Maschinelles Lernen

Sebastian Koch

Amazon Development Center Edinburgh

16. August 2018

Sommerakademie Leysin, AG4

Was ist Maschinelles Lernen?

Unter Maschinellem Lernen (ML) versteht man (meist) statistische Algorithmen, welche in der Lage sind, Muster in Daten zu finden und, anhand dieser, neue, ungesehene Datenpunkte einzuordnen. Die exakte Art und Weise der Vorhersage wird dabei nicht vorgegeben, sondern vom Algorithmus anhand der vorhandenen Struktur in den Daten gelernt.

Bausteine eines ML-Algorithmus

Jeder ML-Algorithmus besteht aus drei Teilen:

- **Darstellung**: Die Darstellung (*representation*) ist der wichtigste Teil des Algorithmus und wird oft mit diesem verwechselt. Sie beschreibt eine Struktur, in welcher die zu lernenden Daten repräsentiert werden, und welche Hypothesen in Bezug auf die Verteilung der Daten gelernt werden können.
- **Evaluation**: Die Evaluationsfunktion (*evaluation*) wird verwendet um die Qualität der gelernten Darstellung der Daten auszuwerten. Sie misst den "Abstand" zwischen vorhergesagten und gemessenen Werten.
- **Optimierung**: Die Optimierungsmethode (*optimisation*) wird benötigt, um unter den möglichen Darstellungen der Daten eine zu wählen, welche einen möglichst guten Wert der Evaluationsfunktion erzielt.

Darstellung

ML-Algorithmen lassen sich grob in zwei Klassen einteilen:

- **Supervised Learning**: Der Algorithmus lernt aus Datenpunkten bestehend aus Eingaben und Ausgaben. Ziel ist es, zu einem neuen Eingabewert den Ausgabewert vorherzusagen. Möglich sind sowohl kontinuierliche Ausgabewerte (Regression) als auch diskrete Ausgabewerte (Klassifikation).
- Unsupervised Learning: Der Algorithmus lernt lediglich aus Eingaben. Ziel ist es, die Daten zu beschreiben und zu gruppieren und neue Eingabewerte der richtigen Gruppe zuzuordnen.

Beispiele

Wir betrachten supervised learning Problemstellungen, wobei jeder Datenpunkt aus einem Eingabewert und einem Ausgabewert besteht.

- Die einfachste und meistverwendete Darstellung für Regressionsprobleme ist die *lineare Regression*.
- Klassifikationsprobleme werden im einfachsten Fall als *logistische Regression* dargestellt.

Evaluation

Die geeignete Evaluationsfunktion hängt von der Darstellung ab. Einfache Beispiele beinhalten:

- Quadratischer Abweichungsfehler (*Squared error*) bei Regressionsproblemen.
- Prozentualer Anteil korrekter Vorhersagen. (*Accuracy*)
- Prozentualer Anteil positiver Beispiele unter den als positiv identifizierten Datenpunkten (*Precision*) und prozentualer Anteil der als positiv identifizierten positiven Datenpunkte (*Recall*)
- Statistische Verteilungs-Divergenzsmaße (z.B. *KL divergence*)

Optimierung

Unter allen gewählten Darstellung genügenden möglichen Lösungen wird mit Hilfe der Evaluationsfunktion (in diesem Kontext auch oft *loss function*) eine ausgewählt. Idealerweise sollte es sich um diejenige handeln, welche die Evaluationsfunktion minimiert. Probleme:

- Oft existiert keine eindeutige Lösung.
- Das Optimierungsproblem ist oft nicht analytisch exakt lösbar.

Daher werden approximierende Methoden zum Finden einer fast optimalen Lösung verwendet. Die simpelste derartige Methode ist *gradient descent*.

Generalisation

Die größte Stärke von ML-Algorithmen ist ihre (potentielle) Fähighkeit, gelernte Strukturen zu generalisieren, d.h. für ungesehene Datenpunkte brauchbare Schätzungen abzugeben. Dies funktioniert jedoch nur, wenn die ungesehenen Datenpunkte der gleichen Verteilung entstammen wie die Datenpunkte mit welchen der Algorithmus trainiert wurde. Selbst in diesem Fall besteht jedoch noch die Gefahr, zu viel bzw. zu wenig Gewicht auf die exakte Verteilung der Trainingsdaten zu legen.

