Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Theoretical Biology 234 (2005) 187–199
Journal of Theoretical Biology
www.elsevier.com/locate/yjtbi

Directed molecular evolution by machine learning and the influence of nonlinear interactions

Richard Fox*

Codexis, Inc., 200 Penobscot Drive, Redwood City, CA 94063, USA

Received 30 August 2004; received in revised form 3 November 2004; accepted 22 November 2004
Available online 20 January 2005

Abstract

Alternative search strategies for the directed evolution of proteins are presented and compared with each other. In particular, two different machine learning strategies based on partial least-squares regression are developed: the first contains only linear terms that represent a given residue's independent contribution to fitness, the second contains additional nonlinear terms to account for potential epistatic coupling between residues. The nonlinear modeling strategy is further divided into two types, one that contains all possible nonlinear terms and another that makes use of a genetic algorithm to select a subset of important interaction terms. The performance of each modeling type as a function of training set size is analysed. Simulated molecular evolution on a synthetic protein landscape shows the use of machine learning techniques to guide library design can be a powerful addition to library generation methods such as DNA shuffling.

Keywords: Directed evolution; Genetic algorithm; DNA shuffling; NK landscape; Machine learning

1. Introduction

Technologies for protein engineering span a wide spectrum: from rational design based on molecular mechanics/dynamics at one end to random mutagenesis by error prone PCR at the other (van Regenmortel, 2000; Chen, 2001). Throughout this spectrum, recursive methods have been used to improve proteins through successive rounds of evolution. In particular, methods based on recombining beneficial diversity from improved variants have enjoyed great success. Such techniques are often able to obtain significantly improved proteins using fewer assays than required by ultra high throughput screening methods (Stemmer, 1994; Ness et al., 1999, 2002). Such recursive, recombination-based techniques are increasingly applied to protein engineering problems and are the focus of continued study (Kurtzman et al., 2001).

Given the widely observed success of these techniques, the next step in the evolution of protein engineering may be had in finding better ways to accelerate the recombination of beneficial diversity. For this to work, the ability to tease out important elements of diversity that contribute to improved function is key. Machine learning techniques used to analyse multivariate data sets are ideally suited to this task and have been used extensively in many engineering fields. Small molecule quantitative structure activity relationships (QSARs) have been used for many years in medicinal chemistry to improve the search for biologically active compounds (Kubinyi, 1997a, b; Byvatov et al., 2003; Byvatov and Schneider, 2004). Likewise peptide engineering has also made use of machine learning techniques to analyse and create improved molecules (Mee et al., 1997; Cho et al., 1998; Bucht et al., 1999; Lee et al., 2000).

Machine learning techniques are becoming more popular in biological engineering. Several researchers have looked at the fitness landscapes of a number of proteins and protein–protein interactions, building

*Tel.: +1 650 980 5616.
E-mail address: richard.fox@codexis.com (R. Fox).

0022-5193/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jtbi.2004.11.031

statistical models for predicting function given the sequence alone (Lu et al., 2001; Aita et al., 2002; De Genst et al., 2002). To date, less work has been done using statistical models to optimize proteins (as opposed to the much smaller peptides). One example is a peptide-QSAR model that was used to optimize the glycosyl phosphatidylinositol (GPI) modification for a protein with a C-terminal signal peptide (Bucht et al., 1999).

The work presented here is similar to that found in earlier studies that promote the use of machine learning techniques for sequence-oriented evolution (Schneider et al., 1994, 1995a, b). Although these earlier studies were focused on peptides and protein precursor cleavage sites, they are conceptually similar to the current work in as much as proteins can be viewed mathematically and phenomenologically as large peptides. Though one study (Schneider et al., 1994) was criticized for possibly not using enough data to train a machine learning algorithm (Darius and Rojas, 1994), the criticism was later found to be unwarranted as highly active, novel peptides were found using the trained neural nets (Schneider et al., 1998; Wrede et al., 1998).

Two important advancements have taken place over the last 10 years to facilitate the use of machine learning techniques for protein engineering: (1) The ability to generate focused combinatorial protein libraries, (2) The availability of cheap, fast, and accurate DNA sequencing. Together these advancements make the use of machine learning techniques for protein engineering feasible today. For many practical applications of interest, only several dozens or hundreds of variants need to be analysed in order to generate meaningful statistical models that can be used to engineer improved variants, a condition well within today's capabilities.

The viability of using machine learning techniques for the directed evolution of proteins was evaluated earlier (Fox et al., 2003). The machine learning based algorithm begins with the creation of a combinatorial protein library, followed by physical assays that generate sequence/activity data. This information is then used to build statistical models that can then be interrogated to design new libraries. The cycle can be repeated, often adding new diversity at each cycle, by some combination of rational design or random mutagenesis. One perceived limitation of the earlier work is that it assumed a given residue's contribution to fitness was independent of context. Such a linear model may not be capable of identifying important nonlinear interactions between residues. The purpose of this work is to help ascertain the degree to which nonlinear interactions (and the use of models that attempt to capture such interactions) affect the efficiency and robustness of statistical, evolutionary search algorithms that are used to engineer improved proteins.

2. Methods

2.1. Problem coding

Protein variants can be created by any number of techniques that are currently available for use in the construction of combinatorial protein libraries. Such techniques include both classical and synthetic DNA shuffling (Stemmer, 1994; Ness et al., 1999, 2002). In both modes of shuffling the process begins with a fixed set of diversity found in either the homologs (for classical or family shuffling) or from rationally targeted diversity (for synthetic shuffling). This starting diversity is typically far less than the theoretically accessible space—even a small protein of 100 residues has 20^{100} possible sequences. Thus only a subset of the potential sequence space can be examined at any one round of evolution. In practice about 5–30% of a 300-residue chain may undergo some variation in a shuffled library. New diversity can be added in subsequent rounds of evolution (through random or site directed mutagenesis) in order to supply the evolutionary fuel required for further gains in fitness. In the present work, a fixed number of positions are assumed to undergo some degree of variation during one round of evolution. The task here is to examine how efficient the algorithm is for such a fixed search space. The inclusion of additional diversity in subsequent rounds of evolution is a separate, somewhat orthogonal consideration that should not detract from the general applicability and usefulness of the proposed method for optimizing proteins.

Let us assume that we have sequence and activity data for S protein variants. In practice, large segments of the sequence alignment may not contain any diversity between the variants and are excluded from further analysis. This does not mean that the variable positions in the alignment are not interacting physically with the fixed regions of the protein. In fact there is likely to be strong interactions with fixed parts of the protein, but since those parts are not varying they do not play a role in building a statistical model of the *local* fitness landscape. The local fitness landscape consists only of those residues that are undergoing variation. Our goal in building a statistical model is to generate an approximation to the local fitness landscape that can be used to extrapolate to nearby regions of sequence space. A *global* fitness landscape, though ideal to have, is well beyond the scope, capability and often the practical need of the engineering process.

