Learning Organizations of Protein Energy Landscapes: An Application on Decoy Selection in Template-Free Protein Structure Prediction

Nasrin Akhter
Liban Hassan
Zahra Rajabi
Daniel Barbará
Amarda Shehu

Abstract The protein energy landscape, which lifts the protein structure space by associating energies with structures, has been useful in improving our understanding of the relationship between structure, dynamics, and function. Currently, however, it is challenging to automatically extract and utilize the underlying organization of an energy landscape to the link structural states it houses to biological activity. In this chapter, we first report on two computational approaches that extract such an organization, one that ignores energies and operates directly in the structure space, and another that operates on the energy landscape associated with the structure space. We then describe two complementary approaches, one based on unsupervised learning and another based on supervised learning. Both approaches utilize the extracted organization to address the problem of decoy selection in template-free protein structure prediction. The presented results make the case that learning organizations of protein energy landscapes advances our ability to link structures to biological activity.

Introduction

The tertiary structures in which the sequence of amino acids that constitute a protein molecule folds in three-dimensional space determine to a great extent the biological activities of a protein; the geometric and physico-chemical complementary of

Nasrin Akhter
Department of Computer Science, George Mason University.

Liban Hassan
Department of Computer Science, George Mason University.

Zahra Rajabi
Department of Computer Science, George Mason University.

Daniel Barbará
Department of Computer Science, George Mason University.

Amarda Shehu
Department of Computer Science, George Mason University, 4400 University Dr., MSN 4A5, Fairfax, VA, 22030, USA. e-mail: amarda@gmu.edu

molecular structures drives molecular docking events, making the ability of a protein to assume specific structures under physiological conditions essential to regulating interactions with molecular partners in the cell [4].

Algorithmic and hardware advances have resulted in an explosion of protein tertiary structure data. It is now possible to generate thousands of tertiary structures for a (target) protein of interest, even when provided only with its amino-acid sequence, in a matter of days, leveraging embarrassing parallelism in supercomputer architectures [26]. Some of the most visible computational methods able to do so are template-free protein structure prediction methods, such as Rosetta [22], Quark [44], and others [34, 11]. These methods operate under the umbrella of stochastic optimization, seeking local minima of some selected energy function that correspond to possibly biologically-active/native structures in the vast structure space. Effectively, these methods probe the structure space (and the associated energy landscape that lifts the structure space with the additional dimension of energy) one structure at a time and yield diverse tertiary structures of a target protein.

The availability of diverse tertiary structures populating the structure space of a target protein presents an opportunity to analyze the probed space and possibly reveal, in an automated fashion, native structures. Though traditionally much of the literature on protein modeling (and template-free structure prediction, in particular) refers to one native structure, there is a growing consensus that the multiplicity of native structures cannot be ignored [39, 26, 40, 33]. We now know that proteins, like many other biological macromolecules, are intrinsically dynamic and undergo structural rearrangements to accommodate different molecular partners and so regulate their biological activities in the cell [4]. Lifting the structure space to additionally associate an energy with each computed structure reveals an energy landscape that is often rich in broad and deep wells/basins [33]; such basins correspond to thermodynamically-stable structural states that are similarly long-lived and harnessed by a protein to participate in diverse cellular processes [5].

In principle, analysis of a computationally-probed protein structure space or its energy landscape ought to reveal the structural states relevant for the possibly diverse menu of biological activities [32]. Doing so remains challenging. Revealing such states necessitates extracting the underlying organization of the structure space and/or the associated energy landscape. In addition, after such an organization is revealed, methods are needed to recognize in it states that are possibly active (or native). Such methods need to operate in the presence of approximations and errors that manifest themselves in possibly sparse and non-uniform distributions of structures sampled from the structure space guided by an energy function that may be inherently inaccurate and steer away from actual native structures.

Though the above task presents many challenges, in this chapter we relate some first steps to tackling it. We report on two computational approaches that extract organizations of protein structure spaces or associated energy landscapes. We then describe two complementary approaches, one based on unsupervised learning and another based on supervised learning, that demonstrate the utility of the extracted organizations in the context of the problem of decoy selection in template-free protein structure prediction. The presented results make the case that learning organizations

of protein energy landscapes advances our ability to link structures to biological activity and that whole data-driven approaches warrant further investigation.

The rest of this chapter is organized as follows. In Section 1.1 we provide some more formalisms, defining the notion of an energy landscape and properly linking it to the structure space, and then summarize pertinent research in the context of decoy selection in template-free protein structure prediction. The approaches proposed for extracting and utilizing the underlying organization of a protein structure or its associated energy landscape are described in Section 2, followed by evaluation in Section 3. The chapter concludes with some thoughts regarding the current shortcomings of this line of work and possible future remedies in Section 4.

Related Work

Energy Landscape

The concept of a landscape (or lifted space) is general, appears in many scientific disciplines [12, 6, 38, 36], and can be defined as follows: a landscape consists of a set X of points, a neighborhood $\mathcal{N}(X)$ defined on X, a distance metric on X, and a function $f : X \to \mathbb{R}_{\geq 0}$ that assigns a score (also known as a height or energy, depending on the application domain) to every point $x \in X$. Every point in X is assigned a neighborhood by the neighborhood function N. In the context of decoy selection, points $x \in X$ are (decoy) tertiary structures, and the function f that scores the decoys is most often referred to as an energy function.

Protein energy landscapes (as probed via designed functions f) are multi-dimensional and multimodal, containing many components or elements, such as basins (or wells) and basin-separating barriers. A local (energy) minimum in the landscape is surrounded by a basin of attraction (becoming its focal minimum), which is the set of points on the landscape from which steepest descent/ascent converges to that focal optimum.

The decoy generation phase in template-free structure prediction methods samples points from an unknown, underlying structure space X, guided by a selected energy function f. The decoys are evaluated via f, and so the output is a set $\{x, f(x)\}$ of evaluated decoys $x \in X$. In light of this, a computational approach can seek to elucidate the organization of the sampled X (the structure space) or the sampled energy landscape available as $\{x, f(x)\}$.

