Different Similarity Measures

$$f^p, f^q : timeseries$$

Manhattan Distance

- Quantifies the absolute magnitude of the difference between time series
- Easy to calculate

$$D_{Man} = (\sum_{t=1}^{N} |f_t^p - f_t^q|)$$

- Implementation: scipy.spatial.distance.cityblock

Euclidean

- Quantifies the Euclidean distance ofthe difference between time series
- Easy to calculate
- More sensitive to outliers, due to it's nonlinear character

$$D_E = \sqrt{(\sum_{t=1}^{N} |f_t^p - f_t^q|^2)}$$

- Implementation: scipy.spatial.distance.euclidean

Mahalanobis

- Quantifies the difference between time series but accounts for non-stationarity of variance and temporalcross-correlation

$$D_{Mah} = \sqrt{(f_t^p - f_t^q)^T \sum^{-1} (f_t^p - f_t^q)}$$

$$\sum : CovarianceMatrix$$

- Implementation: scipy.spatial.distance.mahalanobis

Pearons's Correlation

- Quantifies the degree of linear relationship between time series

$$D_{CC} = \frac{\sum_{t=0}^{N-1}((f_t^p - \overline{f^p}) * (f_{t-s}^q - \overline{f^q}))}{\sqrt{\sum_{t=0}^{N-1}(f_{t-s}^p - \overline{f^p})^2} * \sqrt{\sum_{t=0}^{N-1}(f_{t-s}^q - \overline{f^q})^2}}$$

- Implementation: numpy.corrcoef

Cosine Distance

- Metric used to determine how similar time series are irrespective of their magnitudes.

$$D_{Cos} = \frac{\sum_{i=1}^{n} f_i^p f_i^q}{\sqrt{\sum_{i=1}^{n}(f_i^p)^2}\sqrt{\sum_{i=1}^{n}(f_i^q)^2}}$$

- Implementation: scipy.spatial.distance.cosine

Principal Component Distance

- Quantifies the difference between times PCs that explain the majority of the variance
- Selecting m critical

$$D_{PCA} = \sqrt{\sum_{k=1}^{m}(PC_k^p - PC_k^q)^2}$$

- Implementation: sklearn.decomposition.PCA

Mutual Information

- Measure of the amount of mutual dependence between two random variables

$$MI(f^p, f^q) = -\sum_{f_i^p, f_i^q} p(f_i^p, f_i^q) log_2 \frac{p(f_i^p, f_i^q)}{p(f_i^p)p(f_i^q)}$$

- Implementation: pyinform.mutualinfo.mutual_info

Transfer Entropy

- Quantify information transfer between an information source and destination, conditioning out shared history effects

$$T_{f^p->f^q} = H(f_t^q | f_{t-1:t-L}^q) - H(f_t^q | f_{t-1:t-L}^q, f_{t-1:t-L}^p)$$

- Implementation: pyinform.transferentropy.transfer_entropy

Conditional Entropy

- Measure of the amount of information required to describe a random variable f^q given knowledge of another random variable f^p

$$H(f^q | f^p) = -\sum_{f_i^p, f_i^q} p(f_i^p, f_i^q) log_2 \frac{p(f_i^p, f_i^q)}{p(f_i^p)}$$

- Implementation: pyinform.conditionalentropy.conditional_entropy

Dynamic Time Warping

- Dynamic time warping is an algorithm used to measure similarity between two sequences which may vary in time or speed.
- It works as follows:
  1. Divide the two series into equal points.
  2. Calculate the euclidean distance between the first point in the first series and every point in the second series. Store the minimum distance calculated. (this is the ‘time warp’ stage)
  3. Move to the second point and repeat 2. Move step by step along points and repeat 2 till all points are exhausted.
  4. Repeat 2 and 3 but with the second series as a reference point.
  5. Add up all the minimum distances that were stored and this is a true measure of similarity between the two series.
- Implementation: mlpy.dtw_std