Active and self-learning in a biological screening task

Jan-Thorsten Peter
Active and self-learning in a biological screening task
Submitted by Jan-Thorsten Peter to the University of Skövde as a dissertation towards the degree of M.Sc. by examination and dissertation in the School of Humanities and Informatics.

08.10.2009
I hereby certify that all material in this dissertation which is not my own work has been identified and that no work is included for which a degree has already been conferred on me.

Signature: ____________________________________________
# Contents

1 Introduction 3  
1.1 Research question ........................................... 4  
1.2 Research Method ........................................... 4  
1.3 Overview ..................................................... 4  
2 Background 5  
2.1 Classifier types ............................................. 5  
2.2 Evaluation criteria ......................................... 5  
2.3 Data .......................................................... 6  
3 Calculating a Baseline 7  
3.1 Bayesian Logistic Regression ................................. 8  
3.2 Random Forest ............................................... 8  
3.3 Fast Artificial Neural Network ............................... 8  
3.4 Results ....................................................... 9  
3.5 Combination Results ....................................... 12  
4 Self Learning 13  
4.1 Idea .......................................................... 13  
4.2 Classifier based self-learning ............................... 14  
4.3 Ensemble based self-learning ............................... 15  
5 Active Learning 16  
5.1 Idea .......................................................... 16  
5.2 Retraining neural networks .................................. 16  
5.3 Training new neural networks .............................. 18  
6 Summary and conclusion 20
Abstract

The focus of this master thesis is to analyze the effectiveness of self-learning and active-learning on biological virtual screening. Virtual screening is used in the pharmaceutical industry to increase the effectiveness of biological screening. Self-learning is a technique where parts of the classified test data are reused for training the classifier again. Active learning gives the classifier the possibility to select certain parts of the test data to use them as additional training data. The experiments in the thesis show that both methods can be used to improve the precision of the virtual screening process, but active-learning is more effective due to the additional information that is provided.

1 Introduction

For drug development in the pharmaceutical industry, it is of high importance to find active compounds. To reach this goal, high-throughput screening is used, where a high number of compounds is tested for being active or inactive. This means that the compounds have to be created physically in the laboratory and then tested, if they are active or not. This method is extremely inefficient since the vast amount of tested compounds turn out to be inactive, e.g. the data sets used during this study had less than 1% active compounds. To make this process more efficient, virtual screening is used. Virtual screening preselects the tested data by calculating the probability of a compound to be active or not. In the end, only the compounds with a high probability of being active will be screened. It has been shown that virtual screening is an effective way to increase the efficiency of this screening process using this preselection (Stahura and Bajorath, 2004). Developing methods that increase the accuracy of predicting if a compound is active or inactive directly reduces the costs for the screening process by preventing to test the compounds with a low probability of being active. This saves time and money.

Many methods in this area focus on nearest neighbors approaches with different distance measurements between the compound (Willett, 2006) or fusion methods between those distance measurements (Whittle et al., 2006). In this master thesis, the focus is set on methods based on logistic regression, random forests and artificial neural networks and how the two methods of self-learning and active-learning will perform in this task. Self-learning is a method where a trained classifier classifies the test data and uses the examples with the highest probability of being correct. This can lead to a way of enlarging the space with "known" training examples, but also bears the risk of enforcing errors made earlier in the training. Previous work on self-learning has been done in the area of text recognition (Yarowsky, 1995; Riloff et al., 2003).

In the active-learning method, the classifier can choose a few test samples to be tested. With the new information, the classifier can now try to improve the classification of the other test samples. Even though this method seems to be more costly than self-learning, since some tests have to be run for the active training itself, this method does not necessarily produce new costs. In the here used method to choose the compounds for active learning
only compounds in the upper 5% range where chosen, a range that would have been tested
anyway. This scenario is close to reality. It does not change the development process of
the pharmaceutical companies very much to run these few tests and return the results to
the classifier. Cohn et al. (1996) showed in his work some methods of selecting data for
active learning.

1.1 Research question

The research question in this thesis is to find out, if it is possible to increase the performance
of the virtual screening filtering by either using self-learning methods or the active learning
method. If it is possible, how much will the method increase the performance? A related
research question here is which data should be selected for self and active-learning.

1.2 Research Method

To test self-learning, first a reasonable way to classify the data with standard methods has
to be found. This is done by training different classifiers with the training set and then
testing how well this classifiers performs by classifying the test set. Different settings are
tested for each classifier. These results constitute the baseline to which the improvements
of self and active-learning are compared to.

Two different test setups are used to test self learning. The different settings that were
tested are described in the self learning section. The results of the self learning method
are then compared to the previous found baseline. This whole process is repeated for the
active learning method.

1.3 Overview

The master thesis is organized as followed. In the next section (2) some information on
the data used and an explanation of the used evaluation criteria are given. The following
section (3) describes the used classifiers and the results without self and active learning
processes. Next in section 4, the self-learning idea is described and the test results for the
different data sets are presented. In section 5 is the active learning idea presented in the
same way as self-learning.

The thesis ends with a conclusion and an outlook for possible future research (section
6).
2 Background

2.1 Classifier types

In this thesis three different types of classifiers are used. These are Bayesian Logistic Regression, Random Forest and Artificial Neural Network, which are briefly explained in this section.

