Neuronale Netze

M. Gruber

KW 40, Rev.1

Wir folgen [1], Lec 10.

Wir betrachten neuronale Netze mit Ausgabewerten im Intervall $[-1, 1]$.

Definition 1 (Neuronales Netz, informell) *Ein neuronales Netz ist eine Funktion, die zu Eingabevektoren skalare Ausgabewerte berechnet. Es hat die Form $F_L \circ \ldots \circ F_1$. Jedes F_l ist eine Komposition von zwei Funktionen, einer Matrixmultiplikation und einer nichtlinearen, komponentenweise sigmoidalen Funktion.*

Die Dimension $d^{(0)}$ der Eingabevektoren liegt per Anwendung fest. Die Zahl L und die Dimensionen $d^{(l)}$ der Wertebereiche der F_l für $l = 1, \ldots, L-1$ sind bei der Auslegung des neuronalen Netzes festzulegen. Der Wertebereich von F_L ist eindimensional.

Der Definitionsbereich von F_1 heißt Eingabeschicht (input layer), der Wertebereich von F_L Ausgabeschicht (output layer). Die Wertebereiche von $F_1, \ldots, F_{L-1}$ nennt man verborgene Schichten (hidden layers).

Wenn ein neuronales Netz gefittet wird, werden die Koeffizienten der Matrizen zu Variablen einer Optimierungsaufgabe.

Definition 2 (Neuronales Netz, formal) *Ein neuronales Netz ist eine Funktion*

$$F : \{1\} \times \mathbf{R}^{d^{(0)}} \longrightarrow [-1, 1]$$

mit folgenden Eigenschaften:

1. *$F = F_L \circ F_{L-1} \circ \cdots \circ F_1$ mit*
   (a) *$F_1 : \{1\} \times \mathbf{R}^{d^{(0)}} \longrightarrow \{1\} \times [-1, 1]^{d^{(1)}}$,*
   (b) *$F_l : \{1\} \times [-1, 1]^{d^{(l-1)}} \longrightarrow \{1\} \times [-1, 1]^{d^{(l)}}$ für $1 < l < L$,*
   (c) *$F_L : \{1\} \times [-1, 1]^{d^{(L-1)}} \longrightarrow [-1, 1]$ (d.h. $d_L = 1$).*

   Dabei sind die Dimensionen $d^{(0)}, \ldots, d^{(L-1)} \in \mathbf{N} \setminus \{0\}$ und $d^{(L)} = 1$.
2. *Jedes F_l ist von der Form $F_l(x) = \Theta^{(l)}\big((W^{(l)})^T x\big)$ mit*
   (a) *Matrizen $W^{(l)} \in \mathbf{R}^{d^{(l-1)}+1} \times \mathbf{R}^{d^{(l)}}$ für $1 \leq l \leq L$,*

(b) $\Theta^{(l)} : \mathbf{R}^{d^{(l)}} \longrightarrow \{1\} \times [-1,1]^{d^{(l)}}$, $s \mapsto \begin{bmatrix} 1 & \theta(s_1) & \cdots & \theta(s_{d^{(l)}}) \end{bmatrix}^T$ *für* $1 \leq l < L$,

(c) $\Theta^{(L)} = \theta$.

Dabei ist θ *eine sigmoidale Funktion, z.B.* $\theta = tanh$.

Wie kann ein neuronales Netz trainiert werden?

Gegeben sei eine Trainingsmenge $D = \{[x^{(1)}, y^{(1)}], \ldots, [x^{(N)}, y^{(N)}]\}$. Sei f die unbekannte Funktion, die die Trainingsmenge erzeugt hat ($y^{(n)} = f(x^{(n)})$, $n = 1, \ldots, N$). Sei F das neuronale Netz, das mit f möglichst übereinstimmen soll. Es hängt von den Koeffizienten der $W^{(1)}, \ldots, W^{(L)}$ ab. Denken wir uns die Elemente dieser Matrizen in einem Vektor w angeordnet und schreiben wir $F(x, w)$ statt $F(x)$. Der Fehler, den wir durch geeignete Wahl des w minimieren wollen, ist der *in-sample-error*

$$\mathrm{Err}_{\mathrm{in}}(w) = \frac{1}{N} \sum_{1 \leq n \leq N} (F(x^{(n)}, w) - y^{(n)})^2/2.$$

Der Vollständigkeit halber deuten wir an, wie w konstruiert werden kann. Die Elemente der Matrix $W^{(l)}$ seien $w_{ij}^{(l)}$. Man denke sich die Spalten $w_{*,j}^{(l)}$ zu einem Vektor $w_{*,*}^{(l)}$ und die Vektoren $w_{*,*}^{(l)}$ zu einem Vektor w aneinandergereiht.

