POEM: Parameter Optimization Using Ensemble Methods: Application to Target Specific Scoring Functions

Iris Antes,* Christian Merkwirth, and Thomas Lengauer

Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, D-66123 Saarbrücken, Germany

Received January 31, 2005

In computational biology processes such as docking, binding, and folding are often described by simplified, empirical models. These models are fitted to physical properties of the process by adjustable parameters. An appropriate choice of these parameters is crucial for the quality of the models. Locating the best choices for the parameters is often is a difficult task, depending on the complexity of the model. We describe a new method and program, POEM (Parameter Optimization using Ensemble Methods), for this task. In POEM we combine the DOE (Design Of Experiment) procedure with ensembles of different regression methods. We apply the method to the optimization of target specific scoring functions in molecular docking. The method consists of an iterative procedure that uses alternate evaluation and prediction steps. During each cycle of optimization we fit an approximate function to a defined loss function landscape and improve the quality of this fit from cycle to cycle by constantly augmenting our data set. As test applications we fitted the FlexX and Screenscore scoring functions to the kinase and ATPase protein classes. The results are promising: Starting from random parameters we are able to locate parameter sets which show superior performance compared to the original values. The POEM approach converges quickly and the approximated loss function landscapes are smooth, thus making the approach a suitable method for optimizations on rugged landscapes.

1. INTRODUCTION

Most structural models in computational chemistry, biology, and material sciences use approximate scoring functions, which contain a certain number of adjustable parameters. An appropriate choice of these parameters is a prerequisite for the optimal performance of the models. However, up to now there is no algorithm that ensures the determination of the optimal parameters for models that have degrees of freedom with continuous values, although extensive research has been performed on this topic in the computational biosciences. Various known regression/optimization methods have been applied for this purpose, but no optimal solution has been found.[9] Here we present a new method, POEM, which combines the design of experiments (DOE) approach[1] with ensembles of various regression methods.[2,10−12] We successfully apply and test the method by optimizing the FlexX[3−5] and Screenscore[6] molecular docking scoring functions for the ATPase and kinase protein classes.

In general, scoring functions are optimized with respect to a loss function that measures the quality of the fit of some measurable value achieved by the scoring function with respect to experimental data. The most straightforward method to minimize the loss function with respect to the parameters of the scoring function is to perform a throughout sampling of the whole parameter space. However, this is computationally too costly for most applications. Instead, often local methods such as steepest decent, conjugate gradient, etc. are used to minimize a defined loss function. The drawback of standard minimization methods is that they are very easily trapped in local minima and thus can only be used if the approximate location of the global minimum of the scoring function is known. In all other cases more general methods such as genetic algorithms[13] should be used, and special global optimization techniques[14] have been developed.

Here we describe an alternative approach, which is based on a combination of the design of experiments (DOE) methodology with ensemble methods. The DOE approach is frequently used as an optimization approach for neural networks and has been applied predominantly in the areas of operation research and experimental design.[15] We extend the method to the incorporation of ensembles of various regression models and explore its applicability to parameter optimization in drug screening. In the DOE approach the loss function landscape is explored in three steps. First, loss function values are calculated for a set S of randomly chosen points in parameter space. Second, the surface of the loss function landscape is approximated based on the randomly chosen points in S, using statistical learning techniques. Third, the global minimum of the approximated landscape is located. An experiment is performed for the parameter values of this point, i.e., the exact loss function value is calculated. This value is added to the set S. Steps 2 and 3 are repeated on the augmented data set S until the loss function minima drop below a preset threshold (see Figure 1, outer loop).

Various statistical learning techniques[9] can be used for the approximation of the loss function surface. The most commonly used method is the quadratic polynomial function model. However, due to its restricted flexibility, often this model is not suitable for computer experiments, which tend to result in rugged and complex loss function surfaces.[16] For

* Corresponding author e-mail: antes@mpi-sb.mpg.de.

10.1021/ci050036g CCC: $30.25
Published on Web 09/07/2005

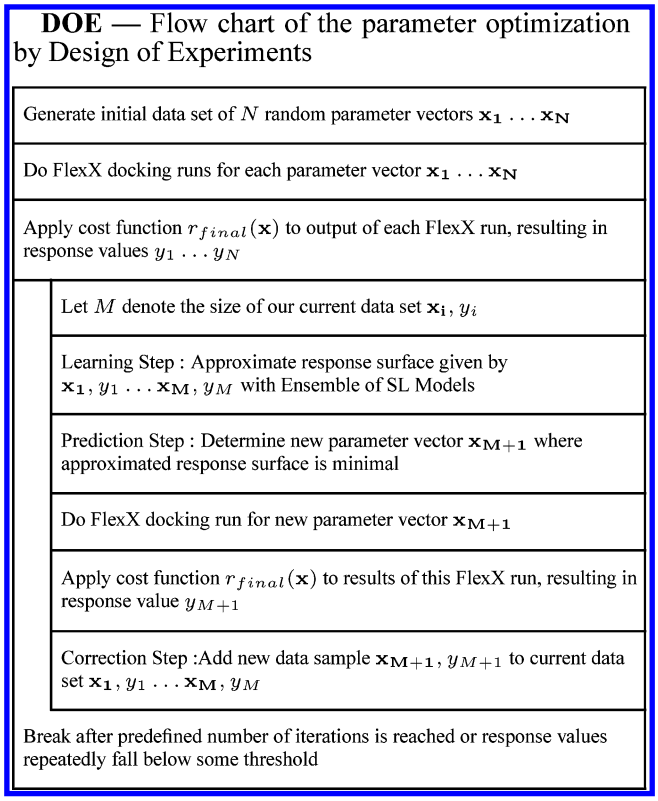

Figure 1. Flowchart of the POEM (Parameter Optimization using Ensemble Methods) algorithm and program.

this purpose more flexible methods such as k-nearest-neighbor models and neural networks[9] are preferable. Neural network models not only have strong generalization capabilities but also exhibit high computational variance. K-nearest-neighbor models exhibit lower computational variance but perform worse than neural networks if generalizing beyond the boundaries of the training data set. To circumvent these disadvantages, so-called ensemble techniques can be used. Ensemble building is a common method of improving the performance of the resulting regression since, generally, an ensemble of individual predictors performs better than any incorporated single predictor.[11] Almost all ensemble methods described so far use models of a single class, e.g. neural networks[2,10−12] or regression trees.[17] We extend the method to building ensembles of different model classes. The theoretical background of our approach is provided by the bias/variance decomposition of the ensemble model.[11]

As test application we chose the improvement of the coefficients of two scoring functions used in molecular docking. Due to the tight time constraints placed on molecular docking methods by virtual screening applications, it is necessary to use scoring functions which can be evaluated fast. Thus the functional form of the scoring functions has to be simple, which makes the fitting of their coefficients to physical properties of the system a challenging task.

There are three kinds of scoring functions, which are commonly used: empirical,[5] knowledge based,[18,19] and force field[20] based scoring functions. Lately also mixtures of these functions have been developed.[21] In general, empirical scoring functions are fitted to the free energies of ligand binding for known structures of ligand/protein complexes.[5] Knowledge based potentials use statistical functions, which are based on the analysis of the ligand/protein interaction geometries in the PDB database.[18,19] Force fields are commonly fitted to accurate quantum chemical calculations.[20] Here we will use only empirical scoring functions. Both scoring functions that we will use, the FlexX[5] and the Screenscore[6] scoring functions, are additive functions, which are based on the physical properties of ligand/protein interactions (see "Methods").

The accuracy of empirical scoring functions is limited by two factors: First, by their simplified functional form and second, by the necessity to fit standard docking parameters to a large, diverse set of proteins to ensure their general applicability. Thus scoring can be improved by either adding new terms, e.g. solvation terms, to the function in order to improve the physical description of the system or by tailoring the function parameters to the protein class under investigation. Tailoring scoring functions to certain target classes has been a research topic for quite some time. Most relevant approaches include additional, mostly structural information available about the system under consideration into the scoring and/or docking procedure. This information can be used to tailor the docking process itself or to pre- and postfilter the docking results.[7] Here we explore another possibility, namely improving the parameters by fitting them to protein class specific data. We do not include additional terms or constraints in our docking and scoring protocol.

