KU LEUVEN

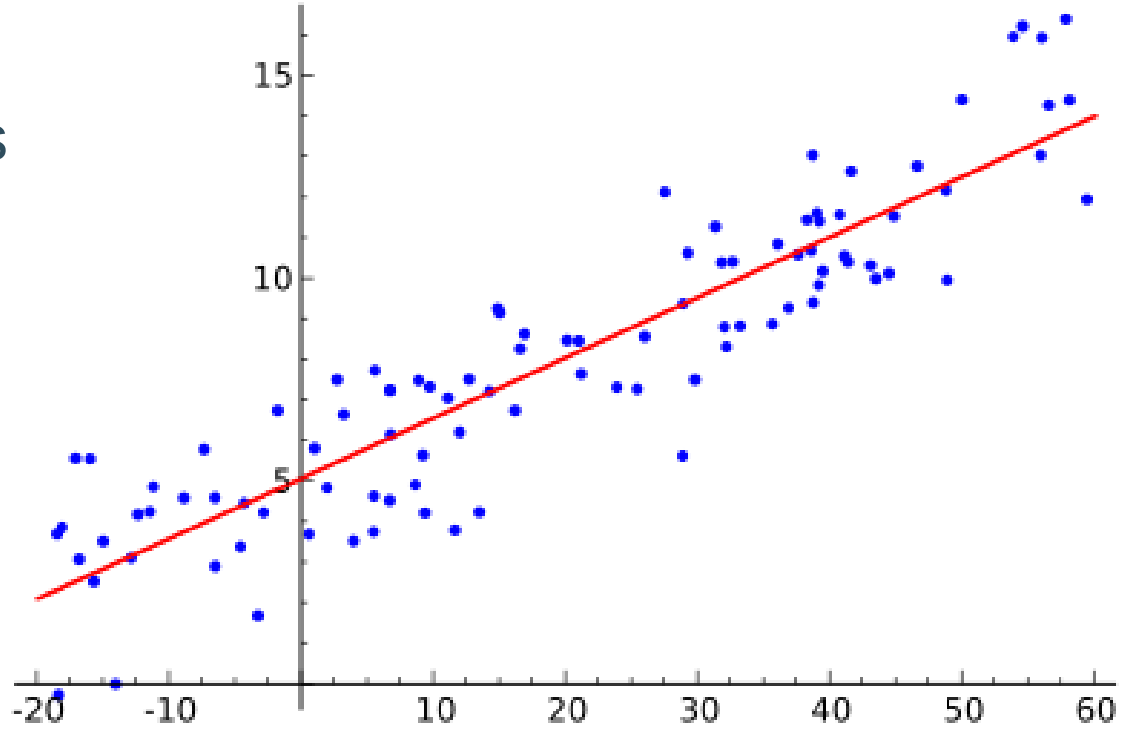
Smart Distribution Systems
Exercise session 1

Mahtab Kaffash – mahtab.kaffash@kuleuven.be

Thijs Peirelinck – thijs.peirelinck@kuleuven.be

Regression

- Statistical model
- Estimating relationships among variables
  - Dependent variable
  - Independent variables
- Linear regression
- Non-linear regression

