Statistik Zusatztutorium

Michael Staniek

Universität Heidelberg

January 22, 2020

Einleitung

- Um maschinelles Lernen zu betreiben, braucht man Vektoren.
- Man kann Texte in Vektoren verwandeln, indem man z.B. Wörter zählt
- $\phi(\text{Alle meine Entchen}) = [0, 1, 0, 0, 1, 0, 0, 1]$
- Oder man hat bereits Features, die Zahlen repräsentieren, wie im Iris Datenset

Blockvektoren Teil 1

- Normalerweise kann man mit einem Gewichtsvektor nur folgendes modellieren: gehört entweder zur Klasse (positives Label) oder nicht.
- z.B. Gewichtsvektor: $[0.3, -0.1]$
- Der Vektor $[-1, 1]$ gehört eindeutig zur negativen Klasse, weil der Score (das Skalarprodukt (Dot Product)) -0.4 ist.
- Der Vektor $[1, -1]$ gehört eindeutig zur positiven Klasse, weil der Score 0.4 ist.
- Wie kann man aber Multiklassenprobleme lösen?

Blockvektoren Teil 2

- Mit mehreren Gewichtsvektoren!
- w1=$[10, 10]$
- w2=$[1, 1]$
- Ein Vektor $[1, 1]$ würde durch w1 eindeutig einen höheren Score bekommen
- Geht das auch mit einem Gewichtsvektor?

Blockvektoren Final

- Ein Gewichtsvektor, der Klassenanzahl*Datendimensionsanzahl groß ist.
- Der zu prüfende Datenpunkt (egal ob Trainingspunkt oder Testpunkt) wird je nach Klasse in einen unterschiedlichen Vektorraum projeziert, um den Score zu ermitteln.
- 3 Klassen: Y=0, 1, 2
- Datenpunkt: $[1, 1]$
- $\phi([1, 1], 0) = [1, 1, 0, 0, 0, 0]$
- $\phi([1, 1], 1) = [0, 0, 1, 1, 0, 0]$
- $\phi([1, 1], 2) = [0, 0, 0, 0, 1, 1]$
- Das Skalarprodukt dieser "aufgepumpten" Vektoren mit dem Gewichtsvektor ist dasselbe, als würde man den unaufgepumpten Trainingspunkt mit 3 verschiedenen Gewichtsvektoren skalarprodukten.

Beispiel

- $\phi([1,1],0) = [1,1,0,0,0,0]$
- $\phi([1,1],1) = [0,0,1,1,0,0]$
- $\phi([1,1],2) = [0,0,0,0,1,1]$
- Gewichtsvektor: $w = [1,2,3,4,5,6]$
- $\phi([1,1],0) * w = 1*1 + 1*2 = 3$
- Äquivalent: $< [1,1],[1,2] >$
- $\phi([1,1],1) * w = 1*3 + 1*4 = 7$
- Äquivalent: $< [1,1],[3,4] >$
- $\phi([1,1],2) * w = 1*5 + 1*6 = 11$
- Äquivalent: $< [1,1],[5,6] >$

Grundlagen des maschinellen Lernens

- Wir brauchen eine Funktion, die uns sagt wie falsch wir liegen, auch bekannt als die Lossfunktion.
- Die Ableitung des Lossfunktion nach unseren Modellparametern zeigt uns, wie wir uns verbessern können (Gradient Descent)
- Gehen wir jetzt alle Losses durch, die wir gelernt haben (andere Reihenfolge)

Perzeptron

Lossfunktion (Perzeptron-Loss):
$max(0, \underset{y \neq y_t}{max} w * \phi(x_t, y) - w * \phi(x_t, y_t))$

Das umschliessende max kann als if-else umgeschrieben werden:
$$\begin{cases} 0 \text{ if } w * \phi(xt, y) - w * \phi(xt, yt) < 0 \\ w * \phi(xt, y) - w * \phi(xt, yt) \text{ else} \end{cases}$$

Die Ableitung nach w ist einfach:
$$\begin{cases} 0 \text{ if } w * \phi(xt, y) - w * \phi(xt, yt) < 0 \\ \phi(xt, y) - \phi(xt, yt) \text{ else, also } w * \phi(xt, y) - w * \phi(xt, yt) >= 0 \end{cases}$$

Perzeptron (Der binäre Fall)

- Multiclass-Loss: $max(0, \max_{y \neq y_t} w * \phi(x_t, y) - w * \phi(x_t, y_t))$
- Mithilfe der Formel $\phi(x, y) = \frac{1}{2} y * \phi(x)$ kann man diesen Loss in einen binären Loss verwandeln (Aufgabenblatt 5 Problem 1)
- Binärer Loss: $max(0, -y * w * \phi(x))$
- Binäres Update:

$$\begin{cases} 0 \text{ if } -y * w * \phi(x) < 0 \\ -y * \phi(x) \text{ else} \end{cases}$$

SVM

Lossfunktion (Hinge-Loss):
$max(0, 1 + \underset{y \neq y_t}{max} w * \phi(x_t, y) - w * \phi(x_t, y_t))$

Ableitung ähnlich wie Perzeptron.

