



Data Intelligence

Time Series Analysis

U Kang
Seoul National University



In This Lecture

- Motivation of time series analysis
- Similarity search
- Linear forecasting
- Deep learning methods



Outline

- ➡ ☐ **Motivation**
- ☐ Similarity Search
- ☐ Linear Forecasting
- ☐ Deep Learning Methods



Problem definition

- Given: one or more sequences

$x_1, x_2, \dots, x_t, \dots$

$(y_1, y_2, \dots, y_t, \dots$
 $\dots)$

- Task

- Find similar sequences
- Forecast future values
- Classify sequences



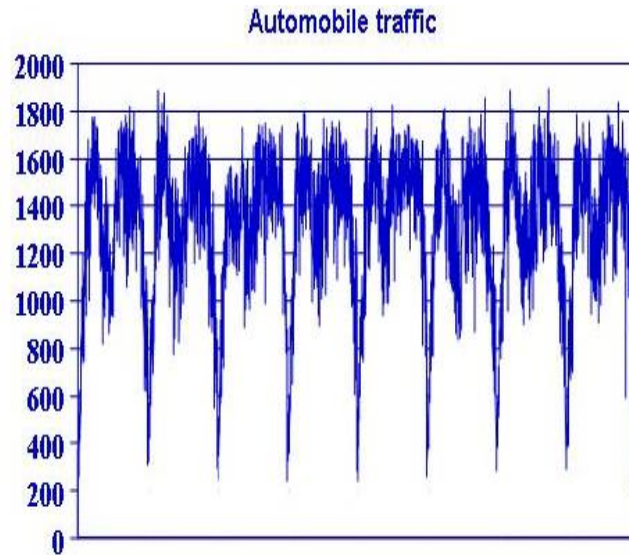
Motivation - Applications

- Financial, sales, economic series
- Medical
 - ECGs +; blood pressure etc monitoring
 - Reactions to new drugs
 - Elderly care
- 'Smart house'
 - Sensors monitor temperature, humidity, air quality
- Video surveillance



Motivation - Applications (cont'd)

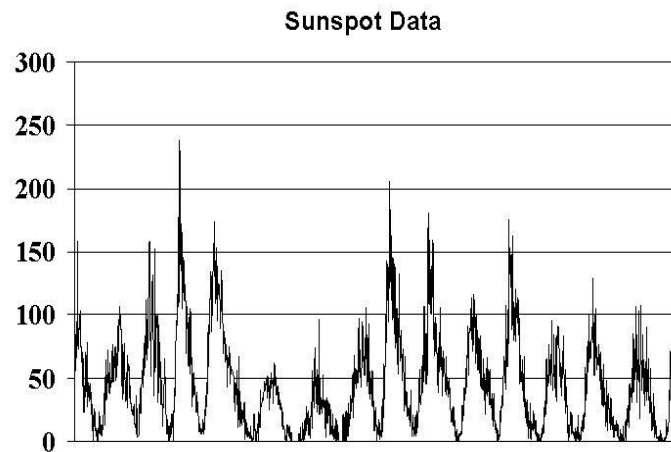
- Civil/automobile infrastructure
 - Bridge vibrations [Oppenheim+02]
 - Road conditions / traffic monitoring





Motivation - Applications (cont'd)

- Weather, environment/anti-pollution
 - Volcano monitoring
 - Air/water pollutant monitoring





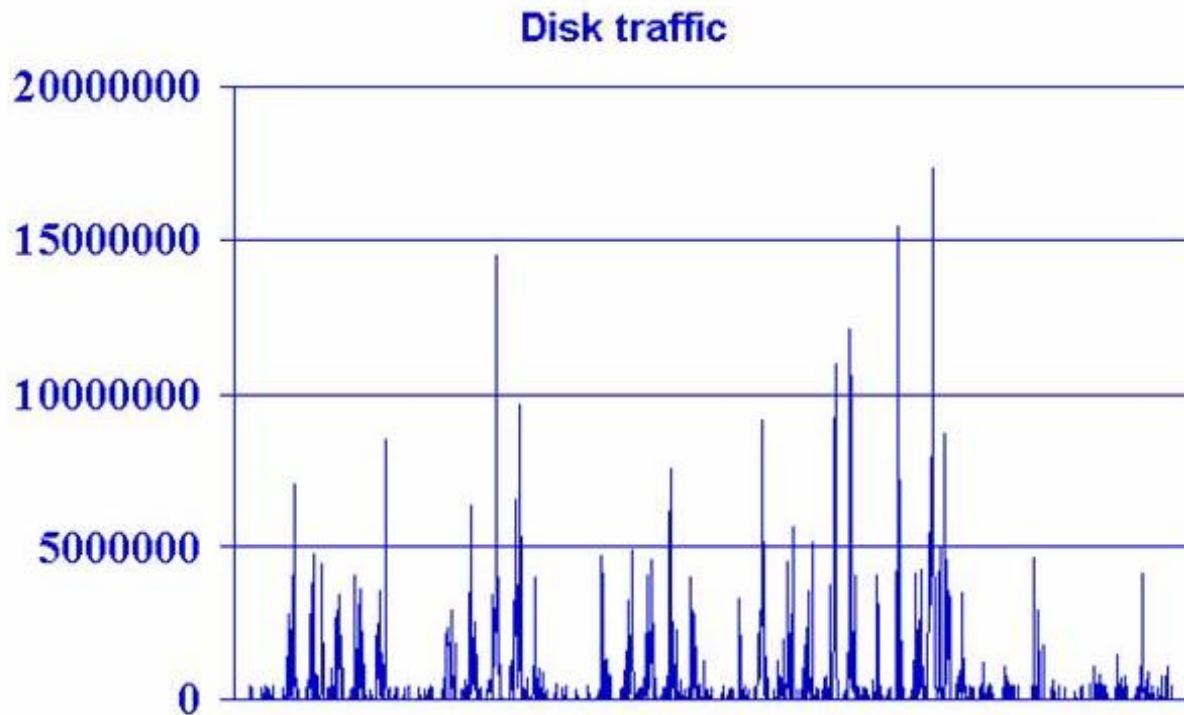
Motivation - Applications (cont'd)

- Computer systems
 - 'Active Disks' (buffering, prefetching)
 - Web servers
 - Network traffic monitoring
 - ...



Stream Data: Disk accesses

#bytes





Outline

☒ Motivation

➔ ☐ **Similarity Search**

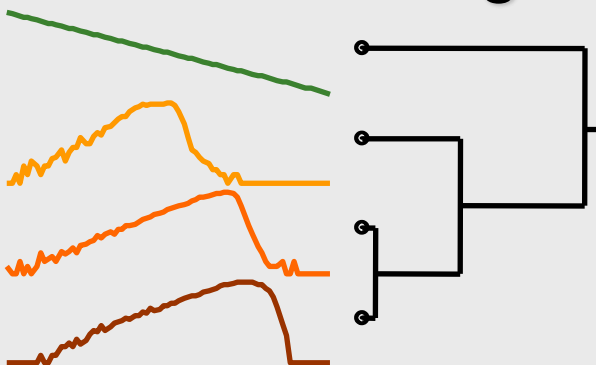
☐ Linear Forecasting

☐ Deep Learning Methods

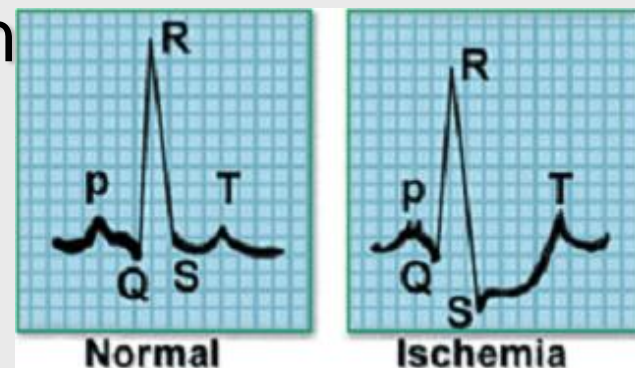


Importance of Similarity Search

Clustering



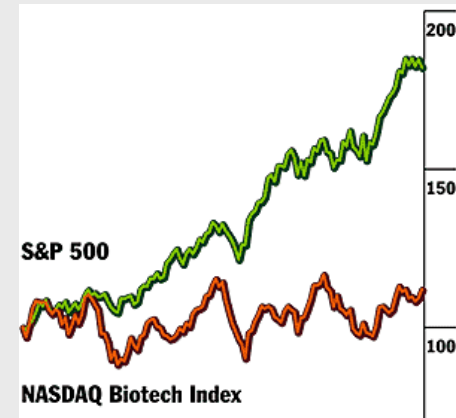
Classification



Rule Discovery



Query by Content

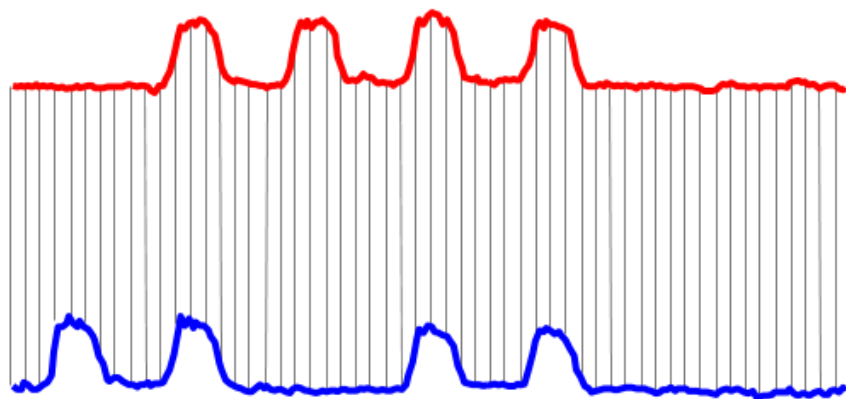




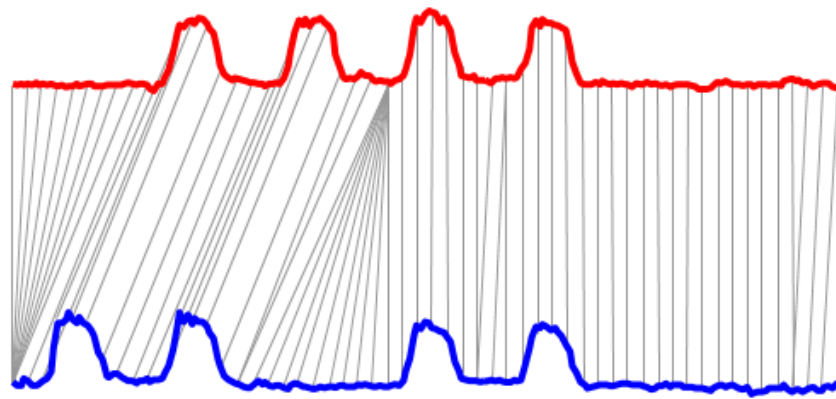
Importance of Distance Functions

Two major families

- Euclidean and Lp norms
- Time warping and variations (DTW etc.)



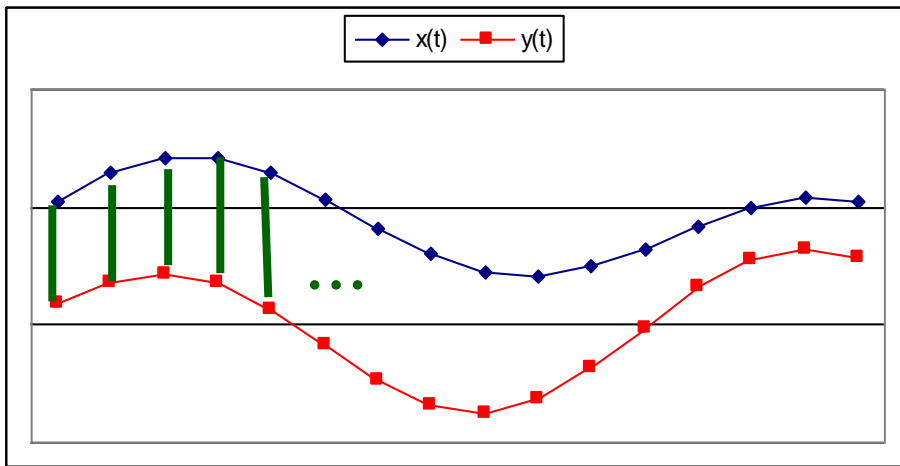
Euclidean: one to one



Time warping: nonlinear alignments



Euclidean and Lp



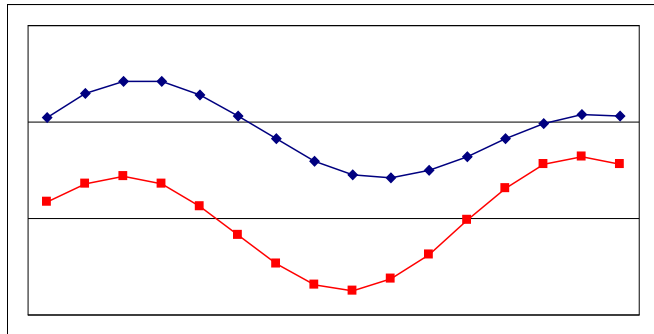
$$D(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

$$\mathcal{L}_p(\vec{x}, \vec{y}) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

