Exponentielle Regression

Projektpraktikum Serie 4

Tom Lambert Yuuma Odaka

January 22, 2018

Übersicht

Einführung in die Theorie

Einführung in die Theorie

Exponentielle Regression

Eine möglichst genaue Approximation an einem Datensatz mithilfe eines Exponentialfunktions, meistens in der Form $ae^{bx} + c$

Einführung in die Theorie

Exponentielle Regression

Eine möglichst genaue Approximation an einem Datensatz mithilfe eines Exponentialfunktions, meistens in der Form $ae^{bx} + c$

- In diesem Fall: $ae^{dx} + be^{-dx} + c$

Programm und Implementierung

Festlegen der Parameter

$$d_k^{(n)} := 0.1 + 0.4k/(n-1) \in [0.1, 0.5]$$
$$k = 0, \ldots, n-1$$

Festlegen der Parameter

$$d_k^{(n)} := 0.1 + 0.4k/(n-1) \in [0.1, 0.5]$$
$$k = 0, \ldots, n-1$$

```
def main(file_name="data.txt", n=7):
    # Code
    pass

if __name__ == "__main__":
    main(file_name="data_sym.txt", n=7)
```

Datei einlesen

```
def read_data(file_name):
  file_object = open(file_name, "r")
  result = []
  for line in file_object.readlines():
    parts = map(lambda part: float(part.strip()),
                line.split(","))
    if len(parts) != 2:
      raise Exception("illegal line format")
    result.append((parts[0], parts[1]))
  file_object.close()
  return result
```

Numpy/Python Objekte erzeugen

```
# data_list = [(x, y), ...]

b = np.matrix(map(lambda pair:
                  [pair[1]], data_list))
a = np.matrix(map(lambda pair:
                  [e**(d * pair[0]),
                   e**(-d * pair[0]),
                   1], data_list))
```

QR Zerlegung und lösen des Gleichungssystems

```
q, r = np.linalg.qr(a)
if r_rank(r) != 3:
    print("Rank of r or q is not 3!")
else:
    # z = q^T * b; rx = z
    z = np.dot(q.T, b)
    x = scipy.linalg.solve_triangular(r, z)
```

QR Zerlegung und lösen des Gleichungssystems

```
q, r = np.linalg.qr(a)
if r_rank(r) != 3:
    print("Rank of r or q is not 3!")
else:
    # z = q^T * b; rx = z
    z = np.dot(q.T, b)
    x = scipy.linalg.solve_triangular(r, z)
```

```
def r_rank(r):
    counter = 0
    eps = 10**-12
    for i in range(0, r.shape[0]):
        if not (-eps < r[i, i] < eps):
            counter += 1
    return counter
```

Berechnung des Residuums, Norm und Kondition

```
r = np.dot(a, x) - b

norm_r = np.linalg.norm(r)

cond_a = np.linalg.cond(a)
cond_ata = np.linalg.cond(np.dot(a.T, a))
```

Berechnung des Residuums, Norm und Kondition

```
r = np.dot(a, x) - b

norm_r = np.linalg.norm(r)

cond_a = np.linalg.cond(a)
cond_ata = np.linalg.cond(np.dot(a.T, a))
```

