Multiple Lineare Regression

Multiple Lineare Regression ermöglicht die Abbildung eines linearen Einflusses mehrerer Parameter auf einen Zielwert. Damit eignet sie sich hervorragend als Methode für die Erstellung von Energieverbrauchsprognosen. in dieser Aufgabe soll zuerst mit Trainingsdaten ein Regressionsmodell erstellt und anschließend dessen Güte mit Testdaten überpüft 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 Regressionsmodells 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: Regressionsmodell erstellen

Die linearen Einflüsse der Features auf den Zielwert werden durch die folgende lineare Modellfunktion beschrieben:

$$Q(t+1) = x(t) \cdot p = p_0 + p_1 \cdot Q(t) + p_2 \cdot Q(t-1) + \ldots + p_{14} \cdot Q(t-13)$$
$$+ p_{15} \cdot T(t) + p_{16} \cdot T(t-1) + \ldots + p_{28} \cdot T(t-13)$$
$$+ p_{29} \cdot S(t+1)$$
$$+ p_{30} \cdot W(t+1)$$

wobei folgende Notationen gelten:

$$p = [p_0, \ldots, p_30]^T,$$

$x(t) = [1, Q(t), \dots, Q(t-13), T(t), \dots, T(t-13), S(t+1), W(t+1)]$,

$Q(t) =$ Wärmeverbrauch von Tag t,

$T(t) =$ Außentemperatur von Tag t,

$S(t) =$ Saisonkomponente von Tag t,

$W(t) =$ Werktagsindikator von Tag t

Da Prognosemodelle im Allgemeinen nicht lineare Einflüsse besitzen und Daten Messfehler behaftet sind, ist es nicht möglich, p so zu wählen, dass in der Modellfunktion für alle t Gleichheit gilt. Folglich handelt es sich um ein Optimierungsproblem bei dem p so gewählt wird, dass $\sum_{t=1}^{m} (Q(t+1) - x(t) \cdot p)^2$ minimal ist.

Es sei $X = \begin{bmatrix} x(1) \\ \vdots \\ x(m) \end{bmatrix} \in \mathbb{R}^{m \times 30}$ und $Q = \begin{bmatrix} Q(1+1) \\ \vdots \\ Q(m+1) \end{bmatrix} \in \mathbb{R}^{m}$.

Dann kann das optimale p^* durch die Moore-Penrose Pseudoinverse $X^{\#}$, die in Matlab mit dem Befehl `pinv(X)` bestimmt werden kann, wie folgt berechnet werden:

$p^* = X^{\#} \cdot Q$.

Hinweis: Als Faktor für p_0 wird der Features-Matrix eine 1-Spalte hinzugefügt

```
% to do: Regressionsmodell erstellen
```

Aufgabe 4: Regressionsmodell testen

Zum Testen des soeben errechneten Regressionsmodells werden die zu Anfang von den Trainingsdaten getrennten Testdaten verwendet.

```
% to do: Regressionsmodell testen
```

Um die Güte der Prognose zu bestimmen werden die durch die Regression 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
```

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
```

Aufgabe 6: Underfitting

Jetzt sollen Aufgaben 1 bis 5 mit dem Datensatz `features_einzelraeume.csv` wiederholt werden. Als Trainingsdaten sollen die ersten 172 Zeilen verwendet werden.

Was fällt beim Vergleich der Ergebnisse auf und wie lässt sich dieser Effekt erklären?

```
clearvars;
clc;

% to do: Regression mit features_einzelraeume.csv
```