



Hands-on tutorials of Δ vina, Gnina and DeepChem

Pei-Ying Chu (朱珮瑩)

Supervisor: Jung-Hsin Lin (林榮信)

Research Center for Applied Sciences, Academia Sinica

2018 Frontiers in Computational Drug Design, Academia Sinica, March 16-20, 2018

Improving Scoring-Docking-Screening Powers of Protein–Ligand Scoring Functions using Random Forest

Cheng Wang^[a] and Yingkai Zhang^{*[a,b]}

The development of new protein–ligand scoring functions using machine learning algorithms, such as random forest, has been of significant interest. By efficiently utilizing expanded feature sets and a large set of experimental data, random forest based scoring functions (RFbScore) can achieve better correlations to experimental protein–ligand binding data with known crystal structures; however, more extensive tests indicate that such enhancement in scoring power comes with significant under-performance in docking and screening power tests compared to traditional scoring functions. In this work, to improve scoring-docking-screening powers of protein–ligand docking functions

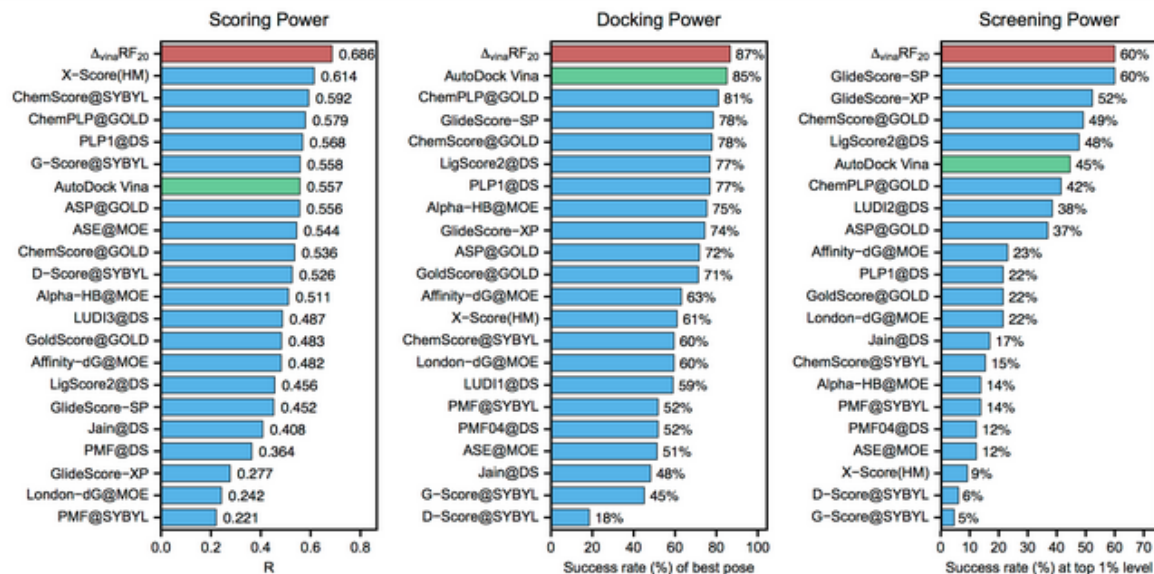
simultaneously, we have introduced a $\Delta_{\text{vina}}\text{RF}$ parameterization and feature selection framework based on random forest. Our developed scoring function $\Delta_{\text{vina}}\text{RF}_{20}$, which employs 20 descriptors in addition to the AutoDock Vina score, can achieve superior performance in all power tests of both CASF-2013 and CASF-2007 benchmarks compared to classical scoring functions. The $\Delta_{\text{vina}}\text{RF}_{20}$ scoring function and its code are freely available on the web at: <https://www.nyu.edu/projects/yzhang/DeltaVina>. © 2016 Wiley Periodicals, Inc.

DOI: 10.1002/jcc.24667

<http://www.nyu.edu/projects/yzhang/DeltaVina/>

$\Delta_{\text{vina}}\text{RF}_{20}$

A new scoring function employs twenty descriptors in addition to the AutoDock Vina score, and has been developed using the Delta Random Forest (ΔRF) algorithm. It has achieved superior performance in all power tests of both CASF-2013 and CASF-2007 benchmarks.



