Aufgabenblatt Deep Learning I

Aufgabe 1: Logistische Regression als ANN

Zeichnen Sie ein logistisches Regressionsmodell mit zwei Input Variablen x_1 und x_2 als Artificial Neural Network (ANN). Wie viele Hidden Layers hat das ANN? Welche Aktivierungsfunktion wählen Sie im Output Layer?

Aufgabe 2: ANN ohne Aktivierungsfunktionen

In dieser Aufgabe schauen wir uns an, was passiert, wenn wir keine Aktivierungsfunktion verwenden. Wir verwenden dazu folgendes ANN:

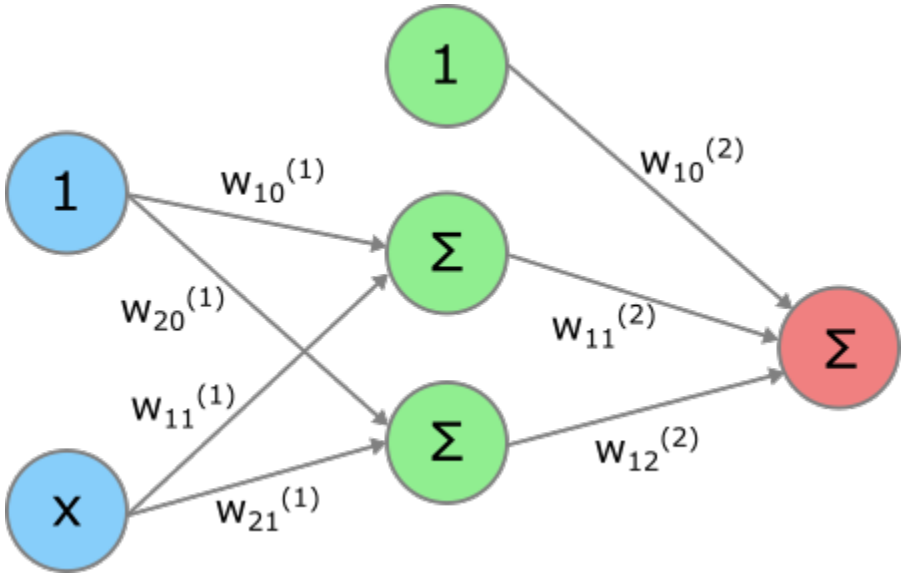

Figure 1: ANN ohne Aktivierungsfunktion

Berechnen Sie in einem ersten Schritt die gewichteten Summen in den beiden Neurons im Hidden Layer. Tipp: im ersten Neuron im Hidden Layer beträgt die gewichtete Summe $z_1^{(1)} = w_{10}^{(1)} + w_{11}^{(1)} \cdot x$. Verwenden Sie dann die berechneten Resultate, um den Output dieses ANNs zu rechnen. Wichtig: das ist hier relativ einfach, da wir eben keine Aktivierungsfunktionen verwenden und die gewichteten Summen aus dem Hidden Layer direkt dem Output Layer gefüttert werden. Wie sieht der Output dieses ANNs grafisch aus?

Aufgabe 3: Log-Loss

Beim Klassifikationsproblem versuchen wir den sogenannten Log-Loss zu minimieren. Die Log-Loss Funktion sieht folgendermassen aus:

$$J(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)]$$

3.a) Ein ANN hat Ihnen für 5 Beobachtungen die Vorhersagen $\hat{y}_i$ berechnet. Die Vorhersagen sowie die tatsächlichen Werte y_i sind wie folgt:

y_i	$\hat{y}_i$
0	0.12
0	0.33
1	0.65
1	0.88
1	0.95

Berechnen Sie den Wert der Log-Loss Funktion für diese 5 Beobachtungen.

3.b) Nun haben wir 5 andere Beobachtungen:

y_i	$\hat{y}_i$
0	0.0
0	0.0
1	1.0
1	1.0
1	1.0

Berechnen Sie wiederum den Wert der Log-Loss Funktion für diese 5 Beobachtungen.

Aufgabe 4: Ableitungen

Wir haben gelernt, dass der Gradient Descent Algorithmus die erste Ableitung verwendet, um einen Schritt in die entgegengesetzte Richtung zu machen. Es ist also wichtig, dass Sie Ableitungen von Funktionen berechnen können.

Berechnen Sie die Ableitung nach x für folgende Funktionen.

- $f(x) = 3x^2$
- $f(x) = 16x + 5$
- $f(x) = x^2 + 2x$
- $f(x) = 15$
- $f(x) = e^{2x}$
- $f(x) = (16x + 24)^3$
- $f(x) = w_0 + w_1 \cdot x$

Aufgabe 5: Gradient Descent für die einfache lineare Regression

In dieser Aufgabe wollen wir den Gradient Descent Algorithmus für das einfache lineare Regressionsproblem anwenden. Der Modelloutput eines einfachen linearen Regressionsmodell lässt sich mit der sogenannten Prognosegleichung $\hat{y}_i = b_0 + b_1 \cdot x_i$ rechnen.

5.a) Ersetzen Sie $\hat{y}_i$ in der *Mean Squared Error* Loss Funktion mit obiger Prognosegleichung und rechnen Sie danach die Ableitung nach b_0 und die Ableitung nach b_1.

5.b) Wir implementieren nun den Gradient Descent Algorithmus für das einfache lineare Regressionsproblem. Und zwar schauen wir uns den für einige von Ihnen altbekannten `housingrents` Datensatz an. Wir rechnen ein einfaches Regressionsmodell mit $y = rent$ und $x = area$. Verwenden Sie für die Implementation des Gradient Descent Algorithmus den unten stehenden R-Code und ergänzen Sie den Code an den vorgesehenen Stellen.

```r
# Lade den housingrents Datensatz
df <- read.csv("housingrents.csv", sep = ";", header = TRUE)

# Wir speichern die Zielvariable und die erklärende Variable als y und x.
# Wichtig: die x-Variable wird mit der Funktion scale() standardisiert.
y <- df$rent
x <- scale(df$area)

# Wir speichern die Anzahl Beobachtungen als n.
n <- nrow(df)

# Die beiden Parameter für den Gradient Descent Algorithmus.
learning_rate <- 0.01
n_iterations <- 1000

# Wir initialisieren die beiden Parameter mit einem
# zufälligen Wert aus der Standardnormalverteilung.
b0 <- rnorm(1)
b1 <- rnorm(1)

# Gradient Descent
for (i in 1:n_iterations) {
  # Berechnung der aktuellen Vorhersagen des Modells
  y_hat <- b0 + b1 * x
  # Gradient Descent Schritt
  b0 <- "IHR CODE"
  b1 <- "IHR CODE"
}

# Wir geben die finalen Parameterwerte aus.
sprintf("Optimales b0 ist %f und optimales b1 ist %f", b0, b1)
```

Optional: Speichern Sie nach jedem Gradient Descent Schritt den Loss und plotten Sie den Verlauf des Loss am Schluss des Codes.

5.c) Rechnen Sie nun das lineare Regressionsmodell mit der bekannten `lm()` Funktion. Denken Sie daran, dass Sie auch hier die x-Variable standardisieren mit der Funktion `scale()`. Vergleichen Sie Ihr Resultat mit dem Resultat aus der vorherigen Aufgabe.