Neuronale Netze - Übung 6

Tobias Hahn
3073375

November 28, 2016

1 Erwartungswert & Hauptkomponenten

1.1 Verständnisfrage

Um herauszubekommen wie viele Punkte gebraucht werden müssen wir zuerst die Anzahl der Regionen ausrechnen, die man mit zwei Hyperebenen abtrennen kann. Die Formel dafür ist (m ist die Anzahl der Hyperebenen, n ist die Dimensionalität der Daten)

$$\text{Anzahl Regionen} = 2 * \sum_{i=0}^{n-1} (\binom{m-1}{i})$$

Für unser Beispiel beträgt die Anzahl der Regionen 4. Jedoch können wir die Aufgabenstellung noch schwieriger machen, indem wir die Punkte einfach auf einer Linie, also in einer Dimension anordnen. Dies ist erlaubt, da wir beliebige Punkte im R2 nehmen können. Dann beträgt die Anzahl der Regionen die wir abtrennen können 3. Wir brauchen dann vier Punkte um sie so anordnen zu können dass sie nicht mehr trennbar sind, auf einer Linie in folgender Abfolge angeordnet: Schwarz, Weiß, Schwarz, Weiß.

1.2 Programmieraufgabe

Die Hauptkomponenten wurden in einem Netz mit Ojas Algorithmus berechnet. Der Code und die Ergebnisse hier:

Im folgenden zuerst der Quellcode für die beiden Klassen, danach zwei Bilder der Ergebnisse.

../meta.py

```
1  import numpy as np
2
3  def erwartung(digits):
4      num_digits = len(set(digits['value']))
5      erwartungswerte = np.zeros((num_digits,len(digits['data'][0])))
6      frequency = np.zeros(num_digits)
7
8      for d in digits:
9          erwartungswerte[d['value']] += d['data']
10         frequency[d['value']] += 1
11
12     for i,e in enumerate(erwartungswerte):
13         erwartungswerte[i] = e / frequency[i]
14
15     return erwartungswerte
16
17 def hauptkomponenten(digits, amount):
18     komponenten = np.random.rand(len(set(digits['value'])),amount,len(digits['data'][0]))
19     lernrate = 1
20     amount = len(digits)
21     for c in komponenten:
22         for k in c:
23             k /= np.linalg.norm(k)
24
25     while (lernrate > 0):
26         example = digits[np.random.randint(0,amount)]
27         example_class = example['value']
28         example_data = example['data']
29         for i,k in enumerate(komponenten[example_class]):
30             scalar = np.dot(example_data,k)
31             komponenten[example_class][i] = k + lernrate * scalar * (example_data - scalar * k)
32             komponenten[example_class][i] /= np.linalg.norm(komponenten[example_class][i])
33             example_data -= scalar * komponenten[example_class][i]
34         lernrate -= 0.001
35
36     return komponenten
```

../train.py

```
1  import numpy as np
2  import meta as m
3  from matplotlib import pyplot as plt
4
5  def extractDigits(filename, expected_num):
6      data_count = 0
7      digit_count = 0
8      data_points_per_digit = 192
9      data_points_per_line = 12
10
11     digits = np.zeros(expected_num, dtype=[('data', 'f', data_points_per_digit), ('value', 'i
           ')])
12
13     with open(filename) as f:
14         lines = f.readlines()
15
16     for i,line in enumerate(lines):
17         digits_line = line.split()
18         if (len(digits_line) == data_points_per_line):
19             for num in digits_line:
20                 digits['data'][digit_count][data_count] = float(num)
21                 data_count += 1
22         elif (len(digits_line) == 10):
23             for i,num in enumerate(digits_line):
24                 if (num == "1.0"):
25                     digits['value'][digit_count] = i
26                     break
27         else:
28             if (data_count == data_points_per_digit and digit_count < expected_num):
29                 digit_count += 1
30                 data_count = 0
31             else:
32                 print("Exited because of wrong data")
33                 raise SystemExit
34
35     if (digit_count == expected_num):
36         return digits
37     else:
38         print("Exited because of few digits")
39         raise SystemExit
40
41 if __name__ == "__main__":
42     name = "./data/digits.trn"
43     number = 1000
44     digits = extractDigits(name, number)
45
46     erwartungswerte = m.erwartung(digits)
47     komponenten = m.hauptkomponenten(digits, 9)
48
49     plt.gray()
50     for i,e in enumerate(erwartungswerte):
51         plt.subplot(10,len(erwartungswerte),i+1)
52         plt.tick_params(axis='both', which='both', bottom='off', top='off', labelbottom='off'
               , right='off', left='off', labelleft='off')
53         plt.imshow(np.reshape(e, (16,12)))
54         for j,k in enumerate(komponenten[i]):
55             plt.subplot(10,len(erwartungswerte),(j+1)*10+i+1)
56             plt.tick_params(axis='both', which='both', bottom='off', top='off', labelbottom='
                   off', right='off', left='off', labelleft='off')
57             plt.imshow(np.reshape(k, (16,12)))
58     plt.show()
59
60     for d in digits:
61         d['value'] = 0
62
63     erwartungswert = m.erwartung(digits)[0]
64     komponenten = m.hauptkomponenten(digits, 15)[0]
65
66     plt.gray()
```

```
67      plt.subplot(1,16,1)
68      plt.tick_params(axis='both',  which='both', bottom='off', top='off', labelbottom='off',
            right='off', left='off', labelleft='off')
69      plt.imshow(np.reshape(erwartungswert, (16,12)))
70      for i,k in enumerate(komponenten):
71          plt.subplot(1,16,2+i)
72          plt.tick_params(axis='both',  which='both', bottom='off', top='off', labelbottom='off'
                , right='off', left='off', labelleft='off')
73          plt.imshow(np.reshape(k, (16,12)))
74      plt.show()
```

