-series data: different lengths, contains artifacts, subjects fall asleep at different times, ... How to compare?

Work with Olivier Pallanca @ École Polytechnique

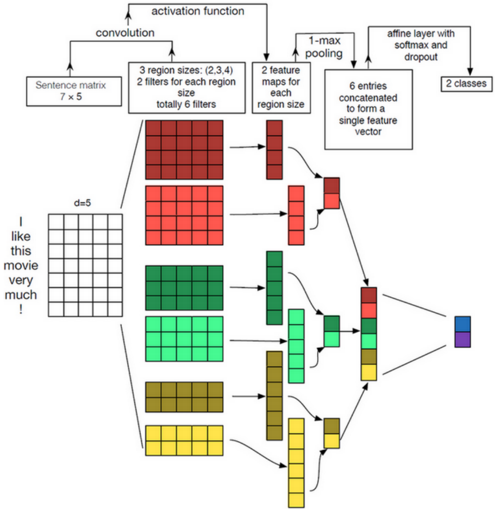


- Certain signals are of interest: Spindles, α-waves, β-waves, ...
- Simple embeddings, e.g.,
  $\phi(\mathbf{x}^{(i)}) = [\texttt{spindles/hour}, \texttt{avg freq of spindle}]$.
- Detection and labelling by an expert is labour intensive.

- There exist many rule-based methods, e.g., wavelet analysis
- But predictive performance is insufficient in many practical settings



Deep learning; $\phi(\mathbf{x}^{(i)}) =$

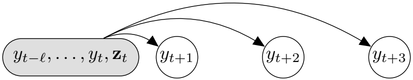


- Many current solutions are inspired by / related to NLP.
- Similar to a 'simple' embedding, but more data-driven.

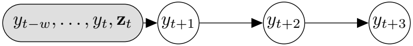
(work in progress)

Outline

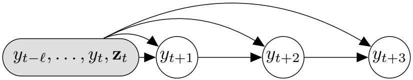
Multi-Step-Ahead Forecasting



Direct

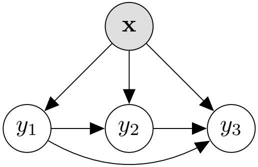


Iterated

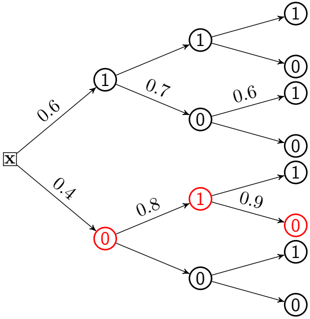


Classifier/Regressor Chain cascade

Classifier Chains



For example, where each $y_t \in \{0, 1\}$



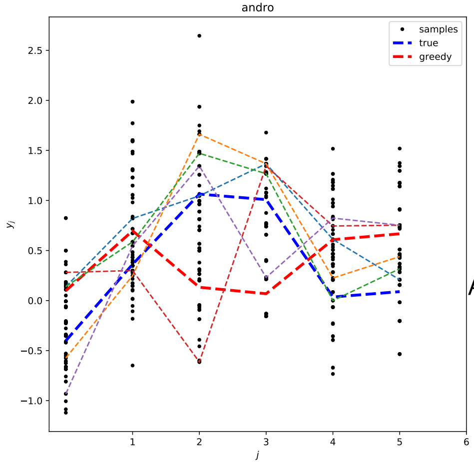
- Predictions become input, across a cascade/chain
- Efficient
- Probabilistic interpretation:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} P(y_t|\mathbf{x}, y_1, \ldots, y_{t-1})$$

$$\hat{\mathbf{y}} = f(\mathbf{x}) = \underset{\mathbf{y} \in \{0,1\}^3}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{x})$$

- Search probability tree (for best prediction) with AI-search techniques (Monte-Carlo search, beam search, A* search, ...)
- Explore structure

Regressor Chains



e.g., where $\mathbf{y} \in \mathbb{R}^6$,

- Sample down the chain

$$y_{t+1} \sim p(y_{t+1} | y_1, \ldots, y_t, \mathbf{x})$$

- More samples = more hypotheses
- Consider different loss functions

Applications:

- Multi-output regression
- Tracking
- Forecasting
- Anomaly detection and interpretation
- Imputation of missing data

One-Step Decision Theory

Under uncertainty, we wish to assign $y^* = f^*(\mathbf{x})$, the best label/hypothesis, $y^* \in \mathcal{Y}$, given $\mathbf{x} \in \mathbb{R}^D$.