2.1.1 Bayesian Logistic Regression

Bayesian Logistic Regression is a method to analyze dependences between variables. Using the given data \((Y_i, X_i), i = 1, \ldots, n\), where \(Y_i\) is a binary outcome, here if the compound is active or inactive, \(X_i\) the feature vector, here the different features for each compound, \(n\) is the number of given examples and \(x_{i,j}\) is the \(j\) feature of \(X_i\). To create the predictions a logistic curve is calculated using the test samples. This curve can be described by the formula: 

\[
f(X_i) = \frac{1}{1 + e^{-z}} \quad \text{and} \quad z = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \ldots
\]

2.1.2 Random Forest

The second classifier here used is a random forest. The idea of a random forest is to take many randomly created decision trees (Breiman et al., 1984) and combine them to an ensemble. A random decision tree selects in a random order features and tries to separate the different training samples by these features. The samples selection used for training is also made randomly. It has been shown that the combination of many random trees performs much better in average than a single tree (Breiman, 2001), even if each tree by itself performs very weak.

2.1.3 Artificial Neural Network

The development of ANN’s has been inspired of how the brain works. An ANN is built up with a number of perceptron, which emulate the neurons in a real neural network, which are connected in a specific way. Each perceptron has a certain threshold and weights on the input signals, which define the output signal of the perceptron. The most common types of ANN’s are feedforward neural networks (Hagan and Menhaj, 1994), which are also here used. Perceptrons are sorted in layers in these networks. The outputs of the perceptrons in the current layer are used as inputs in for the next layer. The input of the network is given by the features of each compound while the classification if defined by the output of the last layer. Artificial Neural Networks have long history starting as early as 1958 (Rosenblatt, 1958).

2.2 Evaluation criteria

Two different evaluation criteria are used to measure the performance of the different classification techniques, like logistic regression, self learning or active learning. These
criteria are the ROC (Fawcett, 2006) and the recovery rate at 5%. Both performance measurements are not based on the error rate, but on the ordering of the classifications. The reason why these performance measurements are used is that the data used in virtual screening is extremely unequally distributed regarding the relation between actives and inactive compounds. It is quite typical for this kind of task to have only 1-5% of active compounds, the in this thesis used data had only 1% active compounds. Simply classifying every compound as inactive would therefore already result in a very good error rate of only 1% without doing any calculations. Using measurements that consider the ranking of the classified compounds results in a far more useful representation of the performance and probably also reflects the actual task of filtering out a small subset better. Since the thesis is closely related to the actual use in the industry, where it is only possible to test a certain amount of compounds, the focus is set on the recovery rate at 5%, which is explained in a moment.

Area under curve is based on the receiver operating characteristic (ROC) curve (Fawcett, 2004). The ROC can be represented by plotting the fraction of true positives vs. the fraction of false positives. The maximal result is 1.0 which is equal to having all active compounds listed at the beginning of the ranking. The minimum is 0.0 which is obviously equal to the exact opposite, having all inactive compounds ranked at the top of the scale. A random classification is expected to result in an AUC of about 0.5.

The here used recovery rate at 5% is defined as the ratio between the number of active compounds in the top 5%, i.e., the 5% most likely active according to the estimated probabilities, of ranked test data divided by the number of active compounds in all the test data.

\[
\text{Recovery rate at 5%} = \frac{\text{active compounds in the top 5%}}{\text{active compounds in test data}}
\]

The maximum and perfect result is 100% and the minimum is 0%. A random distribution is expected to result in a recovery rate that is close to the percentage of active compounds in the test set. The here used limitation of 5% is only sensible, if maximally 5% of the test compounds are active, otherwise it would be impossible to reach the highest possible rate of 100% recovery.

Both evaluation criteria assume that the used classifier is able to rank the compounds in a meaningful way.

2.3 Data

Five different data sets were provided by AstraZeneca to test the different methods on them. These sets are all based on data collected by biological screening. The set labels are: 12455, 31420, 37110, 71522 and 78331. Each data set was divided into different feature sets named ecfi, fgf1, oeselma.gc and foyfi. Each feature set contained different descriptors/features of the compounds. Feature and set names are only used as an identifier, it is not important for this study where the names came from.

The easily observable differences (Table 1) between the data sets are the proportion of active compounds to inactive compounds and as later becomes clear some data sets are
much easier to classify than others.

<table>
<thead>
<tr>
<th>Set</th>
<th>ecfl</th>
<th>fgfl</th>
<th>oeselma gc</th>
<th>foyfi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature type</td>
<td>binary</td>
<td>binary</td>
<td>continuous</td>
<td>binary</td>
</tr>
<tr>
<td># of Features</td>
<td>1024</td>
<td>424</td>
<td>215</td>
<td>1024</td>
</tr>
</tbody>
</table>

Table 2: Data Sets

<table>
<thead>
<tr>
<th>Data Set</th>
<th>12455</th>
<th>31420</th>
<th>37110</th>
<th>71522</th>
<th>78331</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active training examples in %</td>
<td>1.49%</td>
<td>1.17%</td>
<td>1.23%</td>
<td>0.92%</td>
<td>0.68%</td>
</tr>
<tr>
<td>No. of active training examples</td>
<td>161</td>
<td>126</td>
<td>133</td>
<td>97</td>
<td>73</td>
</tr>
<tr>
<td>No. of inactive training examples</td>
<td>10600</td>
<td>10637</td>
<td>10634</td>
<td>10431</td>
<td>10538</td>
</tr>
<tr>
<td>No. of training examples</td>
<td>10761</td>
<td>10763</td>
<td>10767</td>
<td>10528</td>
<td>10611</td>
</tr>
<tr>
<td>Active test examples in %</td>
<td>0.81%</td>
<td>0.72%</td>
<td>0.69%</td>
<td>0.51%</td>
<td>0.39%</td>
</tr>
<tr>
<td>No. of active test examples</td>
<td>1339</td>
<td>1203</td>
<td>1141</td>
<td>851</td>
<td>647</td>
</tr>
<tr>
<td>No. of inactive test examples</td>
<td>163714</td>
<td>163932</td>
<td>163990</td>
<td>164250</td>
<td>164371</td>
</tr>
<tr>
<td>No. of test examples</td>
<td>165053</td>
<td>165135</td>
<td>165131</td>
<td>165101</td>
<td>165018</td>
</tr>
</tbody>
</table>

