10. Übungsblatt zur Vorlesung
Statistische Methoden der Datenanalyse
Abgabe: 10.01.2019 23:59

WS 2018/2019
Prof. W. Rhode

Zeit	Raum	Abgabe im Moodle; Mails mit Betreff: [SMD1819]
Di. 10-12	CP-03-150	tobias.hoinka@udo.edu, felix.geyer@udo.edu und jan.soedingrekso@udo.edu
Di. 16-18	CP-03-150	simone.mender@udo.edu und alicia.fattorini@udo.edu
Mi. 10-12	CP-03-150	mirco.huennefeld@udo.edu und kevin3.schmidt@udo.edu

Aufgabe 28: *Entfaltung in zwei Intervallen* **6 P.**

Betrachten Sie ein Experiment, in dem 2 Typen von Ereignissen gezählt werden. Die Wahrscheinlichkeit ein Ereignis dem falschen Typ zuzuordnen sei ϵ und die Wahrscheinlichkeit für den richtigen Typ $1-\epsilon$. Die Wahrscheinlichkeit der falschen Zuordnung ist also für $1 \to 2$ die gleiche wie für $2 \to 1$. Des weiteren gehen bei dem Experiment leider 20% der Messungen verloren.

Die gemessen Ereigniszahlen $\mathbf{g} = (g_1, g_2)^T$ sind Poisson-verteilt und unkorreliert.

a) Stellen Sie die Antwortmatrix $\mathbf{A}$ auf und den Zusammenhang zwischen $\mathbf{A}$, $\mathbf{g}$ und der wahren Ereigniszahl $\mathbf{f} = (f_1, f_2)^T$.

b) Berechnen Sie $\mathbf{f}$ als Funktion von $\mathbf{g}$ und ϵ.

c) Berechnen Sie die Kovarianzmatrix $\mathbf{V}[\mathbf{f}]$ als Funktion von $\mathbf{g}$ und ϵ.

d) Berechnen Sie $\mathbf{f}$, $\mathbf{V}[\mathbf{f}]$, die Fehler von f_1 und f_2 und den Korrelationskoeffizienten für $g_1 = 200$, $g_2 = 169$ und $\epsilon = 0,1$.

e) Wie **d)**, aber mit $\epsilon = 0,4$. Was hat sich geändert und was bedeutet dies?

f) Was passiert bei $\epsilon = 0,5$?

Aufgabe 29: *Entfaltung mit quadratischen Matrizen* **9 P.**

In dieser Aufgabe sollen die gezeigten Plots aus der Vorlesung nachgebaut werden, um die Vorgänge bei der Entfaltung besser zu verstehen.

a) Die quadratische $n \times n$ Antwortmatrix A wird hier nicht aus MC erzeugt, sondern ist vorgegeben:

$$A = \begin{pmatrix} 1-\epsilon & \epsilon & 0 & 0 & \dots & 0 \\ \epsilon & 1-2\epsilon & \epsilon & 0 & \dots & 0 \\ 0 & \epsilon & 1-2\epsilon & \epsilon & \dots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & \dots & 0 & \epsilon & 1-2\epsilon & \epsilon \\ 0 & \dots & 0 & 0 & \epsilon & 1-\epsilon \end{pmatrix}$$

Was für einen Messprozess beschreibt die Matrix A? Im Folgenden sei $\epsilon = 0,23$. Implementieren sie eine Methode, mit der die Matrix A für beliebige Dimensionen $n \geq 3$ erzeugt werden kann.

b) Die wahre Verteilung f sei im Intervall $[0, 2]$ in 20 gleichgroßen Bins gegeben durch

$$f = [193, 485, 664, 763, 804, 805, 779, 736, 684, 626,\\ 566, 508, 452, 400, 351, 308, 268, 233, 202, 173]$$

Nun wird eine Messung simuliert, indem zuerst die wahre Verteilung f mit A gefaltet wird: $g = A \cdot f$. Die „gemessenen“ Bin-Einträge der Verteilung g^{mess} erhält man durch Ziehen aus einer Poisson-Verteilung mit dem Erwartungswert des Bin-Eintrags aus der vorherigen Faltung: $\lambda_i = g_i$. Dadurch ergeben sich statistische Schwankungen in der Messung, die die Entfaltung erschweren.
Beispiel: Die Faltung ergibt einen Wert λ_i in Bin i der Verteilung g. Dann ergibt sich der gemessene Eintrag g_i^{mess} durch eine Zufallszahl, die aus einer Poisson-Verteilung mit Erwartungswert λ_i gezogen wird.

