

Neuronale Netze

Eine Neuronales Netz kann im Gegensatz zur Linearen Regression auch nicht-lineare Zusammenhänge abbilden. Dafür ist es jedoch deutlich komplexer und stellt bezüglich des Zustandekommens seiner Ausgabe eine Blackbox dar. Ihre Stärken haben Neuronale Netze bei komplexen Aufgabenstellungen wie beispielsweise der Bildanalyse oder dem Verständnis natürlicher Sprache. Hier soll ein Neuronales Netz zur Prognose von Heizenergieverbräuchen eingesetzt werden.

Aufgabe 1: Daten laden

Zuerst wird die Datei `features.csv` mit den Daten mit Hilfe der Funktion `readtable` geladen. Dann werden die Timestamps und die Features separat gespeichert und mit der Funktion `table2array` zu Arrays konvertiert.

```
clearvars;  
clc;
```

```
% to do: Daten laden und Timestamps und Features trennen und zu Arrays konvertieren
```

Aufgabe 2: Aufteilung Trainings- und Test-Daten

In diesem Schritt werden die Daten in Trainingsdaten zum Erstellen und in Testdaten zum Testen des Neuronalen Netzes unterteilt. Als Trainingsdaten werden die ersten 729 Zeilen verwendet da diese einen Zeitraum von zwei Jahren abdecken. Außerdem werden die Zielwerte (1. Spalte) von den Features getrennt. Es sollen also die Arrays `training_targets`, `training_features`, `test_targets` und `test_features` vorliegen. Zusätzlich sollen als Vorarbeit für folgende Schritte auch die Timestamps in `training_timestamps` und `test_timestamps` aufgeteilt werden.

```
% to do: Trainingsdaten und Testdaten aufteilen
```

Aufgabe 3: Neuronales Netz mit Matlab Toolbox erstellen und trainieren

Die Matlab Toolbox stellt eine vollständige Implementierung einer Vielzahl von Neuronalen Netzen zur Verfügung. Mit der Funktion `feedforwardnet` kann ein Feed Forward Netz mit beliebiger Layer Struktur erzeugt werden. Für das Training kann die Funktion `train` genutzt werden. Außerdem kann durch Zuweisung entsprechender Werte zwischen 0 und 1 der Anteil der Validierungsdaten `net.divide.valRatio` und Trainingsdaten `net.divide.trainRatio` festgelegt werden. Hier soll ein Netz mit 14 Units im ersten und 5 Units im zweiten Hidden Layer verwendet werden. Als Validierungsdaten sollen 10% der Trainingsdaten verwendet werden.

Hinweis:

```
% to do: Neuronales Netz erstellen
```

Aufgabe 4: Netz testen

Zum Testen des soeben trainierten Netzes werden die zu Anfang von den Trainingsdaten getrennten Testdaten verwendet.

% to do: Neuronales Netz testen

Um die Güte der Prognose zu bestimmen werden die durch das Netz prognostizierten Zielwerte mit den tatsächlichen Zielwerten verglichen. Hierfür bieten sich verschiedene Metriken an. An dieser Stelle soll

der RMSE (Rooted Mean Square Error) verwendet werden: $RMSE = \sqrt{\frac{\sum_{t=1}^n (y(t) - \hat{y}(t))^2}{n}}$

Hinweise:

- zur Subtraktion zweier Matrizen kann die Funktion `gsubtract` verwendet werden
- für das Quadrat zweier Matrizen kann der Operator `.^` verwendet werden