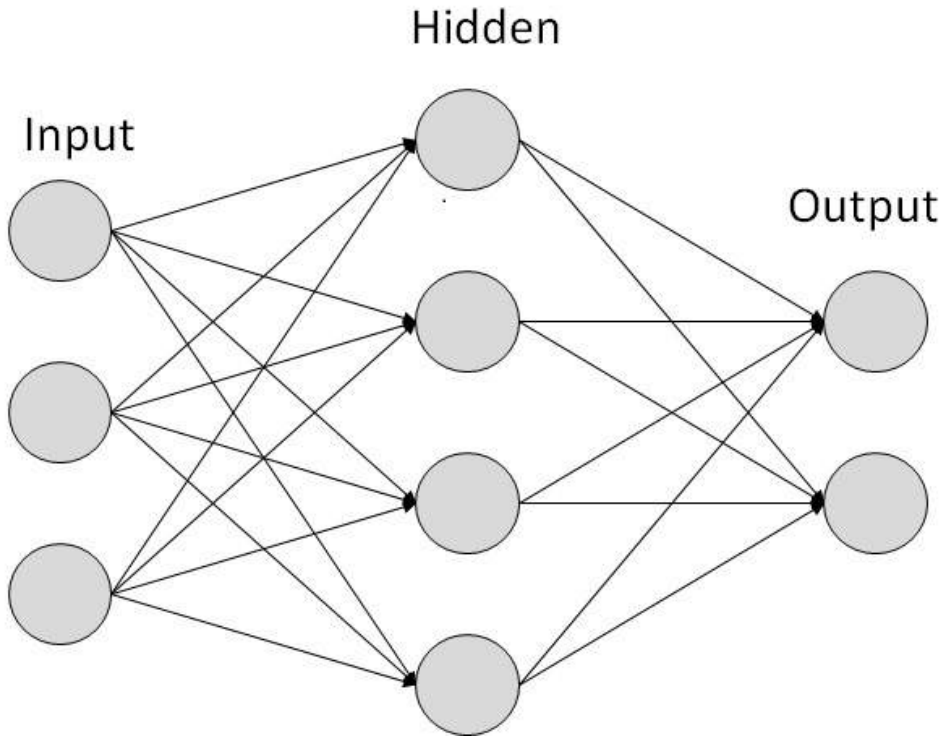


Artificial Neural Networks

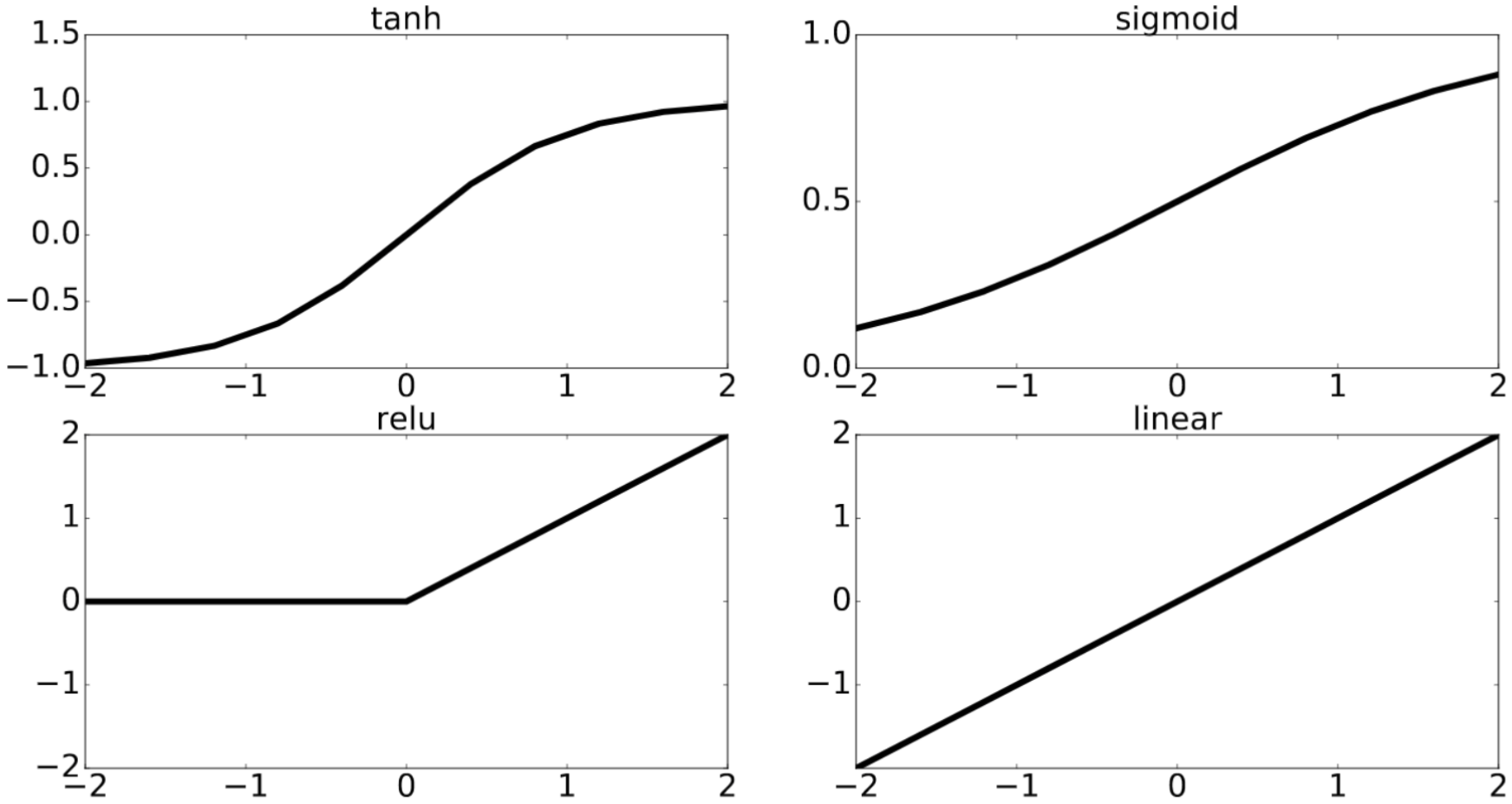
- Computing system capable of massive data processing and knoweldge representation
- Universal approximation theorem: "the standard multilayer feed-forward network with a single hidden layer, which contains finite number of hidden neurons, is a universal approximator among continuous functions on compact subsets of Rn, under mild assumptions on the activation function."