Decoy Selection

Decoy selection refers to the problem of identifying one or more native structures from the set of (decoy) structures computed by a template-free protein structure prediction method. The problem remains open [20, 21], garnering its own evaluation category in the Critical Assessment of protein Structure Prediction (CASP) series of community wide experiments [29]. CASP assesses progress in protein structure prediction and is structured as a competition, where participants submit blind predictions (what they assess to be native or near-native structures) of selected target proteins. The submitted structures are evaluated by independent assessors after one or more native structures per target are made available by recruited wet laboratories. The latest CASP assessment [30] shows that decoy selection remains a bottleneck.

Decoy selection methods can be grouped into single-model, bag-of-models, quasi-single, and machine learning (ML) methods. Single-model methods evaluate each generated/computed decoy via a physics- or a knowledge-based function to associate an energy/score with each decoy [42, 23]. These methods make use of a score threshold to filter out decoys that are putatively not near (similar to) the native structure ("the true answer"). Relying on a threshold, however, is shown to either miss native structures or allow the inclusion of too many non-native ones [29, 14, 43]. In response, a popular approach is to ignore energy altogether and cluster decoys by structural similarity [24, 47], offering one or more of the top highest-populated clusters as prediction.

Cluster-based decoy selection methods implement the bag-of-models approach and leverage the premise that decoys are randomly distributed around the "true answer," which a consensus-seeking method should reveal [28]. This is not a valid assumption, as template-free protein structure prediction methods conduct biased sampling of the structures space (to handle its size and dimensionality), guided by an energy function that often contains inherent biases manifest in invalidating of entire regions of the structure space [37]. Indeed, an active thrust of research in protein structure modeling concerns the design of more accurate energy functions [15]. Currently, cluster-based methods fail to pick up good decoys when applied to hard targets where most decoys are very different from the known native structure(s), are highly dissimilar and sparsely sampled [29].

Quasi single-model methods combine strategies from single-model and bag-of-models methods. They first select some high-quality structures to which then the rest of the decoys are compared [18]. These methods are shown to perform better than single-model and consensus-seeking methods [16, 35]. Finally, a recent but promising line of research leverages ML models, such as Support Vector Machines (SVM) [7], Neural Network [31], and Random Forest [25]. For instance, work in [9] extracts 94 features from a protein structure to build an SVM model for selecting native decoys. Ensemble learning has been shown to outperform SVM learning [27], with or without statistical features derived off decoys.

Though in their infancy, ML methods warrant further evaluation. Some of the efforts related in this chapter can be categorized as building ML models under the umbrella of unsupervised or supervised learning. However, as Section 2 describes, these models do not operate over single decoys; instead they operate over extracted subsets of decoys automatically extracted from the structure space or the energy landscape of a protein of interest.

Methods

Both computational approaches that we propose start by embedding computed tertiary structures in a graph, which is then subjected to techniques that utilize the structure of the graph to partition it into non-overlapping groups of structures. These groups are then subjected either to unsupervised or supervised learning techniques to extract from them the ones most likely to contain biologically-active/native structures, evaluating the selection in the context of decoy selection. Due to the evaluation setting, from now on, we will refer to the tertiary structures as decoys, even though

the methods and techniques described here can extend beyond the specific context of decoy selection. We now proceed to describe each of the methods.

Embedding of Tertiary Structures in a Nearest-neighbor Graph

Let us refer to the set of computed tertiary structures of a protein (in our context, decoys), as Ω. Envision that this set consists of points sampled from a high-dimensional (protein) structure space. To encode the sample proximity (and thus, similarity) in this space, the set can be embedded in a nearest-neighbor graph (nngraph) $G = (V, E)$. The vertex set V is populated with Ω (each decoy becoming a vertex). The edge set E is populated by inferring a local neighborhood structure over each decoy.

The proximity of two decoys is measured via root-mean-squared-deviation (RMSD), after each decoy is superimposed over some reference decoy (arbitrarily, chosen to be the first one, for instance, in Ω); the reason for the superimposition is so as to minimize differences due to rigid-body motions (translations and rotations in SE(3)) [19]. Superimposing all decoys to a reference decoy a priori to the pairwise RMSD computation, rather than conducting the superimposition over every pair of decoys under comparison, saves computational time (from linear to quadratic). Using RMSD to compute the distance between two decoys, a vertex $u \in V$ is connected to vertices $v \in V$ if $d(u,v) \leq \varepsilon$, where ε is a user-defined parameter. Proximity query data structures (such as kd-trees, VP-trees, C-trees, and others) can be used to efficiently extract the nearest neighbors of a vertex (rather than rely on brute-force, all–pair comparisons).

The resulting nngraph may be disconnected, if ε is small and the decoys are the result of a sparse, non-uniform sampling of the structure space. This can be in part remedied by initializing ε to an initial value (ε_0) and then increasing it by $\delta\varepsilon$ over a maximum of n_ε iterations, all the while controlling the density of the resulting nngraph via a specified maximum number (k) of nearest neighbors per vertex. However, the quality of the sampling dictates in large part the quality of the embedding and the rest of the analysis methods proposed here. In the context of decoy selection, embarrassing parallelism of software such as Rosetta ensures a large number of decoys, though their quality varies based on the difficulty of the protein target, as we relate in Section 3.

Extracting Organization via Nngraph Embeddings

The resulting nngraph can now investigated for its organization via two different approaches.

Identifying Communities in the Graph-embedded Structure Space

The first approach does not consider the energy/score of each decoy embedded in the nngraph; that is, the nngraph is seen as a discrete representation of the sampled structure space. Under this treatment, community detection methods, borrowed from the domain of complex network analysis (such as social networks) can be readily utilized to detect communities. These methods effectively conduct clustering of the vertices in the nngraph, leveraging the distribution of edges over vertices in a community versus those outside. There are many community detection methods, but

we select 6 representative, state-of-the-art ones, which we investigate for their ability to expose the underlying organization of the protein structure space.

Specifically, we utilize the Leading Eigenvector (LE) method, the Walktrap (WT) method, the Label Propagation (LP) method, the Louvain (Lo) method, the InfoMap (IM) method, and the Greedy Modularity Maximization (GMM) method. In summary, the LE method aims to maximize modularity over possible partitions of a graph by utilizing the eigenspectrum of the modularity matrix. In the WT method, random walks are used to capture similarities between vertices (or sets of vertices), and agglomerative approach is used to hierarchically combine two adjacent communities at a time. In the LP method, each vertex is given a unique label (thus starting with $|V|$ communities, and labels are iteratively propagated through the network, reaching a consensus on a unique label. The Lo method is based on modularity maximization and assigns vertices to communities based on modularity gain. In the IM method, communities are identified using random walks that analyze the information flow. In the GMM method, vertices are repeatedly joined together into two communities, whose modularity produces the largest increase, producing a dendrogram that encodes good partitions based on high values of modularity. In the interest of brevity, we do not describe these methods in detail, but the interested reader can find a comprehensive review in Ref. [46].

