SML: Exercise 2

Rinor Cakaj, Patrick Nowak

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer Term 2020
Sheet 1

Task 1.1: Density Estimation

We are given data C1 and C2, which we suppose to be generated by 2D-Gaussians with parameters μ_1, Σ_1 and μ_2, Σ_2, respectively.

1.1a)

Assume we are given iid. datapoints $x_i, i = 1, .., n$ which are generated by a 2D-Gaussian. Following the max-likelihood principle, we maximize the log-likelihood function

$$l(\mu, \Sigma, x_1, ..., x_n) = \ln(\prod_{i=1}^{n} p(x_i|\mu, \Sigma)) = \sum_{i=1}^{n} \ln(p(x_i|\mu, \Sigma))$$

for the Gaussian probability density

$$p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma (x-\mu)\right) \quad . \tag{1}$$

We receive

$$l(\mu, \Sigma) := l(\mu, \Sigma, x_1, ..., x_n) = \sum_{i=1}^{n} \left(-\frac{k}{2}\ln(2\pi) - \frac{1}{2} ln(|\Sigma|) - \frac{1}{2}(x_i-\mu)^T \Sigma (x_i-\mu)\right) \tag{2}$$

$$= -\frac{nk}{2}\ln(2\pi) - \frac{n}{2} ln(|\Sigma|) - \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T \Sigma (x_i-\mu) \quad . \tag{3}$$

We compute the derivatives w.r.t. μ and Σ and set them equal to zero. This yields

$$\frac{d}{d\mu} l(\mu, \Sigma, x_1, ..., x_n) = \frac{d}{d\mu} \; -\frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T \Sigma (x_i-\mu)$$

$$= -\sum_{i=1}^{n} \frac{d}{d\mu}\frac{1}{2}(x_i-\mu)^T \Sigma (x_i-\mu) \quad .$$

Using the matrix identity $\frac{d}{dw}\frac{w^T A w}{dw} = 2Aw$ which holds if w does not depend on A and if A is symmetric, we get (with $w = (x-\mu), dw = -d\mu$)

$$0 \overset{!}{=} \frac{d}{d\mu} l(\mu, \Sigma, x_1, ..., x_n)$$

$$0 \overset{!}{=} -\sum_{i=1}^{n} \Sigma^{-1}(x_i-\mu) \quad .$$

Finally, we use that Σ^{-1} is positive definite, so we can leave it out here and get

$$0 \stackrel{!}{=} n\mu - \sum_{i=1}^{n} x_i \quad ,$$

which is solved for the MLE-estimate

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad . \tag{4}$$

Secondly, we need to compute the derivative w.r.t Σ. To do that, we will need some results from mathematical classes. The following is used without prove:

- Cyclic permutations of a matrix product do not change the trace of it:

$$tr\left[ABC\right] = tr\left[CAB\right]$$

- The trace of a scalar is the scalar itself. In particular: the result of a quadratic form $x^T Ax$ is a scalar, such that:

$$x^T Ax = tr\left[x^T Ax\right] = tr\left[x^T xA\right]$$

- $\frac{d}{dA} tr\left[AB\right] = B^T$
- $\frac{d}{dA} \ln|A| = A^{-T}$

As a first result of these assumptions, we can show, that

$$\frac{d}{dA}x^T Ax = \frac{d}{dA} tr\left[x^T xA\right] = \left[xx^T\right]^T = xx^T \ .$$

We now got the tools to re-write the log-likelihood function in (3) to

$$\begin{aligned} l(\mu,\Sigma) &= -\frac{nk}{2}\ln(2\pi) - \frac{n}{2}ln(|\Sigma|) - \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T\Sigma(x_i-\mu) \\ &= C + \frac{n}{2}ln(|\Sigma^{-1}|) - \frac{1}{2}\sum_{i=1}^{n} tr\left[(x_i-\mu)(x_i-\mu)^T\Sigma^{-1}\right] \end{aligned}$$

for a constant C, and taking the derivative w.r.t Σ^{-1} yields

$$\frac{d}{d\Sigma^{-1}}l(\mu,\Sigma) = \frac{n}{2}\Sigma^T - \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)(x_i-\mu)^T$$

