7.4 Rectified linear units und ihre Generalisierung

RLUs sind definiert durch:

(92) $g(x) = max\{0, x\}$

Diese Funktion ist nicht linear und natürlich nur dann interessant, wenn man sie zusammen mit einer anderen, linearen Funktion verwendet. Wir haben also eine Funktion:

(93) $h(\mathbf{x}) = g(M\mathbf{x} + \mathbf{b})$

Das gute an diesen Funktionen ist natürlich ihre Einfachheit. Das Problem ist: sie haben ein langes Plateau, das ist Problem für Optimierung!

Es gibt 2 Generalisierungen von RLUs:

1. Es gibt eine (parametrisierte) Funktion die RLUs generalisiert:

   (94) $g(\vec{x}, \alpha)_i = max(0, x_i) + \alpha_i min(0, x_i)$

   Im Fall von $\alpha_i = 0$ haben wir eine RLU-Funktion. Falls $\alpha_i = -1$, dann wird das ganze zur Betragsfunktion, die man in diesem Fall **absolute value rectification** nennt. Man kann auch α_i zu einem kleinen Wert wie 1^{-n} setzen, mit z.B. $n = 100$; das ergibt **leaky** RLU. In **parametrischen** RLU ist α_i ein Parameter, der "gelernt" wird.

2. Es gibt noch die sog. **maxout units**. Anstatt komponentenweise eine Funktion zu applizieren werden die n Komponenten in Gruppen zu $k < n$ Werten aufgeteilt. Wir nennen diese Gruppe $G_1, ..., G_k$, wobei also

   $$G_1, ..., G_j \in \mathbb{R}^k$$

   Die Ausgabe der maxout Funktion ist wiederum ein Vektor, wobei

   (95) $g : \mathbb{R}^n \to \mathbb{R}^k$

   die Funktion für jede Gruppe eine Komponente ausgibt. Sie ist dementsprechend einfach definiert:

   (96) $g(\vec{z})_i = max(z_j : j \in G_i)$

Dadurch, das wir Gruppen bilden, ist die Funktion etwas robuster und weniger anfällig für Störungen in einzelnen Parametern.

7.5 Ausgabefunktionen

Hier suchen wir Funktionen

$$f : \mathbb{R}^m \to \mathbb{R}$$

Wir haben bereits gesehen, dass Normen diese Funktion erfüllen, aber das sind lineare Funktion und hier nicht von Interesse. Eine sehr wichtige Funktion ist die sog. *max*-Funktion, die einfach das Maximum ausgibt:

$$max(x_1, ..., x_m) = max\{x_1, ..., x_m\} \tag{97}$$

Sie liefert also den höchsten Wert. Das Problem dieser Funktion ist: sie ist nicht differenzierbar. Daher gibt es die sog. *softmax*-Funktion, die eine ähnliche Funktion übernimmt, aber differenzierbar ist (daher *soft*, weil sie keine Knicke hat).

$$softmax(x_1, ..., x_n)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{98}$$

Diese Notation ist gewählt um die Definition übersichtlicher zu machen; falls $\mathbf{x} \in \mathbb{R}^n$, dann ist auch $softmax(\mathbf{x}) \in \mathbb{R}^n$. Die softmax-Funktion ist eine Art "weiche" argmax-Funktion: dadurch dass wir exponentieren bekommen alle Komponenten, die nicht unbedeutend kleiner sind als der Größte, eine vernachlässigbare Größe. Der große Vortei von softmax ist: es ist eine kontinuierliche Funktion. Das ist sehr wichtig für die Optimierung, für die wir Gradienten brauchen. Softmax-Funktionen werden üblicherweise als Ausgabefunktionen verwendet: sie liefern tatsächlich eine Wahrscheinlichkeitsverteilung wie in (86).

Wichtig ist in allen diesen Funktionen, dass wir niemals ein Plateau haben, also z.B. niemals 0 erreicht für eine Sigmoidfunktion. Denn in diesem Fall können wir nicht weiter optimieren, der Gradient gibt uns keinerlei Information.

8 Das einfache Neuron

8.1 Logistische Regression

Logistische Regression ist im engeren Sinne keine Regression, sondern Klassifikation; es ist aber eine Klassifikation, die auf linearer Regression aufbaut. Logistische Regression funktioniert im einfachen Fall, wenn wir 2 Klassen haben, zwischen denen wir auswählen, z.B. A und B. Unser Problem ist also folgendes: wir möchten eine Eingabe in $\mathbb{R}$ nach A oder B klassifizieren. Wir machen das mittels der logistischen Sigmoid-Funktion

$$S(x) = \frac{1}{1 + e^{-x}} \tag{99}$$

Einige Beobachtungen: wir haben

- $lim_{x \to -\infty} S(x) = 0$
- $lim_{x \to \infty} S(x) = 1$
- $S(x) \in (0, 1)$ f.a. $x \in \mathbb{R}$
- $S(x)$ ist streng monoton steigend in x