Identifying Basins in the Graph-embedded Energy Landscape

While the above approach only considers proximity of decoys in the structure space to detect an underlying organization, the graph embedding can be lifted in the energy landscape, additionally considering the energy/score of each decoy, as follows. Each vertex additionally contains the score of the corresponding decoy. Vertices that constitute local minima in the energy landscape are identified first. A vertex $u \in V$ is a local minimum if $\forall v \in V\, f(u) \leq f(v)$, where $v \in N(u)$ ($N(u)$ denotes the 1-neighborhood of u). The remaining vertices are then grouped into *basins* in the landscape as follows. Each vertex u is associated a negative gradient estimated by selecting the edge (u,v) that maximizes the ratio $[f(u)-f(v)]/d(u,v)$, where $d(u,v)$ is the RMSD between the two corresponding decoys. From each vertex u that is not a local minimum, the negative gradient is then followed iteratively (following the edge that maximizes the above ratio) until a local minimum is reached. Vertices that reach the same local minimum are assigned to the basin associated with that minimum, which is considered the *focal* minimum of that basin. This approach of leveraging the nngraph to identify basins in the landscape is first described in [8] as part of the Structural Bioinformatics Library (SBL) suite of structure and structure ensemble analysis algorithms.

Unsupervised Learning for Decoy Selection

Let us generally refer to the communities or basins identified as above as groups G. Different measurements can be associated with a group. Two such measurements are size and energy. Size refers to the number of decoys/vertices in a group. The energy of a group can be defined in two different ways, either as the minimum or average energy/score over the decoys in the group.

Given any of these two measurements, identified groups can then be ranked. For instance, considering only size, the identified groups can be ranked in a descending sorted order, and an automatic unsupervised learning strategy can extract the top c groups and offer them as "prediction" for where native and near-native structures reside. We refer to this strategy as UL-S. Another ranking strategy can additionally consider energy; considering energy alone has long been proven ineffective, as summarized in Section 1. So, instead, in UL-S+E, we consider the top $l > c$ largest groups (in the sorted order), and then re-sort these l groups from lowest to highest energy, selecting the top c of them for prediction. Two more strategies can be devised, based on Pareto optimality in multi-objective optimization, recognizing the unclear interaction between the size and energy of a group.

Suppose we want to select optimally considering various conflicting criteria/objectives. In this scenario, Pareto-optimal solutions are sought, as a single solution minimizing all conflicting objectives simultaneously is typically non-existent; a Pareto-optimal solution cannot be improved in one objective without sacrificing the quality of at least one other objective; i.e., a solution S_1 *Pareto-dominates* another solution S_2 if the following two conditions are satisfied: (1) For all optimization objectives i, $score_i(S_1) \geq score_i(S_2)$; (2) For at least one optimization objective i, $score_i(S_1) > score_i(S_2)$.

One can now associate two additional quantities, Pareto Rank (PR) and Pareto Count (PC) with each group G. PR(G) is the number of groups that dominate G, and PC(G) is the number of communities that G dominates. So, two additional, Pareto-based strategies are proposed. In UL-PR, the identified groups are sorted by low to high PR values, and the top c communities in this sorted order are selected and analyzed. In UL-PR+PC, PC is additionally considered. Communities with the same PR value are sorted from high to low PCs, and the top c groups in this resulting sorted order are selected.

We note that a

Supervised Learning for Decoy Selection

We consider both the classification and regression setting. The training and testing datasets consist of groups of decoys. Two settings are considered, when the groups are all communities identified as above, or all basins, also described above. In the classification setting, a group is labeled as either pure (class 1) or impure (class 0), given a user-defined threshold of purity (with purity measured over a group as described above). In the regression setting, the purity of a group is the actual output variable. In both settings, a group is reduced to a vector of 4 attributes, size, energy, PR, and PC, described above. In the case where models learn over basins, a fifth attribute is considered, basin stability, which relates to the measurement of whether a basin is shallow or deep relative to its nearest saddle [8]. Attributes are normalized to take values in the $[0, 1]$ range.

Most models struggle on classifying data with an imbalanced class distribution; this is the case on decoy datasets, where the number of what are deemed to be near-native decoys (those with lRMSD from a given native structure is within dist_thresh) is far outnumbered by the number of decoys deemed to be non-native.

Since supervised learning models are generally designed to maximize overall prediction accuracy, these methods do not consider the imbalance data distribution. Rather, the majority class data are more focused and the minority/rare class data are ignored and often misclassified. In our evaluation setting on decoy datasets, the number of what are deemed to be near-native decoys (those with lRMSD from a given native structure is within dist_thresh) can be far outnumbered by the number of decoys deemed to be non-native. Three common approaches to address an imbalance class distribution are either to (1) modify the model being utilized, (2) preprocess the data to compensate for the imbalance in the data distribution (notable, not all models are amenable to this change), or (3) select features carefully so as to be less affected by the imbalance class distribution. We adopt the algorithmic approach of using ensemble learning techniques and the data preprocessing approach of data under-sampling to address the imbalanced dataset issue.

We consider three representative ensemble learning methods [48] in the classification setting: Extreme Gradient Boosting (XGBoost), Balanced Bagging (BB), and Remove Tomeklinks (RT). XGBoost, is a fast, scalable implementation of the Gradient Boosting Machine (GBM), itself a boosting-based ensemble approach that adopts a gradient-descent based formulation [13]. Instead of re-weighting the input samples in each iteration, GBM adds a weak learner to minimize an arbitrary differentiable loss function. GBM has been widely used in many machine learning applications with considerable success. XGBoost addresses the overfitting problem in GBM [10] and has rapidly become the most popular boosting technique in data mining and machine learning applications, and is freely available via the xgboost Python package. BB, also referred to as Bootstrap Aggregating, combines predictions of multiple predictors either via an averaging (regression) or voting (classification) scheme. To address class imbalance, BB resamples (under- or over-samples) each bootstrap training set to make them balanced before training is performed. The free Python package imbalanced-learn allows for both over-sampling and under-sampling of data. We opted for the under-sampling option (which greatly reduces overfitting) of the imbalanced-learn to balance our decoy datasets for BB. The RT method employs the concept of tomeklinks, which are data samples that are each-other's nearest neighbors but have different class labels. Removing tomeklinks is an effective way to eliminate the unwanted class overlap and under-sample the data [41]. We utilize Python's scikit-learn's Nearest Neighbor classifier, which uses a kd-tree, to search for nearest neighbors in our implementation of removing tomeklinks. Out of the three options of removing noisy data samples, we choose to remove both majority and minority classes that are nearest neighbors of each-other to under-sample and balance our decoy datasets.