The preprocessing of the data is kept fairly simple. All features which contained only the same value for all compounds are removed. In the oeselma gc set also every feature is linear normalized to values between 0 and 1. This is done with the formula: \[ x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \]
where \( x_{min} \) equals the minimal value of all compounds for that feature, \( x_{max} \) the maximal value for that feature and \( x_i \) the value of the this feature for the current compound. The outliers are removed my simply reducing the 0.5\% highest values to the highest value without them, and the 0.5\% lowest values to the lowest value without them. and outliers are removed. This was not necessary for the other data sets, since they contained only binary data, which has no outliers and is also per definition between 0 and 1. At the end, all 4 feature sets were combined to a merged set.

The provided data is split in to a small set for the training of the classifiers and a much larger set for testing the results. The training set consists of about 6.5\% of all available examples. This reflects the real situation in pharmaceutical companies where the amount of training data tends to be much smaller compared to the amount of test data.

3 Calculating a Baseline

To get a first impression of how easy or hard the data is to classify, the data is classified with three different classifiers. An implementation of Bayesian Logistic Regression (Genkin et al., 2009), a random forest implementation (Narsky, 2009) and a neural network implementation (Nissen, 2009) are used for that.
3.1 Bayesian Logistic Regression

The Bayesian Logistic Regression implementation BBR from Alexander Genkin, David D. Lewis, and David Madigan is freely available on the internet. The implementation is well documented and easy to use. A more detailed explanation of the algorithm and the implementation can be found in Genkin et al. (2007).

3.2 Random Forest

As a Random Forest implementation, the freely available implementation of the StatPatternRecognition (Narsky, 2009) packet is used. Due to problems compiling the latest version, the previous version SPR-08-02-00 is used.¹

3.3 Fast Artificial Neural Network

For the neural network implementation, the FANN library and own scripts are used. These scripts use the python interface of the FANN library to train neural networks and classify the compounds. This makes it easy to use the neural networks in a very flexible way.

In comparison to the other classifiers, the artificial neural network implementation has obviously problems with the fact that the relation between the active and inactive compounds are extremely unbalanced. When using the standard training methods, the resulting network classifies nearly every test compound with the lowest possible probability of being an active compound. This makes it basically impossible to sort the test compounds in any meaningful way and therefore results in an extremely bad recovery rate and AUC score.

The algorithm that trains the neural network was adapted in order to change this behavior. The modified training algorithm overweights the active example such that the neural network is trained by as many active compounds as inactive compounds. This is done by splitting the training data into an active and an inactive set. For every training iteration the following steps are executed:

- Choose with a fair chance (50%/50% random choice) if an active or inactive compound is used for training
- According to the foregoing decision take a random compound from the active or inactive training samples
- Train the network with the chosen compound

Due to the extremely unequal distribution of active and inactive samples, the set with active training samples will be much smaller than the set with inactive samples. Therefore

¹This implementation created many problems. E.g. the temporary file needed by this implementation used in the worst case about 12 GB of the hard drive. This is not easy to handle, if an university computer with space limitations is used.
is the chance for an active compound of being used repeatedly much bigger than for an inactive compound. This heavy overweighting of the active compound results in much more usable probabilities returned by the neural networks.

In the creation and training of these neural networks always many random factors are involved, like setting the initial weights or choosing how often and which compounds are used for training. To reach a stable and comparable result it is therefore necessary to compute the average of more than one neural network. One way to do this is by either taking the average over the resulting recovery rate and AUC score or by taking the average over each classification. It has been shown that the combination of classification results can result in a significant improvement of the overall classification compared to the best single classification (Dietterich, 2000). The idea is that bad classifications from one neural network are balanced out by the classifications of the other neural networks. Therefore, the combined classification results of the neural networks were used to calculate the recovery rate and AUC of the neural networks. Every execution of the script will give a slightly different result, even though these actions stabilize the outcomes a bit. With a higher number of neural networks the variance of the results will decrease, but this will also result in a longer computation time.

One test was to remove badly trained neural networks from the ensemble before classifying the test data. The training set was divided into two sub sets. One with 90% of the data for training and another one with 10% of the data for preselection. It turned out that the performance did not change compared to the normal training without preselection. This is probably due to the extremely small preselection set. For the data set 78331 for example only 8 active compounds and 1054 inactive compounds are used in the preselection set. Of course, this is not enough to get a representative sample of the data. Therefore the whole preselection process was too strongly based on chance to be able to improve the results significantly.

### 3.4 Results

In this section the results of the experiments with the three classifiers: Fast Artificial Neural Network (FANN), Bayesian Logistic Regression (BBR) and Random Forest (RF) are presented. The results are shown separately for each data set. The first number in each field of the tables represents the recovery rate of this classifier in percent, while the second number represents the AUC. The results listed in the FANN row are always based on an ensemble of 30 networks. The results of the Random Forest of 30 random trees.

