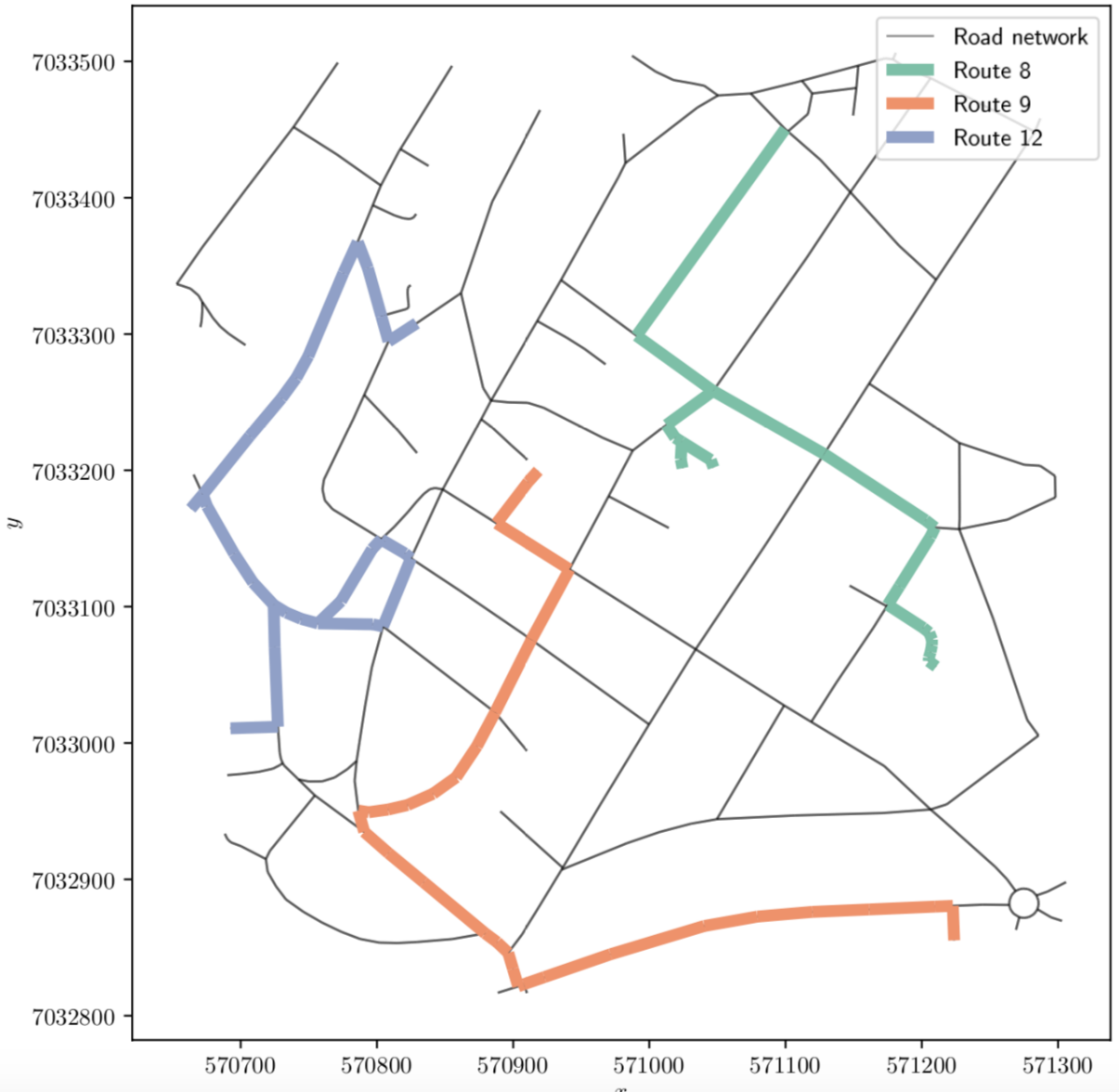
Useful "distance-based" methods for comparison and clustering

Jo Eidsvik

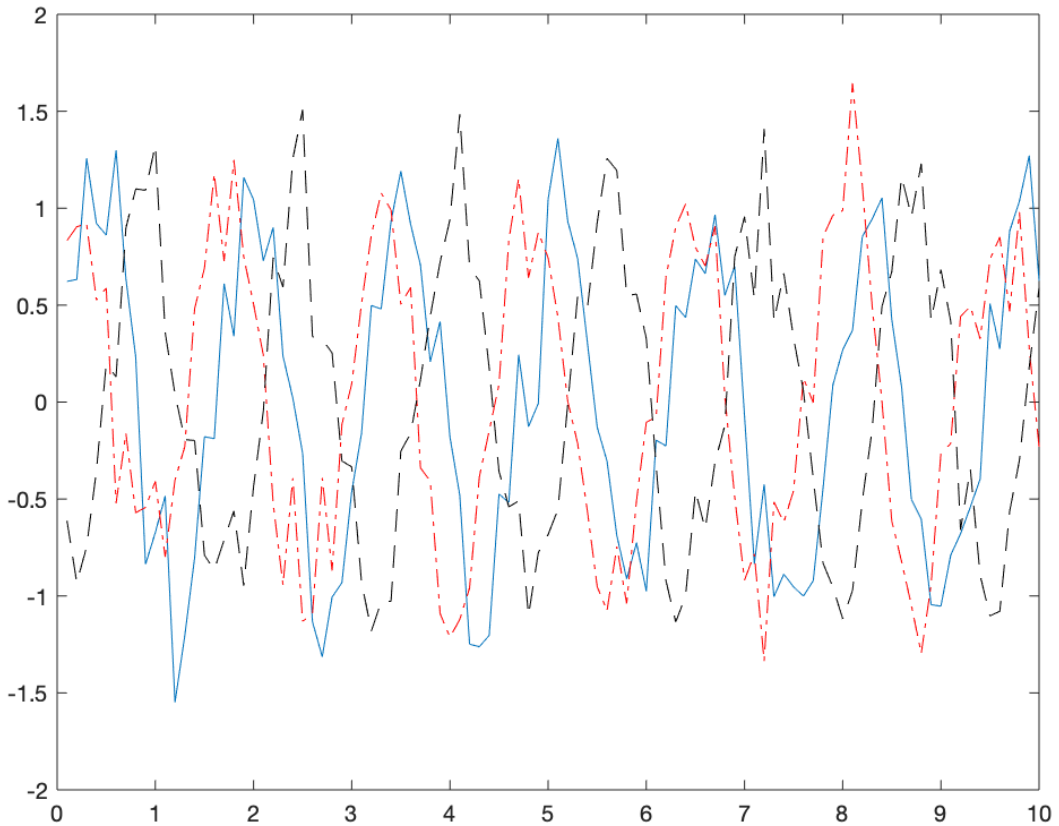
New data types - what is distance?

