A Primer on Time Series Analytics
Landnutzungsklassifikation WS 18/19

Patrick Schäfer

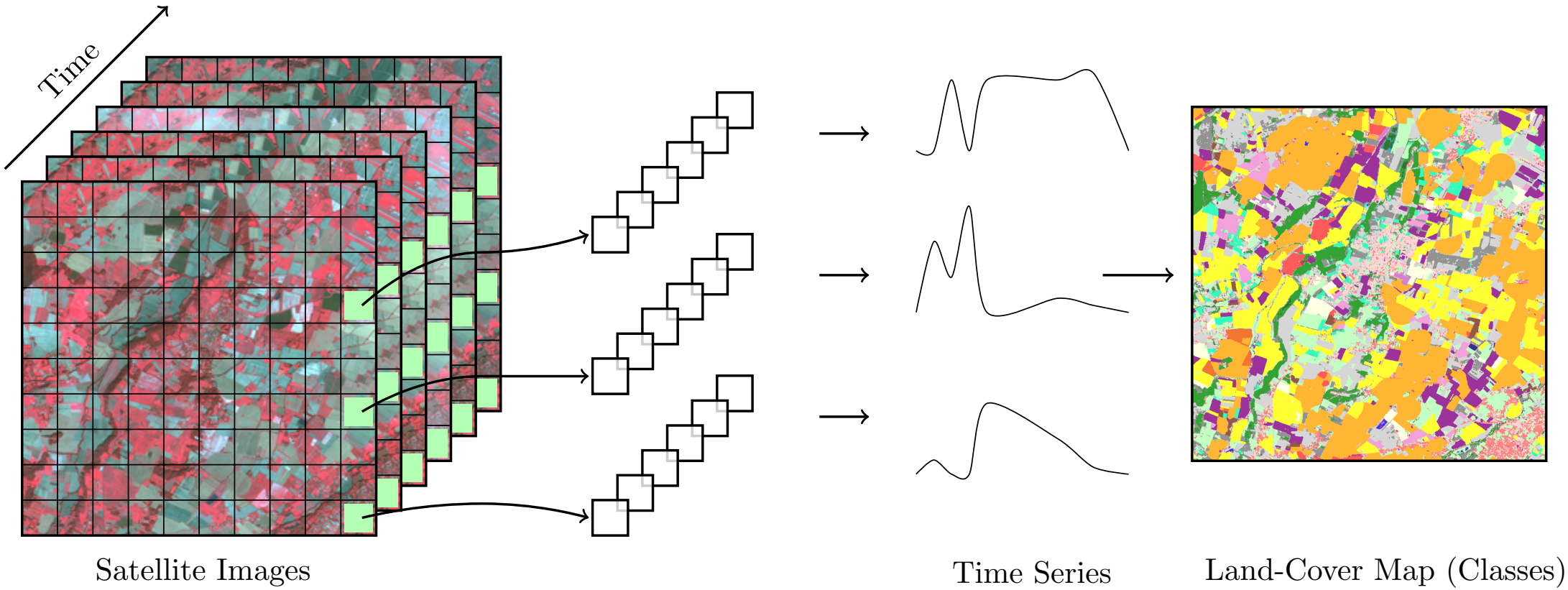
URL: https://hu.berlin/landnutzung

Agenda

- Time Series
- Pre-Processing
- Representations & Classifiers
  - Whole-Series
  - Shapelets
  - Dictionary (Bag-of-Patterns)

- Next steps:
  - **Today: choose a topic**
  - Before **30.11.18**: meet me to discuss topic
  - **07.12.18 15-16 Uhr:** Flash presentation, **RUD 25 4.410**
    - Present ideas and your topic in 5min

From satellite images to Land Cover Classification



Tan, Chang Wei, Geoffrey I. Webb, and François Petitjean. "Indexing and classifying gigabytes of time series under time warping." *Proceedings of the 2017 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2017.

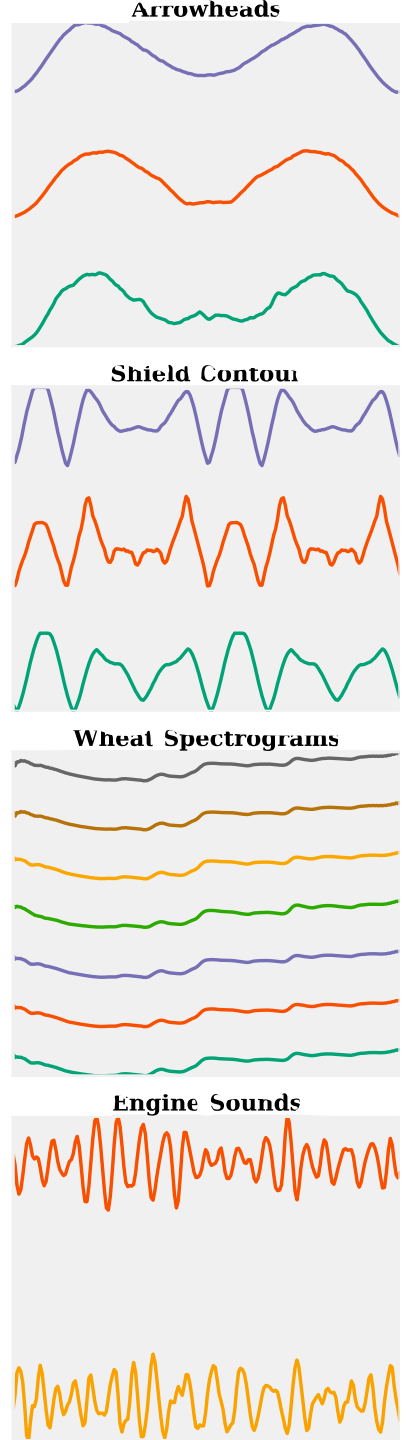
Time Series Definition

- Definition: A Time Series is a sequence (ordered collection) of n real values at time stamps $(\boldsymbol{t_1}, \ldots, \boldsymbol{t_n})$:

$$\boldsymbol{T} = (\boldsymbol{y_1}, \ldots, \boldsymbol{y_n})$$

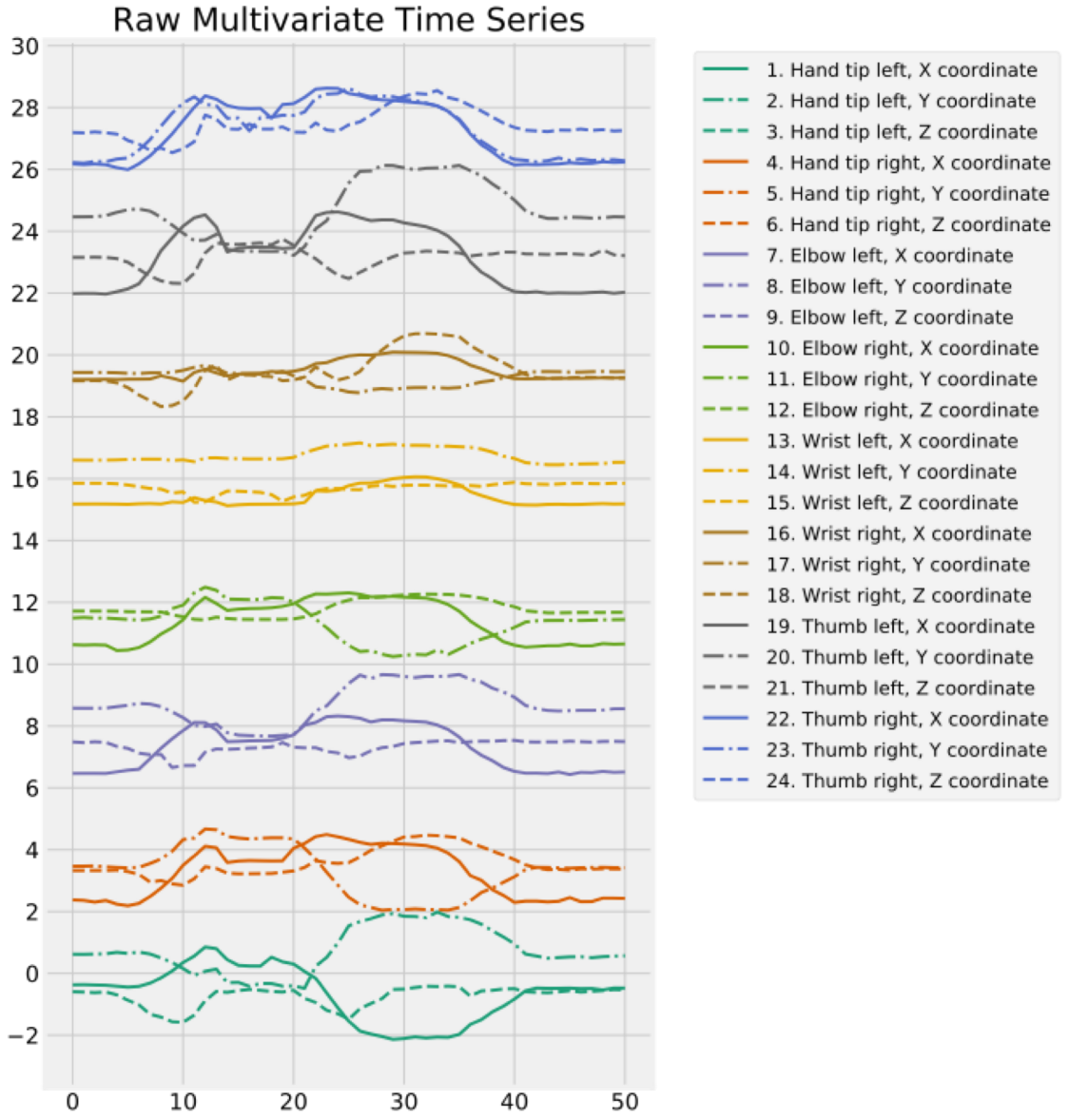
- Time Series may be univariate or multivariate
  - Univariate: a single value y_i is associated with each time stamp t_i.
  - Multivariate: m values $y_i = (k_1, \ldots k_m)$ are associated with each time stamp t_i.

