

SMD-Abgabe

10. Übungsblatt

Lars Kolk

lars.kolk@tu-dortmund.de

Julia Sobolewski

julia.sobolewski@tu-dortmund.de

Jannine Salewski

jannine.salewski@tu-dortmund.de

Abgabe: 10.01.2018

$$\begin{array}{c|c|c|c|c} \overline{28} & \overline{29} & \overline{30} & \overline{31} & \overline{2} \\ \hline 5,25 & 5/9 & 4/5 & 2/5 & 16,25/20 \end{array}$$

TU Dortmund – Fakultät Physik

Frage 28: Entfaltung in zwei Intervallen

a) $\vec{g} = A \cdot 0,8 \vec{f}$
gemessen nur 80% ✓

$$A = \begin{pmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1+\varepsilon \end{pmatrix} \quad \checkmark$$

b) $\vec{f} = \frac{5}{4} A^{-1} \vec{g}$ $A^{-1} = \frac{1}{4(1-2\varepsilon)} \begin{pmatrix} 1-\varepsilon & -\varepsilon \\ -\varepsilon & 1-\varepsilon \end{pmatrix} - \frac{1}{1-2\varepsilon} \begin{pmatrix} 1-\varepsilon & -\varepsilon \\ -\varepsilon & 1-\varepsilon \end{pmatrix} \quad \checkmark$

$$\vec{f} = \frac{5}{4(1-2\varepsilon)} \begin{pmatrix} 1-\varepsilon & -\varepsilon \\ -\varepsilon & 1-\varepsilon \end{pmatrix} \vec{g} \quad \checkmark \quad 0,15/0,15$$

c) $V[f] = B^{-1} V[g] B^{-1}$

$$\text{mit } B^{-1} = \frac{5}{4(1-2\varepsilon)} \begin{pmatrix} 1-\varepsilon & -\varepsilon \\ -\varepsilon & 1-\varepsilon \end{pmatrix}$$

$$V[g] = \begin{pmatrix} g_1^2 & 0 \\ 0 & g_2^2 \end{pmatrix} = \begin{pmatrix} g_1 & 0 \\ 0 & g_2 \end{pmatrix} \quad \checkmark$$

$$V[f] = \left(\frac{5}{4(1-2\varepsilon)}\right)^2 \begin{pmatrix} 1-\varepsilon & -\varepsilon \\ -\varepsilon & 1-\varepsilon \end{pmatrix} \begin{pmatrix} g_1 & 0 \\ 0 & g_2 \end{pmatrix} \begin{pmatrix} 1-\varepsilon & -\varepsilon \\ -\varepsilon & 1-\varepsilon \end{pmatrix} \quad \checkmark$$

$$= \left(\frac{5}{4(1-2\varepsilon)}\right)^2 \begin{pmatrix} g_1(1-\varepsilon)^2 + g_2\varepsilon^2 & -(g_1+g_2)\varepsilon(1-\varepsilon) \\ -(g_1+g_2)\varepsilon(1-\varepsilon) & g_1\varepsilon^2 + g_2(1-\varepsilon)^2 \end{pmatrix} \quad \checkmark$$

1/1

d) $\sigma_{f_1} = \frac{5}{4(1-2\varepsilon)} \sqrt{g_1(1-\varepsilon)^2 + g_2\varepsilon^2} \quad \checkmark$

$$\sigma_{f_2} = \frac{5}{4(1-2\varepsilon)} \sqrt{g_1\varepsilon^2 + g_2(1-\varepsilon)^2} \quad \checkmark$$

$$p(f_1, f_2) = \frac{1}{\sigma_{f_1} \sigma_{f_2}} \text{corr}(f_1, f_2) = \frac{-(g_1+g_2)\varepsilon(1-\varepsilon)}{\sqrt{(g_1(1-\varepsilon)^2 + g_2\varepsilon^2)(g_1\varepsilon^2 + g_2(1-\varepsilon)^2)}} \quad \checkmark$$

