Protein stability: a single recorded mutation aids in predicting the effects of other mutations in the same amino acid site

Gilad Wainreb¹, Lior Wolf²,*, Haim Ashkenazy¹, Yves Dehouck³ and Nir Ben-Tal¹, *

¹Department of Biochemistry and Molecular Biology, George S. Wise Faculty of Life Sciences, ²The Blavatnik School of Computer Science, Tel-Aviv University, Ramat Aviv 69978, Israel and ³Bioinformatique génomique et structurale, Université Libre de Bruxelles, Av Fr. Roosevelt 50, CP165/61, 1050 Bruxelles, Belgium

ABSTRACT

Motivation: Accurate prediction of protein stability is important for understanding the molecular underpinnings of diseases and for the design of new proteins. We introduce a novel approach for the prediction of changes in protein stability that arise from a single-site amino acid substitution; the approach uses available data on mutations occurring in the same position and in other positions. Our algorithm, named Pro-Maya (Protein Mutant stAbility Analyzer), combines a collaborative filtering baseline model, Random Forests regression and a diverse set of features. Pro-Maya predicts the stability free energy difference of mutant versus wild type, denoted as ΔΔG.

Results: We evaluated our algorithm extensively using cross-validation on two previously utilized datasets of single amino acid mutations and a (third) validation set. The results indicate that using known ΔΔG values of mutations at the query position improves the accuracy of ΔΔG predictions for other mutations in that position. The accuracy of our predictions in such cases significantly surpasses that of similar methods, achieving, e.g. a Pearson’s correlation coefficient of 0.78 and a root mean square error of 0.96 on the validation set. Because Pro-Maya uses a diverse set of features, including predictions using two other methods, it also performs slightly better than other methods in the absence of additional experimental data on the query positions.

Availability: Pro-Maya is freely available via web server at http://bental.tau.ac.il/ProMaya.

Contact: nirb@tauex.tau.ac.il; wolf@cs.tau.ac.il

Supplementary Information: Supplementary data are available at Bioinformatics online.

Received on April 30, 2011; revised on September 15, 2011; accepted on October 1, 2011

1 INTRODUCTION

Understanding the mechanisms by which mutations affect protein stability is important for characterizing disease mechanisms and for protein design (Bromberg and Rost, 2009). Hence, the energetics of mutants has been studied extensively through experimental and theoretical approaches.

The methods for predicting the change in a protein’s stability (ΔΔG) that results from a single amino acid mutation can be roughly classified according to the types of effective potentials they rely on: physical effective potentials (PEP), statistical effective potentials (SEP) and empirical effective potentials (EEP). Notably, none of these potentials explicitly take into consideration relevant known mutations at the query position. PEP-based methods use atomic-level representations to capture the underlying physical phenomena affecting protein stability, e.g. van der Waals interactions and dihedral (torsion) angle (Prevost et al., 1991; Seeliger and de Groot, 2010). These techniques are computationally demanding and not applicable to large datasets (Koffman et al., 2000). SEP-based methods are based on the inverse Boltzmann law, which states that probability densities and energies are closely related quantities. Hence, these methods use datasets of proteins of known structures to calculate conditional probabilities that certain residues or atoms will appear in different contexts. Most SEP-based methods use pairwise potentials (Bahar and Jernigan, 1997; Samudrala and Moult, 1998; Suppl., 1995), though some studies have employed higher order potentials; for example Vaisman et al. (1998) used a four-body potential. SEP-based methods are computationally efficient, more robust than PEP-based methods to low-resolution protein structure prediction and are suitable to include known and unknown physical effects (Lazaridis and Karplus, 2000). Methods in the third category (EEP-based) use experimental energy data to calibrate the weights of the energy function terms. The types of energy terms used can vary and might be SEP-, PEP-, physicochemically- or evolution-based methods (Bloom and Glassman, 2009; Gilis and Rooman, 1997; Masso and Vaisman, 2010; Shen et al., 2008). For example, PoPMuSiC-2.0 utilizes a neural network algorithm with SEP features that couple between the identity of the amino acid, secondary structure, accessibility and the spatial distance between amino acids (Dehouck et al., 2009). Conversely, FoldX’s (Guerois et al., 2002) energy function consists of PEP energy terms calibrated using a grid search method on experimental data. The recently developed Prethermut tool (Tian et al., 2010) incorporates the energy terms of FoldX and MODELLER (Salı and Blindell, 1993) into a Random Forests machine regression, and has reached impressive results. The use of a machine learning algorithm enables non-energy-like terms to be incorporated into the scoring function (Capriotti et al., 2005; Cheng et al., 2006; Montanucci et al., 2008). For example, both I-Mutant2.0 (Capriotti et al., 2005) and MUpro (Cheng et al., 2006) encode the identities of the wild-type (WT) and mutant amino acids in addition to the quantity

*To whom correspondence should be addressed.

