8. Übungsblatt

Aufgabe 39 Visualisierung von Gewichten

Betrachten Sie folgende Funktion $f(x) = relu(xW)$, wobei $x = (x_1, x_2, x_3, x_4, x_5)$ und

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

Initialisieren Sie $x = \vec{0}$ und maximieren Sie den zweiten Spaltenvektor $(xW)_1$ mit Gradientenaufstieg für zwei Schritte mit einer Lernrate von 1.

- Interpretieren Sie, was das Ergebnis des Gradientenaufstiegs zeigt.
- Betrachten Sie die Berechnung von xW. Überlegen Sie, bei welcher Architektur Sie diese Art von Berechnung bereits gesehen haben.

Aufgabe 40 Fehlerfunktion: Kreuzentropie

Gegeben sei ein Datensatz von Bildern von fünf verschiedenen Tieren. Jedes Bild zeigt genau ein Tier. Jedes Bild hat zudem ein Label P, welches das abgebildete Tier als One-Hot-Kodierung wie folgt angibt:

Tier	Label
Hund	[1 0 0 0 0]
Katze	[0 1 0 0 0]
Pferd	[0 0 1 0 0]
Adler	[0 0 0 1 0]
Kuh	[0 0 0 0 1]

- Wofür ist die Entropie bzw. die Kreuzentropie im allgemeinen ein Maß? Warum und wie können wir dies für Neuronale Netze nutzen?
- Nehmen Sie an ein Netz, das auf diesen Daten trainiert, generiert für das Bild eines Hundes die Ausgabe Q1 = [0.45 0.25 0.03 0.07 0.2]. Berechnen Sie die Kreuzentropie H(P,Q1). Das Netz klassifiziert das Bild korrekt als Hund. Warum ist die Ausgabe trotzdem problematisch?
- Das Netz hat nun eine längere Zeit trainiert und generiert für das selbe Bild die Ausgabe Q2 = [0.97 0.01 0.003 0.007 0.01]. Berechnen Sie erneut die Kreuzentropie H(P,Q2). Was können wir basierend auf diesen Werten über unser Netz aussagen?

Hinweis : Nehmen Sie für log den natürlichen Logarithmus an.

Aufgabe 41 Aktivierungsfunktion: Softmax

Im überwachten Lernen ist ein häufiger Anwendungsfall die Klassifikation. Ein typisches Beispiel ist die Klassifikation von Bildern wie beispielsweise in der ILSVRC vom http://www.image-net.org/.
Der quadratische Fehler ist für Klassifikationsaufgaben nicht geeignet. Aus diesem Grund verwendet man in der die Kreuzentropie : $H(P,Q) = -\sum_{i=1}^{n} P(i) \log Q(i)$, wobei P und Q jeweils Wahrscheinlichkeitsverteilungen sind. Das bedeutet, dass unsere Netzwerke eine Wahrscheinlichkeitsverteilungen als Ausgabe haben muss. Nehmen wir eine lineare Ausgabe z aus dem Netzwerk an. Auf diese Ausgabe wird dann die Softmaxfunktion angewandt:

$$softmax(z)_i = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

- Nehmen Sie an, dass Bilder von fünf verschiedenen Tieren als Datensatz genutzt werden. Die Labels sind dabei wie folgt one-hot Codiert :

Tier	Label
Hund	[1 0 0 0 0]
Katze	[0 1 0 0 0]
Pferd	[0 0 1 0 0]
Adler	[0 0 0 1 0]
Kuh	[0 0 0 0 1]

  Die Ausgabe des Netzes sei wie folgt:

Eingabebild	Ausgabe y_i
Katze	[6 5 2 3 1]
Pferd	[4 3 8 1 -2]
Pferd	[4 4 1 -1 0]

  Berechnen Sie die Softmaxausgaben.