- Distance between time series, images, songs or videos.
- Similar shopping profiles or medical history?
- Clustering of related temperature, wind and rain datasets!
- Distances between travel routes.
- How quantify if two matrices are similar?

Distance between routes



Distance between curves



Distances

Data $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$, $\boldsymbol{Y}_i = (Y_{i1}, \ldots, Y_{ip})$ for large p.
Distance or dissimilarity of data $D_{ij} = \text{distance}(\boldsymbol{Y}_i, \boldsymbol{Y}_j)$. $i, j = 1, \ldots, n$.

Similar ideas applies to distances or loss-functions, say between densities, and for predictive power or cross-validation purposes?

Topic of course relates to norms and metrics in mathematics, but has gotten increased interest the last years with abundant multivariate data.

Main focus

- Traditional measures for statistical distance. Hausdorff distances between curves and points.
- Multidimensional scaling (MDS) for distance projection.
- Dynamic time warping (DTW) for alignment, and related methods .

Traditional distance measures in statistics: I

Some common norms and distances for data

$$D_{\text{Manhattan}}(\boldsymbol{Y}_i, \boldsymbol{Y}_j) = \sum_{k=1}^{p} \text{abs}(Y_{ik} - Y_{jk})$$

$$D_{\text{Euclid}}(\boldsymbol{Y}_i, \boldsymbol{Y}_j) = \sqrt{\sum_{k=1}^{p} (Y_{ik} - Y_{jk})^2}$$

$$D_{\text{Mahalanobis}}(\boldsymbol{Y}_i, \boldsymbol{Y}_j) = (\boldsymbol{Y}_i - \boldsymbol{Y}_j)^t \boldsymbol{S}^{-1} (\boldsymbol{Y}_i - \boldsymbol{Y}_j)$$

where $\boldsymbol{S}$ is some specified or fitted covariance matrix.
(These measures provide limited visual basis and they are not always natural or aligned for new data types.)

Traditional distance measures in statistics: II

The Kullback-Leibler (KL) distance is often used to measure the difference between two densities f and g.

$$D_{\mathrm{KL}}(f,g) = \int \log \frac{f(\boldsymbol{y})}{g(\boldsymbol{y})} f(\boldsymbol{y}) d\boldsymbol{y}$$

For two Gaussian f and g distributions, : $N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $N(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ the KL divergence is:

$$D_{\mathrm{KL}}(f,g) = \frac{1}{2}(\mathrm{tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^t \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - p + \log|\boldsymbol{\Sigma}_2| - \log|\boldsymbol{\Sigma}_1|)$$

Cross-entropy is $CE = -\int \log g(\boldsymbol{y}) f(\boldsymbol{y}) d\boldsymbol{y}$.
(Often computed empirically for training and test set in machine learning.)

Traditional distance measures in statistics III

Wasserstein distance calculated the minimum effort to transform unit masses from one density function to another (used in machine learning to avoid vanishing gradients). For two Normal distributions:

$$D_W(f,g) = \text{tr}(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 - 2(\boldsymbol{\Sigma}_2^{1/2}\boldsymbol{\Sigma}_1\boldsymbol{\Sigma}_2^{1/2})^{1/2}) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^t(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

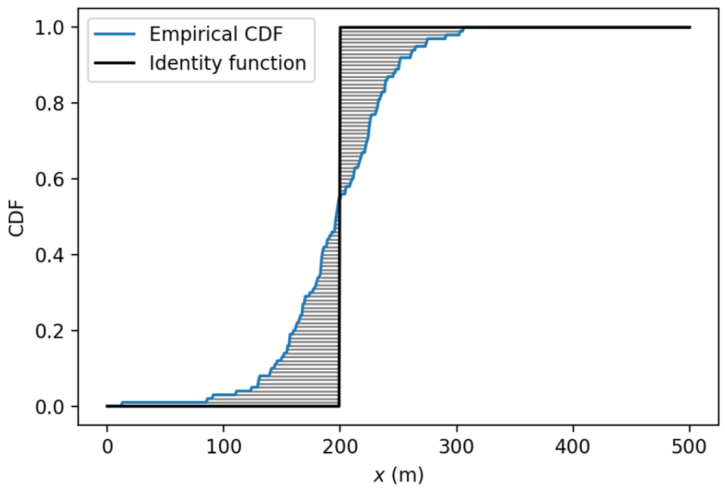
Hellinger distances, Bhattacharya distance, also compute dissimilarity between distributions.

Traditional distance measures in statistics: IV

Brier Score or controlled rank probability score:

$$D_{\text{CRPS}}(f, \boldsymbol{y}) = \sum_{k=1}^{p} \int (F(y_k) - I(y_k > y_k^o))^2 dy_k$$

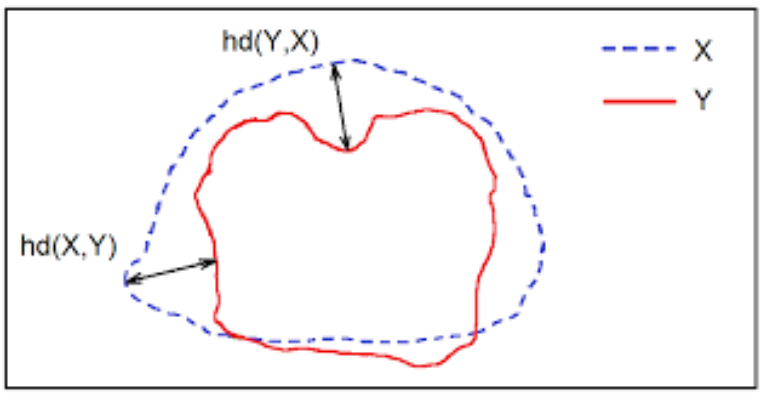
$F(y_k)$ is the cumulative distribution function of variable Y_k.
(Often used to evaluate predictive performance, cross-validation.)