Reference:

Cheng Wang, and Yingkai Zhang, *J. Comput. Chem.*, 2016, DOI: 10.1002/jcc.24667

Improving Scoring-Docking-Screening Powers of Protein-Ligand Scoring Functions using Random Forest

```
## setup environment
$source ~centos/env.bashrc
## Δvina
$cp -p -r ~centos/programs/deltavina ./
$cd deltavina
$cd examples
$dvrf -help
## score for one complex
$ dvrf -r 1a42/1a42_protein_proc_se.pdb
-l 1a42/1a42_ligand_fix.mol2
## score for a list of complex
$dvrf -g pdblast.txt
$cat output.csv
```

$$pK_d \times -1.36 = \Delta G$$

```
## rescore with Δvina
$cd ~/FCDD_tutorials/docking
$mkdir 3_rescoring
$cd 3_rescoring
$mkdir deltaVina
$cp ../1_preparation/1cqp_1_ligand_wH.mol2 ./deltaVina
$cp ../1_preparation/1cqp_1_protein.pqr ./deltaVina

$cd deltaVina
$ls
$dvrf -r 1cqp_1_protein.pqr -l 1cqp_1_ligand_wH.mol2
$ls
$

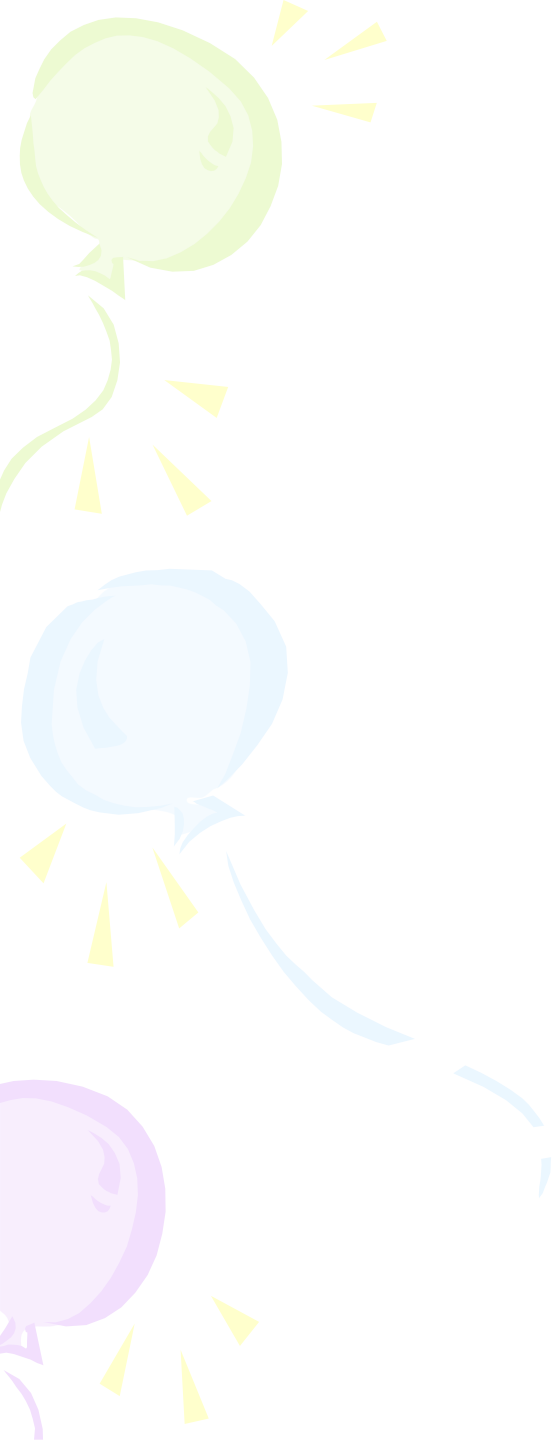
$python ~/centos/programs/deltavina/bin/dvrf20.py -r 1cqp_1_protein.pqr \
-l 1cqp_1_ligand_wH.mol2
```

$$pK_d \times -1.36 = \Delta G$$


$$5.47 \times -1.36 = -7.44 \text{ kcal/mol}$$

A decorative graphic on the left side of the slide features three balloons: a green one at the top, a light blue one in the middle, and a purple one at the bottom. Each balloon has a string and is surrounded by several small yellow triangles, suggesting movement or light.

