9. Übungsblatt

Aufgabe 42 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.

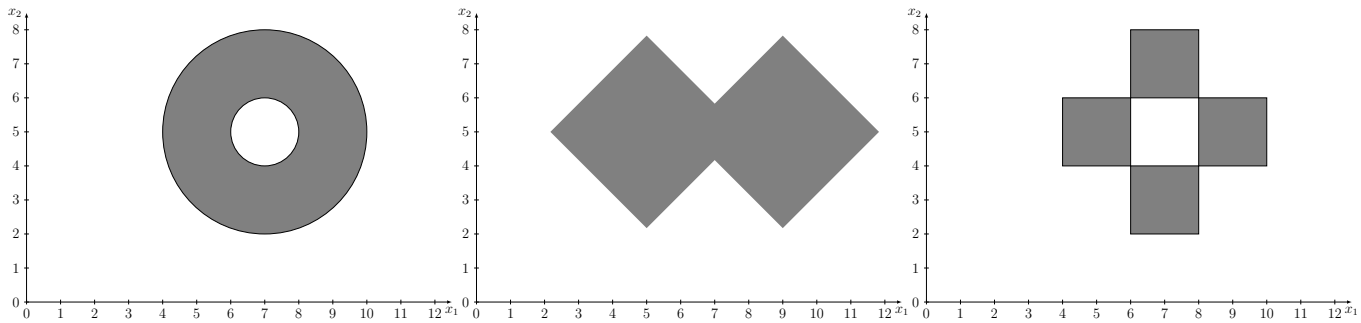
- Interpretieren Sie, was das Ergebnis des Gradientenaufstiegs zeigt.
- Betrachten Sie die Berechnung von xW. Überlegen Sie, bei welcher Architektur Sie diese Art von Berechnung bereits gesehen haben.

Aufgabe 43 Radiale-Basisfunktionen-Netze

Bestimmen Sie die Parameter (Gewichte $\vec{w}_u$ und Radien σ_u) von Radiale-Basisfunktionen-Netzen mit der Aktivierungsfunktion

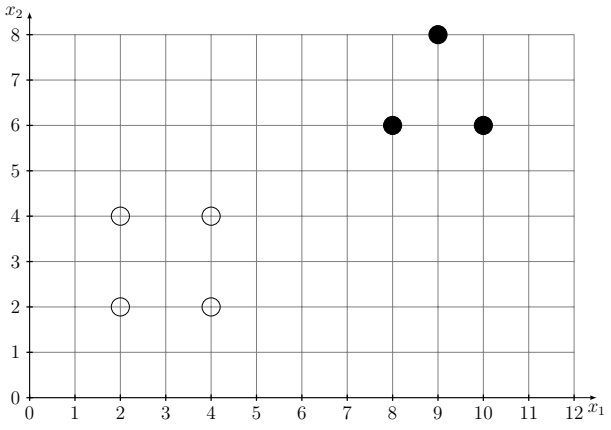
$$f_{\text{act}}^{(u)}(net_u, \sigma_u) = \begin{cases} 1, & \text{wenn } net_u \leq \sigma_u, \\ 0, & \text{sonst,} \end{cases}$$

für die Neuronen der versteckten Schicht, die für Punkte innerhalb der grauen Flächen, die in den unten gezeigten Diagrammen dargestellt sind, den Wert 1 und für Punkte außerhalb den Wert 0 liefern! Ob die Netze für Punkte auf den Rändern der Flächen den Wert 0 oder den Wert 1 liefern, ist gleichgültig. Sie sollten jedoch sicherstellen, dass für jeden Punkt der x_1-x_2-Ebene *entweder* der Wert 0 *oder* der Wert 1 berechnet wird.



Hinweis: Beachten Sie bei der Konstruktion, dass die Ausgaben überlappen und dieser Fall besonders behandelt werden muss.

Aufgabe 44 Radiale-Basisfunktionen-Netze



Die Bestimmung der Gewichte der Verbindungen von den Eingabeneuronen zu den Neuronen der versteckten Schicht — also die Bestimmung der Zentren der radialen Basisfunktionen — und die Bestimmung der Radien gehören zu den Hauptproblemen des Lernens von Radiale-Basisfunktionen-Netzen. Bei Klassifikationsaufgaben verwendet man manchmal statistische Schätzfunktionen, um geeignete (Startwerte für die) Zentren und Radien zu berechnen, jedenfalls dann, wenn zu erwarten ist, dass eine radiale Basisfunktion je Klasse ausreicht.

