Dokumentation MLP Tool

Dieses Tool bietet eine grafische Schnittstelle zur Python-Bibliothek scikit-learn und kann verwendet werden, um einfache neuronale Netze der Klasse **Multilayer Perceptron** (MLP) für Regressionsprobleme zu trainieren.

## Theoretische Grundlagen

### Struktur

Auszug aus [Wikipedia](https://en.wikipedia.org/wiki/Multilayer_perceptron):

Ein MLP besteht aus mindestens drei [Schichten](https://en.wikipedia.org/wiki/Layer_(deep_learning)) von Knoten: einer Eingabe-[Schicht](https://en.wikipedia.org/wiki/Layer_(deep_learning)), einer versteckten [Schicht](https://en.wikipedia.org/wiki/Layer_(deep_learning)) und einer Ausgabe-[Schicht](https://en.wikipedia.org/wiki/Layer_(deep_learning)). Mit Ausnahme der Eingabeknoten ist jeder Knoten ein Neuron, das eine nichtlineare [Aktivierungsfunktion](https://en.wikipedia.org/wiki/Activation_function) verwendet. […] Es kann Daten unterscheiden, die nicht [linear separierbar](https://en.wikipedia.org/wiki/Linear_separability) sind. […]

Da MLPs voll vernetzt sind, verbindet sich jeder Knoten in einer Schicht über ein bestimmtes Gewicht mit jedem Knoten in der folgenden Schicht.

### Aktivierungsfunktionen

Die Ausgabe jedes Neurons wird durch Abbildung seiner gewichteten Eingänge auf die Aktivierungsfunktion bestimmt. Um nichtlineares Verhalten zu modellieren, sind Aktivierungsfunktionen wie Tangens Hyperbolicus oder die Sigmoidfunktion notwendig.

Typische Aktivierungsfunktionen sind:

#### Linear

Die lineare Funktion bildet die Eingänge direkt auf den Ausgang der Neuronen ab.

#### Tangens Hyperbolicus

Diese logistische Aktivierungsfunktion liegt im Bereich von -1 bis 1.

#### Sigmoid

Diese logistische Aktivierungsfunktion liegt im Bereich von 0 bis 1.

#### ReLu

Die rektifizierte lineare Einheitsfunktion ist gleich der linearen Funktion für Werte und ist gleich 0 für Werte .

Diese Funktion wird häufig in Deep Neural Networks verwendet, um dem Problem der verschwindenden Gradienten entgegenzuwirken, welches verhindert, dass Modelle effektiv lernen.

### Solver

MLPs verwenden einen Algorithmus namens [Backpropagation](https://en.wikipedia.org/wiki/Backpropagation), um den Gradienten der Verlustfunktion in Bezug auf jedes Gewicht durch die Kettenregel zu berechnen. Diese Gradienten sind dann für die Optimierer- (oder Solver-) Algorithmen verfügbar, um die Gewichte nach jeder Trainingsepoche zu aktualisieren. Die verfügbaren Optimierer unterscheiden sich hinsichtlich des Ansatzes, das globale Minimum im Parameter-Hyperspace auf eine effiziente Weise zu finden.

## Funktionsübersicht

### Dateieingabe

Die Trainings- und Testdaten werden als Excel-Dateien (.xlsx) eingelesen. Die Trainingsdaten werden verwendet, um das Modell mit der gegebenen Konfiguration zu trainieren. Das Modell wird dann auf die Testdaten angewendet, um seine Genauigkeit zu berechnen.

Die ersten Spalten enthalten die Eingabedaten, die letzte Spalte ist immer für die Ausgabedaten reserviert. Zeilen, die Zeichenketten enthalten, werden während des Eingabevorgangs automatisch verworfen. Dateien, die weniger als zwei Spalten enthalten, werden verworfen, ebenso wie Dateien mit nicht übereinstimmender Spaltenanzahl.

Nach dem Laden eines Datensatzes wird eine Datenvorschau in der Konsole ausgegeben.

### Skalierer

Wenn der Datensatz noch nicht im Bereich von 0-1 liegt, werden durch Aktivieren dieser Option alle Werte der Trainingsdaten so skaliert, dass sie in den Bereich von 0-1 passen und der Faktor und der Minimalwert werden in der Konsole ausgegeben. Diese Werte können dann in anderer Software verwendet werden, um die Ausgabewerte wieder auf den ursprünglichen Bereich zu skalieren.