% to do: RMSE berechnen

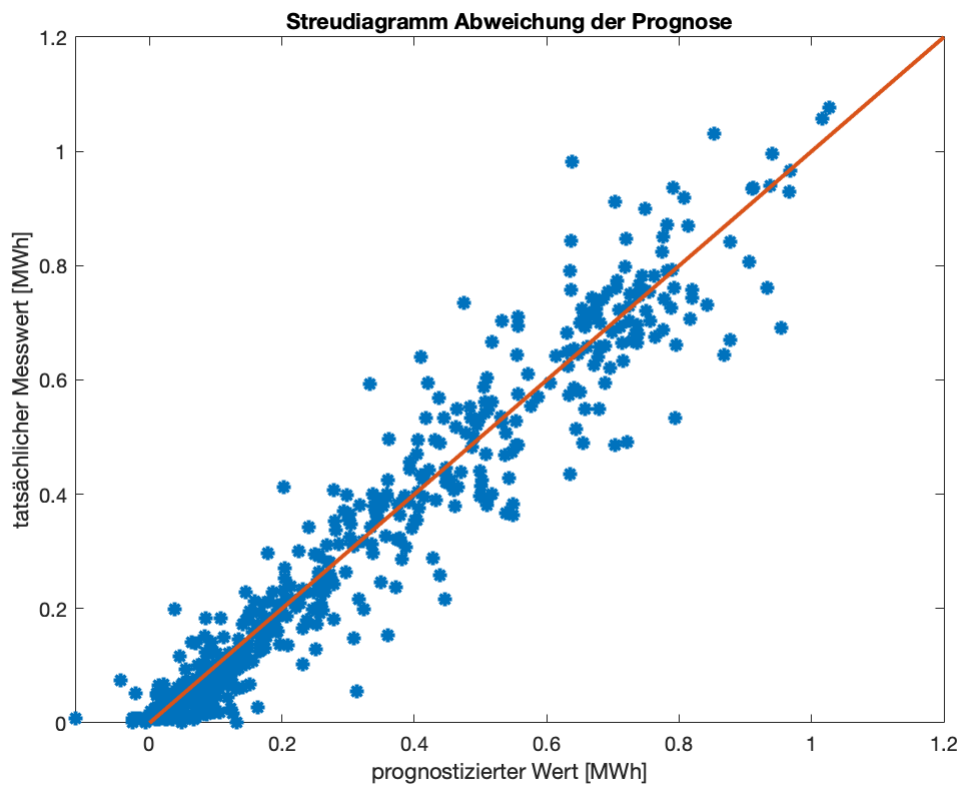
RMSE = 0.0823

Aufgabe 5: Ergebnisse visualisieren

Für einen visuellen Vergleich der Testergebnisse können verschiedene Plots erstellt werden.

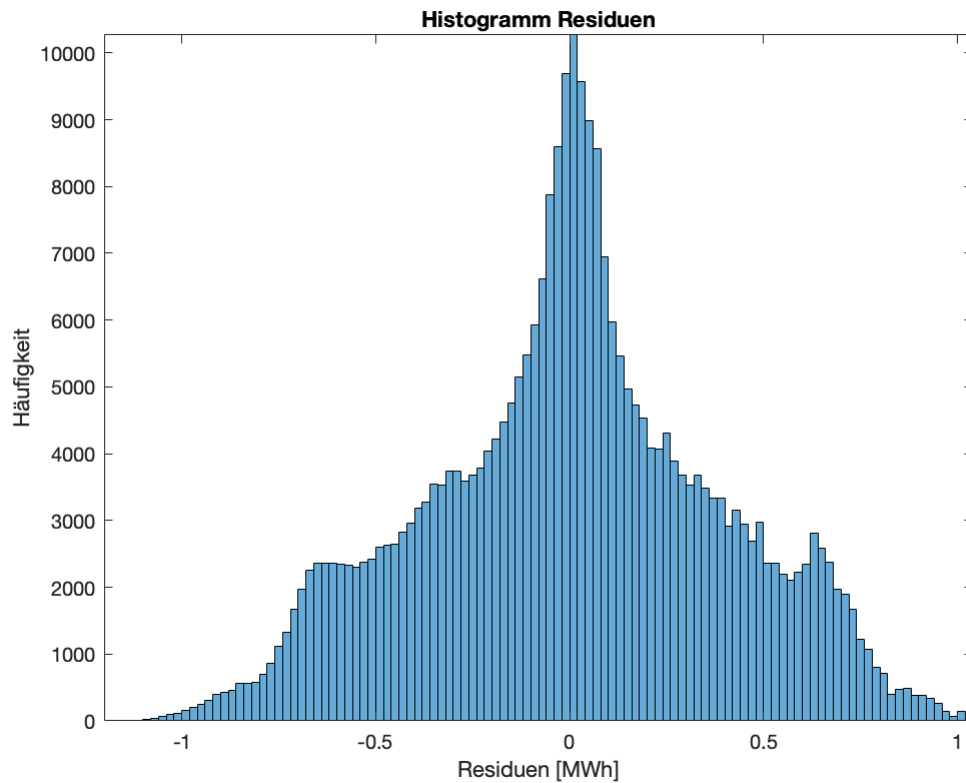
- Ein Streudiagramm mit den prognostizierten Verbräuchen auf der x-Achse und den tatsächlichen Verbräuchen auf der y-Achse:

% to do: Streudiagramm plotten



- Ein Histogramm der Residuen:

% to do: Histogramm plotten



- Ein Liniendiagramm welches den Verlauf der prognostizierten Werte sowie der tatsächlichen Werte über den gesamten Zeitraum zeigt:

Hinweis: um einen schönen Plot zu erhalten müssen die Datenlücken in output und in test_targets zuvor mit NaN gefüllt werden. Dazu müssen diese zuerst zu einer timetable konvertiert werden und diese dann an die Funktion `retime(tt, 'daily')` übergeben werden.

```
% to do: Liniendiagramm plotten
```

