Bioinformatik in der Arzneistoffforschung
im SoSe 2023

Blatt 3

Prof. Dr. Andreas Dominik

Spielregeln

- Die Aufgaben sollen im Praktikum und zuhause einzeln bearbeitet werden.
- Anwesenheit im Praktikum und Abgabe der Lösungen ist Pflicht für die Teilnehmer, des Kurs mit 6 CrP als WP-Modul im Bachelor.
  Für die Teilnahme am Methodenseminar mit nur 3 CrP (Masterstudiengang Bioinformatik und Systembiologie) ist die Teilnahme am Praktikum freiwillig.
- Die Lösung muss in Moodle abgegeben werden. Bitte als *ordentliches* tar-Archiv; d.h. als eine einzige Datei mit dem Namen `Nachnahme.Vorname_Blatt_Nummer.tar.gz`.
  Das Archiv soll nur ein Unterverzeichnis (`Nachnahme.Vorname-Blatt-Nummer/`) enthalten in dem alle Dateien und ggf. weitere Unterverzeichnisse zu finden sind.
- Die Übung wird in den Praktika bis zum **13. Juli 2023** besprochen und muss an diesem Tag bis spätestens 23:59h in Moodle hoch geladen werden. Da Moodle dann dicht macht können die Aufgaben weder nachträglich noch als e-Mail abgegeben werden.

Modellierung der Tryptaseaktivität von SMOLs (20)

1 Mastzelltryptase und Inhibitoren

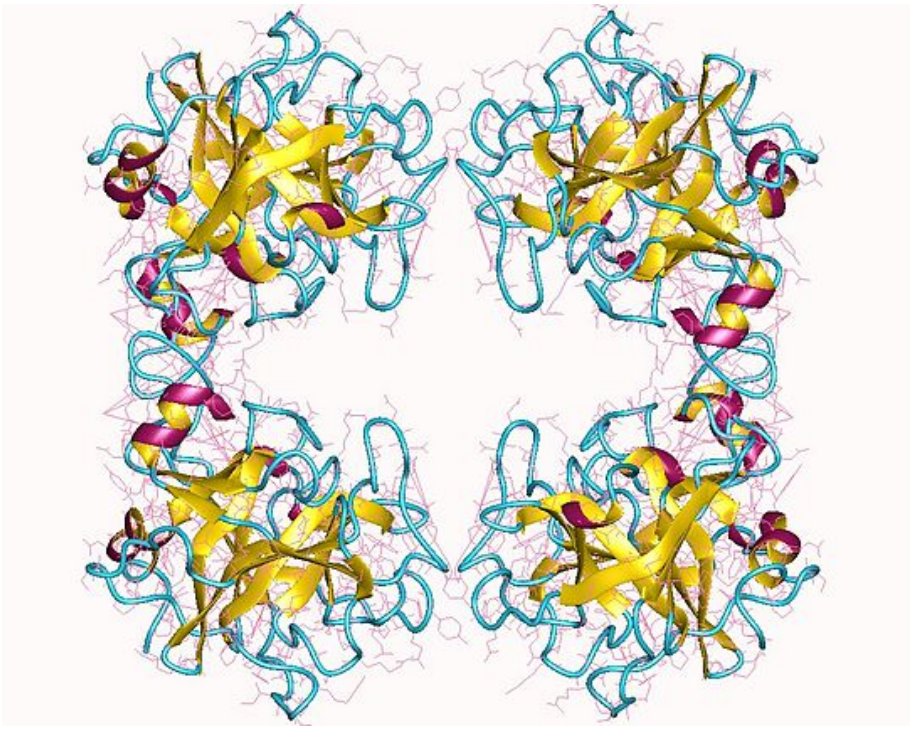
Die Tryptase ist eine ganz besondere Serinprotease, die hauptsächlich in Mastzellen gebildet wird und für die lawinenartige Aktivierung der Mastzellen verantwortlich ist. Sie ist tetramer aufgebaut und hat ihre 4 aktiven Zentren innenliegend in der Pore, was vermutlich für die sehr große Spezifität verantwortlich ist (PAR2-Rezeptor).