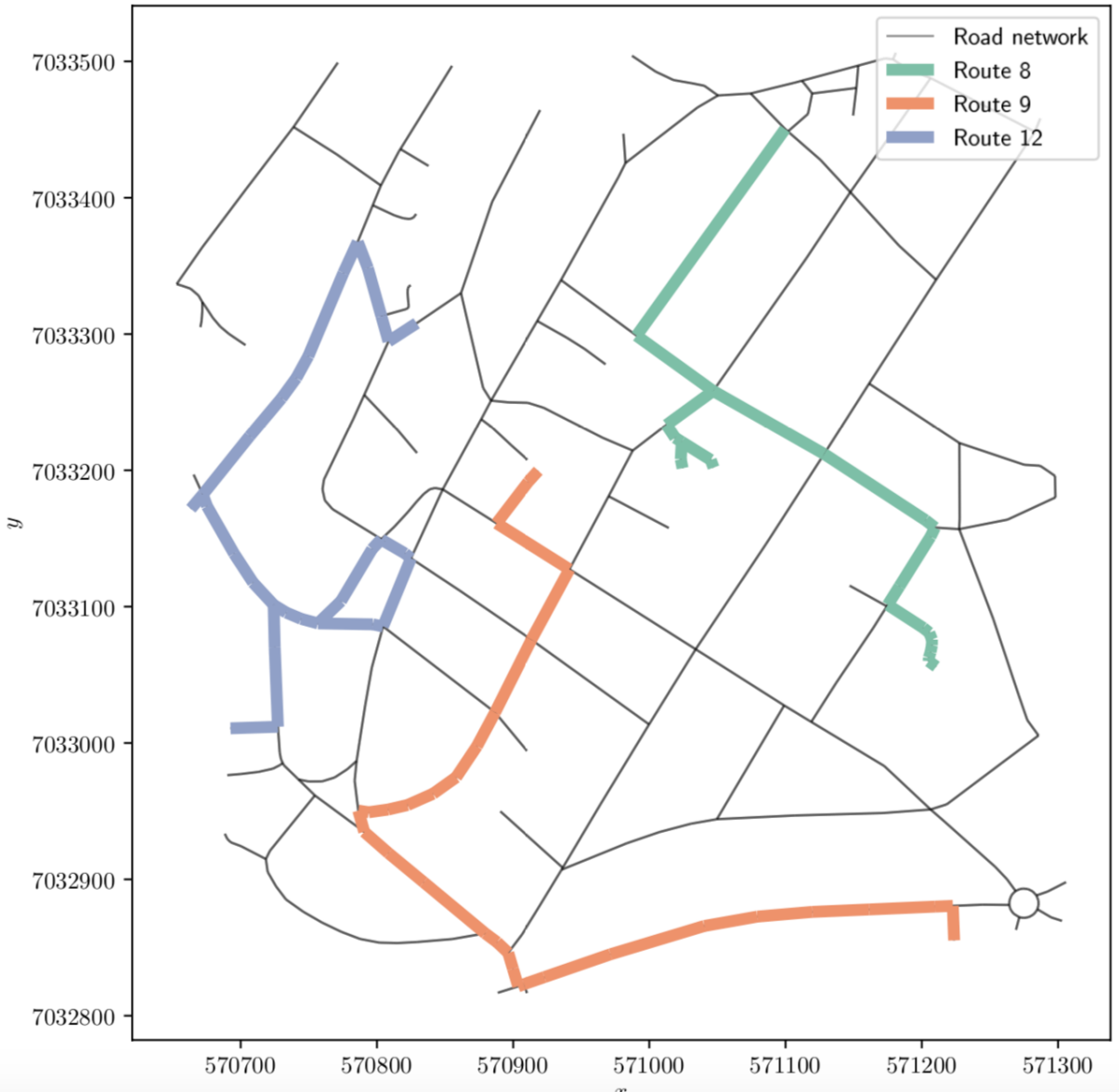
(There are related multivariate alternatives to CRPS.)

Strings, curve or point measures I

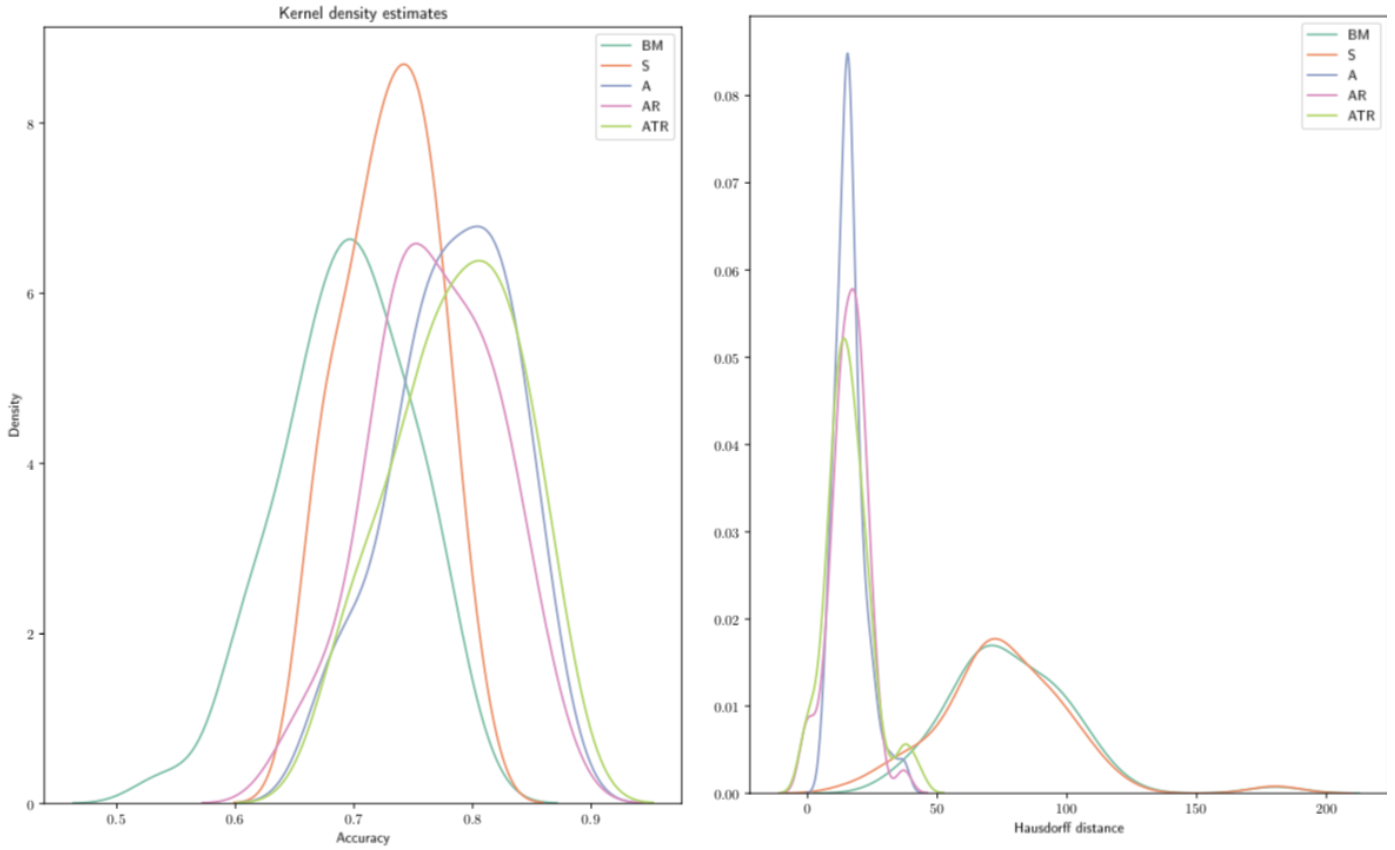
Levenshtein distance counts number of changes to match two strings.
Distance is used a lot in text / word data.
Hausdorff distance is the maximum minimum distance.from one curve or point set to the other.