Minimizing conditional expected loss

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \underbrace{\sum_{y \in \mathcal{Y}} \ell(f(\mathbf{x}), y) P(y|\mathbf{x})}_{\mathbb{E}_{Y \sim P(Y|\mathbf{x})}[\ell(\hat{y}, Y)|\mathbf{x}]}$$

aka risk.

under loss function ℓ, which describes our preferences.

In the case of 0/1 loss (1 if $y \neq \hat{y}$, else 0),

Maximum a Posteriori

$$y^* = \operatorname*{argmax}_{y \in \mathcal{Y}} p(\mathbf{x}|y)P(y) = \operatorname*{argmax}_{y \in \{0,1\}} P(y|\mathbf{x})$$

We can estimate P from the training data.

An intelligent agent wishes to make a decision to achieve a goal. The decision which involves the least risk. Another way of looking at the problem: utility. A rational agent maximizes their expected utility, not necessarily a simple *payoff* (e.g., amount of money):

Expected Utility

$$U(y) = \sum_{y \in \mathcal{Y}} u(y) p(y)$$

with satisfaction/utility $u(y)$ for outcome y. Different agents may have different utility functions, even when 'payoff' is the same item. Instead of labels given input, we can deal with actions given evidence and belief.

- A risk-prone agent will tend to gamble higher stakes
- A conservative (risk-adverse) agent will not
- A risk-neutral agent only cares about payoff y directly

An intelligent agent wishes to make a decision to achieve a goal. The decision which involves the least risk. Another way of looking at the problem: utility. A rational agent maximizes their expected utility, not necessarily a simple *payoff* (e.g., amount of money):

Expected Utility

$$U(y) = \sum_{y \in \mathcal{Y}} u(y)p(y)$$

with satisfaction/utility $u(y)$ for outcome y. Different agents may have different utility functions, even when 'payoff' is the same item. Instead of labels given input, we can deal with actions given evidence and belief.

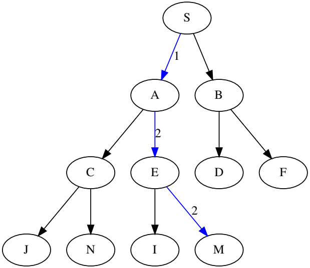
- A risk-prone agent will tend to gamble higher stakes
- A conservative (risk-adverse) agent will not
- A risk-neutral agent only cares about payoff y directly

What about sequential decisions?

Outline

In a Deterministic Environment

(e.g., board games – chess. etc.)

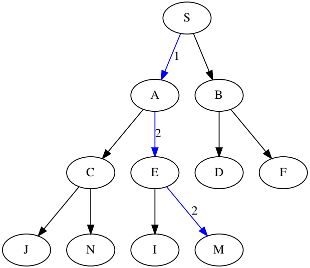


- The state space, e.g., $s_t \in \{A, B, \ldots, M\}$
- An initial state, e.g., $s_0 = S$
- A goal state, e.g., $s_t = M$
- A set of actions, e.g., $a_t \in \{1, 2\}$
- A cost for each branch, e.g., $\text{Cost}(S, A) = 1$

It's just a search! AI-search techniques applicable (DFS, A^*, ...).

In a Deterministic Environment

(e.g., board games – chess. etc.)



- The state space, e.g., $s_t \in \{A, B, \ldots, M\}$
- An initial state, e.g., $s_0 = S$
- A goal state, e.g., $s_t = M$
- A set of actions, e.g., $a_t \in \{1, 2\}$
- A cost for each branch, e.g., $\text{Cost}(S, A) = 1$

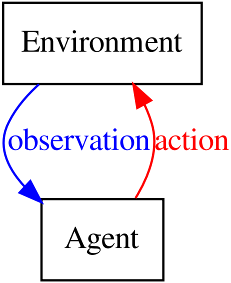
It's just a search! AI-search techniques applicable (DFS, A^*, ...).

