Bachelorarbeit

Jan Philipp Fortowski

xx.xx.2024

Contents

1 Pooling Layer **3**

1.1 Forward Propagation im Pooling Layer 4

1.2 Backpropagation im Pooling Layer 5

2 Unterstützungs Templates **6**

3 Einleitung **8**

3.1 Motivation 8

4 Quellen **9**

5 Liste der Abbildungen **10**

1 Pooling Layer

Pooling Layer werden dazu verwendet, um das sogenannte Downsampling zu betreieben. Downsampling bedeutet einfach, dass eine geringere Auswahl an Daten weitergegeben werden. In einem Netzwerk benutzt man eines dieser Verfahren, um die weiteren Schichten nur mit den nötigsten Daten zu belasten. Gerade in Anbetracht der Größe der Daten, die bei den Convolutional Layern entstehen, kann das viel Zeit sparen. Aber Hauptsächlich sorgt dies auch dafür, dass das Netzwerk weniger anfällig für Veränderungen der Daten wird. leichte Veränderungen der Daten können für gravierende Änderungen, das heißt wenn ein Bild nicht korrekt Zentriert ist, oder wenn es um ein paar Grad gedreht wird, kommt ein Netzwerk schnell durcheinander. Die Convolution Layer erstellen Feature Maps, in denen die Features allerdings auch die gleichen Positionen haben, wie in den Inputs. Dadurch können bei kleinen Veränderungen schon Fehler in den nächsten Schichten entstehen. Dem kann man durch das Downsampling entgegenwirken.

Auf eine Gewisse weise können die Daten bereits in den Convonlutional Layern "gedownsampelt" werden, wenn die Schrittgröße (Stride) beim anwenden der Filter Größer ist als 1.

Wie genau Funktioniert ein Pooling Layer dann? so wie bei den Convolutional Layern werden Filter eingesetzt. Diese sind aber kleiner, und funktionieren anders. Meist ist ein Filter hier nur 2x2 groß, und bei der Anwendung wird eine Schrittgröße von 2 verwendet. Die Filter überlappen die Felder nicht, die bereits behandelt wurden. Auf die Pixel, auf die der Filter angewand wird, werden nun eine Pooling Operation angewand.

Zwei mögliche Pooling Operationen sind das Durschnitts Pooling und das Maximum Pooling.

- Das Durchschnitts Pooling gibt den Durchschnitt aller Werte als Output zurück, während
- das Maximum Pooling nur den Größten Wert der betrachteten Werte zurückgibt.

Es gibt noch weitere, wie zum Beispiel das Globale Pooling, bei dem die Gesammte Feature Map auf einen einzigen Wert reduziert wird. Dadurch soll auf eine möglichst Aggressive Art festgestellt werden, ob ein bestimmtes

Feature überhaupt in der Feature Map vorkommt. Je nach Anwendungsfall kann auch diese Methode Erfolg bringen.

Zunächste wird die Maximum Pooling Layer betrachtet. genau wie bei den Convolutional Layern, wird hier eine StepSize verwendet. Zwar ändert sich diese in den meisten Fällen nicht, aber es ist gut, die Flexibilität zu haben. Die windowrSize legt fest, welche Dimensionen der Filter haben soll.

```
public class MaxPoolLayer extends Layer {

    private int stepSize;
    private int windowSize;

    private int inLength;
    private int inRows;
    private int inCols;
```

1.1 Forward Propagation im Pooling Layer

Wie oben beschrieben, wird hier ein kleiner Filter benutzt, der über die Input Bilder gleitet, und entweder den maximalen Wert, oder den Durchschnittswert zurückgibt. Zunächst wird eine Methode benötigt, die den Filter darstellt, und eine Feature Map durchläuft.

```
public double[][] pool(double[][] input){
        double[][] output = new double[getOutputRows()][
            getOutputCols()];
        for(int r=0; r<getOutputRows(); r+=stepSize){
            for(int c=0; c<getOutputCols(); c+=stepSize){
                double max = 0.0;
                for(int x=0; x<windowSize; x++){
                    for(int y=0; y<windowSize; y++){
                        if(max<input[r+x][c+y]){
                            max=input[r+x][c+y];
                        }
                    }
                }
                output[r][c] = max;
            }
        }
        return output;
    }
```