Artificial Neural Networks - Structure

- Organized in layers: neurons, inputs, outputs and activation function
- Number of neurons in output layer = number of outputs
- Used in optimization, control and **forecasting**



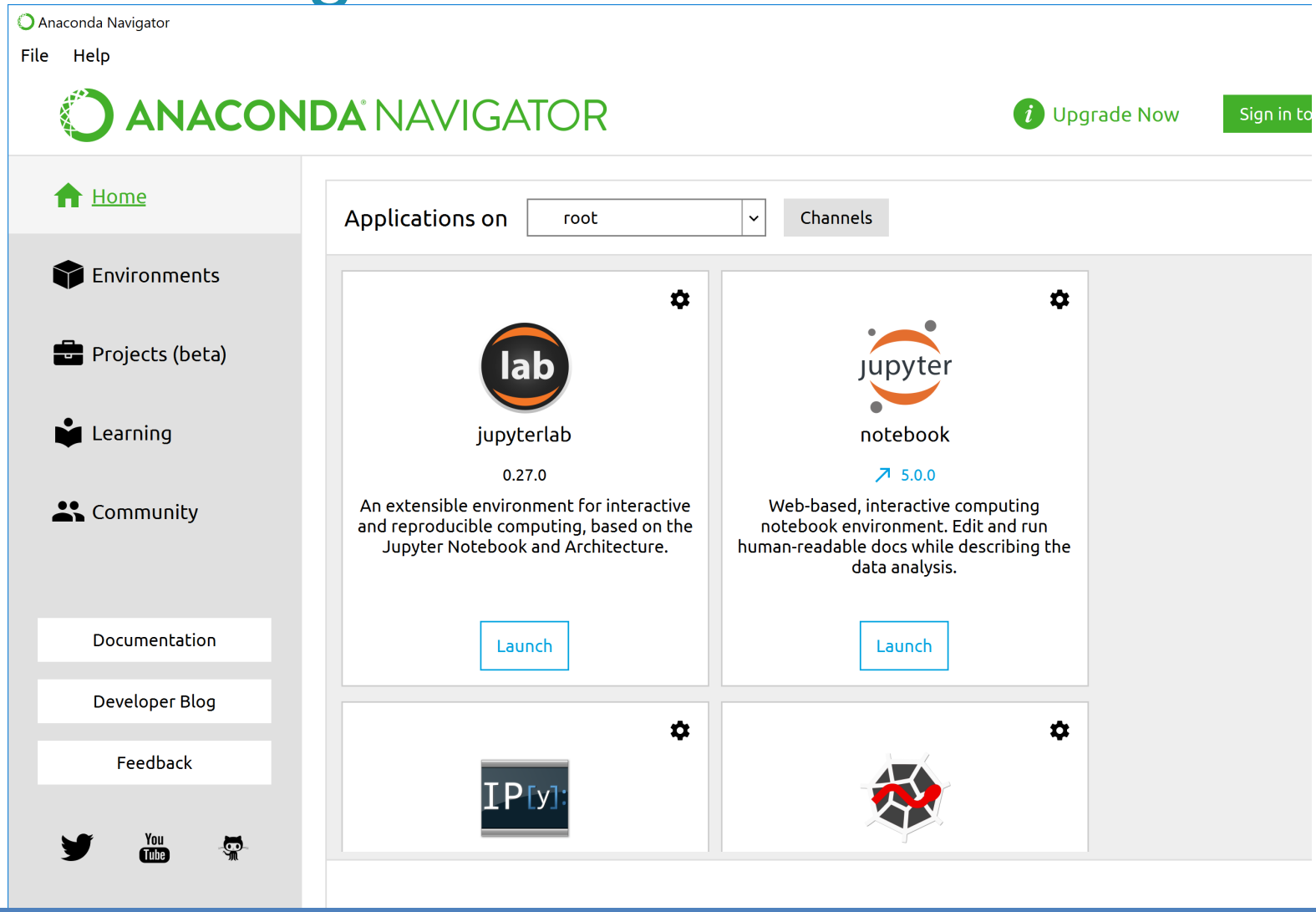
Artificial Neural Networks – Activation functions