We create dummy variables for the N variable positions within the sequence alignment. The example in Fig. 1a shows the first 27 positions from an alignment of protein variants. For brevity we only show the N-terminal portion of the protein alignment but in practice any number of positions along the length of the protein may undergo variation during a round of

	Position																										
Activity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
6.2	M	T	A	Q	D	D	S	Y	S	D	G	K	G	D	Y	N	T	I	Y	L	G	A	V	F	Q	L	N
3.3	M	T	A	Q	D	D	S	Y	S	D	G	R	G	D	Y	N	T	I	Y	L	G	A	V	F	Q	L	N
4.7	M	T	S	Q	E	D	S	Y	S	D	G	K	G	N	Y	N	T	I	M	P	G	A	V	F	Q	L	N
8.5	M	T	A	Q	D	D	S	Y	S	D	G	R	G	D	Y	N	T	I	M	P	G	A	V	F	Q	L	N
9.2	M	K	A	Q	D	D	S	Y	S	D	G	R	G	N	Y	N	T	I	Y	L	G	A	V	F	Q	L	Q
9.8	M	K	S	Q	E	D	S	Y	S	D	G	R	G	D	Y	N	T	I	Y	L	G	A	V	F	Q	L	N
1.5	M	T	A	Q	D	D	S	Y	S	D	G	R	G	D	Y	N	T	I	Y	P	G	A	V	F	Q	L	N
2.7	M	T	A	Q	E	D	S	Y	S	D	G	R	G	E	Y	N	T	I	Y	L	G	A	V	F	Q	L	Q
6.4	M	T	A	Q	D	D	S	Y	S	D	G	K	G	D	Y	N	T	I	M	L	G	A	V	F	Q	L	N
3.5	M	T	A	Q	D	D	S	Y	S	D	G	R	G	E	Y	N	T	I	Y	L	G	A	V	F	Q	L	N

(a)

	Variable																	
Activity	2T	2K	3A	3S	5D	5E	12K	12R	14D	14N	14E	19Y	19M	20L	20P	27N	27Q
6.2	1	0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0
3.3	1	0	1	0	1	0	0	1	1	0	0	1	0	1	0	1	0
4.7	1	0	0	1	0	1	1	0	0	1	0	0	1	0	1	1	0
8.5	1	0	1	0	1	0	0	1	1	0	0	0	1	0	1	1	0
9.2	0	1	1	0	1	0	0	1	0	1	0	1	0	1	0	0	1
9.8	0	1	0	1	0	1	0	1	1	0	0	1	0	1	0	1	0
1.5	1	0	1	0	1	0	0	1	1	0	0	1	1	0	1	1	0
2.7	1	0	1	0	0	1	0	1	0	0	1	1	0	1	0	0	1
6.4	1	0	1	0	1	0	1	0	1	0	0	0	1	1	0	1	0
3.5	1	0	1	0	1	0	0	1	0	0	1	1	0	1	0	1	0

(b)

Fig. 1. Sample protein sequence alignment with activity data (a) and the method of coding the problem for use in statistical model building (b). Amino acid residues are coded as either present (1) or absent (0) for each variable position in the alignment. Non-variable regions of the alignment are discarded from the problem as they contain no useful information for building the statistical models.

evolution. Fig. 1b shows how we obtain the variable coding for use in the machine learning techniques employed below. The presence or absence of a given residue is coded using dummy variables: 1 (present) and 0 (absent). This coding scheme can be used to handle any number of variable residues at one position. A specific example is shown for position 14 where the variables for 14D, 14N, and 14E are all possible for the 3 residues found at that position. This logical coding is useful when there are only a few variable residues at each position. When many residues are present at a given position, alternative codings, such as those that use the physico-chemical properties of the amino acids, are also possible. There is no limit to the number of variable positions that can be included in the model, though more samples are required as the number of variables increase. Typically, practical considerations, such as the exceedingly high dead rate found in highly diverse libraries, will tend to limit the number of variable sites in any one round of evolution.

After coding the problem with dummy variables, the goal then is to create statistical models based on the measured activities in order to predict new variants not seen before. Further, for purposes of engineering improved variants, we would like to understand which residues, or combinations of residues, contribute to increased activity (also referred to as protein fitness in this work).

2.2. Linear model

The linear model used to associate protein sequences with activity can be written as follows:

$$y = c_{1a}x_{1a} + c_{1b}x_{1b} + c_{1\ldots}x_{1\ldots} + c_{2a}x_{2a} + c_{2b}x_{2b} + \ldots + c_{Na}x_{Na} + c_{Nb}x_{Nb} + c_{N\ldots}x_{N\ldots} + c_0, \quad (1)$$

where y is the response (activity), c_{na} the regression coefficient for the residue choice a at position n, x the dummy variable coding ($x_{na} = 1$ or 0) corresponding to

the presence or absence of residue choice a at position n, and c_0 the mean value of the response. This form of the model assumes there are no interactions between the variable residues—each residue choice contributes independently (as determined by its regression coefficient) to the overall activity of the protein.

2.3. Nonlinear model

The nonlinear model includes a certain number of cross-product terms to account for interactions between residues:

$$\begin{aligned} y = {} & c_{1a}x_{1a} + c_{1b}x_{1b} + c_{1\ldots}x_{1\ldots} + c_{2a}x_{2a} + c_{2b}x_{2b} \\ & + \ldots + c_{Na}x_{Na} + c_{Nb}x_{Nb} + c_{N\ldots}x_{N\ldots} + \ldots \\ & + c_{1a,2a}x_{1a}x_{2a} + c_{1a,2b}x_{1a}x_{2b} + c_{1b,2a}x_{1b}x_{2a} \\ & + c_{1b,2b}x_{1b}x_{2b} + \ldots + c_0, \end{aligned} \tag{2}$$

where the variables are the same as those in Eq. (1) but now we have the additional nonlinear terms, e.g. $c_{1a,2b}$ is the regression coefficient for the interaction between residue choice a at position 1 and residue choice b at position 2. The interaction terms are non-zero only when both residue choices are present in a given sequence, owing to the 1 (present), 0 (absent) coding. Thus the magnitude of the regression coefficient is related exactly to effect on activity that a particular *combination* of residues is predicted to have.

2.4. Regression

We use partial least squares (PLS) regression (Wold, 1985) to solve for the regression coefficients. PLS belongs to a class of regression techniques known as latent variable regression of which principal component regression (PCR) also belongs (de Jong et al., 2001). The strength of such techniques lies in the fact that the actual regression step is performed against a smaller number of latent variables that consist of a linear combination of the primitive input variables (in our case the presence/absence of specific residues and/or their combinations). Performing such dimensional reduction before regression typically results in more robust models with improved predictive power. In PCR, the latent structure between the input variables alone is used to create the latent variables. In contrast, with PLS, the latent structure between input and output variables is used to generate the latent variables. Incorporating information from the response (output) variable before dimensional reduction usually leads to models with enhanced predictive power.

Clearly, there are many other viable machine learning techniques one could use here, such as neural networks and support vector machines (SVM). Earlier work using a linear SVM kernel showed worse performance compared to the linear PLS model (data not shown). It is certainly possible that a nonlinear SVM kernel or other nonlinear approaches would perform better than the one described here. However, the purpose of this work is not to identify the optimal machine learning technique, but to investigate the importance and applicability of machine learning techniques in general to the problem of protein engineering. Moreover, the use of PLS along with genetic algorithm-based feature selection techniques described below is well established in the QSAR and peptide QSAR fields (Kubinyi, 1995; Hasegawa et al., 1997; Leardi and Lupianez, 1998; Hasegawa et al., 1999; Daren, 2001) and therefore serves as an appropriate starting point for investigating its usefulness in protein engineering.

