Data processing for IoT

Sandor Markon
Kobe Institute of Computing

ICTP Workshop on Open Source Solutions for the Internet of Things, Jun 25-29, 2017

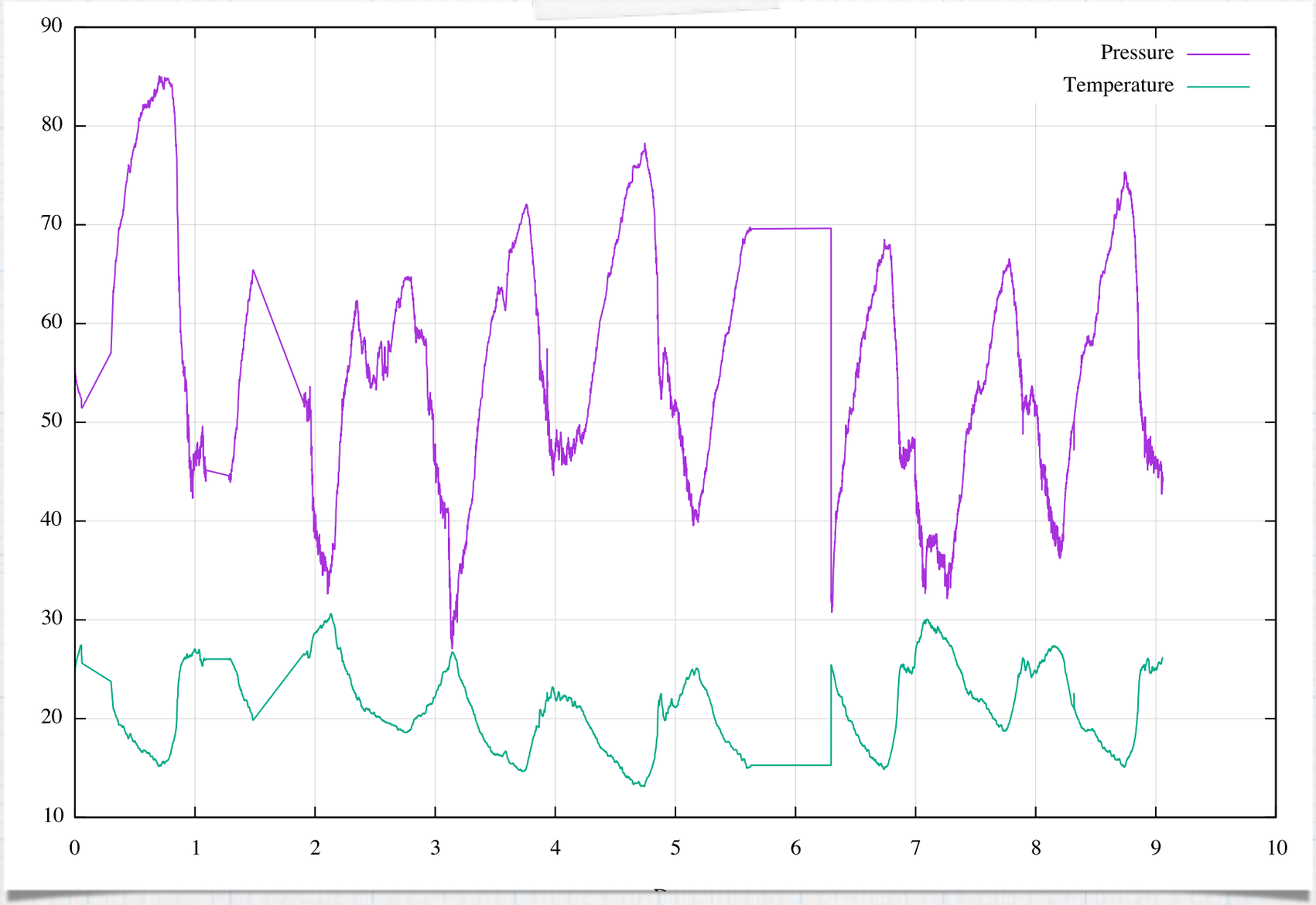
Brief self-introduction

* Born in Budapest, got Dipl. Ing. degree in Electrical Engineering
* 6 years in electromagnetic field analysis for a Hungarian manufacturer, incl. 2 years in Japan
* 29 years in computer applications for a Japanese manufacturer, PhD from Kyoto University
* 12 years teaching at Kobe Institute of Computing
* Besides: 2.5 companies, visiting researcher at hospital...

What to do with your data in the cloud?

* Data:
  capture - send - collect - now what???

* Convert "data" into "information"
* Convert "information" into "knowledge"
* Convert "knowledge" into "action"