Aber: Regularisierungsterm nicht vergessen, wenn man mit SVM arbeitet! Allerdings heisst der obrige Loss direkt Hinge-Loss.
Also: SVM $\rightarrow$ Hinge-Loss Perzeptron und Regularisierung

Der binäre Fall

- Multiclass-Hinge-Loss:
  $max(0, \underset{y \neq y_t}{max} 1 + w * \phi(x_t, y) - w * \phi(x_t, y_t))$
- Mithilfe der Formel $\phi(x, y) = \frac{1}{2} y * \phi(x)$ kann man diesen Loss in einen binären Loss verwandeln (ähnlich Aufgabenblatt 5 Problem 1)
- Binärer Hinge-Loss: $max(0, 1 - y * w * \phi(x))$
- Binäres Update: $\begin{cases} 0 \text{ if } -y * w * \phi(x) < -1 \\ -y * \phi(x) \text{ else} \end{cases}$

Logistic Regression

Lossfunktion (Log-Loss): $-\sum log(p(y|x))$
Lossfunktion (Log-Loss): $-\sum log(\frac{e^{w\cdot\phi(x_i,y_i)}}{\sum_{y\in Y} e^{w\cdot\phi(x_i,y)}})$

Ableitung wie in den Folien zur Logistic Regression (53 - 59) und dem im Wiki stehenden Anhang zur Logistic Regression für Tutorium 4. Gradient:

$\phi(x_t, y_t) - \sum_y p(y|xt) * \phi(xt, y)$

Der binäre Fall

- Multiclass-Log-Loss $-\sum log(\frac{e^{w \cdot \phi(x_i, y_i)}}{\sum_{y \in Y} e^{w \cdot \phi(x_i, y)}})$
- Mithilfe der Formel $\phi(x, y) = \frac{1}{2} y * \phi(x)$ kann man diesen Loss in einen binären Loss verwandeln (Aufgabenblatt 4 Problem 1 und 2)
- Binärer Log-Loss: $log(1 + e^{-y * w * \phi(x)})$

Kernelmethoden

- Beim 10ten Fehler haben wir 10 mal folgenden Term addiert:
- $\phi(x_t, y_t) - \phi(x_t, y)$
- Unser Gewichtsvektor besteht also NUR aus diesen Differenzvektoren!
- also:
- $w = \sum_{t,y} \alpha_{t,y}[\phi(x_t, y_t) - \phi(x_t, y)]$
- wobei $\alpha_{t,y}$ eine Zählvariable ist die zählt wie oft wir für Trainingsbeispiel t das label y falsch vorhergesagt haben.

Herleitung

$$
\begin{aligned}
y* &= argmax_{y*} w^{(i)} * \phi(x, y*) \\
y* &= argmax_{y*} \sum_{t,y} \alpha_{t,y} [\phi(x_t, y_t) - \phi(x_t, y)] * \phi(x, y*) \\
y* &= argmax_{y*} \sum_{t,y} \alpha_{t,y} [\phi(x_t, y_t) * \phi(x, y*) - \phi(x_t, y) * \phi(x, y*)] \\
y* &= \\
&argmax_{y*} \sum_{t,y} \alpha_{t,y} [K(\phi(x_t, y_t), \phi(x, y*)) - K(\phi(x_t, y), \phi(x, y*))]
\end{aligned}
$$

Kernel

- $K(\phi(x_t, y_t), \phi(x, y*)) = \phi(x_t, y_t) * \phi(x, y*)$
- $K(\phi(x_t, y_t), \phi(x, y*)) = (\phi(x_t, y_t) * \phi(x, y*))^2$
- Kerneltrick! Sollen wir das an der Tafel machen?