Docking calculations can be divided into two phases: First, generating a number of placements of the ligand and second, the scoring and ranking of these placements. Due to the different requirements on the scoring function for either phases, often different scoring functions are used for these purposes. In this study we aim at the optimization of existing docking scoring functions for ligand placement. This choice was based on the observation that proper ranking of the ligands tends to be hampered by inaccurate ligand placement, thus making reranking efforts useless. Thus, we include the placement process into the parameter optimization procedure and use an iterative algorithm that performs statistical learning and full docking (ligand placement and ranking) runs alternately. In addition, we base our loss function on the atom-positional rmsd of the ligands position in the binding pocket between the docked structure and the experimental structure. We abstain from the direct inclusion of experimental binding data into our fitting procedure, because we have no access to suitable data for this purpose. We believe that it is hard, in general, to come by such data. However, it is important to explicitly ensure that ranking does not deteriorate during optimization, because, typically, for reranking only the N best solutions are considered. Thus after test calculations using the docking poses with the lowest RMSD values for the optimization, we chose to only consider the RMSD of the docking poses of the five best scoring docking solutions during the optimization. This ensures that the highest ranking solutions coincide with the docking poses with low RMSD.

As test cases we chose the protein classes of kinases and ATPases using the FlexX and Screenscore scoring functions. The kinase protein class was chosen because of its importance in drug design and the availability of a large number of structures in the PDB database. Numerous docking studies with kinases as target molecules can be found in the literature.[22] Kinases are no easy targets for virtual screening

due to the flexibility of their binding pockets.[23] However, we exclude the problem of protein flexibility in this study by restricting our analysis to redocking experiments. The ATPase protein class was chosen because of its increasing importance in drug design[24] and because docking with FlexX was quite problematic on some PDB complexes of this class. Problems occurred especially with the placement of the imidazole region of the ATP/ADP substrates. As scoring function we chose the FlexX scoring function, because it is the standard scoring scheme for ligand placement in the FlexX molecular docking program and often provides better ligand placement than other scoring schemes, which are optimized for reranking. Furthermore, we included the Screenscore function[6] into our testing, because it is known for its better performance for the kinase protein class.

2. METHODS

In our algorithm we combined the DOE method with ensembles of neural networks and k-nearest neighbor methods for the statistical learning part and used the FlexX docking algorithm for the docking part. Based on the DOE idea, the POEM program performs alternating evaluation and learning steps, thereby adding more and more information to the data point set on which the learning is based. This results in an improvement of the quality of our parameter set in an iterative fashion. In the following paragraphs we first discuss the overall data flow and then provide more details on the methods and data sets used.

2.1. Design of Experiments. The general outline of the parameter optimization approach by POEM is shown in Figure 1. The actual DOE procedure corresponds to the outer loop in Figure 1. For initialization, we first define a loss function and generate a set S of N (in our case between 256 and 366 samples) random parameter settings uniformly distributed within the set boundaries. Second, we calculate the loss function values for the random parameter sets and interpolate several loss function surfaces (section 2.2, and "learning step" in Figure 1). Third, in the prediction step ("prediction step", Figure 1) we find the global minimum of the ensemble approximation of the loss function surfaces. For this purpose we employ a simple genetic algorithm that performs a global search within the feasible region. Last, a "real" experiment (in our case a full docking run) is performed for the parameter set of the located minimum, and the results are added to the point set S for the next iteration ("correction step", Figure 1). In this way information about the loss function surface is fed back into the learning step, thus improving the approximation quality of the models constructed in the next round. [This mutual improvement can be seen as a combination of DOE with Active Learning.[11]]

2.2. Ensemble Methods. We use ensemble methods for the approximation of the loss function/parameter landscape (see "learning step", Figure 1), because ensemble methods show an improved ability of generalization compared to any single models and perform more consistently than neural networks,[11] leading to more robust solutions. For our specific application we chose a heterogeneous ensemble of neural networks and k-nearest neighbor models.

The model types employed and the respective training procedures are described in Appendix 1. We predominantly used neural networks, because our response function is quite complex and neural networks are a highly flexible model class. At every nth step we additionally employ a k-nearest neighbor model in order to increase the diversity of our models. We employ three different types of ensembles (which we call "modi"): the plain average, the plain average plus the standard deviation, and the plain average minus the standard deviation. The latter two modi sample the parameter space more broadly. Finally, we increase the diversity within the ensemble by training the single models, which enter the ensemble, on different subsets of the training set. At each cycle the minima of the loss surfaces are evaluated for all three modi. Afterward docking experiments are performed for the three corresponding parameter sets, and the results are added to the training data set.

2.3. Choice of the Loss Function. The exact formulation of the loss function $r(\mathbf{x})$ has a crucial influence on the results of the whole DOE procedure. The loss function must faithfully reflect the properties of the desired solution.

Because we are interested in optimizing ligand placement, we chose to base our loss function on the positional root-mean-square-deviation between the calculated and measured ligand position. In addition, we concentrate on the five best docking solutions for the C protein−ligand complexes of the training set.

Because redocking solutions can be ordered either according to their energetic score or according to the RMSD of the docking pose from the experimental structure, we define two loss functions: In the first, we use the five docking results with the lowest RMSD (which not always coincide with the ones with the lowest score) for the loss function calculation (see below). We will refer to this function as "loss function 1". In the second, we use the RMSD of the five energetically highest ranking solutions for this purpose, thereby implicitly including aspects of the thermodynamics of docking into the analysis (in the following we will refer to this loss function as "loss function 2").

Let $R_{i,j}$ denote the RMSD value of the rank-j docking solution for the complex i in the training set. Let also $k_i(l)$ be the index permutation that sorts the five best docking solutions for the ith complex of the training point set by ascending RMSD value:

$$R_{i,k_i(1)} \le R_{i,k_i(2)} \le \ldots \le R_{i,k_i(5)}\ i \in 1,\ldots, C \qquad (1)$$

The loss function for one parameter setting $\mathbf{x}$ is defined as the average over all fourth-lowest RMSD values $R_{i,k_i(4)}$:

$$r_{\text{prelim}}(\mathbf{x}) = \frac{1}{C}\sum_{i=1}^{C} R_{i,k_i(4)} \qquad (2)$$

Taking the fourth-lowest value for the averaging prevents that the response values are strongly influenced by large outliers in the RMSD values. In addition, this procedure implies that there are three docking solutions, which have equal or better RMSD values among the five highest ranked solutions.

To further enhance robustness we want the loss function to ignore improvements for low RMSD values (e.g. changes from 0.8 to 0.3 Å) and focus on improvements in the higher RMSD regimes (e.g. at ranges from 2.0 to 8.0 Å) according to our goal of improving the ligand placement for instances,

which are docked inaccurately with the original scoring function. Therefore we introduce a piecewise linear insensitivity function $\Gamma(z)$:

$$\Gamma(z) = N \text{ for } z \leq 1.0$$

$$\Gamma(z) = z \text{ else}$$

This leads to

$$r_{\text{final}}(\mathbf{x}) = \frac{1}{C}\sum_{i=1}^{C}\Gamma(R_{i,k_i(4)}) \tag{3}$$

We used $N = 1.0$ and also tested the values $N = 0.0$, 0.5.

2.4. Docking Settings and Choice of Parameters. The docking calculations were performed with the FlexX program[3,4] using the standard values for all program parameters, except for the parameters to be optimized. In FlexX the ligand is treated flexible and the protein rigid during the docking calculations. We performed full docking calculations including ligand placement and scoring at each optimization cycle for all new parameter sets and all structures in the training set. For the FlexX scoring function:

$$\Delta G_{\text{bind}} = \Delta G_{\text{const}} + \Delta G_{\text{rot}}F_{\text{rot}} + \Delta G_{\text{match}}F_{\text{match}} + \Delta G_{\text{lipo}}F_{\text{lipo}} + \Delta G_{\text{ambig}}F_{\text{ambig}} + \Delta G_{\text{clash}}F_{\text{clash}}$$

we optimized all six coefficients (ΔGs) plus one additional geometric parameter: the radius of the phenyl-center interaction (seven coefficients in total). The geometric parameter was included because initial docking experiments showed that its value was the only parameter among several parameters defining the geometry of hydrophobic interactions, which exhibited some importance on the docking results.