http://www.nyu.edu/projects/yzhang/Practicals_YingkaiZhang.tar.gz



Protein–Ligand Scoring with Convolutional Neural Networks

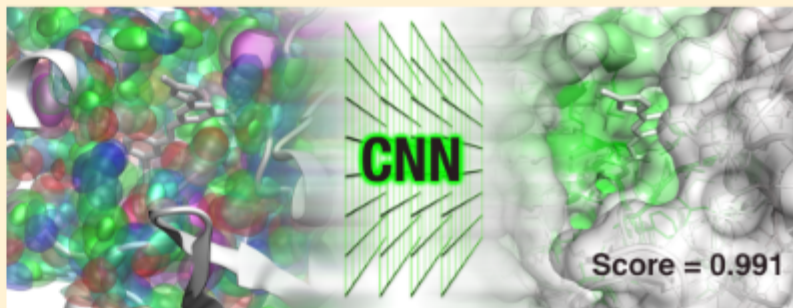
Matthew Ragoza,^{†,‡} Joshua Hochuli,^{‡,¶} Elisa Idrobo,[§] Jocelyn Sunseri,^{||} and David Ryan Koes^{*,||} 

[†]Department of Neuroscience, [‡]Department of Computer Science, [¶]Department of Biological Sciences, and ^{||}Department of Computational and Systems Biology, University of Pittsburgh, Pittsburgh, Pennsylvania 15260, United States

[§]Department of Computer Science, The College of New Jersey, Ewing, New Jersey 08628, United States

Supporting Information

ABSTRACT: Computational approaches to drug discovery can reduce the time and cost associated with experimental assays and enable the screening of novel chemotypes. Structure-based drug design methods rely on scoring functions to rank and predict binding affinities and poses. The ever-expanding amount of protein–ligand binding and structural data enables the use of deep machine learning techniques for protein–ligand scoring. We describe convolutional neural network (CNN) scoring functions that take as input a comprehensive three-dimensional (3D) representation of a protein–ligand interaction. A CNN scoring function automatically learns the key features of protein–ligand interactions that correlate with binding. We train and optimize our CNN scoring functions to discriminate between correct and incorrect binding poses and known binders and nonbinders. We find that our CNN scoring function outperforms the AutoDock Vina scoring function when ranking poses both for pose prediction and virtual screening.




```
## rescore with gnina cnn_scoring
$source ~centos/programs/gnina/useGnina.sh

$cd ~/FCDD_tutorials/docking
$cd 3_rescoring
$mkdir gnina
$cp ../1_preparation/*.pdbqt ./gnina
$cp ../2_docking/autodock_vina/config.txt ./gnina
$cd gnina

$gnina --config config.txt --score_only --cnn_scoring --log
gnina_CNNscoreonly.log
```

(smina) Affinity: -6.01013 (kcal/mol)

