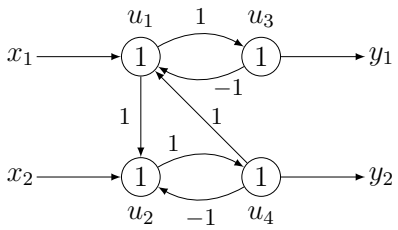
5. Übungsblatt

Aufgabe 23 Aktualisierungsreihenfolge

Gegeben sei das folgende Netz aus Schwellenwertelementen wie in *Vorlesung 2 Folie 10* :



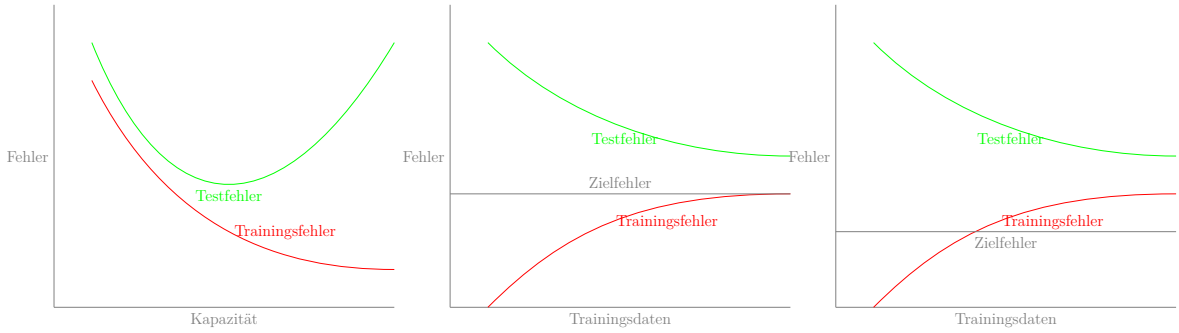
Zeigen Sie, dass es von der Aktivierungsreihenfolge der Schwellenwertelemente abhängt, ob das Netz in einen stabilen Zustand gelangt, wenn die Eingaben $x_1 = 0$ und $x_2 = 1$ angelegt werden!

Aufgabe 24 Bias und Varianz

a) Erklären Sie die Begriffe Bias und Varianz.

b) Erläutern Sie die Verbindung von Bias und Varianz zu Overfitting und Underfitting.

c) Erläutern Sie die Konzepte von Bias und Varianz an folgenden Graphen.



Aufgabe 25 Gewichtsregularisierung

Während der Optimierung gilt für eine quadratische Funktion mit einer Normregularisierung, dass

$$\min_{\theta \in \mathbb{R}^2} ||m - \theta||_2^2 + \lambda ||\theta||_\alpha$$

äquivalent zu

$$\min_{\theta \in \mathbb{R}^2} ||m - \theta||_2^2$$

$$\text{mit } ||\theta||_\alpha \leq \mu$$

im Sinne, dass für alle λ ein μ existiert.

a) Skizzieren Sie für $\alpha \in \{1, 2, \infty\}$ und $\mu = 1$, wo sich die optimalen Lösungen θ des Optimierungsproblems für beliebig $m \in \mathbb{R}^2$ befinden.

b) Erklären sie anhand der Skizzen, welche Normen zu dünn besetzten Lösungen führen.

Hinweis : Die zweite Formel bedeutet, dass die Lösung genommen wird, die im Einheitskreis bezüglich der Norm liegt und $||m - \theta||_2^2$ minimiert.

Aufgabe 26 Dropout

a) Angenommen Sie verwenden ein 10-Schichtiges Neuronales Netz mit je 10 Neuronen pro Schicht. Des Weiteren seien die Schichten untereinander voll verbunden. Wie viele Gewichte und Schwellenwerte müssen in diesem Netz insgesamt trainiert werden?
Hinweis: Bei den 10 Schichten zählt die Eingabeschicht nicht hinzu.

b) Für den Trainingsprozess soll nun die Dropout-Methode verwendet werden. Wie viele Gewichte müssen pro Trainingsdurchlauf beachtet werden, wenn durch den Dropout (durchschnittlich) 5 Neuronen pro Schicht deaktiviert werden?

c) Leiten Sie die Abhängigkeit der Anzahl zu lernender Gewichte zur durchschnittlichen Dropout-Rate her.

d) Erklären Sie die Verbindung zwischen Dropout und Bagging.

Hinweis : In der Ausgabeschicht werden keine Neuronen ausgelassen. Verwenden Sie folgende Bezeichnungen für Variablen

- n_i Anzahl der Neuronen in Schicht i
- d Anteil an Neuronen, die in jeder Schicht durch Dropout deaktiviert wird
- L Anzahl der Schichten

Aufgabe 27 Bonus: Automatische Differenzierungsframework (3)

Dieses Mal wollen wir anfangen die implementierten Funktionen zu nutzen um ein MLP zu erstellen. Danach wollen wir kleine Experimente mit unserem Framework machen.

a) Implementation einer Klasse, die eine lineare Schicht für eine Batch berechnet und verwaltet. Dafür sollen folgende Funktionalitäten erfüllt werden:

- ___init___(self, in_features, out_features) : Erstellen der Tensoren für die Berechnung. Eine typische Initialisierung könnt ihr unter `http://proceedings.mlr.press/v9/glorot10a.html` finden.
- ___call___(self, X) : Nimm einen Tensor $\mathbb{R}^{N \times C}$ entgegen und berechne $WX + b$.
- parameters( ) : Gebe eine Liste von trainierbaren Parametern zurück.

b) Mache dir das trainieren etwas einfacher mit einer Optimizer Klasse:

- ___init___(self, parameters, learning_rate) : Merke dir alle Parameter.

- step( ) : Passe alle Parameter mit der Lernrate an.
- zero_grad( ) : Setze die Gradienten der Parameter wieder auf 0.

c) Erstellen von nicht-linearen Funktionen:

- sigmoid
- relu
- leaky_relu

d) Zu guter Letzt wollen wir die implementierte Funktionalität nutzen um eine Funktion zu lernen. Die Daten für die Funktion findet ihr wieder im jupyter-notebook. Hier solltet ihr euren Layer für mehrere Epochen mit dem Optimizer trainieren. Zusätzlich könnt ihr die Tiefe und Breite variieren oder unterschiedliche Aktivierungsfunktionen wählen. Dieser Prozess besteht aus folgenden Schritten :

i) Initialisiert die Layer eures Models (Klasse oder als globale Variablen)

ii) Implementiert eine Funktion, die eure Layer in der richtigen Reihenfolge aufrufen

iii) Initialisiert euren Optimizer, der alle Parameter der Layer erhält.

iv) Implementiert eure Fehlerfunktion (Mean-Squared-Error, Mean-Absolute-Error)

v) Erstellt eine Trainingsschleife , die immer wieder get_batch ruft, die Vorhersage des Models berechnet und den Fehler bezüglich der Labels bestimmt. Anschließend bestimmt den Gradient und benutzt den Optimizer für den Gradientenschritt. Setzt am Ende jedes Schrittes den Gradienten wieder auf 0.