© The Author(s) 2011. Published by Oxford University Press. This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
To this end, we developed an approach based on adaptation of the method of Cheng et al. (2006; Masso and Vaisman, 2010). However, as previously indicated by Cheng et al., this dataset is highly redundant and may lead to unreliable predictions. Recently, two large non-redundant datasets have been compiled by Potapov et al. (2009; Potapov-DB) and Dehouck et al. (2009; PoPMuSiC-DB), containing 2155 and 2648 mutations, respectively. The datasets comprise \( \Delta \Delta G \) measurements from thermal and denaturant denaturation experiments. To avoid redundancy, each dataset considers only one \( \Delta \Delta G \) value per mutant. In cases where numerous values have been obtained for a single mutant, Potapov et al. set the mutant’s \( \Delta \Delta G \) as the mean of the \( \Delta \Delta G \) measurements, whereas Dehouck et al. determine this value using a weighted average, giving higher weights to measurements taken in physiological conditions (pH close to 7, temperature close to 25°C and without additives). Thus, although the two datasets share 1405 common mutations, the \( \Delta \Delta G \) values assigned to some of these differ.

Preliminary examination of the PoPMuSiC-DB indicated that \( \Delta \Delta G \) values of mutations occurring at the same protein position tend to cluster (data not shown), i.e. \( \Delta \Delta G \) values of mutations in a given position are closer to each other, on average, than to \( \Delta \Delta G \) values in other positions. This suggests making explicit use of known \( \Delta \Delta G \) values to predict the effects of new mutations. To this end, we developed an approach based on adaptation of the baseline model of the BellKor collaborative filtering algorithm (CF) (Koren, 2008). To improve its accuracy, we combined the baseline model algorithm with a content-based model. The content-based model takes into account features of the mutation and its surrounding comprising various sequence, structure, SEP- and EEP-based features. We benchmarked our algorithm extensively by carrying out cross-validation on the PoPMuSiC-DB and Potapov-DB datasets and by running it on an additional validation set. Statistical analysis of the results indicates that Pro-Maya surpasses all the compared methods both when additional \( \Delta \Delta G \) values for the query position are available and when they are not.

### 2 METHODS

Our algorithm treats differently mutations at positions for which a \( \Delta \Delta G \) value for a different mutant is known (denoted MRPM, multi-replacement position mutation) and at positions with no additional known recorded mutations at the query position (denoted SRPM, single-replacement position mutation). Given a query mutation of MRPM, we follow the traditional machine learning scheme. Specifically, the query mutations are fed to a pre-calculated Random Forests regression model (Breiman, 2001) to predict the query’s \( \Delta \Delta G \) denoted as \( \Delta \Delta G^{\text{MRPM}} \) (described in Section 2.1). For MRPMs, as detailed in Figure 1, the predicted \( \Delta \Delta G^{\text{RF}} \) is utilized as an input to an additional prediction step using the integrated baseline- and content-based model, denoted as the collaborative filtering and content-based (CFCB) algorithm. The \( \Delta \Delta G^{\text{CFCB}} \) for the MRPMs is calculated using a Random Forests model trained on a dataset comprising the training dataset and the user reported \( \Delta \Delta G \) records of mutations at the query position. The input to the CFCB algorithm also includes a matrix representation of the known \( \Delta \Delta G \) (described in Section 2.2) and a set of the features. Note, that the \( \Delta \Delta G^{\text{CFCB}} \) in our algorithm is utilized both for the prediction of \( \Delta \Delta G \) mutations and as an input to the CFCB algorithm. The Pro-Maya algorithm predicts the \( \Delta \Delta G \) change of the mutant versus the wildtype protein (i.e. Mutant-WT). Thus, indicating both the magnitude of the stability change and its sign, i.e. whether the mutant is more or less stable than the WT.