Hausdorff distance between routes



Hausdorff distance between routes mapping



(a) Accuracy.

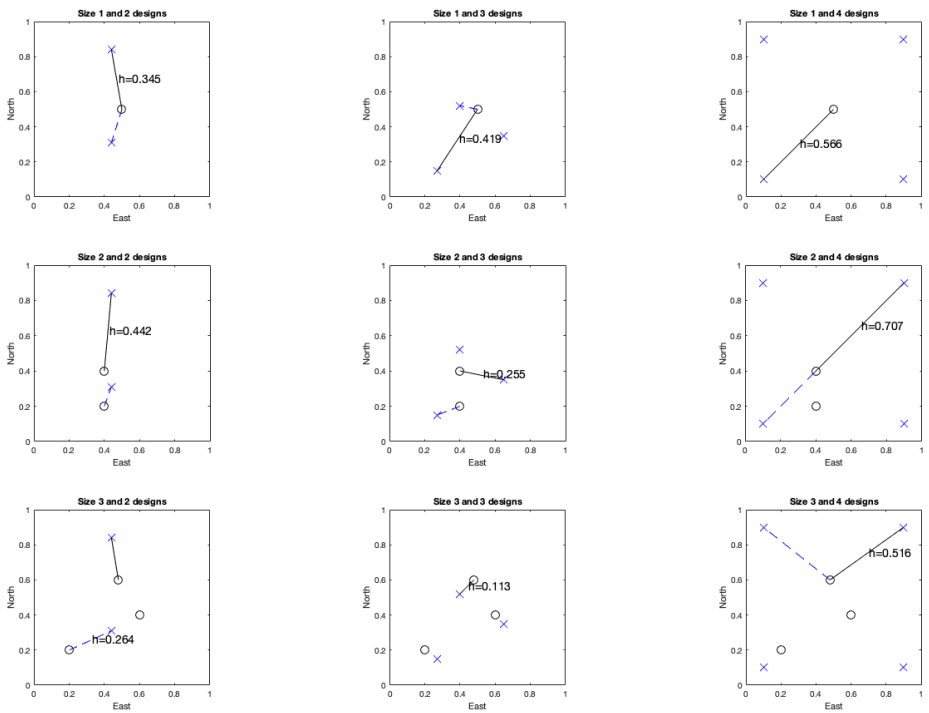
(b) Hausdorff distance.

Hausdorff distance

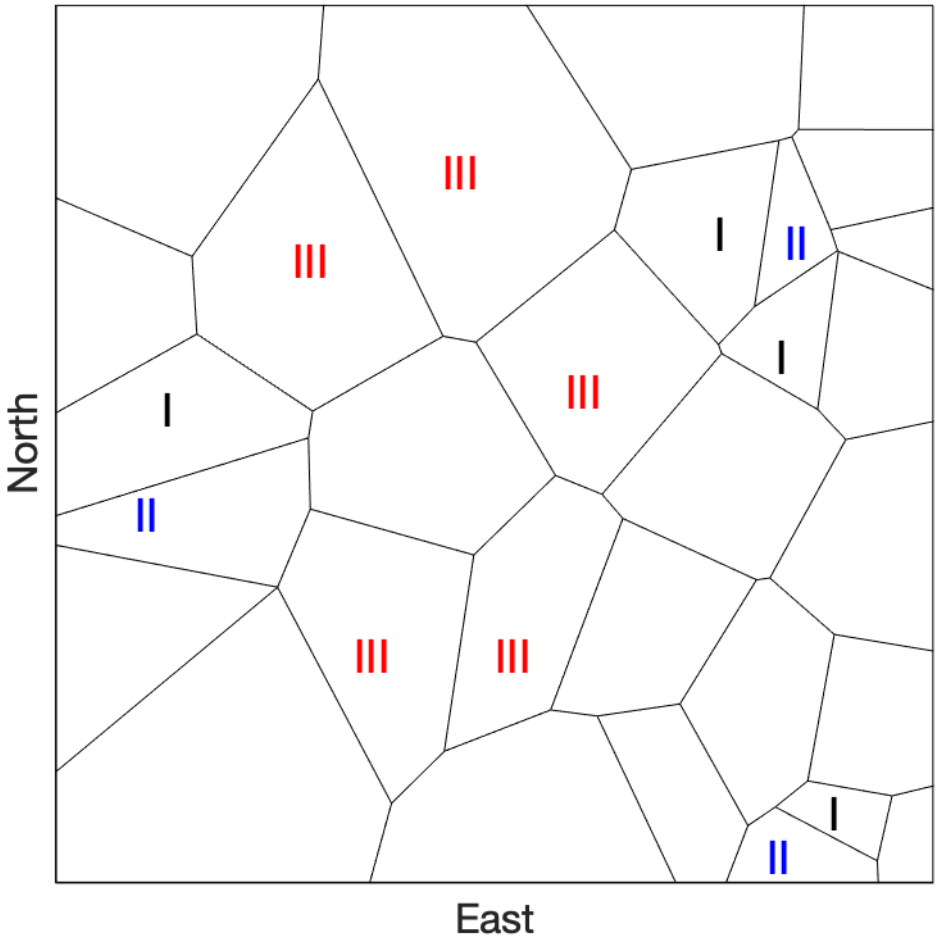
$$h = \text{dist}_1(D, C) = \max\left\{h_H(D, C), h_H(C, D)\right\},$$

$$h_H(D, C) = \max_{i=1:|D|}\left\{\min_{j=1:|C|} ||\boldsymbol{x}_{D,i} - \boldsymbol{x}_{C,j}||\right\}.$$

where $\boldsymbol{x}_{E,i}$ is a point on curve $E = \{C, D\}$.



Hausdorff distance between designs



Hausdorff distance between designs

Question is to allocate maximum n sensors to n possible different sites $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$:

$$
\begin{aligned}
D_0 &= \emptyset, \\
D_1 &= \{(\boldsymbol{s}_1), (\boldsymbol{s}_2), \ldots, (\boldsymbol{s}_n)\}, \\
D_2 &= \{(\boldsymbol{s}_1, \boldsymbol{s}_2), (\boldsymbol{s}_1, \boldsymbol{s}_3), \ldots, (\boldsymbol{s}_{n-1}, \boldsymbol{s}_n)\}, \\
&\vdots \qquad\quad \vdots \\
D_n &= \{(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n)\}, .
\end{aligned}
$$

n possible designs of cardinality one, $\binom{n}{2}$ possible designs of cardinality two, etc.
2^n possible designs in set $D = \{D_0, \ldots, D_n\}\}$.