In the case of the Screenscore scoring function we added the $\Delta G_{\text{clash}}F_{\text{clash}}$ term to the original function, because of the functions use for ligand placement:

$$\Delta G_{\text{bind}} = \Delta G_{\text{rot}}F_{\text{rot}} + \Delta G_{\text{match}}F_{\text{match}} + \Delta G_{\text{lipo}}F_{\text{lipo}} + \Delta G_{\text{ambig}}F_{\text{ambig}} + \Delta G_{\text{clash}}F_{\text{clash}} + \Delta G_{\text{plp\_steric}}F_{\text{plp\_steric}} + \Delta G_{\text{plp\_hbond}}F_{\text{plp\_hbond}} + \Delta G_{\text{plp\_rep}}F_{\text{plp\_rep}}$$

and optimized all coefficients (ΔGs) of the above function, except the ΔG_{rot} parameter (seven coefficients in total). This time we did not include the radius of the phenyl-center interactions into the optimized parameter set, because in contrary to our initial docking results it did not have any significant impact on the parameter optimization of the FlexX scoring function (Figure 5a,c). The upper and lower limits of the sampled parameter space are given in Table 4.

2.5. Training and Validation Sets. Twenty-four ATPase and 36 kinase protein structures were chosen from the PDB database depending on their resolution and the diversity within the sets. For the ATPase set we included members of all three ATPase families. The structures were divided into a training and validation set such that the average docking performance (RMSD) of the individual structures was the same in both sets and the diversity of the structures (number of different types of kinases/ATPases) within each set was maximized. Thus we included members of all three ATPase families (P-loop, actin like, and GHL ATPases) into the training and validation set. The training set included the structures with the PDB code: 1f9t, 1in4, 1ihu, 1byq, 1i5s, 1ii6, 1bg2, 1e3m, 1mmg, 1f2u, 1g6h, 1d2n, 1e1q, 2btf, 1b63, and 1qhh. The validation set contained the following structures: 1fnn, 1h8e, 1g3q, 1hjo, 1ea6, 1e0j, 1e4g, and 1ei1. For the kinase protein class we chose predominantly structures of CDK2 kinases, because there is a large number of high-resolution CDK2 structures in the PDB database and this protein is an important drug target. However, we decided to include structures of other kinases into the training and validation set to enhance the diversity. The resolution of all structures in our training set was better than 2 Å. The training set consisted of the following PDB structures: 1gz8, 1h07, 1h08, 1h0v, 1jvp, 1lp4, 1pxi, 1qpc, 1uu7, 1aq1, 1e1v, 1e1x, 1gii, 1jsv, and 1ke7, and the validation set included 21 structures: 1fvv, 1b38, 1di8, 1dm2, 1fvt, 1g5s, 1gih, 1gij, 1b39, 1ckp, 1e9h, 1gy3, 1h1p, 1h1r, 1h1s, 1hck, 1ke5, 1ke6, 1ke8, 1ke9, 1qmz. In addition to the above training set, we tested 2 more, for which we used 18 and 21 structures, but no significant improvement was found for the larger sets. Thus, due to CPU considerations, we chose the training set containing 15 structures and the validation set containing 21 structures.

2.6. Analysis of the Results. First, we tested the ability of several regression models to approximate the true loss function values for the random parameter sets S chosen at the beginning of our optimization runs. Second, we monitored the progression of the loss function values during the optimization. Third, we calculated the atom-positional rmsd of the calculated and experimental positions of the ligands in the binding pocket for the training and validation sets of all six final, optimized scoring function parameter sets. Fourth, we evaluated the performance of the optimized parameter sets on the other protein class and on the standard FlexX200 test set. Last, we performed a sensitivity analysis, analyzing the importance of each parameter for the loss function improvement. This analysis was based on the Out-of-Train procedure (see the Appendix).

3. RESULTS

3.1. Comparison with Other Regression Models. In Figure 2 and Table 1 we compare the ability of different regression models to approximate the loss function values for our four data sets (kinase/FlexX, kinase/Screenscore, ATPase/FlexX, and ATPase/Screenscore). For this analysis we took the randomly generated set S at the beginning of the optimizations and used different statistical learning methods for the approximation of the loss function landscape. The model types used for this comparison are linear regression (ridge regression), quadratic polynomial regression, k-nearest-neighbors regression, single neural networks, and ensembles of neural networks. Table 1 gives the 10-fold cross-validated mean square errors normalized by the variance of the original outputs. We infer from Table 1 that ensembles of neural networks outperform the other models types significantly. Linear models perform worst and are definitely not able to fit any bowl shaped loss function landscape. Quadratic polynomial models cannot compete with the neural network ensembles, either. Thus ensembles of neural networks are the best choice for our optimization

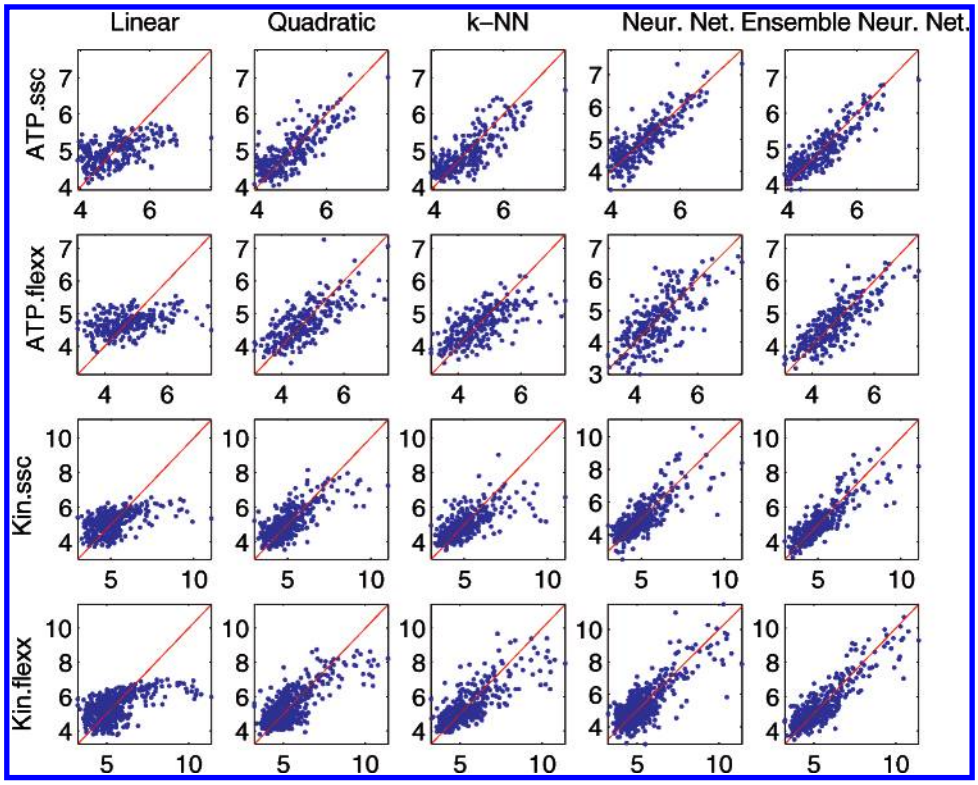

Figure 2. Correlation plots for five model types and four data sets. The data sets consist of randomly generated parameter settings within the boundary region and the respective values of the loss function 2 for each parameter setting. The plots show the 10-fold cross-validated outputs of the model versus the actual value of the loss function. Model types used for this comparison are linear regression (ridge regression), quadratic polynomial regression, k-nearest-neighbors regression, single neural networks, and ensembles of neural networks. Results of this comparison can also be found in Table 1.

Table 1. Relative Mean Square Errors for Five Model Types and Four Data Sets[a]

data set	no. obs.	no. compl.	linear	quadratic	k-nn	neur. net.	neur. net. ensemble
ATP.ssc	256	16	0.75	0.36	0.35	0.31	0.23
ATP.flexx	256	16	0.85	0.46	0.54	0.56	0.33
Kin.ssc	366	15	0.75	0.46	0.57	0.41	0.29
Kin.flexx	366	15	0.71	0.41	0.39	0.39	0.23
average			0.77	0.42	0.46	0.42	0.27

[a] The data sets consist of randomly generated parameter settings within the feasible region and the respective values of the loss function for each parameter setting. Error values where calculated by 10-fold cross-validation. Model types used for this comparison are linear regression (ridge regression), quadratic polynomial regression, k-nearest-neighbors regression, single neural networks, and ensembles of neural networks. Values in the bottom line are averages over the four data sets. The table also details the number of data points (no. obs) and the number of complexes (no. compl) which were used to compute RMSD values that enter the loss function. Results of this comparison are depicted in Figure 2.

task. An additional advantage of neural network ensembles is that they exhibit a drastically reduced computational variance compared to single neural networks, which results in a lower variance of the suggested minima of the loss function. The consistently low errors for the ensembles of neural networks indicate that we do not run into the problem of overfitting, though we do not pay any special attention to model complexity.