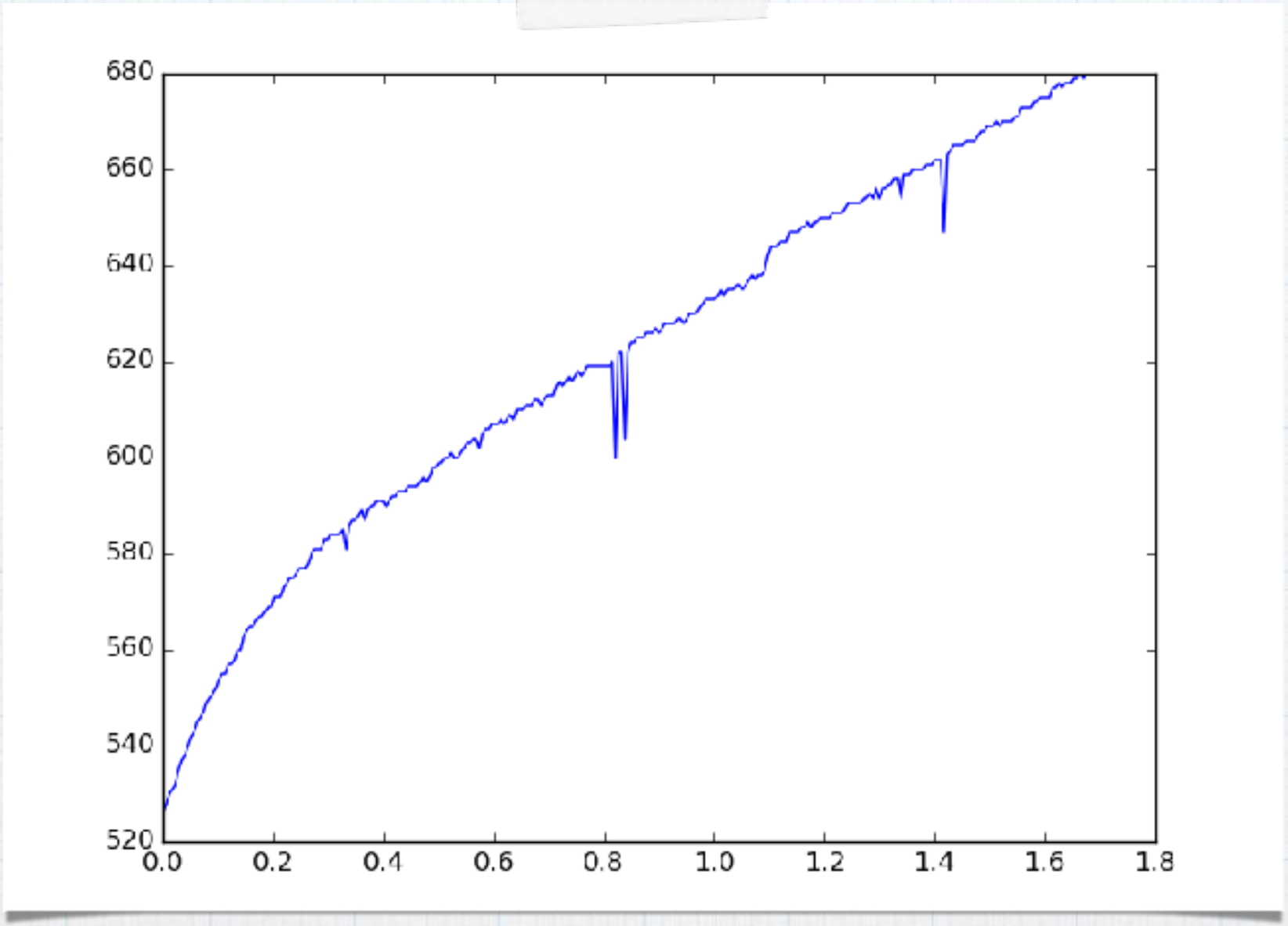


Temporal data

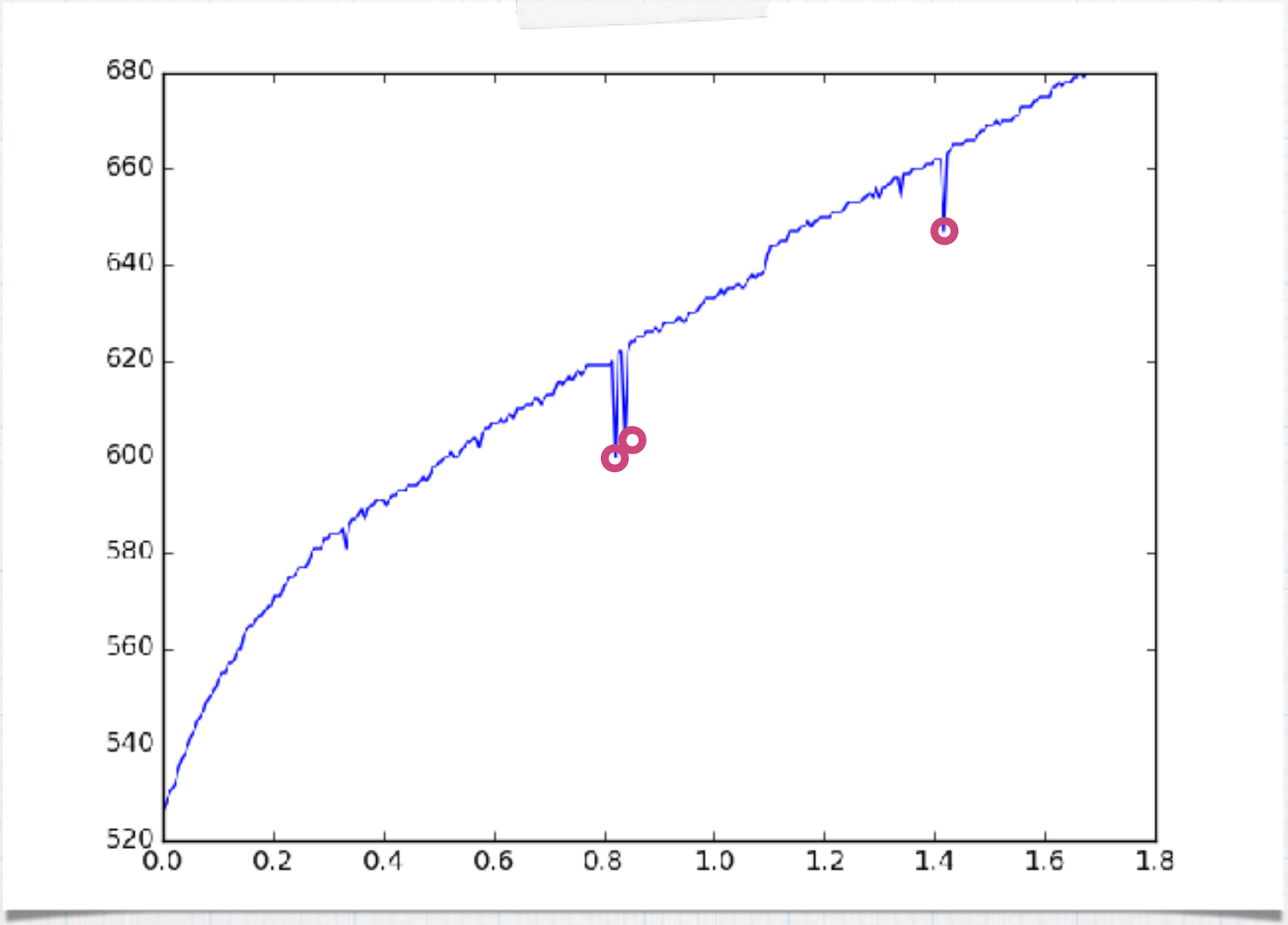
Time series: evidence of some process going on
Usual task: prediction

What to do with time series?

