

Exponentielle Regression

Projektpraktikum Serie 4

Tom Lambert Yuuma Odaka

January 22, 2018

- 1 Einführung in die Theorie
- 2 Programm und Implementierung
- 3 Experimente

Einführung in die Theorie

Exponentielle Regression

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

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

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

Festlegen der Parameter

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

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

```
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
```

```
# 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))
```

```
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):
    if r.shape[0] != r.shape[1]:
        raise Exception("r is not quadratic")

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

```
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))
```

```
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)
```

```
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="...")

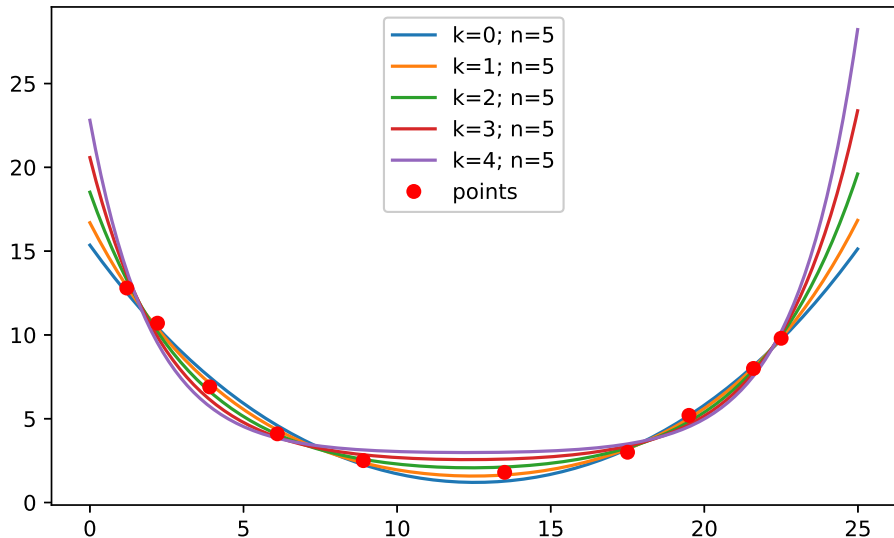
plt.plot(map(lambda pair: pair[0], points),
         map(lambda pair: pair[1], points),
         "ro", label="points")

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

plt.show()
```


Experimente

Experimente

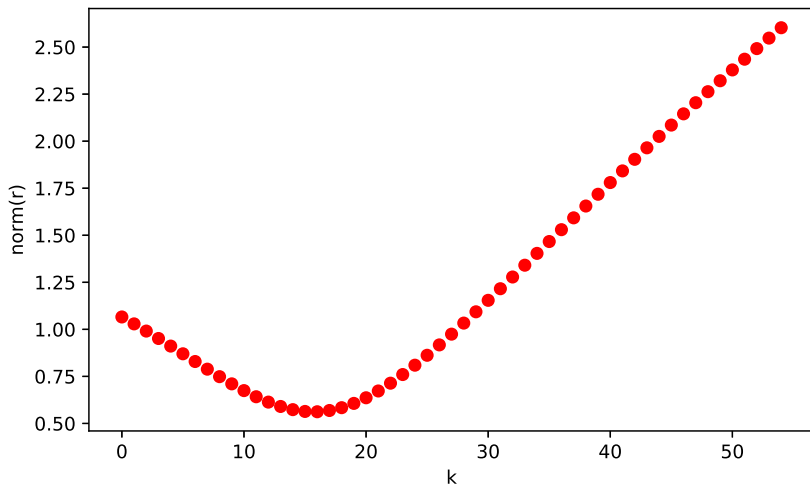


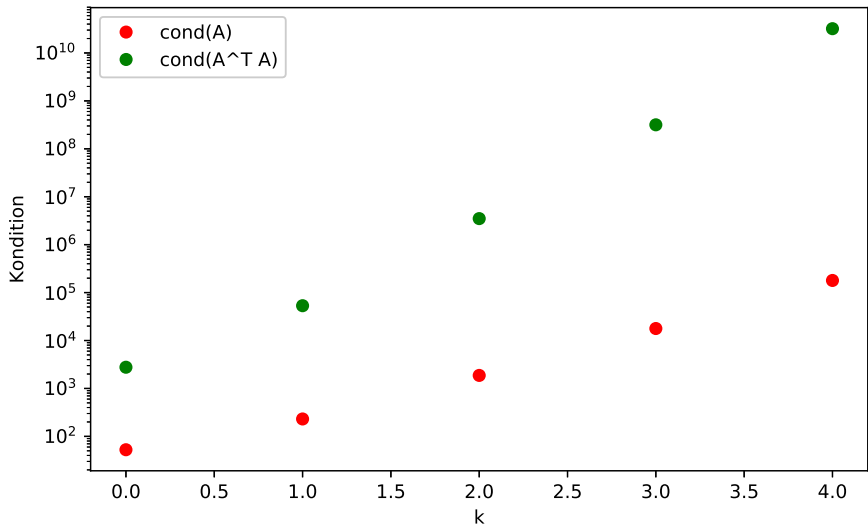
- Norm Des Residuums für $n = 55, k = 16 \Rightarrow d \approx 0.215$ am kleinsten

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

- Norm Des Residuums für $n = 55, k = 16 \Rightarrow d \approx 0.215$ am kleinsten
- Kondition von A steigt etwa exponentiell mit k
- $\text{cond}_2(A^T A) \approx \text{cond}_2(A)^2$

Für $n = 55$





Danke Fürs Zuhören!