Allgemeiner nennt man solche Funktionen **Sigmoidfunktionen**. Ein anderes Beispiel für eine solche Funktion ist

$$T(x) = \frac{x}{1 + |x|} \tag{100}$$

$$U(x) = \frac{x}{1 + x^2} \tag{101}$$

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 1 - \frac{2}{e^{2x} + 1} \tag{102}$$

Was allen diesen Funktionen gemeinsam ist, ist dass sie im Bereich um 0 relativ schnell von 0 Richtung 1 wechseln, sonst fr groe positive/negative Zahlen langsam Richtung 1/0 konvergieren. Es gibt also nur einen kleinen Bereich, in dem eine nicht extreme Änderung im Argument eine signifikante Änderung im Wert verursacht; ansonsten gibt es relativ schnell eine Sättigung. solche Prozesse findet man oft in der Natur, z.B. bei Populationen. Solche logistischen Funktionen haben allerdings die Einschränkung dass sie nur $\mathbb{R} \to \mathbb{R}$

definiert sind. Aktivierungsfunktionen spielen in neuronalen Netzen eine große Rolle, und werden normalerweise auf die einzelnen Komponenten von Vektoren angewendet.

Wenn wir nun 2 Klassen haben, dann können wir einfach sagen: wir interpretieren den Wert als eine **Wahrscheinlichkeit**, also:

$$P(X = A|x) = S(f(x)) \tag{103}$$

wobei f eine lineare Funktion ist, die wir mittels linearer Regression induzieren. Wir transformieren also eine Funktion in die reellen Zahlen zu einer Funktion nach $[0, 1]$:

$$f : \mathbb{R} \to \mathbb{R} \Longrightarrow S \circ f : \mathbb{R} \to [0, 1]$$

Das Ergebnis können wir dann als Wahrscheinlichkeit interpretieren. Das gibt uns nun einen Klassifikator:

$$C(x) = \begin{cases} A, \text{ falls } P(X = A|x) > 0.5 \\ B \text{ andernfalls} \end{cases} \tag{104}$$

Das funktioniert natürlich nur, falls wir nur zwei Klassen (hier A, B) zur Verfügung haben. Intuitiv bilden wir hier die Eingabe auf eine Wahrscheinlichkeit ab, und Klassifizieren nach Maximum Likelihood. Natürlich kann auch das sehr leicht generalisiert werden auf beliebige Eingaben $\mathbf{x} \in \mathbb{R}^n$.

8.2 Bedeutung

Die Bedeutung der logistischen Regression ist erstmal folgende: wir suchen eine Klassifikationsfunktion

$$f : \mathbb{R}^n \to \{0, 1\}$$

Wir tun das aber mittels zweier Zwischenschritte:

$$\mathbb{R}^n \underset{f}{\Longrightarrow} \mathbb{R} \underset{S}{\Longrightarrow} \{0, 1\}$$

Die zugrundeliegende Annahme ist hierbei, dass es einen kritischen Bereich gibt, auf dem die Klassifikation von 0 zu 1 springt und umgekehrt ist; aber damit man das sieht, muss man zunächst eine **numerische Transformation** durchführen, nämlich die lineare Regression. Genauer gesagt, die lineare Funktion f kann die numerischen Werte so transformieren, dass gilt:

$$\text{Falls } f(x) > s, \text{ dann } C(x) = 1, \text{ und falls } f(x) \leq s, \text{ dann } C(x) = 0$$

Das bedeutet, wir haben eine *quasi-lineare Abhängigkeit* von unseren Eingabedaten und den Klassen. Das kann man noch genauer fassen, es gibt nämlich eine genaue Beschreibung wenn wir den **Hyperparameter** des **Grades der Funktion** mitberücksichtigen. Nehmen wir an, wir suchen eine Funktion $f : \mathbb{R} \to \{0, 1\}$. Wir wissen, dass

- Eine lineare Funktion ist hat eine Gerade als Graphen, kann also einen gewissen Wert nur einmal annehmen
- Ein Polynom zweiten Grades hat einen Wendepunkt, kann einen gewissen Wert höchstens zweimal annehmen.
- ...
- Ein Polynom $n$ten Grades hat n Wendepunkte, kann einen gewissen Wert höchstens n-mal annehmen.

Dann heißt das soviel wie:

- Falls wir eine lineare Funktion haben, dann gibt ein $\alpha \in \mathbb{R}$, so dass falls $x < \alpha$, dann $f(x) = 1$ (bzw. 0), ansonsten $f(x) = 0$ (bzw. 1). Wir spalten also den Eingaberaum $\mathbb{R}$ in zwei Teile.

- Falls wir ein Polynom zweiten Grades haben, dann gibt $\alpha_1, \alpha_2 \in \mathbb{R}$, so dass falls $\alpha_1 < x\alpha_2$, dann $f(x) = 1$ (bzw. 0), ansonsten $f(x) = 0$ (bzw. 1). Wir spalten also den Eingaberaum $\mathbb{R}$ in drei Teile.
- ...
- Falls wir ein Polynom n-ten Grades haben, haben wir $\alpha_1, ..., \alpha_n$, die $\mathbb{R}$ in $n + 1$ Teile spalten, und wir klassifizieren auf dieser Basis.