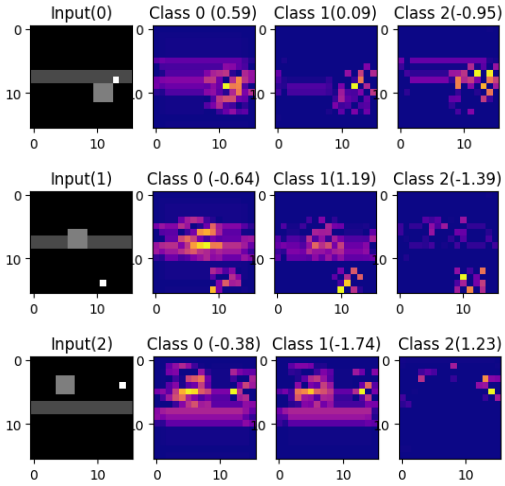
- Da wir bei Klassifikation die Kreuzentropie minimieren wollen $H(labels, prediction)$. Berechnen Sie die Kreuzentropie für die Vorhersagen aus der Tabelle.

- Wie Sie bei der Berechnung sehen, beinhaltet die Kreuzentropie den Term $-\log(softmax(z)_i)$. Setzen Sie die Softmaxfunktion ein und vereinfachen Sie so weit wie möglich. Argumentieren Sie warum dieser Term gut für die Optimierung geeignet ist.

Hinweis : Nehmen Sie für log den natürlichen Logarithmus an.

Aufgabe 42 GradCam

In dieser Aufgabe wollen wir uns mit GradCam beschäftigen. Dafür rechnen wir ein kleines Beispiel von Hand und kucken uns dann ein Beispielproblem an.



a) Bestimmen Sie die Ausgabe von GradCAM. Gegeben sind die Gradienten $\tilde{A}^k = \frac{\partial y^c}{\partial A^k}$ und die Feature Maps.

$$\tilde{A}^1 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad \tilde{A}^2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \tilde{A}^3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 3 & 3 \end{pmatrix}$$

$$A^1 = \begin{pmatrix} 4 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad A^2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad A^3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

b) Wir betrachten nun folgendes Beispielszenario. Gegeben sind die GradCAM Ergebnisse für ein Model das auf Bildern trainiert wurde, auf denen sich eine horizontale Linie und ein großer und ein kleiner Punkt befinden. Die Aufgabe des Model ist es 0 auszugeben, wenn der kleine Punkt auf der Linie ist, 1 unterhalb der Linie und 2 oberhalb der Linie.

Die Beschrifftung des Graphen beeinhaltet bei Input die echte Klasse und bei den Klassen die Werte der unnormalisierten Wahrscheinlichkeiten (vor der Anwendung der softmax Funktion.).

Beantworten folgende Fragen:

- Kann das Modell die gestellte Aufgabe lösen ?
- Ist das Model robust gegenüber dem größeren Punkt ?

Aufgabe 43 Bonus: Automatisches Differenzierungsframework (6)

Dieses Mal wollen wir Convolution differenzierbar implementieren, da es mitverantwortlich ist für den Erfolg von Deep Learning. Dabei gehen wir wie folgt vor :

a) Implementieren Sie als erstes den forward pass der convolution auf numpy arrays in der Funktion h_conv2d. Dabei machen wir folgende Annahmen:

- Der Stride ist immer 1.
- Der Kernel hat immer eine ungerade Größe.

- Es gibt kein Padding.
- Der Dilationfaktor ist 1.
- Das Bild hat die Dimensionen $Batch \times channels \times height \times width$.
- Der Kernel hat die Dimensionen $out\_channels \times channels \times kernel\_height \times kernel\_width$.

b) Implementieren Sie nun den backward pass auf der Basis von numpy arrays in der Funktion h_conv2d_grad. Diese Funktion soll den Gradienten bezüglich des Bildes und des Kernels für die implementierte convolution zurückgeben.

c) Implementieren Sie die differenzierbare conv2d Funktion mit Hilfe ihrer bereits implementierten Funktionen.

d) Implementieren Sie den Conv2D Layer analog zum Linearen Layer. Hier können Sie noch einen Bias in die Convolution einfügen.

e) Nun nutzen Sie ihren implementierten Layer in der Experimentvorlage um ein CNN zu trainieren, das MNIST löst. Den Datensatz bekommen Sie unter `http://yann.lecun.com/exdb/mnist/`. Dieser Datensatz dient zur Erkennung von handgeschriebenen Zahlen.