#### 3.4.1 Data Set 12455

No classifier performed always better than the others on every feature set, but FANN seems to perform best. FANN performs best in 4 out of five sets including the merged set where it reached the overall best performance for this data set. Even though the FANN seems to have performed better, this does not mean that the FANN is superior to the BBR since the results are very close together especially at the merged set. The better score of the
FANN can be the result of the normal variance of the neural networks. The random forest on the other hand did perform clearly worse than the two other classifiers.

Table 3: Data Set 12455

<table>
<thead>
<tr>
<th>Classifier</th>
<th>fgf1</th>
<th>ecfi</th>
<th>foyfi</th>
<th>oeselma_gc</th>
<th>merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>FANN</td>
<td>65.50% / 0.914</td>
<td>82.60% / 0.930</td>
<td>72.60% / 0.935</td>
<td>73.11% / 0.941</td>
<td>85.06% / 0.966</td>
</tr>
<tr>
<td>BBR</td>
<td>50.63% / 0.889</td>
<td>75.65% / 0.948</td>
<td>78.86% / 0.952</td>
<td>65.64% / 0.913</td>
<td>84.46% / 0.967</td>
</tr>
<tr>
<td>RF</td>
<td>50.11% / 0.836</td>
<td>64.75% / 0.879</td>
<td>56.91% / 0.878</td>
<td>68.04% / 0.865</td>
<td>71.47% / 0.914</td>
</tr>
</tbody>
</table>

3.4.2 Data Set 31420

This data set turned out to be the easiest one to classify. This is easy to see since the maximum and average scores are higher than in the other sets. FANN and BBR performed very similar. For some feature sets was BBR was the better classifier and in some cases BBR. The Random Forest performed best for the fgf1 set, but was otherwise outperformed by FANN and BBR.

Table 4: Data Set 31420

<table>
<thead>
<tr>
<th>Classifier</th>
<th>fgf1</th>
<th>ecfi</th>
<th>foyfi</th>
<th>oeselma_gc</th>
<th>merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>FANN</td>
<td>86.95% / 0.967</td>
<td>94.26% / 0.982</td>
<td>93.35% / 0.977</td>
<td>93.59% / 0.981</td>
<td>97.76% / 0.992</td>
</tr>
<tr>
<td>BBR</td>
<td>81.71% / 0.959</td>
<td>96.84% / 0.992</td>
<td>96.50% / 0.987</td>
<td>89.35% / 0.976</td>
<td>98.33% / 0.993</td>
</tr>
<tr>
<td>RF</td>
<td>87.94% / 0.958</td>
<td>90.27% / 0.948</td>
<td>94.82% / 0.948</td>
<td>89.60% / 0.961</td>
<td>89.03% / 0.954</td>
</tr>
</tbody>
</table>

3.4.3 Data Set 37110

The data set turned out to be nearly as easy to classify as the previous data set. Even though the maximal score is with 95.44% slightly lower. Also as before, the results of BBR and FANN are close together, but FANN performs slightly better. The results of the Random Forest are again worse than the results of the other classifiers.

3.4.4 Data Set 71522

This set turned out to be the hardest to classify correctly. The best classification is with 70.51% far behind the the classifications of the previous data sets and a bit worse than the following one. This time the artificial neural networks performed best on every feature set, while BBR was again better than the Random Forest.
Table 5: Data Set 37110

<table>
<thead>
<tr>
<th>Classifier</th>
<th>fgf1</th>
<th>ecfi</th>
<th>foyfi</th>
<th>oeselma_gc</th>
<th>merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>FANN</td>
<td>82.12% / 0.959</td>
<td>92.99% / 0.968</td>
<td>86.42% / 0.972</td>
<td>74.21% / 0.950</td>
<td>95.44% / 0.986</td>
</tr>
<tr>
<td>BBR</td>
<td>75.46% / 0.943</td>
<td>92.81% / 0.983</td>
<td>87.81% / 0.976</td>
<td>67.10% / 0.911</td>
<td>95.35% / 0.989</td>
</tr>
<tr>
<td>RF</td>
<td>73.00% / 0.918</td>
<td>82.12% / 0.931</td>
<td>68.19% / 0.949</td>
<td>74.82% / 0.906</td>
<td>84.84% / 0.954</td>
</tr>
</tbody>
</table>

Table 6: Data Set 71522

<table>
<thead>
<tr>
<th>Classifier</th>
<th>fgf1</th>
<th>ecfi</th>
<th>foyfi</th>
<th>oeselma_gc</th>
<th>merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>FANN</td>
<td>48.06% / 0.851</td>
<td>66.39% / 0.875</td>
<td>66.98% / 0.909</td>
<td>50.12% / 0.862</td>
<td>70.51% / 0.916</td>
</tr>
<tr>
<td>BBR</td>
<td>43.71% / 0.842</td>
<td>60.98% / 0.875</td>
<td>64.51% / 0.897</td>
<td>43.64% / 0.813</td>
<td>68.39% / 0.900</td>
</tr>
<tr>
<td>RF</td>
<td>38.43% / 0.719</td>
<td>57.58% / 0.832</td>
<td>50.41% / 0.838</td>
<td>53.41% / 0.819</td>
<td>64.27% / 0.895</td>
</tr>
</tbody>
</table>

3.4.5 Data Set 78331

The last data set again confirms the results so far that FANN and BBR performed about equally well, while the Random Forest constantly performed worse than the other classifiers. This data set was after the set 71522 the second hardest to classify.