2.5. Determination of nonlinear terms

Assuming for the moment that there are only 2 residues present at each of the N variable positions, then in addition to the $2N$ linear terms in Eq. (2) there are $2N(N-1)$ possible nonlinear terms one could add to the regression equation. For 20 variable positions (each with 2 options) this would result in 760 possible interaction terms, for a total of $40+760=800$ terms altogether. This geometric explosion of variables leads to great flexibility in the model and the increasing likelihood of assigning spurious correlations to the regression coefficients. It is often preferable to ascertain a subset of the most important terms in order to obtain parsimonious models with good predictive power.

Selecting the best terms to use in building statistical models is also known as feature selection and is the subject of much study. Finding the optimal subset of terms to use in building statistical models can be prohibitively expensive if you want to guarantee you have obtained the optimal subset. For example, if we wish to select the best way to choose 80 terms out of the 760 available nonlinear terms, there are over ${}_{760}C_{80} = 10^{109}$ possible ways of doing this—each possibility must be computed separately in order to determine its effectiveness. Even if we restrict the set of possible features to include only combinations of variable positions (rather than each residue explicitly) and choose just 20 terms, there are still over ${}_{190}C_{20} = 10^{11}$ possible subsets.

The problem of finding the best set of features (in our case the interaction terms) is akin to an optimization process. We look to approximate methods capable of finding optimal or near optimal solutions in a practical compute time. To this end the use of genetic algorithms is known to be an effective means of feature selection and has been used in multivariate regression problems to achieve models with enhanced predictive power (Ozdemir, 2002).

Before discussing the genetic algorithm based approach, it should be noted, however, that there are other

feature selection techniques available. They include forward selection, reverse selection and various "add a few remove a few" variations. In forward selection, all possible terms are analysed separately and the one that improves performance the most is selected first. The process continues until the addition of subsequent terms is no longer statistically significant (as measured by some appropriate statistical test such as an F test). In reverse selection, all the terms are added at one time, and the least important term is removed first. The process continues until removal of the least important remaining term is statistically significant. The variations of the forward/reverse selection procedures typically iterate between the two methods until some reasonable convergence criterion is met. The primary disadvantage of the forward selection process is that it is prone to missing important interdependent pairs. Conversely, the reverse selection process, while able to capture important interdependent pairs, suffers from the "curse of dimensionality", where the explosion of variables leads to models that capture spurious correlations and have weakened predictive power.

Also, it should be noted that the explicit introduction of nonlinear terms followed by standard PLS regression is not the only method available for capturing interaction effects between variables. Alternative techniques include those based on SVM-like kernel-type methods (Rosipal and Trejo, 2001; Bennett and Embrechts, 2003) (Rosipal and Trejo, 2001) as well as other generalized nonlinear methods (Berglund and Wold, 1997; Baffi et al., 1999).

Genetic algorithms (GAs) are powerful, generic algorithms that are often used to obtain optimal or near-optimal solutions to complex optimization problems. They are based on the search efficiency observed in natural evolutionary systems. The power of the GA lies in the ability to conduct massively parallel searches for beneficial sub-solutions and recombine them to create more fit variants. In order to leverage this search efficiency the optimization problem is coded in terms of a population of chromosomes and their constituent genes. The task is then to evolve individual chromosomes, exchanging and mutating genetic material between members of the population through successive rounds of evolution. Improved individuals are used as preferred breeding stock in new generations.

For the specific optimization problem we are interested in here (finding an optimal set of interaction terms for use in regression) we can specify an individual chromosome as a model consisting of all possible linear terms and a fixed number of genes corresponding to a set of nonlinear terms. Each gene is used to map to a set of interaction terms that exists between two variable positions in the statistical model. Thus, in one chromosome, say, the gene for interaction terms between positions 2 and 14 is included, and in another chromosome, say, the gene for interaction terms between positions 3 and 5 is included. All the interaction terms corresponding to a given pair of positions are then used in the regression model. Thus, if the gene corresponding to the interaction terms between positions 2 and 14 (see Fig. 1) exists in a given model (chromosome) then the following nonlinear terms are included in the regression equation:

$$c_{2T,14D}x_{2D}x_{14D} + c_{2T,14N}x_{2D}x_{14N} + c_{2T,14E}x_{2D}x_{14E} + c_{2K,14D}x_{2K}x_{14D} + c_{2K,14N}x_{2K}x_{14N} + c_{2K,14E}x_{2K}x_{14E}.$$

For example, the regression coefficient $c_{2T,14D}$ corresponds to the additional increase/decrease in activity attributed to the combination of 2T and 14D when it exists in a specific protein.

In order to select improved models for mating, the fitness of each model (chromosome) is calculated using the root mean square errors in cross validation ($RMSECV$):

$$RMSECV = \sqrt{\sum_{i=1}^{i=S} \frac{(y_i - \hat{y}_i)^2}{S}}, \quad (3)$$

where y_i is the measured activity of ith data point, $\hat{y}_i$ the predicted activity of the ith data point (based on a model trained without it) and S the number of samples in the training set. Cross-validation is a technique used to estimate how well a particular model can predict data points not seen during training (in this case linear regression). Typically, one or more data points are held out of the training set and the resulting model is then used to predict the held out data. Here we hold out 10% of the samples and build a model based on the remaining 90%. The process is systematically repeated 10 times so each data point is held out exactly once. The square of the difference between the measured and predicted values of those samples held out of training contributes to the $RMSECV$ score. Models with lower $RMSECV$ scores are deemed more robust since they tend to predict better samples not seen during training. The $RMSECV$ score is distinctly different from the ordinary residual sum square error ($RMSE$) of the final regression model, which includes all of the original data points in the training set. Though $RMSE$ is essentially guaranteed to be lower than $RMSECV$, it is not typically the best measure of model performance since it only indicates that the final model is self-consistent, but does not guarantee that new data points (those not seen during training) can be robustly predicted.

Initially, a set of 30 chromosomes is randomly generated, picking genes from the set of all possible pairwise interactions. The population fitness is then calculated and the members are rank ordered according to fitness (lower $RMSECV$ scores correspond to a higher fitness). The 3 best scoring models are

automatically retained as members for the next round of evolution—this is known as elitism and guarantees that the algorithm retains the current best solution. Twenty-seven more models (for a total population of 3 elites + 27 children = 30) more progeny models are then created for the next generation by preferentially mating parent models with lower *RMSECV* scores—the lower the *RMSECV* score the more likely the model will be used as breeding stock for the next generation. During mating, a partially matched crossover operator is used to breed two models with each other (Goldberg, 1985). The crossover probability is set equal to 1, meaning that on average there is one crossover between two parents when a single child is produced. The partially matched crossover operator guarantees that each progeny model contains only one instance of a particular interaction term. A mutation rate of 5% is then applied to each progeny model—alleles (alternative interaction terms) are drawn randomly from the pool of available terms not already present in a given model. Experiments with different population sizes, mutation rates, and selective pressures do not show marked differences in the search efficiency or effectiveness—a result typical of the robust nature of genetic algorithms.

The evolution proceeds until the best *RMSECV* scoring model remains constant for at least 25 generations. This optimized model is then used for prediction and engineering of new proteins.

2.6. Directed molecular evolution in silico

After the variable selection and regression steps are complete, we can then interrogate the model in order to engineer the next round of variants. This next step involves the creation of more focused libraries, where a set of residues predicted to confer higher fitness is fixed in a new protein backbone. The remaining set of residues that are predicted to be less important are toggled in the next round library in hopes of achieving yet higher fitness. This gives the next round library the best chance of locking in beneficial residues while continuing to access potentially beneficial regions of sequence space.