### Konfiguration

Die folgenden Parameter können konfiguriert werden: (siehe [scikit-learn documentation](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html) für weitere Informationen)

* **Aktivierungsfunktion:** Die mathematische Funktion, mit der die Ausgabe des Neurons auf der Grundlage seiner Eingabe bestimmt wird. Verfügbare Funktionen sind:
  + ReLu: die rektifizierte lineare Einheitsfunktion,
  + TanH: Tangens Hyperbolicus,
  + Linear:
  + Sigmoid: logistische Sigmoidfunktion,
* **Solver:** Der Lernalgorithmus, der zur Aktualisierung der Verbindungsgewichte nach jeder Epoche verwendet wird. Verfügbare Solver sind:
  + [Adam](https://arxiv.org/abs/1412.6980): Adaptive Moment Estimation. Speichert den exponentiell abfallenden Durchschnitt der vergangenen quadratischen Gradienten und die Gradienten zur adaptiven Berechnung der Lernraten. [🡥](https://ruder.io/optimizing-gradient-descent/)
  + SGD: Klassischer Gradientenabstiegsalgorithmus mit konfigurierbarer Batchgröße.
  + L-BFGS: Implementierung des [Broyden-Fletcher-Goldfarb-Shanno-Algorithmus](https://en.wikipedia.org/wiki/Broyden%E2%80%93Fletcher%E2%80%93Goldfarb%E2%80%93Shanno_algorithm) für begrenzte Speicherkapazität, ein Quasi-Newton-Verfahren, das eine Schätzung der inversen [Hesse-Matrix](https://de.wikipedia.org/wiki/Hesse-Matrix) (Ableitungen zweiter Ordnung) zur Aktualisierung der Gewichte verwendet. Konvergiert sehr schnell für kleinere Datensätze. Dieser Solver besitzt keine Lernkurve.
* **Maximale Anzahl von Epochen:** Das Training wird nach Erreichen dieser Anzahl von Epochen oder nach Erreichen der angegebenen Toleranz beendet.
* **Toleranz:** Wenn sich das Training mindestens 10 Epochen lang nicht um diesen Deltawert verbessert, gilt das Training als beendet.
* **Zufallswert:** Der Seed-Wert für die Generierung von Zufallszahlen, die für die Gewichte und die Bias-Initialisierung sowie für das Batch Sampling verwendet werden. Wenn hier ein ganzzahliger Wert eingestellt wird, führt dies zu reproduzierbaren Ergebnissen.
* **L2-Penalty:** Der Regularisierungsparameter für die Ridge-Regression, der verwendet wird, um große Gewichte und Overfitting zu verhindern, indem der quadrierte Betrag der Koeffizienten als Penalty zur Verlustfunktion hinzugefügt wird. Große L2-Parameterwerte führen zu Underfitting. [🡥](https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c?gi=273b9364d0a7)
* **Momentum:** Fügt dem Gradientenabstieg einen Momentum-Term hinzu, indem ein exponentiell gewichteter Mittelwert addiert wird, der bewirkt, dass die Gewichtsaktualisierungen in Richtung eines globalen Parameter-Minimums beschleunigt werden und verhindert, dass man in lokalen Minima stecken bleibt. 0,9 ist ein typischer Wert.
* **Batch-Größe:** Größe der Minibatches für die stochastischen Solver (SGD und Adam). “auto” wählt das Minimum aus 200 oder der Gesamtzahl der Samples. Eine Größe von 1 entspricht einem stochastischen Gradientenabstieg mit Gewichtsaktualisierungen nach jedem präsentierten Sample. Eine Batch-Größe, die alle Samples umfasst, entspricht dem Batch-Gradientenabstiegs-Algorithmus, bei dem die Gewichte nach der Präsentation aller Samples aktualisiert werden. Die Verwendung von Minibatches (zwischen 1 und allen Stichproben) ist normalerweise der stabilste Ansatz.
* **Lernrate:** Konstante Lernrate für stochastische Solver, die die Schrittgröße beim Aktualisieren der Gewichte steuert.

### Auswertung

Nach Abschluss des Trainings des Modells stehen eine Reihe von Auswertungsmöglichkeiten zur Verfügung:

#### Modell testen

Stellt eine Punktwolke aus den realen und den vorhergesagten Werten im 3D-Raum dar. Diese Option ist nur für zweidimensionale Eingabedaten verfügbar.

#### Lernkurve

Stellt ein 2D-Diagramm dar, das den Verlauf des Lernfehlers über den Trainingszeitraum anzeigt.