Train/Test split

In der Praxis werden oft mehrere Algorithmen trainiert, und mit Hilfe einer Evaluationsfunktion der beste ausgewählt. Um sicherzustellen, dass der Algorithmus generalisiert, anstatt lediglich Trainingsdaten zu reproduzieren, werden daher trainierte Algorithmen mit einem Satz Daten trainiert, und mit einem anderen Satz Daten getestet. Dafür werden vor jeglichem Training die vorhandenen Daten in zufälliger Art und Weise gesplittet, und mit den Testdaten evaluiert.

Overfitting

Das Ziel, eine Struktur nur anhand einer Teilmenge an Daten zu lernen, birgt die Gefahr, dass der Algorithmus sich zu sehr an die spezifische Struktur der Trainingsdaten anpasst. Diese Gefahr ist um so größer, je komplexer und damit flexibler die gewählte Darstellung ist. Dieses Phänomen bezeichnet man als *overfitting*. Das gegenteiligen Verhalten ist ebenfalls möglich und wird als *underfitting* bezeichnet. Dies kann insbesondere bei der Wahl einer nicht ausreichend komplexen Darstellung entstehen.

Bias variance tradeoff

Eng verwandt mit diesem Problem ist der Tradeoff zwischen *bias* und *variance*. Komplexe Modelle sind in der Lage, eine große Varianz in den Daten zu repräsentieren, aber laufen Gefahr, *noise*, d.h. intrinsische Ungenauigkeiten in den Daten, mit zu lernen. Einfachere Modelle hingegen haben diese Fähigkeit nicht, und neigen daher dazu, relevante Informationen nicht zu berücksichtigen und die Daten zu verzerren.

Regularisation

Eine verbreitete Methode gegen overfitting ist das Hinzufügen eines Regularisationsterms. Dieser kann zum Beispiel dazu verwendet werden, komplexeren Darstellungen höhere Kosten aufzuerlegen. Dies zwingt den Algorithmus dazu, tendenziell simplere Darstellungen zu bevorzugen. Es entsteht ein Tradeoff zwischen den erhöhten Kosten für komplexe Darstellungen, und höheren Evaluationsfunktionswerten für einfachere Darstellungen.

Motivation

Neuronale Netzwerke sind ML Algorithmen welche in der Lage sind sehr viel komplexere Hypothesen zu lernen als klassische Algorithmen. Ein klassisches Beispiel ist die Erkennung von handgeschriebenen Ziffern. Es handelt sich hierbei um ein Klassifikationsproblem mit 10 Klassen. Die Eingaben sind jedoch hochdimensional - ein Bild in Farbe mit 50 Pixeln Seitenlänge hat bereits eine 7500-dimensionale Eingabe. Klassische Algorithmen scheitern an dieser Aufgabe (Bilderkennung) sowie vielen weiteren komplexen Aufgaben.

Hypothese

Klassisch programmiert man für jede separate Problemstellung ein separates Programm mit einem separaten Algorithmus. Das menschliche Gehirn hingegen ist in der Lage, eine Vielzahl verschiedenster Aufgaben zu meistern. Es bietet sich die Hypothese an, dass der Aufbau des menschlichen Gehirns diesem erlaubt, sich an jegliche Aufgabe anzupassen. Das Gehirn ist somit in der Lage, seine eigenen Algorithmen für spezifische Probleme zu programmieren. Neuronale Netzwerke sind ein Versuch, die Funktionsweise des menschlichen Gehirns nachzubauen, in der Hoffnung eine Darstellung für einen Algorithmus zu erhalten, welcher sich jeglicher Aufgabe anpassen kann.