In the regression setting, we consider two representative methods: XGBoost, described above, and Support Vector Regression (SVR). For SVR, we use Python's scikit-learn's SVR package, which implements an epsilon-support vector regression with a default radial basis function (rbf) kernel and free parameters of penalty and epsilon.

Evaluation for Decoy Selection

The quality of a subset of decoys (pulled from Ω) offered as "prediction", whether the set is learned in an unsupervised or a supervised setting, can be evaluated based on the number of near-native decoys contained in them via two main metrics: (1) n – the percentage of near-native decoys in a given subset relative to the overall number of near-native decoys in the entire decoy set Ω, and (2) p – the percentage of near-native decoys in a given subset over the number of decoys in that same subset. We note that purity penalizes a large subset that, due to its size, may contain a large number of true positives (near-native decoys) but also a high number of false positives (non-native decoys). The reason for penalizing the number of false positives is by envisioning that a learning strategy may be utilized for further discovery. If a subset of decoys is presented to contain the true answer, but the majority of decoys in it are false positives, then the ratio of noise to signal is too high to be useful. On the contrary, a subset of decoys with more near-native decoys in them is a better "prediction," as the likelihood of selecting a near-native decoys by drawing uniformly at random from it is higher when the number of false positives, non-native decoys, is low. We emphasize that in the context of unsupervised learning, the top $c > 1$ groups are offered as prediction; in this setting, the decoys from these groups are combined (which we note via G_{1-m}), and the resulting subset of decoys is evaluated via the n and p metrics. In the context of supervised learning, the groups labeled as true are pulled together to form a subset of decoys (from the overall set Ω), and this subset is evaluated via the n and p metrics described above.

We note that key to the evaluation is the need to determine whether a decoy is considered near-native or non-native; an RMSD threshold is utilized again. The selected threshold needs to allow populating the positive data set (non-zero number of near-native decoys), which can then be used to evaluate a learning strategy. A threshold `dist_thresh` is used and is set on a per-target basis, as there are protein targets on which the quality of generated decoys suffers greatly from either the size and/or fold of the protein under investigation; that is, the quality of the dataset can vary greatly depending on the protein target at hand. In our drawing of a list of representative target proteins, we consider targets that are easy, medium, and hard in their difficulty for the Rosetta ab-initio structure prediction protocol. For instance, we include in our evaluation targets where Rosetta does not get close to 3Å of the known native structure (the ground truth). Specifically, we consider the following thresholds: If the lowest lRMSD from a given native structure (over all decoys), to which we refer as `min_dist`, is ≤ 0.7 (these are considered easy cases), `dist_thresh` is set to 2Å. Otherwise, `dist_thresh` is set to the minimum value that results in a non-zero number of near-native decoys populating the largest-size cluster obtained via leader clustering; the latter is used as a baseline in our comparison of community selection to cluster-based selection. For medium-difficulty proteins (0.7Å$<$ `min_dist` $<$ 2Å), `dist_thresh` varies between $2-4.5$Å. We set `dist_thresh` to 6Å if `min_dist` $\geq$ 2Å (these are the hard cases). This ensures a non-zero number of near-native decoys to evaluate decoy selection strategies. A detailed analysis of the impact of this threshold on cluster-based decoy selection is

related in our recent work [1], where we evaluate basins identified as described above for utilization in decoy selection.

Implementation Details

All in-house codes are implemented in Python. In the nngraph construction, $\delta\varepsilon$=0.2Å, k=20, n_ε=5, and ε_0 is set so as not to exceed 900K edges, varying in 0.5−2.2Å for most of the test cases, with one particularly challenging case for Rosetta set to 6.0Å. The nngraph construction takes between 26 minutes to 4.25 hours on one CPU over decoy data sets of around $50,000$ decoys for proteins ranging in length from 53 to 93 amino acids. We consider two proximity query data structures, kd-tree versus VP-tree; we select the kd-tree over the VP-tree for fast extraction of nearest neighbors, based on analysis of the time demands as a function of dimensionality (data not shown). The community detection methods take between 7 and 66 minutes using 8-cores and 1GB memory per core. The basin detection approach takes between 26 minutes and 2.25 hours on one CPU. In the unsupervised learning setting, we consider $1 \leq c \leq 3$.

In addition to the n and p metrics to evaluate performance, in the supervised learning setting we consider standard metrics, such as accuracy, precision, recall, F-measure, and the number of hits, which indicates how many truly pure groups can be identified. In the classification setting, 15 rounds of boosting are used for XGBoost, using area under the curve (auc) as the evaluation metric. Two variants of XGBoost are considered, XGBoost with undersampling (XGBoost-us), and XGBoost with under-sampling and averaging (XGBoost -us-avg). In XGBoost-us, the data is under-sampled so that the number of true negatives is equal to the number of true positives (for instance, if the size of the true positive set is m, then the dataset size is $2m$). In XGBoost -us-avg, we have multiple subsets of true negatives which are the same size as the true positive set (for, instance, if the dataset size is n and the size of the true positive set is m, then, the number of true negative sets is $k = n/m$, and the total size of the training dataset is $(k+1)m$; the total number of models is k, where each model is trained on a dataset of size $2m$ (m true positives and m true negatives). To make a final prediction, predictions by each individual classifier trained on each of the k under-sampled dataset are averaged. In BB, we investigate different base learners, such as Random Forest, Decision tree, Gradient boosting, AdaBoost, Randomized decision tree, and a voting classifier. We report the best result obtained by using any of these base learners. In RT, we use kd-tree to compute the nearest neighbors and set this number to 2. In SVR in the regression setting, the free penalty and epsilon parameters are set to the default values of 1.0 and 0.01, respectively. In XGBoost for regression, we use 15 rounds of boosting, using the gblinear booster with linear regression as the learning objective. The evaluation metric is set to both root-mean-squared error (RMSE) and mean absolute error (MAE).