#### Fehlerkurve

Stellt ein 2D-Diagramm der Differenz zwischen den Zieldaten und den Vorhersagen dar. Trainings- und Testdaten werden getrennt dargestellt.

#### Statistiken

Berechnet eine Reihe von relevanten statistischen Eigenschaften, getrennt nach Trainings- und Testdaten. Dazu gehören der R2-Score, minimale und maximale Fehlerwerte, absoluter mittlerer quadratischer Fehler, RMS-Fehler, usw.

Der **R2-Score**, auch [Bestimmtheitsmaß](https://de.wikipedia.org/wiki/Bestimmtheitsma%C3%9F), genannt “[…] stellt den Anteil der Varianz (von y) dar, der durch die unabhängigen Variablen im Modell erklärt wurde. […] Es ist also ein Maß dafür, wie gut unbekannte Stichproben durch den Anteil der erklärten Varianz durch das Modell vorhergesagt werden können.” Die Implementierung des R2-Scores in scikit-learn ist wie folgt definiert:

wobei und .

Dabei ist der vorhergesagte Wert des -ten Samples und der zugehörige tatsächliche Wert für eine Gesamtheit von Samples. [🡥](https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score)

Der R2-Score reicht von 0 bis 1, kann aber auch negativ sein, wenn der Score aus Daten berechnet wird, die nicht zum Trainieren des Netzwerks verwendet wurden.

Die gleiche Berechnung des R2-Scores gilt für die Trendlinienfunktion in Microsoft Excel.

#### Gewichte & Bias

Gibt eine Tabelle mit allen Gewichten im Netzwerk und eine Tabelle mit den Bias-Werten für jedes Neuron aus.

Die Gewichts- und Bias-Matrizen können aus dem Dialog heraus in einer Excel-Datei gespeichert werden.

### Speichern

Öffnet einen Speicherdialog, um das trainierte Modell als PMML-Datei für die spätere Verwendung in anderer Software zu exportieren.

## Einschränkungen

Neuronale Netze gibt es in einer großen Vielfalt von Typen und Anwendungsszenarien. Mit diesem Tool kann allerdings nur eine bestimmte Klasse von Modellen trainiert werden, das Multilayer-Perceptron. Dieses wird zwar auch oft für Klassifizierungsprobleme verwendet, allerdings funktioniert das Tool nur für Regressionsprobleme.

Die Eingabedaten können eine beliebige Anzahl von Dimensionen haben, die Ausgabe ist jedoch immer eindimensional, da dies das am häufigsten verwendete Anwendungsszenario ist. Da dieses Tool auf die MLPRegressor-Methode aus der scikit-learn-Bibliothek zurückgreift, ist die Implementierung von mehr als einer Ausgabe derzeit nicht möglich.

Aufgrund der relativ kleinen Datensätze und der internen seriellen Verarbeitung ist GPU-Beschleunigung für dieses Tool (oder scikit-learn im Allgemeinen) nicht verfügbar. Schnelle CPU-Taktraten helfen, die Trainingszeit zu verringern.

Die Modelloptimierung ist aufgrund der verwendeten MLPRegressor-Methode stark eingeschränkt. Es ist kein Pruning oder Dropout verfügbar. Dennoch sollte die Anpassung der Modellparameter in den meisten Fällen zu zufriedenstellenden Ergebnissen führen.

Dieses Tool bietet keine Funktionen zur Datenvorverarbeitung, abgesehen von der Skalierung der Werte auf 0 bis 1. Der Benutzer ist für die Bereinigung des Datensatzes und die Überprüfung auf fehlerhafte Datenpunkte selbst verantwortlich.

## Fehlerbehebung

Die meisten Anwendungsfehler sollten vom Tool selbst abgefangen werden und das Problem über eine Warnmeldung erklären. Dazu gehören:

* Nicht übereinstimmende Spalten- und Zeilenzahl in Trainings- und Testdatensätzen oder nicht genügend Spalten
* Leere oder beschädigte Datensätze.
* Leere Felder in der Parameterkonfiguration
* Parameterkonfigurationen mit dem falschen Datentyp

Wenn die **Modellinitialisierung** fehlschlägt, liegt dies höchstwahrscheinlich an der Parameterkonfiguration. Bitte überprüfen Sie die Parameterkonfiguration auf korrekte Datentypen. Verwenden Sie als Dezimaltrennzeichen einen Punkt, kein Komma.

Wenn die **Modellanpassung** fehlschlägt, wird dies höchstwahrscheinlich durch ein Problem innerhalb des Datensatzes verursacht.