Der Perceptron-Algorithmus in 20 Minuten

Roland Richter, November 2013

Die Aufgabe

Eine mögliche Anwendung von maschinellem Lernen ist ein autonomes Auto, das Bilder von Verkehrszeichen aufnimmt und diese erkennen muss. Die Datensätze (Bilder) fallen in einige wenige, eindeutig bestimmte *Klassen* (Typ des Verkehrszeichens), und ein konkreter Datensatz soll *klassifiziert* werden.

Dies ist ein Beispiel für "überwachtes Lernen" (*supervised learning*), das vereinfacht in drei wesentlichen Phasen abläuft:

1. Zunächst müssen Datensätze (*samples*) gesammelt und mit der dazu gehörigen Klasse (*label*) "beschriftet" werden.
2. Diese Datensätze werden üblicher Weise in zwei Teile aufgeteilt:
   a) Mit der Trainingsmenge wird der Lernalgorithmus *trainiert*; d.h. Parameter gesucht, mit denen die Datensätze gut voneinander unterschieden werden können.
   b) Mit der Testmenge wird danach *evaluiert*; also überprüft, wie gut der Algorithmus nicht antrainierte Datensätze klassifiziert.
3. Der fertig trainierte Lernalgorithmus kann nun auf neue Datensätze angewandt werden.

Abbildung 1: Eine mögliche Klassifikationsaufgabe ist das Erkennen von Verkehrszeichen. Diese wurde im Rahmen des German Traffic Sign Recognition Benchmark erfolgreich gelöst. Siehe Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32, 323–332.

Um diese Aufgabe mathematisch zu formulieren, wird für jeden Datensatz eine fixe Anzahl d an Kennzahlen (*features*) berechnet. Damit ist jeder Datensatz darstellbar als Punkt $x \in \mathbf{R}^d$. Die einfachste Klassifikationsaufgabe ist, zwischen zwei Klassen zu unterscheiden:

Gegeben: zwei Mengen N ("negativ") und P ("positiv") in $\mathbf{R}^d$
Gesucht: eine Hyperebene in $\mathbf{R}^d$, die N und P trennt

Da eine Hyperebene mit Hilfe ihres Normalvektors w und des Abstands vom Ursprung b als $w^T x + b = 0$ geschrieben werden kann, lässt sich die Aufgabe so umformulieren:

Gesucht: $w \in \mathbf{R}^d$ und $b \in \mathbf{R}$ so, dass
$\forall n \in N : w^T n + b < 0$ und $\forall p \in P : w^T p + b > 0$

Der trainierte Algorithmus klassifiziert ein neues $x \in \mathbf{R}^d$ daher als $x \in P$, wenn $w^T x + b > 0$ ist, und als $x \in N$, wenn $w^T x + b < 0$ ist.

Der Perceptron-Algorithmus

Der Perceptron-Algorithmus (Rosenblatt, 1958) war der erste Algorithmus, der obige Aufgabe lösen konnte. Grundidee ist, dass für jedes noch falsch klassifizierte $n \in N$ oder $p \in P$ der Vektor w in Richtung $-n$ bzw. p gedreht und b angepasst werden muss.

Anfangs werden $w \in \mathbf{R}^d$ und $b \in \mathbf{R}$ beliebig initialisiert. Danach wird folgender Anpassungsschritt für fixes $\epsilon > 0$ so lange durchgeführt, so lange noch falsch klassifizierte Punkte existieren:

- Falls $n \in N$ und $w^T n + b < 0$ oder $p \in P$ und $w^T p + b > 0$:

  bereits richtig klassifiziert, weiter zu nächstem Schritt.

- Falls $n \in N$, aber $\delta := (w^T n + b) \geq 0$:
  $w_{neu} \leftarrow w - \lambda n, b_{neu} \leftarrow b - \lambda$ mit $\lambda := \frac{\delta+\epsilon}{||n||^2+1}$

- Falls $p \in P$, aber $w^T p + b \leq 0$, also $\delta := -(w^T p + b) \geq 0$:
  $w_{neu} \leftarrow w + \lambda p, b_{neu} \leftarrow b + \lambda$ mit $\lambda := \frac{\delta+\epsilon}{||p||^2+1}$

  Nach diesem Schritt ist nämlich

$$\begin{aligned} w_{neu}^T n + b_{neu} &= (w - \lambda n)^T n + b - \lambda \\ &= w^T n - \frac{\delta+\epsilon}{||n||^2+1}||n||^2 + b - \frac{\delta+\epsilon}{||n||^2+1} \\ &= w^T n + b - (||n||^2+1)\frac{\delta+\epsilon}{||n||^2+1} \\ &= \delta - (\delta+\epsilon) = -\epsilon \end{aligned}$$

bzw. analog $w_{neu}^T p + b_{neu} = \epsilon$. Das ursprünglich falsch bewertete n bzw. p wird nach dem Anpassungsschritt also richtig bewertet.

Zwar kann es sein, dass andere, vorher bereits richtig bewertete Punkte jetzt falsch bewertet werden. Es lässt sich aber zeigen, dass der Perceptron-Algorithmus zu einer Lösung konvergiert, falls eine existiert (Rojas, 1996). Der Algorithmus terminiert *nicht*, falls keine Lösung existiert.

Der Perceptron-Algorithmus benötigt im Durchschnitt polynomial viele, im schlimmsten Fall aber exponentiell viele Schritte (Rojas, 1996). So wie andere lineare Methoden lässt er sich durch den "kernel trick" auf nichtlineare Probleme verallgemeinern. Schließlich findet der Algorithmus nur *eine* von vielen Lösungen, aber nicht die "beste". Verfahren, die eine optimale Grenze zwischen N und P finden, wurden erst später erforscht und führten zu den *Support Vector Machines*, siehe z.B. (Bennett & Campbell, 2000).

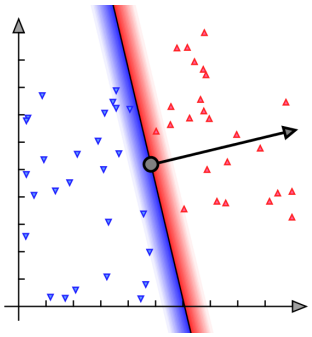

Abbildung 2: Zwei Mengen und eine mögliche Lösung. Die Linie $w^T x + b = 0$ teilt die zweidimensionale Ebene in zwei Halbebenen, in denen $w^T x + b < 0$ (links) bzw. $w^T x + b > 0$ (rechts) gilt.

Literatur

Bennett, K. P., & Campbell, C. (2000). Support vector machines: hype or hallelujah? *SIGKDD Explor. Newsl.*, 2, 1–13.

Rojas, R. (1996). *Neural Networks: A Systematic Introduction*. Berlin: Springer.

Rosenblatt, F. (1958). The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, *65*, 386–408.

Hinweise

Eine interaktive Demonstration des Algorithmus ist auf `http://www.openprocessing.org/sketch/135826` verfügbar.