#### 2.1 Calculation of \( \Delta \Delta G^{\text{RF}} \)

The \( \Delta \Delta G^{\text{RF}} \) is calculated using the Random Forests R implementation (Liaw and Wiener, 2002). The number of trees to grow was set to 600 since the addition of more trees did not change the performance. The number of random features to be searched at each tree node was the square root of the number of features, i.e. 6.

The Random Forests regression utilizes a total of 11 descriptors (F1–F11) with 30 dimensions, which can be roughly divided into sequence- and structure-based features as follows:

1. **Sequence-based features** The multiple sequence alignment (MSA) holds important information regarding the physicochemical preference of the position in the protein. From the MSA, we calculated the position specific scoring matrix (indicating the frequency of the amino acids in each MSA column) and used a physicochemical scale matrix to calculate the weighted average and SD of a physicochemical property. Given a mutation, we measured the degree to which its physicochemical properties deviated from the mean physicochemical preference at the query position. Each query mutation was evaluated according to the following physicochemical properties (F1–F3): hydrophobicity scale (Kessel and Ben-Tal, 2002), molecular weight and isoelectric point (Supplementary Table S1). Based on a related study (Waintreb et al., 2010), we added an additional descriptor measuring the sequence identity of the query protein to the lowest homolog bearing the mutant amino acid (denoted SIDCH (F5)). For example, mutation I48A in the *Hordeum vulgare* chymotrypsin (UniProtKB/Swiss-Prot ID: IC12_HORVU) (The UniProt Consortium, 2010) was shown by Jackson et al. (1993) to cause a major destabilization of the protein. Fifteen homologous proteins with sequence identities of 31–47% to IC12_HORVU feature the amino acid alanine in the corresponding position. Here we set the SIDCH of I48A to 47%. We also included an array of 20 features
The Gaussian regression cross-validation results of PoPMuSiC-2.0 were model's parameters are learned during the training procedure. The optimal \( \Delta \Delta G \) tries to model the relations between the known data points in matrix \( X \). In proteins for which an X-ray crystal structure existed, all structures determined through nuclear magnetic resonance (NMR) were disregarded.

Protein flexibility: to reflect the mobility of the protein's backbone at the mutated positions, we used the B-factors of the crystal structure (F8).

PEP-based features: we made use of \( \Delta \Delta G \) values predicted by the Prethermut tool (Tian et al., 2010) (F9). Prethermut uses a Random Forests machine learning algorithm and combines the energy terms of FoldX and MODELLER (Sali and Blundell, 1993). The energy terms are translated into units of SD from the average of the energy terms calculated over all possible mutations of the whole protein. To calculate the Prethermut prediction value, we conducted a Random Forests regression over the original energy terms (calculated using the Prethermut scripts). As suggested by Tian et al., the number of trees to grow was set to 650 and the number of random features to be searched at each tree node was the square root of the number of features, i.e. 8.

SEK-based features: the amino acid-specific torsion angle potential was calculated according to Parthiban et al. (2006) (F10). In addition, we utilized the PoPMuSiC-2.0 predicted \( \Delta \Delta G \) value, calculated using the energy terms in Dehours et al. (2009) and the Gaussian regression (Rasmussen and Williams, 2006) implementation of Weka (Frank et al., 2004) (F11). The Gaussian regression cross-validation results of PoPMuSiC-2.0 were comparable with the published results. The predicted PoPMuSiC-2.0 \( \Delta \Delta G \) values for mutations that were absent from the Potapov-DB were calculated using the PoPMuSiC-2.0 web server.

2.2 CFCB algorithm

CFCB recommender systems are used by many websites to generate personalized recommendations. For example, when a customer purchases an item on a retail website, such algorithms try to predict which other items the user would enjoy, on the basis of his/her past behavior and similarity to the behavior of other users. CF algorithms use only user-item data to make predictions. Conversely, content-based algorithms rely on the features of users and items for prediction.

In recent years, the main driving force behind the development of CF algorithms has been Netflix's million dollar prize for improving the performance of the site's recommendation system. Here, we chose to utilize a part of the CF solution of the winning group (named BellKor) (Koren, 2008). In order to improve the model’s performance, we extended it using a content-based-model to take into account biological information regarding the mutations.