c) Transformieren Sie die Faltungsgleichung in die Diagonalbasis von $A = U \cdot D \cdot U^{-1}$. Wie lautet die Faltungsgleichung in der neuen Basis und welchen Vorteil bietet sie? Sortieren Sie die Eigenwerte absteigend nach ihrem Betrag und ändern Sie dementsprechend die Reihenfolge der zugehörigen Eigenvektoren in der Transformationsmatrix U.

d) Transformieren Sie die Verteilungen $f \to b$ und $g \to c$ mit der Matrix U aus **c)** in die neue Basis. Berechnen Sie ebenfalls die Kovarianzmatrix der Verteilung b mittels der „BVB“-Formel. Normieren Sie die entfalteten Koeffizienten $b = D^{-1}U^{-1}g^{\text{mess}}$ auf ihre Standardabweichungen und stellen Sie die Einträge b_j in einem Plot gegen ihren Index dar. Was lässt sich über Koeffizienten b_j sagen, die in dem Plot unterhalb der 1 liegen?
Info: Falls Sie die Verteilung g nicht erzeugen konnten, benutzen Sie hier die folgenden Werte:

$$g = [256, 457, 630, 787, 774, 816, 771, 749, 665, 647,\\ 543, 510, 440, 410, 348, 296, 279, 243, 205, 174]$$

e) Regularisieren Sie, indem Sie nicht signifikante Koeffizienten b_j ab einen Cutoff-Index auf 0 setzen. Entfalten sie dann je einmal mit Regularisierung und ohne. Stellen Sie beide Lösungen mit statistischen Fehlern in einem Plot zusammen mit der gegebenen wahren Verteilung dar. Was ist der Unterschied der Lösung mit und ohne Regularisierung?

10. Übungsblatt zur Vorlesung
Statistische Methoden der Datenanalyse
Abgabe: 10.01.2019 23:59

WS 2018/2019
Prof. W. Rhode

Aufgabe 30: *Data Mining Anwendung – Gamma/Hadron Klassifizierung* **5 P.**

In dieser Aufgabe soll eine reale Problemstellung aus der Gammaastronomie gelöst werden. Dafür soll eine Gamma/Hadron Klassifizierung durchgeführt werden mit einem Random Forest.

Problemstellung

Die Problemstellung wird ausführlich in der Vorlesung besprochen. Hier gibt es außerdem Hinweise zu dem Datensatz.

Datensatz

Die Daten sind unter `https://factdata.app.tu-dortmund.de/smd/small` zu finden. Für diese Aufgabe wird die Datei `image_parameters_smd_reduced.hdf5` benötigt.

Hinweise

Benutzen Sie wie in der Vorlesung erwähnt die Python-Bibliotheken `sklearn` und `pandas`. Alles was in dieser Aufgabe verwendet werden muss, ist bereits in den gegebenen Bibliotheken implementiert. Erfinden Sie das Rad nicht neu!

a) Laden Sie das DataFrame mit `pandas.read_hdf` und erstellen Sie die benötigten Labels y_i. Dabei soll y_i gleich 1 sein für die Signal Ereignisse und 0 für den Untergrund. In der Gammaastronomie sind die Gamma-Teilchen (Teilchen-ID 1) das Signal und die Hadronen (Teilchen-ID 14) Untergrund. Die Teilchen-ID ist in der Spalte `corsika_run_header_particle_id` gespeichert. Erstellen sie zusätzlich aus dem DataFrame ein Set von Features X, dass dann zur Klassifizierung genutzt werden kann. Überlegen Sie sich dabei welche Spalten dafür verwendet werden dürfen. Teilen Sie nun die Labels y und Features X in einen Test- und Trainingsdatensatz. Der Testdatensatz soll einen Umfang von 20% aller Daten haben. Sie können dafür z.B. `sklearn.model_selection.train_test_split` verwenden.