* Clean up the dirty data:
  noise
  outliers
  missing data
* Visualize it
* Predict the future



Data with outliers

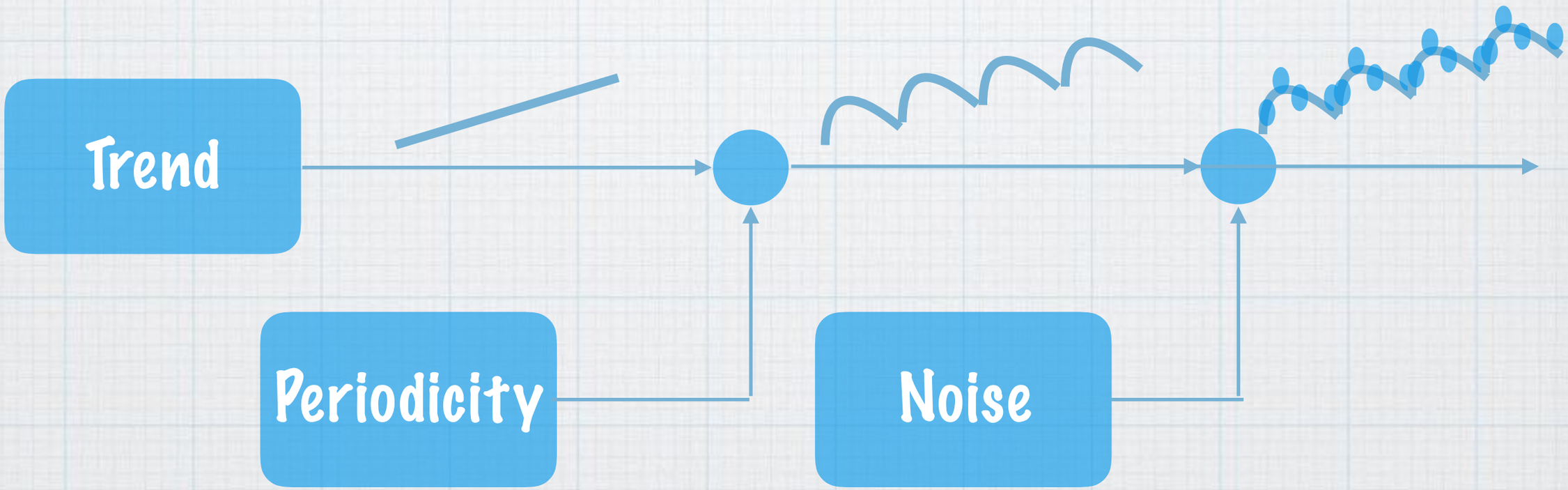


Data with outliers

The spikes are not real data

Why can we predict?

* Data is generated by some process
* Usual assumption: deterministic + noise



ARIMA model

$$X_t - \alpha_1 X_{t-1} - \cdots - \alpha_{p'} X_{t-p'} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$

* The measurement at time t depends on previous instances $t-1, t-2, \ldots, t-p$
* It also depends on the "noise" at t and at $t-1, t-2, \ldots, t-q$

Time series in Python

* Numerics: numpy, scipy
* Plotting: matplotlib
* Time series: pandas
  http://pandas-docs.github.io/pandas-docs-travis/
* Prediction (statistical): pyflux
  http://www.pyflux.com
* Prediction (AI): FB Prophet
  https://arnesund.com/2017/02/26/using-facebook-prophet-forecasting-library-to-predict-the-weather/



Healthy



Diseased

Image data

Photos, videos: evidence of some status
Usual task: classification

Images + Python + AI

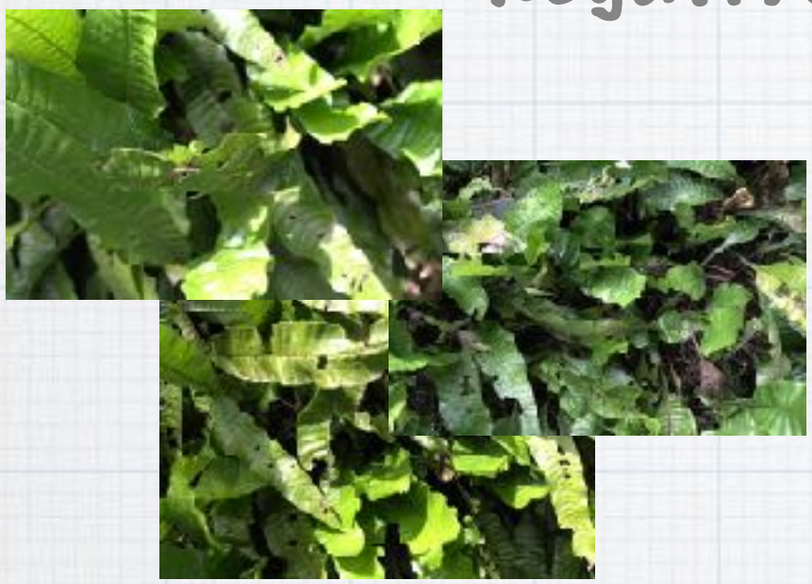
* Basics: OpenCV
* Image features: Mahotas
* Classification: milk

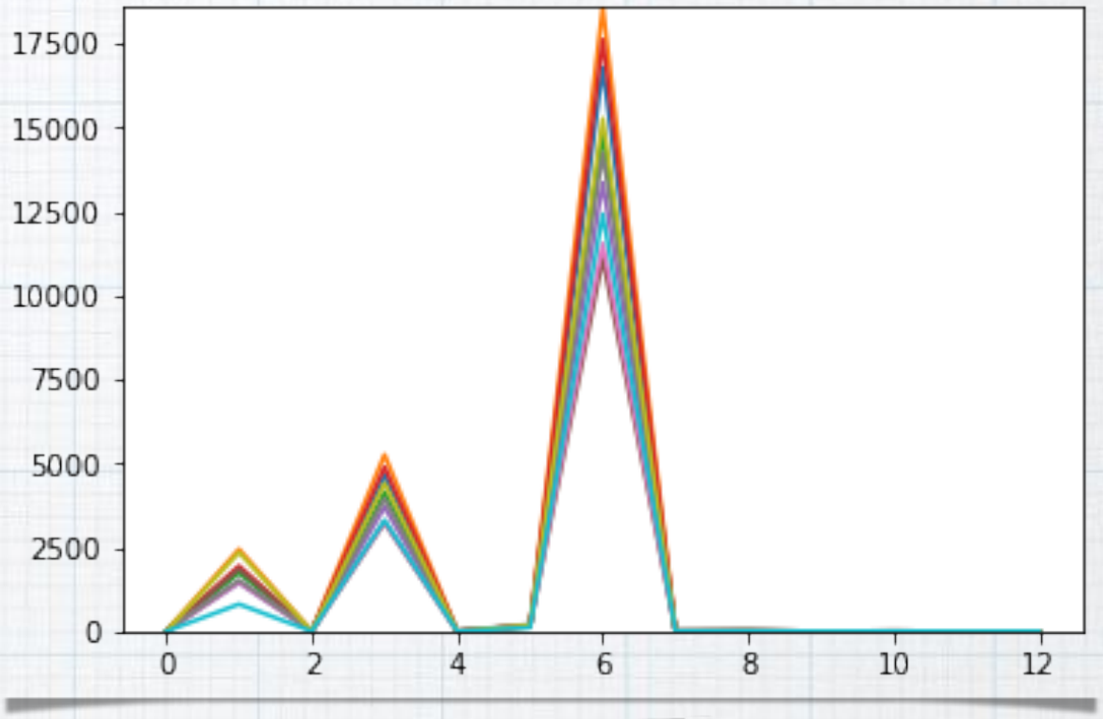
(or scikit-learn: http://scikit-learn.org/stable/)