für $\varepsilon=0,1$: $V[f] = \begin{pmatrix} 399,63 & -81,08 \\ -81,08 & 339,07 \end{pmatrix}$

$$\sigma_{f_1} = 19,99 \quad \checkmark$$

$$\sigma_{f_2} = 18,41 \quad \checkmark$$

$$\rho = -0,22 \quad \checkmark$$

f wurde nicht ausgerechnet

1,5/2

$$\vec{f} = \begin{pmatrix} 254,84 \\ 206,40 \end{pmatrix}$$

e)

für $\epsilon = 0,4$: $V[f] = \begin{pmatrix} 3868 & -3459 \\ -3459 & 3626 \end{pmatrix}$ ✓

$$\alpha_{f_1} = 62,20 \quad \checkmark$$

$$\alpha_{f_2} = 60,22 \quad \checkmark$$

$$g = -0,92 \quad \checkmark$$

$$\vec{r} = \begin{pmatrix} 327,5 \\ 133,75 \end{pmatrix} \quad \checkmark$$

'was hat sich geändert
+ was bedeutet
dies?'

~~0,25~~
0,5

f) Für $\epsilon = 0,5$ lässt es sich nicht berechnen, da sonst durch 0 geteilt werden müsste in dem Vorfaktor. ✓
Außerdem wäre die Zuteilung in die beiden Gruppen 1 und 2 somit zwecklos, da die Wahrscheinlichkeiten genau 50% betragen und somit ein Ereignis zufällig zu einer der beiden Gruppen zugewiesen wird.

✓

~~0,5~~
0,5

Aufgabe 29

a)

Die Antwortmatrix beschreibt die "Detektorantwort", also Messung, aus einem Bin.

*ja aber was für ein Messung?
Was wird genau modelliert? 05/19.*

```
In [9]: import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import diags
np.random.seed(0)
```

```
In [10]: def Matrix(Dimension, Epsilon):
    C = diags([Epsilon, 1-2*Epsilon, Epsilon], [-1, 0, 1], shape=(Dimension,
Dimension)).toarray()
    C[0][0]=C[Dimension-1][Dimension-1]=1-Epsilon
    return C
print('Test für 5x5:')
Matrix(5, 0.23)
```

Test für 5x5:

```
Out[10]: array([[0.77, 0.23, 0. , 0. , 0. ],
[0.23, 0.54, 0.23, 0. , 0. ],
[0. , 0.23, 0.54, 0.23, 0. ],
[0. , 0. , 0.23, 0.54, 0.23],
[0. , 0. , 0. , 0.23, 0.77]])
```



1/1 P.

b)

```
In [11]: f=np.array([193, 485, 664, 763, 804, 805, 779, 736, 684, 626,
566, 508, 452, 400, 351, 308, 268, 233, 202, 173])
A=Matrix(20, 0.23)
g=np.dot(A, f)
print('g = ', g, '\n')
gmess=np.random.poisson(g)
print('Die poisson gezogenen g sind dann: ', gmess)
```

```
g = [260.16 459.01 645.6 749.66 794.8 798.79 775.09 733.93 682.62 625.54
566.46 508.46 452.92 400.69 352.38 308.69 269.15 233.92 202.46 179.67]
```

```
Die poisson gezogenen g sind dann: [262 465 640 745 873 825 780 684 705 623
534 510 438 398 358 346 262 243
209 167]
```



1/1 P.

c)

Mit der Faltungsgleichung $g=A \cdot f$ und der Diagonalbasis $A=U \cdot D \cdot U^{-1}$ folgt:

$$U^{-1} \cdot g = D \cdot U^{-1} \cdot f.$$



$b = ?$

$c = ?$

1,5/2

```
In [12]: Werte, Vektoren = np.linalg.eig(A)
Index=Werte.argsort() ← absteigend nach Betrag sollte
Werte=Werte[Index]
Vektoren=Vektoren[:,Index] führt werden
```