b) Zur Klassifizierung soll ein Random Forest (`sklearn.ensemble.RandomForestClassifier`) verwendet werden. Einer der wichtigsten Hyperparameter ist die Anzahl der Bäume. Testen Sie in einer 5-fachen Kreuzvalidierung auf dem Trainingsdatensatz, welche der folgenden Einstellungen für dieses Problem am besten geeignet ist:

- n_estimators: 1
- n_estimators: 10
- n_estimators: 100

Nutzen sie als Kriterium die roc_auc, area under curve der receiver operating characteristic, (`sklearn.metrics.roc_auc_score`). Für die Kreuzvalidierung können Sie z.B. `sklearn.model_selection.cross_val_score` nutzen. Tipp: über den Parameter `n_jobs` kann das Fitten des Random Forests parallelisiert werden.

c) Erstellen Sie nun einen Random Forest Classifier mit der zuvor bestimmten Anzahl an Bäumen und trainieren das Model auf dem Trainingsdatensatz. Wenden Sie es anschließend auf den Testdatensatz an. Achten Sie dabei darauf `predict_proba` anstatt `predict` zu verwenden.

d) Evaluieren Sie nun die Performance des Klassifiers. Erstellen Sie dazu eine ROC Kurve und berechnen sie die AUC (`sklearn.metrics.roc_curve` und `sklearn.metrics.roc_auc_score`). Plotten Sie zudem die Verteilung der Gamma- und Hadronereignisse entlang der classification score. Interpretieren Sie die Ergebnisse: ist es möglich das Signal vom Untergrund zu trennen? Wie würden die Ergebnisse für einen perfekten Klassifizierer aussehen?

Aufgabe 31: *Data Mining Anwendung – Energie Rekonstruktion* ***BONUS*** **5 P.**

In dieser Aufgabe soll eine reale Problemstellung aus der Gammaastronomie gelöst werden. Es wird die Energie von Gamma-Events mittels eines Random Forests und eines Neuronalen Netzes durchgeführt.

Problemstellung

Die Problemstellung wird ausführlich in der Vorlesung besprochen. Hier gibt es außerdem Hinweise zu dem Datensatz.

Datensatz

Die Daten sind unter `https://factdata.app.tu-dortmund.de/smd/small` zu finden. Für diese Aufgabe werden beide Dateien benötigt. Die Datei `smd_deeplearning_gammas_reduced.hdf5` beinhaltet zwei Keys: charges und energy. Charges beschreibt die integrierte Ladung der jeweiligen Pixel des FACT-Teleskops für jedes gemessene Event. In dem key energy sind die Energien der Gammas gespeichert. Die Einheit ist GeV. Sie können die Daten mittels der Python-Bibliothek `h5py` einlesen:

```
import h5py

with h5py.File("smd_deeplearning_gammas_reduced.hdf5", "r") as f:

    energy = f['energy'][:]

    charges = f['charges'][:]
```

Die Datei `image_parameters_smd_reduced.hdf5` kann mittels `pandas.read_hdf` eingelesen werden. Dort befindet sich ein DataFrame mit verschiedenen Features und MC-Label. Die Energie der Gammas in Einheiten von GeV ist in der Spalte `corsika_event_header_total_energy` gegeben.

Hinweise

Benutzen Sie wie in der Vorlesung erwähnt die Python-Bibliotheken `sklearn` und `tensorflow`. Alles was in dieser Aufgabe verwendet werden muss, ist bereits in den gegebenen Bibliotheken implementiert. Erfinden Sie das Rad nicht neu!

a) Lesen Sie das DataFrame mit `pandas.read_hdf` aus der Datei `image_parameters_smd_reduced.hdf5` ein. Verwenden Sie im Folgenden nur Gamma Ereignisse (`corsika_run_header_particle_id` $== 1$) mit einer Energie über 500 GeV (`corsika_event_header_total_energy` > 500) . Erstellen Sie aus diesen Ereignissen die benötigten Label y (Energie der Gammas) und ein Set aus Features X, dass dann für die Energierekonstruktion genutzt werden kann. Wenden sie den 10-er logaritmhus auf die Energien an. Warum ist dies sinnvoll? Überlegen Sie sich welche Spalten verwendet werden dürfen. Teilen Sie nun die Labels y und Features X in einen Test- und Trainingsdatensatz. Der Testdatensatz soll einen Umfang von 20% aller Daten haben. Sie können dafür z.B. `sklearn.model_selection.train_test_split` verwenden.