In our CF scenario, there is a list of possible mutation outcomes (MU, i.e. all possible amino acids), a list of mutation positions (defined by the protein and the residue number) and the experimental \( \Delta \Delta G \) values for some of the mutations at these positions. The data can be stored in a sparse matrix of size \( n \times m \), where \( n \) denotes the number of MUs and \( m \) denotes the number of positions. Each cell \( x_{ui} \) of the matrix \( r \) indicates the \( \Delta \Delta G \) of a mutation to amino acid \( u \) at position \( i \) (see, for example Supplementary Fig. S1A).

For clarity, special indexing letters \( a \) and \( i \) are reserved for distinguishing MUs and positions, respectively.

2.2.1 The prediction models

The BellKor CF algorithm (Koren, 2008) tries to model the relations between the known data points in matrix \( r \). The model’s parameters are learned during the training procedure. The optimal model is later utilized to predict \( \Delta \Delta G \) values of unknown mutations in positions with known \( \Delta \Delta G \) values for other mutations.

The BellKor model integrates three types of approaches to CF: a baseline model, a neighborhood model and the latent factor model. Our CFCB algorithm integrates the BellKor baseline estimator model with a content-based model. We also implemented the neighborhood and latent factor models, but according to our analysis their incorporation into the model does not improve the prediction accuracy significantly, although it might in certain cases (Supplementary Material). A schematic representation of all models can be seen in Supplementary Figure S1.

2.2.2 The baseline estimator model

Different MUs and positions have different \( \Delta \Delta G \) tendencies. For example, the \( \Delta \Delta G \) of a mutation at a buried position in a protein is usually larger than that of the same mutation at an exposed position. Similarly, we would expect that in most cases the consequences of mutation to proline would be more severe than a mutation to alanine. Hence, each position and MU is ascribed unique baseline estimators, denoted \( b_i \) and \( b_u \), respectively. Thus, for every \( x_{ui} \) we define a baseline estimator \( b_{ui} = b_i + b_u \), with \( b_i \) denoting the overall average of all \( \Delta \Delta G \) in \( r \). The variables \( b_i \) and \( b_u \) are learned during the training stage of the algorithm (described in Section 2.2.2).

2.2.3 The content-based model

The baseline model does not use any explicit description of the mutation. In order to describe the biological aspects of the mutation, we use a linear regression solution (with no intercept) [Equation (1)] with a subset of the features (described in Section 2.2.2): solvent accessibility, torsional statistical field, Prethermut MODELLER-based features, the SIFT predicted compatibility of the mutated amino acid to the query position (Ng and Henikoff, 2003) and \( \Delta \Delta G \) predictions by PoPMuSiC-2.0. In addition, we also use as a feature the SIFT score [Equation (2)]. In Equation (1), \( x_{ui} \) is the set of \( d \) features \( x_{ui,1}, x_{ui,2}, \ldots, x_{ui,d} \), describing the mutation whose \( \Delta \Delta G \) indices in matrix \( r \) are \( u \) and \( i \). \( F \) denotes a set of \( d \) descriptor coefficients. As is often done in linear regression, each descriptor is normalized across all positions and MUs so that its average is zero and the SD is 1. \( F \) is learned during the training stage described in Section 2.2.2 using the stochastic gradient descent.

2.2.4 The integrated model

The integrated model [Equation (2)] combines the baseline- and content-based models. \( y_{ui} \) denotes the predicted \( \Delta \Delta G \). In Equation (1), \( x_{ui} \) is the set of \( d \) features \( x_{ui,1}, x_{ui,2}, \ldots, x_{ui,d} \), describing the mutation whose \( \Delta \Delta G \) indices in matrix \( r \) are \( u \) and \( i \). \( F \) denotes a set of \( d \) descriptor coefficients.

\[ y_{ui} = b_{ui} + \sum_{g=0}^{d} x_{ui,g} F_g \]  

2.2.5 The CFCB training and prediction procedures

As in any machine learning algorithm, the aim of the training procedure is to obtain parameters that fit the model to the observed data best. Unconventionally, the CFCB model is trained for every server query in order to identify the parameters of the newly added user-reported mutations, e.g. the baseline estimator of the newly added position. The model with the optimized set of parameters presumably describes best the relations between the known \( \Delta \Delta G \)s in matrix \( r \) and is used to predict the unknown MRMP queries.