3.2. Optimizing the Scoring Functions for the Kinase Protein Class. In the first step we produced 366 random parameter sets within the given limitations (see "Methods") of the parameter space. The docking data of these parameter sets formed the basis for the first optimization cycle. During all following cycles three data points were added to the data set S, which corresponded to the minimum parameter sets of the three modi used during the ensemble averaging. Overall we performed three different optimization runs for the kinase protein class: one for the FlexX scoring function, using loss function 2, one using the Screenscore function with the same loss function, and one combining the Screenscore function with loss function 1.

In Figure 3a the loss function values for the random parameter sets and the parameter sets resulting from the learning procedure are given with respect to the progression of the optimization run for the FlexX scoring function using loss function 2. The first 366 values correspond to the random distribution of parameters at the beginning of our run. Several high value peaks can be observed during the optimizations. These peaks correspond to the values of the minimum parameter sets of the ensemble which uses the plain average plus the standard deviation for averaging. Thus they are our "exploration shots", introduced for comprehensively sampling all of the parameter space. Thus high values indicate that we indeed are able to sample unfavorable regions of the parameter space. In Figure 3b cuts through the final landscape of the approximated loss function are shown for the most important of the optimized parameters (see also Figure 5a and Table 4). The plots verify that, indeed, we have found stable minima within our boundary limits. Because of the optimization approaches, the ensemble averaging, and the different modi used in the POEM

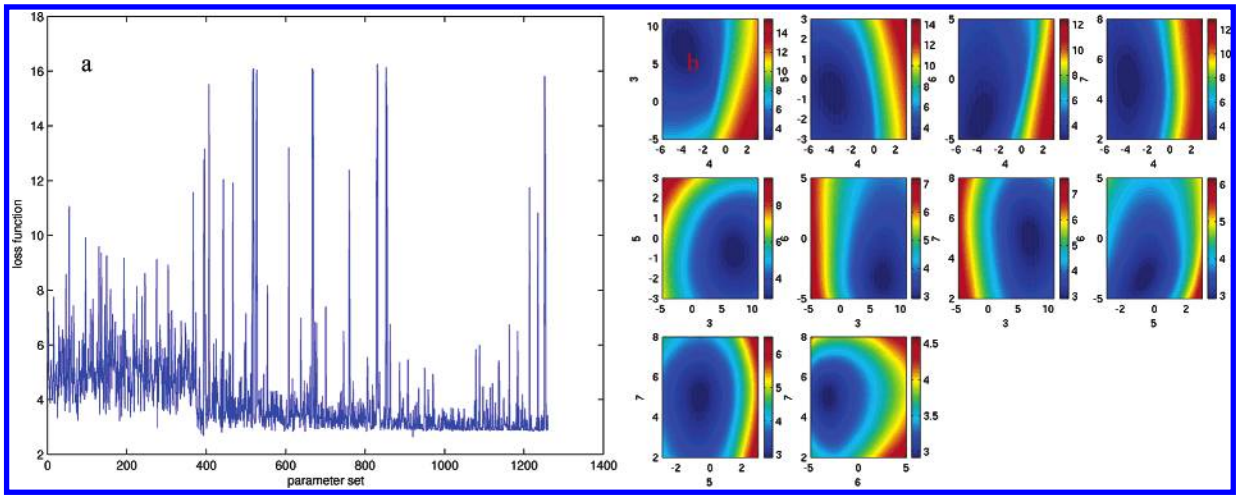

Figure 3. Part (a) shows the progression of the loss function during optimization for the kinase/FlexX function optimization using loss function 2. In (b) 2D cuts through the final approximation of the loss function landscape are shown. The labels and values of the axes of the landscape plots correspond to the scoring parameters plotted in Figure 5.

Table 2. Average Atom-Positional r.m.s.d. (in Å) of the Ligands Position in the Binding Pocket between the Best Docking Pose within the Five Lowest Ranked Solutions and the Experimental Structure for Loss Function 1 (LF1) and Loss Function 2 (LF2)[a]

	training set		validation set	
data set	original params	optimized params	original params	optimized params
kinase/FlexX/LF2	2.76	1.49	1.55	1.32
kinase/Screenscore/LF2	1.29	1.36	1.45	1.25
kinase/Screenscore/LF1	1.19	1.05	1.25	1.30
ATPase/FlexX/LF2	3.10	1.82	4.42	2.16
ATPase/Screenscore/LF2	2.93	1.64	4.05	2.32
ATPase/FlexX/LF1	1.72	1.00	1.81	1.76

[a] Values are averaged over all structures in our training and validation sets, respectively.

Table 3. Average Ranks of the Energetic Score of the Docking Pose with the Lowest RMSD for Loss Function 1 (LF1) and Loss Function 2 (LF2)[a]

	training set		validation set	
data set	original params	optimized params	original params	optimized params
kinase/FlexX/LF2	62	56	21	19
kinase/Screenscore/LF2	68	60	9	10
kinase/Screenscore/LF1	68	64	9	13
ATPase/FlexX/LF2	28	24	37	11
ATPase/Screenscore/LF2	35	19	66	52
ATPase/FlexX/LF1	28	35	37	51

[a] Values are averaged over all structures in our training and validation sets, respectively.

program, our approximated loss function landscapes are smooth, allowing for efficient sampling of these landscapes, which are quite rugged in reality.

To evaluate the quality of the obtained minima we analyzed the parameter set with the lowest loss function value after each optimization run and compared the results with the values obtained using the original scoring function parameters. For this purpose we first calculated the atom positional rmsd of the ligand position in the binding pocket between the best docking pose within the five best ranking solutions of our docking run and the experimental structure. In Table 2 the averages over these RMSD values for the training and validation sets are given for all three optimizations. For the FlexX scoring function a strong decrease in the RMSD can be observed for the training set and a slight decrease for the validation set. Regarding the Screenscore results, the improvements after optimization are much smaller, and no significant differences can be observed between the results obtained with the two different loss functions. The optimization using loss function 1 provides slightly lower values for the RMSD as can be expected from its definition. Because we introduced the two loss functions in order to identify the one with the better ranking characteristics, we also compared the final rank of the pose with the overall lowest RMSD for all three optimizations. The results are given in Table 3. An improvement in the ranking can be observed for both loss functions for the training set, which is smaller in the case of loss function 1, however. For the Screenscore validation sets the quality of ranking decreases after optimization, but the overall changes are small, which is in conformance with the small RMSD improvements. In the case of the FlexX scoring function using loss function 2, the ranking improves considerably for both sets. From these results we draw the following conclusion: If the original scoring function does not perform well, large improvements in the RMSD, as well as in the ranking, can be observed after optimization. However, if the original scoring results are already of high quality, the improvements of the respective average values after optimization are small. Because no definite conclusion could be drawn from the results above about which loss function to use, we also tested both loss functions for the ATPase protein class using the FlexX scoring function (see section 3.3 and Table 3). These tests led to the decision to finally use the loss function 2 in the further studies.

In Figure 4a,b the RMSD values obtained with the original and optimized parameter sets are plotted for all structures of the two training and validation sets for both scoring functions using loss function 2. The following trend can be observed: Using the optimized parameter sets we are able to dock all instances within a RMSD below 3.0 Å, most of them below 2.0 Å. We observe a drastic improvement for instances, in which FlexX showed a low performance initially (Figure 4a, standard score: 6.0−7.0 Å, optimized score 0.3−2.0 Å), but the quality of the decent docking instances (0.0−