Zur Minimierung des *in-sample errors* bietet sich das klassische Gradientenabstiegsverfahren bezüglich w an. Es erfordert allerdings einen hohen Rechenaufwand, denn bei jeder Berechnung des Gradienten muss die gesamte Lernmenge ausgewertet werden. Eine kostengünstige Alternative ist das stochastische Gradientenabstiegsverfahren (s. [2], p.97). Es bleibt nicht so leicht in lokalen Minima stecken und hat sich bei neuronalen Netzen bewährt. Bei jedem Schritt des stochastischen Gradientenabstiegs wird ein $n \in \{1, \ldots, N\}$ zufällig bestimmt und nur der Gradient des ausgelosten Elementarfehlers

$$e_n(w) = (F(x^{(n)}, w) - y^{(n)})^2/2$$

zur Berechnung eines neuen w herangezogen:

$$w_{\mathrm{neu}} \leftarrow w_{\mathrm{alt}} - \eta \nabla e_n(w_{\mathrm{alt}}).$$

Die Größe η ist der Lernkoeffizient, ein Tuningparameter des Verfahrens. Als θ können wir z.B.

$$\theta(s) = \tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$$

wählen. Diese Funktion hat die rechenfreundliche Ableitung $\theta'(s) = 1 - \theta(s)^2$.

Der Ausgabewert eines neuronalen Netzes wird schichtweise von der Eingabeschicht bis zur Ausgabeschicht berechnet *(feed forward network)*. Wenn man das Signal $s^{(l)} = {W^{(l)}}^T F_l \circ \cdots \circ F_1(x)$ kennt, das als Ergebnis der Matrixmultiplikation mit ${W^{(l)}}^T$ in der l-ten Schicht ankommt, kann man mit diesem Wert weiterrechnen bis zur Ausgabeschicht, als wäre $s^{(l)}$ der Eingabevektor eines kleineren neuronalen Netzes mit $L - l + 1$ Schichten. Auch den Elementarfehler e_n kann man als

Funktion von w und $s^{(l)}$ auffassen und entsprechend differenzieren. In diesem Sinne ist

$$\frac{\partial e_n}{\partial w_{ij}^{(l)}} = \frac{\partial e_n}{\partial s_j^{(l)}} \cdot \frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}}. \tag{1}$$

Auf die Berechnung des ersten Faktors in (1) werden wir uns im Folgenden konzentrieren. Wir führen für ihn das Symbol

$$\delta_j^{(l)} = \frac{\partial e_n}{\partial s_j^{(l)}}(s^{(l)})$$

ein. Für den zweiten Faktor halten wir fest:

$$\frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}} = x_i^{(l-1)} = F_{l-1} \circ \cdots \circ F_1(x).$$

Nun also zu den Größen $\delta_j^{(l)}$. Im Fall $l = L$ gibt es nur ein j, nämlich $j = 1$, und man hat

$$\delta_1^{(L)} = (\theta(s_1^{(L)}) - y_n)(1 - \theta(s_1^{(L)})^2).$$

Schicht für Schicht kann man nun im Netz zurückrechnen. Bevor wir dies tun, werfen wir noch einen Blick auf den Zusammenhang zwischen den Signalen $s^{(l-1)}$ und $s^{(l)}$. Es ist

$$s^{(l)} = {W^{(l)}}^T \Theta^{(l-1)}(s^{(l-1)}), \tag{2}$$

insbesondere

$$s_j^{(l)} = \sum_{1 \le k \le d^{(l)}} w_{kj}^{(l)} \cdot \theta(s_k^{(l-1)}), \tag{3}$$

folglich

$$\frac{\partial s_j^{(l)}}{\partial s_i^{(l-1)}}(s^{(l-1)}) = w_{ij}^{(l)} \cdot (1 - \theta(s_i^{(l-1)})^2). \tag{4}$$

Nun zur Rekursion für die $\delta^{(l)}$. Angenommen, man kennt $\delta^{(l)} = [\delta_1^{(l)}, \ldots, \delta_{d^{(l)}}^{(l)}]$ schon. Dann ist

$$\delta_i^{(l-1)} = \frac{\partial e_n}{\partial s_i^{(l-1)}}(s^{(l-1)}) = \sum_{1 \le j \le d^{(l)}} \frac{\partial e_n}{\partial s_j^{(l)}}(s^{(l)}) \cdot \frac{\partial s_j^{(l)}}{\partial s_i^{(l-1)}}(s^{(l-1)}) = \sum_{1 \le j \le d^{(l)}} \delta_j^{(l)} \cdot \frac{\partial s_j^{(l)}}{\partial s_i^{(l-1)}}(s^{(l-1)})$$

und mit (4) somit

$$\delta_i^{(l-1)} = \sum_{1 \le j \le d^{(l)}} \delta_j^{(l)} \cdot w_{ij}^{(l)} \cdot (1 - \theta(s_i^{(l-1)})^2) \quad \text{für} \quad l = 2, \ldots, L. \tag{5}$$

Die Beziehung (5) kennt man unter dem Namen *back propagation*.