Set-up Machine Learning Environment

- Download and install Anaconda
- Install deep learning libraries
- More info on Toledo

Anaconda navigator



Get started

- Download all exercises: https://github.com/thijsp/Machine-Learning-in-Power-Systems
- Open Anaconda
- Import *environment.yml*
- Launch jupyter lab (in correct environment)
- Brows to exercise folder
- Launch *exercise_session_1.ipynb*
- Read all information (go to provided links with additional information) and complete the exercises.

Documentation

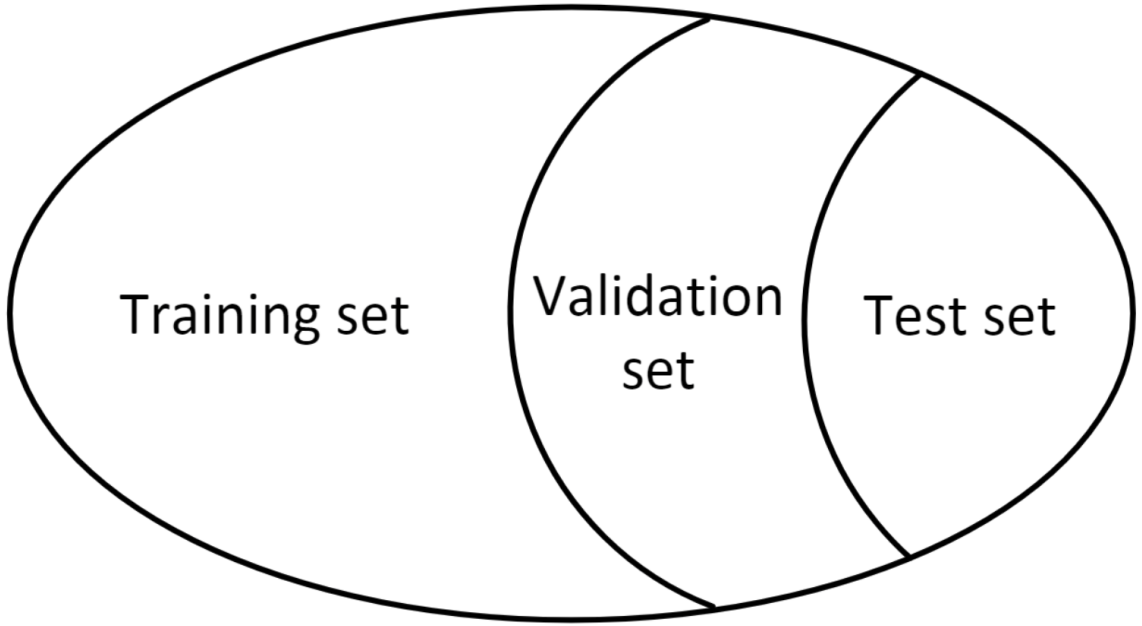
- Numpy: https://docs.scipy.org/doc/
- Pandas: https://pandas.pydata.org/pandas-docs/stable/
- Scikit-learn: http://scikit-learn.org/stable/documentation.html
- Keras: https://keras.io/

Lab session tasks

1. Linear regression
    1. Generating data
    2. Plotting
2. Non-linear regression: kernel regression
3. Non-linear regression: neural networks
4. Overfitting
5. Regularization: early-stopping
6. Importing real data: DataFetcher