Results

Table 1 lists the 10 protein targets selected for evaluation. The decoy ensemble of each target is generated from its amino-acid sequence, running the Rosetta *ab-initio* protocol [22] around $50,000$ times in an embarrassingly parallel fashion in the

Mason Argo supercomputing cluster to obtain ensembles of around $50,000$ decoys per target; the actual ensemble sizes are shown in Column 5 in Table 1. Column 3 shows the Protein Data Bank [3] identifier (PDB id) for a known, crystallographic native structure of each protein. This structure is used to determine which decoys can be considered near-native (via the d_thresh parameter). Table 1 divided the targets into three categories (easy, medium, and hard). This categorization is made evident by findings reported later, but it also emerges from analysis in terms of the lowest (l)RMSD over all decoys from the corresponding native structure. The lowest lRMSD over the decoys for each protein is listed in Column 6 in Table 1 (referred to as min_dist).

Table 1: Column 2 shows the PDB ID of a known native structure for each test case. Columns 3 and 4 show the fold (* indicates native structures with a predominant β fold and a short helix) and the length (number of amino acids), respectively. Column 5 shows the size of the decoy set Ω generated via Rosetta, and column 6 shows the lowest lRMSD from the known native structure over the decoy ensemble.

	PDB ID	**Fold**	**Length (# aas)**	$\lvert\Omega\rvert$	**min_dist (Å)**
Easy	1.1dtdb	$\alpha+\beta$	61	$57,839$	0.51
	2.1tig	$\alpha+\beta$	88	$52,099$	0.60
	3.1dtja	$\alpha+\beta$	74	$53,526$	0.68
Medium	4.1hz6a	$\alpha+\beta$	64	$57,474$	0.72
	5.1c8ca	β^*	64	$53,322$	1.08
	7.1sap	β	66	$51,209$	1.75
Hard	8.2ezk	α	93	$50,192$	2.56
	9.1aoy	α	78	$52,218$	3.26
	10.1isua	*coil*	62	$60,360$	5.53

The findings are presented as follows. First, we demonstrate that two out of the 6 community detection methods are superior in terms of the qualities of the communities they detect. We then evaluate the top communities by each of these two methods via the n and p metric in the unsupervised learning setting, comparing communities to basins. Our findings show that basins are of higher quality. Finally, we relate the performance of the various classification and regression models over identified basins.

Comparison of Community Detection Methods

We consider a comprehensive list of 15 recommended metrics to evaluate the quality of communities identified from a graph via community detection methods [45]. These metrics are scoring functions that mathematically formalize the community-wise connectivity structure of a given set of vertices and characterize high-scored sets as communities. Specifically, we consider the Fraction Over Median Degree (FOMD), the Max ODF (Out Degree Fraction), the Flake ODF, the Triangle(Triad) Participation Ratio, the Internal Edge Density, the Average Internal Degree, the Cut Ratio, the Expansion, the Edges Inside, the Conductance, the Normalized Cut, the Coverage,

the Average ODF, the Modularity, and the Separability metric. A description of these metrics can be found in Ref. [45].

In Figure 1 we relates the comparison along two selected metrics that represent our findings. Specifically, the top panel of Figure 1 shows the comparison along coverage, which measures the number of intra-community edges to $|E(G)|$, and is defined as: $f(S) = \frac{\omega(C)}{\omega(G)}$, where $\omega(C) = \sum_{i=1}^{k} \omega(E(v_x, v_y)); v_x, v_y \in C_i$. Higher values mean that there are more edges inside the communities than edges linking different communities; ideally, communities are disconnected from one another, yielding a maximum coverage of 1. The comparison shows that three methods reach the higher coverage values, GMM, Lo, and LP. The bottom panel of Figure 1 shows the comparison along cut ratio. Cut ratio measures the fraction of existing edges (out of all possible edges) leaving a community, and an average value (averaged over all communities) can be reported to compare different community detection methods. Lower scores correspond to better communities. Due to the wide range over the different community detection methods, we relate log_{10} of this metric in the bottom panel of Figure 1, which shows that the three methods reaching the lowest values are again GMM, Lo, and LP. Many of the metrics (data not shown) point to these three methods as superior over others. In particular, as the bottom panel of Figure 1, GMM and Lo are two of the best-performing methods. The communities that they identify are visualized for one selected target protein in Figure 2.

Evaluation of Unsupervised Learning for Decoy Selection

We now restrict our evaluation in the context of unsupervised learning over communities detected via Lo or GMM, or basins detected as described in Section 2 (we refer to that approach as BF for basin finder) with UL-S, UL-S+E, UL-PR, and UL-PR+PC, evaluating via the n and p the decoys over G_{1-x}, with $x \in \{1,2,3\}$. In the interest of space, we only relate evaluation along G_1 and G_{1-3}, and only relate results from UL-S+E (which is the top or second-top performing unsupervised learning strategy over all datasets, only rarely displaced from the top by UL-PR in a few datasets). Figure 3 shows the comparison along n, and Figure 4 shows the comparison along p. Figure 3 shows no clear winners among community- or basin-based unsupervised learning, but Figure 4, which compares purity, shows the superiority of learned basins over learned communities, suggesting that an unsupervised learning approach operating over basins is more likely to pick up purer subsets of decoys for decoy selection. In particular, these results suggest that ignoring the organization in the energy landscape (and conducting all analysis in the structure space) comes at the cost of allowing false positives in selected decoys. For this reason, the results related below evaluate supervised learning over identified basins.