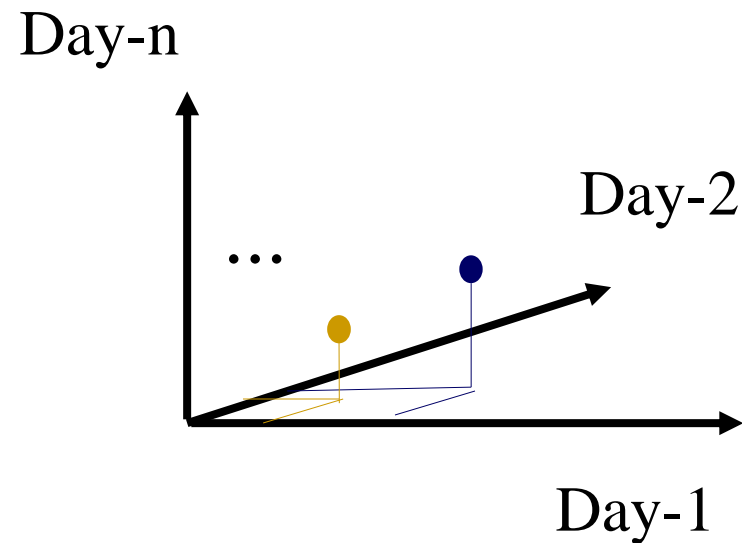
- \mathcal{L}_1 : city-block = Manhattan
- \mathcal{L}_2 = Euclidean
- \mathcal{L}_∞



Observation #1



- Time sequence \rightarrow n-d vector

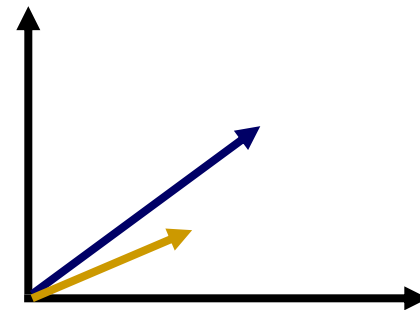
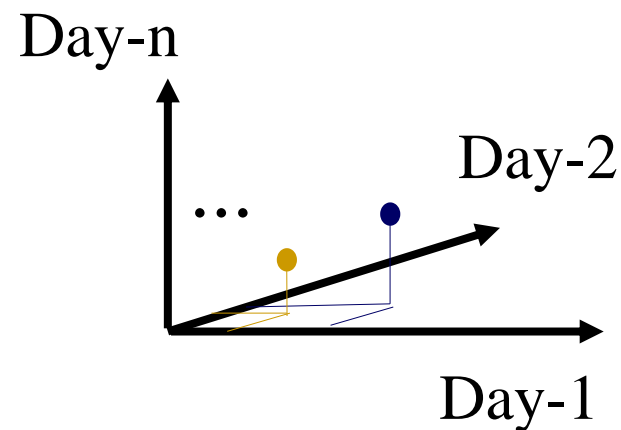




Observation #2

Euclidean distance is closely related to

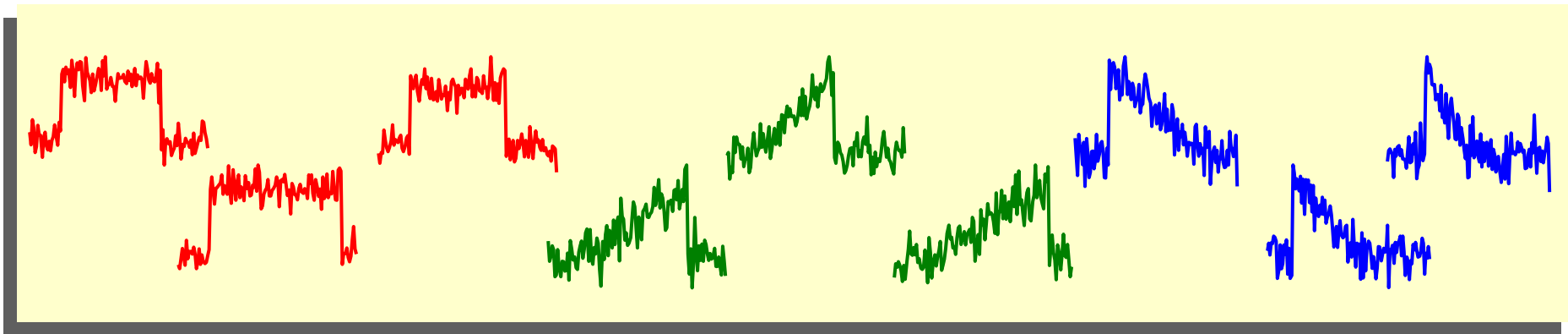
- ❑ Cosine similarity
- ❑ Dot product





Limitation of Euclidean Distance

Classification Experiment on **Cylinder-Bell-Funnel** Dataset



Training data consists of 10 exemplars from each class.

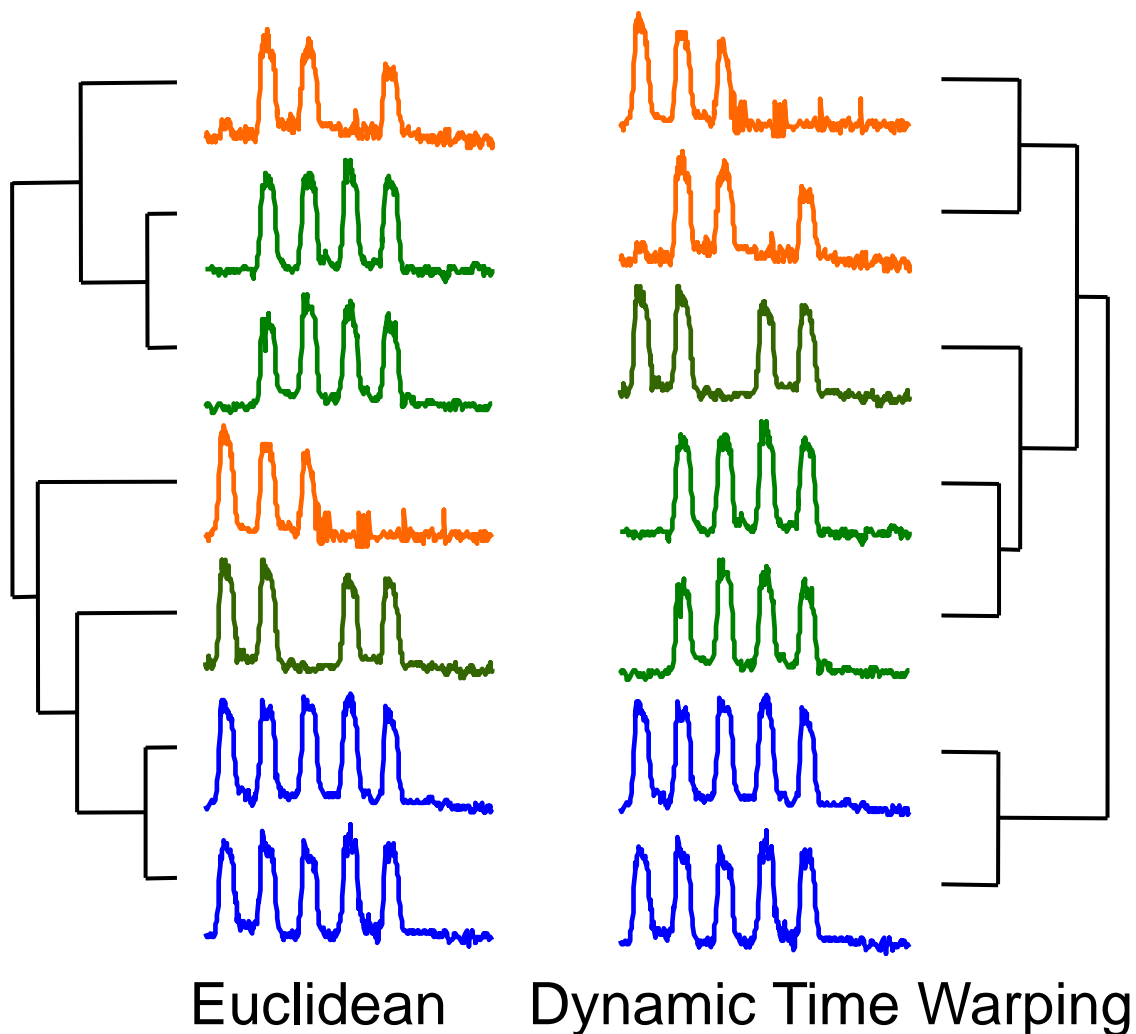
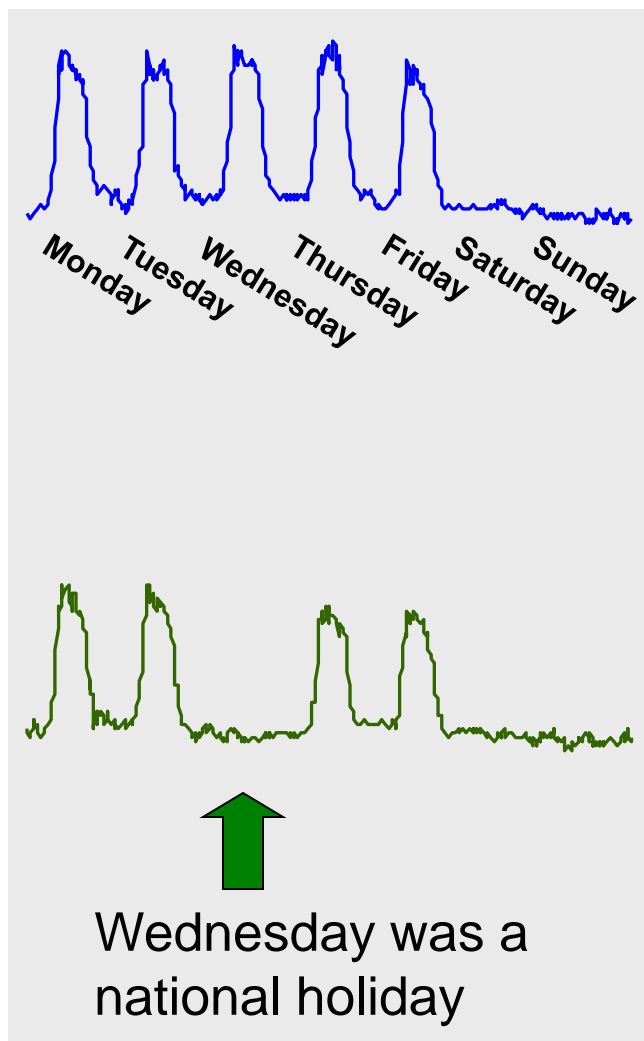
- (One) Nearest Neighbor Algorithm
- “Leaving-one-out” evaluation, averaged over 100 runs

- Euclidean Distance Error rate
- Dynamic Time Warping Error rate

26.10%
2.87%



Limitation of Euclidean Distance





Applications of DTW

Bioinformatics: Aach, J. and Church, G. (2001). Aligning gene expression time series with time warping algorithms. *Bioinformatics*. Volume 17, pp 495-508.

Robotics: Schmill, M., Oates, T. & Cohen, P. (1999). Learned models for continuous planning. In *7th International Workshop on Artificial Intelligence and Statistics*.

Medicine: Caiani, E.G., et. al. (1998) Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. *IEEE Computers in Cardiology*.

Chemistry: Gollmer, K., & Posten, C. (1995) Detection of distorted pattern using dynamic time warping algorithm and application for supervision of bioprocesses. *IFAC CHEMFAS-4*

Gesture Recognition: Gavrilu, D. M. & Davis, L. S. (1995). Towards 3-d model-based tracking and recognition of human movement: a multi-view approach. In *IEEE IWAFAFR*

Meteorology/ Tracking/ Biometrics / Astronomy / Finance / Manufacturing ...