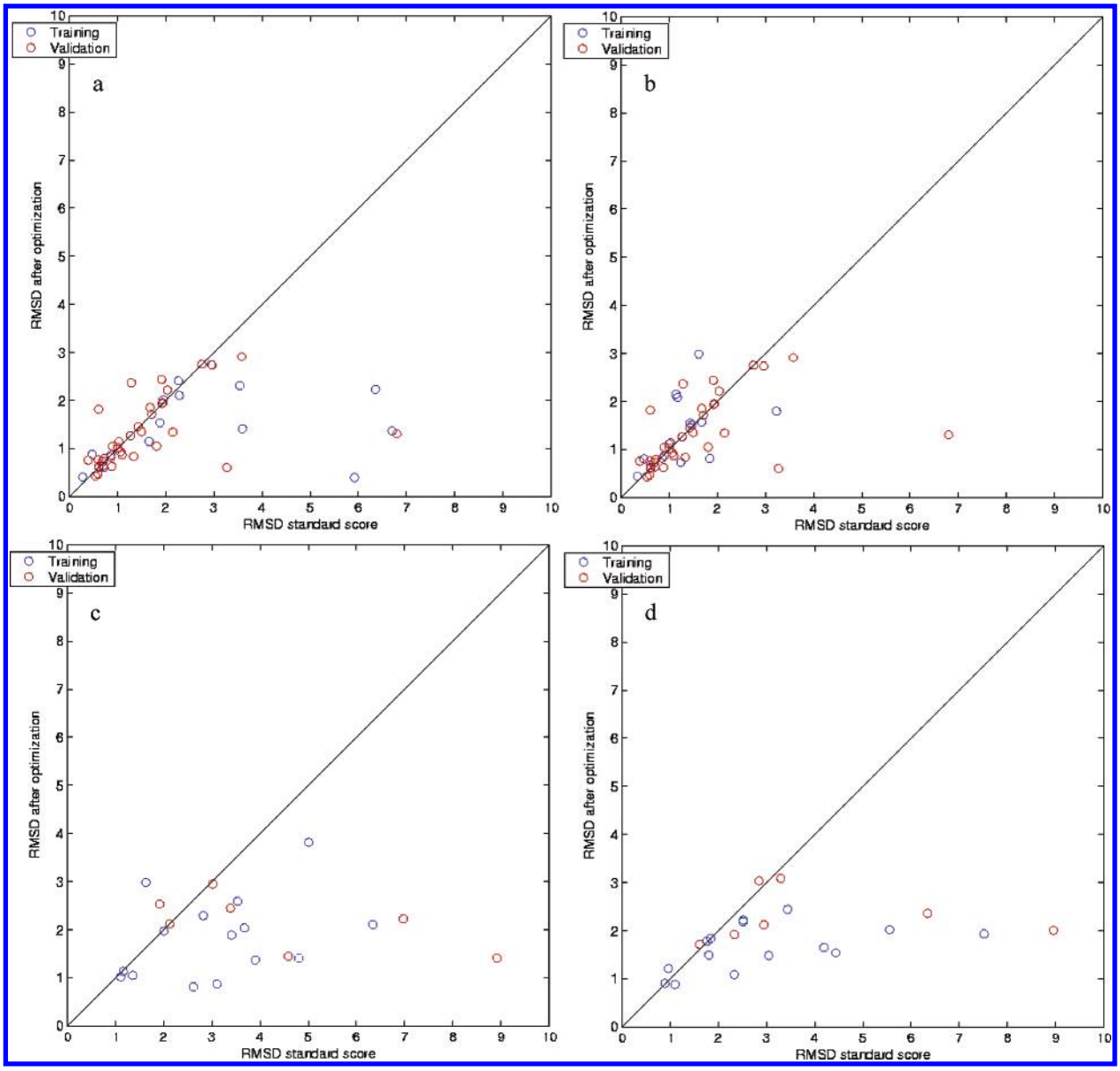

Figure 4. Atom-positional rmsd of the ligands position in the binding pocket between the best docking pose within the five energetically lowest ranked solutions (loss function 2) and the experimental structure for the original (*x*-axis) and the optimized scoring function parameter sets (*y*-axis): (a) kinase/FlexX function, (b) kinase/Screenscore function, (c) ATPase/FlexX function, and (d) ATPase/Screenscore function.

3.0 Å) does not improve considerably. Thus, for the single docking instances the results mirror those observed for the average values in Table 2: Bad docking instances are improved considerably, whereas good ones stay nearly the same. First we suspected that the reason for this phenomenon lies in the use of the insensitivity parameter $N = 1.0$ Å and thus we performed two optimizations with the value of N set to 0.0 Å and 0.5 Å, but no further improvement could be found even without the use of this parameter. Comparison of the average RMSD values (Table 2) with values obtained while testing the general performance of the scoring functions (see section 3.6) led us to the conclusion that the reason lies in us reaching the limit of accuracy for the applied scoring functions.

We further analyzed the performance of the applied ensemble approach by checking the correlation of the actual loss function values from the docking calculations with the approximated values by the learned loss function landscape. In addition, we performed sensitivity analyses for the optimized parameters. The correlation and sensitivity plots are given in Figure 5. The left-hand plots in Figure 5a,b depict the correlations for the kinase/FlexX and kinase/Screenscore data sets (using loss function 2), respectively. A high correlation can be observed in both cases, the correlation coefficients are 0.95 (kinase/FlexX) and 0.98 (kinase/Screenscore). In the Screenscore case two point clusters can be seen. The additional high-value cluster corresponds to sampled unfavorable regions. In the right-hand plots of Figure 5a,b the results of the sensitivity analyses are shown. The sensitivity analysis measures the relative importance of the optimized parameters for the optimization procedure. In the case of FlexX, the value of the ΔG_{lipo} parameter is the most important parameter; in the case of Screenscore it is the value of $\Delta G_{\text{plp\_steric}}$.

3.3. Optimizing the Scoring Functions for the ATPase Protein Class. We optimized the parameters of the FlexX and the Screenscore scoring functions for the ATPase protein class. For this purpose we produced 256 random parameter

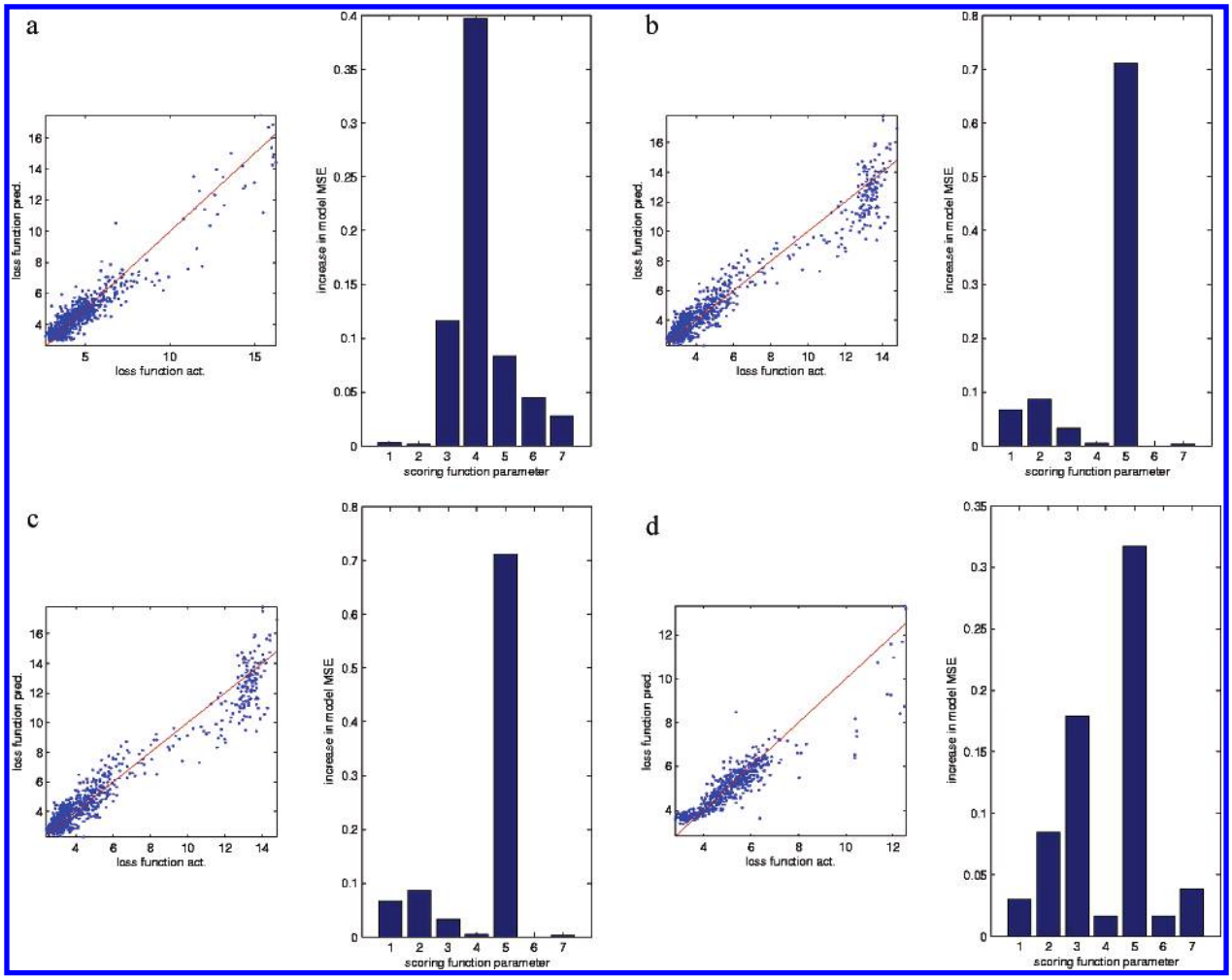

Figure 5. Correlation and sensitivity plots for the (a) kinase/FlexX (b) kinase/Screenscore (c) ATPase/FlexX, and (d) ATPase/Screenscore data sets (loss function 2). The left-hand plots show the correlation of the actual loss function values from docking with the corresponding values on the approximated loss function landscape. The right-hand plots show the results of the sensitivity analysis for each case, and the values correspond to the relative importance of the single parameters for the optimization process (see the Appendix for further details).

sets within the given limitations of the parameter space. Afterward we applied the same optimization procedure as described in the previous section. We performed again three different optimizations, only this time we tested both loss functions for the FlexX scoring function and only used loss function 2 in combination with the Screenscore function.