d

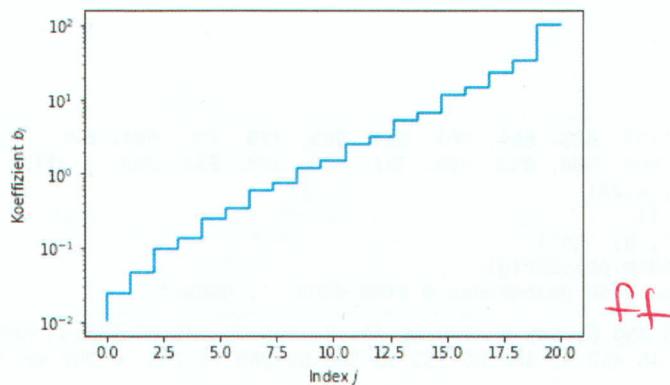
```
In [13]: U=Vektoren
Uinv=np.linalg.inv(U) ←
D=np.diag(Werte) ✓
Dinv=np.linalg.inv(D)

b=Uinv @ f ✓
c=Uinv @ gmess ↴
kg=np.diag(gmess)
```

*die Zeilen von
bei A gesetzt*

```
In [14]: Vg = np.diag(gmess) #Kovarianzmatrix von gmess
B = Dinv@Uinv
bvb = B@Vg@B.T
bvbvar = np.diag(bvb)
bvbstan = np.sqrt(bvbvar)
bvbskal = np.abs(b/bvbstan)
```

```
In [15]: x=np.linspace(0,np.size(bvbskal),np.size(bvbskal))
plt.step(x, bvbskal, )
plt.yscale('log')
plt.xlabel(r'Index $j$')
plt.ylabel(r'Koeffizient $b_j$')
plt.show()
```



ff

In []:

Was lässt sich über die Koeff. sagen, wenn sie unterhalb von 1 liegen?
1/22.

e) 0/22.

Aufgabe 30

a)

```
In [18]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

```
In [19]: df = pd.read_hdf('image_parameters_smd_reduced.hdf5')

# Labels erstellen

y = np.array(df['corsika_run_header_particle_id'])
y[y==1] = 1
y[y==14] = 0
```

```
In [20]: # Feature-Set erstellen

size = np.array(df['size'])
width = np.array(df['width'])
length = np.array(df['length'])
islands = np.array(df['num_islands'])
pixel = np.array(df['num_pixel_in_shower'])
charge_mean = np.array(df['photoncharge_shower_mean'])

X = np.vstack((size, width, length, islands, pixel, charge_mean)).T
```

ihr hättest noch
mehr features
verwenden können

```
In [21]: # Trainings- und Test-Datensatz erstellen

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=0)
```

✓

NJP

b)

```
In [22]: # Klassifizierung mit Random Forest
def auc(n_estimators):
    clf = RandomForestClassifier(n_estimators=n_estimators, max_depth=2,
                                 random_state=0, n_jobs=-1)
    validation = cross_validate(clf, X_train, y_train, cv=5,
                                scoring=['roc_auc'], return_train_score=True)
    print('n_estimators = {:.0f}: \n {:.3f} +- {:.3f}'.format(n_estimators,
                                                             validation['test_roc_auc'].mean(),
                                                             validation['test_roc_auc'].std()))
    auc(1)

n_estimators = 1:
0.604 +- 0.006
```

macht das Ergebnis vermutlich schlecht

```
In [23]: auc(10)
```

```
n_estimators = 10:
0.669 +- 0.007
```

✓

```
In [24]: auc(100)
```

```
n_estimators = 100:
0.687 +- 0.004
```

✓

Die Area Under Curve ist bei 100 estimators am besten.

1/1P.

c)

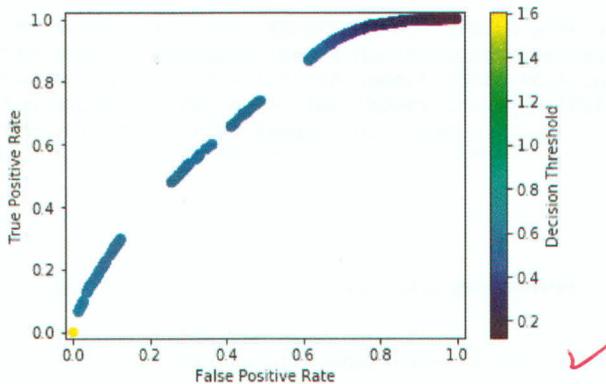
```
In [25]: forest = RandomForestClassifier(n_estimators=100, max_depth=2,
                                         random_state=0, n_jobs=-1)
forest.fit(X_train, y_train)
predict = forest.predict_proba(X_test)
```

0,5/0,5

d)

