Recursive optimization

- Viterbi algorithm
- Sequential optimization
- Dynamic time warping

All these are examples of dynamic programming (DP).
Approximate dynamic programming and reinforcement learning.

Recall Hidden Markov models

$x_i \in \{0, \ldots, k\}$, $p(x_i | x_{i-1}, \ldots, x_0) = p(x_i | x_{i-1})$,
$p(y_i | x_i, \ldots, x_0, y_{i-1}, \ldots, y_1) = p(y_i | x_i)$.
Recursive forward *summation*.

$$p(x_i | y_1, \ldots, y_{i-1}) = \sum_{x_{i-1}} p(x_i | x_{i-1}) p(x_{i-1} | y_1, \ldots, y_{i-1})$$

$$p(x_i | y_1, \ldots, y_i) = \frac{p(x_i | y_1, \ldots, y_{i-1}) p(y_i | x_i)}{\sum_{x_i} p(x_i | y_1, \ldots, y_{i-1}) p(y_i | x_i)}$$

Optimal sequence

$$[\hat{x}_1, \ldots, \hat{x}_n] = \mathrm{argmax}\{p(x_1, \ldots, x_n | y_1, \ldots, y_n)\}$$

Recursive forward *maximization*.

$$\delta_k(1) = p(x_1 = k)p(y_1 | x_1 = k)$$
$$\delta_l(i+1) = max_k\{\Delta_{k,l}(i, i+1)\},$$
$$\Delta_{k,l}(i, i+1) = \delta_k(i)p(x_{i+1} = l | x_i = k)p(y_{i+1} | x_{i+1} = l)$$

Optimal sequence by backtracking - 'path' selection

$$[\hat{x}_n] = \mathrm{argmax}_l\{\delta_l(n)\}$$

$$[\hat{x}_i | \hat{x}_{i+1}, \ldots, \hat{x}_n] = \mathrm{argmax}_k\{\Delta_{k,\hat{x}_{i+1}}(i, i+1)\}$$

Optimal sequence vs marginal probabilities

Joint maximum:

$$[\hat{x}_1, \ldots, \hat{x}_n] = \text{argmax}\{p(x_1, \ldots, x_n | y_1, \ldots, y_n)\}$$

Marginal maximum:

$$[\tilde{x}_i] = \text{argmax}\{p(x_i | y_1, \ldots, y_n)\}, \quad i = 1, \ldots, n.$$

Joint maximization is possible for this Markov model because of pairwise coupling.

Sequential decisions

1. First, make best decision among many.
2. See outcome of selection.
3. Then, depending on the outcome $x_i \in \{1, \ldots, k\}$, make next best decision.

The sequential selections depends on the outcome of the previously selected nodes. Conditioning influences the conditional probabilities for models with statistical dependence.

Selection of drilling locations

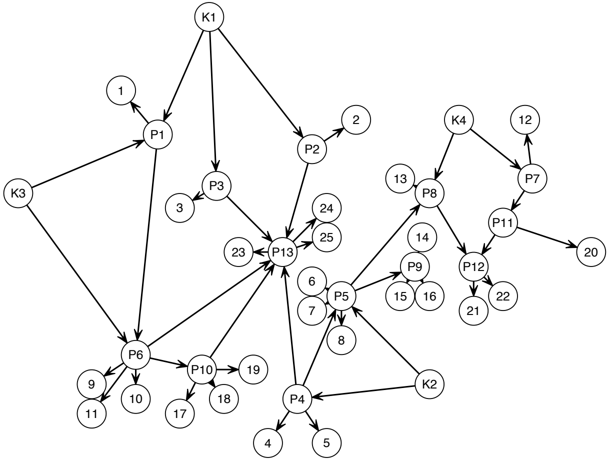
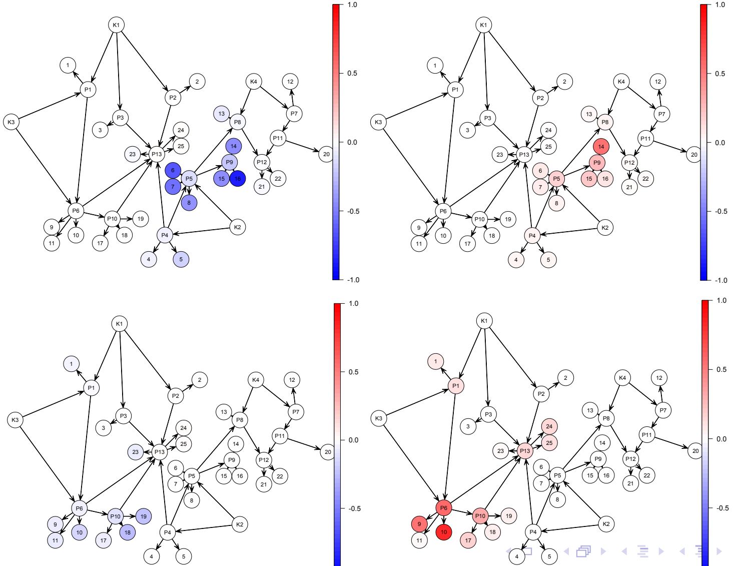