Durch Inhibition der Tryptase sollten sich überschießende Immunreaktionen (z.B. bei Autoimmunerkrankungen, Asthma, Morbus Crohn, Neuodermitis, usw.) regulieren lassen.

Die Active Site der humanen Mastzelltryptase hat große Ähnlichkeit zu anderen Serinproteasen (z.B. Trypsin, Thrombin). Der Wirkstoff des Blutegels (Hirudin) bindet auch an die humane Tryptase und es gibt seit ca. 20 Jahren ein optimiertes Peptid (Leech-derived tryptase Inhibitor).

Deshalb ist es erstaunlich, dass es bisher keine etablierten Tryptaseinhibitoren auf dem Markt gibt.

Das wollen wir ändern!



1lto

2 Technologie

Das Ziel der QSAR-Studie besteht darin, aus der Aktivität bekannter SMOLs mit Hilfe von Deep-Learning ein Modell zu optimieren, mit dem dann die Wirkung hypothetischer Moleküle vorhergesagt werden kann. Dazu benötigt man:

- Programmiersprache → Python und Jupyter
- Daten für die Inhibition → ChEMBL-Datenbank
- Chemische Molelüle in Python → RDKit
- Neuronale Netze → Tensorflow und Keras
- ~~CombiChem → NCGC Library Synthesizer~~
- Visualisierung → matplotlib

Installation

Python und so

Idealerweise sollte die Installation vieler möglicherweise von einander abhängiger Python-Pakete innerhalb eines virt. Environments gemacht werden. Deshlab:

Conda

- Download Anaconda-Installer: https://www.anaconda.com/download/
- `$ Anaconda-latest-Linux-x86_64.sh`
- Terminal window neu öffnen (python macht aus Linux fast ein Windows ;o)
- `$ conda list` für test
- ev. Conde aktualisieren: `$ conda update -n base -c defaults conda`

RDKit

- `$ conda create -c rdkit -n my-rdkit-env rdkit`
- `$ conda activate my-rdkit-env` fehler --> conda init bash --> restart git bash
- falls das nicht geht, ins Anaconda-Dir wechseln und: `$ source activate my-rdkit-env`

Jupyter einrichten

Da in jedem virt. Env. ein separater Python-Kernel rennt, der mit dem Rest der Rechners nichts tun haben will, muss ein vEnv seinen Kernel Jupyter zur Verfügung stellen. Im aktivierten vEnv muss der ipyKernel installiert und dann Jupyter konfiguriert werden:

- falls nicht schon aktiviert, `$ conda activate my-rdkit-env`
- `$ conda install ipkernel`
- `$ python -m ipykernel install --user --name=my-rdkit-env`
- Jetzt müsste es in Jupyter einen neuen Kernel my-rdkit-env geben, der in diesem vEnv rennt.

Tensorflow und Keras

Wir verwenden für das neuronale Netz eine möglichst einfache Installation:

- alles im vEnv!
- `$ conda install matplotlib`
- `$ pip install tensorflow` und ggf. `tensorflow-gpu`
- `$ pip install keras`

Dann kann die Installation im Notebook getestet werden:

```
import tensorflow as tf
hello = tf.constant('Hello, I am TensorFlow!')
tf.print(hello)

>>> b'Hello, I am TensorFlow!'

import keras
>>> Using TensorFlow backend.
```

Jupyter oder Skript

Jupyter-Notebooks eigenen sich perfekt dazu, neue Technologien kennen zu lernen und mit den Daten zu spielen. Für die eigentliche Anwendung sind später Skripte, die man auf der Kommandozeile aufrufen kann, praktischer. Idealerweise hat man ein `prepare`-Skript, das mit einem Datensatz aufgerufen wird und die Daten vorbereitet, ein `train`-Skript, das das neuronale Netz trainiert und ein `predict`-Skript, das pK_d-Werte für Moleküle vorhersagt.

