MATLAB
Neural Network Toolbox
uczenie sieci
(wygodnie)
Grupy zastosowań

1. *Fitting a function* - Dopasowywanie funkcji
2. *Pattern recognition* - Rozpoznawanie wzorców
3. *Clustering data* - Klasteryzacja danych

- dogłębie
- wygodnie

- *Using command-line functions*
- *Using graphical user interfaces*
FITTING A FUNCTION

COMMAND LINE FUNCTIONS

- example: housing
przykład 1. *Fitting a function*

- Neural networks are good at fitting functions and recognizing patterns. In fact, there is proof that a fairly simple neural network can fit any practical function.

- Suppose, for instance, that you have data from a housing application [HaRu78]. You want to design a network that can predict the value of a house (in $1000s), given 13 pieces of geographical and real estate information.

- You have a total of 506 example homes for which you have those 13 items of data and their associated market values.

- You can solve this problem in three ways:
  - Use a command-line function
  - Use a graphical user interface for function fitting `nftool`
  - Use a graphical user interface `nntool`
przykład 1. *Fitting a function*

- **Using Command-Line Functions**

  1. **Load the data**
     consisting of input vectors and target vectors, as follows:

     ```
     load house_dataset
     ```

  2. **Create a network**

     For this example, you use a feed-forward network with the default tan-sigmoid
     transfer function in the hidden layer and linear transfer function in the output
     layer. This structure is useful for function approximation (or regression) problems. Use 20 neurons (somewhat arbitrary) in one hidden layer. The network has one output neuron, because there is only one target value associated with each input vector.

     ```
     net = newfit ( houseInputs , houseTargets , 20 ) ;
     ```
przykład 1. *Fitting a function*

- **Using Command-Line Functions**

3. **Train the network**

   The network uses the default [Levenberg-Marquardt algorithm](https://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm) for training. The application randomly divides input vectors and target vectors into three sets as follows:
   - 60% are used for training
   - 20% are used to validate that the network is generalizing and to stop training before overfitting
   - the last 20% are used as a completely independent test of network generalization

To train the network, enter:

```matlab
net = train ( net , houseInputs , houseTargets ) ;
```
przykład 1. *Fitting a function*

- **Using Command-Line Functions**

3. **Train the network**

   During training, the following training window opens. This window displays training progress and allows you to interrupt training at any point by clicking **Stop Training**.
3. Train the network

This training stopped when the validation error increased for six iterations, which occurred at iteration 23. If you click Performance in the training window, a plot of the training errors, validation errors, and test errors appears, as shown in the following figure.
przykład 1. *Fitting a function*

- Using Command-Line Functions

3. Train the network

In this example, the result is reasonable because of the following considerations:

- The final mean-square error is small
- The test set error and the validation set error have similar characteristics
- No significant overfitting has occurred by iteration 17 (where the best validation performance occurs)
przykład 1. **Fitting a function**

- **Using Command-Line Functions**

4. Perform some analysis of the network response

If you click **Regression** in the training window, you can perform a linear regression between the network outputs and the corresponding targets. The following figure shows the results.
przykład 1. *Fitting a function*

- **Using Command-Line Functions**

  4. **Perform some analysis of the network response**

    In this case, the network response is satisfactory, and you can now use `sim` to put the network to use on new inputs.

    To get more experience in command-line operations, try some of these tasks:

    - during training, open a plot window (such as the regression plot), and watch it animate.
    - plot from the command line with functions such as `plotfit`, `plotregression`, `plottrainstate` and `plotperform`.
przykład 1. *Fitting a function*

- Using Command-Line Functions - podsumowanie:

```matlab
load house_dataset

net = newfit ( houseInputs, houseTargets, 20 )

net = train ( net, houseInputs, houseTargets )

[ net, Y, E, Pf, Af, tr ] = adapt ( net, houseInputs, houseTargets )

plotperform
plottrainstate
plotfit
plotregression
```
FITTING A FUNCTION
GUI - NEURAL FITTING TOOL

- example: simple fitting
przykład 1. *Fitting a function*

- **Using the Neural Network Fitting Tool GUI**

  1. Open the **Neural Network Fitting Tool** with this command:

     `nftool`
przykład 1. *Fitting a function*

- **Using the Neural Network Fitting Tool GUI**

3. **Click Load Example Data Set in the Select Data window.**
   The Fitting Data Set Chooser window opens

   ![Fitting Data Set Chooser](image)

   - **Select a data set:**
     - Simple Fitting Problem
     - House Pricing
     - Abalone Rings
     - Beluga
     - Body Fat
     - Building Energy
     - Chemical
     - Computer Activity
     - Engine
     - House Price

   **Description**
   - **Filename:** simplefit_dataset
   - Estimate the outputs of an arbitrary one-input/one-output function.
   - An estimator can be found by fitting the inputs and targets.
   - The data set consists of 67 samples.
   - "simplefitInputs" is a 1x67 matrix.
   - "simplefitTargets" is a 1x67 matrix.