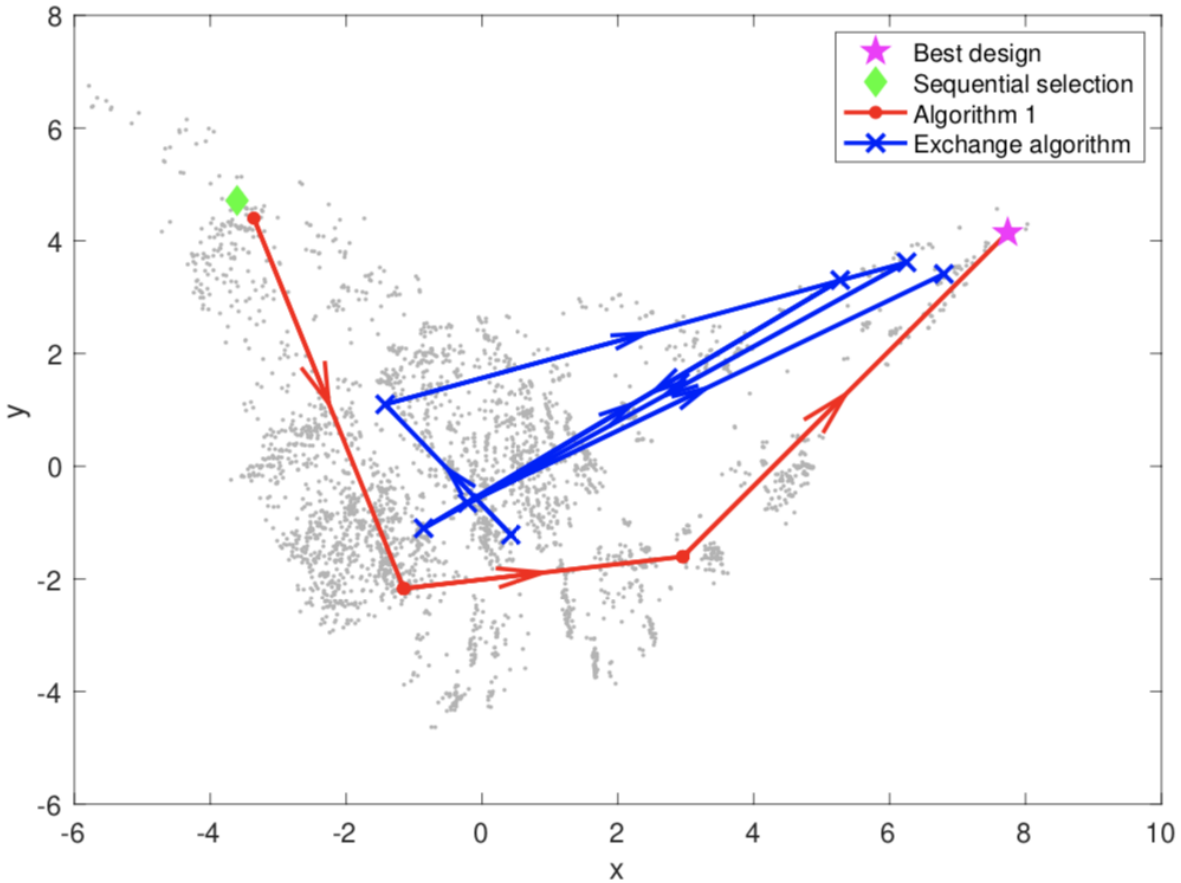
Finding useful designs

Algorithm 1: Search for designs by Bayesian optimisation.

Result: Design D^+ with the largest information gain I^+.
Iteration $t = 0$;
$\Delta I^+ = 1$;
Evaluate I for m_0 randomly selected designs $D_{t,(1)}, \ldots, D_{t,(m_0)}$ to get $I_{t,(1)}, \ldots, I_{t,(m_0)}$;
$I^+ = \max\{I_{t,(1)}, \ldots, I_{t,(m_0)}\}$;
$\mathcal{F}_t = \{(I_{t,(j)}, D_{t,(j)}); j = 1, \ldots, m_0\}$;
while $t \leq T_{max}$ *or* $\Delta I^+ = 0$ **do**
  $t = t + 1$;
  Mix existing design sites and random sites to suggest M designs ;
  Compute the Hausdorff distances for the suggested and available designs ; ▷ `expression` (7)
  Fit a GP model for I given all evaluations ; ▷ `expression` (6)
  Compute EI over I^+ for each of the M design ; ▷ `expression` (16)
  Find the m designs with largest EI to obtain $D_{t,(1)}, \ldots, D_{t,(m)}$;
  Evaluate $I(D_{t,(1)}), \ldots, I(D_{t,(m)})$; ▷ `expression` (5)
  $I^+ = \max\{I^+, I_{t,(1)}, \ldots, I_{t,(m)}\}$, $D^+ = \{D; I(D) = I^+\}$;
  $\mathcal{F}_t = \mathcal{F}_{t-1} \cup \{(I_{t,(1)}, D_{t,(1)})\} \cup \ldots \cup \{(I_{t,(m)}, D_{t,(m)})\}$;
  Compute an average increase over the last buffer of iterations ΔI^+;
end

2^n possible designs in set $D = \{D_0, \ldots, D_n\}\}$.
Comparing all designs is only feasible in smaller cases. Hausdorff distances (or other) to measure similarity in designs can help guide search for useful designs.

Optimal designs



Representation of the spatial Hausdorff distance for the best 3000 designs (grey

Projected distances - MDS

Datasets $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$, $\boldsymbol{Y}_i = (Y_{i1}, \ldots, Y_{ip})$ for large p.
Embed the dissimilarity of data $D_{ij} = \text{distance}(\boldsymbol{Y}_i, \boldsymbol{Y}_j)$ in a smaller dimension (typically 2) such that close points in the 2 dimensional plane are also close in the p dimensional space.
Idea of Multi-dimensional scaling (MDS) goes back to Torgerson (1950s) and Kruskal (1960-70s).

MDS mathematics

$$\text{Stress}(x_1, \ldots, x_n) = \sqrt{\sum_{i \neq j} (D_{ij} - |x_i - x_j|)^2}$$

Find $x_1, \ldots, x_n$, locations in 2 dimensional space that best **visualize differences and clusters in the data**. (x attributes are centered at the origin.)