To ascertain the importance of each residue we can evolve now, *in silico*, new proteins using the statistical model developed above. The information contained in these computer-evolved proteins can then be used to synthesize novel proteins in the lab and test them on physical assays. This is the crux of the matter: if we can obtain a reasonable *in silico* approximation of the actual, *local* experimental fitness function, we can use the information to perform *in silico* evolution much more rapidly and than required by in vitro screening and evolution, possibly sidestepping additional rounds of in vitro evolution or achieving higher fitness gains for the same amount of bench time effort.

We conjecture that the optimization of the resulting nonlinear statistical model belongs to a class of optimization problems known as *NP*-hard, appearing analogous to the protein design problem (Pierce and Winfree, 2002) and the maximum satisfiability problem (Raman et al., 1998). Given the well-documented intractability of *NP*-hard problems, our particular optimization problem is unlikely to be solved in less than exponential time, by say, evaluating all 2^N possibilities. Unfortunately, such an exhaustive search is computationally expensive for large N and we prefer to use an approximate algorithm instead.

We again turn to the genetic algorithm for solution to a complex combinatorial optimization problem—this time to optimize the best-predicted protein given an *in silico* fitness function. For this genetic algorithm the chromosomes now correspond to a protein and the genes correspond to residues. This process is analogous to directed evolution by DNA shuffling, where the child proteins contain a mix of residues from both parents. To perform the evolution we start with a random population of 200 proteins (selected from the space of possible sequences contained within the space of variable positions and the residues they are observed to have). Next, we apply an elitism rate of 5% (take the top 10 forward each round), preferentially breed the more fit proteins together (those predicted to have higher fitness) using a crossover probability of 1, and apply a mutation rate of 5% to each child protein. Recursive cycles of selection, mating, breeding and mutation are performed until the best computer generated protein is selected after no improvement is seen for at least 15 generations.

For the linear model, the GA is not required as the effect of each residue change is independent from the variable state at other positions and the best predicted protein is just one that ensures every term in Eq. (1) is maximized (i.e. choose the residue with the positive regression coefficient).

2.7. Next round library design

Rather than synthesize the single best-predicted protein we seek to generate a focused combinatorial library of proteins based on the information contained in the statistical model. We define the sensitivity of a given residue choice as the average change in predicted fitness between a given residue choice at a particular position in the best-predicted protein and the alternative choices available at that variable position. Because the best-predicted protein is usually of high fitness, each alternative choice is often met with a reduced predicted activity. We then rank order these sensitivities (fitness changes) from largest to smallest. Those with the largest sensitivities then correspond to residues choices that are important to increased protein function. These residue choices are then fixed in the next round protein

backbone. We are then free to vary residues with smaller sensitivities, leading to the creation of a focused combinatorial protein library.

2.8. Synthetic data source

The concept of a fitness landscape dates all the way back to Wright's seminal studies in theoretical biology (Wright, 1932). Such fitness landscapes enable us to think about important features of molecular evolution and to ask questions we would not normally be able to answer given the limitations on the number and type of laboratory experiments we can conduct with real proteins.

In order to assess the performance of the above models described we make use of a synthetic data source known as the *NK* fitness landscape (Kauffman, 1993), where N is the number of variable positions in a simulated protein and K is the epistatic coupling between variable residues. The value of K ranges from $K=0$, where each variable residue's contribution to fitness is independent of every other position, to $K=N-1$, where each variable residue's contribution to fitness depends on the state of every other variable position. The fitness function for the *NK* landscape can be written as follows:

$$W=\frac{1}{N}\sum_{i=1}^{N} w_i, \tag{4}$$

where w_i is the partial contribution to fitness at variable position i, N the number of variable positions, and W the total protein fitness (activity). The partial contributions to fitness (the w_i values) are calculated by populating a lookup table of random values drawn from a gamma distribution with a mean and variance of 0.35. For every combination of the residue choices at position i and K other positions that are epistatically linked to it, the table contains a randomly generated partial fitness value (placed in the table when the landscape is first created). The K other positions that are epistatically linked to position i can either be chosen randomly or set manually. The choice does not appear to alter the statistical features of the landscape. In this work, we set them manually so as to gauge whether our machine learning algorithms are able to tease out the interaction terms we specify to exist.

The choice for the value of K is system dependent. When only a few residues are varied and they are widely separated in 3D structure (>10 Å), K is more likely to be small ($K<0.5$). Conversely, when many residues are varied and they are close in 3D structure (<4 Å), K may tend to be higher ($K>1$). The effect of K on linear modeling was evaluated earlier (Fox et al., 2003). Current best estimates from unpublished data indicate for a number of systems of practical interest K is not likely to be very large ($0<K<3$). In this work, we choose the value $K=1$ as an intermediate choice between very smooth and more rugged landscapes. Very rugged landscapes are expected to respond differently to various evolution and machine learning strategies.

NK landscapes and their related spin glass models studied in physics are designed to capture the important, statistical features of real protein landscapes and have been used in a number of theoretical studies (Bogarad and Deem, 1999; Earl and Deem, 2004; Yu and Levitt, 2004). As always, care should be taken in interpreting and extrapolating the results of this or any other study based on purely simulated data.

2.9. Model for DNA shuffling and HTP screening

In order to compare the effectiveness of the machine learning-based search algorithms with more traditional search algorithms such as DNA shuffling or high throughput screening (HTP), we need to capture the salient aspects of these latter techniques. In one DNA shuffling format, the one simulated in this work, genes that code for improved proteins are taken as parents and bred together in the next round of evolution using homologous recombination by random fragmentation followed by (PCR). A detailed description of the model used for comparison can be found elsewhere (Fox et al., 2003).

The HTP search algorithm is straightforward to implement and consists of screening a number of variants equal to the size of the HTP screen. No 2nd round library is constructed for this method. The highest fitness obtained is that found in the best protein in the 1st screen.

3. Results

Using the *NK* landscape described above, we generated an initial training set with $S=40$ samples, $N=20$ variable positions (with two residue choices available at each position) and $K=1$ (to reiterate, for $K=1$ each variable position is functionally coupled to one other variable position). Another $T=100$ samples were generated for use in testing the quality of the statistical models. The process for creating training and test sets was repeated again, but this time using $S=100$ samples in the training set and another $T=100$ samples for the test set. The training sets were used to construct both linear and nonlinear models using the methods described above.

For the nonlinear feature selection method, a fixed number of 20 interacting position pairs (corresponding to 80 nonlinear terms in the regression equation) were selected using the algorithm described above. Ideally, the feature selection algorithm would find the true 20

interacting positions of the underlying *NK* fitness landscape since $K = 1$ and every position is defined to interact with exactly one other position. In practice, it is not generally possible to know the true number of interacting positions *a priori* and some domain knowledge would likely be helpful in fixing the upper limit of interacting positions to search for. We did not study the effect of searching for more or less than the true number of interacting positions known to exist in the *NK* landscape. One might predict that searching for too many terms might lead to overfitting the training data (leading to models with poor predictive power), while not searching for enough of the important interaction terms would lead to underfitting the true structure of the fitness landscape (also leading to models with less than optimal predictive power).