CNNscore: 0.8159278631

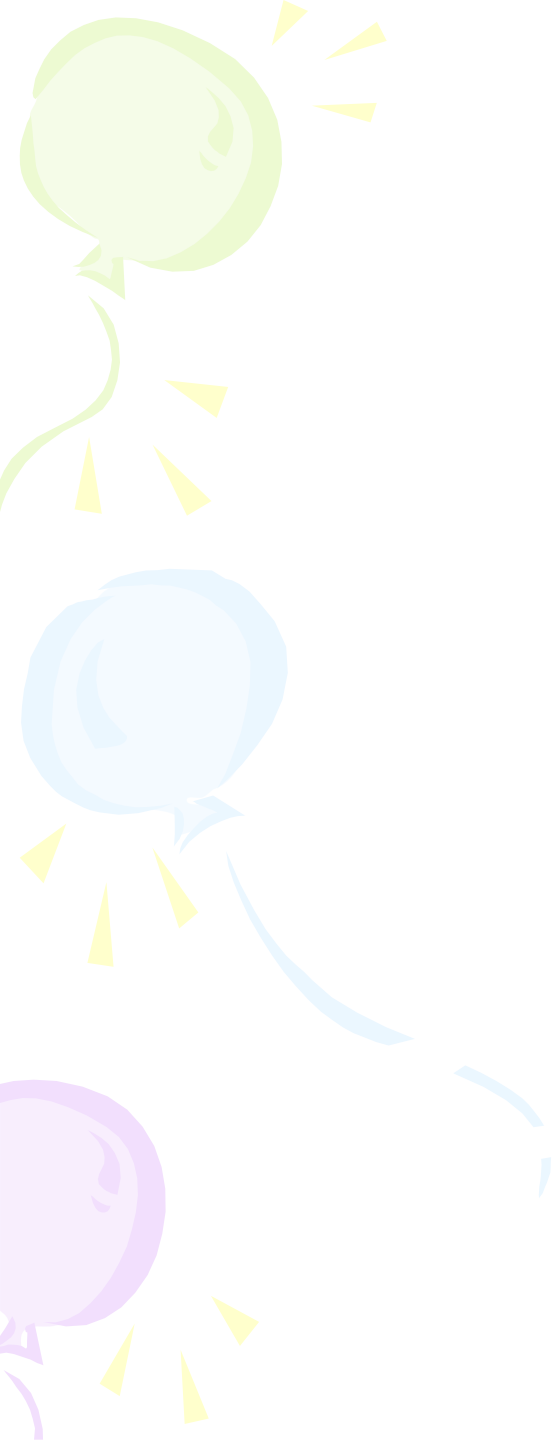
CNNaffinity: 6.3308925629 (-8.61 kcal/mol)

$$\text{pK}_d \times -1.36 = \Delta G$$



Exercise

- Rescore docked poses (autodock4 or autodock vina) with other scoring functions, such as Δ vina and gina
- Rescore virtual screening result with other scoring function



DeepChem

<https://deepchem.io/>



deepchem

About

Blog

Tutorials

Discuss

Docs

What is DeepChem?

DeepChem is a python library that provides a high quality open-source toolchain for deep-learning in drug discovery, materials science, quantum chemistry, and biology.

About Us

DeepChem is possible due to notable contributions from many people including Peter Eastman, Evan Feinberg, Joe Gomes, Karl Leswing, Vijay Pande, Aneesh Pappu, Bharath Ramsundar and Michael Wu (alphabetical ordering). DeepChem was originally created by [Bharath Ramsundar](#) with encouragement and guidance from Vijay Pande.

DeepChem started as a Pande group project at Stanford, and is now developed by many academic and industrial collaborators. DeepChem actively encourages new academic and industrial groups to contribute!

DeepChem is a Python library democratizing deep learning for science.



Tutorials

<https://deepchem.io/docs/notebooks/index.html>

The following tutorials show off various aspects or capabilities of DeepChem. They can be run interactively in Jupyter (IPython) notebook. Download the [notebook files](#) and open them in Jupyter:

```
$ jupyter notebook
```

Applications Tutorials

- Graph Convolutions For Tox21
- Multitask Networks On MUV
- Creating a high fidelity DeepChem dataset from experimental data
- Predicting Ki of Ligands to a Protein
- Basic Protein-Ligand Affinity Models
- Quantum Machinery with gdb1k
- Modeling Solubility

Machine Learning Methods Tutorials

- MNIST with DeepChem and TensorGraph
- Conditional Generative Adversarial Network
- Using DeepChem Splitters
- MNIST GAN
- Pong in DeepChem with A3C
- SeqToSeq Fingerprint
- TensorGraph Mechanics
- Using Tensorboard

