Protein Prediction I for Computer Scientists

Machine Learning & Secondary Structure Prediction

June 22\textsuperscript{nd}/27th, Summer Term 2017

Burkhard Rost & Lothar Richter

PP1CS SoSe 17
Lecture and exercise

- [link](https://www.rostlab.org/teaching/ss17/pp1cs)
- Announcements, slides and videos
- **Lecture** Tuesdays (10:00-11:30 am) and Thursdays (10:00 – 11:30 am)
- Room MW 1801 (Mechanical Engineering)
- **Exercise** Thursdays 12:30 – 14.00 pm
  - Room Hörsaal 3 (MI 00.06.011, Lecture hall 3) and mostly MW2250 on Tuesday 13-15

- **Register** for the lecture and exam in TUM online

PP1CS SoSe 17
Exercise

- Exercise wiki

  https://i12r-studfilesrv.informatik.tu-muenchen.de/sose17/pp4cs1/index.php/Main_Page
# Exercise – Topics and Schedule

<table>
<thead>
<tr>
<th>Slot</th>
<th>Thursday</th>
<th>Tuesday</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>May 4&lt;sup&gt;th&lt;/sup&gt;</td>
<td>May 9&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Structure of the Exercise / Biological Background</td>
</tr>
<tr>
<td>2</td>
<td>May 11&lt;sup&gt;th&lt;/sup&gt;</td>
<td>May 16&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Biological background</td>
</tr>
<tr>
<td>3</td>
<td>May 18&lt;sup&gt;th&lt;/sup&gt;</td>
<td>May 23&lt;sup&gt;rd&lt;/sup&gt;</td>
<td>Protein structures</td>
</tr>
<tr>
<td>4</td>
<td>Jun 1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>Jun 13&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Alignments</td>
</tr>
<tr>
<td>5</td>
<td>Jun 8&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Jun 20&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Resources for Biological Information / Formats</td>
</tr>
<tr>
<td>6</td>
<td>Jun 22&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>Jun 27&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Machine Learning incl. Tricks / Secondary Structure Prediction</td>
</tr>
<tr>
<td>7</td>
<td>Jun 29&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Jul 4&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Homology Modeling / Prediction of Other Protein Features</td>
</tr>
<tr>
<td>8</td>
<td>Jul 6&lt;sup&gt;th&lt;/sup&gt;</td>
<td></td>
<td>Wrap Up – Questions</td>
</tr>
</tbody>
</table>

**WED**   | **Jul 12<sup>th</sup>**  | **EXAM**   |

PP1CS SoSe 17
Ideas

- A machine learning device can generalize from real world observations into a “formal” model.
- Each model reflects only a few aspects of reality.
- No model can completely represent the reality, i.e. a photograph of a dog remains a photograph and not a real dog.
- The model should reflect a concept or commonalities and not individual characteristics.
Inductive Bias

- every learning scheme discards some aspects of reality to construct a model
- this may differ between different learning schemes
- this might also already happens on the level of feature extraction, i.e. choosing the types and values to represent an observation
- this is not to be mistaken with the predictive bias
Some vocabulary

- learning scheme: a specific learning algorithm producing a model like decision trees, rule based systems, SVMs, Bayesian networks, etc.
- attribute/feature: a variable describing a specific aspect of real world observations, like body weight, color, certain property found yes/not
- instance: a single observation describing an observed event by assigning values for each feature used to represent this observation
Some vocabulary II

- training: phase of analyzing real world observations in a formalized representation to derive parameters and/or internal structure

- test phase: phase of model application to determine the reliability of statements (predictions) on instances not used for training

- label: an attribute selected to be predicted
Types of Learning

- depending on the presence of a label we distinguish between supervised and unsupervised learning
- unsupervised learning: concept learning, frequent item sets, clustering
- supervised learning: everything with labeled data which allows to make a prediction
Data Preprocessing

Before you can start to build a model in most cases the data are subjected to various preprocessing steps:

- Feature Extraction
- Discretization
- Feature Selection
Feature Extraction/Construction

- Conversion of observation records into a formalized, computer-readable representation
- Definition of an attribute type
- Assignment of appropriate attribute values
- This implies a strong involvement of the analyst
- Important: common sense, background knowledge from expert domains
Feature Selection

- remove values from instances, i.e. discard some features of a data set because these are:
  - irrelevant
  - redundant
  - noisy/faulty

- possible benefits:
  - improve efficiency and accuracy
  - prevent overfitting
  - save space
Feature Selection Strategies

- unsupervised: based on domain knowledge, random sampling

- supervised:
  - measures consider the class (filtering): Gini-index, information gain, relief, ...
  - use a learning scheme’s performance (wrapping):
    - select the set of attributes which leads to best performance
    - forward selection: increase the set of attribute 1 by 1
    - backward elimination: decrease the set of attributes 1 by 1
Machine Learning and Bioinformatics