For the smaller training set size of $S = 40$, the linear model was capable of correlating the measured and predicted values reasonably well, but demonstrated weaker correlation when tested against data not seen in the training set (Fig. 2a). This result was interesting considering the linear model was used to model a fundamentally nonlinear fitness landscape. In this case, the linear model is able to capture reasonably well the *average* contribution to fitness for the choice of a given residue, independent of context. With enough of these average contributions taken together, the linear model could roughly predict the measured response. The test results for the linear model were marginally better when the training size was increased to $S = 100$ (Fig. 2b). At some point the linear model's predictive power ceases to increase regardless of the amount of training data available. This tendency of low complexity models to underfit the data is known as the bias and it stems from the inability of the model to capture the true underlying problem structure (Hastie et al., 2003).

When the nonlinear model (using variable selection) was trained using only $S = 40$ samples, the correlation for the training set was excellent (Fig. 2c). Unfortunately, the model contained low predictive power, as evidenced by its poor correlation with measured values in the test set. Clearly, this nonlinear model, with so

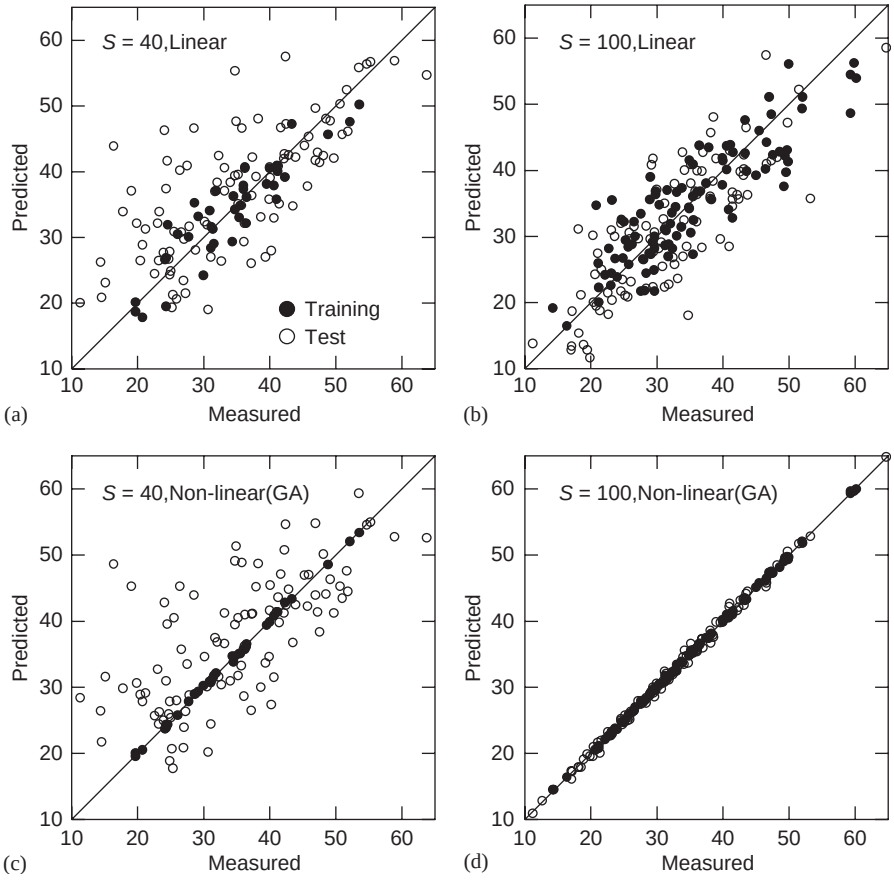

Fig. 2. The effect of training set size and modeling strategy. (a) through (d) display the predicted and measured fitness values of various models. The *x*-axis is the measured fitness value, computed using the synthetic data source described in the text. The *y*-axis is the model's predicted fitness. The dark circles represent the training predictions (data used in training is the same data used in prediction). The open circles represent the test predictions (the data set used in training is discarded and a new data set not seen by the model is used in prediction). *S* is the number of samples used to train each model. Two different modeling strategies are shown, one with only linear terms in the regression equation, and the other a subset of nonlinear terms selected, using a GA strategy described in the text, to optimize prediction accuracy.

many potential variables, and paucity of training data, was able to essentially memorize the data set it was trained on. This tendency of high complexity models to overfit the data is known as the variance. This bias/variance tradeoff represents a fundamental problem in machine learning (Hastie et al., 2003) and some form of validation or testing is almost always required to address it when dealing with new or uncharacterized machine learning problems. Satisfyingly, for the larger training set ($S = 100$) the nonlinear model performed exceedingly well in both the training prediction and, more importantly, the test prediction (Fig. 2d). The test predictions were so good that most of the data points are obscured by the dark circles use to plot the training set.

The results shown in Fig. 2 are only a snapshot of the results for one randomly generated *NK* landscape. Statistical studies were required to better ascertain trends in the various strategies. To this end a series of 20 landscapes were generated and a matrix of strategies were tested against each landscape and the results were averaged for each strategy. The matrix consisted of varying the training set size ($S = 20$, 40, and 100) and using different machine learning strategies: linear, nonlinear (using GA-based variable selection), and nonlinear (using all variables). Three different correlation metrics were used to assess the performance of the various modeling techniques. In each case the correlation coefficient between the measured and predicted fitness values was recorded and averaged over the 20 landscapes. The correlation coefficient used to asses model performance is defined as follows:

$$r^2 = \frac{\left(\sum_{i=1}^{M}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})\right)^2}{\sum_{i=1}^{M}(y_i - \bar{y})^2 \sum_{i=1}^{M}(\hat{y}_i - \bar{\hat{y}})^2}, \quad (5)$$

where y_i is measured value of the ith data point, $\hat{y}_i$ the predicted value of the ith data point, $\bar{y}$ the mean value of the measurements, $\bar{\hat{y}}$ the mean value of the predictions, and M the number of data points in the training or test set.

The first metric used to judge a model's performance is that of the correlation coefficient based on predicting the training data itself and is referred to as the training set correlation coefficient. This metric measures a given strategy's ability to generate internally consistent models.

A second metric is based on the correlation coefficient observed when performing cross-validation and is referred to as the cross-validation correlation coefficient. For this cross-validation, a total of 20 random 50/50 splits of the training data were performed, using one split to predict the values in the other. In general this results in a more conservative estimate of a model's ability to predict data points not seen in the training set, provided no information from the left out data has leaked into the machine learning process, either directly or indirectly (a subtlety discussed in a moment).

The third metric, and the only real gold standard in measuring a modeling strategy's predictive power, consists of predicting an entirely new set of test data not used in any way to train the model, viz. the test set correlation coefficient. To measure this metric a new set of $T = 100$ variants is generated and the model developed based on the training data is used to predict the activities of the new variants.

The results of the numerical experiments are shown in Fig. 3 for each correlation coefficient metric described above. The error bars represent 95% confidence intervals of the mean correlation coefficient averaged over 20 runs. For the training set metric the correlation coefficients are generally high for both linear and nonlinear approaches alike. The strategy of using a nonlinear model with variable selection is capable of building a near perfect (internally consistent) model each time, regardless of the number of data points in the training set. However, this is no guarantee of predictive power. For the linear modeling strategy the correlation coefficient actually decreases somewhat as the training size increases. This is due to the fact that the linear model, with a relatively low capacity for over fitting data is less able (less prone) to fitting spurious correlations as the training size increases. With more observations the linear model cannot account for as much of the variation that is actually present in the data.