b) Erstellen Sie einen Random Forest Regressor (`sklearn.ensemble.RandomForestRegressor`), der die Energie der Gammas rekonstruieren kann. Schätzen Sie dann die Energie für die Events des Testdatensatzes.

c) Evaluieren Sie nun das Ergebnis. Berechnen Sie dafür die Standardabweichung und den Mittelwert der Residuen $\Delta y = y_{\text{true}} - y_{\text{pred}}$ und vergleichen Sie diese mit der Standardabweichung der wahren Energien. Erstellen Sie zudem ein Histogramm der Residuen und ein Korrelationsplot der wahren und geschätzten Energien y_{true} und y_{pred}. Dies kann z.B. mittels `plt.hexbin(y_pred, y_true, mincnt=1)` erfolgen. Interpretieren Sie die Ergebnisse. Was bedeuten die einzelnen Ergebnisse/Plots? Wie würden diese für einen perfekten Energieschätzer aussehen?

Tipp: wenn Sie hierfür eine Funktion erstellen (z.B. `evaluate_performance(y_true, y_pred)`), dann verringert sich der Aufwand in späteren Aufgabenteilen.

d) Im Folgenden wird ein Neuronales Netz erstellt und anhand der integrierten Ladung der einzelnen Pixel die Energie geschätzt. Lesen Sie zunächst die Daten aus der Datei `smd_deeplearning_gammas_reduced.hdf5` ein. Berechnen Sie auch hier

den 10-er Logarithmus der Energie für die Label y. Führen Sie eventuell notwendige Datenaufbereitungsschritte und Transformationen für die charges durch. Teilen Sie anschließend die Daten in ein Trainings- und Testdatensatz wie oben bereits beschrieben.

e) Erstellen Sie nun ein convolutional neural network (CNN) zur Energierekonstruktion. Nutzen Sie dafür `tensorflow.keras` und die folgenden netzwerk layer:

```
conv_settings = {
'filters': 10,
'kernel_size': (5, 5),
'activation': 'elu',
'padding': "same",
}


layers = [
tf.keras.layers.Conv2D(input_shape=[50, 50, 1], **conv_settings),
tf.keras.layers.Conv2D(**conv_settings),
tf.keras.layers.MaxPool2D(pool_size=2),
tf.keras.layers.Conv2D(**conv_settings),
tf.keras.layers.MaxPool2D(pool_size=2),
tf.keras.layers.Conv2D(**conv_settings),
tf.keras.layers.MaxPool2D(pool_size=2),
tf.keras.layers.Flatten(),
tf.keras.layers.Dropout(rate=0.2),
tf.keras.layers.Dense(20, activation='elu'),
tf.keras.layers.Dense(20, activation='elu'),
tf.keras.layers.Dense(1, activation=None),
]
```

Das Netzwerk kann dann über `model = tf.keras.models.Sequential(layers=layers)` erstellt werden. Zur Minimierung soll der AdamOptimizer verwendet werden zusammen mit dem Mean Squarred Error (MSE) als loss function:
`model.compile(optimizer='adam', loss='mse')`.

f) Trainieren Sie nun das Netz für 15 Epochen mit einer batch size von 32 und einem Validierungssplit von 10%. Stellen Sie den Verlauf der loss-Kurve für den Trainings- und Validierungsdatensatz in einem plot dar. Interpretieren Sie die loss-Kurve.

Ist das Training konvergiert? Gibt es Overfitting? Falls ja, wie kann man dem entgegenwirken?

g) Schätzen Sie nun die Energie für den Testdatensatz und evaluieren Sie das Ergebnis, wie bereits für den Random Forest in Teilaufgabe c) beschrieben.

h) Trainieren sie nun ein vollverzweigtes (dense/fully connected) Netzwerk. Dabei sollten keine `Conv2D` layer verwendet werden. Stellen Sie auch hier die Loss-Kurve dar und evaluieren das Ergebnis. Was fällt Ihnen bezüglich des Trainingsverlaufs und der Anzahl der freien Parameter (`model.summary()`) im Vergleich zu dem CNN auf?

i) Vergleichen und interpretieren Sie die Ergebnisse des Random Forests und der beiden Neuronalen Netze. Kann der Random Forest auch zur Rekonstruktion der Energie auf Basis der charges benutzt werden?