MDS plot in 2 dimensions

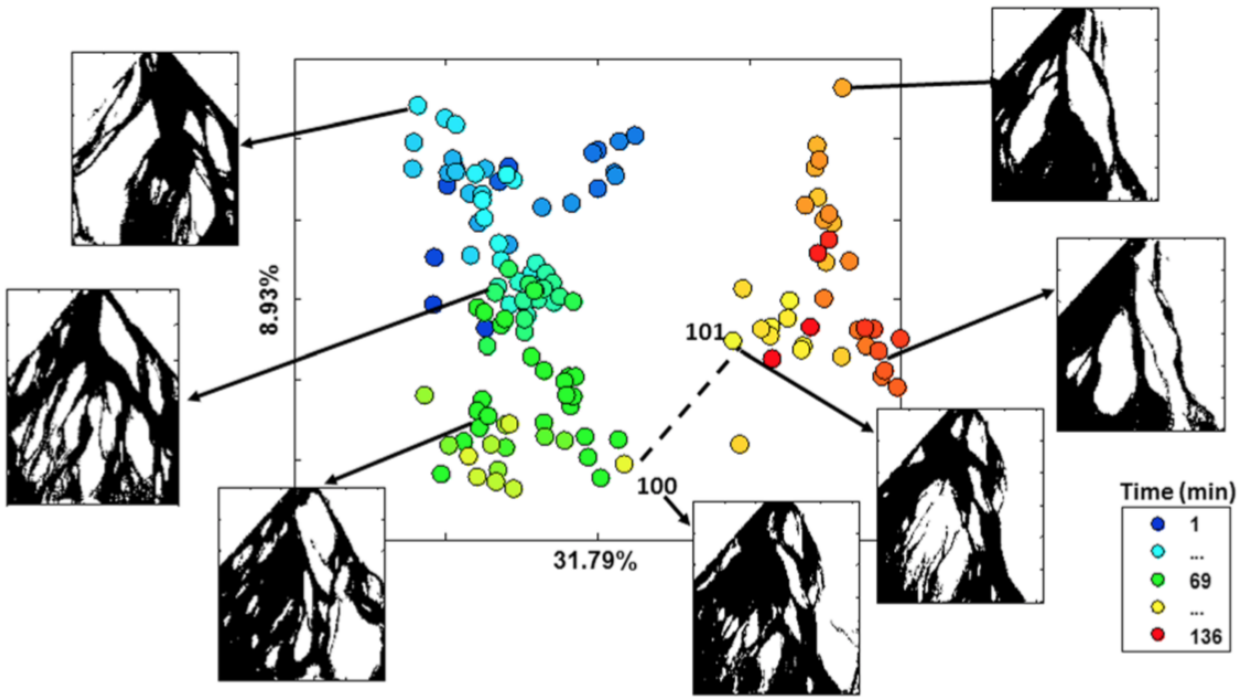


Figure: MDS of physical modeling data of drainage data (Scheidt et al. (2017).

MDS algorithm

Gradient descent is one method to solve for x_i, $i = 1, \ldots, N$.

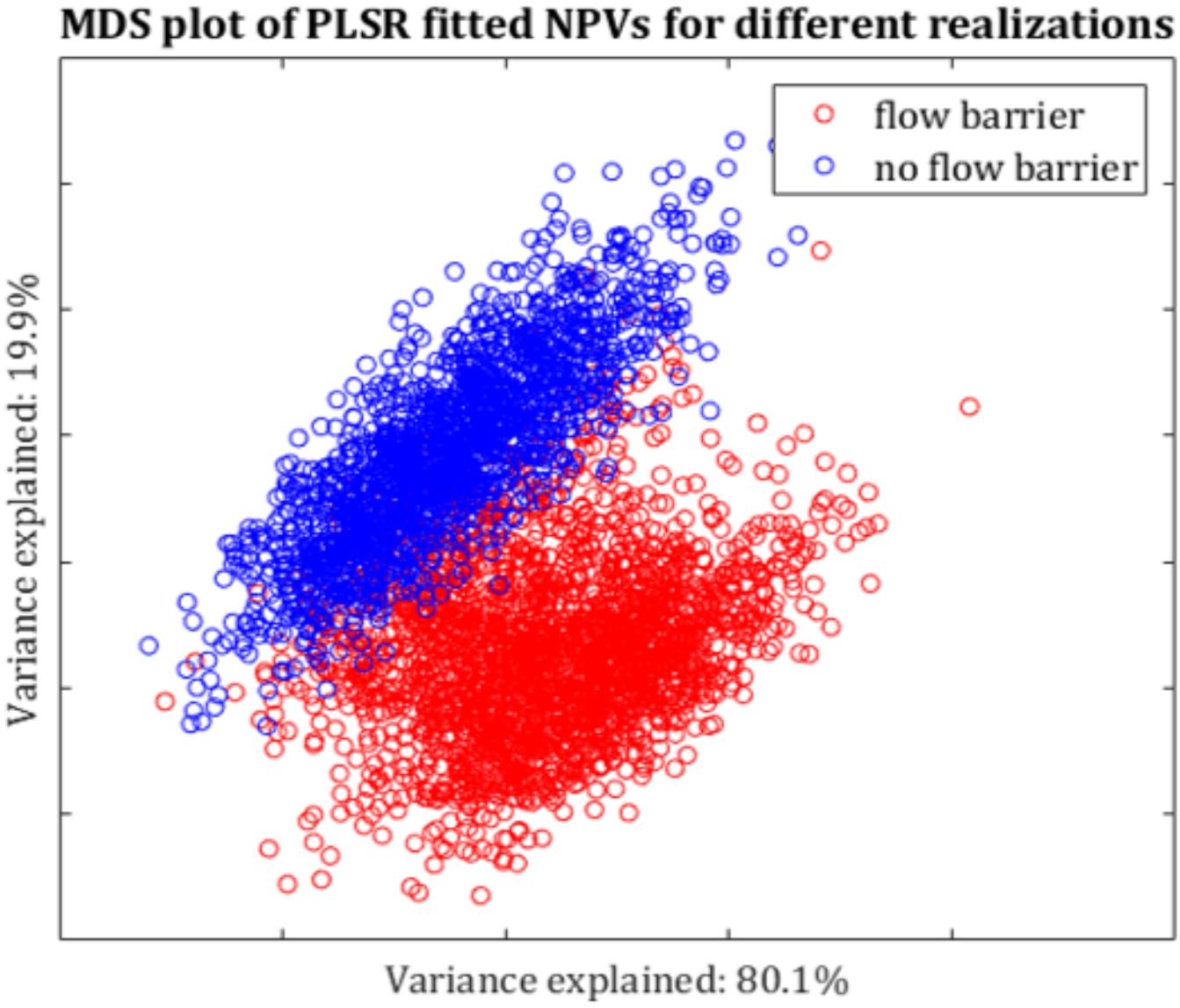
$$\text{argmin}_{x_1,\ldots,x_N} \sqrt{\sum_{i \neq j} (D_{ij} - |x_i - x_j|)^2}$$

Find $x_1, \ldots, x_N$, locations in 2 dimensional space that best visualize differences in the data.
Actual implementation depends on distance measures (Sect 6.2 in Buja et al.)