The cross validation metric generally demonstrates lower correlation coefficients compared to those found in the training set. This is due to the fact that the models are being asked to predict data points held out of the regression step. For the nonlinear model with variable selection the cross-validation correlation coefficient appears to be markedly better than the methods that do not employ variable selection. However, because the cross-validation correlation coefficient is effectively included in the variable selection process (by minimizing *RMSECV* during evolution), information from the cross-validated points can creep into the training strategy. Even though the regression step itself does not include the predicted data points, the data points to be predicted were used in the evolution of cross products in such a way as to be predicted well upon cross-validation. In this case the cross-validation metric should be used with caution. In some ways this is a testament to the power of the genetic algorithm—it can create consistent solutions given a finite set of data, but the power to find such solutions may lead to false models of reality. This is an inherent problem in using such strategies in the absence of reliable metrics that can be used to limit in the model's inherent capacity to be over trained.

The test metric is the best measure of a given modeling strategy's true predictive power. For small

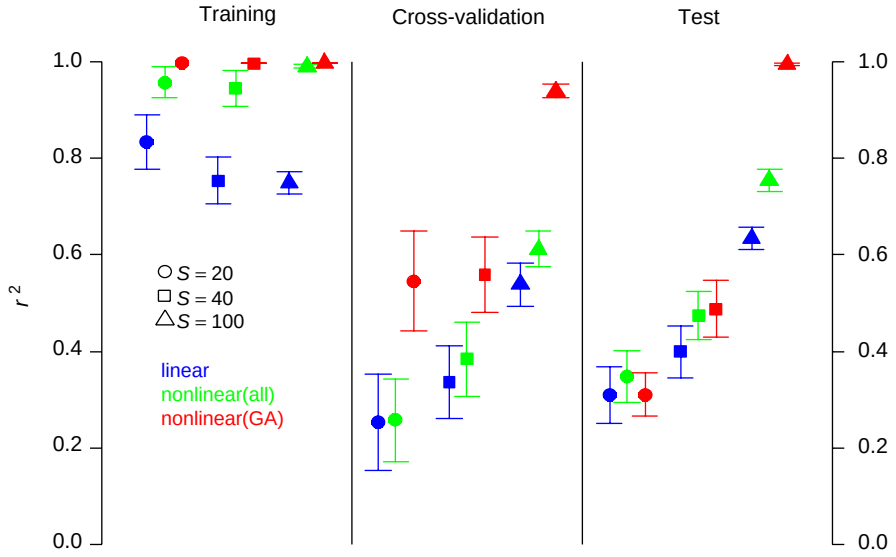

Fig. 3. Correlation coefficients for various training set sizes and modeling strategies using an *NK* synthetic data source with $N = 20$, $K = 1$ and 2 residue choices at each variable position. The *y*-axis is the squared correlation coefficient, r^2, between measured and predicted fitness values. The training set sizes, S, correspond the number of sequence-activity samples used to build each model. The linear modeling strategy (blue) contains only the 40 linear terms in the regression equation. The nonlinear(all) strategy (green) additionally includes all 760 possible pairwise interaction terms. The nonlinear(GA) strategy (red) employs a genetic algorithm based variable selection technique that includes only 80 additional cross product terms that are putatively important. Each value is calculated by averaging the correlation coefficient over 20 runs. The error bars represent the 95% confidence interval for the true mean of the correlation coefficient. The training data corresponds to the correlation found between the measured and predicted values of the full training set. The cross-validation data corresponds to the correlation found when randomly holding out 50% of the samples in the training set and predicting the held out samples using the remaining 50% for training. The test data corresponds to the correlation found when predicting a new set of 100 samples not seen during training.

training set size ($S = 20$), all three strategies suffer from relatively low correlation coefficients. The results improve somewhat with higher training set sizes ($S = 40$). Interestingly, at these training set sizes the linear model is not altogether much worse (practically speaking) than the nonlinear models. The increased capacity of the nonlinear models is balanced somewhere in the middle of the bias/variance tradeoff, resulting in models with a greater ability to capture the true problem structure but have some propensity for overfitting. Finally, at $S = 100$, sufficient training data exists to create excellent models using the strategy employing nonlinear models and GA-based variable selection.

The use of random 50/50 splits for cross-validation can lead to an underestimate of the generalization capability of the model, especially for models that did not employ variable selection. For example, see Fig. 3 for the nonlinear(all) case when the training size $S = 100$. The test correlation is significantly higher than that found in cross-validation. The 50/50 split tends to be more conservative than the 90/10 split used earlier in the variable selection routine itself because fewer samples are available in the regression step. We originally used a leave-one-out cross-validation method to assess model performance but the generalization ability was often overestimated, especially for the nonlinear variable selection case (data not shown). The 50/50 cross-validation split seemed to given an appropriate balance between the overestimate of generalization ability for models with variable selection compared to the underestimate found in models without variable selection.

Ultimately, we are interested in using the statistical models developed above for protein engineering. To this end we conducted numerical simulations of directed molecular evolution. Each simulation consisted of the following: (1) Generate S random protein variants. (2) Screen the S variants using a randomly generated *NK* landscape. (3) Rank order either the variants or the residues from their highest to lowest values. (4) Engineer another library of variants using alternative evolution strategies based on the rankings in step 3 and screen the library on the same fitness landscape. The highest fitness found in screening 100 random variants in the 1st round served as the baseline fitness against which all other strategies were then be compared.

Each evolution strategy was tested using different screening sizes and subsequent round library sizes: HTP (to simulate 1st round high throughput screening), Classical (to simulate recombination-based evolution, viz. DNA shuffling), and linear/nonlinear modeling approaches (to simulate the machine learning based approaches). The simulations were based on averaging the best fitness values (after subtracting the baseline fitness) obtained after the 2nd round of evolution on 300 randomly generated *NK* landscapes, beginning with $N_1 = 20$ variable positions and two options at each

position for the 1st round and $K = 1$. For the HTP algorithm only the best 1st round fitness value was recorded since no evolution took place for this evolution strategy. Holding the effective library size constant in the 2nd round is important in order to achieve fair comparisons between the algorithms. Each search strategy was tried using two different effective library sizes for the 2nd round of evolution: $N_2 = 10$ and 5 variable positions, corresponding to sizes of 1024 and 32, respectively. The effective library sizes were calculated using the method described in earlier work (Fox et al., 2003).

In general the HTP screening strategy was inefficient at achieving a certain fitness value for a given screening size (Fig. 4). In contrast, the classical, recombination based strategy was much more economical in terms of the total number of assays needed to achieve a certain fitness value. For example, screening 100,000 variants in HTP yields slightly lower activity than that achieved by classical, recombination-based evolution screening only 1000 variants per round over 2 rounds. In fairness, however, there are time and materials costs associated with multiple rounds of evolution that are not captured in this comparison. The relative efficiency of the classical strategy compared to massive HTP sampling has been well studied. DNA shuffling is analogous to a genetic algorithm practiced in a laboratory environment (Gustafsson et al., 2001). The same GA search efficiency used with success in developing the machine learning techniques presented here are also present in the classical, recombination based techniques currently used to engineer improved proteins.

When the 2nd round library size was kept larger (2^{10}) the fitness increase for the classical method was superior to the smaller (2^5) 2nd round library, illustrating the importance of maintaining diversity in GA-based systems. The need for diversity preservation is consistent with R.A. Fisher's fundamental theorem of natural selection, which states, "The rate of increase in fitness of any organism at any time is equal to its additive genetic variance in fitness at that time" (Fisher, 1930). The stark contrast between the HTP and classical, recombination based methods exemplifies the very low probability of randomly generating proteins with high fitness, even when sampling a large number of variants. Such high combinatorial complexity makes anything like an exhaustive search very inefficient, as too many stars have to align to achieve high fitness, reminding us of Fisher's famous aphorism, "natural selection is a mechanism for generating an exceedingly high degree of improbability" (Edwards, 2000).