Man fasst dazu die radiale Basisfunktion als skalierte Wahrscheinlichkeitsdichte auf und bestimmt den Erwartungswert und die Standardabweichung der Verteilung z.B. mit einer Maximum-Likelihood-Schätzung. Als Beispiel betrachten wir ein Radiale-Basisfunktionen-Netz mit zwei Eingängen, zwei versteckten Neuronen und zwei Ausgabeneuronen, das den rechts gezeigten Datensatz klassifizieren soll. Die versteckten Neuronen verwenden den Euklidischen Abstand als Netzeingabefunktion und die Gaußfunktion $f_{\text{act}}(net, \sigma) = e^{-\frac{net^2}{2\sigma^2}}$ als Aktivierungsfunktion. Bestimmen Sie geeignete Zentren $\vec{w}$ und Radien σ für die beiden Klassen mit Hilfe einer Maximum-Likelihood-Schätzung! Was müssen Sie bei der Bestimmung der Radien beachten?

Hinweis: die üblichen Funktionen zur Berechnung des Mittelwerts und der Varianz sind Maximum-Likelihood Schätzer

Aufgabe 45 Radiale-Basisfunktionen-Netze

Bestimmen Sie die Parameter (Gewichte $\vec{w}_u$ und Biaswert θ_u) eines einfachen Radiale-Basisfunktionen-Netzes, das die Implikation $x_1 \rightarrow x_2$ berechnet! Alle Basisfunktionen sollen den Radius $\frac{3}{2}$ haben. Die versteckten Neuronen sollen den Maximumabstand als Netzeingabefunktion und eine Dreiecksfunktion

$$f_{\text{act}}(\text{net}_u, \sigma_u) = \begin{cases} 0, & \text{wenn net}_u > \sigma_u, \\ 1 - \frac{\text{net}_u}{\sigma_u}, & \text{sonst.} \end{cases}$$

als Aktivierungsfunktion besitzen.

Aufgabe 46 Bonus: Automatische Differenzierungsframework (7)

In dieser Aufgabe wollen wir ein einfaches RNN implementieren. Gehen Sie dafür wie folgt vor :

a) Implementieren Sie die concat Funktion. Diese soll eine Liste von tensoren bekommen und eine Dimension an der diese Tensoren Concatiniert werden.

$$concat([\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{pmatrix}, \begin{pmatrix}7\\ 8\end{pmatrix}], dim = 1) = \begin{pmatrix}1 & 2 & 3 & 7\\ 4 & 5 & 6 & 8\end{pmatrix}$$

Die dazugehörige Funktion finden Sie in numpy als concatenate.

b) Implementieren Sie ein RNN mit Hidden-to-Hidden-Connections, das durch die folgenden Formeln beschrieben wird.:

$$h_t = \phi(Wx_t + Uh_{t-1} + b_h)$$

$$y_t = Vh_t + b_y$$

Hier ist ϕ eine beliebige nichtlineare Aktivierungsfunktion. Beachten Sie das die forward Funktion den initialen Hidden-State als Parameter hat. Außerdem hat die Eingabe die Dimensionen $batch \times in\_channels \times t$ und der Hiddent-state $batch \times hidden\_channels$. Die Ausgabe ist wieder ein Tensor in dem alle y_t vorhanden sind. Zusätzlich sollen die Daten der Hidden States zu jedem Zeitschritt als Liste von numpy arrays zurückgegeben werden.

c) Im nächsten Schritt wollen wir unser RNN benutzen um eine kleine Aufgabe zu lösen. Nutzen Sie dafür die Experiment Vorlage. Der hier vorhandene Datensatz generiert Sequenzen aus drei Symbolen.

- 0 : Klammer auf
- 1: Klammer zu
- 2: Sonstiges

Die Zielwerte kodieren die Tiefe der Klammmern. Diesen Wert soll unser Netz vorhersagen. Trainieren Sie zunächst auf den Daten wie Sie vorgegeben sind. Sollten Sie Schwierigkeiten haben Ihr RNN zu trainieren multiplizieren Sie y mit 10 und versuchen es erneut. Warum macht diese Änderung einen merklichen Unterschied ? Experimentieren Sie mit unterschiedlichen Größen von Hiddent-States und visualisieren Sie, was im Hidden-State gespeichert wird.