- The dimensionality of a time series refers to the number of values at each time stamp



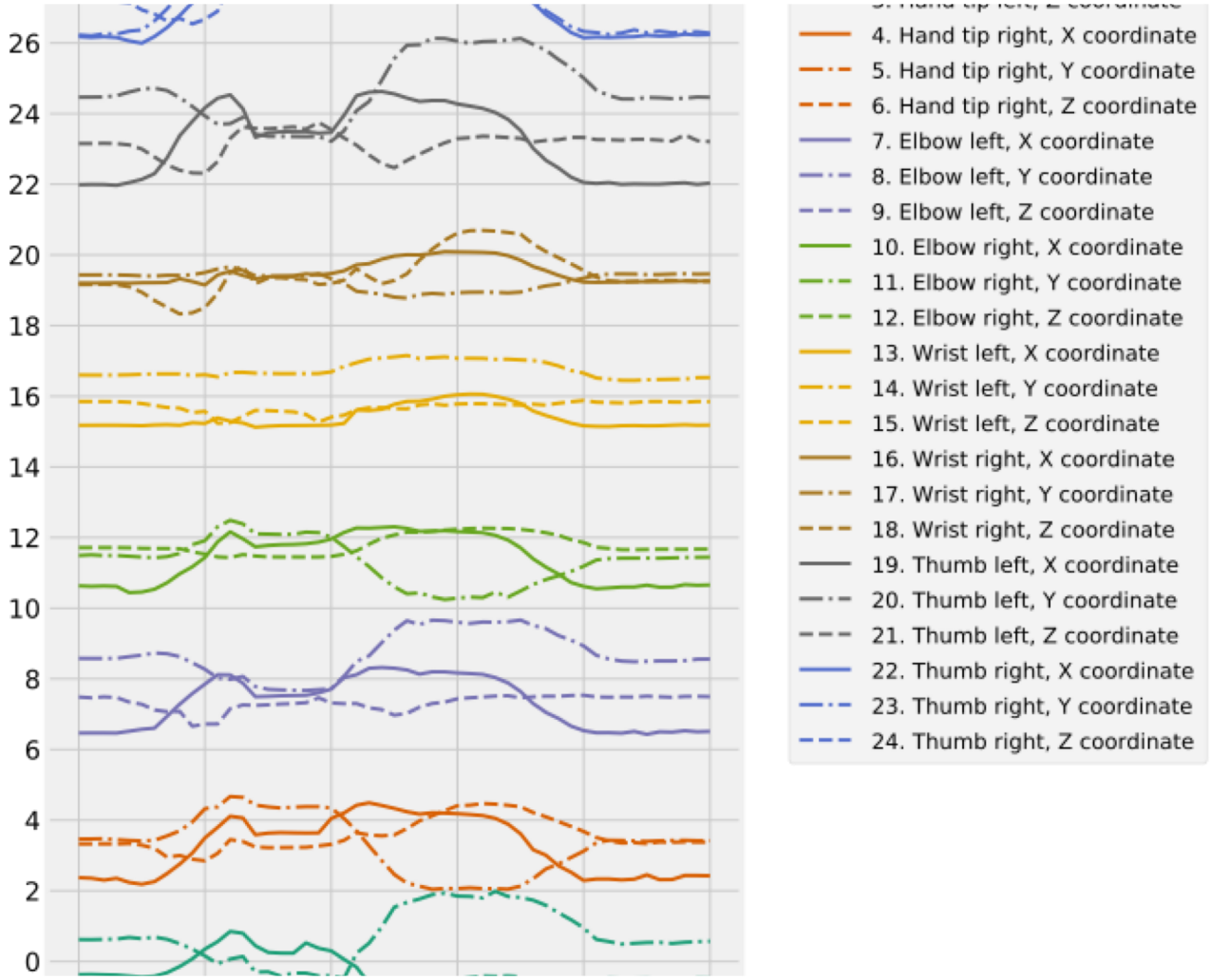
Multivariate Time Series (MTS)

- Arise when interconnected streams record data over time:
  - Weather observations: humidity, temperature
  - Satellite images: different reflectances (sensors)
  - Gesture recognition of users performing isolated gestures: 8 sensors recording x/y/z coordinates (Figure to the right)



MTS are characterized by

1. Interplay of dimensions:
   Individual features vs. interplay of features in different dimensions
2. Phase invariance:
   Signals may not be synchronized in time / characteristic features may appear anywhere
3. Irrelevant data/dimensions:
   Only small periods in time and in a few dimensions may contain relevant information



Pre-processing

Pre-processing: Normalization

- Time series need to be normalized, especially when different kinds of sensors are used to record the data
- Normalization puts the data on the same scale to make comparisons meaningful (i.e. Fahrenheit and Celsius)
- Two methods are commonly used:
  - Range-based normalization:
    preserves relationship between samples
  - Z-normalization:
    looses relationship between samples,
    can be useful when min/max values are unknown

Pre-processing: Range-based normalization

- Range-based normalization:
  The minimum and maximum values over all time series $T \epsilon D$ are determined, then each value of a time series $\boldsymbol{T} = (\boldsymbol{y_1}, \ldots, \boldsymbol{y_n})$ is mapped to range [0,1] by:

$$range\_norm(T) = \left(y'_1, \ldots, y'_n\right)$$

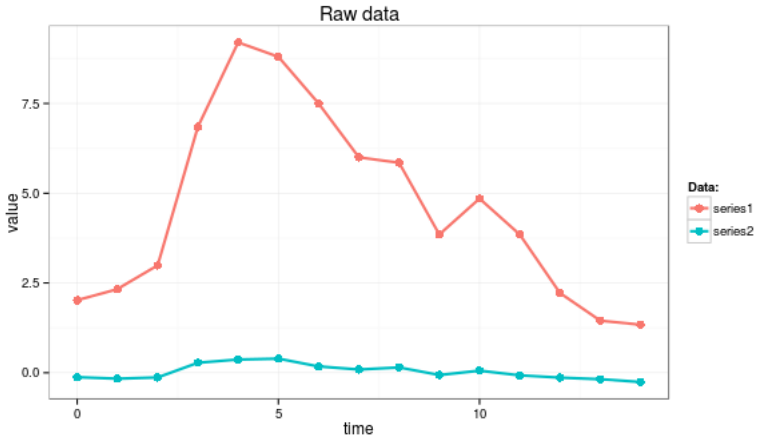
$$\text{with } y'_i = \frac{y_i - min}{\max - min}$$

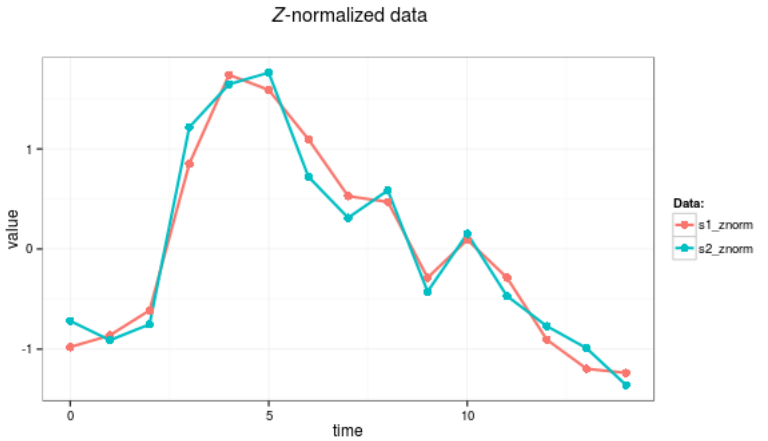
Pre-processing: Z-Normalization

- Zero-mean-normalization:
  Let μ and σ be the mean and standard deviation of the time series $\boldsymbol{T} = (\boldsymbol{y_1}, \ldots, \boldsymbol{y_n})$, then

$$znorm(T) = \left(y'_1, \ldots, y'_n\right)$$

with $y'_i = \frac{y_i - \mu}{\sigma}$





https://jmotif.github.io/sax-vsm_site/morea/algorithm/znorm.html

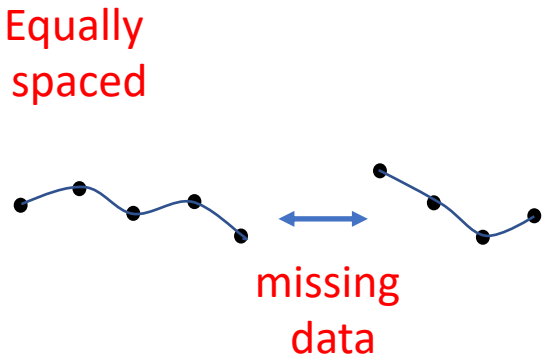
Pre-Processing: Missing Data

- It is common for time series to contain missing data
- Possible directions:
    1. Remove all records with missing entries, which is not practical when all data contains missing values
    2. Impute missing values, but imputation error affects overall classification accuracy
    3. Use a model that works with missing data

Imputation Methods

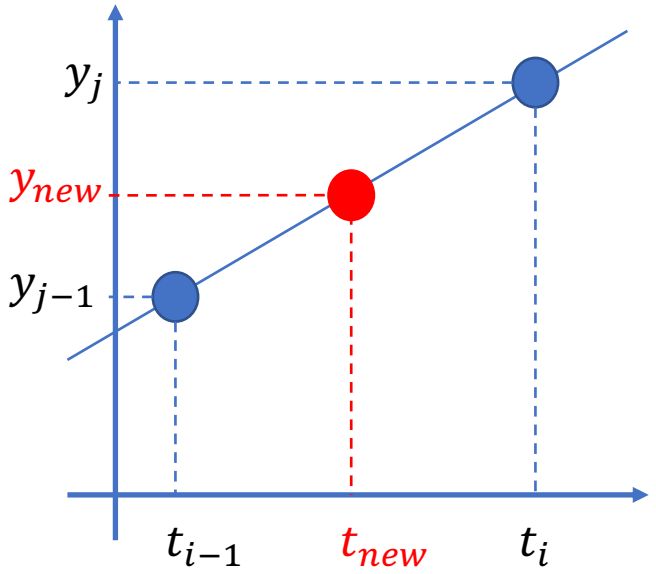
- mean,
- median,
- last observation carried forward,
- next observation carried backward,
- spline interpolation,
- linear interpolation