Diese Skripte lassen sich dann auch leicht auf andere Rechner kopieren, um z.B. das Training auf einem GPU-Server auszuführen.

3 QSAR

3.1 Daten

Die Datenbank ChEMBL enthält Bindungsdaten von sehr vielen Arzneistoffen und Molekülen. Der Trainingsdatensatz wurde aud ChEMBL exportiert und enthält die chemische Struktur jedes Moleküls als SMILES-String, sowie die Dissoziationskonstante K_d für die Reaktion des Liganden *Lig* mit der Tryptase *Trp*:

$$\text{(Lig – Trp)} \rightleftharpoons \text{Lig + Trp}$$

Es ist

$$K_d = \frac{c(Lig) \cdot c(Trp)}{c(Lig - Trp)}$$

d.h. K_d ist umso kleiner, je mehr das Gleichgewicht auf der Seite des gebunden Liganden liegt. In der Praxis liegen die Werte zwischen 1 und 10^{-10} mol/l, wobei z.B. ein Wirkstoff mit $K_d = 10^{-9}$ mol/l als *nanomolar* oder mit $K_d = 10^{-6}$ mol/l als *mikromolar* bezeichnet wird. Da sich die Werte um Größenordnungen unterscheiden, ist es sinnvoll auf eine logarithmische Skala zu wechseln und den sog. pK-Wert anzugeben:

$$pK_d = -\log_{10} K_d$$

Ein nanomolarer Wirkstoff hat beispielsweise einen $pK_d = 9$.

Da die Zahl der in ChEMBL gelisteten Daten nicht allzu groß ist, werden für diese Aufgabe nur ein Trainings- und Validierungsdatensätze angegeben. Der finale Testdatensatz fehlt. Wer *sauber* arbeiten will, kann die Validierungsdaten teilen um einen getrennten Testdatensatz zu erhalten. Den Trainingsdatensatz sollte man nicht verkleineren, da dieser bereits um unteren Limit der Größe für Deep-Learning ist.

3.2 Vorbereiten der Daten mit RDKit

Die Vorgehensweise ist im Notebook `prepare.ipynb` ausgeführt! Nehmen Sie das Notebook als Grundlage und erweitern Sie es soweit notwendig.

Sie müssen zunächst die Trainings- und Validierungsdaten vorbereiten und Descriptoren berechnen.

Später müssen Sie diesen Workflow auch auf die hypothetischen Moleküle anwenden, die Sie als kombinatorischen Bibliothek aufgebaut haben.

3.3 MLP

Erste Schritte mit Keras

Der Einstieg ins Deep-Learning ist mit Keras sehr einfach. Es gibt fast unendlich viele Tutorials für jeden Geschmack. Meine Empfehlung ist mit `MNIST` und einem MLP zu beginnen (CNNs sind mächtiger, aber komplizierter und für den Anfang schwieriger zu verstehen). Das folgende Tutorial zeigt sehr schön, wie man schrittweise das Netz aufbauen kann, ohne nur wild herumzuprobieren:
`https://victorzhou.com/blog/keras-neural-network-tutorial/`

Wenn das funktioniert können Sie gerne auch noch ein CNN ausprobieren, z.B.
`https://keras.io/examples/mnist_cnn/`

Der `MNIST`-Datensatz ist aber sehr einfach, deshalb tauschen Sie ihm im nächsten Schritt durch den Datensatz `MNIST-fashion` aus. Dieser ist genau gleich aufgebaut, hat aber statt der Ziffern kompliziertere Bildchen von Kleidungsstücken.

MLP für unser Problem

Das Modell zur Vorhersage soll als multi-Layer-Perceptron (MLP) umgestzt werden. Die Vorgehensweise ist im Notebook `mlp.ipynb` ausgeführt! Nehmen Sie das Notebook als Grundlage und erweitern Sie es soweit notwendig.