Simple AI for images

* Generate numerical feature vectors
* Select part of the images as a training set:
  N_p positive samples
  N_n negative samples
* Train a classifier with the training set
* Test it with the rest of the data

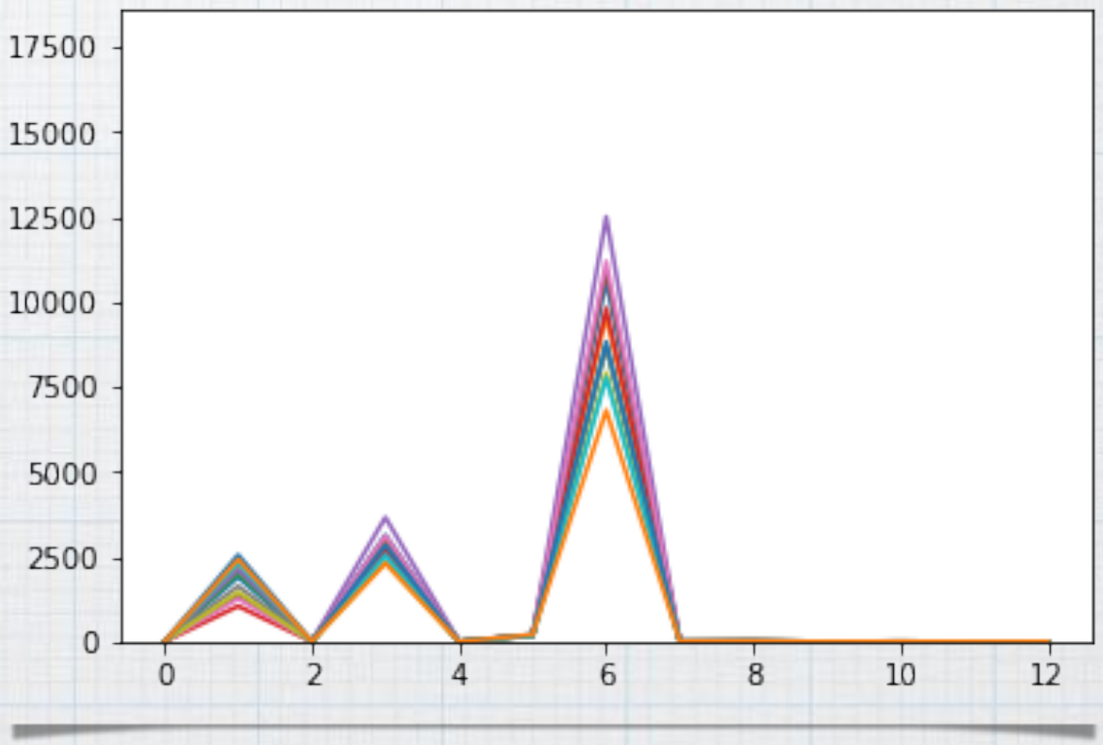
negatives



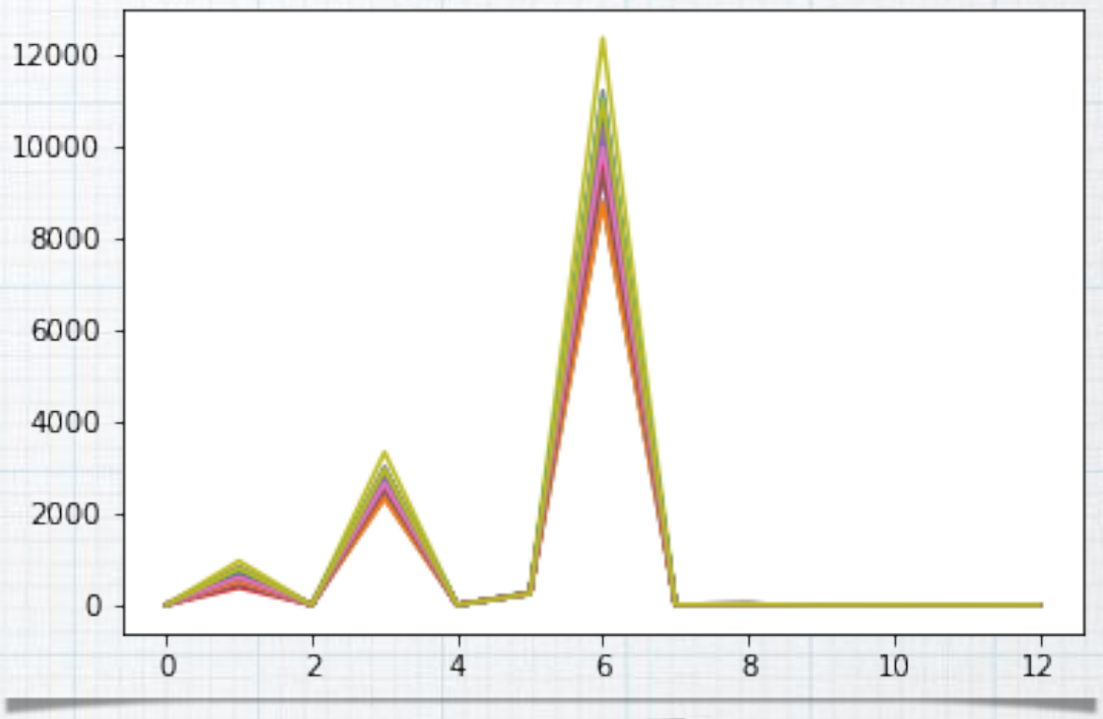


Features

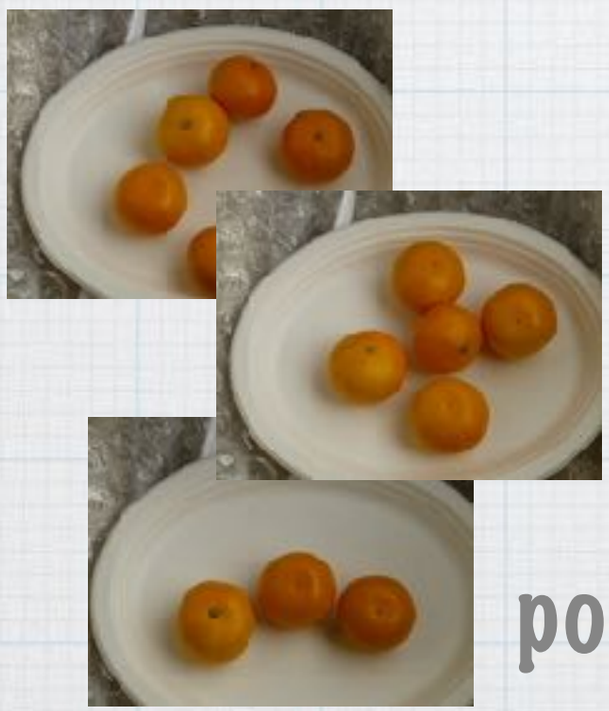
positives