The training procedure is performed using a stochastic gradient descent algorithm that attempts to minimize the associated regularized squared error function [Equation (3)] and determines the following parameters: \( b_i, b_u \) and \( F \). Thus, starting with random values for the parameters, it randomly loops over all the known \( \Delta \Delta G \)s in \( r \) (which is composed of all known mutations across all proteins in the training dataset) and modify the parameters by moving in the opposite direction of the gradient [Equation (4)]. The descent iterations continue until the difference between the root mean square error between the predicted \( \Delta \Delta G \)s and the known \( \Delta \Delta G \)s ([predicted \( \Delta \Delta G \)s-observed \( \Delta \Delta G \)s] of the current iteration and the previous iteration is smaller than \( \epsilon \). During the training, we used the following meta parameters: (learning rate) \( \eta \) = 0.02, (regularization factor) \( \lambda \) = 0.0025 and \( \epsilon = 0.00001 \).

\[ \min_{b_i, b_u} \sum_{u \in \text{MUs}, \, i \in \text{Positions}} (x_{ui} - y_{ui})^2 + \lambda (b_i^2 + b_u^2 + \sum_{g=1}^{d} F_g^2) \]
The randomly selected folds were maintained throughout the prediction To train and assess our algorithm, we utilized two publicly available /Delta1/Delta1 scheme, i.e. the calculation of the Prethermut, PoPMuSiC-2.0, (PDB) structures (NMR and C datasets: the PoPMuSiC-DB with 2648 mutations in 137 proteins and the DB for PoPMuSiC-2.0 (Dehouck et al., 2009). Both datasets have been previously used as corresponding SWISS-PROT entries (Jain et al., 2009). For each iteration, we randomly selected 60% of the cross-validation performance measures, we used a bootstrap procedure with 1000 iterations. To assess how the number of mutations with known G values in the query position affect the prediction accuracy, we compared the performance models of Prethermut and PoPMuSiC-2.0 had to be retrained with the modified training set. Since for the Potapov-DB we do not have the PoPMuSiC-2.0 statistical force field components (needed for the retraining), with the modified training set. Since for the Potapov-DB we do not have /Delta1/Delta1 mutations excluded from the PoPMuSiC-DB. This validation set has been previously used by Dehouck et al. (2009) on the Potapov-DB (Supplementary Table S4). At each iteration of LOO-all, LOO-neglected and LOO-unseen, the results for the SRPM set indicate the performance for mutations at positions that are absent from the training set. For this mutation subset, our prediction scheme does not involve the CFCB algorithm and relies solely on the Random Forests regression and on the quality of the features. Here, our prediction scheme consists of the boosting process our PCC showed an average (minor) improvement of 0.02±0.1 over the PCC of Prethermut, the best of the other methods.

Interestingly, each method achieved a lower RMSE for the PoPMuSiC-DB set than for the Potapov-DB set. This trend is also seen in the cross-validation results of the 1405 mutations shared by the two datasets (data not shown). Possible explanations are suggested in the Section 4 below.

Pro-Maya’s performance was also evaluated on a validation set of mutations excluded from the PoPMuSiC-DB. This validation set has been previously used by Dehouck et al. to benchmark PoPMuSiC-2.0, Dmutant (Zhou and Zhou, 2002), Auto-MUTE (Masso and Vaisman, 2010), FoldX (Guerios et al., 2002), CUPSAT (Parthiban et al., 2006), Extras (Yin et al., 2007) and I-Mutant2.0 (Capriotti et al., 2005). Both the PCC and RMSE values indicate that Pro-Maya performs better than these aforementioned methods (Table 2; Supplementary Table S5). For the entire validation set and for its SRPM and MRPM subsets. As can be seen in Table 2, Pro-Maya’s PCC on the entire validation set reaches a value of 0.79, constituting an improvement of 0.07 and of 0.1 over the PCCs obtained by Prethermut and by PoPMuSiC-2.0, respectively. To estimate how well Pro-Maya performs on query mutations at proteins that are not homologous to any of the proteins in the training set, we compared the performance of the LOO-unseen with the performance of the LOO-all (Supplementary Table S4). Interestingly, although the performance of the \( \Delta G_{\text{RF}} \) of the

