Lernverfahren autonomer Roboter - Übung 5

Tobias Hahn
3073375

December 3, 2016

Übung 5

1 Nearest Neighbour Classifier

1.1 Distanzmetriken

Für den Klassifizierer ist es notwendig eine Metrik einzuführen die die Distanz zwischen zwei Punkten als Zahl ermittelt und damit vergleichbar macht. Die drei für die Implementierung verwendeten Distanzmetriken werden im folgenden vorgestellt.

1.1.1 Normalisierte euklidische Distanz

Für die normalisierte euklidische Distanz werden die Quadrate der Abstände zwischen den beiden Beispielvektoren für jede Dimension aufsummiert, wobei die Abstände jeweils noch duch die Varianz der Beispiele in dieser Dimension dividiert werden (daher normalisiert). Die Wurzel dieser Summe ist die Distanz. Für das Beispiel wurde die Varianz aus den Beispielen in den Trainingsdaten berechnet und zur Berechnung herangezogen, denkbar ist auch ein Ansatz der dies aus den Testdaten generiert. Jedoch sollten gerade die Testdaten anhand der Trainingsdaten klassifiziert werden, wodurch auch dieser Wert aus den Trainingsdaten kommen sollte.

Relevanter Teil des Codes:

Normalisierte euklidische Distanz

```
1    (In der Hauptmethode des Klassifizierers)
2    sds = np.std(X_train, axis=1)
3    distances = [metric(data, compare, sds=sds) for compare in X_train]
4
5    (Die Distanzmetrik selber)
6    def normalized_euclidean_distance(self, data_a, data_b, **kwargs):
7        return np.sqrt(np.sum([(a - b)**2 / sd**2 for a, b, sd in zip(data_a, data_b, kwargs
             ['sds'])]))
```

1.1.2 Chebyshev Distanz

Eine andere Metrik zur Berechnung der Distanz, welche auch in der Vorlesung vorgestellt wurde, ist die Chebyshev Distanz. Hierbei werden die Beträge der Distanzen der beiden Vektoren betrachtet, wobei sich die Distanz aus der größten Distanz ergibt, also den Abstand der beiden Vektoren in der Dimension darstellt in der dieser am größten ist.

Relevanter Teil des Codes:

Chebyshev Distanz

```
1    def chebyshev_distance(self, data_a, data_b, **kwargs):
2        return max([abs(a - b) for a, b in zip(data_a, data_b)])
```

1.1.3 Cityblock Distanz

Die Cityblockdistanz wurde in der Vorlesung nicht vorgestellt, kann jedoch hier[1] nachgelesen werden. Hierbei werden die Beträge der Abstände zwischen zwei Vektoren für jede Dimension aufsummiert.

Relevanter Teil des Codes:

Cityblock Distanz

```
1    def cityblock_distance(self, data_a, data_b, **kwargs):
2        return np.sum([abs(a - b) for a, b in zip(data_a, data_b)])
```

[1] https://docs.tibco.com/pub/spotfire/6.5.2/doc/html/hc/hc_city_block_distance.htm

1.2 Accuracy

Der implementierte Klassifizierer wurde am bekannten Irisdatenset geprüft, wobei der Parameter für die Anzahl der betrachteten Nachbarn von 1 bis 100 variiert wurde. Dieser Parameter gibt an die wieviel nächsten Punkte des betrachteten Punktes zur Klassifizierung verwendet werden, wobei die Klassenlabel betrachtet werden. Das Klassenlabel das in den Daten am öftesten vorkommt wird verwendet. Die Nähe wird dabei durch die normalisierte euklidische Distanz angegeben.

1.2.1 Plot

Die Resultate sind in folgendem Plot zu sehen:

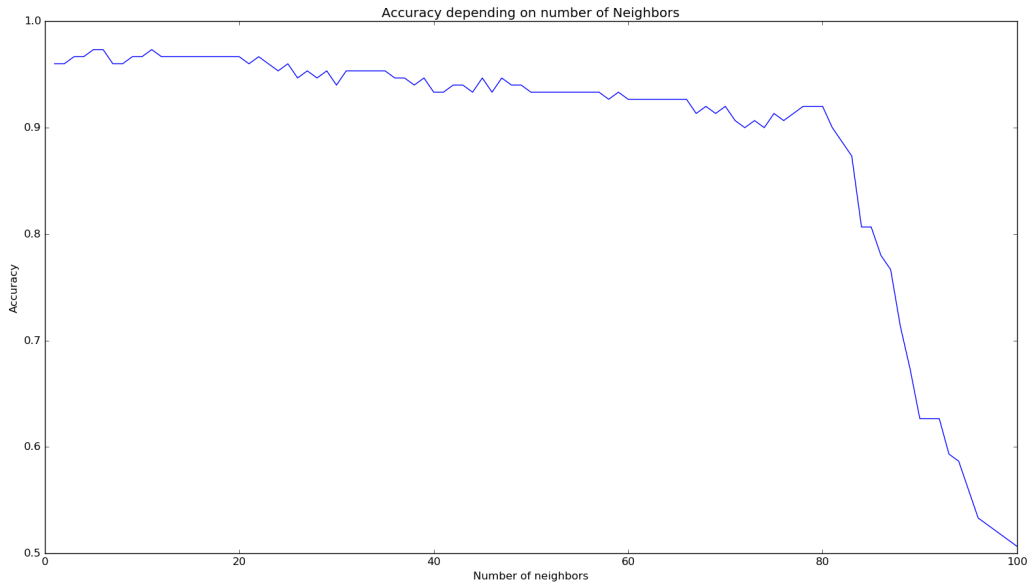

Figure 1: Die Accuracy zu der verwendeten Anzahl von Nachbarn

1.2.2 Erklärung

Wie man sehen kann bleibt die Accuracy bis zu einer gewissen hohen Anzahl an Nachbarpunkten (80+) relativ stabil, wobei das Maximum bei 5 erreicht ist und danach ein leichter Abstieg stattfindet. Über 80 nimmt die Accuracy rapide ab, bis sie bei 100 Nachbarn bei annähernd 0.5 landet.

Die relative Stabilität bis 80 lässt sich dadurch erklären, dass die Punkte der drei Labelcluster relativ gut getrennt sind. Daher bleiben auch die Nachbarn mit richtigem Label in der Tendenz in der Mehrzahl, wenn man mehr Nachbarn betrachtet. Der rapide Abfall nach 80 ist dadurch erklärbar, dass bei so einer hohen Anzahl an Nachbarn die Wahrscheinlichkeit dass ein Punkt als Nachar in Betracht gezogen wird fast Unabhängig von seiner Entfernung wird, die Häufigkeit der Label in der Nachbarschaft sich der Verteilung der Label in der Grunddatenmenge annähert. Dadurch wird die Klassifizierung mehr zu einer Aussage darüber welches Label am öftesten in der Datenmenge vorkommt, anstatt etwas zur Klassenzugehörigkeit zu sagen: Kleiner Cluster haben gar keine Mitglieder mehr, während größere Cluster viel zu viele haben.

1.3 Andere Distanzmetriken

Der gleiche Versuch wurde auch mit anderen Distanzmetriken gestartet. Die Versuche ähneln sich sehr stark, wie man an den folgenden beiden Plots sieht:

1.3.1 Plot

Die Resultate sind in folgenden Plots zu sehen:

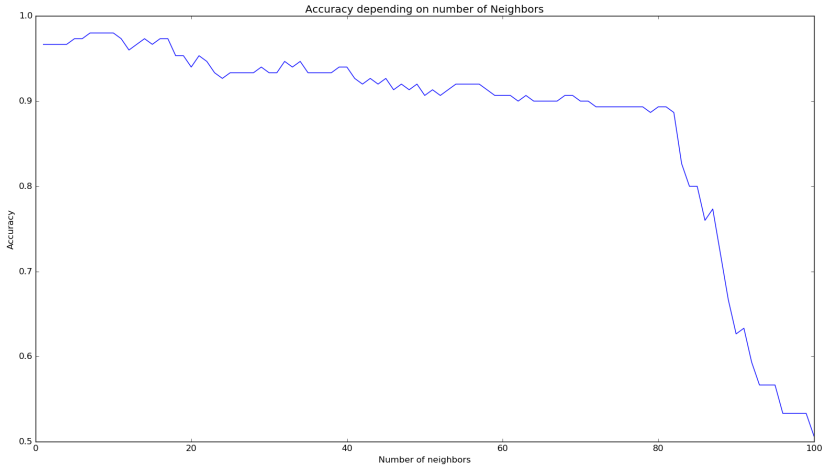

Figure 2: Die Accuracy zu der verwendeten Anzahl von Nachbarn mit Chebyshev Distanz

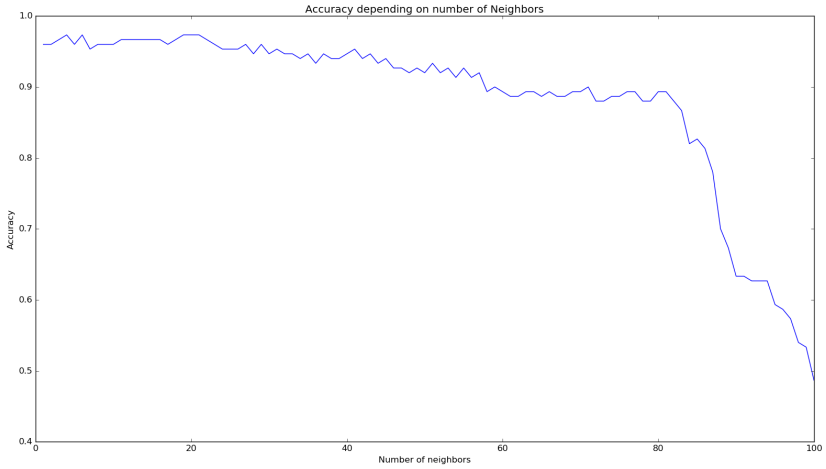

Figure 3: Die Accuracy zu der verwendeten Anzahl von Nachbarn mit Cityblock Distanz

1.3.2 Erklärung

Wie man sieht ist die Accuracy ziemlich identisch. Dies liegt daran dass alle drei Distanzmetriken auf dem Abstand der Punkte im Raum abhängen, und daher die ähnlichen Argumente wie bei Punkt Zwei zutreffen.