8.3 Lernen & Anwendung

Wir betrachten folgendes Beispiel: wir möchten, als abhängige, diskrete Variable vorhersagen, ob ein Student die Prüfung besteht (0 oder 1; abhängig bedeutet nur: wir möchten das vorhersagen). Als Prädiktor nutzen wir die Anzahl der Stunden, die der Student gelernt hat. Unsere Regressionsfunktion soll eine einfache lineare Funktion sein. Unser Datensatz sieht nun wie folgt aus (nennen wir den Datensatz $D1$):

Stunden gelernt	4	5	6	7	8	9	10
Bestanden	0	0	1	0	1	1	1

Eine lineare Regression liefert hier folgendes Ergebnis (danke R):

(105) $f(x) = 0.1786x - 0.6786$

Wir haben also eine positive Abhängigkeit von Lern-Stunden und Klausur-Bestehen (zum Glück); aber dieses Ergebnis ist noch unbefriedigend: wir können das nicht sinnvoll als Wahrscheinlichkeit interpretieren. Wir setzen aber nun diesen Term ein in unsere Aktivierungsfunktion, und defineren:

(106) $$P(\text{bestehen}|n \text{ Stunden lernen}) = \frac{1}{1 + e^{-f(x)}} = \frac{1}{1 + e^{-(0.1786x - 0.6786)}}$$

Das liefert uns nun ordentlich Werte:

Stunden	4	5	6	7	8	9	10
P(bestehen)	0.508	0.553	0.597	0.639	0.679	0.717	0.752

Das ist schon besser, aber nicht wirklich optimal, insbesondere stört die Asymmetrie (bei 4 haben wir bereits eine ziemlich hohe Wahrscheinlichkeit!).

Die logistische Funktion interagiert mit der linearen auf eine Art und Weise, die nicht optimal ist. Wir sollten also stattdessen folgendes suchen:

$$(107)\ argmin_{a,b\in\mathbb{R}} \sum_{(x,y)\in D} \frac{1}{1+e^{-(ax+b)}} - y$$

Wir nutzen also die Aktivierungsfunktion, um den Unterschied zwischen 0 und 1 sichtbar zu machen, da eine lineare Funktion hierzu nicht in der Lage ist. Um 107 auszurechnen gibt reichen keine elementaren Methoden, man muss also etwas komplexere numerische Optimierung anwenden (der Computer kann das). R macht das mit dem Kommando:

> glm(x ~ y, family = binomial())

Nun definieren wir:

> g <- function(x){1/(1+exp(-(1.251*x-8.113)))}

Das sollte nun stimmen; wir bekommen dementsprechend:

Stunden	4	5	6	7	8	9	10
P(bestehen)	0.0427	0.135	0.353	0.656	0.869	0.959	0.988

Tadaa! Wir bekommen also eine Wahrscheinlichkeit. Was aber interessanter ist, ist folgende Frage: gegeben diese Ergebnisse, wie ist die Wahrscheinlichkeit, dass die Abhängigkeit des Bestehens von der Lernzeit reiner Zufall ist? Das ist eine klassische statistische Frage, und im Zusammenhang mit logistischer Regression gibt es den sog. **Wald-test**:

$$(108)\ \frac{(\theta_{ML}-\theta)^2}{var(\theta_{ML})}$$

Er gibt die Wahrscheinlichkeit, dass die gegebene Verteilung zufällig ist, also dass es keine Abhängigkeit der beiden Variablen (Lernzeit / Bestehen) gibt. Das ist verwandt mit dem Test des Likelihood-Verhältnisses, was ein wenig einfacher zu verstehen ist. Seien

- ω unsere Beobachtungen, also die Daten in D1;
- H_0 die Nullhypothese, also Unabhängigkeit der Variablen: Lernzeit hat keinen Einfluss auf das Bestehen, Wahrscheinlichkeit von Bestehen oder Nicht-bestehen wird durch Maximum Likelihood geschätzt.

- H_1 die Alternativhypothese, also unsere durch logistische Regression berechnete bedingte Verteilung.

Dann haben wir den Likelihood-Quotienten:

$$(109)\quad R(\omega) := \frac{P(\omega|H_0)}{P(\omega|H_1)} \qquad \text{(Sonderfall für } P(\omega|H_1) = 0)$$

Hierauf kann man nun den Schwellentest S_t anwenden, $t > 0$, und üblicherweise $t = 0.05$. Zur Erinnerung: das ist der Test, der sich für H_0 entscheidet falls $R(\omega) > t$, und für H_1 andernfalls, also:

$$(110)\quad S_t(\omega) = \begin{cases} H_0, \text{ falls } R(\omega) > t \\ H_1 \text{ andernfalls.} \end{cases}$$

Das können wir/Sie nun ausrechnen – aber nicht hier!