Table 7: Data Set 78331

<table>
<thead>
<tr>
<th>Classifier</th>
<th>fgf1</th>
<th>ecfi</th>
<th>foyfi</th>
<th>oeselma_gc</th>
<th>merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>FANN</td>
<td>42.50% / 0.864</td>
<td>69.55% / 0.926</td>
<td>67.85% / 0.910</td>
<td>48.45% / 0.900</td>
<td>72.80% / 0.940</td>
</tr>
<tr>
<td>BBR</td>
<td>41.42% / 0.860</td>
<td>65.84% / 0.927</td>
<td>68.00% / 0.923</td>
<td>52.01% / 0.900</td>
<td>73.26% / 0.945</td>
</tr>
<tr>
<td>RF</td>
<td>39.57% / 0.784</td>
<td>59.35% / 0.830</td>
<td>57.81% / 0.836</td>
<td>47.37% / 0.815</td>
<td>63.21% / 0.945</td>
</tr>
</tbody>
</table>

3.4.6 Conclusion

The fast artificial neural network and the Bayesian Logistic Regression perform about equally well on the data. The Random Forest on the other hand showed most of the time the worst performance of all classifiers. Constantly through all tests did the merged feature set perform better than the others. This is not surprising since it probably contains more information than all other feature sets. Due to the large amount of features also the training and classification of the test set took more time than training and classification of the other, smaller feature sets. Boström (2007) showed that using more feature sets by fusing them can improve the resulting classifier.

It has also been shown that there is a clear ordering of the data sets in the way how difficult they are to classify. The sets 31420 and 37110 performed with a recall rate of
nearly 100% in the top five percent by far the best. At the other end there are the two data sets 71522 and 78331. Both data sets only got a maximal recall of slightly over 70%. The data set 12455 with a maximal recall of about 85% is in the middle between of the other sets.

Using all classifiers on all data sets each time consumes a lot of calculation time. Therefore the focus here is on the most promising sets and classifiers for the self learning and active learning tests. The two sets 78331 and 71522 have the most undiscovered active compounds in this test field and therefore have the most possibilities to be improved. The neural networks performed well on this data and it is possible to simply train the neural network with the new data without the need of retraining the whole neural network. For these reasons the focus here lies on these two data sets and FANN as a classifier for the self learning and active learning experiments.

### 3.5 Combination Results

Combining different kinds of classifier often improves the final result compared to the best result of a single classifier (Dietterich, 2000). To test this, the results of the three classifiers in every possible combination were combined by taking the average of the classification over all used classifiers. The following table shows that the combination of the two weaker classifiers, Random Forest and Bayesian Logistic Regression, does surprisingly perform better than the combination of the two best classifiers, FANN and BBR. The Random Forest very effectively improved the classifications, which leads to the assumption that the classifications made by the Random Forest strongly diverge from those calculated by FANN and BBR. The small improvement of the FANN and BBR combination compared to just FANN is a sign that the probabilities of these two classifiers are very alike compared to the random forest. This assumption is also supported by the very close results of these two classifiers in the previous tests.

The further experiments are only based on the neural networks ensemble for the earlier mentioned reasons of performance and simplicity, even though the BBR and RF combination outperforms it. This experiment was only run on the 71522 data set, since it wasn’t the focus of this study.

<table>
<thead>
<tr>
<th>Combination</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>64.28 % / 0.895</td>
</tr>
<tr>
<td>BBR</td>
<td>68.39 % / 0.901</td>
</tr>
<tr>
<td>FANN</td>
<td>70.86% / 0.908</td>
</tr>
<tr>
<td>BBR and FANN</td>
<td>70.98% / 0.907</td>
</tr>
<tr>
<td>FANN and RF</td>
<td>71.33% / 0.923</td>
</tr>
<tr>
<td>BBR, RF and FANN</td>
<td>72.27% / 0.920</td>
</tr>
<tr>
<td>BBR and RF</td>
<td>72.50% / 0.917</td>
</tr>
</tbody>
</table>
4 Self Learning

4.1 Idea

The idea behind self-learning is that a once trained classifier is used to classify the test set and the predictions that are most likely correct are then used for further training of the classifier. Predictions are considered to have a high probability of being correct if the probability returned by the classifier is close to 0 for inactive compounds or close to 1 for active compounds.

Self-learning has been shown to improve the outcome of the classifications quite significantly in different domains like natural language processing e.g. Yarowsky (1995) or Riloff et al. (2003).

The idea is that the same approach also should work with virtual screening. Compounds that are very likely to be active are used as additional active training samples while compounds that are very likely to be inactive are used as additional inactive samples for additional training of the neural networks. While this hopefully enlarges the amount of useful training samples, this method does also bear the risk of enforcing previously made errors.

The problem with virtual screening and self-learning is that we also have only very few active samples in the test data. This creates the problem that even when using only the 1% of compounds that are most likely to be active, still more inactive than, active examples are found. So the data added by the self learning algorithm is often not correctly classified and is therefore adding false information to the classifier. This would probably be a smaller problem with data where the active and inactive samples are more equally distributed.

Two self-learning methods that are considered here are the classifier based self-learning and ensemble based self-learning. The main difference between these two methods is that in the case of classifier based self-learning, the classifications from one classifier are used to train this classifier again, while in the other case, the classification of an ensemble of neural networks is used to train a completely new ensemble of neural networks.