Logistic units

The kleinste Funktionseinheit des Gehirns is das Neuron. Das Gehirn verarbeitet Informationen durch das Weiterleiten elektrischer Schocks welche von Neuron zu Neuron weiter verarbeitet werden. Jedes Neuron empfängt die Information mehrerer anderer Neuronen und leitet davon abhängig einen elektrischen Schock an viele andere Neuronen weiter. Das Neuron in Neuronalen Netzwerken ist die *logistic unit*.

Informationsweitergabe

In Neuronalen Netzwerken wird Information durch die logistic units verarbeitet. Eine logistic unit besteht aus mehreren Eingabezellen, sowie einer *activation function* welche die eingehende Information verarbeitet. Die eingehenden Informationen werden zunächst gewichtet, mittels der activation function verrechnet, und die entstehende Ausgabe weitergeleitet.

Neuronale Netzwerke

Ein Neuronales Netzwerk ist der Zusammenschluss mehrerer logistic units zu einer komplexen Struktur. Die einfachsten Netzwerke bestehen aus mehreren Lagen (*layers*), innerhalb welcher die Neuronen nicht interagieren. Nebeneinander liegende Lagen sind miteinander verknüpft, und es werden Informationen in eine Richtung weiter geleitet. Die Eingabelage (*input layer*) entspricht der Dimensionalität der Eingabe. Die letzte Lage (*output layer*) muss der Problemstellung angepasst sein. Im Falle der Erkennung von Ziffern hat die output layer beispielsweise Dimensionalität 10. Die Lagen in der Mitte werden als *hidden layers* bezeichnet.

Training eines neuronalen Netzwerkes

Das Ziel besteht darin, dass der Algorithmus selbst die Gewichtungen berechnet, mit welchen Informationen von Neuron zu Neuron weiter gegeben werden. Dies geschieht in mehreren Schritten – in jedem Schritt wird ein weiterer Datenpunkt verwendet um das Netzwerk weiter zu trainieren. Zu Beginn wird das Netzwerk *initialisiert*: Die Gewichtungen werden mit kleinen Zufallszahlen initialisiert.

Training eines neuronalen Netzwerkes

Nun updaten wir die zufällig initialisierten Gewichtungs-Parameter des Netzwerkes mittels einer Optimierungsmethode, beispielsweise gradient descent. Die Methode mittels welcher die Gradienten der Parameter in Neuronalen Netzwerken berechnet wird nennt sich *backpropagation*.

Convolutional Neural Networks (CNNs)

CNNs sind ein Kategorie Neuronaler Netwerke, bei welchen Informationen zwischen layern lediglich in der jeweiligen lokalen Umgebung weitergeleitet werden. Dieser Ansatz beruht auf dem biologischen Vorbild des rezeptiven Feldes bei Sinnesrezeptoren, z.B. der Netzhaut des Auges. Ihre Anwendung liegt vornehmlich in der Verarbeitung von Bild- und Audiodateien.

Convolutional Layer

Ein CNN besteht aus mehreren *convolutional layers*. Die ist ein layer bei welchem der input eines jeden Neurons lediglich von räumlich nahe liegenden Neuronen des input layer – bis zu einer definierten Entfernung – abhängt. Man spricht auch vom rezeptiven Feld des jeweiligen Neurons. Darüber hinaus sind die Kantengewichte standarmäßig für jedes Neuron des convolutional layer identisch, man spricht hierbei von *shared weights*.

Pooling Layer

Darüber hinaus können CNNs zwischen den convolutional layers, sowie vor dem output layer, weitere Lagen enthalten, die sogenannten *pooling layers*. In einem pooling layer fasst jedes Neuron die Informationen des rezeptiven Feldes des input layer so vereinfachend zusammen. Die meistverbreitete Variante sind *max pooling layers*, bei welchen jedes Neuron das maximum seines rezeptiven Feldes übernimmt. Die Motivation in diesem Fall besteht darin, die Aktivierung des Neurons mit der größten zu übernehmen.