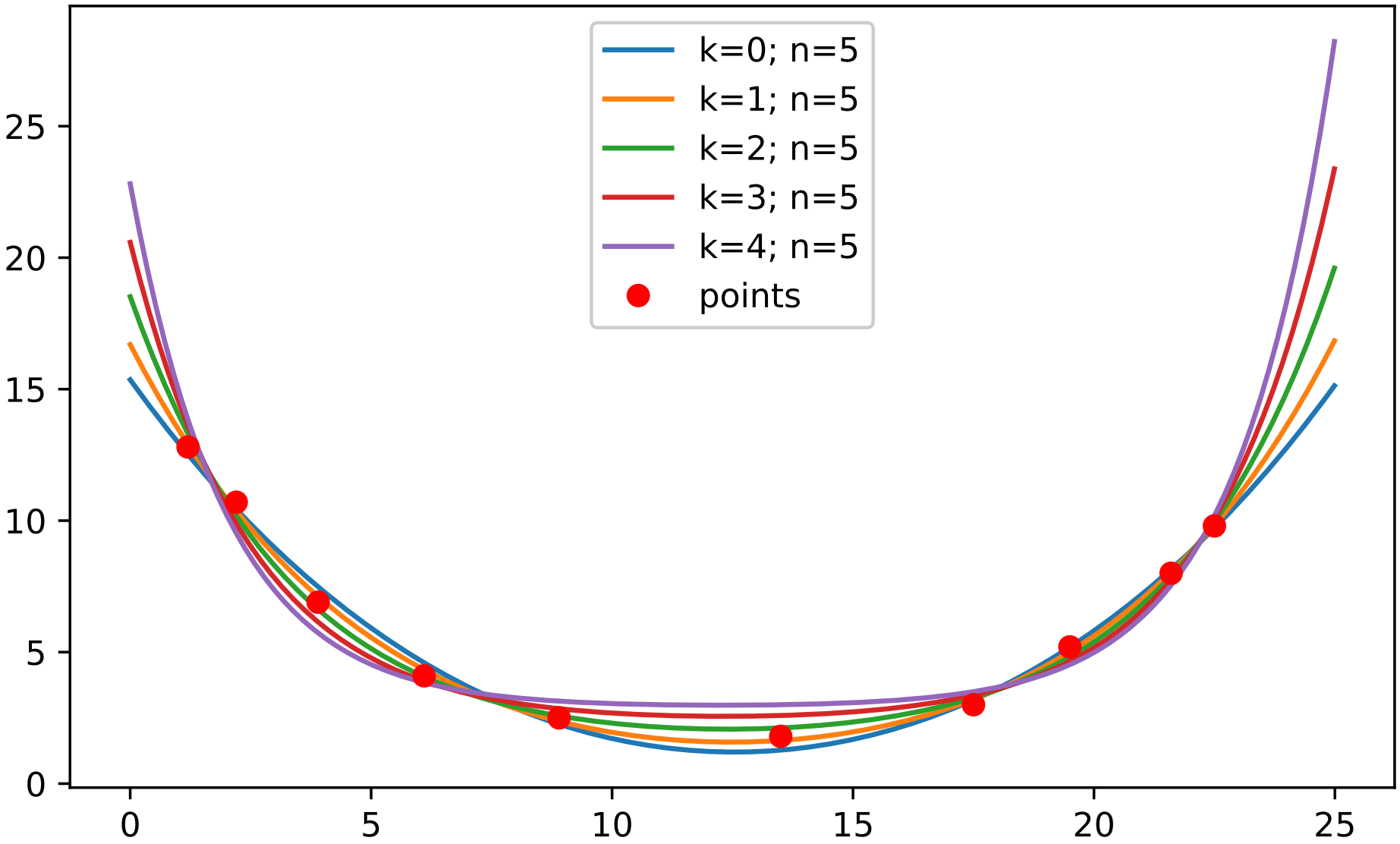
```
t = (x.item(0), x.item(1), x.item(2), d, k, n)
parameter_list.append(t)
```

Plotten der Grafik

```
x = numpy.arange(0, 25, 0.01)
for entry in parameter_list:
    a, b, c, d, k, n = entry
    plt.plot(x, a * (e ** (d * x)) +
                b * (e ** (-d * x)) + c,
             "-", label="...")

plt.plot(map(lambda pair: pair[0], points),
         map(lambda pair: pair[1], points),
         "ro", label="points")

plt.legend()
plt.xlabel("x")
plt.ylabel("y")

plt.show()
```

Experimente

Experimente



Ergebnisse

- Norm Des Residuums für $n = 55, k = 16 \Rightarrow d \approx 0.215$ am kleinsten