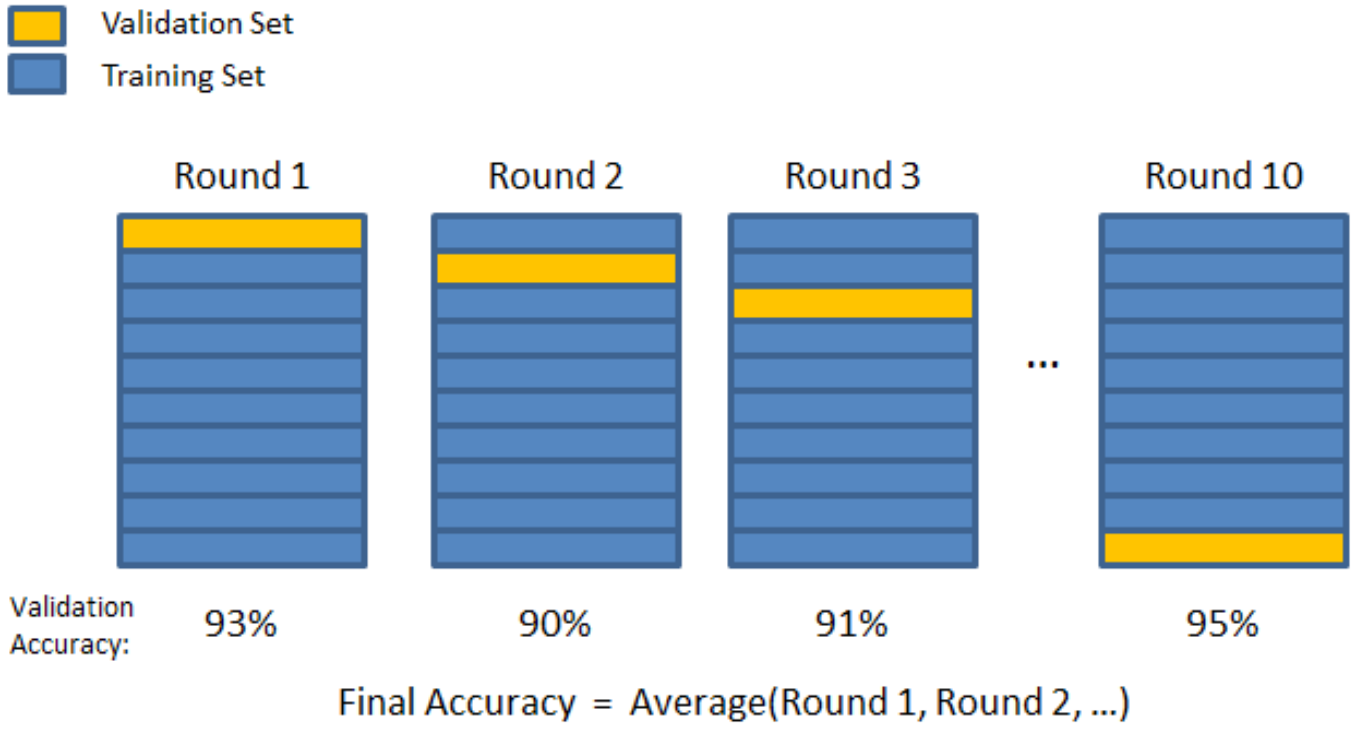
Data set

- **Training set:** for training/learning network parameters
- **Validation set:** tuning hyperparameters: number of hidden layers/neurons
- **Test set:** to evaluate final network performance



Cross-validation

- To avoid losing training data for validation set
- Check scikit-learn!



Data

- DataFetcher collects real-time data
- Data from:
  - Belgian day-ahead electricity market: https://www.belpex.be/
  - Belgian TSO: http://www.elia.be/en/grid-data/dashboard
    - Wind production
    - Solar production
    - Total load
- Extra challenges related to using real data!

Competition

- Create a neural network to forecast day-ahead electricity prices
  - Try to have a good prediction:
    - Gain insight in data and preproces inputs
    - Use additional (relevant) features (use DataFetcher as an example)
    - Tune network architecture
- Submit your results
- See which team did best!
  - Evaluation is based on the Mean Squared Error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Final report

- Create a **predictions.csv** file with 24 rows, each row should contain the forecast for that hour. Predict the belpex prices for **27 April 2018**.
- Write a report of max. 5 pages (including plots) with the results of your work
  - The data you used and the preprocessing you performed.
  - The architecture of the neural network and the reason why you chose this architecture (explain the experiments you performed to make your decision).
  - The final results and performance of the model, e.g. what is the test-set performance.
- Deadline: **26 April 2018, at 2 pm**