- today biology has to span between two extremes:
  - statements on the nucleotide level (one level below genes)
  - statements the individual/population level on the other hand
- the gain in speed to generate sequence data (nucleotide sequences) has clearly outpaced the speed of analysis and knowledge discovery
- current lab technology even cannot fill the gap between sequence and structure
Role of DM / ML

Data Mining helps to:

- structure the data and compress the data
- filter out mistakes and outliers because of experimental errors and other noise
- reduce redundancy
- replace wet lab analyses with predictions
- detect interesting relationship and models and directs man power towards points where it is needed

PP1CS SoSe 17
Overview of the Steps in KDD

Data mining is a step in the KDD process that consists of applying data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns (or models) over the data. Note that the space of patterns is vast, and the goal is to find patterns that are valid, novel, useful, and understandable.

The discovered patterns should be valid on new data with some degree of certainty. We also want patterns to be novel (at least to the system and preferably to the user) and potentially useful, that is, lead to some benefit to the user or task. Finally, the patterns should be understandable, if not immediately then after some postprocessing.

The previous discussion implies that we can define quantitative measures for evaluating extracted patterns. In many cases, it is possible to define measures of certainty (for example, estimated prediction accuracy on new data) or utility (for example, gain, perhaps in dollars saved because of better predictions or speedup in response time of a system). Notions such as novelty and understandability are much more subjective. In certain contexts, understandability can be estimated by simplicity (for example, the number of bits to describe a pattern). An important notion, called interestingness (for example, see Silberschatz and Tuzhilin [1995] and Piatetsky-Shapiro and Matheus [1994]), is usually taken as an overall measure of pattern value, combining validity, novelty, usefulness, and simplicity. Interestingness functions can be defined explicitly or can be manifested implicitly through an ordering placed by the KDD system on the discovered patterns or models.

Given these notions, we can consider a pattern to be knowledge if it exceeds some interestingness threshold, which is by no means an attempt to define knowledge in the philosophical or even the popular view. As a matter of fact, knowledge in this definition is purely user oriented and domain specific and is determined by whatever functions and thresholds the user chooses.

Data mining is a step in the KDD process that consists of applying data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns (or models) over the data. Note that the space of patterns is vast, and the goal is to find patterns that are valid, novel, useful, and understandable.

The discovered patterns should be valid on new data with some degree of certainty. We also want patterns to be novel (at least to the system and preferably to the user) and potentially useful, that is, lead to some benefit to the user or task. Finally, the patterns should be understandable, if not immediately then after some postprocessing.

The previous discussion implies that we can define quantitative measures for evaluating extracted patterns. In many cases, it is possible to define measures of certainty (for example, estimated prediction accuracy on new data) or utility (for example, gain, perhaps in dollars saved because of better predictions or speedup in response time of a system). Notions such as novelty and understandability are much more subjective. In certain contexts, understandability can be estimated by simplicity (for example, the number of bits to describe a pattern). An important notion, called interestingness (for example, see Silberschatz and Tuzhilin [1995] and Piatetsky-Shapiro and Matheus [1994]), is usually taken as an overall measure of pattern value, combining validity, novelty, usefulness, and simplicity. Interestingness functions can be defined explicitly or can be manifested implicitly through an ordering placed by the KDD system on the discovered patterns or models.

Given these notions, we can consider a pattern to be knowledge if it exceeds some interestingness threshold, which is by no means an attempt to define knowledge in the philosophical or even the popular view. As a matter of fact, knowledge in this definition is purely user oriented and domain specific and is determined by whatever functions and thresholds the user chooses.
ML Tools employed in Bioinformatics and Cooccurrence of Methods

taken from “The rise and fall of supervised machine learning techniques”


PP1CS SoSe 17
Possible Explanations the Prevalence of ANNs and SVMs

- they are capable to handle a huge number of attributes
- they are quite robust against uninformative features
- they implicitly adjust feature weights during the training phase
- they work sufficiently well
Possible Explanations the Prevalence of ANNs and SVMs

- You do not need to have an idea about the meaning of an input
- i.e. no background knowledge or understanding for feature selection or even stronger for feature generation necessary
- Disadvantage: These methods are “black box” models, so inspecting the model does not really increase your knowledge/understanding
How is Machine Learning Influenced by Underlying Assumptions?