Beispiele für CNN-Anwendungen

- Bilderkennung: Handgeschriebene Ziffern, Unterscheidung von 100en verschiedener Klassen von Bildern.
- Spracherkennung: Parsen von gesprochener Sprache, Wortkategorie-Erkennung, Maschinelles Übersetzen.
- Reinforcement Learning: Schätzung zu erwartender Belohnungen, zum Beispiel beim Erlernen von Videospielen, sowie AlphaGo.
- Generieren von Bilder (GANs) und Audiodateien (WaveNet)

Recurent Neural Networks (RNNs)

Recurrent neural networks sind Netzwerke, bei welchen Informationen nicht nur zu nachliegenden Lage weitergegeben werden. RNNs können Verbindungen innerhalb eines layers, sowie entgegen der natürlichen Richtung zu vorliegenden layers enthalten. Es handedlt sich um eine extrem Die Motivation besteht darin, durch Rückkopplungen Informationen über einen längeren Zeitraum speichern zu können, und somit besonders effizient Problemstellungen zu bearbeiten, welche die Verarbeitung von sequenziellem Input erfordern.

Long short-term memory units (LSTM)

LSTM units sind Einheiten, welche beim Aufbau von RNNs zu Verwendung kommen können. Ein RNN welches aus LSTMs besteht wird auch als LSTM Netzwerk bezeichnet. Eine LSTM-Zelle kann ihren Input für eine gewisse Zeit von mehreren 1000en Zeitschritten speichern. Eine LSTM-Zelle enthält ein *input gate*, ein *forget gate* sowie ein *output gate*. Diese kontrollieren die Menge an Information welche in die Zelle einfließt, vergessen wird bzw. zur Weiterverarbeitung hinaus gelassen wird. Sie eignen sich hervorragend für Problemstellungen, bei welchen vor langer Zeit angetroffene Informationen schnell wieder abgerufen werden sollen.

Beispiele für RNN-Anwendungen

- RNNs werden zur Vorhersage von Zeitreihen verwendet.
- Darüber hinaus gibt es erfolgreiche Anwendungen bei der Erkennung von Handschrift, Grammatik, natürlicher Sprache, Musik, Rhythmen etc.
- LSTMS sind eine populäre Methode in der Verarbeitung natürlicher Sprache. LSTMs in Kombination mit CNNs können zum Beispiel Bilder mit Bildunterschriften versehen.

ReLu Layer

Es gibt über die Sigmoid function hinaus weitere Optionen für die Aktivierungsfunktion an den Neuronen. Ein weiteres klassisches Beispiel ist die tanh Funktion. In der Anwendung von Neuronalen Netzwerken hat sich die ReLu Aktivierung durchgesetzt: Eine Kombination von linearer Aktivierung bei positivem Input und keiner Aktivierung bei negativem Input.In der Regel sind die Aktivierungen einer layer stets vom gleichen Typ, weswegen man in der Implementierung in diesem Fall auch von *ReLu layern* spricht.

Embedding layers

Ein *embedding layer* wird verwendet um Daten von unregelmäßiger Dimension in eine regelmäßiges Dimension zu überführen. Das mit Abstand verbreitetste Beispiel ist die Verarbeitung von Text. Möchte man zum Beispiel die Häufigkeit von Wörtern als Eingabe für ein Neuronales Netz verwenden, so besteht das Problem dass die Anzahl möglicher Wörter nicht fest ist. Man löst dieses Problem indem man derartige Daten in Vektoren einer festen Dimension umwandelt. Beispielsweise erhalten die $n-1$ häufigsten Worte eine eigenen Dimension, während alle übrigen sich die letzte Dimension teilen.

Dropout regularisation

Genau wie die simpleren Algorithmen können auch Neuronale Netzwerke zum overfitting neigen. Eine solche Methode ist die *dropout* regularisation. Dabei werden während des Trainings eine feste Anzahl zufällig ausgewählter Neuronen abgeschaltet, d.h. sie werden inklusiver ihrer Verbindungen zu entfernen. In jedem Trainingsschritt wird die zu löschende Anzahl Neuronen neu gewählt, und die endgültigen Gewichtungen durch Mittelwertbildung berechnet.