In the case of classifier based self-learning, the test data is classified by each classifier. The classifications from one neural network are then used to continue the training of this neural network. All so trained neural networks are then combined to one ensemble, which is now used to classify the test data once more.

An alternative approach is to use a whole ensemble to get a first classification of the test data and then select the self-learning samples out of this ranking, different to the first approach where only one classifier was used for the ranking. Then, a new classifier is trained with these new self-learned training samples and the original training samples. With this approach the classifiers will most likely learn less wrongly classified self learning samples, but at the same time it bears the risk of loosing some diversity between the different classifiers. Using the classifications of the ensemble carries some of the information learned by the other classifiers back to each single classifier. This will probably improve the performance of each single classifier, but it interferes with the idea of having different
classifiers that even out each other’s errors.

The compounds here used for self learning are always chosen from the compound that have the highest probability of being correct and active. This is done by first ordering the compounds according to their probabilities and then selecting different slices from the test data. In the next step, one slice from this data is selected and used for self learning. The possible slices used here are always chosen from the most likely compounds. The smallest slice consist of the 0.2% of compounds with the highest probability of being active, while the slice with the most compounds used in this experiment contained the 1% with the highest probability of being active.

### 4.2 Classifier based self-learning

In this experiment ensembles of 5 neural networks were trained. Each single one was then used to classify the test data and chose the new active training samples based only on it is own classifications out of a certain training range. The same network was then again trained using additionally the new self-learned samples. The classifications of the single networks were then combined in the ensembles for the final classification. Each experiment was repeated twice, with new random seeds, to make the results more reliable.

In the following tables, the first row shows how many training iterations are used for retraining the neural networks, the first column defines the size of the used self-training range. E.g. 0.6% means that every compound between 0% and 0.6% is used as new active training sample.

#### 4.2.1 71522

The results turned out to be extremely unstable anyhow. Some networks improve after self-training iterations, some get worse. There is no obvious constant trend in these results. The most likely cause for this is that the networks are too much influenced by the random factors that are involved in training. This means that 5 networks are not enough to return stable results. It is not possible to increase the number of networks used at this point due to technical limitations during this study.

<table>
<thead>
<tr>
<th>Iterations:</th>
<th>0</th>
<th>30000</th>
<th>60000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2%</td>
<td>67.86% /0.882</td>
<td>69.68% /0.882</td>
<td>69.21% /0.878</td>
</tr>
<tr>
<td>0.4%</td>
<td>69.85% /0.883</td>
<td>69.33% /0.890</td>
<td>68.74% /0.877</td>
</tr>
<tr>
<td>0.6%</td>
<td>69.33% /0.869</td>
<td>70.15% /0.906</td>
<td>69.62% /0.894</td>
</tr>
<tr>
<td>0.8%</td>
<td>70.03% /0.893</td>
<td>69.91% /0.895</td>
<td>70.09% /0.893</td>
</tr>
<tr>
<td>1.0%</td>
<td>68.44% /0.850</td>
<td>70.68% /0.858</td>
<td>68.39% /0.889</td>
</tr>
</tbody>
</table>
4.2.2 78133

This data set enforces the previous assumption mentioned that random factors have a high impact on the test results in this setup using so few networks.

Table 10: Test results classifier based self learning for data set 78133

<table>
<thead>
<tr>
<th>Iterations:</th>
<th>0</th>
<th>30000</th>
<th>60000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2%</td>
<td>72.72% /0.890</td>
<td>72.48% /0.916</td>
<td>72.48% /0.896</td>
</tr>
<tr>
<td>0.4%</td>
<td>69.86% /0.853</td>
<td>67.31% /0.864</td>
<td>68.00% /0.860</td>
</tr>
<tr>
<td>0.6%</td>
<td>70.32% /0.837</td>
<td>68.77% /0.823</td>
<td>72.33% /0.855</td>
</tr>
<tr>
<td>0.8%</td>
<td>70.71% /0.870</td>
<td>70.32% /0.873</td>
<td>71.79% /0.877</td>
</tr>
<tr>
<td>1.0%</td>
<td>72.64% /0.887</td>
<td>70.86% /0.915</td>
<td>72.25% /0.916</td>
</tr>
</tbody>
</table>

4.3 Ensemble based self-learning

The other self-learning method used is based on the classifications made by the whole ensemble. Since this time the initially trained networks can be reused for all different settings, it was here possible to increase the number of used network without losing to much calculation time. 60 neural networks are used in the first ensemble. After the test data has been classified by this first ensemble, the compounds that are most likely to be active are added as active examples to the training data for the second ensemble. Only active samples were added to the training data since the amount of inactive training samples is already very big. The impact of adding a few more inactive training samples is therefore expected to be much smaller than the expected impact of adding new active compounds. The learning range shows the percentage of the classified test data that is used for self-learning.

After adding the self learned training samples to the training data a new ensemble was trained. This time only ten networks are used due to performance reasons. The training of one network on the merged set takes about one and a half hours on a Intel Core2 Quad CPU with 2.66GHz and 4 GByte Ram.