Evaluation of Supervised Learning for Decoy Selection

Three settings are considered for the evaluation of classification methods: (1) Training of a model on a single target protein (its decoy dataset) an testing on another single protein; (2) Constructing a separate model for each of the three difficulty categories (easy, medium, hard), training a model on 60% of the decoys over all decoys combined over datasets in a category and training the model on 40% of the

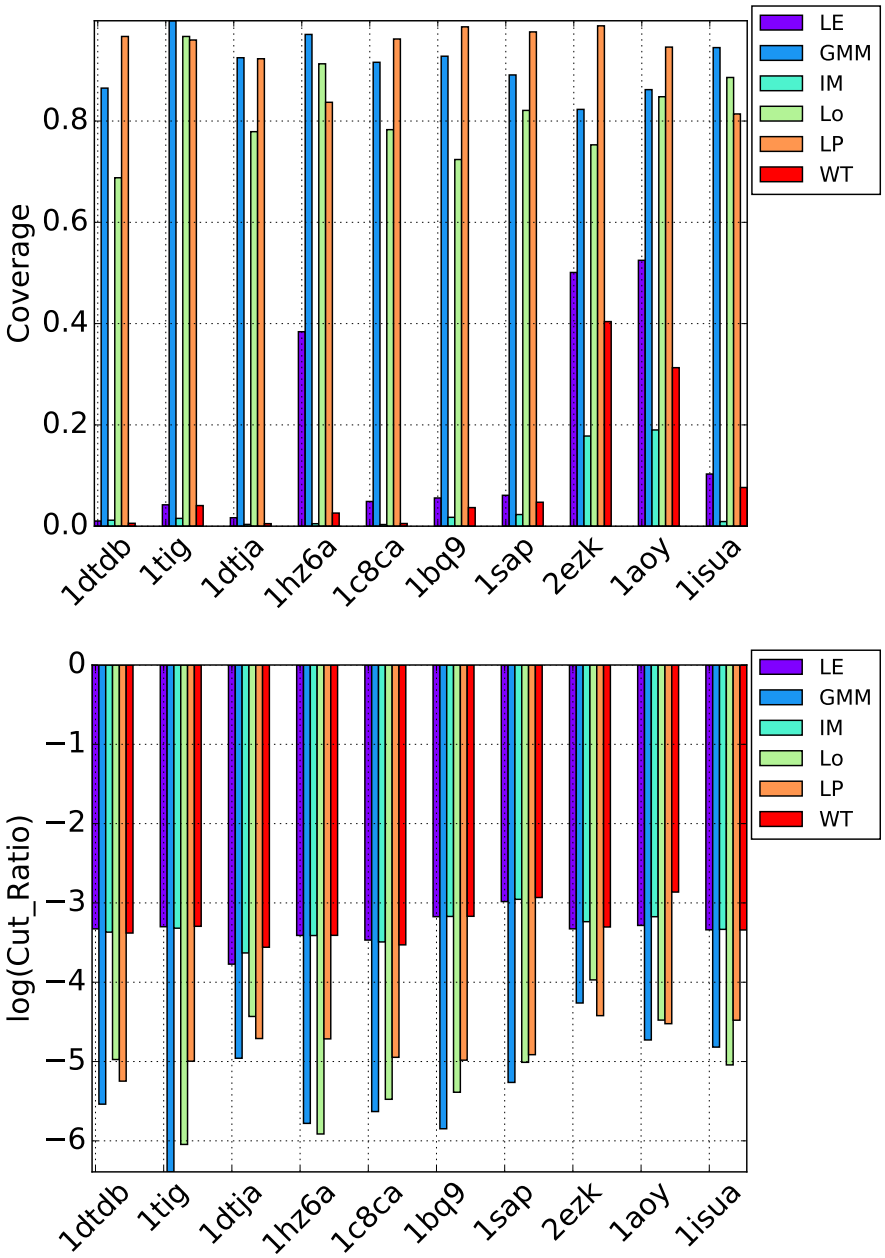

Fig. 1: Comparison of 6 community detection methods (encoded by different colors) on each of the 10 datasets along coverage (top panel) and cut ratio (bottom panel).

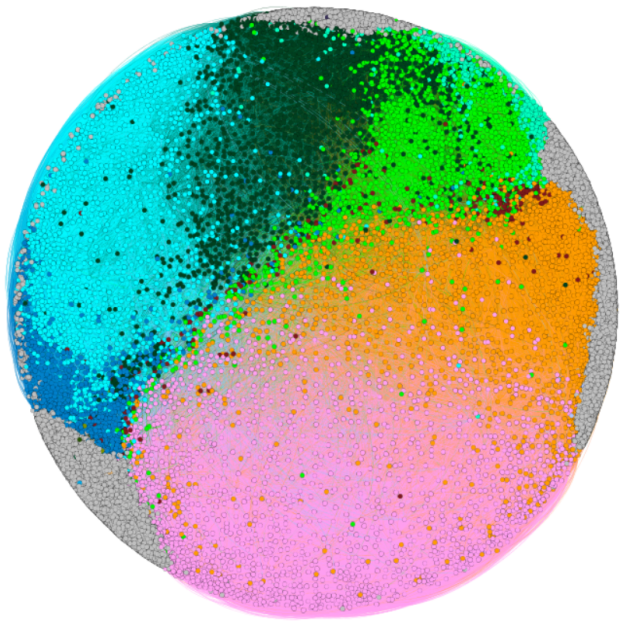

Fig. 2: Communities identified via the Lo community detection method on the decoy dataset of the target protein with a known native structure under PDB id 1dtja are visualized via the force atlas2 layout in Gephi [2]. This layout uses a force-driven, physics-inspired process, where nodes repel and edges attract, to flatten out a graph on a plane and visualize color-coded communities [17].

remaining decoys in that category; (3) Constructing a separate model for each of the three difficulty categories (easy, medium, hard), training a model on all but one of the proteins in a category and testing it on the remaining protein in that category. Evaluation of regression methods is conducted only in the first setting.

Tables 2-4 show performance in each of the three settings for the different classification methods. The best results consisting of high n and p are also highlighted in red and bold. The second-best results where both n and p are also comparatively high are highlighted in blue and bold.

Table 2 shows the classification results in setting 1. In this setting, we predict the n and p for one test protein given another protein as training data. Here, we investigate two categories: easy and medium. Our objective is to show the effect of mixing proteins in different categories as training and test data in the same model. Rows 1, 2, and 5 in Table 2 show the results when we train and test the models on proteins of the same category. The remaining rows show the results when we train and test the models on proteins of different categories.