and plugging in $\hat{\mu}$ as an estimation of μ and setting equal to zero finally gives us

$$\begin{aligned} 0 &\stackrel{!}{=} \frac{d}{d\Sigma^{-1}}l(\hat{\mu},\Sigma) \\ 0 &\stackrel{!}{=} \frac{n}{2}\Sigma^T - \frac{1}{2}\sum_{i=1}^{n}(x_i-\hat{\mu})(x_i-\hat{\mu})^T \end{aligned}$$

which is solved for the (biased) MLE estimate

$$\tilde{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i-\hat{\mu})(x_i-\hat{\mu})^T \tag{5}$$

1.1b)

We compute the prior probabilities of C1 and C2, using the following python code. We read the number of data points in each class and divide it by the sum of total data points in both classes.

```
import numpy as np

link1="../hw2/dataSets/densEst1.txt"
link2="../hw2/dataSets/densEst2.txt"

def get_lengths():
        l1=0;
        l2=0;
        for line in open(link1):
                l1=l1+1
        for line2 in open(link2):
                l2=l2+1
        return (l1,l2)

def get_priors(l1,l2):
        p_C1=l1/(l1+l2)
        p_C2=l2/(l1+l2)
        return(p_C1,p_C2)
```

Calling

```
lengths=get_lengths()
print(get_priors(lengths[0],lengths[1]))
```

we get the following results for the prior probabilities: p(C1)=0.239 and p(C2)=0.761 .

1.1c)

Having a data set X and an estimator $\hat{\theta}$ on the true parameter θ, we define the bias of an estimator as the expected deviation from the true parameter. We get the formula

$$bias(\hat{\theta}) = \mathbb{E}_X\left[\hat{\theta}(X) - \theta\right]$$

We call an estimator unbiased iff $bias(\hat{\theta}) = 0$ and biased otherwise.
From the lecture we know that the MLE of the mean of a Gaussian is unbiased, but the MLE of the variance of a Gaussian is biased. In fact, an unbiased estiamtor on the variance would be the sample covariance matrix

$$\hat{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu})^T \ . \tag{6}$$

To calculate the conditional distribution densities $p(x|C_i)$ we need to estimate the underlying parameters μ_i and Σ_i. For both classes we use the MLE-estimate of the mean, which is unbiased and calculated like in (4). We compute the biased MLE-estimate $\tilde{\Sigma}$ for the variance via (5) and the unbiased estimate $\hat{\Sigma}$ via (6). We wrote the following python code

```
def extract_data(t):
        a=np.empty((0,2),float)
        for line in t:
                v=np.array([[line.split()[0],line.split()[1]]],float)
                a=np.append(a,v,axis=0)
        return a
```

```
def get_mean_estimation(points):
        m=np.zeros(2)
        m[0]=sum(points[:,0])
        m[1]=sum(points[:,1])
        m=m/len(points)
        return m

def get_biased_var_estimation(mean_est,points):
        l=points-mean_est
        s=np.zeros((2,2))
        for line in l:
                s=s+np.outer(line,line)
        s=s/len(l)
return s

def get_unbiased_var_estimation(mean_est,points):
        l=points-mean_est
        s=np.zeros((2,2))
        for line in l:
                s=s+np.outer(line,line)
        s=s/(len(l)-1)
return s

def print_results(link):
        a=extract_data(open(link))
        mean=get_mean_estimation(a)
        print("mean=",mean)
        sigma=get_unbiased_var_estimation(mean,a)
        print("biased_Sigma=",get_biased_var_estimation(mean,a))
        print("sigma=",sigma)
```

which gives us the results for class C1

```
print_results(link1)

mean= [-0.70681374 -0.81343083]
biased_Sigma= [[9.01952586 2.67287085]
               [2.67287085 3.59633965]]
sigma= [[9.05742302 2.6841014 ]
        [2.6841014  3.61145033]]
```

and for class C2

```
print_results(link2)

mean= [3.98534252 3.98438364]
biased_Sigma= [[4.1753815  0.02758324]
        [0.02758324 2.75296323]]
sigma= [[4.18087542 0.02761954]
        [0.02761954 2.75658555]]
```