Softwarewerkzeuge der Bioinformatik

Prof. Dr. Volkhard Helms
PD Dr. Michael Hutter, Markus Hollander,
Andreas Denger, Marie Detzler, Larissa Fey

Saarland University
Department of Computational Biology

Winter semester 2020/2021

Tutorial 10

4. Februar 2021

Data visualization & Machine Learning

In diesem Tutorial werden sie einige der Grundkonzepte von Data Science kennenlernen. Sie werden *pandas* benutzen um Daten zu manipulieren und visualisieren, und anschließend ihr eigenes Machine Learning Modell mit der *scikit-learn* library erstellen. Das dritte Projekt, was heute ausgegeben wird, baut auf den Methoden auf die in diesem Tutorial beschrieben sind, und geht dabei etwas mehr ins Detail.
Der Code der in diesem Tutorial benutzt wird ist auch als Textdatei auf der Website verfügbar, damit er einfach kopiert werden kann. Text der aus einer PDF kopiert wurde kann unerwünschte Symbole und Zeilenumbrüche enthalten wenn er in ein anderes Formular eingefügt wird.

Exercise 10.1: Vorbereitung

Für dieses Tutorium werden wir die Website kaggle.com benutzen. Kaggle ist ein soziales Netzwerk für data scientists, dass einfache Kollaboration ermöglicht, sowie das Teilen von Datensätzen und Analysen. Außerdem werden dort Wettbewerbe ausgetragen in denen bis zu sieben-stellige Preisgelder für die beste Analyse eines Datensatzes ausgegeben werden.
Vor allem können sie damit ihre eigenen Berechnungen in der Cloud durchführen, ohne zusätzliche Software zu installieren.

(a) Besuchen sie kaggle.com und erstellen sie einen Benutzeraccount. Sie können sich entweder mit einem Google-Account anmelden, oder mit ihrer Email-Adresse und einem Passwort. Im letzteren Fall bekommen sie eine Bestätigungs-Mail an ihre Adresse geschickt. Klicken sie auf den Link in der Mail um den Account zu aktivieren.

(b) Loggen sie dich ein und erstellen sie ihr erstes Notebook. Auf der linken Seite der Website finden sie die Option **Code** (mit dem Symbol <>). Klicken sie darauf, und wählen sie + **New Notebook**. Wählen sie *Python* als die Sprache aus, und *Notebook* als den *type*. Klicken sie auf *create*. Sie können auf den Namen des Notebooks oben links klicken um den Namen zu ändern. Wählen sie einen Namen den sie wiedererkennen, wie z.B. *SWW Tutorial 10*.

(c) Ihr Notebook ist nun in ihrem Benutzeraccount gespeichert. Dieser lässt sich von der Startseite aus über das Profilbild oben rechts erreichen. Dort findet sich das Notebook unter *Your Profile*, und dem Reiter *Notebooks*.

Exercise 10.2: Einführung in Jupyter Notebooks

Was sie nun vor sich haben ist ein *Jupyter Notebook*. Ein notebook besteht aus Zellen (*Cells*). Eine Zelle kann entweder eine *Code Cell* sein, welche Python-Code enthält, oder eine *Markdown cell*, welche geschriebenen Text beinhaltet, z.B. um zu erklären was sie in ihrer Analyse tun. Sie können neue *code cells* hinzufügen indem sie das + Symbol oben in der Leiste drücken. Code in Zellen kann ausgeführt werden durch Klicken auf den Pfeil neben dem Plus während die Zelle ausgewählt ist. Mit *Run All* können alle Zellen der Reihenfolge nach von oben bis unten ausgeführt werden.

(a) Ihr Notebook enthält bereits eine Zelle. Wählen sie diese aus, und drücken sie auf das Papierkorb-Symbol oben rechts an der Zelle um ihren Inhalt zu löschen, damit wir von vorne anfangen können.

(b) Schreiben sie *2+2* in eine leere Zelle, und klicken sie den Pfeil-Knopf. Das Resultat wird anschließend unter der Zelle angezeigt.

(c) Erstellen sie eine neue code cell mit dem + Symbol, schreiben sie *print("Hello World")* in die Zelle, und führen sie die Zelle aus. Die Zeichenfolge *Hello World* sollte nun unter der Zelle zu sehen sein.