For the evolution strategy based on the nonlinear modeling, the interaction terms were not identified using the methods described earlier, i.e. evolving for terms using a GA to find an optimal subset. Instead, because the larger training size ($S = 100$) was seen earlier to be sufficient to consistently identify all the important nonlinear terms (and the algorithm for calculating them is otherwise time consuming for 300 separate runs), profound knowledge of the interacting positions were fixed in the nonlinear model equation.

The machine learning-based algorithms (both linear and nonlinear) fared better than either the classical or HTP algorithms in terms of the number of assays

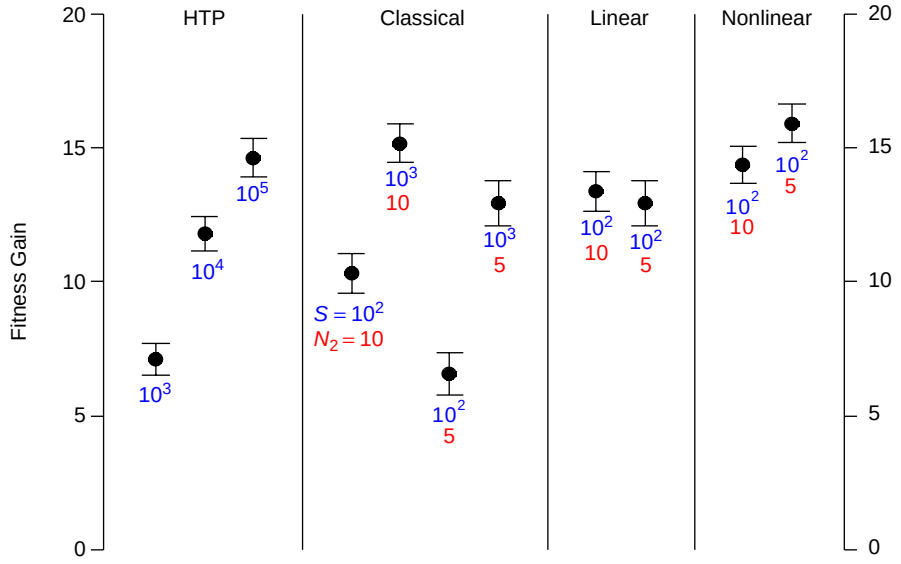

Fig. 4. Comparison of different directed evolution strategies: HTP (high throughput screening), Classical (recursive, recombination based, viz. DNA shuffling), Linear and Nonlinear (machine learning guided). The y-axis shows the fitness gain achieved for each strategy after two rounds of directed evolution (HTP has only one round). Each value is calculated by averaging the fitness gain over 300 simulated runs. The error bars represent the 95% confidence interval for the true mean of the fitness gain. Each strategy lists the number of variants screened in each round (blue), S, and for the recursive strategies, the second round library size (red), N_2.

required to achieve a certain fitness value. For example, the nonlinear strategy ($S = 100,\ N_2 = 5$) was more efficient than either the classical ($S = 1000,\ N_2 = 10$) or HTP ($S = 100,000$) strategies as they all achieved nearly the same fitness, but at much higher screening costs for the latter approaches. Unlike the classical strategy, the machine learning based algorithm did not generally suffer when making smaller, more focused 2nd round libraries. In fact, for the nonlinear model, the fitness level improved the most when going to smaller, more focused libraries.

The improvement using the best nonlinear strategy ($S = 100,\ N_2 = 5$) compared to the best linear strategy ($S = 100,\ N_2 = 10$) was statistically significant (p-value $< 2.7 \times 10^{-6}$). However, the linear modeling strategy was still relatively efficient compared to the classical and HTP strategies and has the virtue of being easily interpreted by biologists and biochemists looking to ascertain the importance of various residue choices. Conversely, the nonlinear model may prove useful when interaction information can be combined with other information (such as that derived from protein structure) to help guide researchers and protein engineers to investigate areas of the protein that may be sensitive to nonlinear effects.

4. Conclusion

The results presented here extend earlier work in the emerging field of machine learning based directed evolution of proteins. Specifically, the impact of using nonlinear terms in the modeling step was investigated. The results of simulated molecular evolution show that for mildly rugged fitness landscapes the use of nonlinear terms may be warranted provided enough data can be generated to train the models. For sparse data sets, the more robust and easy to interpret linear models outperform alternative strategies such as high throughput screening and classical recombination-based evolution. The simulations reveal that large gains in efficiency over high throughput screening can be achieved firstly by recombination based approaches and secondly by further coupling with machine learning guided evolution.

Because the geometric explosion of all possible pair wise interaction can lead to models with poor predictive power, it behooves us to look for ways to reduce the set of potential terms. We can start by including just those interaction terms that represent residue positions that are close together in 3D structure. For example, those with side-chains close enough to be within van der Waals contact (< 4 Å). Such residues are more likely to exhibit nonlinear interactions than those that are distant from one another in structure (Wells, 1990). This would limit the model complexity to terms that are more likely to capture real effects (instead of spurious correlations), resulting in more robust predictions and improved engineering outcomes.

Some of the methods described here are currently being used to optimize full proteins and we hope to present the experimental results in a future publication.

References

Aita, T., Hamamatsu, N., Nomiya, Y., Uchiyama, H., Shibanaka, Y., Husimi, Y., 2002. Surveying a local fitness landscape of a protein with epistatic sites for the study of directed evolution. Biopolymers 64, 95–105.

Baffi, G., Martin, B., Morris, A., 1999. Non-linear projection to latent structures revisited: the quadratic PLS algorithm. Comput. Chem. Eng. 23, 395–411.

Bennett, K., Embrechts, M., 2003. An optimization perspective on partial least squares. In: Suykens, J., Horvath, G., Basu, S., Micchelli, J., Vandewalle, J. (Eds.), Advances in Learning Theory: Methods, Models and Applications, NATO Science Series III Computer & Systems Sciences. IOS Press, Amsterdam, pp. 227–250.

Berglund, A., Wold, S., 1997. INLR, implicit non-linear latent variable regression. J. Chemotreics 11, 141–156.

Bogarad, L.D., Deem, M.W., 1999. A hierarchical approach to protein molecular evolution. Proc. Natl Acad. Sci. USA 96.

Bucht, G., Wikstrom, P., Hjalmarsson, K., 1999. Optimising the signal peptide for glycosyl phosphatidylinositol modifcation of human acetyhcholinesterase using mutation analysis and peptide-quantitative structure–activity relationships. Biochim. Biophys. Acta 1431, 471–482.

Byvatov, E., Schneider, G., 2004. SVM-based feature selection for characterization of focused compound collections. J. Chem. Inf. Comp. Sci. 44, 993–999.

Byvatov, E., Fechner, U., Sadowski, J., Schneider, G., 2003. Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. J. Chem. Inf. Comp. Sci. 43, 1882–1889.

Chen, R., 2001. Enzyme engineering: rational design versus directed evolution. Trends Biotech. 19.