MDS extensions

- Dissimilarities can be metric or nonmetric distance between data inner-products (Sect 4.2 in Buja et al.)
- Dissimilarities use PCA (or PLS) with smoothing kernels.
- Dissimilarities based on neighborhood embeddings (tSNE), using conditional probabilities, kernels with heavy tails and divergence measures.

MDS for datasets



Dynamic time warping (DTW)

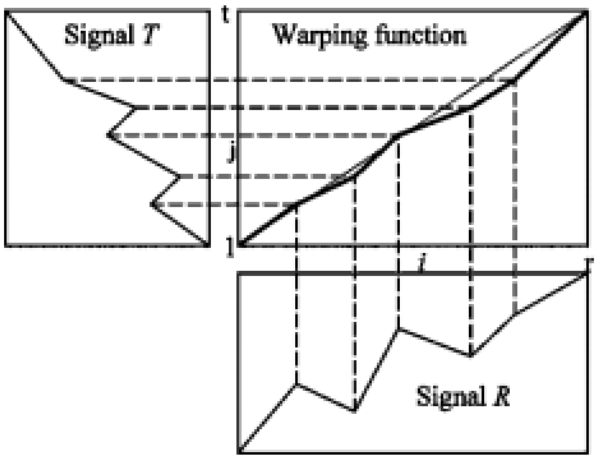


Figure: Illustration of warping function

Used a lot in speech recognition.

Gas pipe data

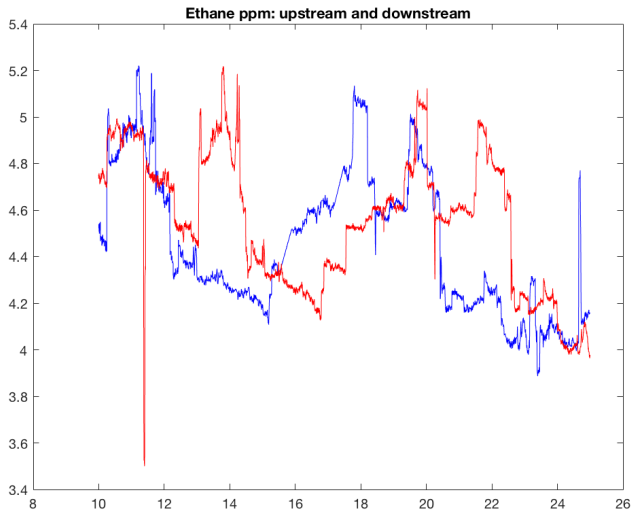


Figure: Gas-pipe ethane measured in Norway, and in Germany.

Must align time series.

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 $(i, p(i))$.

Optimization problem

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

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

Aligning sequences

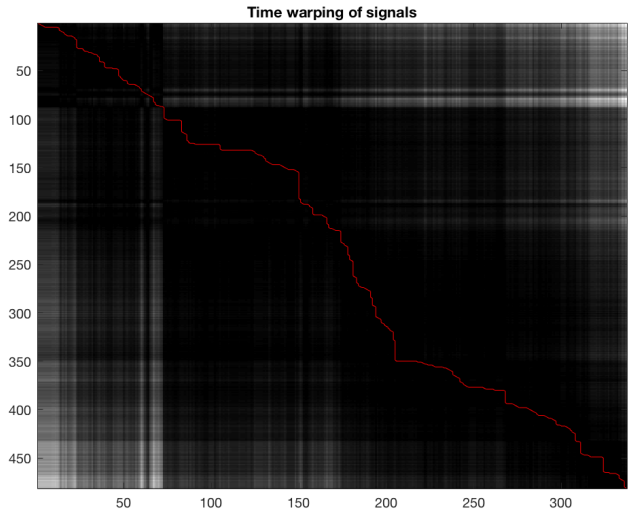


Figure: Alignment matrix for ethane time series data, and optimal warping path.

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

$\hat{p}$ er path som minimerer distansen D_p.

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

Recursive optimization

- Viterbi algorithm
- Sequential optimization

Examples of dynamic programming (DP).
Approximate dynamic programming and reinforcement learning.