   **This simple example can be solved with the Neural Network Fitting Tool (nftool).**

4. **Select Simple Fitting Problem, and click Import.**
   This brings you back to the **Select Data window**
przykład 1. *Fitting a function*

- **Using the Neural Network Fitting Tool GUI**

5. Click **Next** to display the **Validate and Test Data** window, shown in the following figure. The validation and test data sets are each set to 15% of the original data.
przykład 1. *Fitting a function*

- Using the Neural Network Fitting Tool GUI
  
  6. Click **Next** and choose the number of hidden neurons
przykład 1. *Fitting a function*

- Using the Neural Network Fitting Tool GUI

7. Click **Next** to see the **Train network** window
przykład 1. *Fitting a function*

- **Using the Neural Network Fitting Tool GUI**

  8. **Click Train**

This time the training continued for the maximum of 1000 iterations.
przykład 1. Fitting a function

- Using the Neural Network Fitting Tool GUI

przykład 1. *Fitting a function*

- Using the Neural Network Fitting Tool GUI

10. Under Plots, click **Regression**.

For this simple fitting problem, the fit is almost perfect for training, testing, validation data.
przykład 1. *Fitting a function*

- Using the Neural Network Fitting Tool GUI

11. View the network response - Under *Plots*, click *Fit*
przykład 1. *Fitting a function*

- Using the Neural Network Fitting Tool GUI

12. Click **Next** in the Neural Network Fitting Tool to evaluate the network.
przykład 1. *Fitting a function*

- Using the Neural Network Fitting Tool GUI
RECOGNIZING PATTERNS
COMMAND LINE FUNCTIONS

- example: cancer
przykład 2. Recognizing patterns

- In addition to function fitting, neural networks are also good at recognizing patterns.

- For example, suppose you want to classify a tumor as benign or malignant, based on uniformity of cell size, clump thickness, mitosis, etc. [MuAh94]. You have 699 example cases for which you have 9 items of data and the correct classification as benign or malignant.

- As with function fitting, there are three ways to solve this problem:
  - Use a command-line solution,
  - Use the nprtool GUI,
  - Use nntool.
przykład 2. Recognizing patterns

• Using Command-Line Functions

1. Use the cancer data set as an example

   This data set consists of 699 nine-element input vectors and two-element target vectors.

   Load the tumor classification data as follows:

   ```
   load cancer_dataset
   ```
przykład 2. Recognizing patterns

- Using Command-Line Functions

2. Create a network

For this example, you use a pattern recognition network, which is a feed-forward network with tan-sigmoid transfer functions in both the hidden layer and the output layer.

As in the function-fitting example, use 20 neurons in one hidden layer:
- The network has two output neurons, because there are two categories associated with each input vector
- Each output neuron represents a category
- When an input vector of the appropriate category is applied to the network, the corresponding neuron should produce a 1, and the other neurons should output a 0.

To create a network, enter this command:
```matlab
net = newpr ( cancerInputs , cancerTargets , 20 ) ;
```
przykład 2. Recognizing patterns

• Using Command-Line Functions

3. Train the network

The pattern recognition network uses the default
Scaled Conjugate Gradient algorithm for training.

The application randomly divides the input vectors and target vectors
into three sets:
- 60% are used for training
- 20% are used to validate that the network is generalizing
  and to stop training before overfitting
- The last 20% are used as a completely independent test
  of network generalization

To train the network, enter this command:

```matlab
net = train( net, cancerInputs, cancerTargets );
```
przykład 2. **Recognizing patterns**

- **Using Command-Line Functions**

3. **Train the network**
przykład 2. Recognizing patterns

- Using Command-Line Functions

4. Analyze the network performance

![Performance Graph](image)
przykład 2. **Recognizing patterns**

- **Using Command-Line Functions**

4. **Analyze the network performance**
przykład 2. Recognizing patterns

- Using Command-Line Functions - podsumowanie:

```matlab
load cancer_dataset

net = newpr( cancerInputs , cancerTargets , 20 );

net = train( net , cancerInputs , cancerTargets );

plotperform
plottrainstate
plotconfusion
plotroc
```
RECOGNIZING PATTERNS