```
In [26]: fpr, tpr, t = roc_curve(y_test, predict[:, 1])
plt.scatter(fpr, tpr, c=t, cmap='viridis')
plt.colorbar(label='Decision Threshold')
plt.xlim([-0.02, 1.02])
plt.ylim([-0.02, 1.02])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

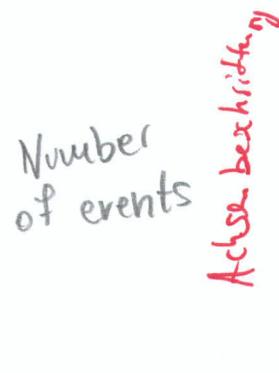
Out[26]: Text(0, 0.5, 'True Positive Rate')



```
In [27]: print('ROC-AUC-Score: {:.3f}'.format(roc_auc_score(y_test, predict[:, 1])))
ROC-AUC-Score: 0.688
```

In [28]: plt.hist(predict[:, 1], bins=25)
plt.xlabel('Classification Score')

Out[28]: Text(0.5, 0, 'Classification Score')



getrennt für Signal und Untergrund
2 Histogramme erstellen!

~~Verteilung der Hadron
Ergebnisse fehlt
-Wichtig zum Vergleichen~~
Sieht sehr komisch aus,
da

Die Area Under Curve ist mit 0.688 relativ niedrig und die ROC-Curve liegt nur knapp über der Winkelhalbierenden. Dadurch ist der Klassifizierer nur leicht besser als zufällig zu raten und ist deshalb nicht wirklich dazu geeignet das Signal vom Untergrund zu trennen. Dies sieht man ebenfalls an der Verteilung der Gamma- und Hadronenergebnisse. Es sind zwar zwei Peaks erkennbar (der hohe Doppel-Peak bei niedrigerem Classification Score für die Hadronen und der kleinere Peak rechts daneben für die Gammas), allerdings liegen diese so nah beieinander, dass der Gamma-Peak bei größerem binning nicht mehr zu erkennen ist. Für einen perfekten Klassifizierer geht die Area Under Curve gegen 1 und die ROC-Curve nähert sich einer Stufenfunktion. Die Verteilung entlang des Classification Scores hätte nur zwei Bins. Den Hadronen-Bin bei 0 und den Gamma-Bin bei 1.

1,5
2,5

Aufgabe 31

```
In [8]: import h5py
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

a)

```
In [9]: df = pd.read_hdf('../Aufgabe30/image_parameters_smd_reduced.hdf5')

df = df[df.corsika_run_header_particle_id == 1]
df = df[df.corsika_event_header_total_energy > 500]
df.corsika_event_header_total_energy = np.log10(df.corsika_event_header_total_energy)

# Feature-Set erstellen

size = np.array(df['size'])
width = np.array(df['width'])
length = np.array(df['length'])
islands = np.array(df['num_islands'])
pixel = np.array(df['num_pixel_in_shower'])
charge_mean = np.array(df['photoncharge_shower_mean'])

X = np.vstack((size, width, length, islands, pixel, charge_mean)).T
y = np.array(df['corsika_event_header_total_energy'])

# Trainings- und Test-Datensatz erstellen
```

Es ist sinnvoll die Energien zu logarithmieren, da diese um mehrere Größenordnungen größer als die Ladungen sind

er wäre einfacher gewesen, wenn du mehr mit dataframes gearbeitet hättest
z.B. sind df.drop() mögliche Funktionen

✓

0,5P.

b)