The self-training shows clear improvements of the classification results compared to the networks without self-learning, also there is a notably drop at 0.2%. This is probably due too to strong over-fitting at this point. The recovery rate on the data set 71522 improves by up to 3.3% and up to 1.2% on the data set 78331. It is interesting that at the same time, the AUC reduces on the data set 78331. Also worth mentioning is the behavior of the AUC in the 71522 data set. At the learning ranges from 0.4% to 0.6%, the AUC changes in the opposed direction of the recovery rate.
Table 11: Test results ensemble based self learning

<table>
<thead>
<tr>
<th>Learning range:</th>
<th>71522</th>
<th>78331</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0%</td>
<td>70.74% /0.876</td>
<td>72.80% /0.920</td>
</tr>
<tr>
<td>0.2%</td>
<td>69.09% /0.843</td>
<td>71.56% /0.851</td>
</tr>
<tr>
<td>0.4%</td>
<td>73.03% /0.892</td>
<td>73.72% /0.901</td>
</tr>
<tr>
<td>0.6%</td>
<td>71.68% /0.896</td>
<td>74.03% /0.916</td>
</tr>
<tr>
<td>0.8%</td>
<td>71.00% /0.911</td>
<td>72.72% /0.911</td>
</tr>
<tr>
<td>1.0%</td>
<td>72.20% /0.908</td>
<td>72.95% /0.910</td>
</tr>
</tbody>
</table>

5 Active Learning

5.1 Idea

The idea behind active learning is to first test a certain amount of the data that needs to be classified before classifying all the data. The additional information can be used to retrain the used classifiers and thereby improve the overall classification results. It is interesting to not only find the compounds with the highest probability to be active, but also which of these compounds will improve the final classification the most when tested first.

This method is obviously not fairly comparable with the previously used methods since it makes use of information unavailable to the other methods. Nevertheless it is still worth seeing how much the classifications can improve using this new possibility, to find out if active-learning is worth using or not.

Different approaches, how to select data for active learning are discussed by Cohn et al. (1996). The here tested method, to see if it improves the results compared to not using self-learning, is to choose different slices from the test data with high probabilities of containing many active compounds.

5.2 Retraining neural networks

The first approach is to train 5 neural networks in an ensemble and classify the test data with it. In the next step a range from the test data is chosen, which should be used as additional training data. As possible ranges to be chosen for training are the the top 6% of the classified test data splinted into slices each containing 1% of the data. Top x% means in this context the x% that are most likely to be active. So the first possible range contains all compounds in the top range from 0% to 1%, the second range contained everything between 1% and 2% and so on. Using slices from 0% to x% like in the self-learning experiment are not useful here, since the object is to find the best compounds to be used when given a fix number of tests. In this case we are allowed to test 1% of the data.

The same neural networks used for the first classifications were now trained with the new training data. After every 30000 training iterations, an evaluation on the test data was run to see how well the ensemble performed at that point. The results are based on running
the same experiment three times. As final results the average of all three experiments is used.

5.2.1 71522

The main problem with these experiments is that the results obtained from the trained networks are already largely different from each other even before the first active learning step, as can be seen in the column with 0 Iteration steps. This makes the comparison between the different ranges hard, but it is still possible to analyze the improvements made by using active learning.

The biggest improvement happens in the range 0% to 1% going from 0 iterations to 60000 iterations. The results drop again starting with 90000 iterations, which is a strong sign of overfitting of the neural networks.

While most ranges show some improvements, the results of 4% to 5% and 5% to 6% even get worse than the result without active-learning at some points. A possible reason for this could be that the neural networks are overfitting due to the additional training steps. Especially in these areas, not many new active compounds are found and therefore the new iterations do not add much new information, but simply add more iterations which probably lead to overfilling. Since the whole performance of these tests barely beat the previous tested classifying methods and the results vary a lot, this setup is probably not suitable for testing this method. Using only five networks in one ensemble seems to be too unstable as already shown in the self-learning experiments. Using a higher amount of neural networks is not possible due to the test setup and computational restrictions during this study.

Table 12: Test results classifier based active learning for data set 71522

<table>
<thead>
<tr>
<th>Iterations:</th>
<th>0</th>
<th>30000</th>
<th>60000</th>
<th>90000</th>
<th>120000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%-1%</td>
<td>65.62%  /0.842</td>
<td>71.56% /0.864</td>
<td>73.03% /0.879</td>
<td>71.91% /0.864</td>
<td>71.26% /0.875</td>
</tr>
<tr>
<td>1%-2%</td>
<td>66.74% /0.835</td>
<td>70.56% /0.838</td>
<td>66.80% /0.780</td>
<td>67.33% /0.791</td>
<td>69.21% /0.810</td>
</tr>
<tr>
<td>2%-3%</td>
<td>71.68% /0.887</td>
<td>72.38% /0.807</td>
<td>71.79% /0.803</td>
<td>73.56% /0.820</td>
<td>72.15% /0.809</td>
</tr>
<tr>
<td>3%-4%</td>
<td>67.13% /0.850</td>
<td>72.46% /0.852</td>
<td>68.70% /0.802</td>
<td>70.15% /0.811</td>
<td>68.07% /0.798</td>
</tr>
<tr>
<td>4%-5%</td>
<td>66.21% /0.826</td>
<td>66.09% /0.801</td>
<td>67.92% /0.818</td>
<td>65.68% /0.796</td>
<td>65.68% /0.796</td>
</tr>
<tr>
<td>5%-6%</td>
<td>64.10% /0.816</td>
<td>63.57% /0.781</td>
<td>66.03% /0.799</td>
<td>63.86% /0.783</td>
<td>68.91% /0.825</td>
</tr>
</tbody>
</table>

5.2.2 78133

Just like in the data set 71522 the results vary again a lot and make it therefore hard to interpret the results. The first range from 0% to 1% resulted in the biggest increase of the recovery rate. This suggests that the huge amount of new active compounds found for training overweights knowledge gained by using compounds from a lower ranges.
5.3 Training new neural networks

Retraining the neural networks showed some structural weaknesses of the experiment setup, therefore the active training test was rerun with a different setup.