Pre-processing: Imputation



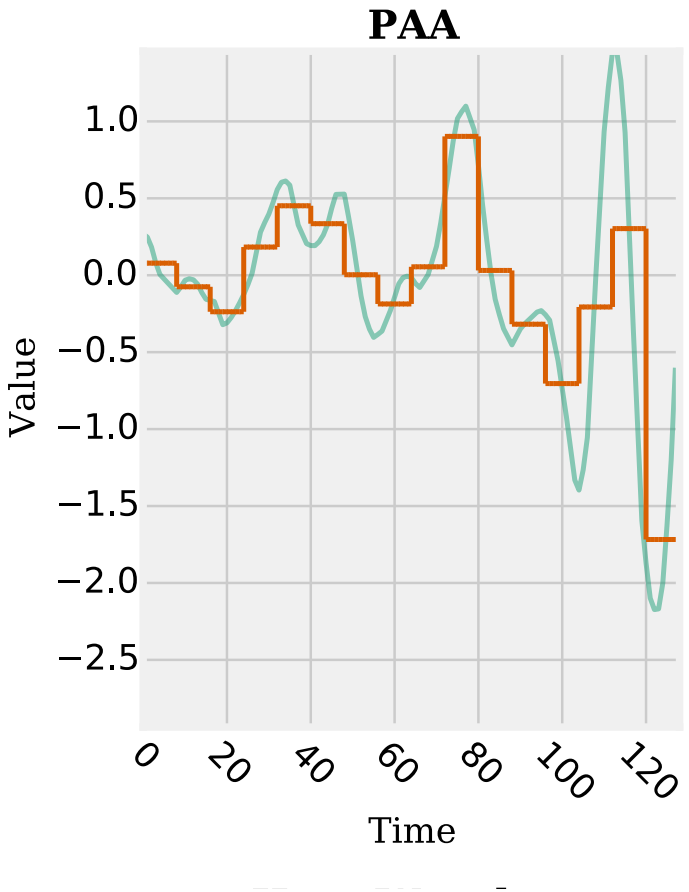
- Linear interpolation estimates the (missing) values at the desired time stamps by fitting a line
- Linear interpolation: y_{j-1} and y_j are values of the time series at times t_{i-1} and t_i

$$y_{new} = y_{j-1} + \frac{(t_{new} - t_{i-1}) \cdot (y_j - y_{j-1})}{t_i - t_{i-1}}$$



Pre-processing: Noise Removal

- Remove short-term fluctuation and noise
- Binning (Averaging):
  - Divide the data into *disjoint* intervals of size k. Then calculate the mean values $\mathrm{T} = (\bar{y}_1 \ldots \bar{y}_w)$, with $w = \frac{n}{k}$, in each interval: $\bar{y}_i = \frac{\sum_{j=k(i-1)+1}^{ki} y_j}{k}$
  - Reduces the number of points by a factor of k
- Smoothing (Moving-Averages)
  - Divide the data into *overlapping* intervals of size k over which the averages are calculated
  - Thus, the average is computed at each time stamp $[t_1, t_k], [t_2, t_{k+1}], \ldots$ rather than only at the interval intersections $[t_1, t_k], [t_{k+1}, t_{2k}], \ldots$



Time Series Analytics

Representations and Classifiers

Time Series Approaches

- Time series approaches are composed of
  - a time series representation, and
  - a classifier

- Representations can be divided into:
  - Global: Using the whole time series
  - Local: Using sub-sequences
    - Shapelets: absence or presence of characteristic substructures
    - Bag-of-Patterns (Dictionaries): frequency of occurrences of substructures

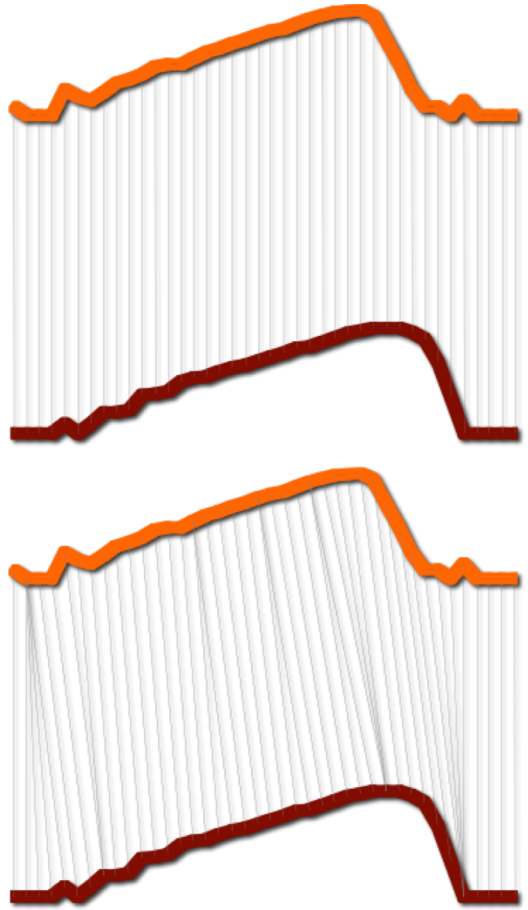
- Any base-classifier can then be trained on this feature space (embedding)

A List of Approaches

- **(non-time series) based-Classifiers**
  - SVM, logistic regression, random forests/decision trees, gradient boosting trees, XGBoost
- **Whole-Series-based Classifiers**
  - 1-NN Dynamic Time Warping
  - 1-NN Euclidean Distance
  - Proximity Forests
- **Shapelet-based Classifiers**
  - Univariate**:** Fast Shapelets (FS), Learning Shapelets (LS), Shapelet Transform (ST)
  - Multivariate**:** gRSF
- **Dictionary-based Classifiers**
  - Univariate**:** BoP, SAX VSM, TSBF, BOSS, BOSS VS, WEASEL
  - Multivariate**:** SMTS, WEASEL+MUSE, LPS
- **Deep Learning Classifiers**
  - ResNet**,** FCN, Encoder, MLP, Time-CNN, TWIESN, MCDCNN, MCNN, t-LeNet
- **Ensembles of Core Classifiers**
  - Univariate: EE PROP, COTE

1) Whole Series

- Compares two whole time series
- The *similarity* of time series Q and T is expressed by a real value using a *distance measure*: $D(Q,T) \rightarrow \mathbb{R}_0^+$
- A *similarity measure* is the inverse of the distance measure: it qualifies *similar (/dissimilar)* time series by a *small (/large)* value
- Most common methods are *Euclidean distance (ED)* and *Dynamic Time Warping Distance (DTW)*
- Others: Longest Common Subsequence or Edit Distance



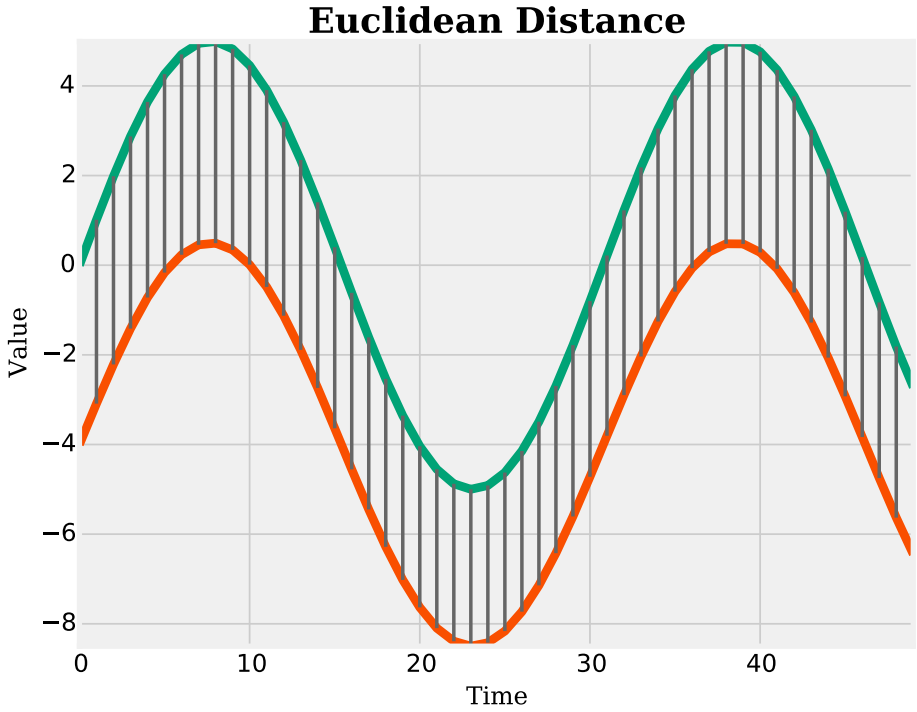
Euclidean Distance

DTW

1) Whole Series: Euclidean Distance (ED)

- Definition: The *Euclidean distance* between two time series $Q = (q_1, \ldots, q_n)$ and C = $(c_1, \ldots, c_n)$, both of length n, is defined as:

$$D_{ED}(Q, C) = \sqrt{\sum_i (q_i - c_i)^2}$$

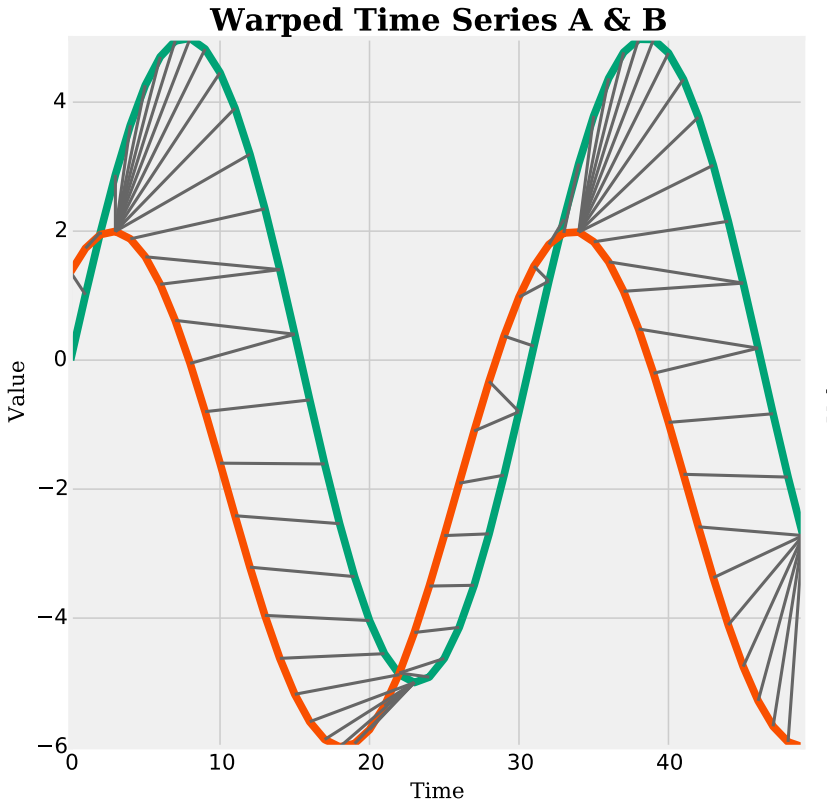


- The ED applies a linear alignment of the time axis
- ED cannot cope with variable length time series
- ED runtime is O(n)

1) Whole Series: Dynamic Time Warping (DTW)

- *Dynamic Time Warping* applies an elastic transformation of the time axis to detect similar shapes that have a different phase
- This is essentially a peak-to-peak and valley-to-valley alignment of two time series
- Intuition: An extension of the ED, which uses two indices i and j representing both time axis
- Find indices (i,j) such that total distance is minimal:

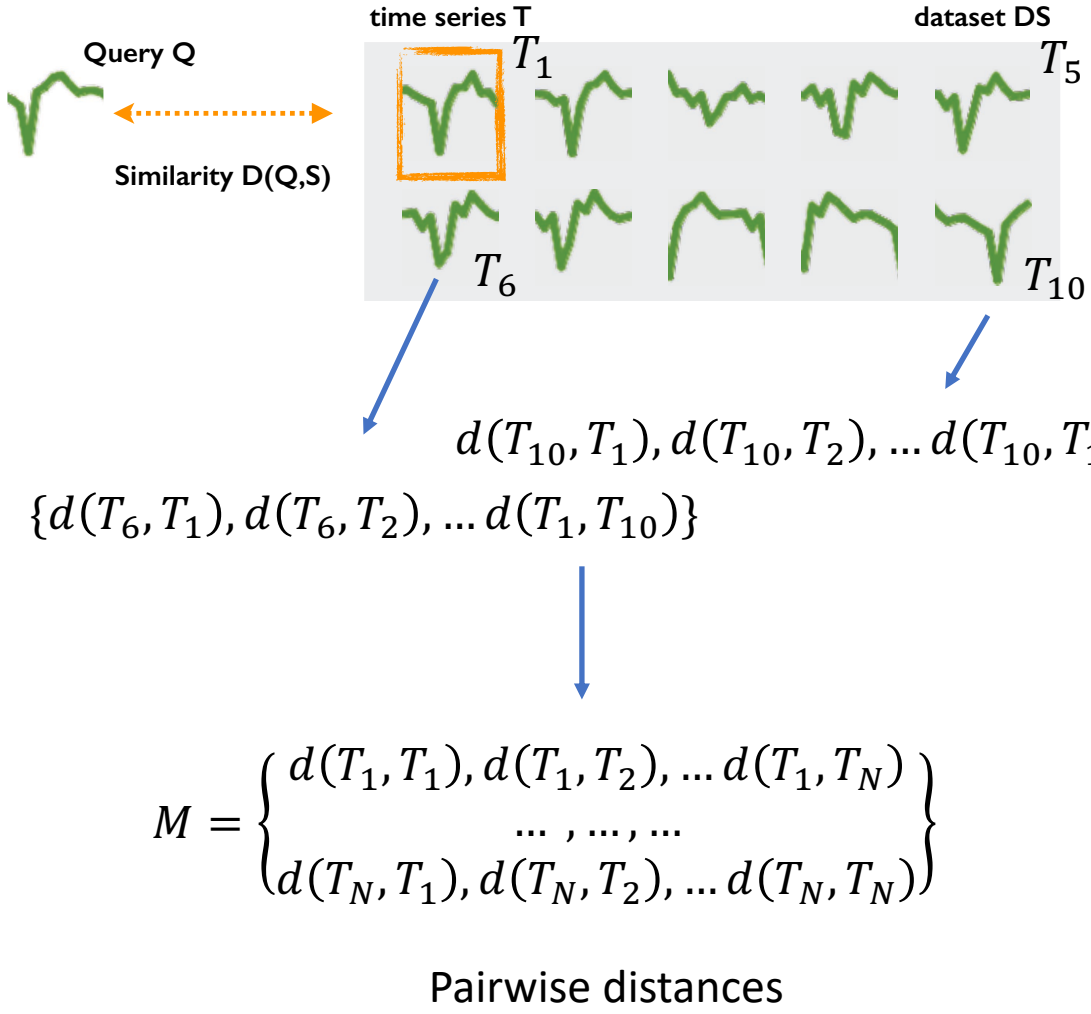
$$D_{DTW}(Q, C) = \sqrt{\sum_{(i,j)} (q_i - c_j)^2}$$



1) Whole Series: Classification

1. Most common: 1-Nearest Neighbour
   - Find the one sample that minimizes distance to the to-be-labelled sample and use it's class label

2. Distance-Space-Embedding + classifier:
   - Build a matrix M on pairwise distances between all time series in D_{train}
   - Train classification model on this M
   - To predict a novel sample S:
     - Compute distance from S to all samples T in D_{train}
     - This results in the feature vector:
       $V = [d(S, T_1), d(S, T_2), \dots d(S, T_N)]^T$
     - Predict the label for this vector V using trained model



Subsequence vs Whole Series

- We wish distinguish between two kinds of plants: what features should one use?
- The contour of a leaf can in fact be interpreted as a time series
- Instead of using the entire shapes, it is better to only compare small subsections
- Here: the defining difference is that Urtica dioica has a stem that connects to the leaf at almost 90 degrees

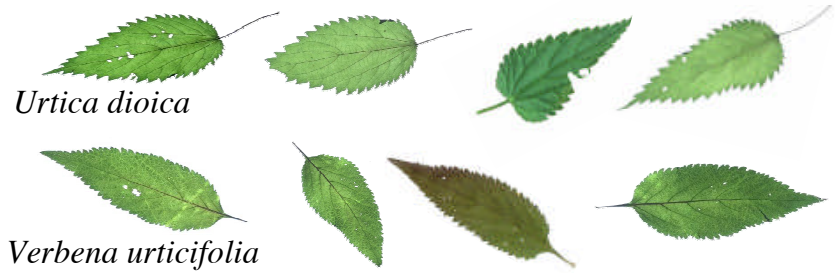


Figure 1: Samples of leaves from two species. Note that several leaves have the insect-bite damage

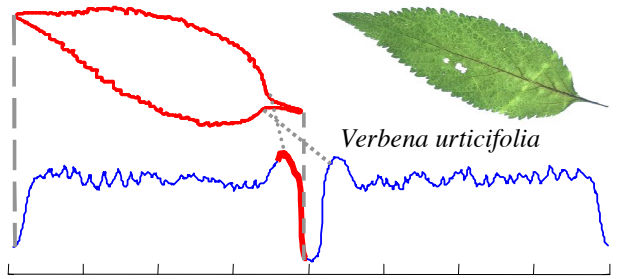
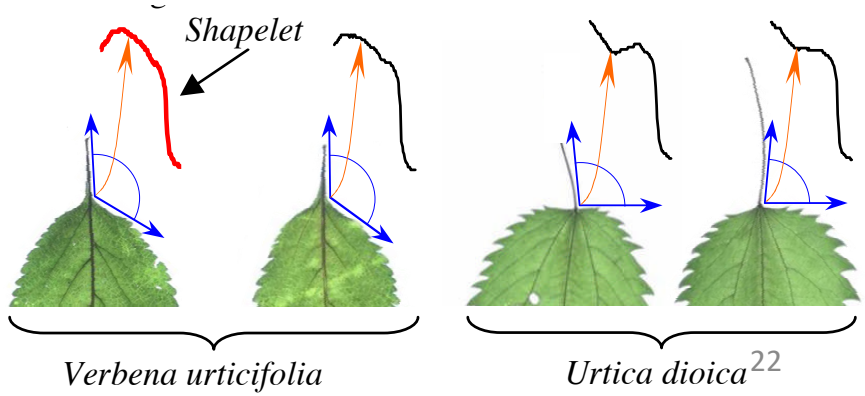


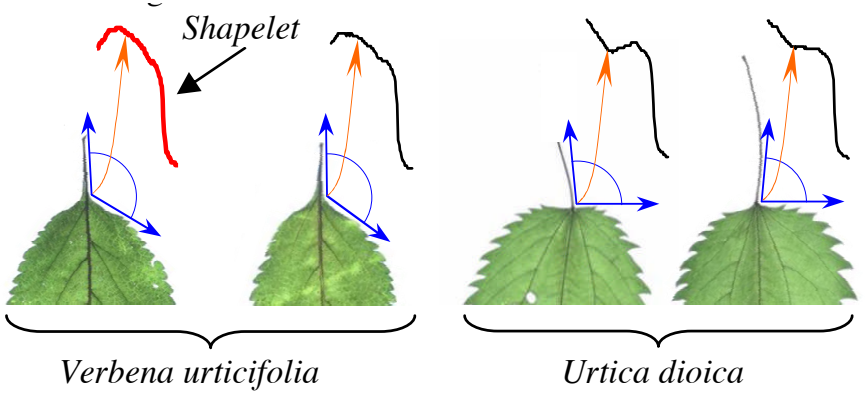
Figure 2: A shape can be converted into a one dimensional "time series" representation. The reason for the highlighted section of the time series will be made apparent shortly



Ye, Lexiang et al.. "Time series shapelets: a new primitive for data mining." *SIGKDD* 2009.

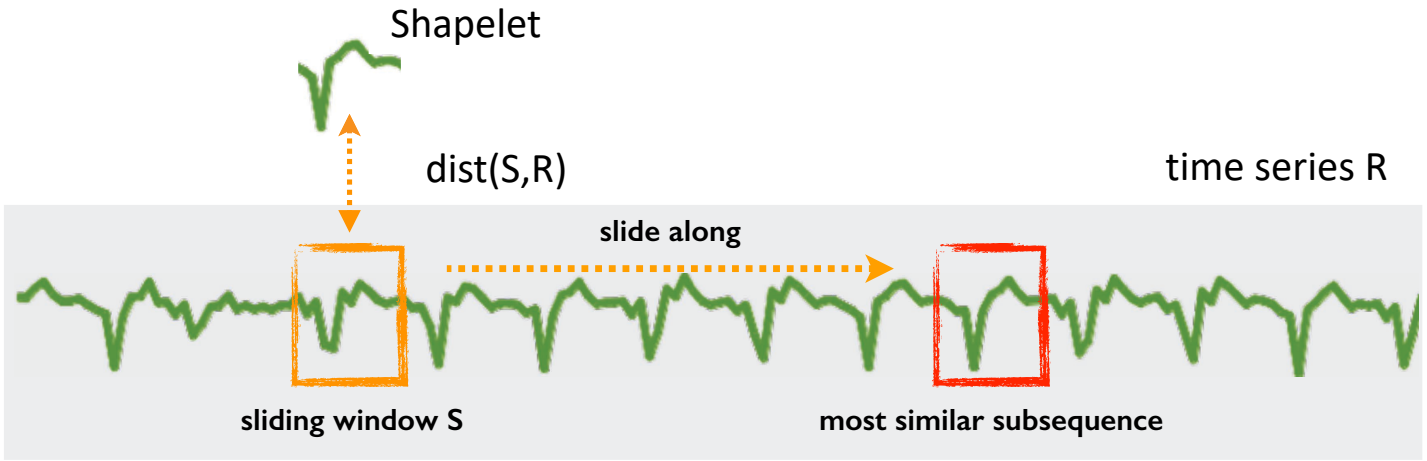
2. Shapelet-based

- Shapelets are time series sub-sequences that are maximally representative of a class label
- Shapelets are interpretable, but training does not scale to large datasets due to high computational complexity (cubic to bi-quadratic in TS length)
- Representatives:
  - Univariate: Shapelet Transform (ST), Learning Shapelets (LS), Fast Shapelets (FS)
  - Multivariate: gRSF



Ye, Lexiang et al.. "Time series shapelets: a new primitive for data mining." *SIGKDD* 2009.