We again chose the parameter set with the lowest loss function value after optimization for analysis and first calculated the atom positional rmsd of the ligands position in the binding pocket between the best docking pose within the five best ranking solutions of our docking run and the experimental structure. Regarding the average RMSD values in Table 2 and ranking values in Table 3 for the three ATPase parameter optimizations, we observed that for loss function 2 the RMSD values obtained with the original parameter sets are much worse than in the kinase case and that the improvements through the parameter optimization runs are thus much larger. The final values for the average RMSD are within the same range as for the kinases. This again suggests that we have reached the accuracy limit for the given functional forms of the scoring functions for these protein classes. This time the ranking deteriorates considerably for loss function 1, whereas large improvements can be observed for loss function 2. The latter observation led to our decision to continue our study based on loss function 2. The RMSD values for the single complexes are plotted in Figure 4c,d for the FlexX and Screenscore parameter sets, respectively. This time a significant overall improvement can be observed, especially for the "bad" cases in the standard set (RMSD > 3.0 Å). Some RMSD values drop from 8.0 Å to below 2.0 Å. It is noteworthy that not only the docking quality of the bad cases in the training but also in the validation set improves considerably, indicating that no overfitting was done.

In the next step we again checked the correlation of the actual loss function values from docking with the corresponding values from the approximated loss function landscape and performed a sensitivity analysis. The results are given in Figure 5c,d. A high correlation between the actual and approximated loss function values can also be found in this case. The correlation coefficients are 0.91 for the ATPase FlexX and 0.92 for the ATPase/Screenscore cases, respectively.

The sensitivity analysis of the FlexX scoring function shows that the parameter optimization is strongly influenced by the parameters 3 and 4 (Figure 5d, the label numbering corresponds to the order of the parameters as given in Table 3) and moderately influenced by the parameters 5, 6, and 7. The parameters 3 and 4 are the main interaction parameters for the hydrophilic (ΔG_{match}) and hydrophobic (ΔG_{lipo})

Table 4. Original and Optimized Scoring Parameters for the FlexX and Screenscore Scoring Functions, Optimized for the ATPase and Kinase Protein Classes Using Loss Function 2 and the Upper and Lower Limits of the Parameters Used To Define the Parameter Space of Optimization

data set	ΔG_{const}	$\Delta G_{rotbond}$	ΔG_{match}	ΔG_{lipo}	ΔG_{ambig}	ΔG_{clash}	$R_{phenyl-center}$
orig FlexX	5.4	1.4	1.0	−0.17	−0.17	−0.34	4.5
kinase/FlexX opt	3.346	2.259	2.893	−1.254	−0.538	−1.062	4.86
ATPase/FlexX opt	5.862	3.015	1.492	−1.526	−1.004	−1.045	7.336
lower limits	0.0	0.0	−5.0	−6.0	−3.0	−5.0	2.0
upper limits	10.0	5.0	10.0	3.0	3.0	5.0	8.0
data set	ΔG_{match}	ΔG_{lipo}	ΔG_{ambig}	ΔG_{clash}[a]	ΔG_{plp_steric}	ΔG_{plp_hbond}	ΔG_{plp_rep}
orig Screenscore[a]	1.0	−0.07	−0.07	−0.34	0.12	0.6	6.0
kinase/Screenscore opt	4.580	−0.308	−0.003	−0.083	1.479	0.338	17.252
ATPase/Screenscore opt	1.267	−2.433	1.211	−1.036	0.542	0.161	11.933
lower limits	−5.0	−6.0	−3.0	−5.0	−3.0	−2.0	0.0
upper limits	10.0	3.0	3.0	5.0	6.0	4.0	40.0

[a] The ΔG_{clash} parameter is not part of the original Screenscore function, because the function was originally optimized for reranking purposes only and not for ligand placement. Because here we consider the placement of the ligand, the ΔG_{clash} parameter was added to the original function.

interactions. The overproportional dependence on parameter 4 and the strong dependence on parameter 5 (ΔG_{ambig}) confirms our original observations that hydrophobic interactions are not only important forces in docking for the ATPases protein class, but also that their correct weighting is crucial for an improved scoring scheme. The only geometric term in our parameter set, namely parameter 7 (phenyl-center interactions), does not play a significant role. The sensitivity analysis for the Screenscore scoring function, given in Figure 5d, shows a surprisingly low dependence on the parameters 1 and 2, which correspond to the parameters 3 and 4 in the FlexX case. The parameters which are most important are parameter 3 and 5, which correspond to G_{ambig} and the steric PLP term.

3.4. Performance of the Optimized Parameter Sets in Other Cases. Due to our restriction to the structures of one specific protein class during the parameter optimization, we cannot expect that the optimized parameters work as well as the standard parameters for other protein classes. However, if they are within a physically meaningful, robust range for molecular docking, they should still give decent docking results for other targets. To test this, we performed docking runs for the kinase validation set using the parameters optimized for the ATPase protein class and vice versa. In addition, we checked the performance of the new parameters on the standard FlexX200 data set. For the kinase validation set the average RMSD values obtained with the ATPase/FlexX and ATPase/Screenscore optimized parameters and using the best RMSD within the five best scoring docking solutions (according to the definition of loss function 2) were 1.76 and 2.17 Å, respectively. For the ATPase validation set, the average RMSD values were 2.43 and 3.49 Å for the kinase/FlexX and kinase/Screenscore optimized parameter sets, respectively. In the first case both values are worse than those obtained with the standard and the optimized scoring parameter values (see Table 2). In the second case, the values lie between those obtained with the optimized and standard parameter sets.

To check the performance of the optimized parameter sets on a larger data set, we docked into all structures of the FlexX200 data set. The FlexX200 data set is a set of 200 diverse protein−ligand complex structures, which was designed to provide an efficient test set for molecular docking approaches. We calculated the average RMSD values for the 200 complexes using the best RMSD within the five best scoring docking solutions for the four optimized parameter sets and the original parameters: FlexX score original: 3.06 Å, Screenscore original: 2.78 Å, kinase/FlexX opt.: 3.23 Å, kinase/Screenscore opt.: 2.97 Å, ATPase/FlexX opt.: 3.66 Å, ATPase/Screenscore: 3.41 Å. As expected, the optimized parameter sets do not perform as well as the original parameters. However, the absolute differences are small; therefore, we conclude that our parameter sets lie in a general meaningful range for molecular docking. Due to the tailored training set used, our optimized parameter sets cannot be expected to show an equal performance on a diverse, general test set if compared with the original, nontailored parameter sets.

3.5. Scoring Function Parameters. Comparing the optimized parameters with the original parameters (Table 4) two trends are visible. First, the signs of the parameters stay the same in all but one case, although the boundary limits were chosen large enough to allow for sign changes. This indicates that the relative contributions of the different functional terms are retained. Second, parameter optimization increases the absolute values of the parameters. This can be attributed to the fact that the applied loss function does not contain any energetic terms and, in addition, is optimized with the goal of stronger binding (i.e. lower RMSD) in mind. Thus, this could probably be avoided through an inclusion of binding affinity information into the loss function; in our case this was not possible due to the lack of experimental data for this purpose. However, if such information is available it would be straightforward to include it into the loss function.

4. DISCUSSION

We developed the program POEM for parameter optimization. The program combines the DOE approach with ensembles of various statistical learning techniques and the FlexX molecular docking program. Its performance on a complex rugged parameter landscape was tested by optimizing the FlexX and Screenscore molecular docking scoring function parameters for the kinase and ATPase protein classes.