Figure 1: Für alle Ziffernklassen: Oben Erwartungswert, danach erste bis neunte Hauptkomponente

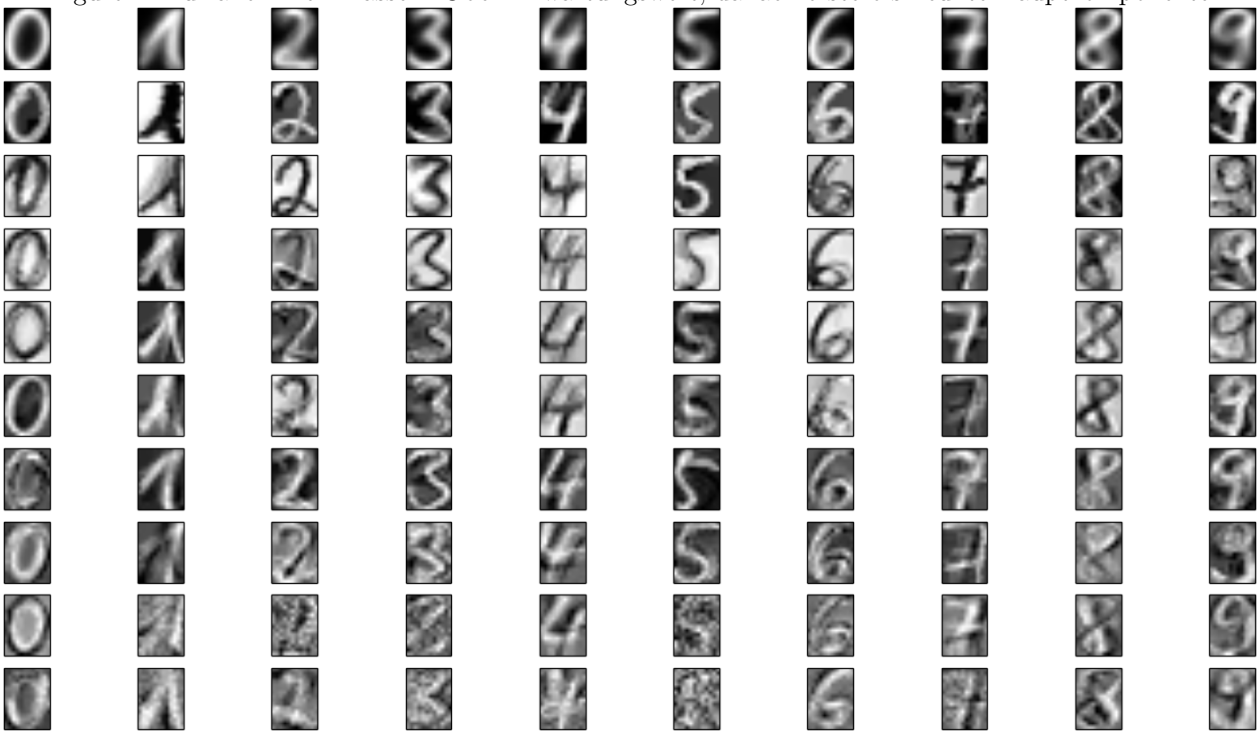

Figure 2: Für alle Ziffern: Links Erwartungswert, danach erste bis fünfzehnte Hauptkomponente