Damit haben wir nun ein effizientes Verfahren zur Berechnung des w-Gradienten von e_n in der

Hand:

$$\frac{\partial e_n}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} \cdot x_i^{(l-1)}. \tag{6}$$

Beispiel 1 (Computer-Experiment) *Gegeben war eine Trainingsmenge vom Umfang* $N = 100$ *(s. Abb. 1), links). Schwarze Punkte sind mit* -1 *gelabelt, grüne mit* $+1$.

Ein neuronales Netz mit drei fünfdimensionalen hidden layers wurde an den Daten trainiert. Gefunden wurde eine Funktion mit Werten in $[-1, 1]$*, dessen Kontur man in Abb. 1 rechts sieht. Als Gewichtsmatrizen des neuronalen Netzes ergaben sich*

$$W^{(1)} = \begin{pmatrix} -0.566017 & -1.02236 & -0.655917 & 0.605474 & -0.0495186 \\ -0.178773 & -1.35201 & 1.30193 & 1.49504 & 0.620174 \\ 1.28258 & 1.39478 & -0.289099 & 1.644 & 0.369747 \end{pmatrix}$$

$$W^{(2)} = \begin{pmatrix} -0.915004 & -0.674408 & -0.423988 & -0.666365 & -0.41378 \\ -0.85416 & -0.410386 & 0.407706 & -1.00429 & -0.0335844 \\ -0.90027 & -0.810325 & 1.14244 & -0.658086 & -0.752653 \\ -0.170535 & 0.00132637 & -0.102541 & 1.48672 & 0.283398 \\ -0.536865 & 0.157238 & -1.48445 & 1.51083 & 0.320752 \\ 0.252262 & -0.960197 & -0.779086 & -0.0912184 & -0.0267426 \end{pmatrix}$$

$$W^{(3)} = \begin{pmatrix} 0.550016 & -1.20572 & -0.517775 & 0.688077 & -0.512404 \\ -1.44701 & -0.815882 & 0.420368 & 0.395188 & 0.0258037 \\ -0.811572 & -0.494787 & 0.25017 & -0.221192 & 0.210993 \\ -0.223064 & -0.892331 & -1.37032 & 0.991742 & -0.848886 \\ -0.413874 & 0.120176 & -1.07813 & -1.80178 & 1.33963 \\ -0.207105 & -0.197274 & 0.341746 & -0.735518 & 0.41511 \end{pmatrix}$$

$$W^{(4)} = \begin{pmatrix} -0.071606 \\ 1.54443 \\ -1.59185 \\ -1.89651 \\ -1.53374 \\ 1.01529 \end{pmatrix}$$

Der Lernkoeffizient für den stochastischen Grandientenabstieg war $\eta = 0.1$*. Die Iteration wurde beendet, sobald das Norm-Quadrat des Gradienten den Wert* 10^{-10} *unterschritt. Die obere Schranke für die Anzahl der Iterationen war* 10^7*.*

Literatur

[1] Y. S. Abu-Mostafa. Learning from Data. Caltech CS 156.

[2] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning From Data.* AMLBook, 2012.

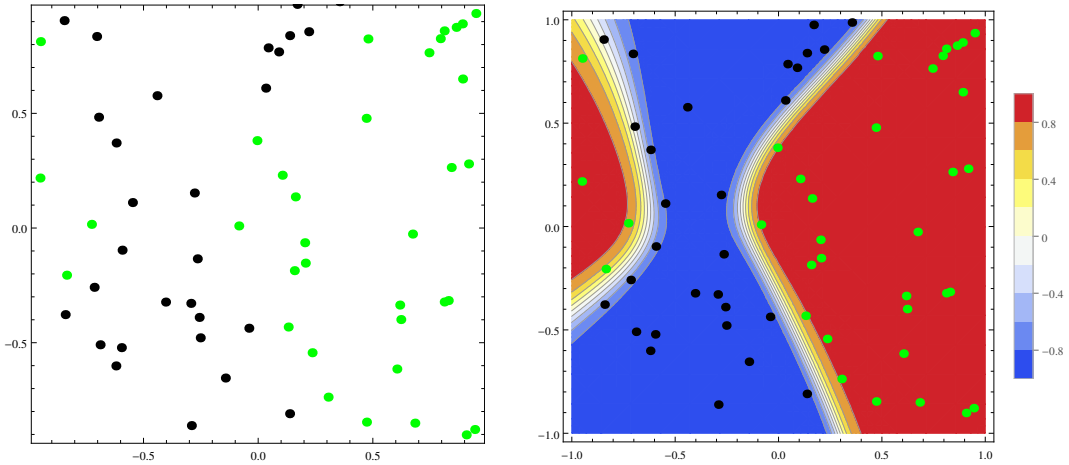

Abbildung 1: Trainingsmenge (links) und Konturplot des neuronalen Netzes (rechts).