Figure: Network of 25 drilling prospects, identified with the nodes from 1 to 25, where we can possibly drill.

Evidence propagation



Sequential decisions

Value is defined by nested equations:

$$v = \max_{i \in N} \left\{ \sum_{j=1}^{k} p(x_i = j) \left[r_i^j + \delta \max_{s \in N-1} \left\{ \sum_{l=1}^{k} p(x_s = l | x_i = j)(r_s^l + \dots), 0 \right\} \right], 0 \right\}$$

- Reward r_i^j.
- Discounting factor δ.
- Sequence of maximizations and expectations.

Way of life

1. First, decide which node, if any, to exploit first.
2. Then, depending on the outcome $x_i \in \{1, \dots, k\}$, which node to exploit next, if any, and so on.

The sequential selections depends on the outcome of the previously selected nodes. Conditioning influences the conditional probabilities for models with statistical dependence. Greedy **exploitation** often yields poor **exploration**. Need both!

DP solves the optimization problem by working backwards:

1. First, decide whether to drill the last prospect, conditional on the first $N-1$ observables.
2. Then, decide which prospect to drill if there are two nodes left, and so on, until the initial empty set.

Combinatorial complexity - approximations required.

Common approximation methods : heuristics

Naive strategy:

1. First, decide best from marginals.
2. Then, decide second best from marginal, third best marginal, if positive rewards, and so on.
3. Value approximation by naive selection:

$$v_N = \sum_{i=1}^{N} \max \left\{ \sum_{j=1}^{k} r_i^j p(x_i = j), 0 \right\} ,$$

Common approximation methods : heuristics

Myopic strategy:

1. First, decide best from marginal. $i_{(1)}$.
2. Observe, $x_{i_{(1)}} = j$, and condition based on data.
3. Then, decide second $i_{(2j)}$ from conditional distribution, observe $x_{i_{(2j)}}$, condition, and continue if positive rewards, and so on.
4. Value approximation by myopic selection:

$$
\begin{aligned}
v_1 &= \max\left\{\sum_{j=1}^{k} r_i^j p(x_{i_{(1)}} = j), 0\right\} \\
v_2 &= \sum_{j=1}^{k}\left(\max\left\{\sum_{l=1}^{k} r_{x_{i_{(2j)}}}^l p(x_{i_{(2j)}} = l | x_{i_{(1)}} = j), 0\right\}\right) p(x_{i_{(1)}} = j) \\
v_M &= \sum_{i=1}^{N} \delta^{i-1} v_i,
\end{aligned}
$$

Other heuristics

- Look-ahead stategies account for the next stages, but not all future rewards.
- Rolling-horizon look-ahead startegies, conditioning every step, then look-ahead.
- In large problems value computation is approximated over Monte Carlo samples, playing the game of the strategy.

Comparison of some heuristics

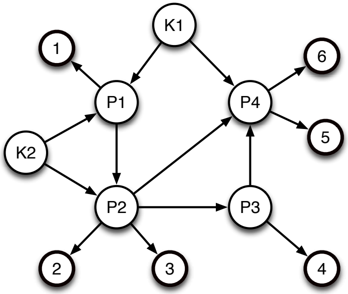


Figure: BN used for small case. N=6.

Comparison of some heuristics

Table: Results of the sequential exploration program different heuristics.

