Allgemeines

Die vorgesehene Bearbeitungszeit für das Übungsblatt endet am Freitag, 04.12.2020, um 12:00 Uhr. Kurze Zeit später werden die Lösung und das nächste Übungsblatt hochgeladen.

Falls Sie anonymes Peer-Feedback geben und erhalten wollen, laden Sie Ihre Lösung zu den Programmieraufgaben innerhalb der Frist in das ILIAS-Übungsobjekt "Abgaben" hoch. Feedback kann erst nach Ablauf der Abgabefrist gegeben werden. Geben Sie das Feedback bis zum Mittwoch, 09.12.2020, 23:55 Uhr. Erst nach Ablauf dieser Feedbackfrist wird dann das erhaltene Feedback sichtbar. Weitere Details zum Peer-Feedback finden Sie in der Datei "Feedback_geben.pdf" im ILIAS.

1 Online-Test 2

Der zweite Online-Test steht ab dem 23.11.2020, 00:00 Uhr im ILIAS bereit. Er beschäftigt sich mit den Vorlesungskapiteln 2 (Statistische Grundlagen) und 3 (Informatik-Grundlagen).

2 Hauptkomponentenanalyse (PCA)

Ziel dieser Aufgabe ist es, eine Principal Component Analysis (PCA) anzuwenden und die Ergebnisse zu interpretieren. Hierfür verwenden wir wieder den eingebauten Datensatz `iris`, welcher bereits vom ersten Übungsblatt bekannt ist.

a) **[Berechnung]** Berechnen Sie eine PCA für den `iris`-Datensatz ohne die Zielvariable `Species`. Nutzen Sie dafür die Funktion `prcomp()` und verwenden Sie deren eingebauten Parameter zur Normalisierung. Welche Komponenten hat das Ergebnisobjekt? Warum haben wir die Attribute überhaupt normalisiert? *(Basis)*

b) **[Erklärte Varianz]** Ermitteln Sie aus dem Ergebnisobjekt, welcher Anteil der Varianz von den einzelnen Hauptkomponenten bzw. kumuliert über die ersten k Hauptkomponenten abgedeckt wird. Plotten Sie die erklärte Varianz über die einzelnen Hauptkomponenten. Sie können hierfür entweder manuell einen Plot erstellen oder darauf vertrauen, dass die generische `plot()`-Funktion für das Ergebnisobjekt der PCA überschrieben wurde. Wie verteilt sich die erklärte Varianz und warum ist das so? *(Basis)*

c) **[Plot der Attribute]** Plotten Sie die ersten beiden Attribute der ursprünglichen Daten in einem Scatter Plot gegeneinander. Plotten Sie danach die ersten beiden Hauptkomponenten gegeneinander. Was fällt Ihnen auf? *(Basis)*

d) **[Korrelation]** Berechnen Sie die paarweise Korrelation der Attribute in den ursprünglichen Daten sowie nach der PCA-Transformation. Falls Sie eine graphische Korrelationsmatrix bevorzugen, können Sie das Paket `corrplot` verwenden. Warum erhalten Sie das Ihnen vorliegende Ergebnis? *(Vertiefung)*

e) **[MSE]** Implementieren Sie eine Funktion für die mittlere quadrierte Abweichung (MSE) zwischen zwei Matrizen: $MSE(x, \hat{x}) = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{i,j} - \hat{x}_{i,j})^2$ *(Vertiefung)*

f) **[Rekonstruktionsfehler]** Berechnen sie den MSE, wenn die Daten auf jeweils eine der Hautkomponenten und wieder zurück transformiert werden. Testweise können Sie auch auf die ersten k Hauptkomponenten transformieren, für verschiedene Werte von k. Vergleichen Sie das mit dem MSE, wenn jeweils ein Attribut der ursprünglichen Daten beibehalten und der Rest durch den Mittelwert ersetzt wird. *(Vertiefung)*

3 Entropie

Ziel dieser Aufgabe ist es, Entropie zu nutzen, um geeignete Attribute und Werte für eine Partitionierung des `iris`-Datensatzes zu finden. Eine vergleichbare Vorgehensweise findet sich beim Trainieren von Entscheidungsbäumen, was wir in der nächsten Übung betrachten.

a) **[Split-Entropie (1)]** Implementieren Sie eine Funktion `splitEntropy()`. Diese soll einen numerischen Attribut-Vektor `x`, einen kategorischen Klassen-Vektor `y` und einen numerischen Split-Punkt `splitPoint` erhalten. Die Funktion soll die Entropie des Splits berechnen (Formel siehe Vorlesung) und als Zahl zurückgeben. Nutzen Sie 2 als Basis für den Logarithmus. Sie können ihre Funktion mit folgendem Code testen:

```
x <- 1:5
y <- factor(c(1, 1, 1, 0, 0))
differences <- c(
  splitEntropy(x, y, 0) - -sum(c(3/5, 2/5)*log2(c(3/5, 2/5))),
  splitEntropy(x, y, 2.5) - -3/5*sum(c(1/3, 2/3)*log2(c(1/3, 2/3))),
  splitEntropy(x, y, 3.5) - 0
)
stopifnot(all(abs(differences) < 1e-10))
```

Warum verwenden wir überhaupt den letzten Befehl? *(Basis)*

b) **[Split-Entropie (2)]** Implementieren Sie eine Funktion `splitEntropyForData()`. Diese soll ein `data.frame` namens `dataset` und einen String namens `target` erhalten. Ihre Funktion soll die Split-Entropie für alle Attribute und alle sinnvollen Split-Punkte berechnen. Sie können davon ausgehen, dass alle Spalten bis auf die, deren Name mittels `target` übergeben wird, numerisch sind. Wählen Sie einen Rückgabetyp, der die nötigen Daten für die folgenden Aufgaben speichern kann. *(Basis)*

c) **[Plot]** Berechnen Sie die Split-Entropie für alle möglichen Attribute und Split-Punkte des `iris`-Datensatzes. Plotten Sie die Split-Entropie in Abhängigkeit des Split-Punktes für alle Attribute. Was schließen Sie aus den Ergebnissen? *(Vertiefung)*

d) **[Split-Bestimmung]** Finden Sie den besten Split für den `iris`-Datensatz. Betrachten Sie die Plots des Datensatzes von Übungsblatt 1 und überlegen Sie, ob der gefundene Split Sinn ergibt. *(Basis)*

4 Bring Your Own Theoretical Task

Denken Sie sich für die nächste Übungssitzung eine Frage aus, wie sie in der mündlichen Prüfung gestellt werden könnte. Sie können sich von der Art her an den möglichen Prüfungsfragen am Ende der Vorlesungskapitel orientieren. Inhaltlich sollte sich die Frage auf eines der Vorlesungskapitel 1 bis 3 beziehen. Vom Niveau her sollte die Frage nicht einfach Wissen 1:1 abfragen, sondern Verständnis fordern. Beispielsweise könnte es darum gehen, Sachverhalte zu vergleichen, einzuordnen, zu analysieren etc. Wir werden die Fragen in der Übungssitzung zunächst in Kleingruppen diskutieren und danach die interessantesten Fragen im Plenum besprechen. Sie müssen die Frage nicht zusammen mit ihrem Code hochladen.