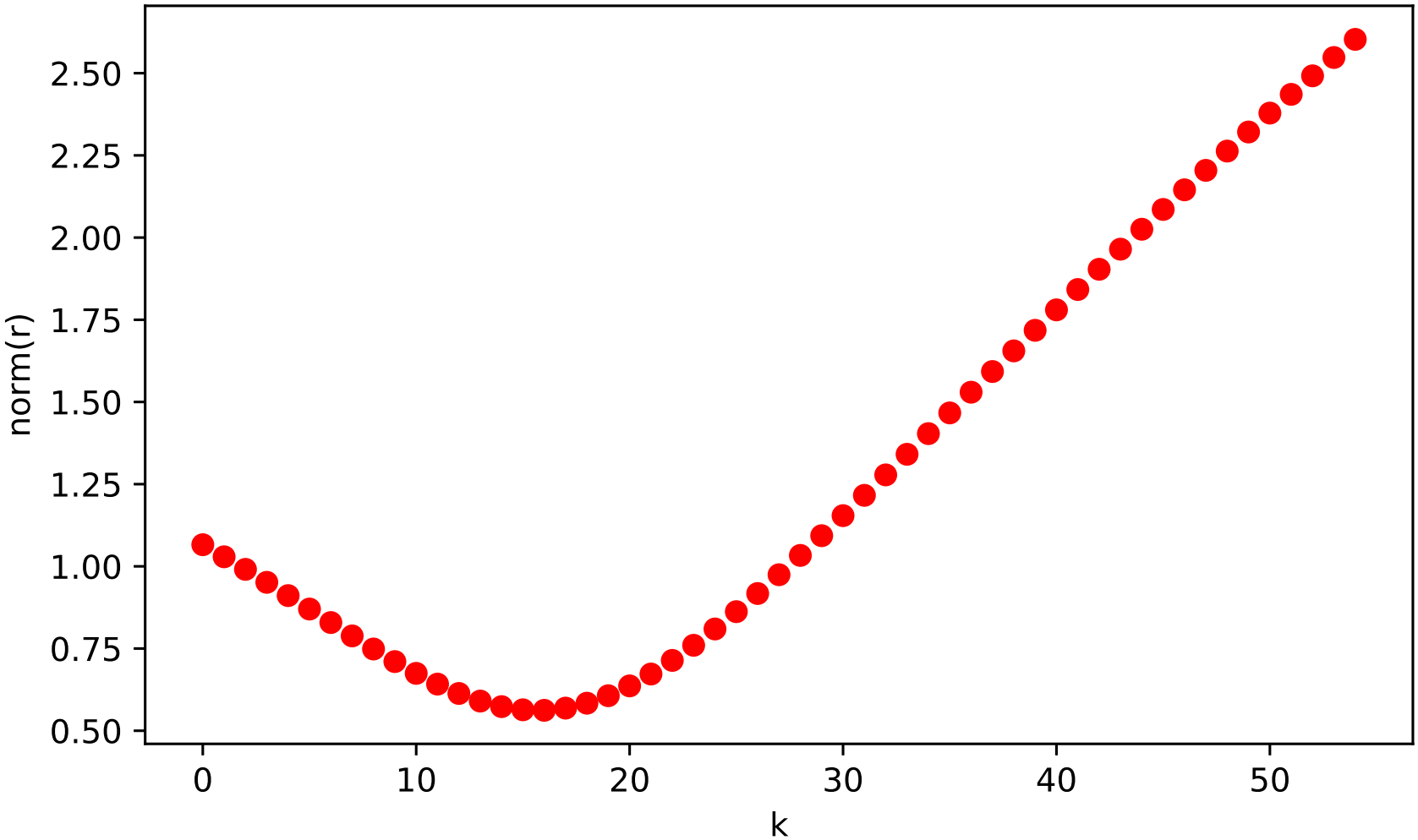
Ergebnisse

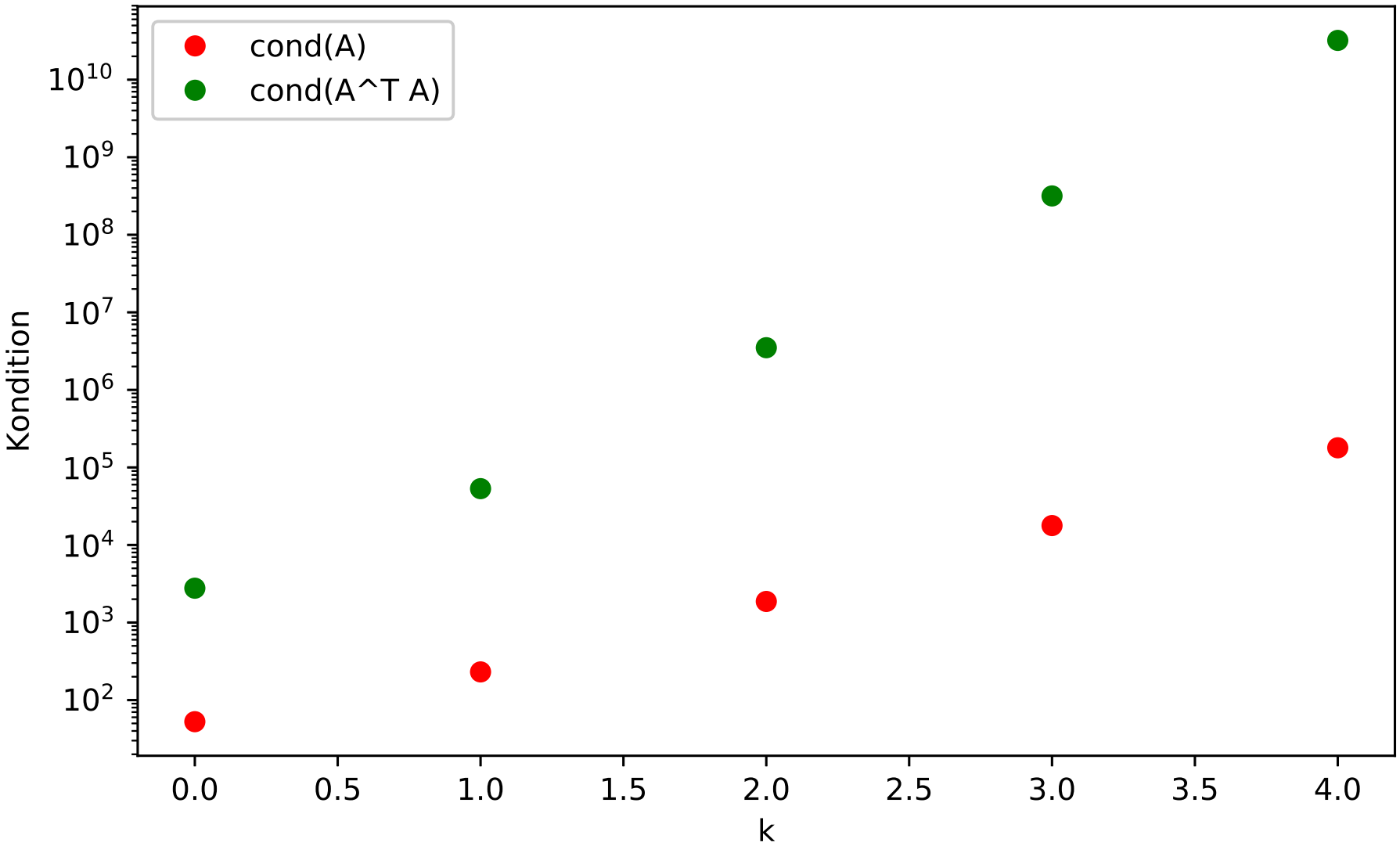
- Norm Des Residuums für $n = 55, k = 16 \Rightarrow d \approx 0.215$ am kleinsten
- Kondition von A steigt etwa exponentiell mit k

Ergebnisse

- Norm Des Residuums für $n = 55, k = 16 \Rightarrow d \approx 0.215$ am kleinsten
- Kondition von A steigt etwa exponentiell mit k
- $cond_2(A^T A) \approx cond_2(A)^2$

Für $n = 55$




cond(A)
cond(A^T A)
Kondition
10^{10}
10^{9}
10^{8}
10^{7}
10^{6}
10^{5}
10^{4}
10^{3}
10^{2}
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
k

Danke Fürs Zuhören!