Principles of Machine Learning: Exercise 2

Alina Pollehn (3197257), Julian Litz (3362592), Manuel Hinz (3334548)
Felix Göhde (3336445), Felix Lehmann (3177181), Caspar Wiswesser (3221493)
Adrian Köring (3347785), Greta Günther (3326765), Linus Mallwitz (3327653)
Niklas Mueller-Goldingen (3363219), Jennifer Kroppen (2783393)

19.11.2023

Task 2.1.1-2 :: Loading

Instead of removing the outliers, its easier to keep the inliers:

```
inliers = w > 0
X = np.stack([h[inliers], w[inliers]])
# X.shape = [2, 37] = [F, N]
```

Maximum Likelihood Estimation of a Gaussian via empirical mean and covariance:

```
μ = np.mean(X, axis=1, keepdims=True)
# μ = [[175.729], [73.865]] with μ.shape=[2, 1]
S = np.cov(X - μ, ddof=1) # ddof = degree of freedom = scales with 1/(n-1)
# S == (1 / (N-1)) * (X - μ) @ (X - μ).T
# S == [[ 75.925,  64.546 ],
#       [ 64.546, 186.953]]
```

Task 2.1.3 :: Predictions

Conditional Probability of a Gaussian:

```
for h in np.arange(140, 220, step=10):
    μw = μ[1] + S[1, 0] * S[0, 0]**(-1) * (h - μ[0])
    Sw = S[1, 1] - S[1, 0] * S[0, 0]**(-1) * S[0, 1]
```

	Height	Weight	Covariance
0	140	43.490072	132.081358
1	150	51.991338	132.081358
2	160	60.492604	132.081358
3	170	68.993869	132.081358
4	180	77.495135	132.081358
5	190	85.996401	132.081358
6	200	94.497666	132.081358
7	210	102.998932	132.081358

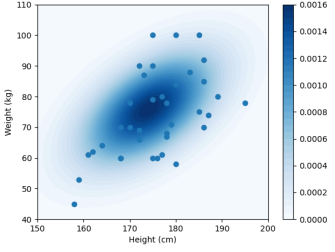


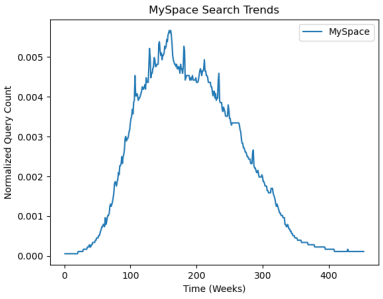
Figure: Estimated PDF and given data

Task 2.1.3 :: Pondering

+ Results seem plausible – can mostly judge are around mean though
- Fixed variance doesnt match expectations
  - maybe taller people → more variance?
  - there are more average sized people → maybe more variance?
- Cube-Square-Law / BMI suggests a non-linear / quadratic relationship?
- Plausibility? Test-Set and more 'human-readable' metrics!

Task 2.2.1 :: Loading

```
import pandas as pd
# Loading data as pandas Dataframe
df = pd.read_csv("myspace.csv",
                 header=None,
                 names=["Week", "Accesses"])
# Filter leading zeros only with cumulative sum
df = df[np.cumsum(df.Accesses) > 0]
h = np.array(df.Accesses, dtype=float)
t = np.arange(1, len(h)+1, dtype=float)
```

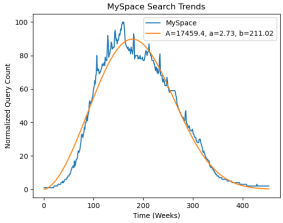


Task 2.2.1 :: ManuDiff Newton

Task 2.3 :: Scipy CurveFit

```
def weibull(t, A, alpha, beta):
    ab, tb = alpha / beta, t / beta
    return A * ab * tb**(alpha - 1) * np.exp(-tb**alpha)
(A, alpha, beta), _ = curve_fit(weibull, t, h, p0=[1000, 1.0, 1.0])
```

- mostly works just like that
- could have also scaled the data (see 2.5) instead of adding the amplitude parameter



Task 2.4

Task 2.5 :: Scipy Minimize

```
def KL(f, q): return np.sum(f * np.log(f / q))

def objective(x):
    return KL(weibull(t, alpha=x[0], beta=x[1]), q)
result = minimize(objective, x0=[1.0, 100.], bounds=[(0, 10), (0, 500)])
```

- Bounds are important, but also not: $[-\infty, \infty]$ works, too
- Feels odd anyway, because a gradient method optimizing $1 \rightarrow 2.8$ shouldn't come near 0 anyway

$\Rightarrow$ Adding bounds just change the algorithm underneath

- Without bounds 'BFGS' is choosen and doesnt quite work
- 'L-BFGS-B' or 'SLSQP' without bounds works equally well