Instead of stepwise sampling the loss function landscape as applied in traditional optimization approaches, our

approach fits an approximate function to known data points on this landscape, predicts the minima of this function, and feeds the resulting information back to the data set, on which a new landscape is approximated. Thus, from cycle to cycle, as the information on the landscape grows, the predicted minima approach the "real" minima more and more accurately. The main advantage of this approach is that, due to the iterative augmentation of information on the loss function landscape and the algorithms directed search strategy, the required size of the training set and the number of cycles necessary to reach a decent minimum are small. In all of our optimization runs we needed to perform only 200−300 optimization cycles, which corresponds to 600−900 docking runs, to reach a stable minimum, using a training set of 15(16) structures. Because at each cycle we perform docking runs for 3 independent parameter sets (one for each of our "modi"), these calculations can be performed in parallel. Thus our CPU requirement is the same as for 200−300 steps of straightforward optimization. The actual CPU time depends on the docking performance for the single docking runs, which in our case was around 10 s, and the number of structures included in the training set.

The most crucial decisions for an optimal performance of our approach are the choices of an appropriate loss function and of a suitable statistical learning method for its adequate approximation. Thus we spent considerable time optimizing our loss function. We tested various functional forms and combinations of RMSD values. During this development two points became apparent: First, no robust, stable solution could be found if only the energetically/positionally best ranking solution was considered. We found that considering the RMSD of the five best solutions is a reasonable compromise. The robustness of the results was especially enhanced by the use of the fourth best ranking RMSD value within the five chosen solutions instead of the best RMSD. The reason for this is that if using the fourth value the first to third best values are implicitly included in the optimization, thus leading to a more stable result. Second, we observed that using the five docking poses with the lowest RMSD values for the calculation of the loss function led to final results, which had inferior energetic ranking of the pose with the lowest RMSD, than if the five energetically best ranked solutions were used. The latter loss function 2 can enforce reasonable docking poses for the highest ranking solutions. During the search for a suitable statistical learning technique, we found that ensembles of neural networks provided the most consistent and robust results (Figure 2). Therefore we decided to base the approximation of the loss function landscape on this method.

The final results for our two test cases—kinases and ATPases—proved that the POEM optimization approach worked well for its purpose. Three main results could be observed: First, we were able to obtain considerable improvements—up to 6 Å—for all cases, in which bad (RMSD > 3.0 Å) placement was observed with the original scoring functions. Second, using loss function 2 the energetic ranking of the poses with the smallest RMSD also improved in most cases after optimization, the amount of improvement was large if also the average RMSD improved considerably and vice versa (Tables 2 and 3). Third, the maximum performance of the optimized scoring functions is comparable to the maximum performance of the original parameter sets. Because the obtained minimum average RMSD values are within the same range as the values for the original scoring functions, if applied to the diverse FlexX200 test set (see section 3.4), we conclude that we reached the limit of accuracy within the given functional form of these scoring functions. For further improvement it would be necessary to modify the functional form of the scoring functions. This, however, is beyond the scope of this work.

Nevertheless, we are able to improve the performance for cases in which the original scoring functions did not lead to decent docking poses. Especially for our second test case, the ATPase protein class, in which the original scoring functions did not perform well (Table 2), the improvement is considerable: The average RMSD values improved by about 50% for loss function 2 (Table 2). In the case of the kinase protein class this was also the case for the few inaccurate docking poses obtained with the original scoring functions. However, due to the better performance of the original scoring function for most complexes of this class, the overall improvements are much smaller. The large improvements gained for the bad docking cases are a promising result, especially considering that we restricted ourselves to the optimization of the scoring functions coefficients and did not adapt their functional forms.

The optimization strategy and loss functions used in this study were developed for our goal of improving ligand placement. Because this is not the focus in traditional empirical scoring function optimization approaches, which mostly focus on the correct ranking of the given placements, we developed our own RMSD based optimization procedure and loss functions. We did not include any energetic contribution into the loss function due to the lack of consistent, high quality experimental data. However, we found that our procedure can ensure proper ranking of the best placed poses within the given limitations of the used scoring function. Thus we suggest to use the resulting scoring function for ligand placement and combine it with more advanced scoring schemes for the final reranking. In general, our loss function can easily be extended to the inclusion of energetic information, if available. In addition, our optimization procedure is independent of the functional form of the scoring function and can be used for the optimization of various kinds of scoring functions.

5. CONCLUSIONS

The above results allow three major conclusions on the performance of the POEM algorithm. First,—starting from a random distribution—the POEM algorithm is capable of locating a parameter set which shows an improved performance compared with the original docking function parameters. The optimized parameters show optimal performance within the scoring functions functional limits, even for cases in which the original scoring functions performed much worse. It is thus suited not only for the improvement of existing but also for the parametrization of new scoring functions. Second, the POEM algorithm converges rapidly, in our case within 200−300 cycles, and leads to robust minima of the loss function/parameter landscape. Third, the ensemble averaging methodology leads to a smooth approximated loss function landscape, despite the rugged character of the studied landscapes, thus avoiding of getting

stuck in the local minima. Thus, due to its properties the POEM approach is a suitable method for parameter optimization on rugged landscapes, and it would be interesting to evaluate its performance also for other purposes than molecular docking scoring function optimization.

ACKNOWLEDGMENT

Financial support was provided by DFG grant No. Le 491/11 [C.M.]. We would like to thank Jörg Rahnenführer, Holger Claussen, Christian Lemmen, and Andreas Steffen for very helpful discussions and Andreas Kämper for his computational support.

APPENDIX

1. Description of Model Types. Multilayer Neural Networks. We use a multilayer feed-forward neural network (MLP: Multi Layer Perceptron) with the *tanh*(**x**) as nonlinear element. To increase the ensemble diversity, we initialize the weights with Gaussian distributed random numbers with zero mean and scaled variances, following a suggestion of LeCun et al.[25] We use two hidden layers and the number of neurons is chosen at random (3−9 neurons in the first layer, 4−32 in second layer). A first-order training scheme based on the iRPROP+ algorithm[26] with the improvements given in ref 27 is used. As regularization method we use the common weight decay with the penalty term

$$P(\mathbf{w}) = \lambda \sum_{i=1}^{N} \frac{\mathbf{w}_i^2}{1 + \mathbf{w}_i^2} \tag{4}$$

where **w** denotes the *N*-dimensional weight vector of the MLP and the regularization parameter is chosen to $\lambda = 0.001$. Input and output variables are normalized before entering the first-order training.

Nearest-Neighbor Models. A k-nearest-neighbor model takes a kernel-weighted average over the observations y_i in the training set closest to the query point to produce the outcome

$$f(\mathbf{x}) = \frac{1}{\sum w_i} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} w_i y_i \tag{5}$$

where $N_k(\mathbf{x})$ denotes the *k*-element neighborhood of **x**, given a proper metric. [Self matches of data points (i.e. each point is considered to be its own nearest neighbor) are prohibited by default since this would strongly bias the error on the training set.] Common choices for the metric are the L_1, L_2, and L_∞ norm. To compensate for irrelevant input dimensions, distances are computed using a weighted metric:

$$d(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{D} m_i (x_i - y_i)^L \right)^{1/L} \quad 0 \le m_i \le 1 \tag{6}$$

The vector **m** of metric coefficients is adjusted by a genetic algorithm (GA) that works on a population of vectors of metric coefficients. The GA starts with a population of randomly initialized individuals. The higher the dimension *D*, the more of the initial metric coefficients are artificially set to zero to favor sparse solutions. A fitness value is assigned to each individual according to its leave-one-out error on the training training set, which can be computed efficiently with the fast nearest neighbor algorithms ATRIA.[28] The smaller the loss, the higher the fitness value assigned. To create the individuals of the next generation, two individuals of the current generation are selected randomly with a probability proportional to their fitness in order to create two offsprings by straightforward crossover mating. Additionally, random mutations in single coefficients of the offsprings are introduced with a probability 0.2. To prevent losing reasonable solutions, the best solutions of the current generation are copied into the next generation (*N*-elitist approach). The population evolves over a predefined number of generations or until the diversity within the population shrinks below some given threshold.The best individual is selected to become the final vector **m** of metric coefficients.

The smoothing kernel weights w_i are distance dependent:

$$w_i = \left(1 - \left(\frac{d_i}{d_k}\right)^p\right)^p \tag{7}$$

The parameter *p* of this smoothing kernel is chosen out of the set {0.0, 0.5, 1.0, 2.0, 3.0}.