(d) Sie können Werte zu Variablen zuweisen. Diese Werte sind anschließend in der Umgebung des Notebooks gespeichert und können später wieder benutzt werden. Schreiben sie *a=5* in eine leere code cell, und führen sie sie aus. Öffnen sie eine neue Zelle, und schreiben sie nur *a*. Dies wird ihnen den Wert anzeigen der in *a* gespeichert ist. Wenn die letzte Zeile einen Wert enthält oder zurückgibt, dann wird dieser Wert unter der Zelle angezeigt.

(e) Erstellen, editieren, starten und löschen sie ein paar Zellen, bis sie den Eindruck haben dass sie die Grundkonzepte von Notebooks verstanden haben.
Hier sind einige nützliche Tastenkürzel:

- STRG+Enter (Command+Enter auf Mac) führt die Zelle aus die gerade ausgewählt ist.
- Umschalt+Enter führt die momentan ausgewählte Zelle aus, und wählt anschließend die nächste Zelle aus. Gibt es keine nächste Zelle, wird eine neue erstellt.
- Mit ESC können sie in den *command mode* wechseln. In diesem Modus können sie mit den Pfeiltasten im Notebook navigieren, und zusätzliche shortcuts benutzen, wie z.B. *c* und *v* für kopieren und einfügen, *a* und *b* um eine neue Zelle über (above) oder unter (below) der derzeit ausgewählten Zelle einzufügen, oder *dd* um eine Zelle zu löschen. Sie können zurück in den *edit mode* wechseln indem sie Enter drücken.

Exercise 10.3: Laden des Datensatzes

In dieser Übung werden sie einen *Non-Small-Cell Lung Carcinoma* (NSCLC) Datensatz analysieren. Der GEO Identifier ist GSE74706. Der Datensatz enthält Samples von Krebspatienten, sowie von gesundem Gewebe als Referenz. Die Website Cumida sammelt Microarray Daten von GEO, und bereitet sie für Machine Learning vor. Indem wir Cumida's Version der Daten benutzen müssen wir vor dem Start der Analyse keine weiteren Vorbereitungen treffen.

(a) Klicken sie auf **+Add data** oben rechts. Wählen sie *Upload* oben rechts in dem Fenster, neben dem X. Wählen sie das zweite Symbol von oben auf der linken Seite. Nennen sie ihren Datensatz *lungcancer*. Kopieren die die folgende URL und fügen sie unter **Remote Files** ein:

https://sbcb.inf.ufrgs.br/data/cumida/Genes/Lung/GSE74706/Lung_GSE74706.csv

Klicken sie auf **+Add Remote Files**, dann auf **Create**. Der Upload wird ungefähr 1-2 Minuten dauern.

Alternativ können sie den Datensatz auch über den Link herunterladen, und ihn manuell in das Notebook hochladen.

(b) Nun werden wir den Datensatz in einen *DataFrame* einlesen, eine Datenstruktur die von dem *pandas* Paket für Python bereit gestellt wird. Ein DataFrame ist im Prinzip eine Tabelle mit Reihen und Spalten, ähnlich zu einer Excel-Tabelle oder einer SQL Datenbank. Hier ist eine Anleitung zu DataFrames.

(1) Zuerst müssen sie den Speicherort finden den die Datei auf dem Server hat. Oben rechts ist ein Ordner mit dem Name *Input*. Der Datensatz ist gespeichert unter *input* → *lungcancer* → *Lung_GSE74706.csv*. Hier könne sie den Pfad der Datei kopieren.

(2) Diese Zelle lädt die *.csv* Datei in einen DataFrame mit dem Name *lungcancer_df*:

```
import pandas as pd
file_path = "../input/lungcancer3/Lung_GSE74706.csv"
lungcancer_df = pd.read_csv(file_path)
```

(3) Erstellen sie eine neue Zelle, schreiben sie nur *lungcancer_df*, und führen sie sie aus. Nun wird die Tabelle darunter angezeigt. Wie viele Zeilen und Spalten hat sie?