Shapelet Distance



- Measure the distance between any two sub-sequences S and R:

$$dist(S,R) = \sum_{i=1}^{l} (s_i - r_i)^2.$$

- Slide subsequence S over the time series R, and search offset i with minimal distance to S:

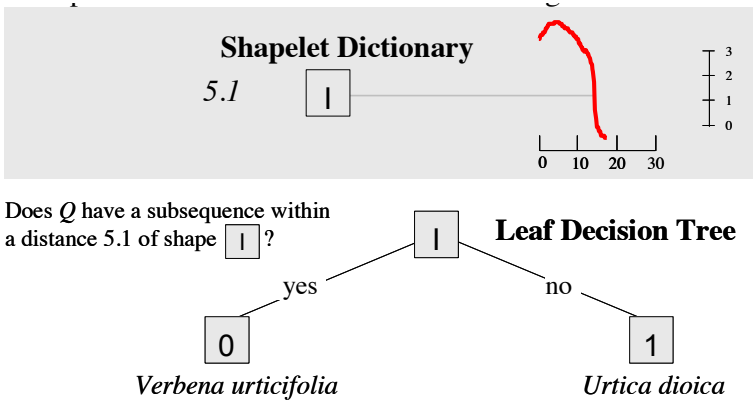
$$d_{i,S} = \min_{R \in W_{i,l}} dist(S,R).$$

- Computational complexity: $O\big(l(n-l)\big)$ for Shapelet length l

2. Shapelet-based: Classification

1. Decision Tree:
   - Branching based on the distance to a Shapelet

2. Shapelet-Distance-Embedding
   - Measure the distance between k Shapelets and each time series in D_{train}
   - Train classification model on this Matrix
   - To predict a novel sample T:
     - Compute distance from T to all k Shapelets
     - This results in the k-dim feature vector: $V = [d(T, S_1), d(T, S_2), \dots d(T, S_k)]^T$
     - Predict the label for this vector V using the trained model



Ye, Lexiang et al.. "Time series shapelets: a new primitive for data mining." *SIGKDD* 2009.

$$M = \begin{Bmatrix} d(S_1, T_1), d(S_1, T_2), \dots d(S_1, T_N) \\ \dots \\ d(S_k, T_1), d(S_k, T_2), \dots d(S_k, T_N) \end{Bmatrix}$$

kxN Matrix of pairwise distances

Shapelet Discovery

```
Shapelet selectShapelet(
     Time Series Dataset D,
     WindowLength min,
     WindowLength max)
    best = 0
    bestShapelet = Ø
    for l = min to max do
        candidates = generateCandidates(D,l)
        for all subsequence S in candidates do
              dist = findDistances(S,D)
              quality = evaluateCandidate(S, dist)
              if quality > best then
                  best = quality
                  bestShapelet = S
    return bestShapelet
```

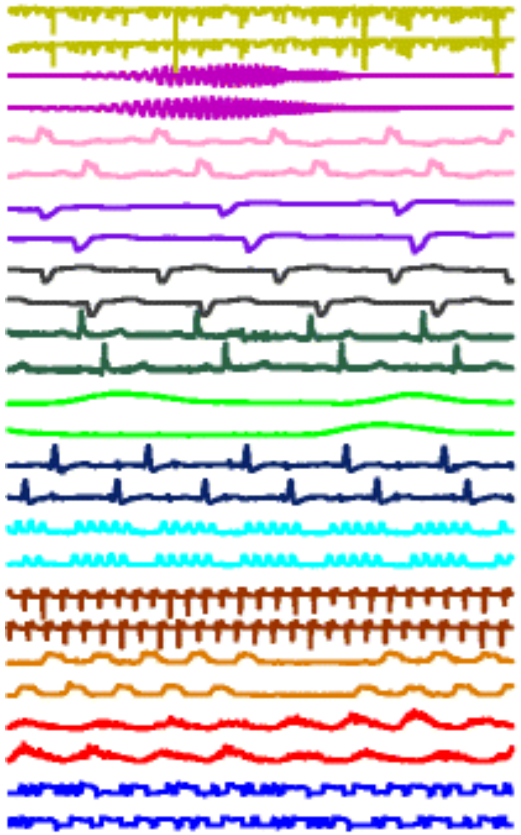
- A shapelet consist of a subsequence of a time series and a distance threshold
- Every subsequence of the time series in the dataset D is a potential candidate
- GenerateCandidates: The Shapelets are found by an exhaustive search of all subsequence lengths between min and max
- The subsequence distance between a shapelet candidate and the dataset is calculated and some measure of quality is used
- Typically one uses the Euclidean distance and Information Gain to measure goodness of a shapelet

Computational Complexity

- The naïve Shapelet discovery algorithm has a computational complexity of $O(N^2n^3)$ for dataset size N and time series length n
  - The size of the candidate set is $\mathrm{O}(Nn^2)$
  - Checking one candidate takes $O(Nn)$ when using the Euclidean distance
- This makes the naïve algorithm infeasible for most real-world problems
- There are optimizations to speed up Shapelet discovery (at the cost of accuracy) by using a random candidate set, random projections, and lower bounding distances

Single occurrence vs frequency of occurrences

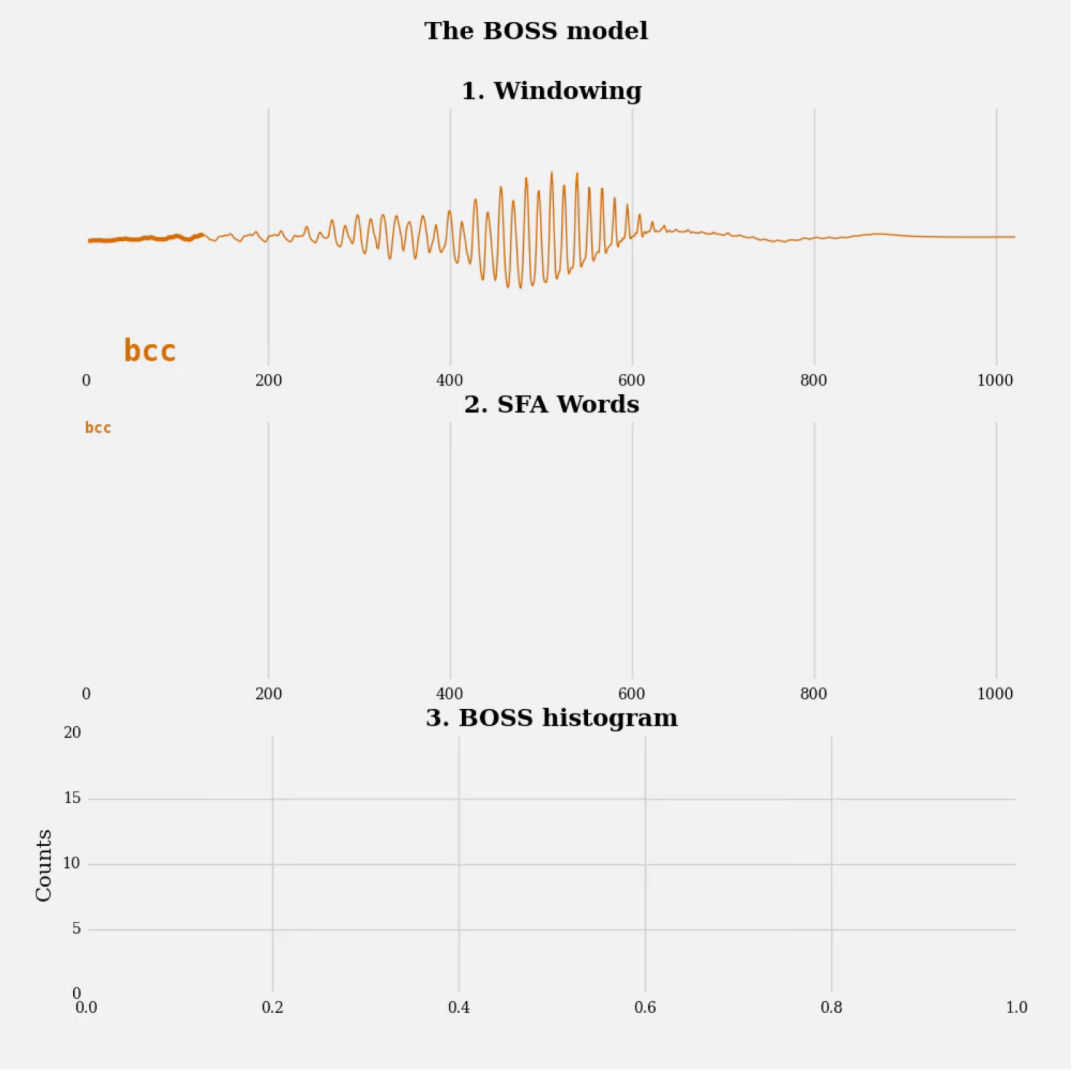
- Many signals are inherently periodic/repetitive (heartbeats, network traffic, weather, …)
- We describe a signal by the frequency of occurrence of patterns
- Similar to the bag-of-words representation for documents, which is a histogram of word counts
- Problem: how to count the occurrences of real-valued-subsequences?



Lin, Jessica, Rohan Khade, and Yuan Li. "Rotation-invariant similarity in time series using bag-of-patterns representation." *Journal of Intelligent Information Systems* 39.2 (2012): 287-315.