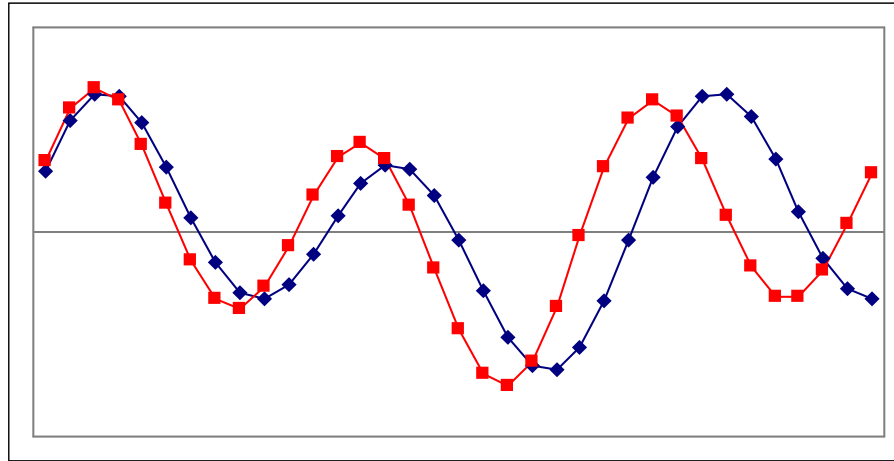


Time Warping

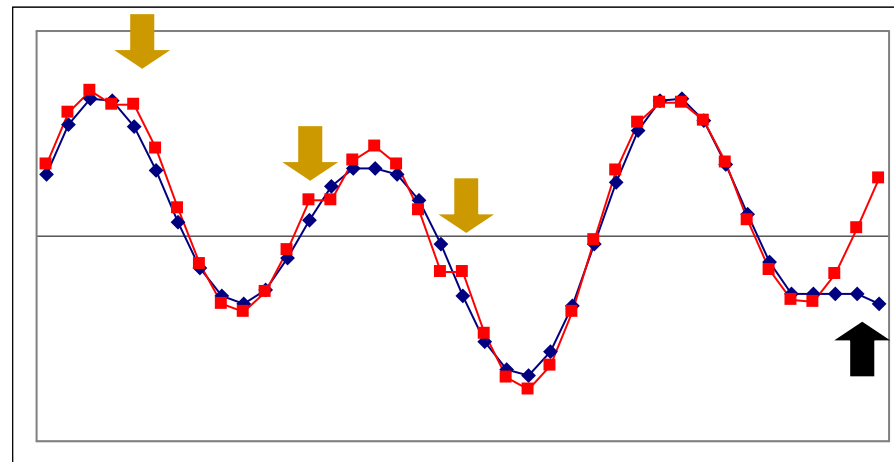
- Allow accelerations - decelerations
 - (with or w/o penalty)
- Then compute the (Euclidean) distance (+ penalty)
- Related to the string-editing distance



Time Warping



‘stutters’:





Time warping

Q: how to compute it?

A: dynamic programming

$D(i, j)$ = cost to match

prefix of length i of first sequence x with prefix of
length j of second sequence y



Time warping

Thus, with no penalty for stutter, for sequences

$$x_1, x_2, \dots, x_i \quad y_1, y_2, \dots, y_j$$

$$D(i, j) = \|x[i] - y[j]\| + \min \begin{cases} D(i-1, j-1) & \text{no stutter} \\ D(i, j-1) & \text{x-stutter} \\ D(i-1, j) & \text{y-stutter} \end{cases}$$

Similar to string-edit distance



Dynamic Time Warping

■ Example: DTW(x, y)

□ $x_1 = 3, x_2 = 5$

□ $y_1 = 3, y_2 = 4, y_3 = 5$

x
3
5

	j = 1	j = 2	j = 3
i = 1	$D(i,j) = 0$	$D(i,j) = 1$	$D(i,j) = 3$
i = 2	$D(i,j) = 2$	$D(i,j) = 1$	$D(i,j) = 1$

y
3
4
5

$$D(i, j) = \|x[i] - y[j]\| + \min \begin{cases} D(i-1, j-1) & \text{no stutter} \\ D(i, j-1) & \text{x-stutter} \\ D(i-1, j) & \text{y-stutter} \end{cases}$$



Time warping

- Complexity: $O(M*N)$ - quadratic on the length of the strings
- Many variations (penalty for stutters; limit on the number/percentage of stutters; ...)
- Popular in voice processing [Rabiner + Juang]



Lower Bounding

We can speed up similarity search under DTW by using a lower bounding function.

Intuition

Try to use a cheap lower bounding calculation as often as possible.

Only do the expensive, full calculations when it is absolutely necessary.

Algorithm Lower_Bounding_Sequential_Scan(Q)

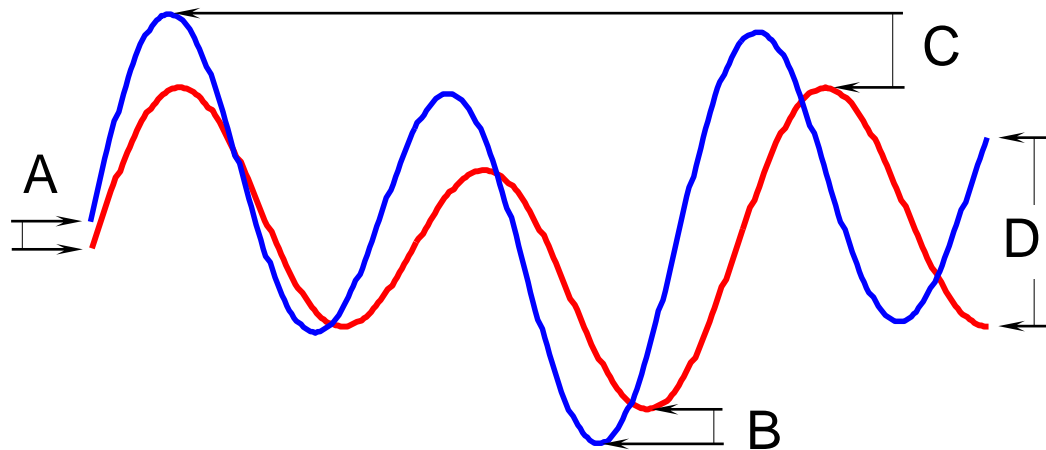
```
1.  best_so_far = infinity;
2.  for all sequences in database
3.    LB_dist = lower_bound_distance( $C_i$ , Q);
4.    if LB_dist < best_so_far
5.      true_dist = DTW( $C_i$ , Q);
6.      if true_dist < best_so_far
7.        best_so_far = true_dist;
8.        index_of_best_match =  $i$ ;
9.      endif
10.   endif
11. endfor
```



Lower Bound of Kim et. al.



LB_Kim



Kim, S, Park, S, & Chu, W. *An index-based approach for similarity search supporting time warping in large sequence databases*. ICDE 01, pp 607-614

The squared difference between the two sequence's first (A), last (D), minimum (B) and maximum points (C) is returned as the lower bound

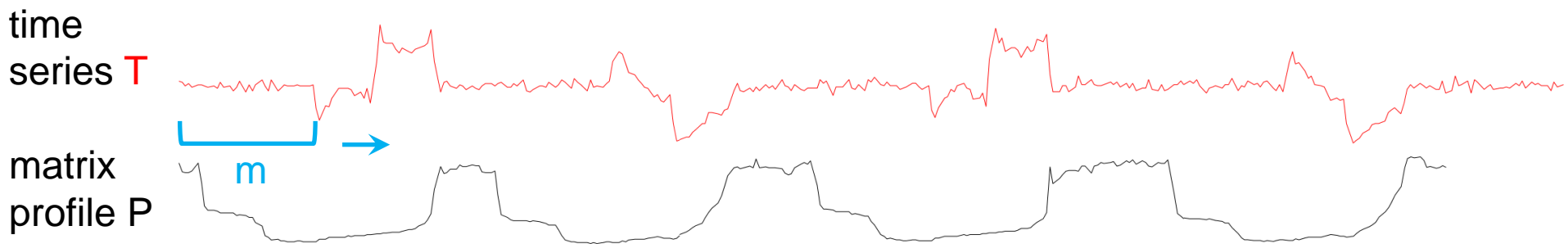
Many other lower bounding techniques are possible: LB_Keogh, etc.



Matrix Profile

■ Matrix Profile

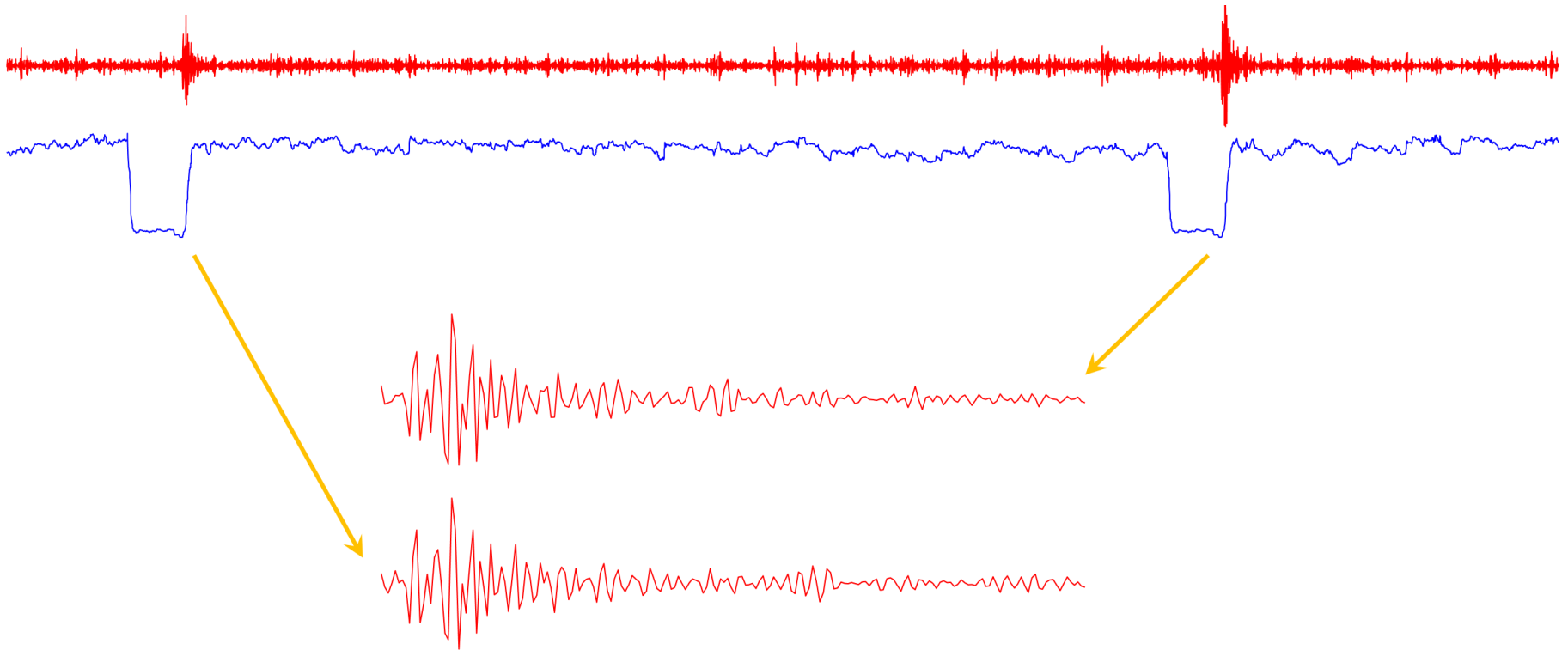
- The distance to the corresponding nearest neighbor of each subsequence can be stored in a vector called matrix profile





Matrix Profile

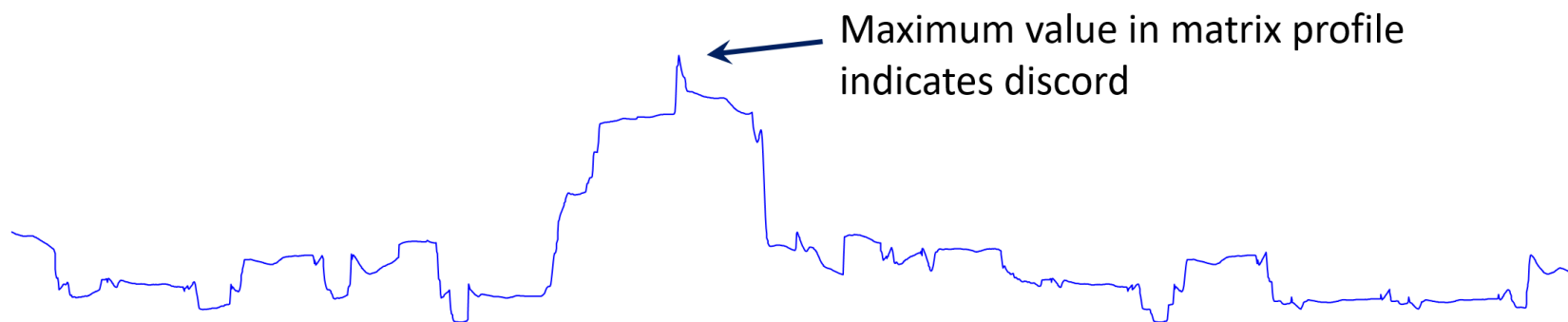
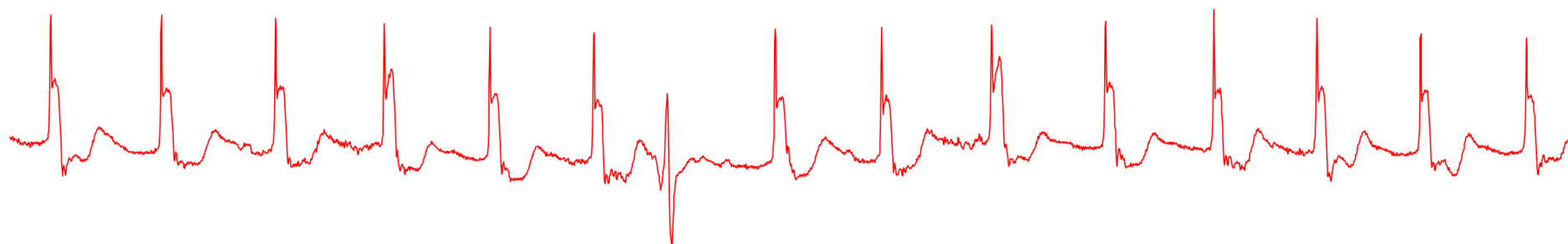
- Repeated earthquakes