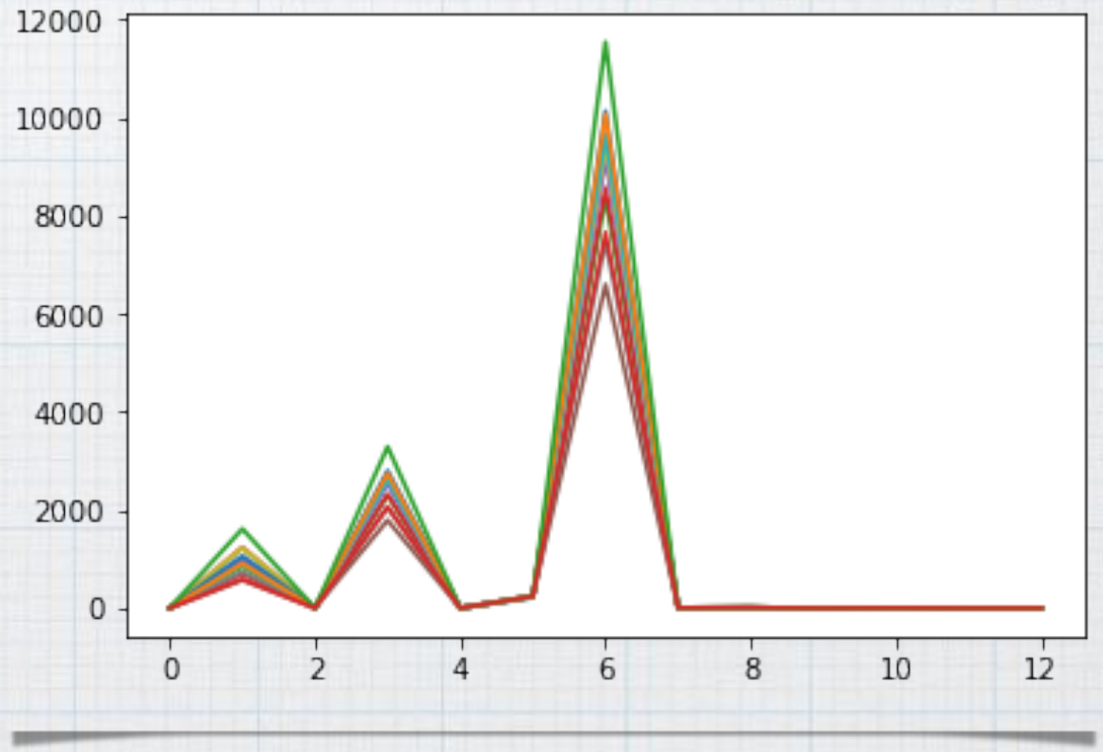
negatives



Features

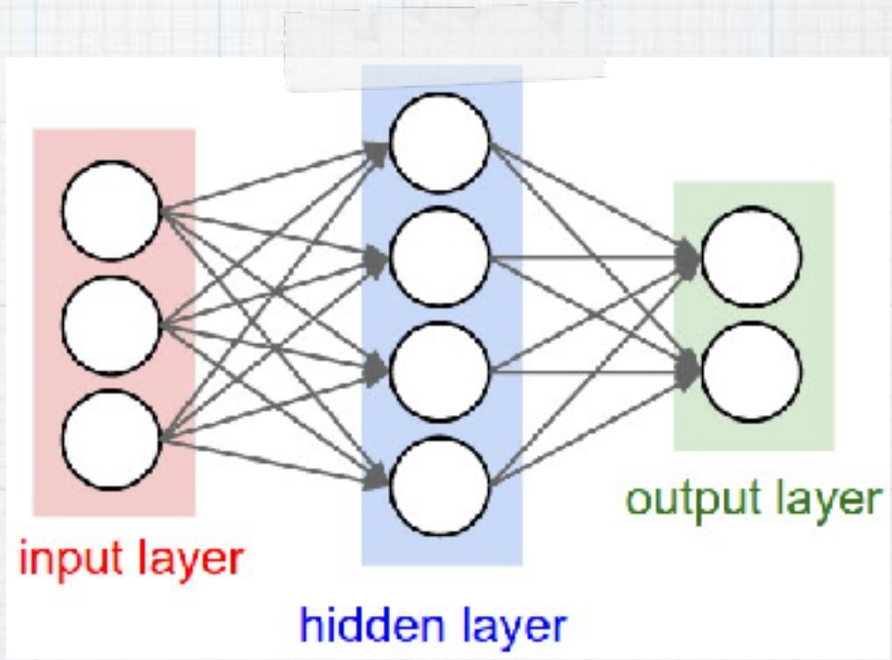


positives



Neural network

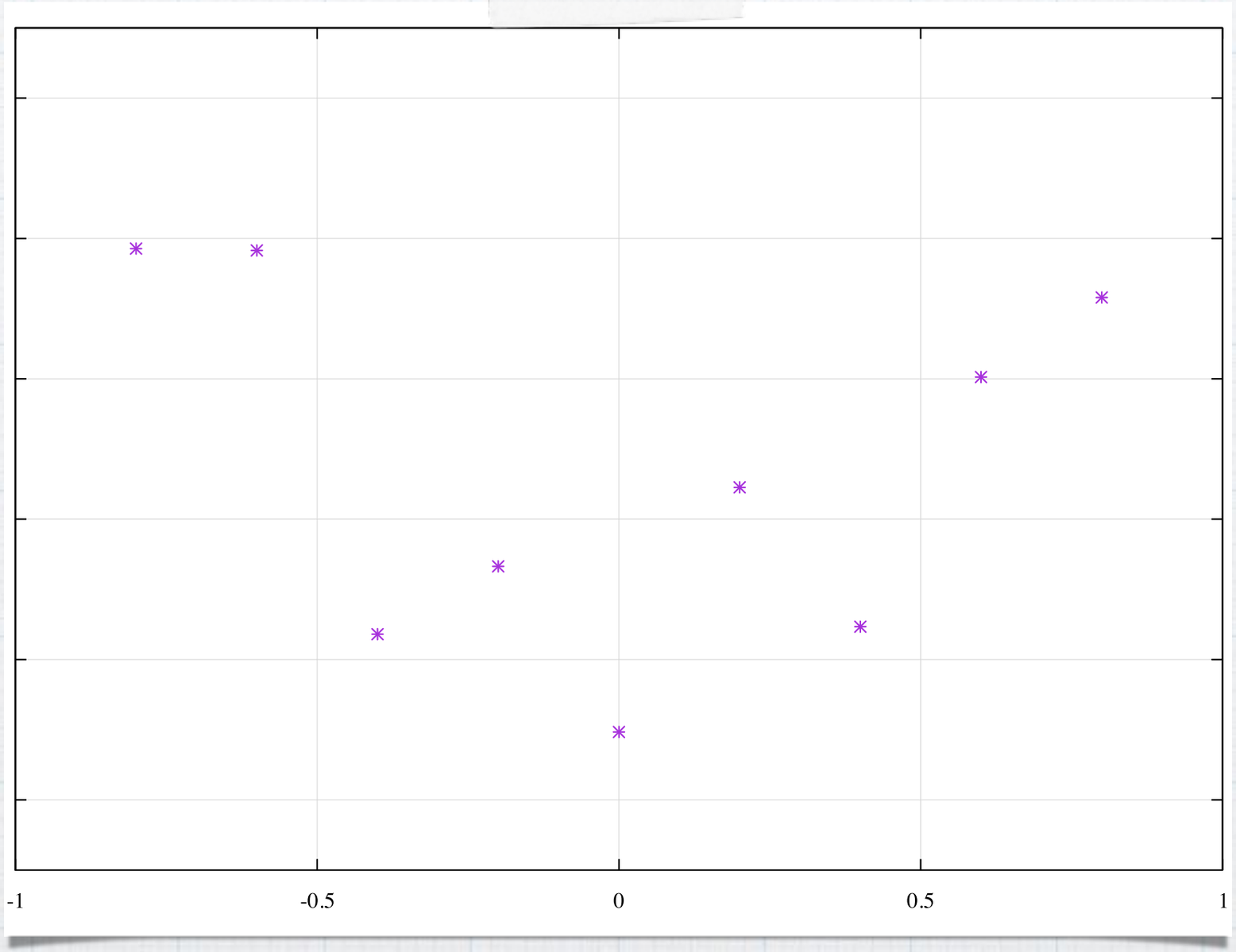
* Feed features to input layer
* Calculate hidden and output layer activations through connection weights
* Compare output with known correct output
* Adjust weights until output becomes correct for all input samples