3. Dictionary-based

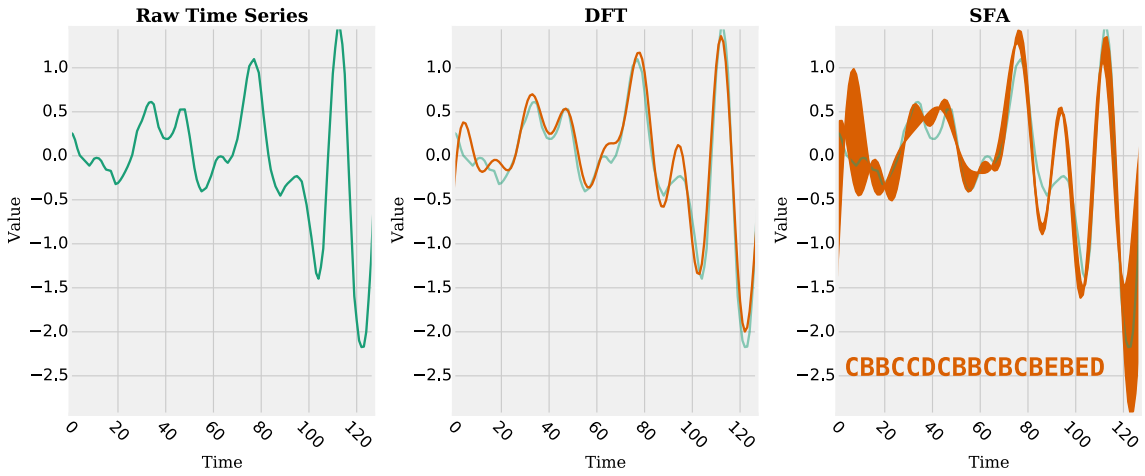
- A bag-of-patterns (histogram) of feature counts is used as input to classification
- This approach is fast (linear complexity), noise reducing, but order of substructures gets lost
- Representatives:
  - Univariate: WEASEL, Bag-of-SFA-Symbols (BOSS), Bag-of-Patterns (BoP), Time Series Bag of Features (TSBF)
  - Multivariate: SMTS, WEASEL+MUSE, LPS



1. Baydogan, M.G., Runger, G.: Learning a symbolic representation for multivariate time series classification. DMKD 29(2), 400–422 (2015)
2. Schäfer, P., Leser, U.: Fast and Accurate Time Series Classification with WEASEL. CIKM pp. 637–646 (2017)

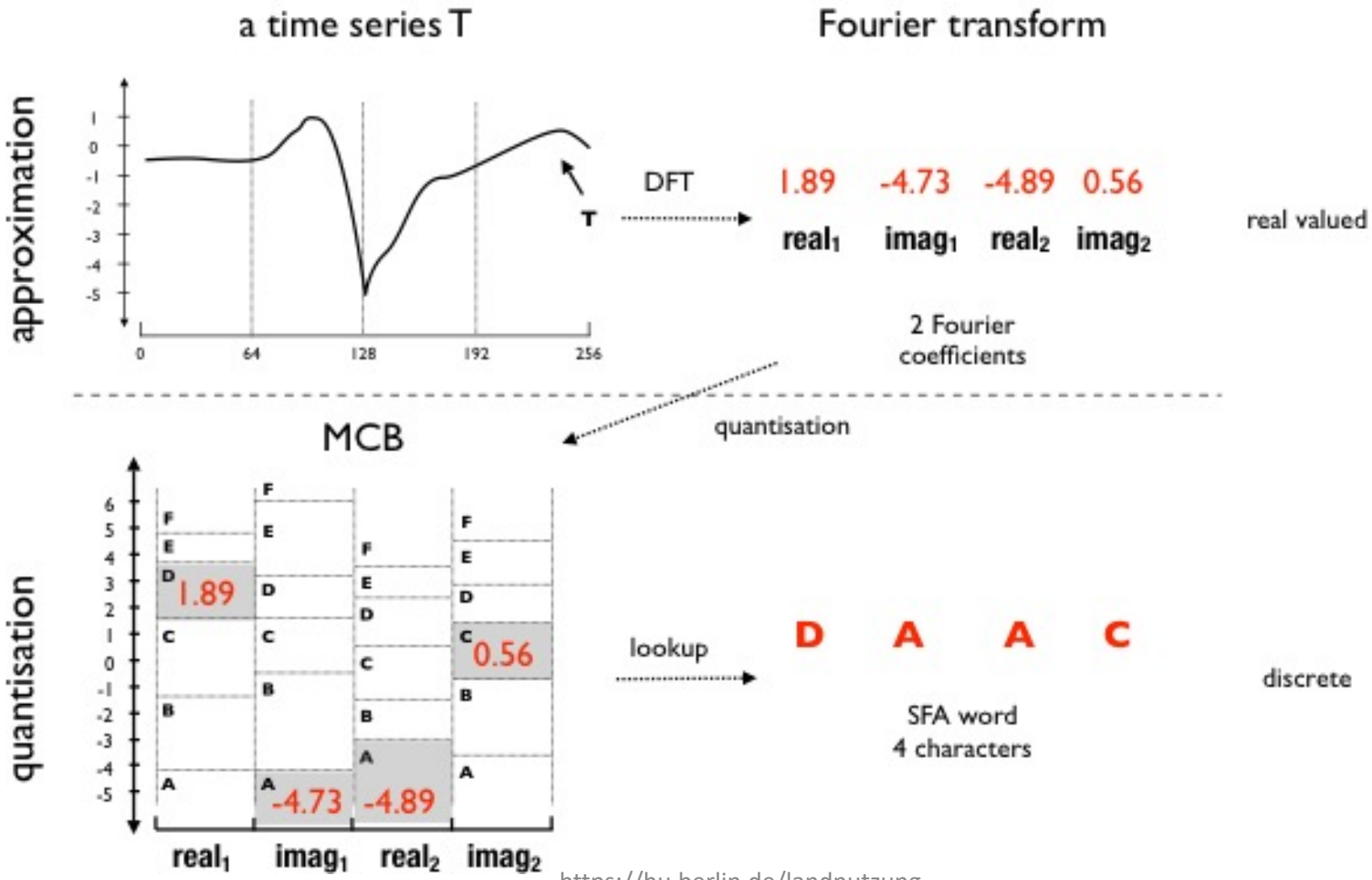
Symbolic Fourier Approximation (SFA)

- SFA represents each real valued subsequence by a word
- SFA is composed of
  - *a) approximation* using the Fourier transform and
  - b) a data adaptive *discretization*
- The discretization intervals are learned from the Fourier transformed data distribution



Raw:	DFT	Discretization
0.2679	0	C
0.2480	-8.81	B
0.1828	-20.7	B
0.0817	-11.9	C
0.0051	-6.28	C
-0.023	-8.02	D
-0.052	-0.67	C
-0.082	15.31	B
-0.111	-18.7	B
-0.075	-18.36	C
-0.032	-5.67	B
-0.022	-16.84	C
-0.029	-8.919	B
[...]	[...]	[...]

Symbolic Fourier Approximation (SFA)



Algorithm

```
Histogram BOSSTransform(
       TimeSeries sample ,
       WindowLength w,
       Wordlength l,
       Symbolc c)
    Histogram boss = {}
    for subsequence S in sliding_windows(sample, w)
        String word=SFA(S, l, c)
        if word != lastWord // numerosity reduction
             boss[word]++ // increase counts
        lastWord = word
    return boss
```

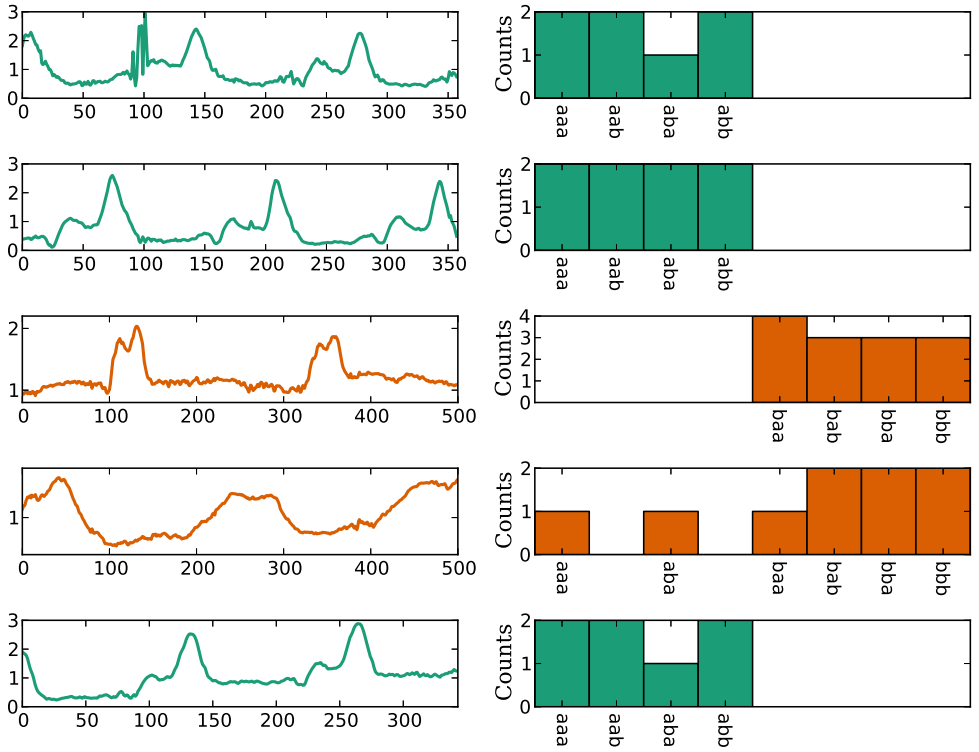
- The basis algorithm extracts sliding windows of length w
- Each sliding window is transformed to a word of length l and c symbols using SFA
- Numerosity reduction removes duplicates:
  bcc bcc bcc bcc bcc bcc bcc bcc **ccc** ccc **bcc bcb** bcb bcb bcb
  becomes:
  bcc ccc bcc bcb
- The words are added to a histogram

Runtime

- The runtime is dominated by the DFT of each window
  - There are n-w+1 sliding windows of length w
  - The Fourier transform has to be applied to each window, thus $O(n\ w\ \log w)$
- But sliding windows overlap!
- Using the *Momentary Fourier transform* computations of overlapping windows can be saved
- This results in a runtime of just $O(n\ +w\ \log w)$

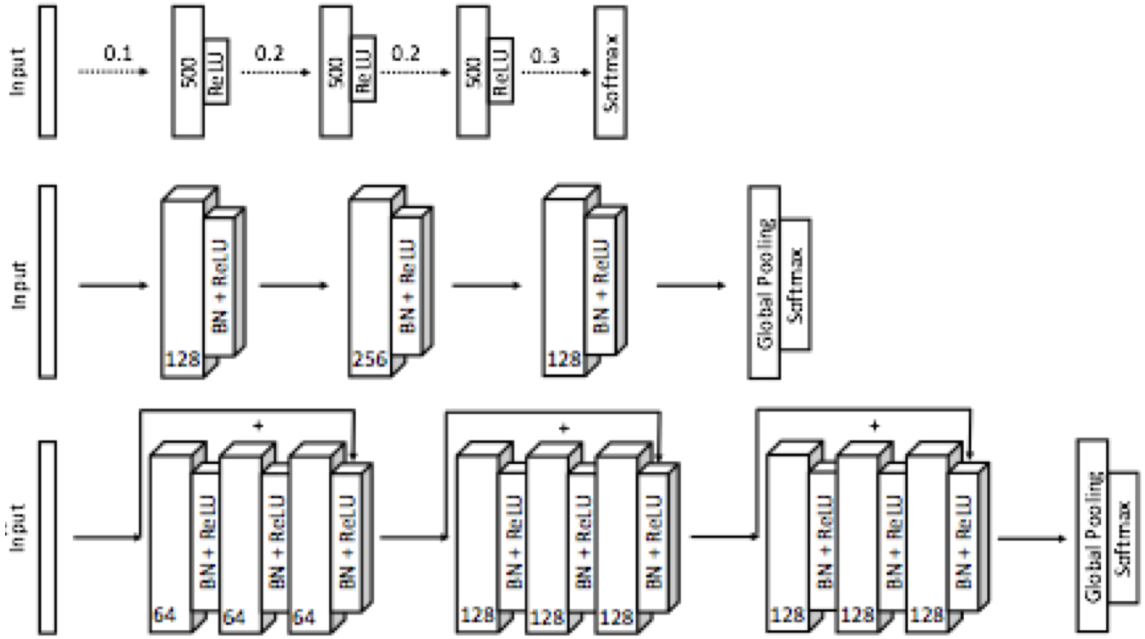
How to use Bag-of-Patterns for Classification