Matrix Profile

■ Abnormal heartbeat detection from ECG





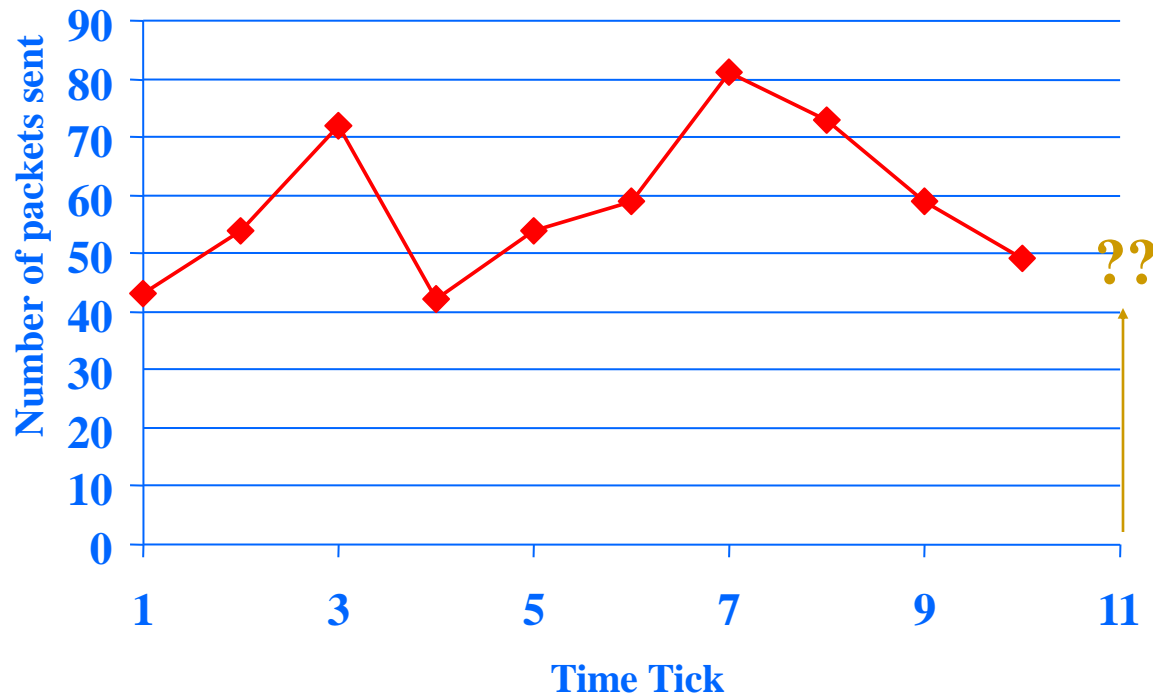
Outline

- ☒ Motivation
- ☒ Similarity Search
-  ☐ **Linear Forecasting**
- ☐ Deep Learning Methods



Forecasting

- Example: given x_{t-1}, x_{t-2}, \dots , forecast x_t



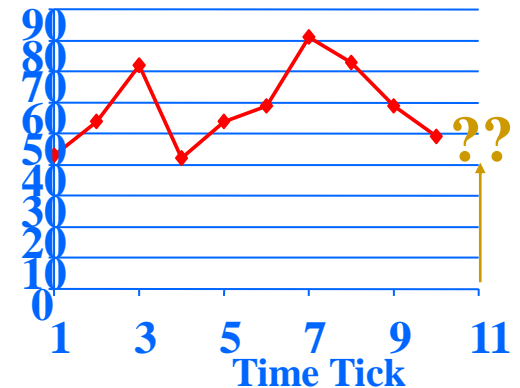


Forecasting

- Solution: try to express x_t as a linear function of the past: x_{t-1}, x_{t-2}, \dots , (up to a window of w)

Formally:

$$x_t \approx a_1 x_{t-1} + \dots + a_w x_{t-w} + noise$$



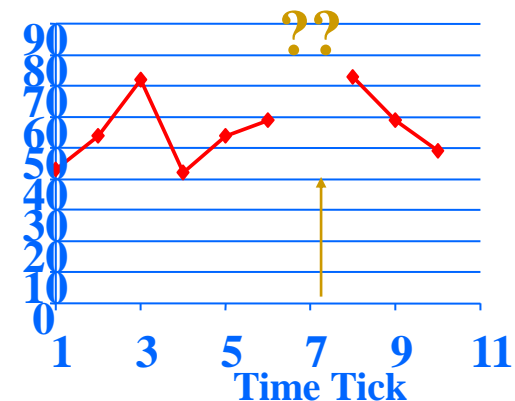


(Problem: Back-cast; interpolate)

- Solution - interpolate: try to express x_t as a linear function of the past AND the future:

$x_{t+1}, x_{t+2}, \dots x_{t+w_{future}}; x_{t-1}, \dots x_{t-w_{past}}$
(up to windows of w_{past}, w_{future})

- Exactly the same algo's

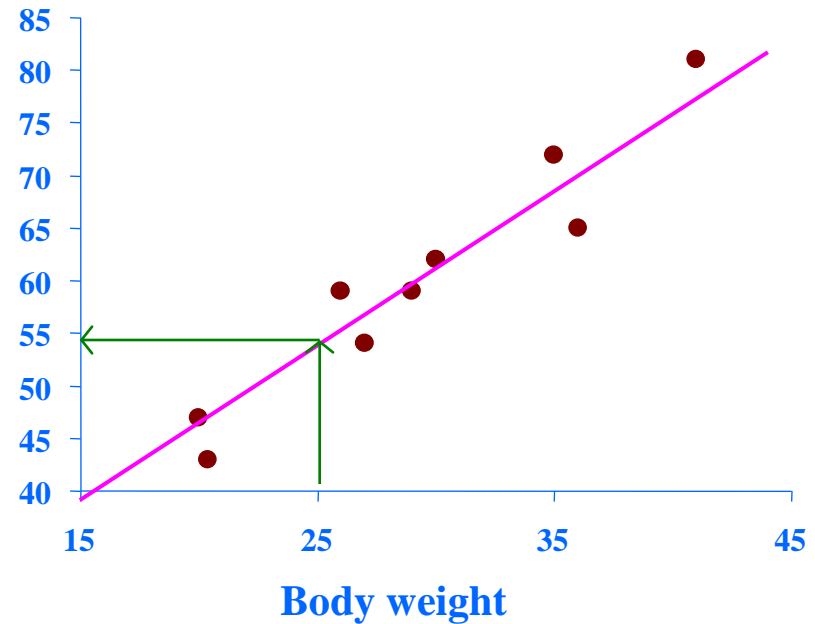




Linear Regression: idea

<i>patient</i>	<i>weight</i>	<i>height</i>
1	27	43
2	43	54
3	54	72
...
N	25	??

Body height



- Express what we don't know (= 'dependent variable')
- as a linear function of what we know (= 'indep. variable(s)')



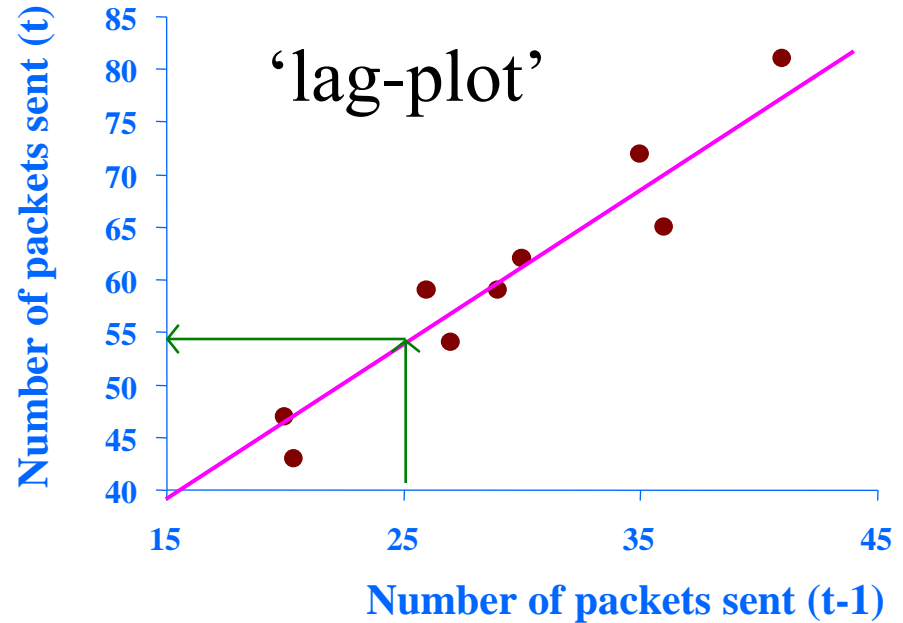
Linear Auto Regression:

<i>Time</i>	<i>Packets Sent(t)</i>
1	43
2	54
3	72
...	...
N	??



Linear Auto Regression:

Time	Packets Sent ($t-1$)	Packets Sent(t)
1	-	43
2	43	54
3	54	72
...
N	25	??

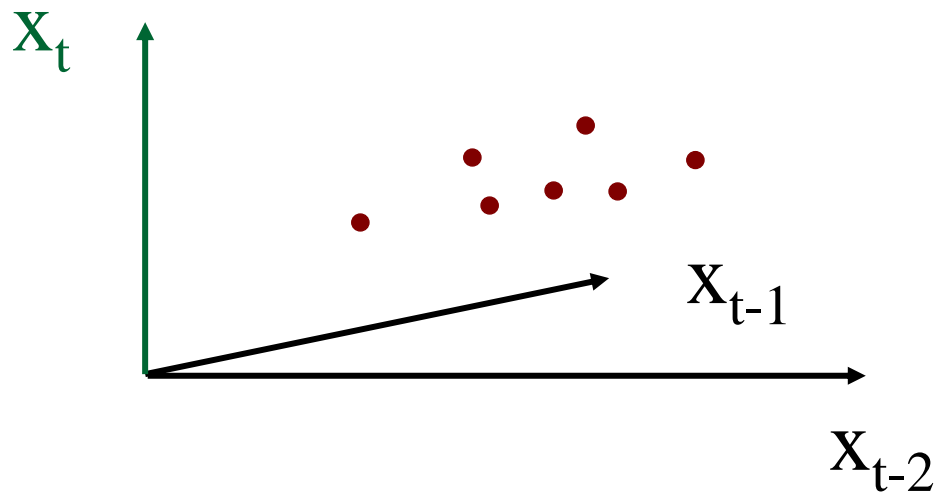


- lag $w=1$
- Dependent variable = # of packets sent ($S[t]$)
- Independent variable = # of packets sent ($S[t-1]$)



More details:

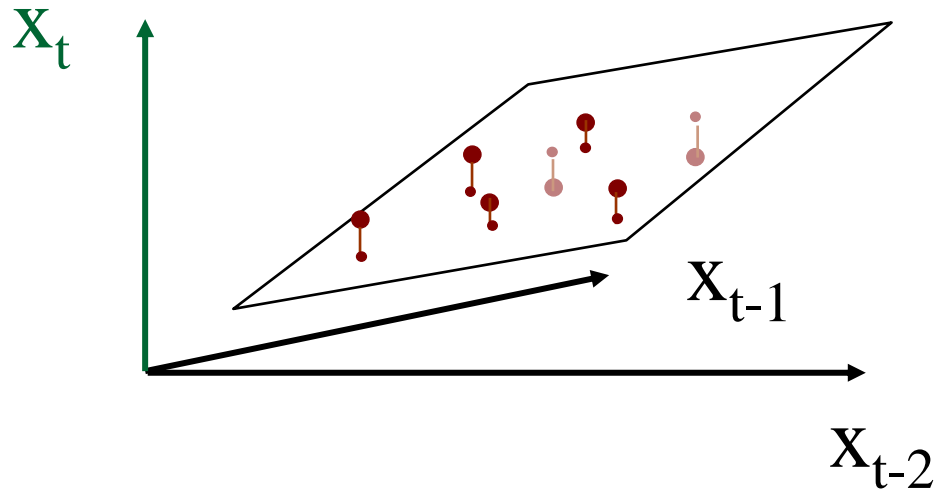
- Q1: Can it work with window $w > 1$?
- A1: YES!