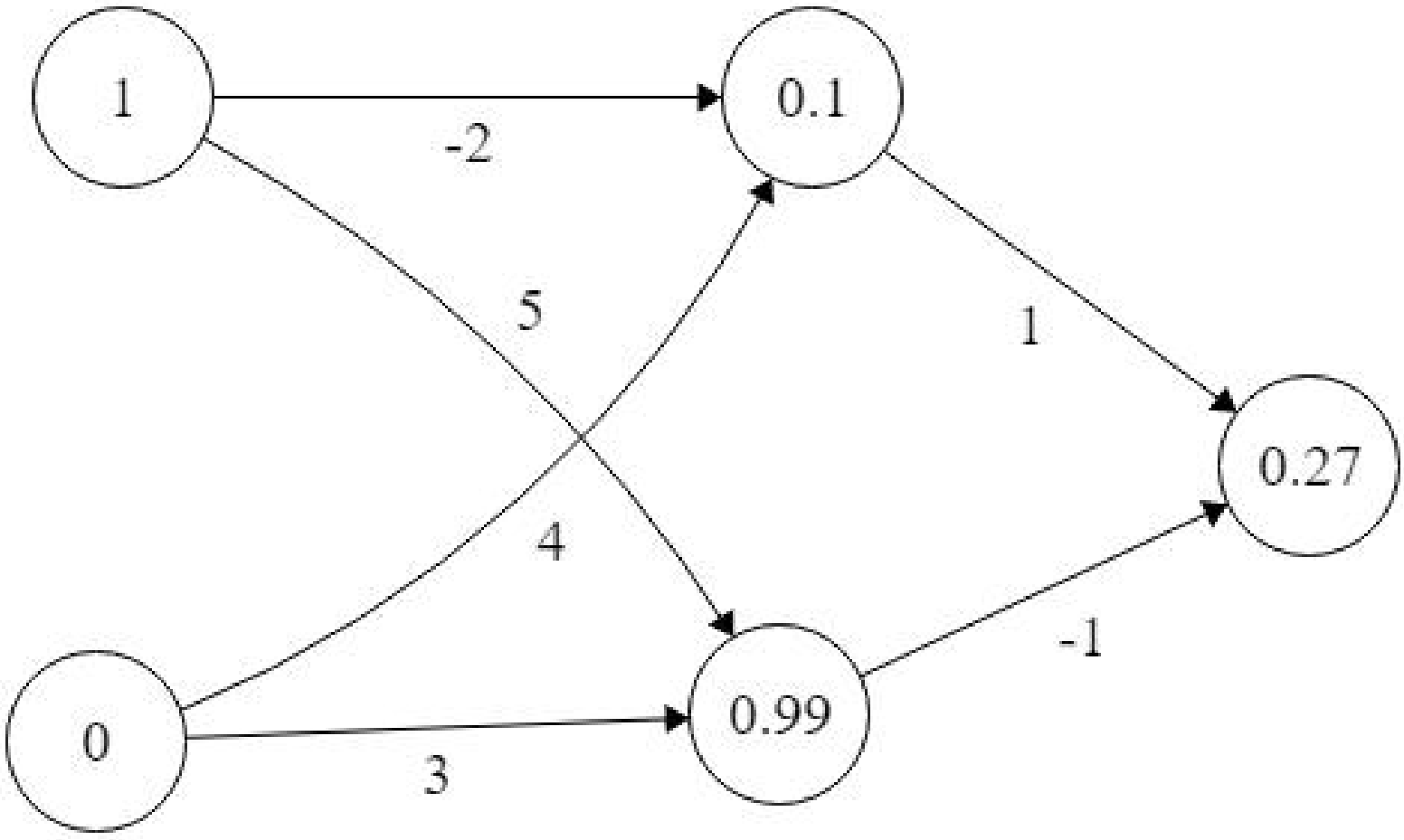
Neuronale Netze

- Können auch das XOR-Problem lösen.
- Immer 2 Layer zusammen könnten als 1 Multiclass Perzeptron interpretiert werden
- $=>$ Multilayerperzeptron

Neuronale Netze

- Können auch das XOR-Problem lösen.
- Immer 2 Layer zusammen könnten als 1 Multiclass Perzeptron interpretiert werden
- $=>$ Multilayerperzeptron

Neuronale Netze



Fragen

Habt ihr sonst irgendwelche Fragen? IRGENDWELCHE?
Irgendwas was wir genauer durchgehen sollen, irgendwas was ich erklären soll/nochmal erklären soll?

Zu lernen

- Alle Aufgabenblätter-Beweise
- Alles was in diesem Zusatztutorium angeschnitten worden
- Alles in den nachfolgenden Fragen, alles in den Zusatzfragen

Aufgabe 1

Schreibe alle Loss-Funktionen auf (Multiclass)

Aufgabe 2

Zeige, wie du die Multiclass Lossfunktion von Logistic Regression in die binäre verwandeln kannst. Benutze dafür $\phi(x, y) = \frac{1}{2} * y * \phi(x)$

Aufgabe 3

Schreibe die Formeln aller Aktivierungsfunktionen auf und zeichne den korrespondierenden Graphen.

Aufgabe 4

Was bringt Regularisierung? Wie sieht der Regularisierungsloss für die L2-Norm aus, wie das Update?

Aufgabe 5

Warum ergibt folgende Formel eine Wahrscheinlichkeitsverteilung?
Wo wird sie überall benutzt?

$$\frac{e^{w \cdot \phi(x_i, y_i)}}{\sum_{y \in Y} e^{w \cdot \phi(x_i, y)}}$$

Aufgabe 6

Ist LogisticRegression ein Generatives oder Diskriminatives Modell?

Aufgabe 7

Zeige das unordered und das ordered monomial Mapping an einem Beispielvektor.

Aufgabe 8

Trainingsvektor=$[3, 4]$
Learningrate=0.1
Gewichtsvektor=$[0, 0, 0, 0, 0, 0]$
Wieviele Labels gibt es?
Klasse 1 ist das korrekte Label, wie sieht der Gewichtsvektor im nächsten Schritt aus?

Aufgabe 9

Zeige, dass ein input-hidden-outputlayer Neuronales Netz ohne nichtlineare Aktivierungsfunktion nur so "mächtig" ist wie ein input-outputlayer Neuronales Netz.

Aufgabe 9

Zeige, wie man das XOR Problem mit einem NN löst.