

# Künstliche Intelligenz - Übung 7

Adrian Neumann (4140810) und Naja von Schmude (4127652)

12. Juni 2009

## 1 Das Neuronale Netz

Zunächst wurde der Backpropagation mit Online und Batchverfahren zum lernen implementiert.

```
private Matrix[] backpropagation(ObjectToClassify otc) {
    Matrix t = new Matrix(1, m);
    t.set(0, otc.getLabel(), 1);

    // FEED FORWARD

    // Ausgabe Input-Layer
    Matrix o = new Matrix(otc.getFeatures(), 1);
    Matrix o_bar = new Matrix(1, n + 1);
    o_bar.setMatrix(0, 0, 0, n - 1, o);
    o_bar.set(0, n, 1);

    // Ausgabe Hidden-Layer
    Matrix o_1 = o_bar.times(W_1_bar);
    for (int i = 0; i < k; i++) {
        double s = sigmoid(o_1.get(0, i));
        o_1.set(0, i, s);
    }
    Matrix o_1_bar = new Matrix(1, k + 1);
    o_1_bar.setMatrix(0, 0, 0, k - 1, o_1);
    o_1_bar.set(0, k, 1);

    // Ausgabe Output-Layer
    Matrix o_2 = o_1_bar.times(W_2_bar);
    for (int i = 0; i < m; i++) {
        double s = sigmoid(o_2.get(0, i));
        o_2.set(0, i, s);
    }

    // Ableitungen
    Matrix D_2 = new Matrix(m, m);
    for (int i = 0; i < m; i++) {
        double d = o_2.get(0, i) * (1 - o_2.get(0, i));
        D_2.set(i, i, d);
    }

    Matrix D_1 = new Matrix(k, k);
    for (int i = 0; i < k; i++) {
        double d = o_1.get(0, i) * (1 - o_1.get(0, i));
        D_1.set(i, i, d);
    }

    Matrix e = o_2.minus(t);

    // Fehler.
    Matrix errorV = new Matrix(1, m);
    for (int i = 0; i < m; i++) {
        errorV.set(0, i, ((o_2.get(0, i) - t.get(0, i))
```

```

        * (o_2.get(0, i) - t.get(0, i)) / 2));
    }
    double errorSum = 0;
    for (int i = 0; i < m; i++) {
        errorSum += errorV.get(0, i);
    }

    error.add(errorSum);

    // BACKPROPAGATION

    Matrix delta_2 = e.times(D_2);
    Matrix delta_1 = delta_2.times(W_2.transpose()).times(D_1);
    Matrix delta_W_2_bar = o_1_bar.transpose().times(delta_2);
    Matrix delta_W_1_bar = o_bar.transpose().times(delta_1);

    return new Matrix[] { delta_W_1_bar, delta_W_2_bar };
}

```

Der Rückgabewert enthält dann die Änderungen für die Gewichtsmatrizen  $\bar{W}_1$  und  $\bar{W}_2$ , die durch das Testen des übergebenen Datums berechnet werden. Es ist dabei aber zu beachten, dass die Lernkonstante nicht hier dazu multipliziert wird, sondern an anderer Stelle, nämlich im Batch- oder Onlineverfahren. Dort werden die Lernkonstanten nämlich mittels Silva-Almeida angepasst. D.h. dass die Lernkonstante vergrößert wird, wenn man einen Wert bei zwei aufeinanderfolgenden Iterationen in die selbe Richtung korrigiert, und ansonsten sie verringert.

```

private void onlineLearning() {
    Matrix[] deltas_Old = new Matrix[] { new Matrix(n + 1, k),
        new Matrix(k + 1, m) };
    Matrix[] deltas;
    Matrix[] lambdas = new Matrix[] { new Matrix(n + 1, k, 1),
        new Matrix(k + 1, m, 1) };
    boolean go = true;
    Random rand = new Random();

    while (go) {
        // Gewichtsänderungen berechnen
        deltas = backpropagation(trainingsdata.get(rand
            .nextInt(trainingsdata.size())));
        lambdas = calculateLearningFactors(deltas_Old, deltas, lambdas);

        for (int m = 0; m < deltas.length; m++) {
            for (int i = 0; i < deltas[m].getRowDimension(); i++) {
                for (int j = 0; j < deltas[m].getColumnDimension(); j++) {
                    deltas[m].set(i, j, (-1) * lambdas[m].get(i, j)
                        * deltas[m].get(i, j));
                }
            }
        }
    }

    // Gewichte updaten
    W_1_bar.plusEquals(deltas[0]);
    W_2_bar.plusEquals(deltas[1]);
    W_2 = W_2_bar.getMatrix(0, k - 1, 0, m - 1);
    deltas_Old = deltas;

    System.out.println(error.avg());

    // Gucken, ob man genügend richtig klassifiziert ...
    go = false;
    for (int i = 0; i < 15; i++) {
        ObjectToClassify otc = trainingsdata.get(rand
            .nextInt(trainingsdata.size()));
        int classify = classify(otc);
    }
}