More details:

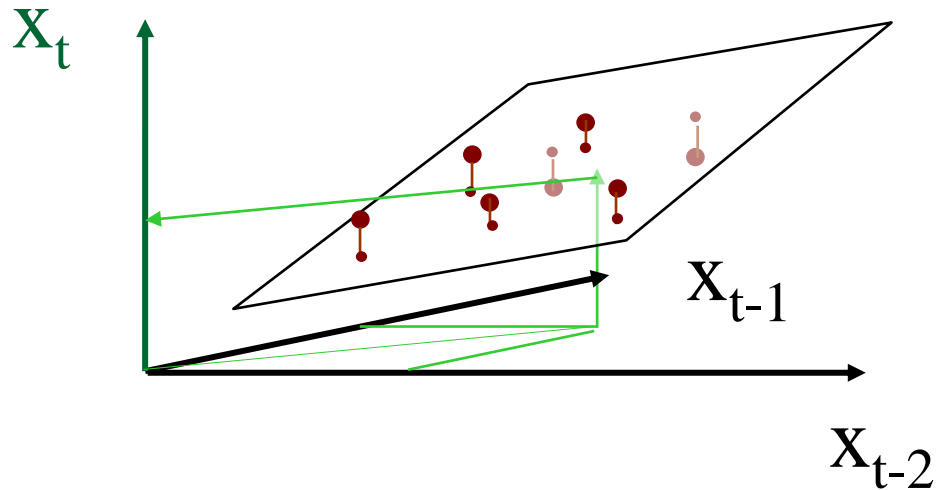
- Q1: Can it work with window $w > 1$?
- A1: YES! (we'll fit a hyper-plane, then!)





More details:

- Q1: Can it work with window $w > 1$?
- A1: YES! (we'll fit a hyper-plane, then!)





More details:

- Q1: Can it work with window $w > 1$?
- A1: YES! The problem becomes:

$$\mathbf{X}_{[N \times w]} \times \mathbf{a}_{[w \times 1]} = \mathbf{Y}_{[N \times 1]}$$

- OVER-CONSTRAINED
 - \mathbf{a} is the vector of the regression coefficients
 - \mathbf{X} has the N values of the w indep. variables
 - \mathbf{y} has the N values of the dependent variable



More details:

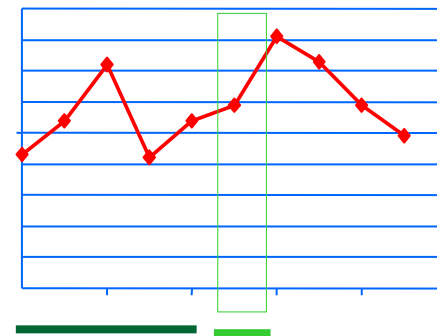
$$\blacksquare \mathbf{X}_{[N \times w]} \times \mathbf{a}_{[w \times 1]} = \mathbf{y}_{[N \times 1]}$$

Ind-var1

Ind-var-w

time

$$\begin{bmatrix} \underbrace{X_{11}, X_{12}, \dots, X_{1w}}_{\text{Ind-var1}} \\ X_{21}, X_{22}, \dots, X_{2w} \\ \vdots \\ \vdots \\ \vdots \\ X_{N1}, X_{N2}, \dots, X_{Nw} \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_w \end{bmatrix} = \begin{bmatrix} \underline{y_1} \\ y_2 \\ \vdots \\ \vdots \\ \vdots \\ y_N \end{bmatrix}$$





More details:

$$\blacksquare \mathbf{X}_{[N \times w]} \times \mathbf{a}_{[w \times 1]} = \mathbf{y}_{[N \times 1]}$$

Ind-var1

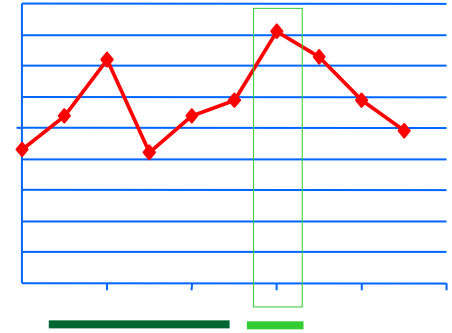
Ind-var-w

time

$$\begin{bmatrix} X_{11}, X_{12}, \dots, X_{1w} \\ X_{21}, X_{22}, \dots, X_{2w} \\ \vdots \\ \vdots \\ \vdots \\ X_{N1}, X_{N2}, \dots, X_{Nw} \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_w \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ \vdots \\ y_N \end{bmatrix}$$

time

time





More details

- Q2: How to estimate $a_1, a_2, \dots, a_w = \mathbf{a}$?
- A2: with Least Squares fit

$$\mathbf{a} = (\mathbf{X}^T \times \mathbf{X})^{-1} \times (\mathbf{X}^T \times \mathbf{y})$$

- If \mathbf{X} has linearly independent columns
- \mathbf{a} is the vector that minimizes the RMSE from \mathbf{y}
- If \mathbf{X} has linearly dependent columns, then use Moore-Penrose pseudo-inverse



Aside: Pseudoinverse



Moore-Penrose Pseudoinverse

- Assume we want to solve $Ax = y$ for x
- What if A is not invertible?
- What if A is not square?
- Still, we can find the 'best' x by using pseudoinverse
- For an $(n \times m)$ matrix A , the pseudoinverse A^+ is an $(m \times n)$ matrix



Moore-Penrose Pseudoinverse

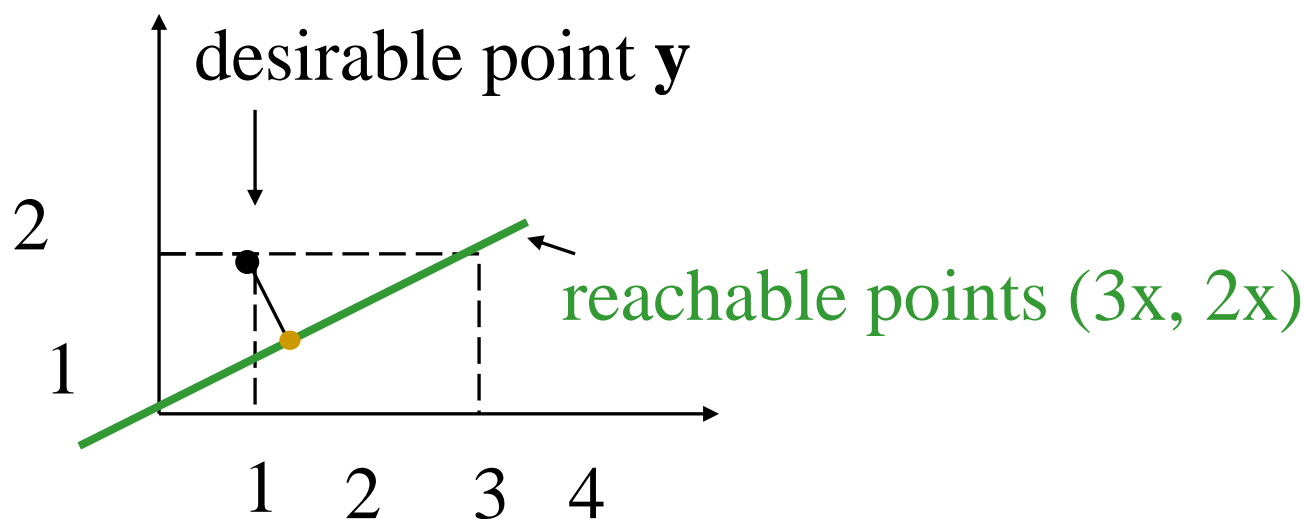
$$x = A^+ y$$

- If the equation has:
 - Exactly one solution: this is the same as the inverse
 - No solution: this gives us the solution with the smallest error $\|Ax - y\|_2$
 - Over-specified case
 - Many solutions: this gives us the solution with the smallest norm of x
 - Under-specified case



Pseudoinverse: Over-specified case

- No solution: this gives us the solution with the smallest error $\|Ax - y\|_2$
- $[3 \ 2]^T [x] = [1 \ 2]^T$ (i.e., $3x = 1$, $2x = 2$)

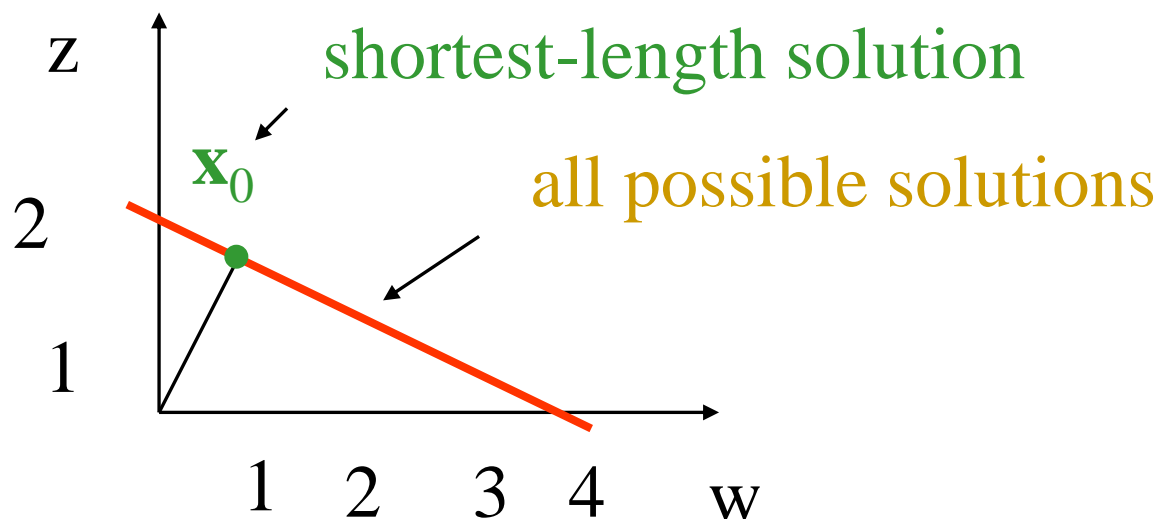


- $([3 \ 2]^T)^+ [1 \ 2] = \dots = 7/13$
- This method is called the 'least square'



Pseudoinverse: Under-specified case

- Many solutions: this gives us the solution with the smallest norm of x
- $[1 \ 2] [w \ z]^T = 4$ (i.e., $1w + 2z = 4$)



- $[1 \ 2]^+ 4 = [1/5 \ 2/5]^T 4 = [4/5 \ 8/5]$



Computing the Pseudoinverse

- The SVD allows the computation of the pseudoinverse:

$$A = UDV^{\top}$$

$$A^{+} = VD^{+}U^{\top},$$

Take reciprocal of non-zero entries



End of Aside



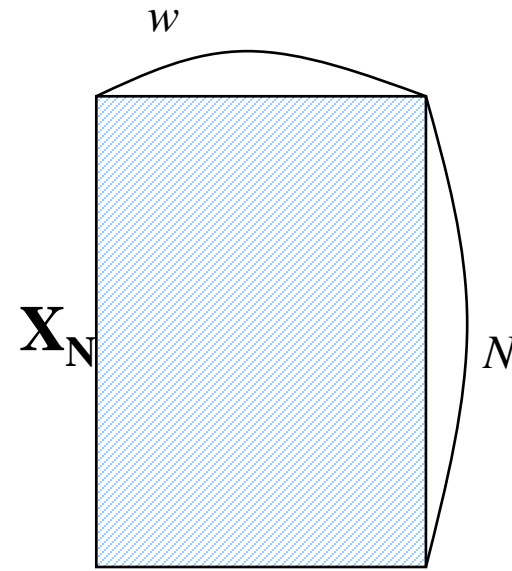
More details

■ Straightforward solution:

$$\mathbf{a} = (\mathbf{X}^T \times \mathbf{X})^{-1} \times (\mathbf{X}^T \times \mathbf{y})$$

\mathbf{a} : Regression Coeff. Vector

\mathbf{X} : Sample Matrix



• Observations:

- Sample matrix \mathbf{X} grows over time
- needs matrix inversion
- $\mathbf{O}(N \times w^2)$ computation
- $\mathbf{O}(N \times w)$ storage