\[ e_i = e_j = y_i \]
\[ b_i = b_j + \gamma \ (e_i - \lambda \cdot b_j) \]
\[ F = F + \gamma \ (e_i - \lambda \cdot b_j) \]

\[ F = F + \gamma \ (e_i - \lambda \cdot b_j) \]
As can be seen, Pro-Maya performs better on the entire validation set and subsets. The FAQ section also contains a detailed description of Pro-Maya’s performance for the Random Forests and collaborative filtering results on the SRPM and MRPM subsets. Thus, we cannot estimate Pro-Maya’s performance on these classes, although there is no reason to believe that the performance over them will differ significantly from the rest.

3.2 How do the number and type of mutations with known ΔAG values in the query position affect the prediction accuracy?

Figure 2 shows that Pro-Maya’s prediction accuracy increases significantly with the addition of a single or two known mutations at the query position, and that the accuracy does not improve further significantly with the addition of more than two records.

Intuitively, we might expect that the prediction accuracy of the CFCB algorithm should be correlated with the level of similarity between the physicochemical properties of the query and recorded mutations. To examine this hypothesis, for each of the mutations predicted by the CFCB algorithm in the PoPMuSiC-DB, we measured the shortest physicochemical distance [using the Miyata matrix (Miyata et al., 1979)] from the query mutation amino acid to any of the recorded mutations. For example, given a query mutation to isoleucine at residue 29 in the apomyoglobin protein (PDB id: 1bcv chain A), we measured the shortest Miyata distance to 0.29, which is the Miyata distance between isoleucine and methionine. Here, we set the shortest Miyata distance to 0.29, which is the Miyata distance between isoleucine and methionine. The correlation between the Miyata distances of all query mutations with known ΔAG values at the query position.

An analysis of Pro-Maya’s LOO-unseen versus the SCOP classification (Supplementary Table S6) of the proteins shows that Pro-Maya performs similarly on the All α, All β, a+β and α/β SCOP classes with a PCC ranging from 0.59 to 0.64 for the SRPM and 0.8–0.83 for the MRPM. The PoPMuSiC-DB includes low number of mutations from the Coiled-coil, Multi-domain and Small proteins SCOP classes. Thus, we cannot estimate Pro-Maya’s performance on these classes, although there is no reason to believe that the performance over them will differ significantly from the rest.

Pro-Maya performs better on the entire validation set and subsets. The FAQ section also contains a detailed description of Pro-Maya’s performance for the Random Forests and collaborative filtering results on the SRPM and MRPM subsets. Thus, we cannot estimate Pro-Maya’s performance on these classes, although there is no reason to believe that the performance over them will differ significantly from the rest.

3.2 How do the number and type of mutations with known ΔAG values in the query position affect the prediction accuracy?

Figure 2 shows that Pro-Maya’s prediction accuracy increases significantly with the addition of a single or two known mutations at the query position, and that the accuracy does not improve further significantly with the addition of more than two records.

Intuitively, we might expect that the prediction accuracy of the CFCB algorithm should be correlated with the level of similarity between the physicochemical properties of the query and recorded mutations. To examine this hypothesis, for each of the mutations predicted by the CFCB algorithm in the PoPMuSiC-DB, we measured the shortest physicochemical distance [using the Miyata matrix (Miyata et al., 1979)] from the query mutation amino acid to any of the recorded mutations. For example, given a query mutation to isoleucine at residue 29 in the apomyoglobin protein (PDB id: 1bcv chain A), we measured the shortest Miyata distance to 0.29, which is the Miyata distance between isoleucine and methionine. Here, we set the shortest Miyata distance to 0.29, which is the Miyata distance between isoleucine and methionine. The correlation between the Miyata distances of all query mutations with known ΔAG values at the query position.

An analysis of Pro-Maya’s LOO-unseen versus the SCOP classification (Supplementary Table S6) of the proteins shows that
We tested Pro-Maya extensively using cross-validation on two datasets and an additional validation dataset, and found that it outperformed current methods for the prediction of mutation stability. Our results demonstrate that the availability of as few as one or two records in the query position improve the prediction accuracy of $\Delta\Delta G$ values of additional mutations in that position. This improvement is independent of the amino acid identity of these records and of the sequence identity of the query protein to the training set. Thus, a systematic alanine-scanning mutagenesis of all the amino acids in a protein could greatly increase Pro-Maya’s prediction accuracy for any mutation in the protein.