Quadratic Polynomial Models. We implemented quadratic polynomial models

$$f(\mathbf{x}) = \beta_1 x_1 + \beta_2 x_1 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_2 + \ldots \tag{8}$$

by basis expansion of the original parameter vector **x** to a basis containing all products up to degree 2 of the original input variables. The linear coefficients β are computed by ridge regression on this augmented input variables, which are standardized before entering the actual regression. The optimal ridge penalty λ is automatically determined by leave-one-out cross-validation (see ref 9 chapter 3).

2. Out-of-Train Technique. The out-of-train technique is a method for assessing the extra-sample error and can be regarded as a combination of traditional cross-validation (CV) and ensemble averaging. Like in traditional cross-validation, the data set is repeatedly divided into training and test partitions. For one given partitioning, a model is constructed only on samples of the training partition. Test samples are not used for model selection, deriving of stopping criteria or the like. The OOT output for one sample of the data set is the average of the outputs of models for which this sample was not part of the training set (out-of-train). The OOT output can be used to compute estimates of the extra-sample error or extra-sample classification rate.

Unlike Breiman's out-of-bag (OOB) technique (see ref 17) which creates bootstrap replicates of the data set, the OOT technique does not allow samples to occur repeatedly in the training fraction of one model. This could impair some statistical learning algorithms such as the proposed k-NN algorithm. Similar to traditional CV, OOT tends to overestimate the generalization error due to the smaller size of each training partition. Unlike CV, it accounts for the ensemble gain by averaging the outputs of several models. Displaying slightly overestimated error rates should not contradict the conservative approach of this study.

3. Sensitivity Analysis. Sensitivity analysis is a general method for accessing the importance of input variables in supervised learning settings. The method relies on the out-of-train technique (OOT) to compute a measure like the mean

square error (MSE) for the ability G_{orig} of inferring the desired outcome from the input variables given. The information contained in the nth variable of the input data set is destroyed by randomly permuting the values of this variable over all samples while keeping all remaining variables unchanged, resulting in an mutilated *surrogate* data set $D_{n-\text{mutilated}}$. Let $G_{n-\text{mutilated}}$ denote the measure of quality when the OOT output is computed for $D_{n-\text{mutilated}}$ using the models that were trained on D_{orig}. Retraining of models on $D_{n-\text{mutilated}}$, besides being computationally more expensive, would mask the sensitivity of correlated variables. The increase in the MSE after permutation compared to the MSE on the original input set can be seen as a measure for the importance of the nth variable:

$$I_n = G_{orig} - G_{n-\text{mutilated}} \tag{9}$$

This procedure is repeated several times, and results are averaged to produce more reliable estimates. However, the soundness of the estimates of the variable importance I_n depends strongly on a high quality of modeling G_{orig} on the original data set.

REFERENCES AND NOTES

(1) Montgomery, D. C. *Design and Analysis of Experiments*; Wiley: New York, 1991.

(2) Hansen, L. K.; Salamon, P. Neural Network Ensembles. *IEEE Trans. Pattern Anal. Machine Intelligence* **1990**, *12* (10), 993−1001.

(3) Rarey, M.; Kramer, B.; Lengauer, T.; Klebe, G. A fast flexible docking method using an incremental construction algorithm. *J. Mol. Biol.* **1996**, *261* (3), 470−489.

(4) Rarey, M.; Kramer, B.; Lengauer, T. Multiple automatic base selection: Protein−ligand docking based on incremental construction without manual intervention. *J. Comput.-Aided Mol. Des.* **1997**, *11* (4), 369−384.

(5) Böhm, H.-J. The development of a simple empirical scoring function to estimate the binding constant for a protein−ligand complex of known three-dimensional structure. *J. Comput.-Aided Mol. Des.* **1994**, *8*, 243−256.

(6) Stahl, M.; Rarey, M. Detailed analysis of scoring functions for virtual screening. *J. Med. Chem.* **2001**, *44* (7), 1035−1042.

(7) Jansen, M.; Martin, E. J. Target-biased scoring approaches and expert systems in structure based virtual screening. *Curr. Opin. Chem. Biol.* **2004**, *8*, 359−364.

(8) Wang, R.; Xueliang, F.; Lu, Y.; Wang, S. The PDBbind Database: Collection of Binding Affinities for Protein−Ligand Complexes with Known Three-Dimensional Structures. *J. Med. Chem.* **2004**, *47*, 2977−2980.

(9) Hastie, T.; Tibshirani, R.; Friedman, J. H. *The Elements of Statistical Learning*; Springer-Verlag: 2001.

(10) Perrone, M. P.; Cooper, L. N. When Networks Disagree: Ensemble Methods for Hybrid Neural Networks. In *Neural Networks for Speech and Image Processing*; Mammone, R. J., Ed.; Chapman-Hall: 1993; pp 126−142.

(11) Krogh, A.; Vedelsby, J. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*; Tesauro, G., Touretzky, D., Leen, T., Eds.; The MIT Press: 1995; Vol. 7, pp 231−238.

(12) Naftaly, U.; Intrator, N.; Horn, D. Optimal ensemble averaging of neural networks. *Network, Comput. Neural Syst.* **1997**, *8*, 283−296.

(13) Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley Pub Co.: 1989.

(14) Horst, R.; Pardalos, P. M.; Thoai, N. V. Introduction to Global Optimization. *Nonconvex optimization and its applications*; Kluwer Academic Publishers: 2000; Vol. 48.

(15) Lin, T. Y.; Tseng, C. H. Optimum design for artificial neural networks: an example in a bicycle derailleur system. *Eng. Appl. Artif. Intelligence* **2000**, *13*, 2.14.

(16) Kleijnen, J. P. C.; Sargent, R. G. A methodology for fitting and validation of metamodels in simulation. *Eur. J. Operational Res.* **2000** *120*, 14−29.

(17) Breiman, L. Bagging predictors. *Machine Learning* **1996**, *24* (2), 123−140.

(18) Muegge, I.; Martin, Y. C. A general and fast scoring function for protein−ligand interactions: A simplified potential approach. *J. Med. Chem.* **1999**, *42* (5), 791−804.

(19) Gohlke, H.; Hendlich, M.; Klebe, G. Knowledge-based scoring function to predict protein−ligand interactions. *J. Mol. Biol.* **2000**, *295* (2), 337−356.

(20) MacKerell, A. D.; Bashford, D.; Bellott, M.; Dunbrack, R. L.; Evanseck, J. D.; Field, M. J.; Fischer, S.; Gao, J.; Guo, H.; Ha, S.; Joseph-McCarth, D.; Kuchnir, L.; Kuczera, K.; Lau, F. T. K.; Mattos, C.; Michnick, S.; Ngo, T.; Nguyen, D. T.; Prodhom, B.; Reiher, W. E.; Roux, B.; Schlenkrich, M.; Smith, J.; Stote, R.; Straub, J.; Watanabe, M.; Wiorkiewicz-Kuczera, J.; Yin, D.; Karplus, M. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Chem. Phys.* **1998**, *102*, 3586−3616.

(21) Muryshev, A. E.; Tarasov, D. N.; Butygin, A. V.; Butygina, O. Y.; Aleksandrov, A. B.; Nikitin, S. M. A novel scoring function for molecular docking. *J. Comput.-Aided Mol. Des.* **2003**, *17* (9), 597−605.

(22) Muegge, I.; Enyedy, I. J. Virtual screening for kinase targets. *Curr. Med. Chem.* **2004**, *11* (6), 693−707.

(23) Cavasotto, C. N.; Abagyan, R. A. Protein flexibility in ligand docking and virtual screeningto protein kinases. *J. Mol. Biol.* **2004**, *337* (1), 209−225.

(24) Chene, P. ATPases as drug targets: Learning from their structure. *Nature Rev. Drug Discovery* **2002**, *1* (9), 665−673.

(25) LeCun, Y.; Bottou, L.; Orr, G.; Muller, K. Efficient backprop. In *Neural Networks: Tricks of the trade*; Orr, G., Muller, K., Eds.; Springer: 1998.

(26) Riedmiller, M.; Braun, H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*; San Francisco, CA, 1993; pp 586−591.

(27) Igel, C.; Hüsken, M. Improving the Rprop Learning Algorithm. In *Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000)*; Bothe, H., Rojas, R., Eds.; ICSC Academic Press: 2000; pp 115−121, .

(28) Merkwirth, C.; Parlitz, U.; Lauterborn, W. Fast exact and approximate nearest neighbor searching for nonlinear signal processing. *Phys. Rev. E* **2000**, *62* (2), 2089−2097.

CI050036G