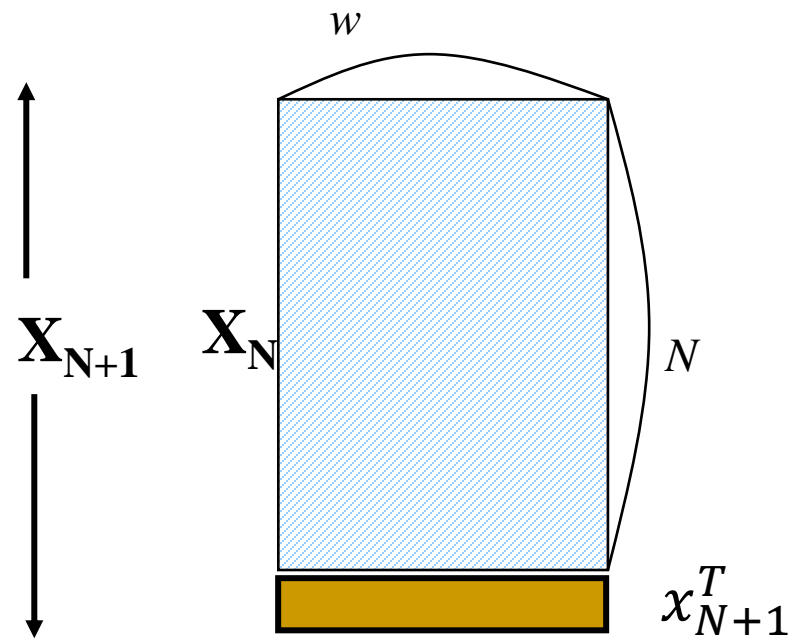
Even more details

- Q3: Can we estimate \mathbf{a} incrementally?
- A3: Yes, with the brilliant, classic method of 'Recursive Least Squares' (RLS) (see, e.g., [Yi+00], for details).
- We can do the matrix inversion, WITHOUT inversion! (How is that possible?!)
- A: our matrix has special form: $(\mathbf{X}^T \mathbf{X})$



More details

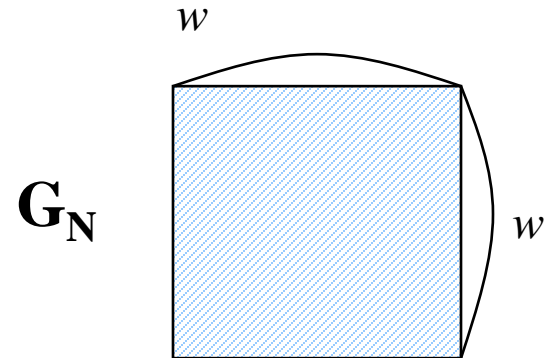
At the $N+1$ time tick:





More details

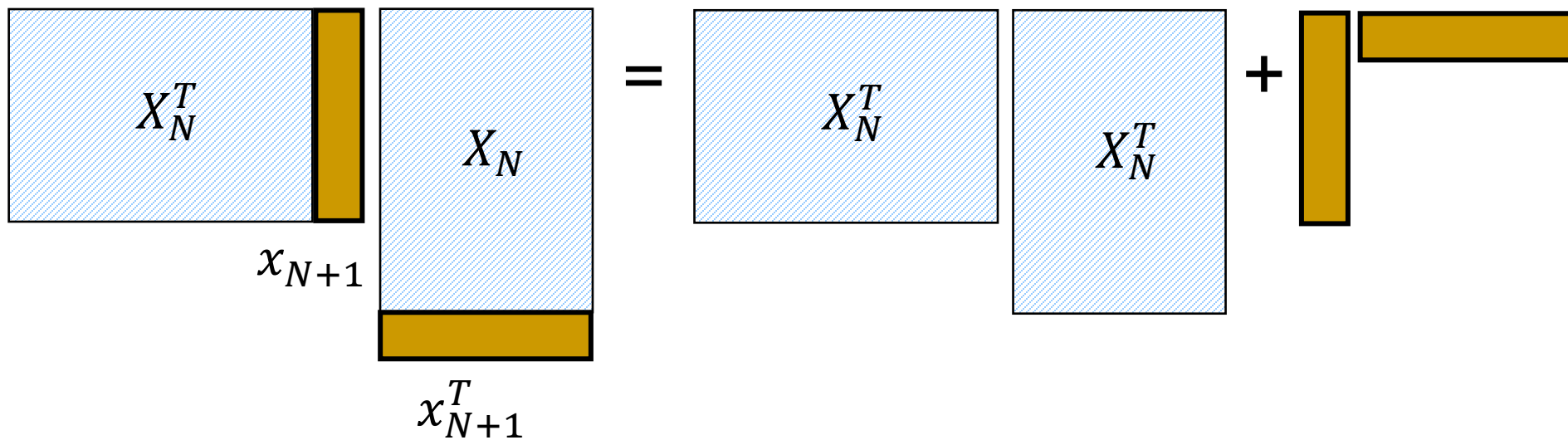
- Let $\mathbf{G}_N = (\mathbf{X}_N^T \mathbf{X}_N)^{-1}$ (“gain matrix”)
- \mathbf{G}_{N+1} can be computed recursively from \mathbf{G}_N





More details

- $G_{N+1} = (X_{N+1}^T X_{N+1})^{-1} = (X_N^T X_N + x_{N+1} x_{N+1}^T)^{-1}$





ASIDE: Sherman-Morrison Formula

Suppose $A \in \mathbb{R}^{n \times n}$ is an invertible square matrix and $u, v \in \mathbb{R}^n$ are column vectors. Then $A + uv^T$ is invertible iff $1 + v^T A^{-1} u \neq 0$. In this case,

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} uv^T A^{-1}}{1 + v^T A^{-1} u}.$$

Here, uv^T is the outer product of two vectors u and v .

https://en.wikipedia.org/wiki/Sherman%E2%80%93Morrison_formula



More details

- Let $G_N = (X_N^T X_N)^{-1}$ (“gain matrix”)
- G_{N+1} can be computed recursively from G_N
- $$\begin{aligned} G_{N+1} &= (X_{N+1}^T X_{N+1})^{-1} = (X_N^T X_N + x_{N+1} x_{N+1}^T)^{-1} \\ &= (X_N^T X_N)^{-1} - \frac{(X_N^T X_N)^{-1} x_{N+1} x_{N+1}^T (X_N^T X_N)^{-1}}{1 + x_{N+1}^T (X_N^T X_N)^{-1} x_{N+1}} \\ &= G_N - \frac{G_N x_{N+1} x_{N+1}^T G_N}{1 + x_{N+1}^T G_N x_{N+1}} \end{aligned}$$

No inversion required, thanks to Sherman-Morrison formula



Comparison:

■ Straightforward Least Squares

- ❑ Needs huge matrix (**growing** in size)
 $O(N \times w)$
- ❑ Costly matrix operation
 $O(N \times w^2)$

■ Recursive LS

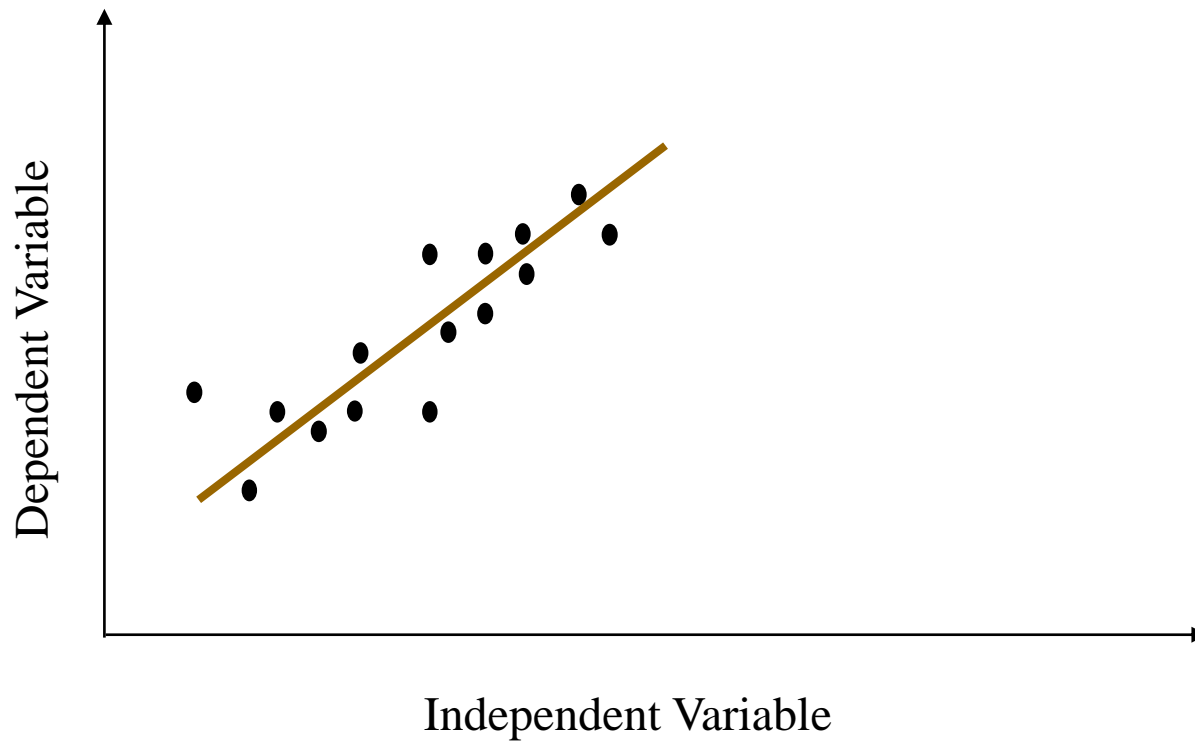
- ❑ Need much smaller, fixed size matrix
 $O(w \times w)$
- ❑ Fast, incremental computation
 $O(1 \times w^2)$
- ❑ **No matrix inversion**

$$N = 10^6, \quad w = 1-100$$



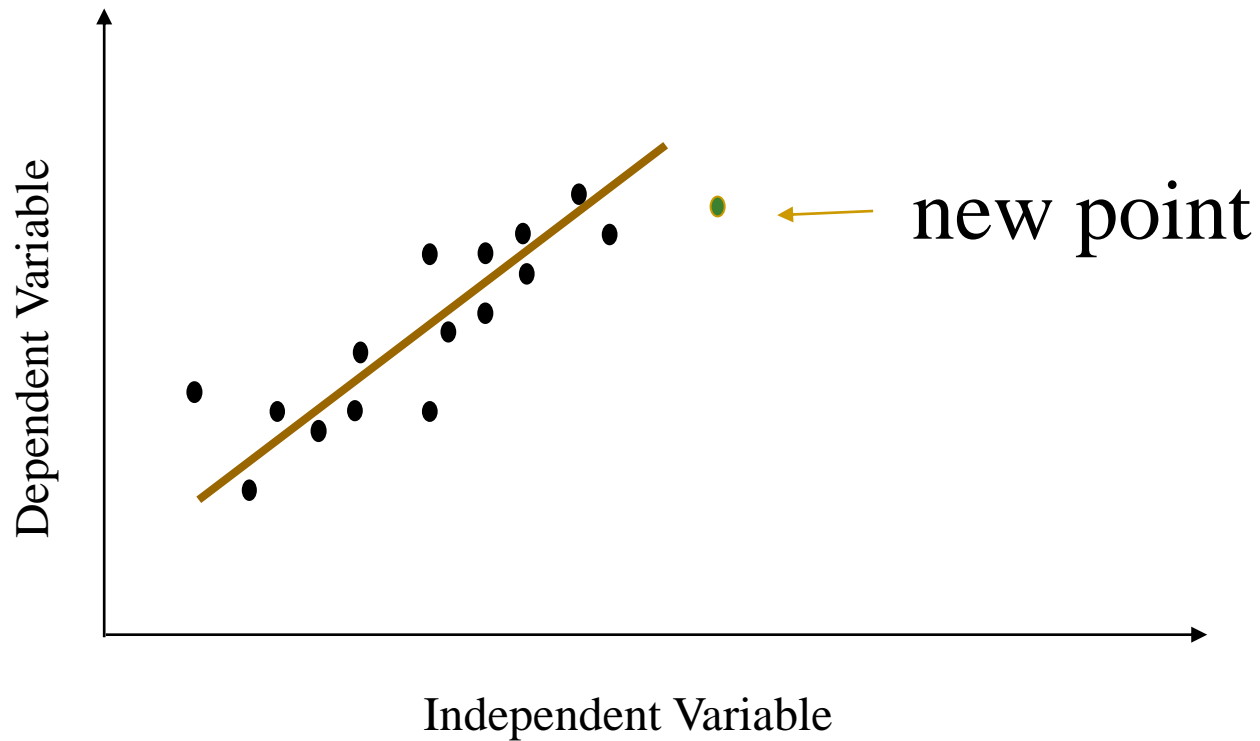
Pictorially:

■ Given:





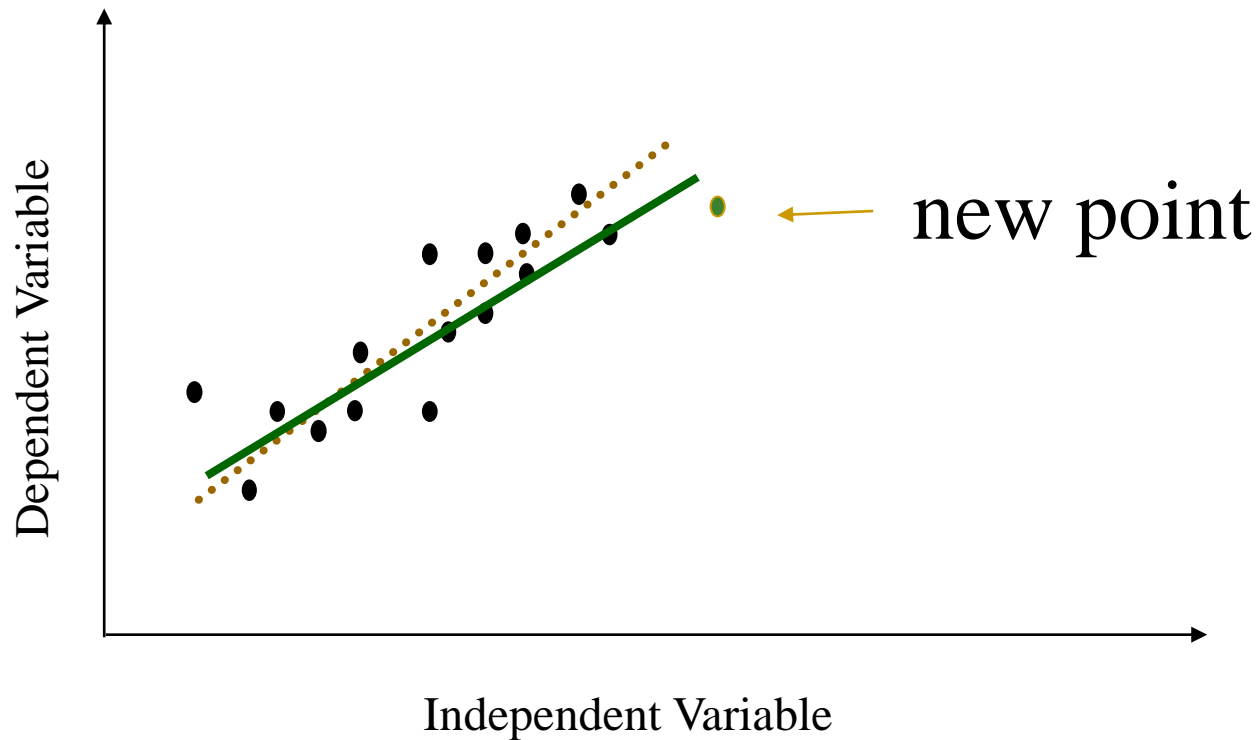
Pictorially:





Pictorially:

RLS: quickly compute new best fit



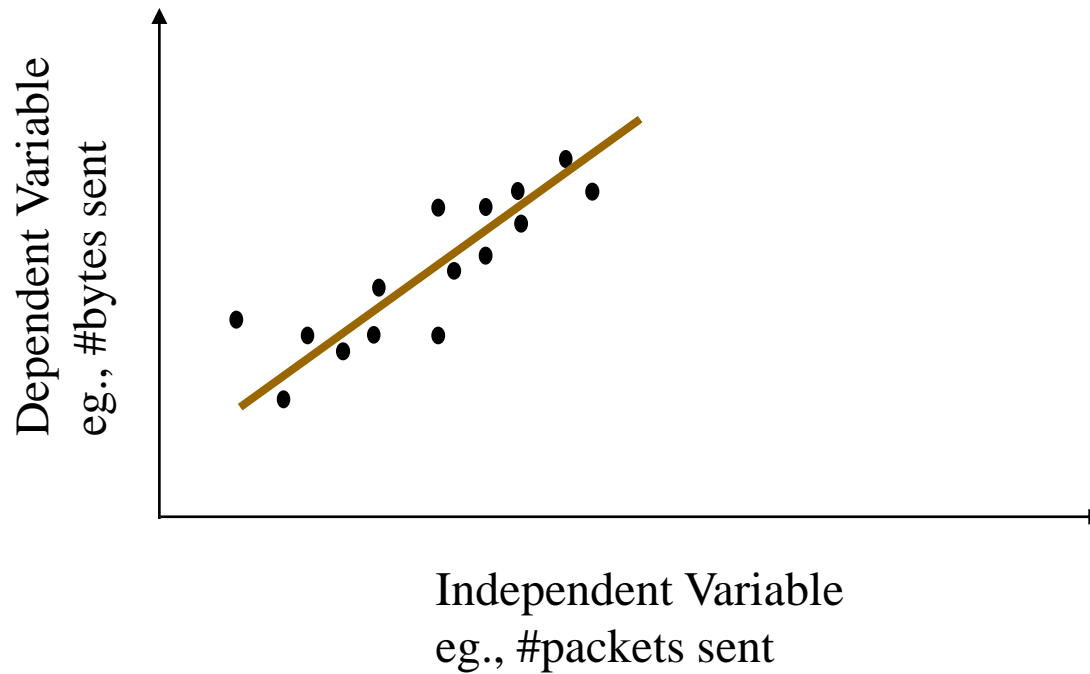


Even more details

- Q4: can we 'forget' the older samples?
- A4: Yes - RLS can easily handle that $[Y_i+00]$:

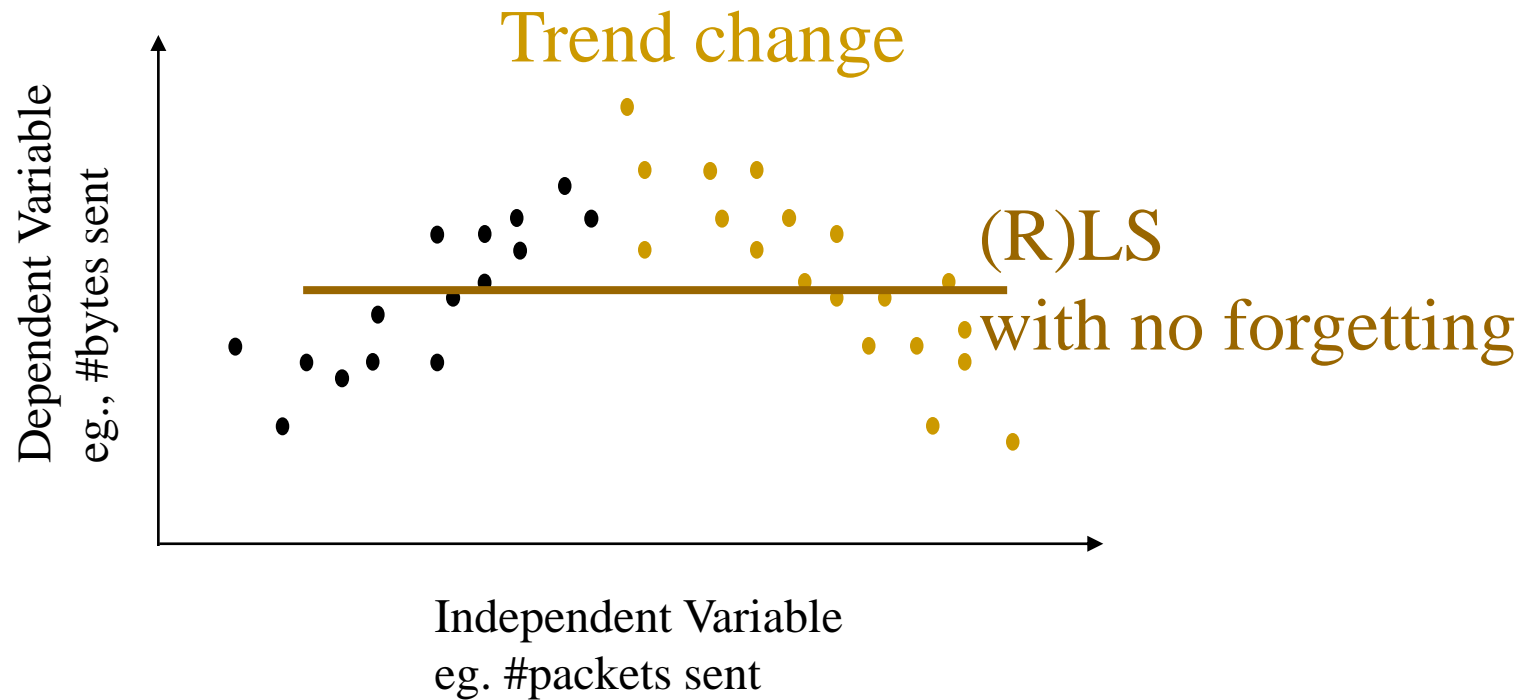


Adaptability - 'forgetting'



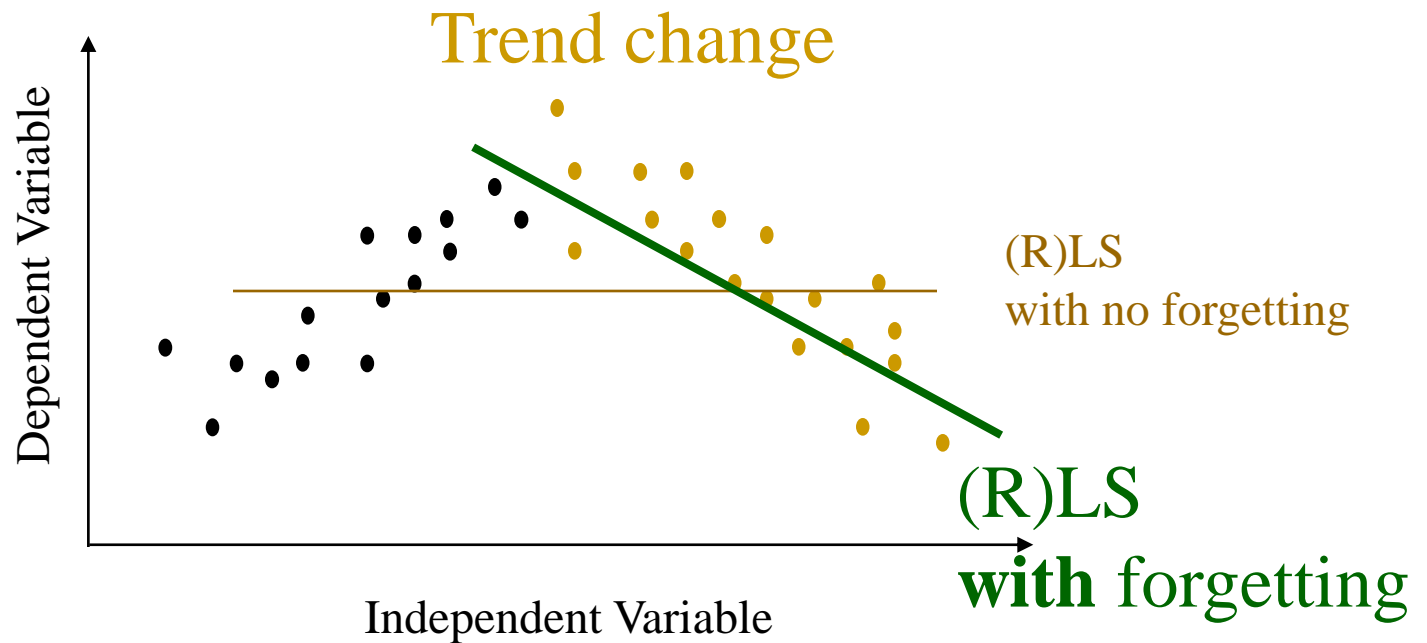


Adaptability - 'forgetting'






Adaptability - 'forgetting'



- RLS: can *trivially* handle 'forgetting'



Outline

- ☒ Motivation
- ☒ Similarity Search
- ☒ Linear Forecasting
-  ☐ **Deep Learning Methods**



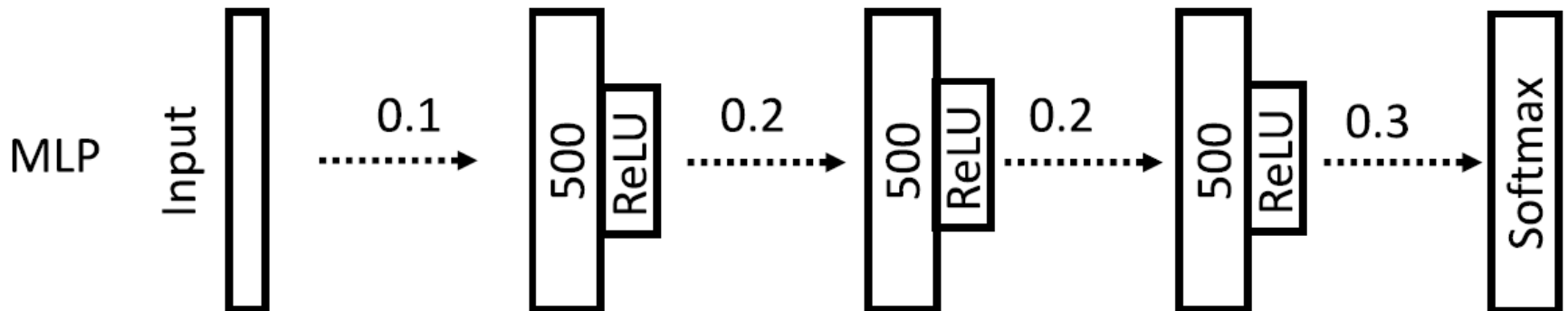
Overview

- Deep learning methods are applicable for both time series classification and regression