- there are a number of assumptions in the various processing steps
- the performance depends on that these assumptions hold
- very often we cannot really check or proof if this is true
Background distribution

- we assume that the background distribution is uniform
- i.e. the underlying source emits instances with constant probabilities over time
- possible solutions:
  - use many features to represent complex scenarios
  - use stream mining algorithms which update parameters
Insufficient Model Complexity
Unfair Sampling

- due to “experimental” reasons the sample represents only a special subset of the entities
- especially difficult for lazy learning methods like k-nearest-neighbors
- possible solutions:
  - remove redundancy
  - use stratification
  - check variance and identify difficult instances
Redundancy Reduction – More Reasons
Redundancy Reduction

- the collection of the data is typically governed by a specific research task
- the sampling of the “global” distribution is not fair
- models try to minimize the error OVER ALL instances
- stay “local” with your predictions (know your limits)
- apply redundancy reduction to make the data a “fair” sample
Redundancy Reduction


Suitable Representation

- we assume that the selected feature set can represent the concept to learn
- since we often do not know causal relationships we might mistake employed features with real causing ones
- e.g. number of storks and births in Germany
- e.g. opening umbrellas and rain
- remedy: use background knowledge, careful interpretation
Representation of the Concept

<table>
<thead>
<tr>
<th>Weather</th>
<th>Temperature</th>
<th>Wind</th>
<th>playTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>rainy</td>
<td>high</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>dry</td>
<td>medium</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>medium</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>dry</td>
<td>medium</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

- with this attributes it is really hard to learn favorable conditions for playing tennis
- we unconsciously assume that these attribute are sufficient to describe the scenario
### Representation of the Concept

<table>
<thead>
<tr>
<th>Weather</th>
<th>Temperature</th>
<th>Wind</th>
<th>Buddy</th>
<th>playTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>rainy</td>
<td>high</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>dry</td>
<td>medium</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>medium</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>dry</td>
<td>medium</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

- actually there no proof to show whether important attributes are missing or not
- for this aspect background knowledge and experience are most important
Performance Estimation

- training a model means discarding and keeping some information from the observed instances
- depending on the learning scheme instance specific information is stored too
- instance specific information leads to overfitting
- overfitting: a prediction model is biased towards the training examples, i.e.:
  - better performance on training examples
  - worse performance on new instances
More Realistic Estimations

- most optimistic estimation: Resubsitution error (determine performance on the set completely used for training)
- if you have a lot of data: determine the error on an independent test set
More Realistic Estimations II

- LOOCV: Leave one out cross validation
  - always one example is hold out for testing, the remaining for training
  - n iterations with n instances, final result is the average
  - still quite biased
  - use to check the influence of individual instances
  - if you have a small number of instances
More Realistic Estimations III

- n-fold cross validation, typically n=10
  - partition the data in n partition
  - use n-1 partitions for training
  - use 1 partition for performance assessment
  - repeat with a different hold-out partition
  - average performance

- every step where class information is considered has to be included in the loop! (done after partitioning)
Artificial Neural Networks

- still most prevalent machine learning scheme in bioinformatics
- typically Feed-Forwards Multi-Layer-Perceptrons

- Error Back-Propagation
  ([http://home.agh.edu.pl/~vlisi/Al/backp_t_en/backprop.html](http://home.agh.edu.pl/~vlisi/Al/backp_t_en/backprop.html))
Artificial Neural Networks

- different activation functions

- consider the number of free parameters in respect to the number of available training instances

- determine the number of epochs
Artificial Neural Networks

- both too many free parameters (edges) as well as overtraining leads to overfitting
- initialize the weights with random values from the linear region of the activation function
- repeat several times to avoid getting stuck in local minima
- learning the weights AND determine the optimum number of epoch belong to the training phase (s.t. referred as training and cross training or training and validation)
Use case

- Develop a predictor for Proteases belonging to a certain fold (3D-structure)
- Check PDB for respective entries (structure & function annotation)
- search database for similar sequences
- sanity check: predict structure elements / check function annotations
- => compile positive training set
Use case

- compile an appropriate negative training set with
  - some fold but different function
  - same function but different folds
  - in real life problems: this is a major challenge
- decide about the coding (features) and recode your data set
- train you method
- evaluate your method
- estimate stability/confidence
Overfitting

- performance feature (undesired)
- detection:
  - performance on test instances is significantly worse than on training instances
- reasons:
  - too high model complexity (too many parameters)
  - too many training epochs (neural networks)
Class Imbalances

- learning schemes tend to minimize the prediction error over all instances
- if the positive class is small then errors on the positive does matter anymore (false negatives)

solutions:
- oversample minority class
- downsample the majority class
- assign weights to the different error types
HSSP Curve

Distance from new HSSP-curve

Sequence identity implies structural similarity!

Don't know region

Distance from curve = +10

Distance from curve = -10

C Sander & R Schneider 1991 Proteins 9:56-69
B Rost 1999 Prot Engin 12, 85-94

© Burkhard Rost (TUM Munich)

Tuesday May 6, 2014

PP1CS SoSe 17