Classification

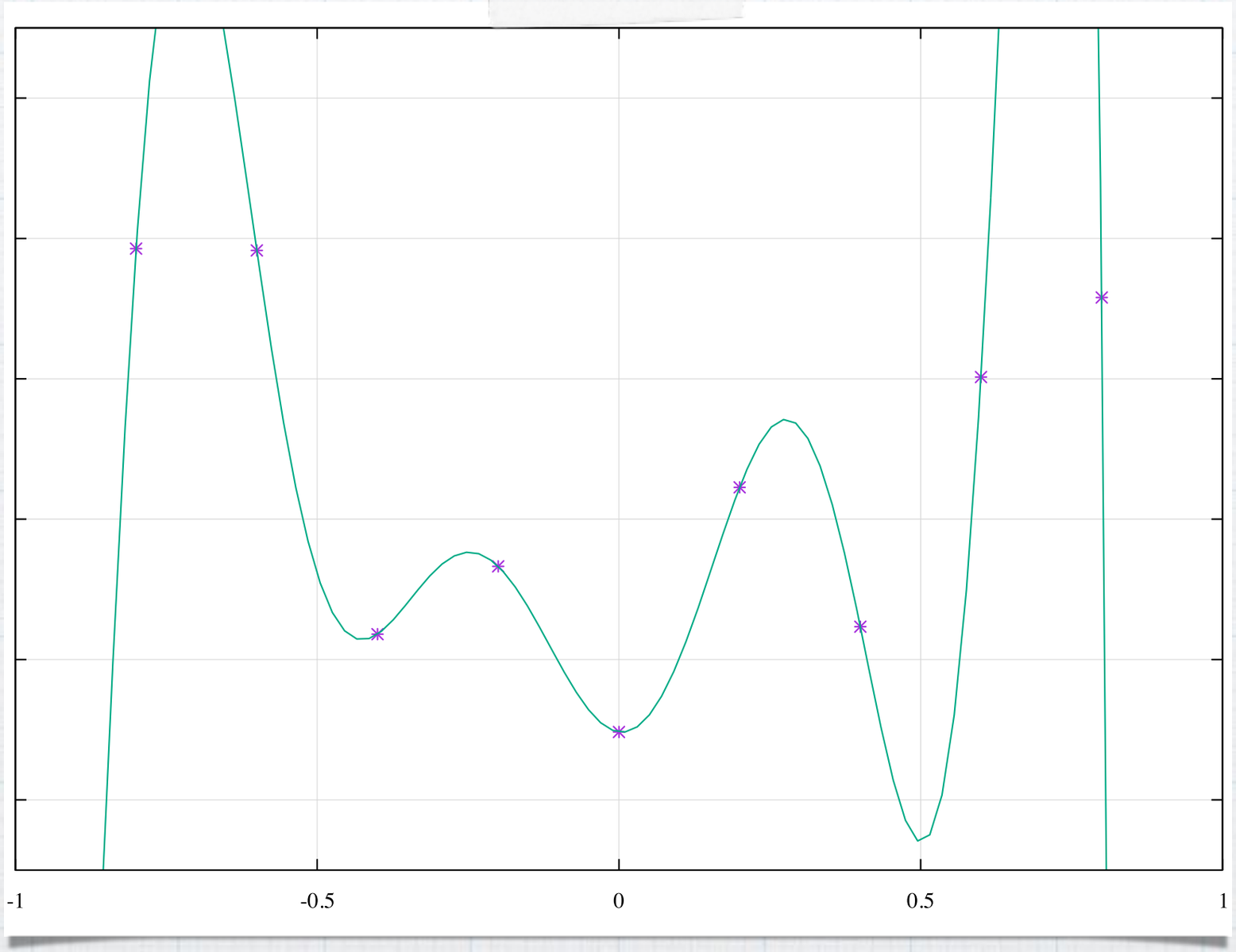
* After training, the neural network classifies correctly the training samples
* Hopefully it will also correctly classify unknown data
  * Some common problems (and many others...):
    * overfitting (can classify only the training set)
    * poor flexibility (too simple for the task)
  * Illustration: with curve fitting