Cho, S.J., Zheng, W., Tropsha, A., 1998. Rational combinatorial library design. 2. Rational design of targeted combinatorial peptide libraries using chemical similarity probe and the inverse QSAR approaches. J. Chem. Inf. Comp. Sci. 38, 259–268.

Daren, Z., 2001. QSPR studies of PCBs by the combination of genetic algorithms and PLS analysis. Comput. Chem. 25, 197–204.

Darius, F., Rojas, R., 1994. Simulated molecular evolution or computer-generated artifacts? Biophysical J. 67, 2120–2122.

De Genst, E., Areskoug, D., Decanniere, K., Muyldermans, S., Andersson, K., 2002. Kinetic and affinity predictions of a protein–protein interaction using multivariate experimental design. J. Biol. Chem. 277, 29897–29907.

de Jong, S., Wise, B., Ricker, L., 2001. Canonical partial least squares and continuum power regression. J. Chemotreics 15, 85–100.

Earl, D.J., Deem, M.W., 2004. Evolvability is a selectable trait. Proc. Natl Acad. Sci. USA 101.

Edwards, A., 2000. Perspectives on the genetical theory of natural selection. Genetics 154, 1419–1426.

Fisher, R.A., 1930. The Genetical Theory of Natural Selection. Oxford University Press, Oxford.

Fox, R., Roy, A., Govindarajan, S., Minshull, J., Gustafsson, C., Jones, J., Emig, R., 2003. Optimizing the search algorithm for protein engineering by directed evolution. Prot. Eng. 16, 589–597.

Goldberg, D.E., Lingle, R., 1985. Alleles, loci, and the traveling salesman problem. In: Greffenstette, J.J. (Ed.), Proceedings of the First ICGA, pp. 154–159.

Gustafsson, C., Govindarajan, S., Emig, R., 2001. Exploration of sequence space for protein engineering. J. Mol. Recog. 14, 308–314.

Hasegawa, K., Miyashita, Y., Funatsu, K., 1997. GA strategy for variable selciton in QSAR studies: GA-based PLS analysis of caclium channel antagonists. J. Chem. Inf. Comp. Sci. 37, 306–310.

Hasegawa, K., Kimura, T., Funatsu, K., 1999. GA strategy for variable selection in QSAR studies: application of GA-based region selection to a 3D-QSAR study of acetylcholinesterase inhibitors. J. Chem. Inf. Comp. Sci. 39, 112–120.

Hastie, T., Tibshirani, R., Friedman, J., 2003. The Elements of Statistical Learning. Springer, New York.

Kauffman, S., 1993. The Origins of Order. Oxford University Press, New York.

Kubinyi, H., 1995. Evolutionary variable selection in regression and PLS analyses. J. Chemotreics 10, 119–133.

Kubinyi, H., 1997a. QSAR and 3D QSAR in drug design Part1: methodology. Drug Disc Tech. 2, 457–467.

Kubinyi, H., 1997b. QSAR and 3D QSAR in drug design Part 2: applications and problems. Drug Disc Tech. 2, 538–546.

Kurtzman, A., Govindarajan, S., Vahle, K., Jones, J., Heinrichs, V., Patten, P., 2001. Advances in directed protein evolution by recursive genetic recombination: applications to therapeutic proteins. Curr. Opin. Biotech. 12, 361–370.

Leardi, R., Lupianez, G., 1998. Genetic algorithms applied to featrue selection in PLS regression: how and when to use them. Chemomet. Intell. Lab. Sys. 41, 195–207.

Lee, M., de Jong, S., Gade, G., Poulos, C., Goldsworthy, G., 2000. Mathematical modelling of insect neuropeptide potencies. Are quantitatively predictive models possible? Insect Biochem. Mol. Biol. 30, 899–907.

Lu, S.M., Lu, W., Qasim, M.A., Anderson, S., Apostol, I., Ardelt, W., Bigler, T., Chiang, Y.W., Cook, J., James, M.N., et al., 2001. Predicting the reactivity of proteins from their sequence alone: Kazal family of protein inhibitors of serine proteinases. Proc. Natl Acad. Sci. USA 98, 1410–1415.

Mee, R.P., Auton, T.R., Morgan, P.J., 1997. Design of active analogues of a 15-residue peptide using *D*-optimal design, QSAR and a combinatorial search algorithm. J Peptide Res. 49, 89–102.

Ness, J.E., Welch, M., Giver, L., Bueno, M., Cherry, J.R., Borchert, T.V., Stemmer, W.P., Minshull, J., 1999. DNA shuffling of subgenomic sequences of subtilisin. Nat. Biotech. 17, 893–896.

Ness, J.E., Kim, S., Gottman, A., Pak, R., Krebber, A., Borchert, T.V., Govindarajan, S., Mundorff, E.C., Minshull, J., 2002. Synthetic shuffling expands functional protein diversity by allowing amino acids to recombine independently. Nat. Biotech. 11, 11.

Ozdemir, M., 2002. Evolutionary computing for feature selection and predictive data mining. In: Engineering Science, Rensselaer Polytechnic Institute, Troy, New York, p. 285.

Pierce, A.N., Winfree, E., 2002. Protein design is NP-hard. Protein Eng. 15, 779–782.

Raman, V., Ravikumar, B., Rao, S.S., 1998. A simplified NP-complete MAXSAT problem. Inform. Process. Lett. 65, 1–6.

Rosipal, R., Trejo, L., 2001. Kernel partial least squares regression in reproducing kernel Hilbert space. J. Mach. Learning Res. 2, 97–123.

Schneider, G., Schuchhardt, J., Wrede, P., 1994. Artificial neural networks and simulated molecular evolution are potential tools for sequence-oriented protein design. Comp. Appl. Biosci. 10, 635–645.

Schneider, G., Schuchardt, J., Wrede, P., 1995a. Development of simple fitness landscapes for peptides by artificial neural filter systems. Biol. Cybern. 73, 245–254.

Schneider, G., Schuchardt, J., Wrede, P., 1995b. Peptide design in Machina: development of artificial mitochondrial protein precursor cleavage sites by simulated molecular evolution. Biophys. J. 68, 434–447.

Schneider, G., W., S., G., W., Muller, J., Nissen, E., Ronspeck, W., Wrede, P., Kunze, R., 1998. Peptide design by artificial neural networks and computer-based evolutionary search. Proc. Natl Acad. Sci. USA 95, 12179–12184.

Stemmer, W.P.C., 1994. Rapid evolution of a protein in vitro by DNA shuffling. Nature 370, 389–391.

van Regenmortel, M.H., 2000. Are there two distinct research strategies for developing biologically active molecuels: rational design and empirical selection? J. Mol. Recog. 13.

Wells, J.A., 1990. Additivity of mutational effects in proteins. Biochemistry 29, 8509–8517.

Wold, H., 1985. Partial least squares. In: Kotz, S., Johnson, N.L. (Eds.), Encyclopedia of Statistical Sciences. Wiley, New York, pp. 581–591.

Wrede, P., Landt, O., Klages, S., Fatemi, A., Hahn, U., Schneider, G., 1998. Peptide design aided by neural networks: biological activity of artificial signal peptidase I sleavage sites. Biochemistry 37, 3588–3593.

Wright, S., 1932. The roles of mutation, inbreeding, crossbreeding and selection in evolution. Proceedings of Sixth International Conference on Genetics, vol. 1, pp. 356–366.

Yu, X., Levitt, M., 2004. Simulating protein evolution in sequence and structure space. Curr. Opin. Struct. Biol. 14, 202–207.