	Naive	Myopic	**Exact**	Dpt1	Dpt2	Dpt3
$i_{(1)}$	3	3	6	6	6	6
$i_{(2)} \| x_{i_{(1)}} = dry$	4	Q	3	3	3	3
$i_{(2)} \| x_{i_{(1)}} = gas$	4	2	5	2	5	5
$i_{(2)} \| x_{i_{(1)}} = oil$	4	2	5	2	4	4
$i_{(3)} \| x_{i_{(1)}} = dry, x_{i_{(2)}} = dry$	6	Q	Q	Q	Q	Q
$i_{(3)} \| x_{i_{(1)}} = dry, x_{i_{(2)}} = gas$	6	Q	2	2	2	2
$i_{(3)} \| x_{i_{(1)}} = dry, x_{i_{(2)}} = oil$	6	Q	2	2	2	2
$i_{(3)} \| x_{i_{(1)}} = gas, x_{i_{(2)}} = dry$	6	4	4	5	4	4
$i_{(3)} \| x_{i_{(1)}} = gas, x_{i_{(2)}} = gas$	6	4	4	5	4	4
$i_{(3)} \| x_{i_{(1)}} = gas, x_{i_{(2)}} = oil$	6	4	4	5	4	4
$i_{(3)} \| x_{i_{(1)}} = oil, x_{i_{(2)}} = dry$	6	4	4	5	3	5
$i_{(3)} \| x_{i_{(1)}} = oil, x_{i_{(2)}} = gas$	6	4	4	4	2	2
$i_{(3)} \| x_{i_{(1)}} = oil, x_{i_{(2)}} = oil$	6	4	4	4	2	2
Final Value	0.63	1.67	4.960	3.85	4.84	4.93
Time	0.24 sec	0.24 sec	85.6 sec	0.43 sec	3.52 sec	16.11 s

Simulation regression approaches

Training:

1. Run many different strategies
2. Note results (rewards) along the way, for each strategy.
3. Fit a regression model for rewards based on this training data.

Regression:

1. First, decide best from regression function.
2. Observe, and condition based on data.
3. Then, update with observations, and continue with next best positive rewards, according to the regression model, and so on.

Reinforcement learning using neural networks for training are very popular at the moment. (AlphaGo).

Gas pipe data

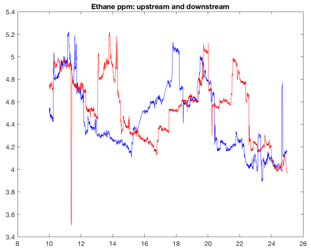


Figure: Gas-pipe ethane measured in Norway, and in Germany.

Must align time series.

Dynamic time warping (DTW)

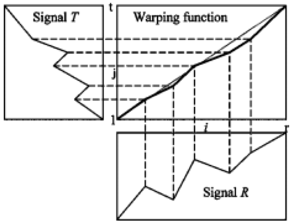


Figure: Illustration of warping function

Used a lot in speech recognition.

Optimization problem

Time series x_i, $i = 1, \ldots, n$, y_j, $j = 1, \ldots, m$.
Define path $p : \{i, j\}$, $i = 1, \ldots, n$.

$$\hat{p} = \text{argmax } \{D_p(x, y)\}$$

$$D(p) = \sum_{i=1}^{n} (x_i - y_{p(i)})^2.$$

Distances

Constraints on path p.
Cumulative distances

$$D(i,j) = \min\{D(i-1,j-1), D(i,j-1), D(i-1,j)\} + (x_i - y_j)^2$$

$D(i,j)$ computed in cumulative manner, moving forward, from the nearest neighbors.
d and D form two alignment matrices over paths $(i, p(i))$.

Aligning sequences

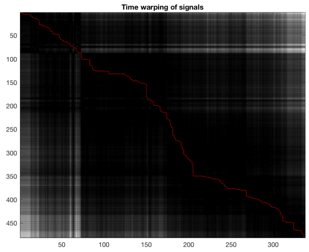


Figure: Alignment matrix for ethane time series data, and optimal warping path.

Optimal (global) path is solved by backward selection in the cumulative distance matrix:

$$\hat{p} = \text{argmax}\ \{D_p(x, y)\}$$

Full scale optimization can be slow, windowing, bounds, etc. can speed up algorithm.

Project : Viterbi and DTW

- **Viterbi for HMM:** Reconsider the HMM from the former project (Normalizing constant), with the given data, and fixed hyper-parameters. Run a Viterbi algorithm to find the joint optimal solution of states $x_i \in \{0,1\}$, $i = 1, \ldots, n$. Compare with marginal maximum solution.
- **DTW for time series:** Simulate a mean zero, unit variance Gaussian process $x_1, \ldots, x_{500}$ with correlation $\mathrm{Corr}(x_t, x_s) = \exp(-\frac{3|t-s|}{200})$. Define a monotone warping path $p(i)$, with slow fluctuation around the diagonal and $p(1) = 1$, $p(500) = 500$. Simulate a warped signal $y_j = x_{p(i)} + N(0, 0.2^2)$. Compute and visualize the cumulative distance matrix and derive the optimal path of the alignment. Compare with the specified path. See also code: matlab: 'dtw()', R: install.packages("dtw")