This time the first classification is based on a much bigger ensemble. For the second step, the old networks are not used and trained again, but new neural networks are created and trained with the new training data. The ensemble for the first classification was created out of 60 neural networks, the same way as was already done for the self learning earlier. For the second step, a smaller ensemble of 5 networks was used. For the final classification only the small ensemble was used. To make the results more reliable, each experiment was run 3 times and the average of the results are presented.

As a second baseline, additional to the previously used baseline of the best performance without any active learning, it is also tested how the classifiers perform when adding just randomly chosen compounds to the training data. This is done by selecting 1% of the training data and then training 5 new neural networks just as is done for the active learning. In the following table this baseline is called ”Random Selection”.

The results this time are quite impressive. For both data sets the recovery rate is boosted up to well over 80%. Using the top 1% shows clearly the best results. This reinforces the assumption from the retrained neural networks that new active training

<table>
<thead>
<tr>
<th>Set</th>
<th>71522</th>
<th>78133</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>70.74% /0.876</td>
<td>72.80% /0.920</td>
</tr>
<tr>
<td>Random Selection</td>
<td>72.60% /0.858</td>
<td>74.50% /0.861</td>
</tr>
<tr>
<td>0%-1%</td>
<td>83.41% /0.940</td>
<td>84.87% /0.927</td>
</tr>
<tr>
<td>1%-2%</td>
<td>73.28% /0.865</td>
<td>81.31% /0.904</td>
</tr>
<tr>
<td>2%-3%</td>
<td>79.18% /0.895</td>
<td>72.67% /0.847</td>
</tr>
<tr>
<td>3%-4%</td>
<td>80.38% /0.905</td>
<td>77.81% /0.899</td>
</tr>
<tr>
<td>4%-5%</td>
<td>75.31% /0.874</td>
<td>76.37% /0.877</td>
</tr>
</tbody>
</table>

The results this time are quite impressive. For both data sets the recovery rate is boosted up to well over 80%. Using the top 1% shows clearly the best results. This reinforces the assumption from the retrained neural networks that new active training
compounds are most important for the training of the classifiers. Significant is also that in both data sets the results are getting much worse after the first high results. The data set 71522 drops in the range from 1% to 2% down to a recovery rate of 73.28%, but recovers again in the range from 2% to 3% to 79.18% and even to 80.38 for the range 3%-4%. The same is true for the data set 78133, which drops down to 72.67% at the range from 2% to 3%, but recovers at the range from 3% to 4% back to 77.81%.

This seems to be the area where the information gain from choosing compounds from an area of high uncertainty starts to overweight the advantage of finding more active compounds in the slightly higher areas.
6 Summary and conclusion

After setting a baseline with well known classifier approaches like Bayesian Logistic Regression, Random Forests and neural networks it was clear that the data set 71522 and 78331 would be the hardest to classify and thereby the best sets for testing improvements. The best recovery rate (70.51%) for the 71522 was reached with a ensemble of neural networks. Using a combination of the Bayesian Logistic Regression and Random Forest it was possible to boost this result a bit more up to 72.50%. This marked the baseline for further experiments.

The experiments with self-learning showed that using the right learning range, here from 0% to 0.4%, can improve the classification results significantly by up to 3.3%. This approach also showed weaknesses of using a neural network ensemble with a to small number of networks involved. Besides this problem the training of new networks seems to be the cleaner approach compared to retraining the previous used network for the first classification.

The first group of experiments based on the active-learning approach showed the same weaknesses as the classifier based self-learning approach did. It showed again too much variation in the results to draw conclusions from them. Nevertheless, was it still possible to see that using the compounds with the highest probability of being active can result in the highest increase of the recovery rate.

The second group of experiments with active-learning shows far more stable results than the first experiment. It shows clearly that using the upper 1% for active learning gives the biggest improvements of the results. Interesting to see is that the improvements get much smaller when using a range directly after the first high, but increases again significantly for later ranges. This could be a result of choosing compounds for active learning, where the classifiers are very unsure about and thereby using compounds that are able to add more information to the classifiers than compounds where the classifier is already very sure about there nature. That the highest improvement is reached in the top area is easily explained by the nature of the data used. The training set has only a very small amount of active compounds it can learn from, the data set 78331 has for example only 73 actives compounds in the training data. Adding the top 1% of the test data adds 380 new active compounds to the training data. This means that in the second training step more than 6 times as many active compounds were available compared to the first training step. This largely increased amount of active training samples makes it easier for the classifier to learn the patterns which determine if a compound is active or not and this leads to better classification results.

These results show that the intuitive method of choosing the compounds with the highest likelihood of being active, like it is already done in the pharmaceutical industry by normal virtual screening, also results in the best results for active learning.

Future work maybe lie in combining the self-learning and active-learning to improve the results even more. Using self learning to classify the test data, then using active-learning and then again self-learning could result in even higher results. Another approach could be using sequential active-learning, where after each training step a new region is chosen.
for active learning until the maximum amount of tested data is reached. It is also possible
to think about different ways of choosing the compounds used for active-learning. It could
be possible to use compounds that are the furthest away from the known training samples,
which could result into a bigger knowledge gain.

Acknowledgment

I like to thank at this point Ulf Norinder, Henrik Boström and Ronnie Johansson for
providing me with the data from AstraZeneca needed for these studies, for giving me
many useful ideas for this research and reviewing this thesis.
References