What if environment is stochastic?

Markov Decision Processes (MDP)

MDPs are models that seek to provide optimal solutions for stocastic sequential decision problems.

$$\text{MDP} = \text{Markov Chain} + \text{One-step Decision Theory}$$



Now we have a model with

- $\mathcal{P}(s'|s,a)$ transition function
- $\mathcal{R}(s',a,s)$ reward function

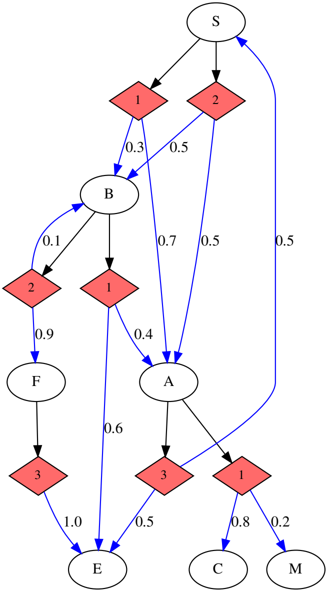
Objective: obtain a policy

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

which maximizes expected reward:

$$\mathbb{E}[R_0|s_0 = s] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t)\right]$$

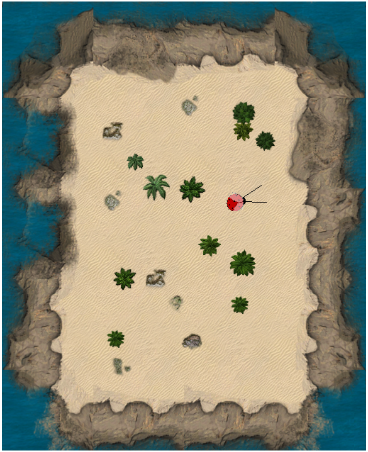
solution can be found via dynamic programming! Just need the model ...

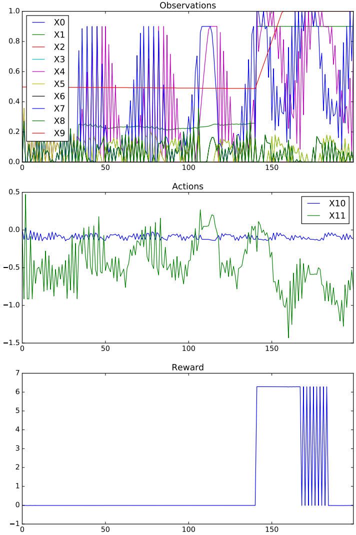


Reinforcement Learning

We don't have the model!

- Don't have transition/reward functions.
- No input-output training pairs, just reward signal.
- The agent needs to experiment! Exploration vs exploitation.
- Deep neural net can learn a model
- ...over millions of iterations.
- Emerging applications:
  - Gameplay
  - Robotics (usually trained in simulation)
  - Parameter-tuning, etc. (as a tool)
- Transfer learning is promising




Observations
X0
X1
X2
X3
X4
X5
X6
X7
X8
X9
Actions
X10
X11
Reward

1. Time Series
2. Filtering
3. Forecasting
4. Embedding
5. Classifier and Regressor Chains
6. Sequential Decision Making

Wrap Up

- Time series are everywhere
- Established machine learning and AI methods can be applied
- Automatically parametrize domain-expert knowledge
- Powerful deep learning methodologies can remove intensive tasks (by an expert), but not (yet) the expert!

Challenges (to move to stronger AI):

- Deep neural networks need computational power
- . . . and need a lot of *labelled* data (of high quality)
- . . . and are often difficult to interpret.
- Agents need to learn in a stochastic environment on a weak/sparse reward signal.
- Reinforcement learning is still underdeveloped, but holds interesting potential.

Artificial Intelligence for Time-Series and Sequential Decision Making

Jesse Read

Outline

1. Time Series
2. Filtering
3. Forecasting
4. Embedding
5. Classifier and Regressor Chains
6. Sequential Decision Making