Table 2 reveals that easy cases perform extremely well when both training and testing are done in the same easy category. The highest purity is 98.3 (train: 1dtdb, test: 1wapa), and there are two models that provide more than 90% of both n and p (XGBoost and BB). Three out of five models output both high n and p (more than 80%): BB, RT, and XGBoost. The last row of Table 2 indicates that the easy

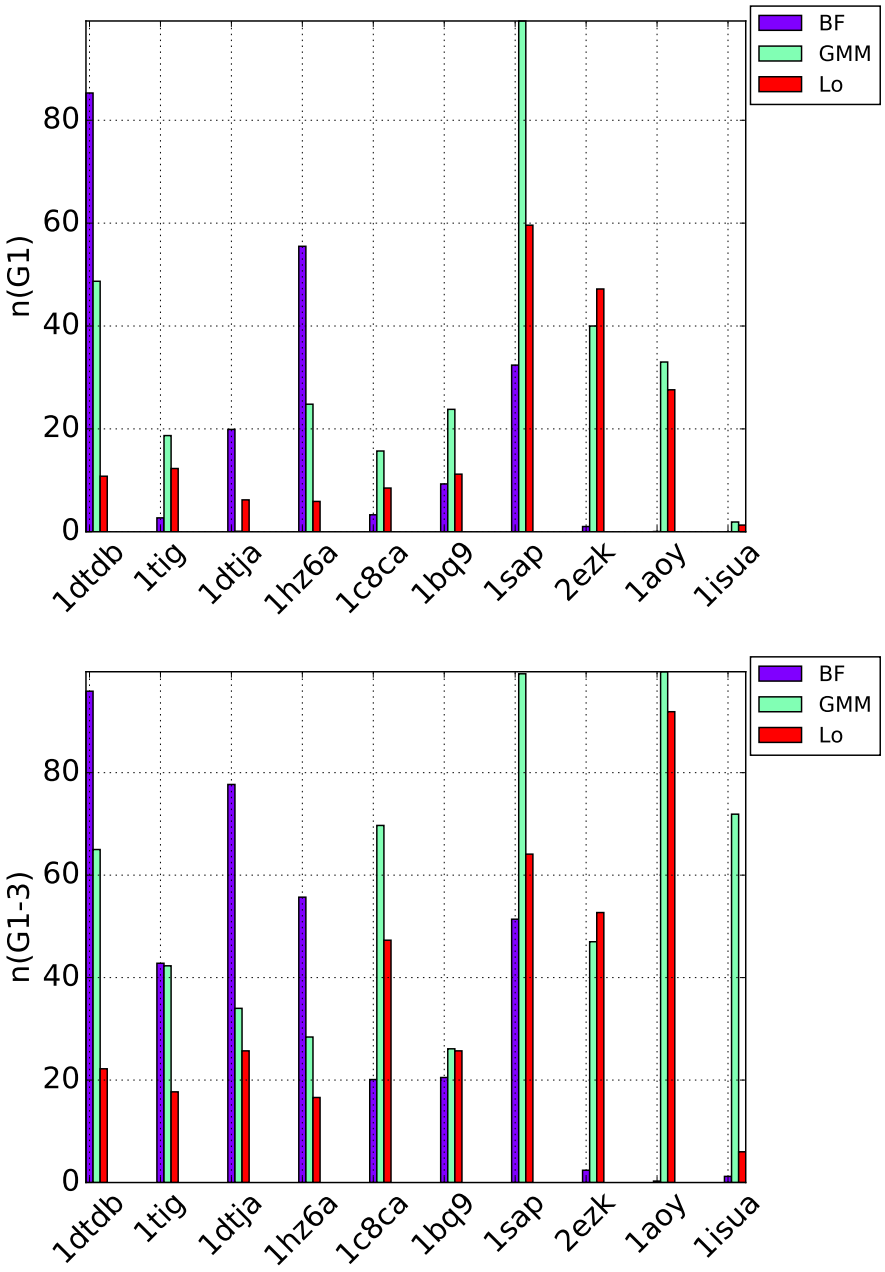

Fig. 3: Comparison of communities and basins in terms of n, selected by size and energy.