Das MLP wird mit den Trainingsdaten trainiert und dann mit den Validierungsdaten getestet. Trennen Sie von den Validierungsdaten einen Teil als Testdatensatz ab. Als Ergebnis plotten Sie jeweils die vorhergesagte, gegen die tatsächliche Inhibition.

Optimierung des Modells

Da ein neuronales Netz sehr viele sog. *Hyperparameter* hat (Zahl der Neurone, Zahl der Schichten, Aktivierungsfunktion, Trainingsdauer, viele Arten der Regularisierung, ...), fällt es am Anfang nicht so leicht eine guten Parametersatz zu finden.

Eine wiederkehrende Idee ist beispielsweise mithilfe eines Grid-Search oder eines Optimierungsalgorithmus den Parameterraum abzutasten. Das ist aber weder vernünftig noch sinnvoll noch erfolgversprechend.

Ein schneller Weg zum optimalen Netz ist beispielsweise:

- Daten in Train, Validation und Test teilen (siehe
  `https://tarangshah.com/blog/2017-12-03/train-validation-and-test-sets/`
- immer die Accuracy (oder ein passendes Fehlermaß) für Trainings- und Validierungsdaten berechnen

  weder zu kurz nor zu lang
- immer den Verlauf beider Werte während des Trainings beobachten (zu kurzes Training ist nicht gut → unvollständiges Training; zu langes Training ist auch nicht gut → Übertraining)
- mit einem sehr kleinen *Spiel-Netz* beginnen
- das Netz schrittweise vergrößern, bis es nahezu 100% Accuracy auf den Trainingsdaten erreicht - dabei darf das Ergebnis für die Validierungsdaten schlecht sein

- erst jetzt vorsichtig regularisieren (*Dropout*-Schichten und *early-Stopping* sind einfach und genügen meist bereits)
- die Regularisierung schrittweise erhöhen, bis die Ergebnisse für Train und Vali gleich gut sind
- wenn das Ergebnis zu schlecht ist, Netz vergrößern und Regularisierung anpassen
- wenn Sie zufrieden sind, werden die Testdaten vorhergesagt.

4 Virtuelles Screening

Das Modell zur Vorhersage der Aktivität kann dazu verwendet werden, einen große Zahl hyothetischer Moleküle *in-silico* zu testen, um potentielle Wirkstoffe zu erkennen.

4.1 ZINC

Die ZINC-Datenbank enthält mehrere 100 Millionen Moleküle, die auch alle verfügbar sind (man kann sie z.B. bei einem Hersteller bestellen).

Der Einfacheit halber habe ich Ihnen eine große Textdatei zusammengestellt, in der die SMILES von knapp einer Milliarde Moleküle enthalten sind. Sie können diese von der Hessenbox herunterladen:

`hessenbox`

4.2 vHTS

Für das virtuelle Screening laden Sie SMILES mit Hilfe von RDKit, berechnen, wie für die Trainingsdaten die Deskriptoren und sagen die Aktivität für jedes Molekül voraus.

Dabei gibt es natürlich einiges zu beachten:

- Dabei müssen Sie vermutlich irgendwie tricksen, und batchweise arbeiten, weil alle Moleküle auf einmal sicher zu viel sind.
- Auch ist es eine gute Idee das neuronale Netz mit relativ großen Minibatches zu füttern, um einen vertretbaren Durchsatz zu erreichen.
- Sie können davon ausgehen, dass die Tryptaseinhibitoren, mit denen Sie trainiert haben zum großen Teil auch in der ZINC-Datenbank enthalten sind. Sie sollten diese ausfiltern (ich würde *nach* der Vorhersage filtern, weil es ist ja eine Art Qualitätskontrolle, dass die bekannten Inhibitoren auch im Screening gefunden werden).

Deliverables

1. Alle Pythonskripte oder Notebooks
2. Plots *Vorhersage* gegen *ist* für Train, Vali und Test
3. Liste der 100 besten Moleküle der ZINC-Datenbank mit vorhergesagter Aktivität und Moleülstruktur! (das kann man mit RDKit machen)