Training data



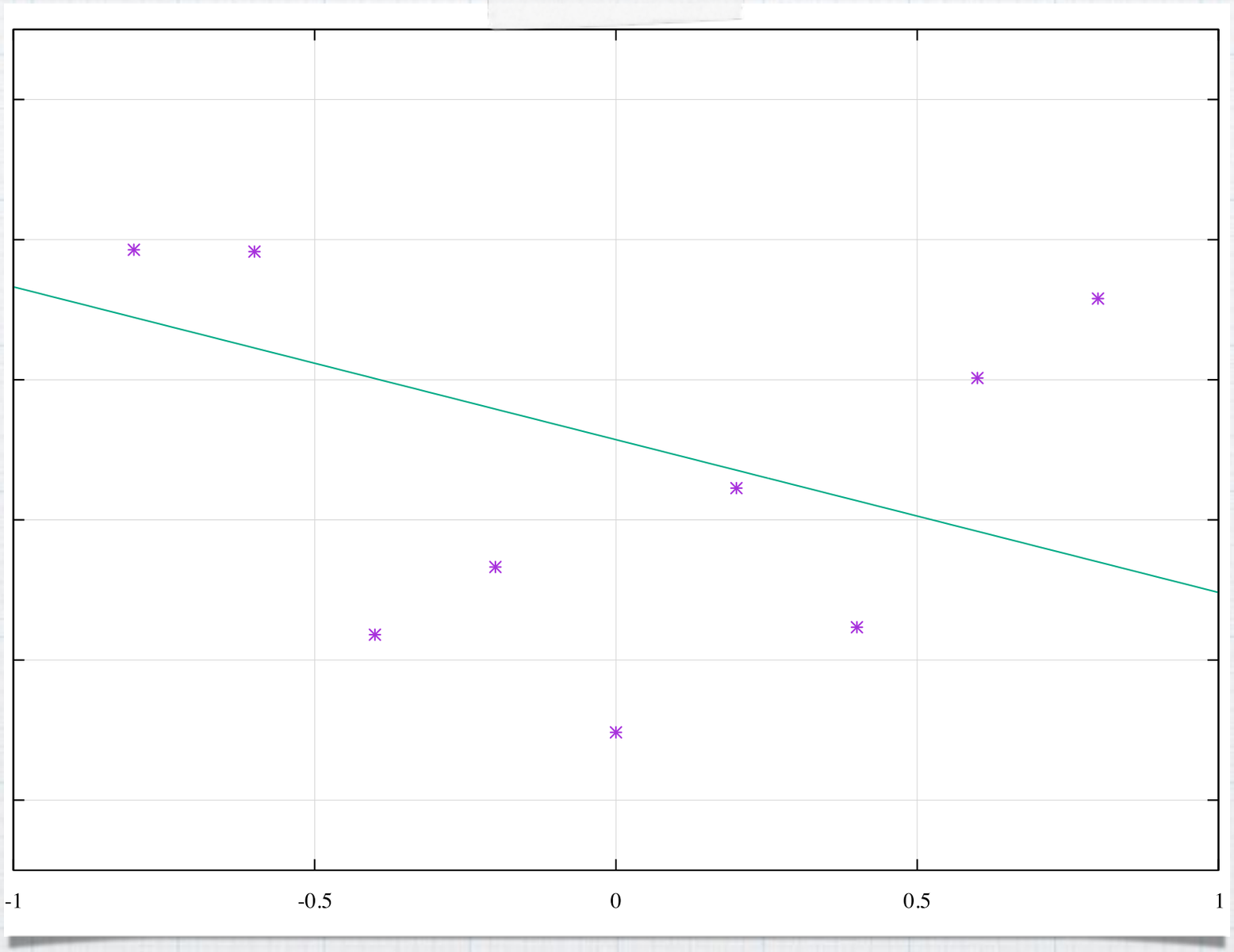
Find a good match for the points

Overfitting



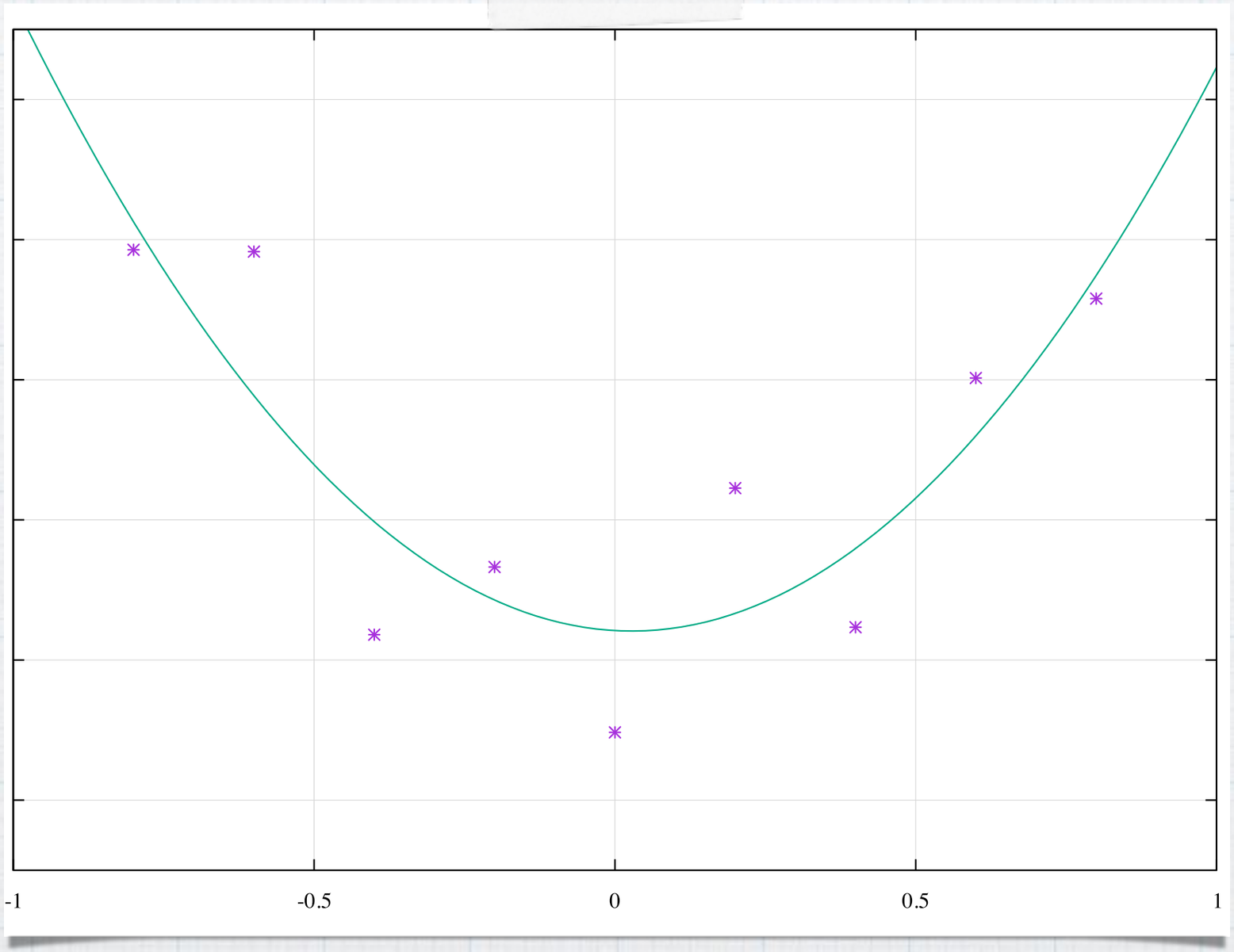
Too generic function: cannot find the hidden rule

Underpowered



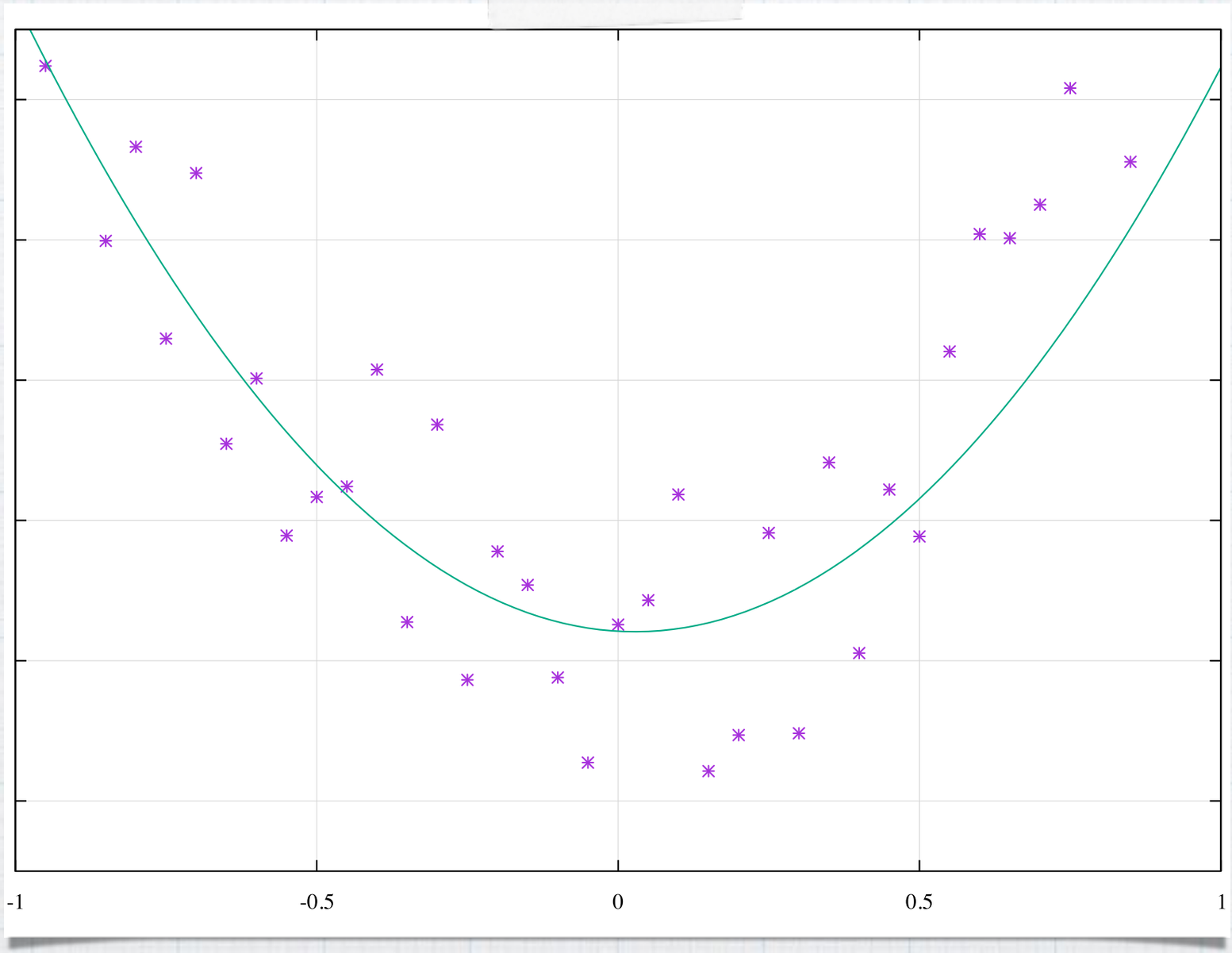
Too simple function: cannot follow the data

Best fit



2nd order polynomial matches the hidden rule

Best fit, new data



A good fit will match the new data too

Some hints for neural networks

* Prepare **LOTS** of data!
* Choose the right network topology (multi-layer, feed-forward; recurrent; etc.)
* Choose the right complexity
* Use regularization to fight over-fitting
* Try self-organizing feature processing