1.2 Backpropagation im Pooling Layer

2 Unterstützungs Templates

Die Gliederung dieser Arbeit, entspricht einer organischen Herangehensweise oder einer Anleitung, wie ein Netzwerk aufgebaut wird. Jedes Kapitel entspricht einem Teil des Netzwerkes, also den Funktionen, der Initialisierung, der Query oder Abfrage und dem Lern- oder Backpropagation-Algorithmus. Jedes der Kapitel startet mit einer Übersicht, über die Funktionalität, erläutert die Theorie dahinter, teilweise auch mathematisch, und schließt, mit dem daraus resultierenden Code ab.

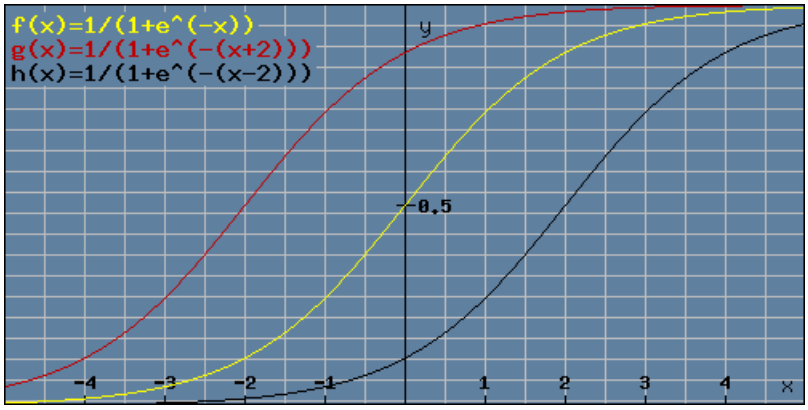

Figure 1: Sigmoidfunktion mit Bias

- Dendriten
- der Zellkörper
- das Axon

```java
public class NeuralNetwork {
    Layer[] layers;
    double learnRate;    // Initialisierung
    public NeuralNetwork(double learnRate, int... layerSizes)
        {
        layers = new Layer[layerSizes.length - 1];
        for (int i = 0; i < layers.length; i++) {
            layers[i] = new Layer(layerSizes[i], layerSizes[i
                + 1]);
        }
        this.learnRate = learnRate;
    }
}
```

Aber verfolgen wir diesen Gedanken doch einmal an einem Beispiel: Sei $f(w)$ unsere Funktion:

$$f(w) = w^2 + 3$$

Wir nennen die kleine Verschiebung von w jetzt h. Dann gilt zumindest schon einmal:

$$\Delta w = (w + h) - w$$

Und gekürzt:

$$\Delta w = h$$

Dann wäre die Änderungsrate also:

$$\ddot{A}nderungsrate = \frac{\Delta e}{\Delta w}$$

$$\frac{\Delta e}{\Delta w} = \frac{f(w + h) - f(w)}{h}$$

Wenn wir uns dann die Mühe machen, $f(w)$ aus zuschreiben, ergibt sich daraus:

$$\frac{\Delta e}{\Delta w} = \frac{(w + h)^2 + 3 - (w^2 + 3)}{h}$$

Dann fangen wir an Klammern auszurechnen:

$$\frac{\Delta e}{\Delta w} = \frac{w^2 + w * h + w * h + h^2 + 3 - w^2 - 3}{h}$$

$$\frac{\Delta e}{\Delta w} = \frac{w^2 + w * h + w * h + h^2 - w^2}{h}$$

$$\frac{\Delta e}{\Delta w} = \frac{w * h + w * h + h^2}{h}$$

h kürzen:

$$\frac{\Delta e}{\Delta w} = w + w + h$$

$$\frac{\Delta e}{\Delta w} = 2w + h$$

Und jetzt zum Interessanten Teil. Da wir h nicht gleich 0 setzten können, können wir allerdings h gegen 0 laufen lassen, dann verwenden wir die Leibniz-Notation. Das bedeutet, dass wir anstatt Δw und Δe, wobei Δ eine sehr kleine Vergrößerung darstellt, jetzt dw und de verwenden, wobei d für eine unendlich kleine Vergrößerung steht, eine sogenannte Infinitesimalzahl. Das

sieht dann ungefähr so aus:

$$\frac{de}{dw} = \lim_{h \to 0} 2w + h$$

$$\frac{de}{dw} = 2w$$

3 Einleitung

3.1 Motivation

Neuronale Netzwerke werden vor allem für Klassifikationsverfahren verwendet. In der Praxis gibt es viele Anwendungsbereiche, in denen es vorteilhaft ist, große Mengen von Daten automatisch zu klassifizieren. Einige Beispiele, wären die Bild- und Schrifterkennung, die man dazu verwendet, Kennschilder von Autos, maschinell auszulesen. Solche Technologien werden immer häufiger auf Parkplätzen und Autobahnen eingesetzt. Aber, auch fast jedes Handy kann mittlerweile Schrift erkennen, die mit der Kamera aufgenommen wird. Auch in der Medizin, beim Auswerten von Röntgenbildern, in der Biologie, zum Erkennen von Pflanzen auf Fotos und noch vielem mehr, werden Neuronale Netzwerke inzwischen eingesetzt.
Es gibt kaum einen Bereich, in dem sich nicht eine mögliche Anwendung für Neuronale Netzwerke finden lässt. Ganz besonders aufwändige Netzwerke, werden mittlerweile auch für konstruktive Aufgaben verwendet, wie zum Beispiel Sprachmodelle, unter anderem ChatGPT und Bilder generierende KIs, wie Midjourney. Typische Probleme, bei welchen neuronale Netzwerke eingesetzt werden, sind komplexe Aufgaben, mit gigantischen Datenmengen. Eine Aufgabe ist zu komplex, wenn eine Lösung nicht manuell programmiert werden kann, unter anderem, bei der Bilderkennung. Außerdem sollten solche Probleme automatisiert werden, weil es sich nicht lohnt, Menschen dazu einzusetzen. In solchen Fällen, werden neuronale Netzwerke eingesetzt. Im Laufe der Zeit sind die modernen Netzwerke immer komplexer geworden und liefern nahezu menschliche Ergebnisse, im Bruchteil der Zeit, die Menschen dafür brauchen würden. Aber wie genau funktioniert so ein Netzwerk?

4 Quellen

- 1. Alexey Kravets, "Forward and Backward propagation of Max Pooling Layer in Convolutional Neural Networks", URL: https://towardsdatascience.com/forward-and-backward-propagation-of-pooling-layers-in-convolutional-neural-networks-11e36d169bec

- 2. T. Rashid, Neuronale Netze selbst programmieren - Ein verständlicher Einstieg mit Python. Paderborn: O'Reilly Verlag; 2017.

- 3. S. Lague, How to Create a Neural Network (and Train it to Identify Doodles). Hrsg. auf dem YouTube Kanal [Sebastian Lague](`https://www.youtube.com/@SebastianLague`). Ohne Jahr [zitiert am 16. September 2023]. Abrufbar unter: URL: `https://www.youtube.com/watch?v=hfMk-kjRv4c`.

- 4. "Ableitung der Sigmoid-Funktion" URL: `https://ichi.pro/de/ableitung-der-sigmoid-funktion-91708302791054`

- 5. "What is the role of the bias in neural networks?" URL: `https://stackoverflow.com/questions/2480650/what-is-the-role-of-the-bias-in-neural-networks`.

- 6. "How do I choose the optimal batch size?", URL: `https://ai.stackexchange.com/questions/8560/how-do-i-choose-the-optimal-batch-size`

5 Liste der Abbildungen

List of Figures

1 Sigmoidfunktion mit Bias 6