<https://deepchem.io/docs/deepchem.html>

Subpackages

- deepchem.data
- deepchem.dock
- deepchem.feats
- deepchem.hyper
- deepchem.metalearning
- deepchem.metrics
- deepchem.models
- deepchem.molnet
- deepchem.nn
- deepchem.rl
- deepchem.splits
- deepchem.trans
- deepchem.utils

```
##  
$source activate deepchem  
  
$cd ~/FCDD_tutorials  
$mkdir deepchem  
$cd deepchem  
$cp -p -r ~/programs/deepchem/examples/pdbbind ./  
$cd pdbbind  
  
## train Sklearn RF models on PDBbind dataset  
$python pdbbind_rf.py  
  
## train Tensorflow models on PDBbind dataset  
$python pdbbind_tf.py  
  
$source deactivate
```

```
"""
```

```
Script that trains Sklearn RF models on PDBbind dataset.
```

```
"""
```

```
from __future__ import print_function
```

```
from __future__ import division
```

```
from __future__ import unicode_literals
```

```
__author__ = "Bharath Ramsundar"
```

```
__copyright__ = "Copyright 2016, Stanford University"
```

```
__license__ = "GPL"
```

```
import os
```

```
import deepchem as dc
```

```
import numpy as np
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
from deepchem.molnet import load_pdbbind_grid
```

```
# For stable runs
```

```
np.random.seed(123)
```

```
split = "random"
```

```
subset = "full"
```

```
pdbbind_tasks, pdbbind_datasets, transformers = load_pdbbind_grid(  
    split=split, subset=subset)
```

```
train_dataset, valid_dataset, test_dataset = pdbbind_datasets
```

```
metric = dc.metrics.Metric(dc.metrics.pearson_r2_score)
```



```
current_dir = os.path.dirname(os.path.realpath(__file__))
model_dir = os.path.join(current_dir, "%s_%s_RF" % (split, subset))

sklearn_model = RandomForestRegressor(n_estimators=500)
model = dc.models.SklearnModel(sklearn_model, model_dir=model_dir)

# Fit trained model
print("Fitting model on train dataset")
model.fit(train_dataset)
model.save()

print("Evaluating model")
train_scores = model.evaluate(train_dataset, [metric], transformers)
valid_scores = model.evaluate(valid_dataset, [metric], transformers)

print("Train scores")
print(train_scores)

print("Validation scores")
print(valid_scores)
```

pdbbind_rf.py

```
(deepchem) [centos@fcdd01 pdbbind]$ python pdbbind_rf.py
/home/centos/.conda/envs/deepchem/lib/python3.5/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float) type`.
  from ._conv import register_converters as _register_converters
Loading dataset from disk.
TIMING: dataset construction took 1.966 s
Loading dataset from disk.
TIMING: dataset construction took 0.590 s
Loading dataset from disk.
TIMING: dataset construction took 0.587 s
Loading dataset from disk.
Fitting model on train dataset
Evaluating model
computed_metrics: [0.9655430349178509]
computed_metrics: [0.4965790894642907]
Train scores
{'pearson_r2_score': 0.9655430349178509}
Validation scores
{'pearson_r2_score': 0.4965790894642907}
(deepchem) [centos@fcdd01 pdbbind]$ █
```

```
(deepchem) [centos@fcdd01 pddbnd]$ python pddbnd_tf.py
/home/centos/.conda/envs/deepchem/lib/python3.5/site-packages/h5py/_init_.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Loading dataset from disk.
TIMING: dataset construction took 1.949 s
Loading dataset from disk.
TIMING: dataset construction took 0.597 s
Loading dataset from disk.
TIMING: dataset construction took 0.596 s
Loading dataset from disk.
2018-03-14 13:34:33.799368: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2
Ending global_step 999: Average loss 8391.57
Ending global_step 1999: Average loss 973.418
Ending global_step 2999: Average loss 394.828
Ending global_step 3999: Average loss 222.237
Ending global_step 4999: Average loss 154.611
Ending global_step 5999: Average loss 110.241
Ending global_step 6999: Average loss 94.5004
Ending global_step 7999: Average loss 70.9858
Ending global_step 8999: Average loss 57.3503
Ending global_step 9999: Average loss 48.7454
Ending global_step 10999: Average loss 38.9426
Ending global_step 11999: Average loss 35.509
Ending global_step 12999: Average loss 28.3201
Ending global_step 13999: Average loss 26.5567
Ending global_step 14200: Average loss 22.4757
TIMING: model fitting took 532.188 s
Evaluating model
computed_metrics: [0.9643913125971887]
computed_metrics: [0.4570597578157433]
Train scores
{'pearson_r2_score': 0.9643913125971887}
Validation scores
{'pearson_r2_score': 0.4570597578157433}
(deepchem) [centos@fcdd01 pddbnd]$
```