Multi Layer Perceptron (MLP)

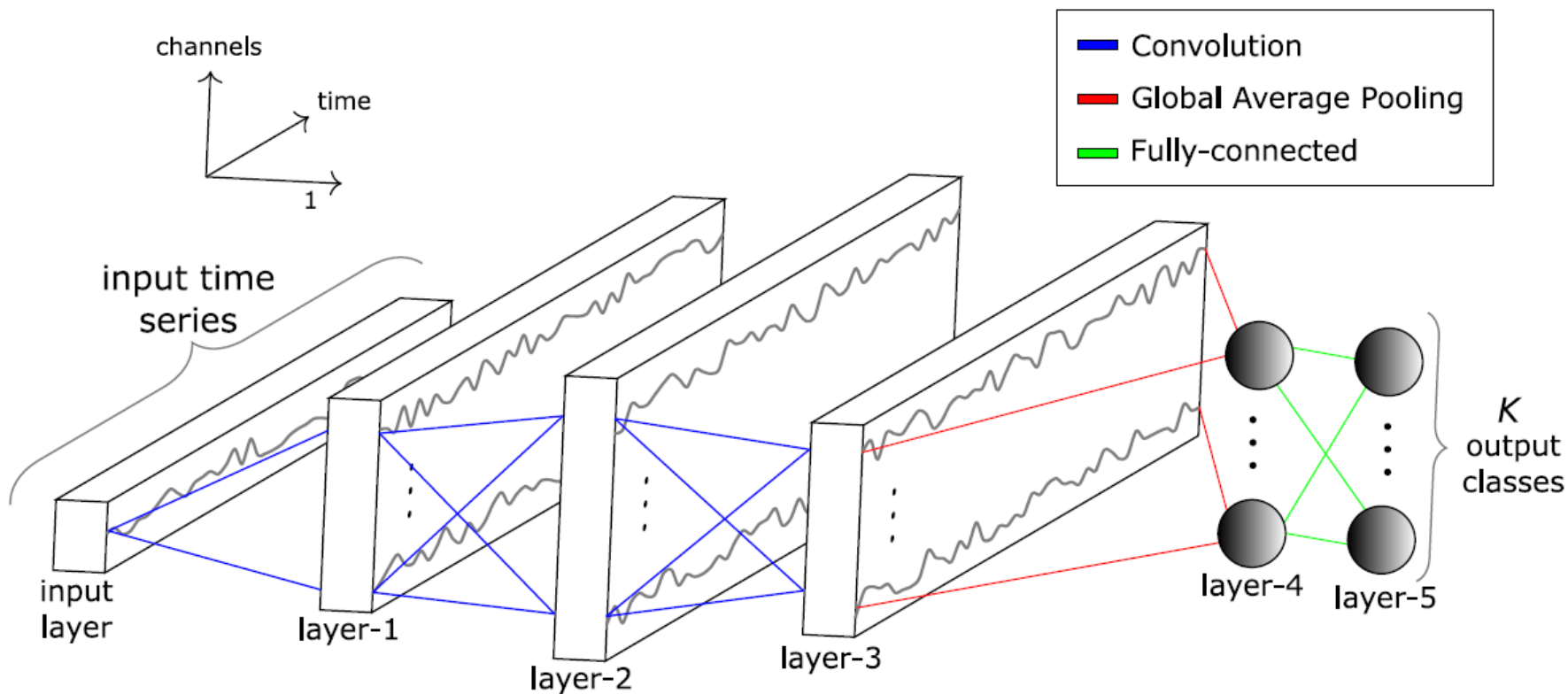
- The simplest form of a deep learning architecture
- Contains neurons that are fully-connected to each other
- Loses the temporal information when learning the features
- Depends on the length of the input time series
- Does not contain any transferable layer (across different datasets)





Convolutional Neural Network

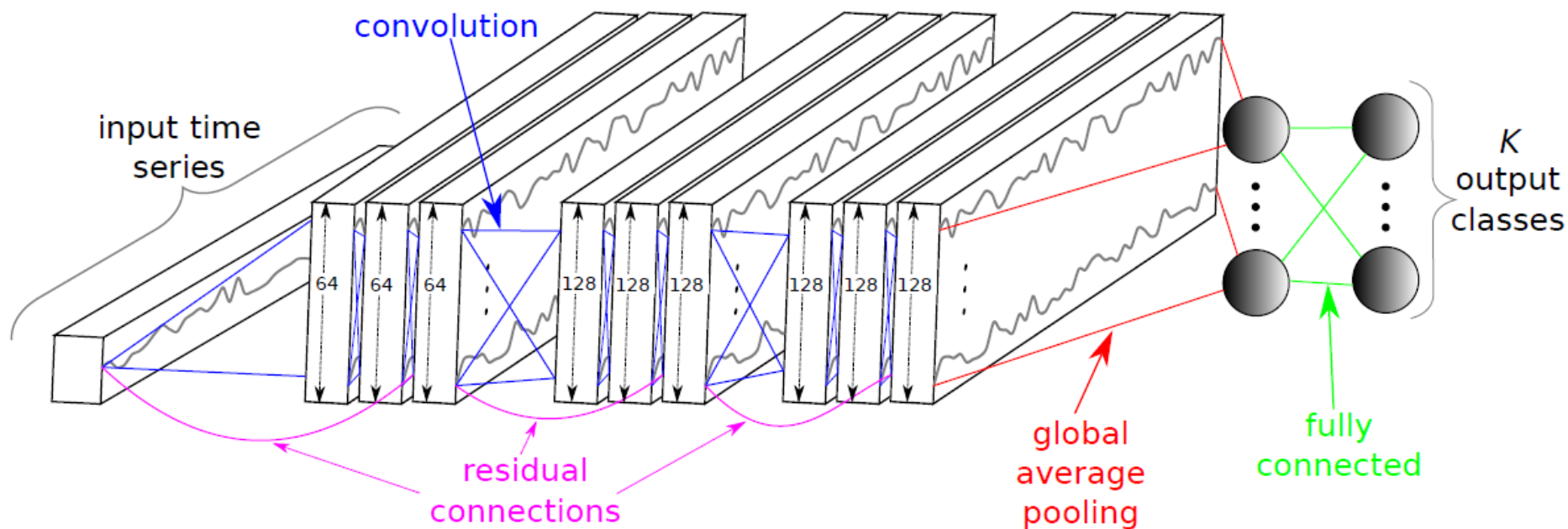
- Detects discriminative features that are temporally invariant
- Contains 4 transferable layers (out of 5 in total)





Residual Network (ResNet)

- A deeper FCN with residual connections
- Has the ability to learn to skip unnecessary convolutions

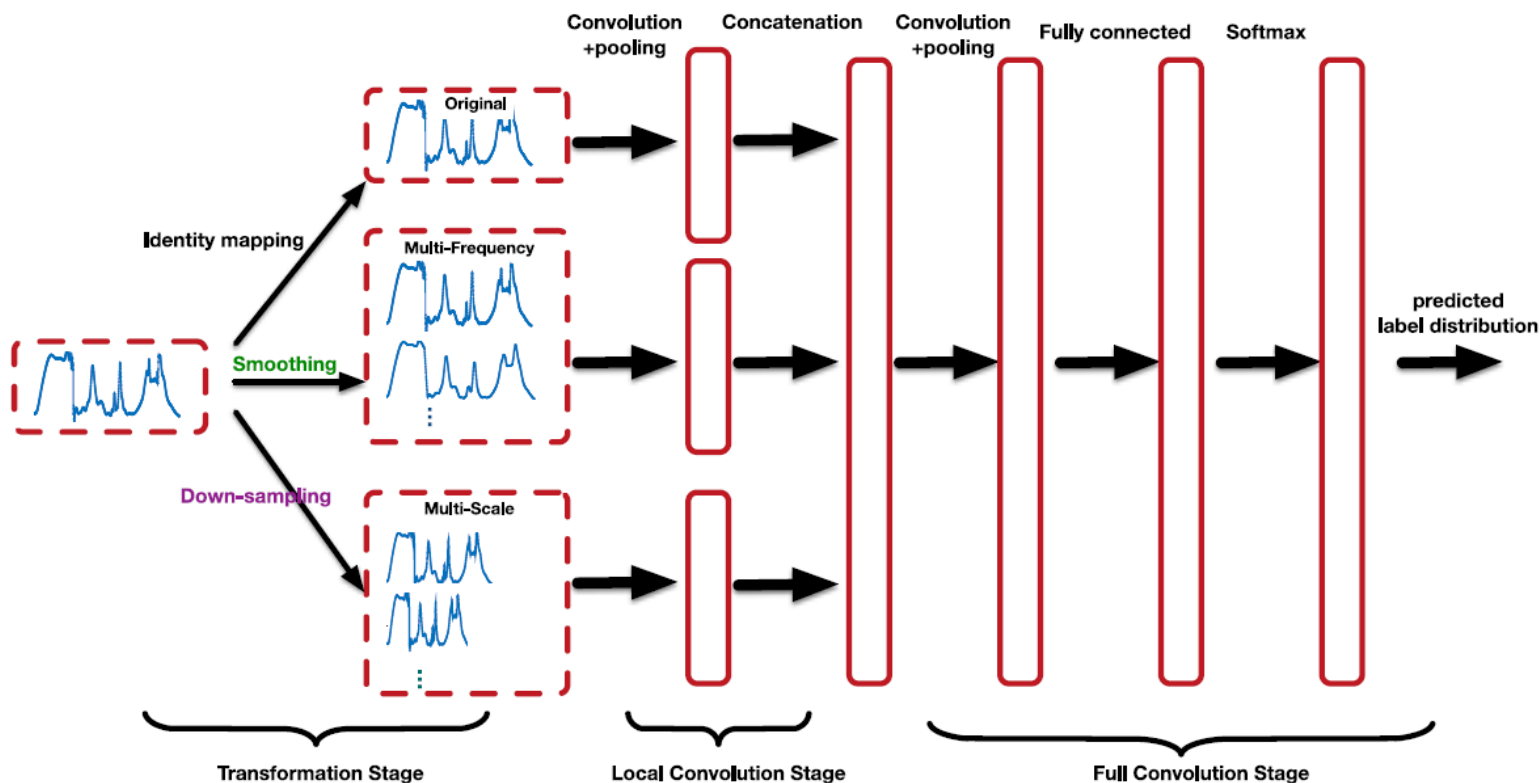




Multi-scale CNN (MCNN)

[Cui et al., Multi-scale convolutional neural networks for time series classification]

- One of the earliest DNN approaches validated on the UCR archive
- Transforms an univariate time series into multiple time series
- Introduces a cropping data augmentation technique

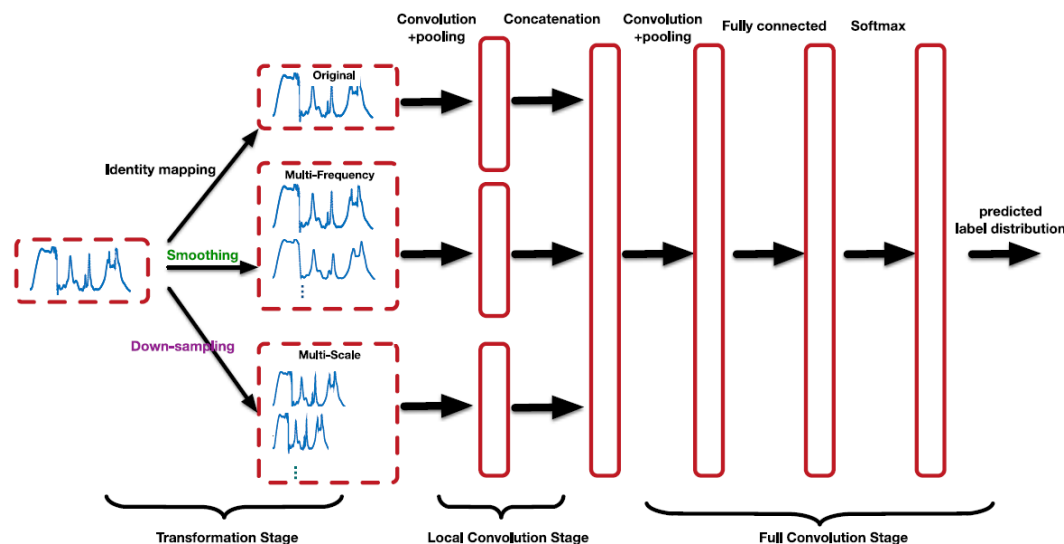




Multi-scale CNN (MCNN)

[Cui et al., Multi-scale convolutional neural networks for time series classification]

- Multi-frequency: remove noises by adopting low-freq. filters
 - Generate multiple new input time series by smoothing (using moving averages) with different window sizes
- Multi-scale: capture temporal patterns at different time scales (long-term and short-term)
 - Generate multiple new input time series by keeping only every k th data points





What You Need to Know

- Motivation of time series analysis
- Similarity search
- Linear forecasting
- Deep learning methods



Questions?