```

```

        if (classify != otc.getLabel()) {
            // System.out
            // .println("Noch nicht gut genug ... Weitermachen!");
            go = true;
            break;
        }
    }
}

System.out.println("Netz online fertig trainiert!");
}

```

Beim Online-Verfahren nimmt man solange man noch nicht gut genug ist (hier ist gut genug dadurch definiert, dass man es fehlerfrei schaffen muss 20 Daten aus der Trainingsdatenbank richtig zu klassifizieren) ein zufälliges Datum aus der Testdatenbank, rechnet die Änderungen und aktualisiert entsprechend die Gewichte. Eine andere Möglichkeit wäre sicherlich zu prüfen, ob der durchschnittliche Fehler kleiner als eine gewisse Grenze ist ...

```

private void batchLearning() {
    Matrix[] deltas_Old = new Matrix[] { new Matrix(n + 1, k),
        new Matrix(k + 1, m) };
    Matrix[] deltas_Sum = new Matrix[] { new Matrix(n + 1, k),
        new Matrix(k + 1, m) };
    Matrix[] deltas = null;
    Matrix[] lambdas = new Matrix[] { new Matrix(n + 1, k, 0.1),
        new Matrix(k + 1, m, 0.1) };
    boolean go = true;
    Random rand = new Random();

    while (go) {
        // Gewichtsänderungen berechnen für alle Trainingsdaten
        for (ObjectToClassify otc : trainingsdata) {
            deltas = backpropagation(otc);
            lambdas = calculateLearningFactors(deltas_Old, deltas, lambdas);

            for (int m = 0; m < deltas.length; m++) {
                for (int i = 0; i < deltas[m].getRowDimension(); i++) {
                    for (int j = 0; j < deltas[m].getColumnDimension(); j++) {
                        deltas[m].set(i, j, (-1) * lambdas[m].get(i, j)
                            * deltas[m].get(i, j));
                    }
                }
                deltas_Sum[m].plusEquals(deltas[m]);
            }

            deltas_Old = deltas;
        }

        // Gewichte updaten
        W_1_bar.plusEquals(deltas_Sum[0]);
        W_2_bar.plusEquals(deltas_Sum[1]);
        W_2 = W_2_bar.getMatrix(0, k - 1, 0, m - 1);

        System.out.println(error.avg());

        // Gucken, ob man genügend richtig klassifiziert ...
        go = false;
        for (int i = 0; i < 5; i++) {
            ObjectToClassify otc = trainingsdata.get(rand
                .nextInt(trainingsdata.size()));
            int classify = classify(otc);
            if (classify != otc.getLabel()) {
                // System.out
                // .println("Noch nicht gut genug ... Weitermachen!");
                go = true;
            }
        }
    }
}

```

```

        break;
    }
}

System.out.println("Netz_online_fertig_trainiert!");
}

```

Die Batch-Methode unterscheidet sich dahingehend, dass bevor die Änderungen auf die aktuellen Gewichtsmatrizen übertragen werden, erstmal alle Änderungen für jedes Datum der Trainingsdatenbank akkumuliert werden. Das Abbruchkriterium bleibt das selbe. Das Batch-Verfahren ist natürlich wesentlich langsamer.

## Aufgabe 1

XOR lässt sich ziemlich gut mit den oben beschriebenen Methoden klassifizieren. Dabei ist erkennbar gewesen, dass man mit nur zwei Neuronen in der verdeckten Schicht maximal eine Erkennungsrate von 50% erhält. Bei allen drüber liegenden Werten waren in unseren Test 75 oder 100 % immer erreichbar. Aufgrund der kleinen Datenmenge ist das ganze auch extrem fix, obwohl man schon erkennt, dass bei mehr Neuronen es etwas länger braucht, aber auch nicht merklich. Zudem ist es extrem von den initialisierten Gewichten abhängig, die ja jedesmal zufällig gesetzt werden. Wir haben festgestellt, dass das Batchverfahren in diesem Fall schneller geht, aber auch meistens etwas schlechtere Erkennungsraten als die Onlinevariante hat.

## Aufgabe 2

Es muss angemerkt werden, dass sich der Datensatz nicht wirklich gut klassifizieren lässt, aufgrund der Tatsache, dass über 50 % der Daten einer Klasse angehören, und somit die Datensätze nicht gleichverteilt sind. Sogar mit KNN haben wir keine Erkennungsrate über 60% erhalten! Das neuronale Netz funktionierte dann auch nur wirklich im Onlinemodus, das Batchverfahren terminierte überhaupt nicht (oder dauerte uns zu lange ;-). Das Ergebnis des Onlineverfahrens war auch stets das selbe. Und zwar wurden alle Datensätze aus Klasse 0 perfekt erkannt, dafür kein anderes. Das liegt an der oben genannten ungleichen Verteilung. Immerhin hat man so immer eine Erkennungsrate über 50% gehabt. Als eine gute Anzahl an Neuronen für die verdeckte Schicht hat sich 12/13 herausgestellt. Dabei ging das Trainieren auch recht fix.