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

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

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

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

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

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

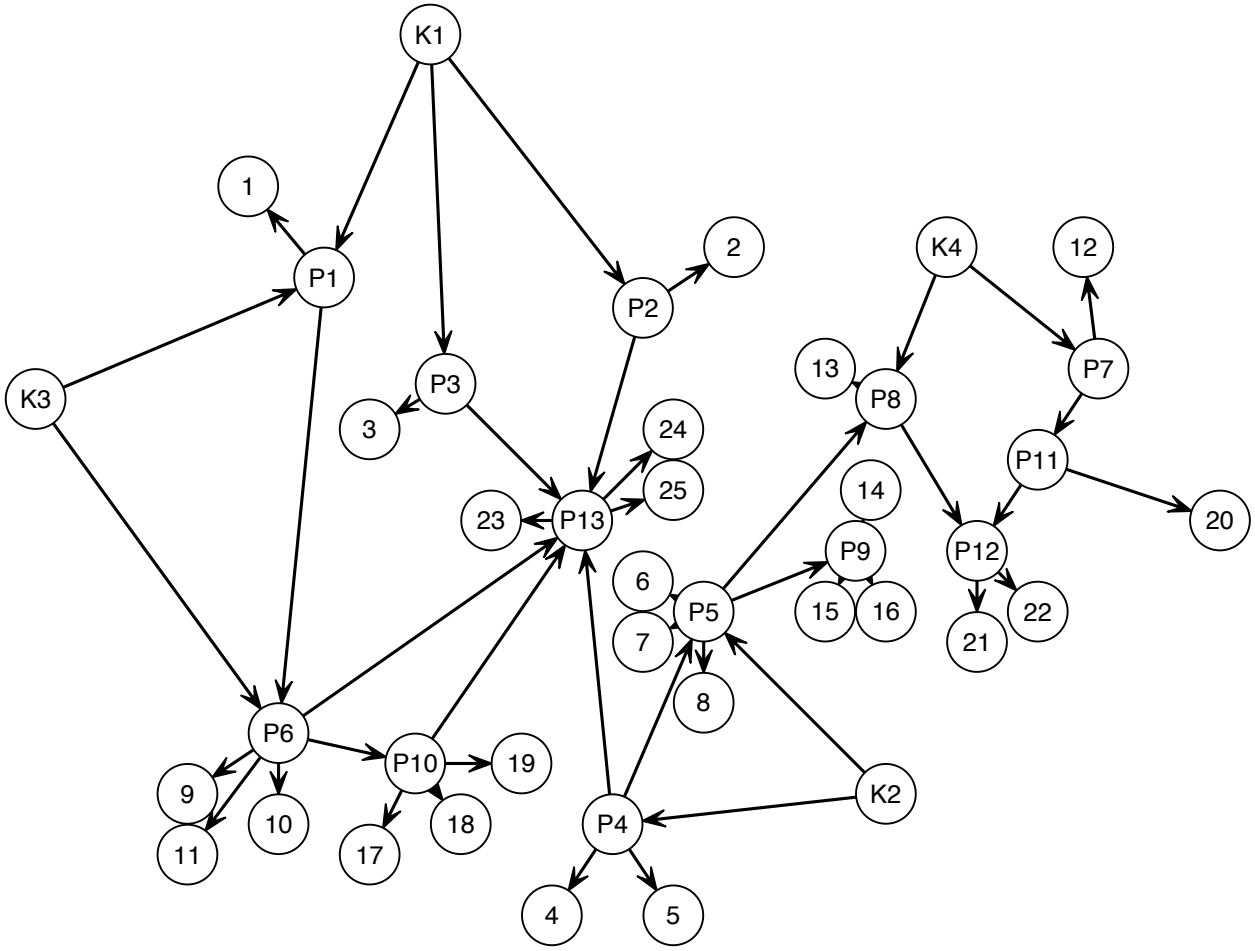
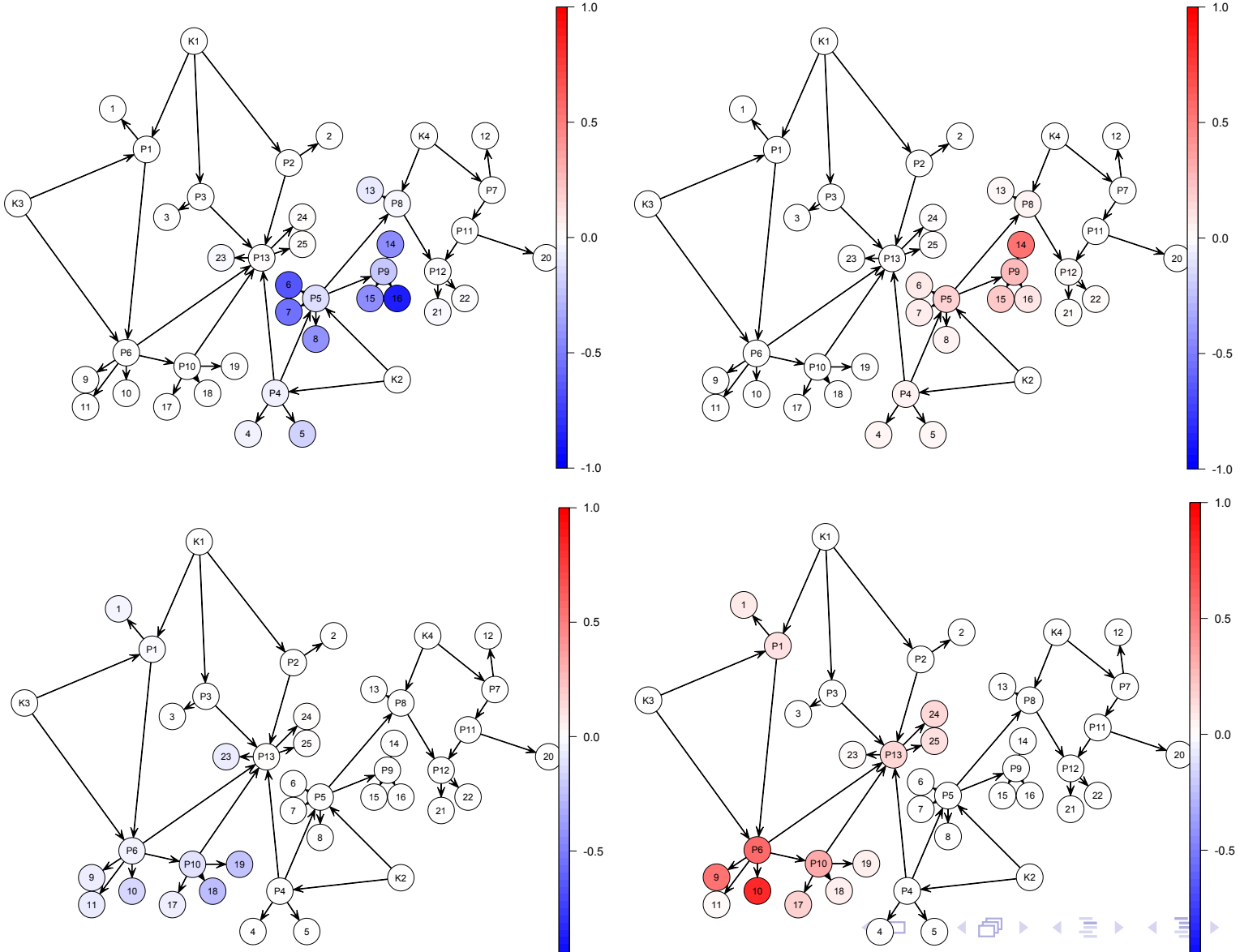


Figure: Network of 25 drilling prospects, identified with the nodes from 1 to 25, where we can possibly drill.

Evidence propagation



Sequential decisions

Value is defined by nested equations:

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

- 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 $x_i \in \{1, \ldots, k\}$, 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:

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

Other heuristics

- Look-ahead stategies account for the next stages, but not all future rewards.
- Rolling-horizon look-ahead strategies, 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

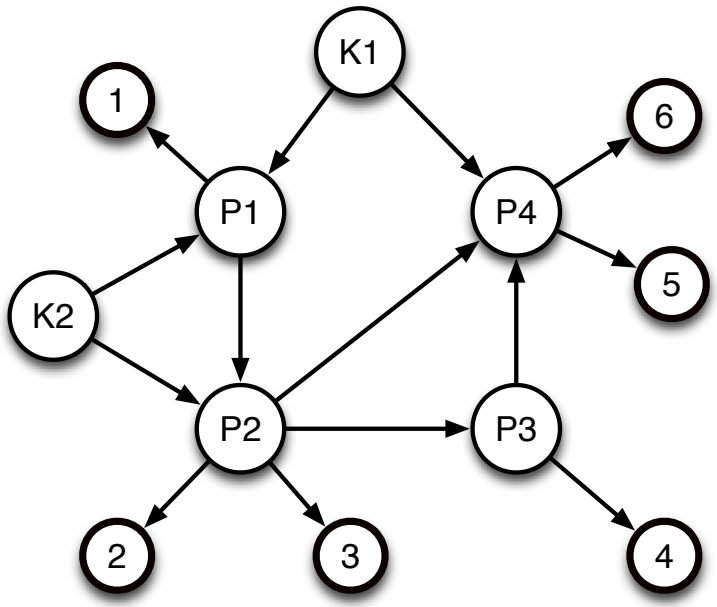


Figure: BN used for small case. N=6.

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).