The performance of our Random Forests prediction scheme on the SRPM subset is slightly better than that of the other methods we investigated. We attribute the improvement to the use of an inhomogeneous feature set comprising PEP-, SEP- and evolution-based features, including predictions by the Prethermut (Tian et al., 2010) and PoPMuSiC-2.0 (Dehouck et al., 2009) tools. Previous prediction methods, in contrast, have been based on features of a single type (e.g. only PEP).

Pro-Maya’s RMSEs for mutations in the PoPMuSiC-DB set are consistently lower than those for the Potapov-DB set. This is presumably because of the different procedures used for compilation of each dataset. PoPMuSiC-DB’s compilation procedure used a weighted average of the identical mutations occurring in different conditions to calculate the $\Delta\Delta G$ values that are most likely to occur at physiological conditions. Whereas, the Potapov-DB compilation procedure gives equal weight to the various conditions at which $\Delta\Delta G$ was measured. Our prediction scheme does not take into account the conditions at which $\Delta\Delta G$ was measured. Thus, it assumes that all measurements were taken under the same conditions. Therefore, the PoPMuSiC-DB mutation set, which is characterized by more homogenous experimental conditions, is presumably more suitable for our prediction scheme, as indicated by the low RMSE value. To achieve more accurate predictions, we trained the Pro-Maya web server using the PoPMuSiC-DB set. Thus, the server is best suited for predicting mutations at physiological conditions.

Pro-Maya’s improved accuracy is facilitated by the use of a baseline estimator that utilizes known $\Delta\Delta G$ records to determine a position-specific baseline $\Delta\Delta G$ (b) model. The underlying assumption of Pro-Maya is that the $\Delta\Delta G$ of a mutation is strongly dependent on properties that are inherent to the amino acid position in the protein (e.g. solvent accessibility, amino acid identity, interaction with the environment and secondary structure). Thus, on average all mutations at the same position are expected to have similar $\Delta\Delta G$ values. Therefore, the position baseline $\Delta\Delta G$ which presumably reflects the inherent properties of the position can roughly model the query mutation. To fully model a mutation, Pro-Maya also uses a content-based model and a MU-specific $\Delta\Delta G$ baseline-based model. These models describe the mutation outcome attributes (e.g. physicochemical properties) and predict the $\Delta\Delta G$ shift from the position baseline. Nevertheless, it is expected that mutations with an irregular $\Delta\Delta G$ that differs much from the position $\Delta\Delta G$ baseline would be harder to predict.

By design, Pro-Maya is not very suitable as a classifier of whether a mutation would stabilize or destabilize the protein; a classifier should be trained to this end. CF algorithms have been developed mainly for online electronic commerce applications and are particularly useful for exploiting large datasets very rapidly. To the best of our knowledge, their use in biology is quite scarce (Efteh et al., 2006). The success of the CF algorithm in this study and the capability of the neighborhood- and latent factor-based models to identify biological properties (discussed in the Supplementary Material) suggest that the CF approach could be applied to additional problems in biology. Examples include the identification of deleterious mutations in single nucleotide polymorphism data, the detection of true protein–protein interactions in noisy yeast two-hybrid and massspectrometry data, as well as the prediction of ligand and drug molecules that could bind target proteins. Our CF algorithm and its integration with the neighborhood- and latent factor-based models can be readily adapted to these problems.

ACKNOWLEDGEMENTS

We would like to thank Daphna Meroz for helpful discussions and Amit Ashkenazi for her assistance in the web server construction. Funding: This study was supported by a grant from the German-Israeli Project Cooperation (DIP). Y.D. is Postdoctoral Researcher at the Belgian F.R.S.-FNRS. I.A. is a fellow of the Edmond J. Safra Program in Bioinformatics at Tel-Aviv University.

Conflict of Interest: none declared.

REFERENCES


Bronberg, Y. and Rost, B. (2009) Correlating protein function and stability through the analysis of single amino acid substitutions. BMC Bioinformatics, 10 (Suppl. 8), S8.


Jan, E. et al. (2009) Infrastructure for the life sciences: design and implementation of the UniProt website. BMC Bioinformatics, 10, 136.