- 1-Nearest-Neighbour search over histograms
- TF-IDF model on histograms
- Or using a classifier:
  - Obtain histograms and train a classifier on these histograms
  - Predict a novel time series based on its histogram using the model



4. Deep Learning-based

- Deep Learning was successful in many domains reaching human performance level
- End-to-end NN approaches in TSC are based on Convolutional Neural Networks (CNN), Fully Connected Networks (FC), or Recurrent Neural Networks (RNN)
- Representatives: ResNet, FCN, Encoder, MLP, Time-CNN, TWIESN, MCDCNN, MCNN, t-LeNet.



Top to bottom: MLP, FCN, ResNet

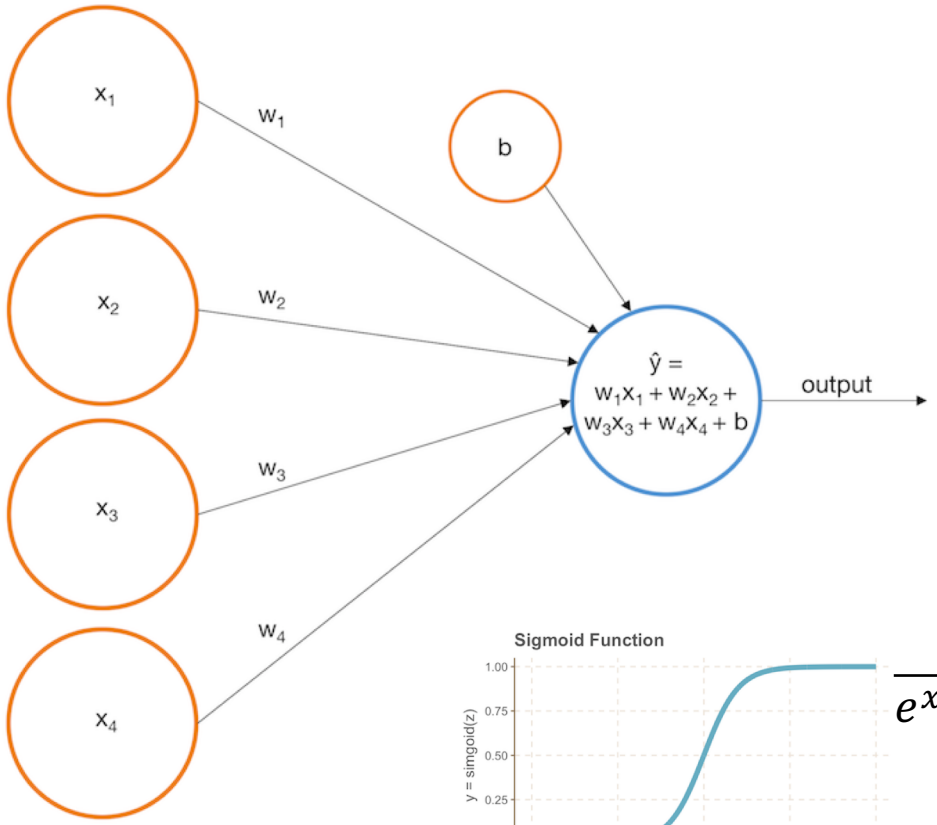
1. Fawaz, Hassan Ismail, et al. "Deep learning for time series classification: a review." arXiv preprint arXiv:1809.04356 (2018).

Perceptrons

- The most basic architecture of a neural network is called *perceptron*
- It consists of two layers of nodes: input nodes (data points) and a single output node
- The perceptron performs a mathematical computation on the inputs $T = (x_1, \dots x_n)$:

$$y = f(w \cdot T + b)$$

  with w being a set of weights, bias term b and activation function f
- An activation function (e.g. Sigmoid) is used to filter the output of the perceptron to [0,1]
- This gives a probability estimate
- Temporal values are independently treated from each other, thus the temporal information is lost



$$\frac{e^x}{e^x + 1}$$

http://blog.kaggle.com/2017/11/27/introduction-to-neural-networks/

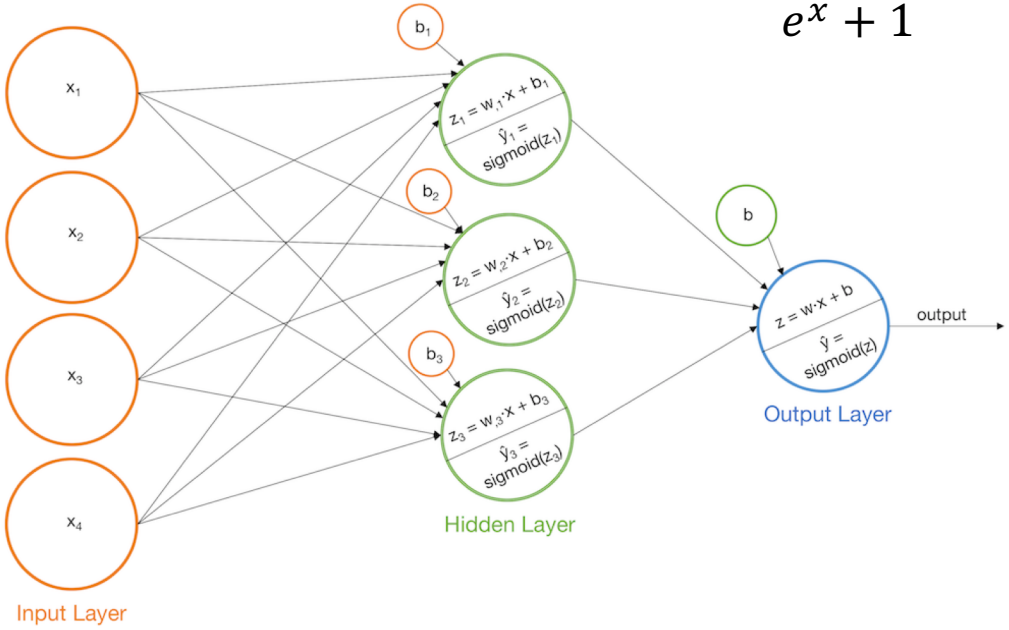
Multi Layer Perceptrons (MLP)

- Most simple and traditional architecture of deep NN, also known as Fully-Connected Network (FC)
- MLP have hidden layers in addition to inputs and output nodes
- Fully-connected: Every perceptron is connected to every perceptron in the previous layer
- Connections are still modelled by weights:

$$y = f(w \cdot T + b)$$

- The output gives a probability estimate
- Learning:
  - Forward pass: The inputs for the training instances are fed into the neural network. The error on the training data is estimated
  - back-propagation: We update the model‘s weights in a backward pass such that the train error is minimized
- Still temporal information is lost

$$\frac{e^x}{e^x + 1}$$

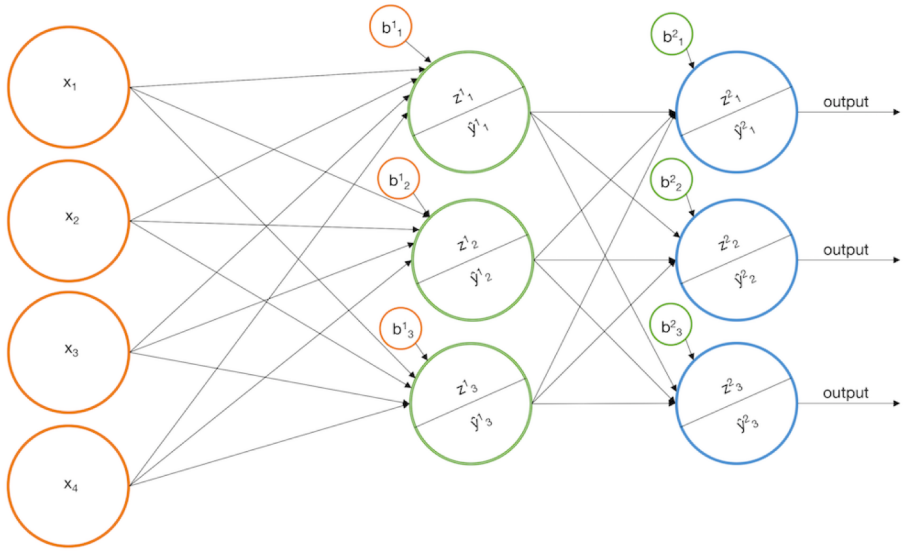


http://blog.kaggle.com/2017/11/27/introduction-to-neural-networks/

Multiclass Classification (“Softmax Layer”)

- Extends the network for multiclass classification using multiple output nodes
- Each output node corresponds to one class
- The Softmax function maps the outputs, such that these sum up to 1
- This gives probability estimates per class:

$$\{p(class_1), \ldots, p(class_k)\}$$



Fully Convolutional Neural Network

- A convolution can be seen as applying and sliding a filter over the time series
- Results in a filtered time series
- Several (128 to 256) filters are applied to learn multiple discriminative features
- These filters are trained automatically using a feed-forward pass followed by back-propagation
- The network is invariant to the length of the time series

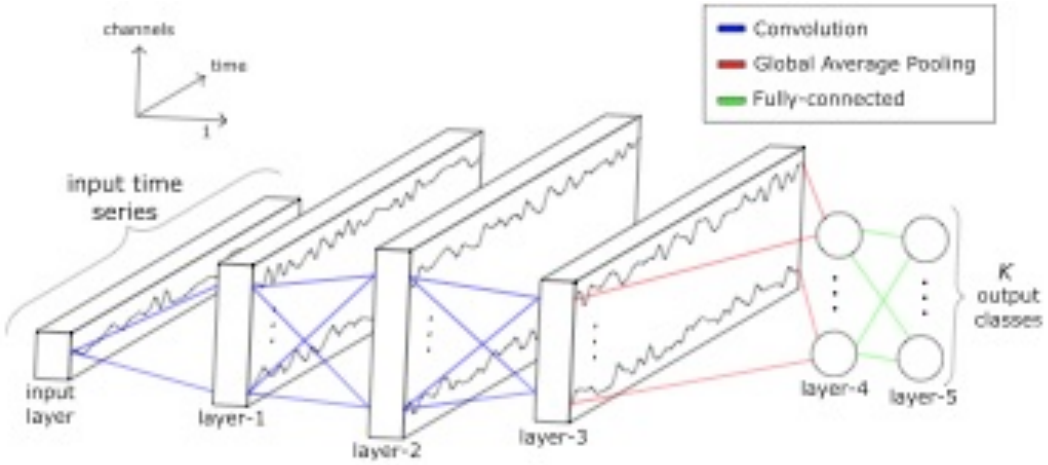


Fig. 3: Fully Convolutional Neural Network architecture

1. Fawaz, Hassan Ismail, et al. "Deep learning for time series classification: a review." arXiv preprint arXiv:1809.04356 (2018).