GUI - NEURAL PATTERN RECOGNITION TOOL

- example: simple classes
przykład 2. Recognizing patterns

- Using the Neural Network Pattern Recognition Tool GUI

1. Open the Neural Network Pattern Recognition Tool window with this command:

nprtool
przykład 2. Recognizing patterns

- Using the Neural Network Pattern Recognition Tool GUI

3. Pattern Recognition Data Set Chooser

Select a data set:
- Simple Classes
  - Iris Flowers
  - Breast Cancer
  - Types of Glass
  - Thyroid
  - Wine Vintage

Description
- Filename: simpleclass_dataset
- Two-element vectors are assigned to four classes.
- Samples may be classified using clustering (using only input data) or with Pattern Recognition or Fitting (with input and target data).
- The data set consists of 1000 samples.
- "simpleclassInputs" is a 2x1000 matrix of values.
- "simpleclassTargets" is an 4x1000 matrix, where each ith column indicates which category the ith iris belongs to with a 1 in one element (and zeros in the other elements).

This simple example can be solved with the Neural Network Pattern Recognition Tool (nprttool) or Clustering Tool (nctool).
CLUSTERING DATA

COMMAND LINE FUNCTIONS

- example: iris flowers
przykład 3. Clustering data

- Clustering data is another excellent application for neural networks.

- This process involves grouping data by similarity.

  For example, you might perform:
  - Market segmentation by grouping people according to their buying patterns
  - Data mining by partitioning data into related subsets
  - Bioinformatic analysis by grouping genes with related expression patterns

- Suppose that you want to cluster flower types according to petal length, petal width, sepal length, and sepal width [MuAh94].
  You have 150 example cases for which you have these four measurements.
przykład 3. Clustering data

- Using Command-Line Functions

2. Create a network

For this example, you use a self-organizing map (SOM).

This network has one layer, with the neurons organized in a grid.

When creating the network, you specify the number of rows and columns in the grid:

```
net = newsom ( irisInputs , [ 6 , 6 ] ) ;
```
przykład 3. *Clustering data*

- Using Command-Line Functions

3. Train the network.

The SOM network uses the default batch SOM algorithm for training.

```
net = train ( net , irisInputs ) ;
```
przykład 3. **Clustering data**

- Using Command-Line Functions
przykład 3. *Clustering data*

- **Using Command-Line Functions**

5. For SOM training, the weight vector associated with each neuron moves to become the center of a cluster of input vectors.

In addition, neurons that are adjacent to each other in the topology should also move close to each other in the input space.

The default topology is hexagonal; to view it, click *SOM Topology*
przykład 3. *Clustering data*

- **Using Command-Line Functions**

6. To view the U-matrix, click **SOM Neighbor Distances**
przykład 3. *Clustering data*

- **Using Command-Line Functions**

  To get more experience in command-line operations, try some of these tasks:

  - During training, open a plot window (such as the SOM weight position plot) and watch it animate
  - Plot from the command line with functions such as:
    - `plotsomhits`
    - `plotsomnc`
    - `plotsomnd`
    - `plotsomplanes`
    - `plotsompos`
    - and `plotsomtop`
CLUSTERING DATA

GUI - NEURAL CLUSTERING TOOL

- example: simple clusters
przykład 3. *Clustering data*

- Using the Neural Network Clustering Tool GUI

1. Open the **Neural Network Clustering Tool** window with this command:

```
nctool
```
przykład 3. Clustering data

- Using the Neural Network Clustering Tool GUI

7. Click Train
przykład 3. *Clustering data*

- Using the Neural Network Clustering Tool GUI

8. Investigate some of the visualization tools for the SOM.

   Under the *Plots* pane, click *SOM Topology*.
przykład 3. *Clustering data*

- **Using the Neural Network Clustering Tool GUI**

8. Investigate some of the visualization tools for the SOM.

   Under the **Plots** pane, click **SOM Neighbor Connections**.
przykład 3. **Clustering data**

- Using the Neural Network Clustering Tool GUI

8. Investigate some of the visualization tools for the SOM.

Under the **Plots** pane, click **SOM Neighbor Distances**.
przykład 3. *Clustering data*

- Using the Neural Network Clustering Tool GUI

8. Investigate some of the visualization tools for the SOM. Under the **Plots** pane, click **SOM Weight Planes**.
przykład 3. Clustering data

- Using the Neural Network Clustering Tool GUI

8. Investigate some of the visualization tools for the SOM.

Under the Plots pane, click SOM Weight Positions.
przykład 3. *Clustering data*

- Using the Neural Network Clustering Tool GUI

8. Investigate some of the visualization tools for the SOM.

Under the Plots pane, click SOM Sample Hits.
przykład 3. **Clustering data**

- Using the Neural Network Clustering Tool GUI

8. Investigate some of the visualization tools for the SOM.

Under the **Plots** pane, click **SOM Sample Hits**.