```
In [10]: forest = RandomForestRegressor(n_estimators=100, max_depth=15, random_state=0, n_jobs=-1)
forest.fit(X_train, y_train)
predict = forest.predict(X_test)
```

✓

0,25P.

c)

```
In [27]: def evaluate_performance(y_true, y_pred):
    delta_y = y_true - y_pred
    mean_delta_y = np.mean(delta_y)
    std_delta_y = np.std(delta_y)
    mean_y_true = np.mean(y_true)
    std_y_true = np.std(y_true)

    print('Delta y = {:.3f} +- {:.3f}'.format(mean_delta_y, std_delta_y))
    print('y_true = {:.3f} +- {:.3f}'.format(mean_y_true, std_y_true))

    # Histogramm der Residuen

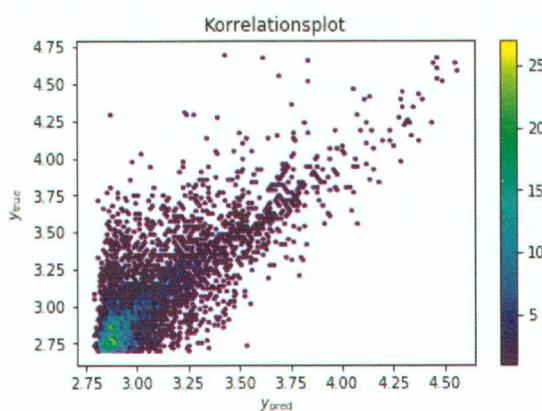
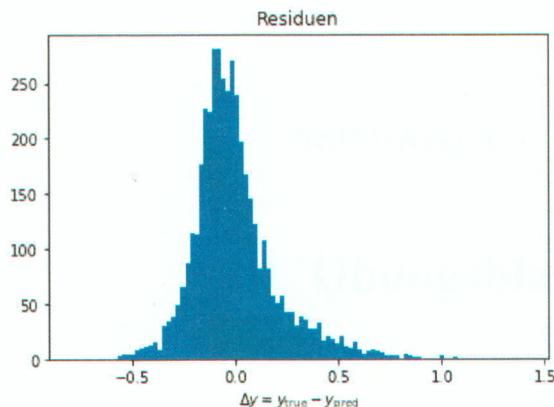
    plt.hist(delta_y, bins = 100)
    plt.xlabel(r'$\Delta y = y_{\text{true}} - y_{\text{pred}}$')
    plt.title(r'Residuen')
    plt.show()
    plt.clf()

    # Korrelationsplot

    plt.hexbin(y_pred, y_true, mincnt=1)
    plt.colorbar()
    plt.title(r'Korrelationsplot')
    plt.xlabel(r'$y_{\text{pred}}$')
    plt.ylabel(r'$y_{\text{true}}$')
    plt.show()
```

In [28]: `evaluate_performance(y_true, predict)`

Delta $y = -0.003 \pm 0.208$
 $y_{\text{true}} = 3.083 \pm 0.329$

*sieht gut aus!*

1P.

<Figure size 432x288 with 0 Axes>

Bei einem perfekten Energieschätzer bestünde das Histogramm der Residuen aus einem einzigen Bin bei 0 und der Korrelationsplot würde nur aus der Winkelhalbierenden bestehen. Da dies beim verwendeten Schätzer nicht so ist, ist dieser somit auch nicht perfekt. Allerdings liegt das Maximum der Verteilung der Residuen bei 0 und der Korrelationsplot zeigt auch den richtigen Trend, woraus man ablese kann, dass der Schätzer durchaus eine relativ gute Energieschätzung liefert.

d

In [13]: `with h5py.File('smd_deeplearning_gammas_reduced.hdf5', 'r') as f:`
 `energy = f['energy'][:]`
 `charges = f['charges'][:]`In [17]: `charges = charges[~np.isnan(charges)]`
`v = np.log(energy)`

Alternativ: nans ersetzen (durch 0 oder
 Mittelwert)

✓ 0,25P.