(c) Als nächstes werden wir die Daten etwas aufräumen. In einem ersten Schritt speichern wir die *type*-Spalte (also ob es sich um ein Krebs- oder Referenz-sample handelt) in einer separaten Liste. Als nächstes erstellen wir einen DataFrame der nur die Genexpression enthält, also aus dem die Spalten *samples* und *type* entfernt sind. Erstellen sie eine Zelle die so aussieht:

```
labels_df = lungcancer_df["type"]
features_df = lungcancer_df.drop(["type", "samples"], axis=1)
```

(d) Schlussendlich werden wir einige Statistiken für die Daten berechnen, um einen Überblick zu bekommen.

(1) Zählen sie wie oft jedes label in dem Datensatz vorkommt:

```
labels_df.value_counts()
```

Wie *normal* und *NSCLC* (Lungenkrebs) samples befinden sich in dem Datensatz?

(2) Berechnen sie Statistiken für die Genexpressions-Daten:

```
features_df.T.describe()
```

Ein normalisierter Datensatz hat die Eigenschaft dass die Mittelwerte (mean) und die Standardabweichung (std) zwischen den Samples sehr ähnlich sind. Ist der Datensatz gut genug normalisiert oder ist eine Normalisierung notwendig?

(3) Erstellen sie ein Histogramm der durchschnittlichen Expression der Gene:

```
features_df.mean().hist()
```

Beschreiben sie kurz was sie auf dem Plot sehen.

Exercise 10.4: Lineare Support Vector Machine

Unser Ziel ist es, ein Machine Learning Prgramm zu schreiben dass auf samples trainiert wird bei denen bereits bekannt ist ob sie von einem Krebspatienten oder einem gesunden Menschen stammen. Sobald das Programm die Unterschiede in der Genexpression zwischen den zwei Gruppen gelernt hat, wird es in der Lage sein vorherzusagen ob eine bisher unbekannte Probe von einem Krebspatienten stammt oder von gesunden Zellen, basierend auf den Genexpressionsdaten dieser neuen Probe.

(a) Erstellen sie zunächst eine Zelle, die alle Funktionen und Pakete importiert die wir benötigen werden:

```
import numpy as np
from sklearn.svm import LinearSVC
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
from mlxtend.plotting import plot_decision_regions
```

(b) Die Machine Learning Pakete die wir benutzen werden setzen voraus dass die Daten in einem bestimmten Format vorliegen. Der DataFrame mit dem Expressionsdaten und die Liste der Labels müssen zu *numpy arrays* konvertiert werden, einer weiteren oft genutzten Datenstruktur. Die Labels, also *normal* und *NSCLC*, müssen zu Zahlen umgewandelt werden (0 und 1). Es ist eine gängige Konvention, die Tabelle mit den Features X zu nennen, und die Liste der Labels y:

```
X = features_df.to_numpy()
y = np.where(labels_df == "normal", 0, 1)
```

(c) Die Tabelle mit den Genen X hat derzeit 29148 Dimensionen pro sample, also eine pro Gen. Um einen zweidimensionalen Plot zu erstellen, der zeigt was der Algorithmus tut, muss die Dimension von 29148 auf zwei reduziert werden. Dafür werden sie eine der am häufigsten genutzten Methoden für die Reduzierung der Dimensionen verwenden: Principal Component Analysis (PCA). PCA funktioniert am besten wenn die Features vorher standardisiert wurden, was die *scale* Funktion erledigt:

```
pca = PCA(n_components=2)
X_std = scale(X)
X_pc2 = pca.fit_transform(X_std)
```

(d) Nun werden sie den Machine Learning Algorithmus trainieren. Eine lineare Support Vector Machine (SVM) versucht die optimale Hyperebene zu finden welche die Daten in zwei Bereiche unterteilt, basierend auf den Samples mit denen es trainiert wird. In diesem Fall wird die SVM eine gerade Linie, also eine Hyperebene im zweidimensionalen Raum, zeichnen. SVMs gehen davon aus dass die Daten standardisiert sind, also müssen wir erneut *scale* benutzen. Das Training geschieht durch die Funktion *fit*.

```
X_pc2 = scale(X_pc2)
svm = LinearSVC()
svm.fit(X_pc2, y)
```

(e) Erstellen sie einen Plot von der trainierten SVM und dem Datensatz:

```
plot_decision_regions(X=X_pc2, y=y, clf=svm)
```

Interpretieren sie den Plot. Funktioniert eine SVM gut auf diesen Datensatz?

Have fun!