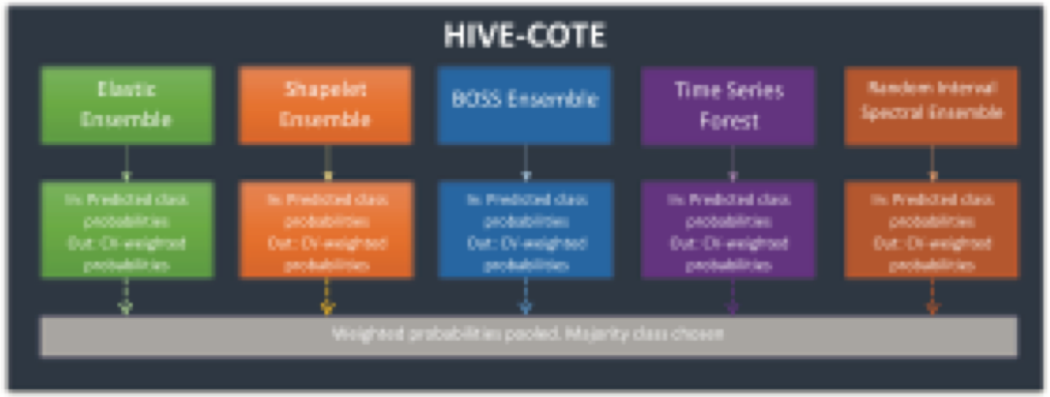
Recurrent Neural Network

- So far, rarely applied for time series classification:
  - Suffer from vanishing gradient problem on long time series
  - Computationally harder to train

- Thus, not mentioned here… but still interesting approach

5. Ensembles

- Ensembles combine different time series classifiers using bagging or majority voting
- Highest accuracy by combining different representations but high computational complexity
- Representatives:
  - Univariate: Elastic Ensemble (EE PROP), Collective of Transformation Ensembles (COTE)



Hive Cote: Ensemble over similarity-/shapelet-/dictionary-based classifiers

1. Bagnall, Anthony, et al. "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances." *Data Mining and Knowledge Discovery* 31.3 (2017): 606-660.

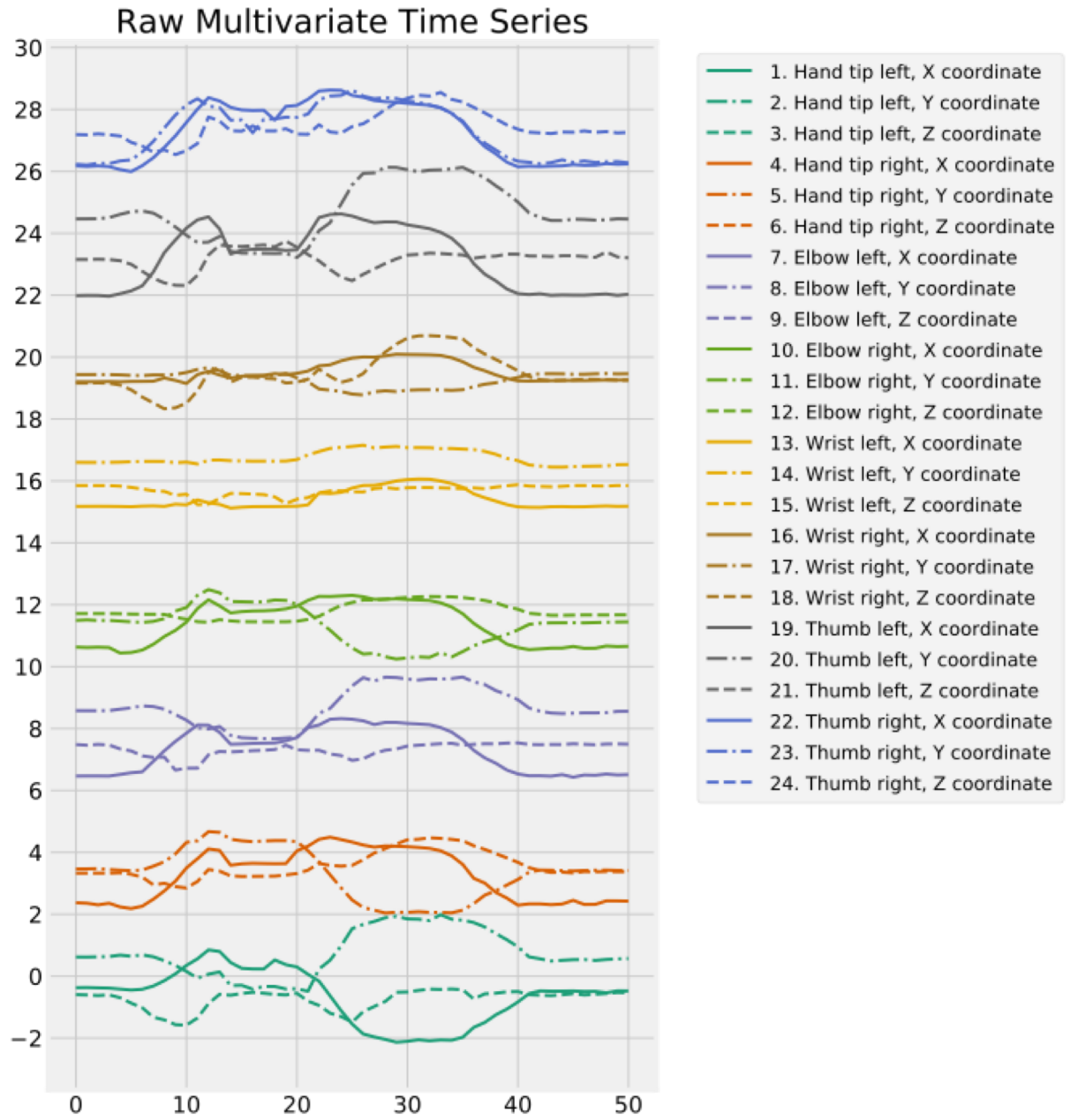
Which approach to use?

Related: A TS challenge on gesture data

- Participants are given a training set of labelled multivariate time series representing isolated gestures captured with a Kinect system by different users

- Datasets are 24 dimensional

- AALTD Challenge: https://aaltd16.irisa.fr/challenge/



Official leaderboard

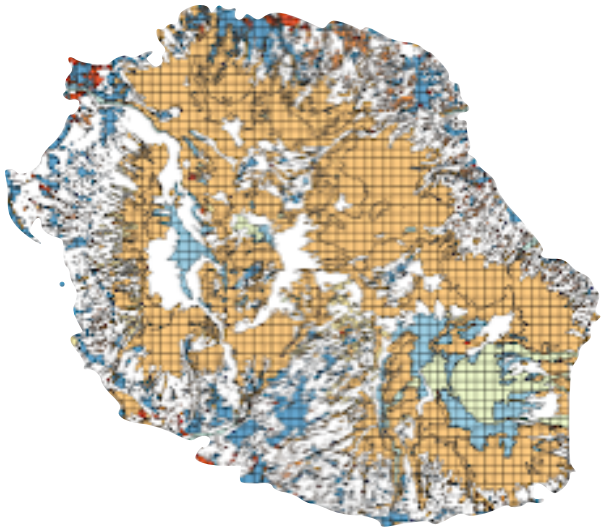
The following leaderboard has been computed on the whole test set.

Task 1

Rank	Team name	Method name	Accuracy	Number of submitted runs (max. 10)
1.	UCRDMYeh	bofSC + randShape	0.961	1
2.	Mustafa Baydogan	SMTS	0.956	3
–	Lemaire-Boullé, Orange Labs	Automatic Feature Construction + Selective Naive Bayes	0.956	2
4.	HU-WBI	MWSL	0.950	3
5.	CIML	RC	0.944	3
–	UCRDMYeh	convNet	0.944	1
–	UCRDMYeh	bofSC	0.944	1
8.	UEA	COTE	0.939	4
9.	UEA	HESCA	0.933	2
–	Josif Grabocka	LearningShapelets	0.933	1
11.	UCRDM	pDTWKerSVM + RandSub	0.928	2
–	UEA	Rotation Forest Benchmark	0.928	3
13.	HU-WBI	BOSS	0.911	4
14.	DDIG	Softmax+RandShapes	0.906	1

Related: Challenge on Sattelite Data (Reunion Island)

- They used satellite time series of Landsat 8 images collected over Reunion Island in 2014 [1]
  - 81714 pixels
  - 10 spectral features: seven reflectance bands and three vegetation indices (NDVI, NDWI, BI)
  - 23 time stamps: 16 days revisit time:
  - 2 spatial-coordinates: longitude and latitude
  - 9 land cover classes (manually classified)
  - preprocessed: atmospherically corrected, geometrically corrected, and cloud-masked



ID	Land cover class	Samples
1	Urban Areas	16000
2	Other built-up surfaces	3236
3	Forests	16000
4	Sparse Vegetation	16000
5	Rocks and bare soil	12942
6	Grassland	5681
7	Sugarcane crops	7656
8	Other crops	1600
9	Water	2599

(top) The Reunion Island site and (bottom) the corresponding Land Cover Classes [1]

- [1] TiSeLaC Challenge: https://sites.google.com/site/dinoienco/tiselc

Questions?

Next Steps

- **Today: choose a topic**
- Before **30.11.18**: meet me to discuss topic
- **07.12.18, 15-16 Uhr:** Flash presentation, **RUD 25 4.410**
  - Present ideas and your topic in 5min