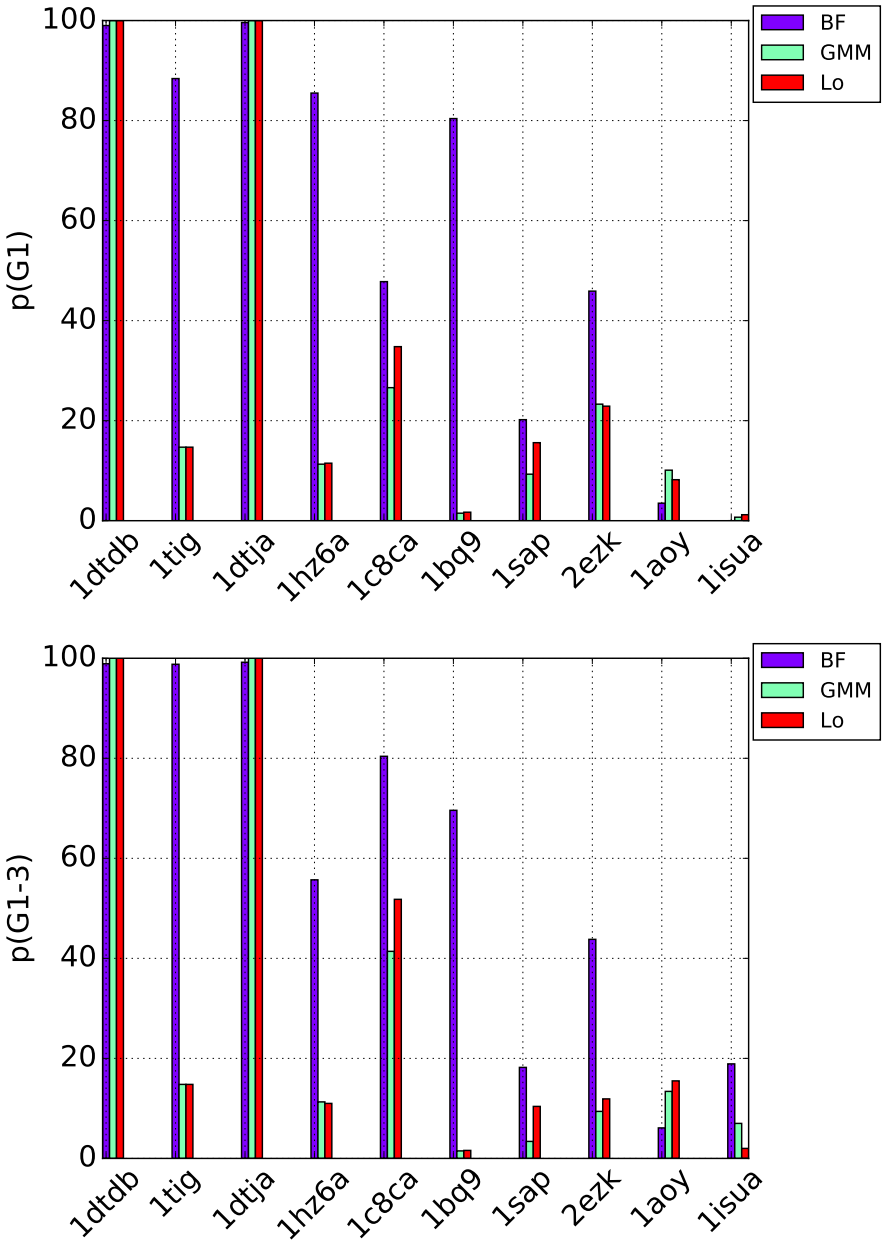

Fig. 4: Comparison of communities and basins in terms of p, selected by size and energy.

Table 2: n: proportion of true positives, p: purity, hit: number of true pure basins detected, pr: precision, r: recall, f: f-measure, acc: accuracy

(train size, test size)	Train, test, # pure basins: train, test	XGBoost	XGBoost -us	XGBoost -us-avg	BB	RT
		n%, p%, hits	n%, p%, hits	n%, p%, hits	n%, p%, hits	n%, p%, hits
		pr, r, f, acc	pr, r, f, acc	pr, r, f, acc	pr, r, f, acc	pr, r, f, acc
train: easy test: easy (3441, 5398)	Train: 1dtdb, test: 1wapa, # pure basins: 13, 18	**98.3**, **94.3**, 15	100, 60, 18	99.9, 59.3, 17	**99.8, 80.6**, 16	**98.2, 85.6**, 14
		0.54, 0.83, 0.65, 0.99	0.1, 1.0, 0.13, 0.96	0.1, 0.99, 0.15, 0.95	0.3, 0.9, 0.44, 0.99	0.41, 0.78, 0.54, 0.99
train: easy test: easy (3441, 2473)	Train: 1dtdb, test: 1dtja, # pure basins: 13, 89	35.2, **94.2**, 17	95.1, 72.6, 47	94.3, 73.1, 43	**90.96**, **91.7**, 29	20, **99.6**, 1
		0.85, 0.2, 0.31, 0.96	0.3, 0.53, 0.37, 0.94	0.31, 0.5, 0.36, 0.93	0.7, 0.33, 0.44, 0.97	1.0, 0.01, 0.02, 0.96
train: easy test: medium (3441, 6435)	Train: 1dtdb, test: 1c8ca, # pure basins: 13, 210	31.6, 31.9, 17	86.2, 20.6, 154	81.3, 21.7, 131	**56.3, 31.4**, 42	**41.3**, **40**, 11
		0.2, 0.08, 0.11, 0.96	0.08, 0.73, 0.14, 0.71	0.08, 0.63, 0.14, 0.74	0.14, 0.2, 0.17, 0.93	0.14, 0.05, 0.08, 0.96
train: easy test: medium (3441, 8571)	Train: 1dtdb, test: 1fwp, # pure basins: 13, 240	3.5, **96.4**, 1	45.7, 37.9, 82	45.9, 38.6, 81	**18.5, 63.6**, 10	25.7, 39.9, 17
		1.0, 0.004, 0.01, 0.97	0.23, 0.34, 0.28, 0.95	0.24, 0.34, 0.25, 0.94	0.4, 0.04, 0.08, 0.97	0.23, 0.07, 0.11, 0.97
train: medium test: medium (6435, 8571)	Train: 1c8ca, test: 1fwp, # pure basins: 210, 240	9.61, **64.57**, 12	**20, 60.3**, 35	**33.9, 52.7**, 65	14.1, 69, 19	2.3, 58.8, 13
		0.41, 0.05, 0.09, 0.97	0.37, 0.15, 0.2, 0.97	0.3, 0.27, 0.26, 0.96	0.42, 0.08, 0.13, 0.97	0.34, 0.05, 0.1, 0.97
train: medium test: medium (6435, 276)	Train: 1c8ca, test: 1hz6a, # pure basins: 210, 5	13, 20.4, 3	11.7, 12.3, 3	**65.4, 28.4**, 3	**60.7, 39**, 3	16, 10.7, 2
		0.2, 0.6, 0.3, 0.95	0.18, 0.6, 0.27, 0.94	0.11, 0.7, 0.2, 0.88	0.25, 0.6, 0.35, 0.96	0.13, 0.4, 0.2, 0.94
train: medium test: easy (6435, 5398)	Train: 1c8ca, test: 1wapa, # pure basins: 210, 18	23.2, 30.4, 17	**100, 55**, 18	**94.7, 46**, 17	**100, 66.4**, 18	23.1, 37.2, 16
		0.07, 0.94, 0.13, 0.96	0.05, 1.0, 0.1, 0.94	0.04, 0.99, 0.07, 0.91	0.08, 1.0, 0.15, 0.96	0.13, 0.89, 0.23, 0.98

cases are good targets even if the training is done on proteins of a different category (medium). We can achieve as high as 100% of n, and purity p is satisfactory (55% and 66.4%). Moreover, easy cases are also good for training even if the target is of a different category (medium case, Rows 3 and 4). Satisfactory purity p is obtained by two models, XGBoost (p is 96.4%) and BB (p is 63.6%). When training and testing are done both on the medium category, performance is not as high as in the other scenarios, but it is still good; the maximum purity achieved in this scenario is 64.57 (Row 5). In summary, BB and XGBoost perform the best among all five models in terms of achieving better p and n. Under-sampling the dataset (XGBoost-us and

XGBoost-us-avg) helps in obtaining comparatively good purity when the target becomes harder (medium case).

Table 3 shows the classification results in setting 2. In this setting, we combine all the test cases of the same category into one dataset, and then do a $60-40$ split to obtain a training and a testing dataset. Table 3 shows the utility of under-sampling the dataset. The best performance in terms of n and p is obtained by XGBoost, when the dataset is under-sampled (n is 89%, and p is 89%). Good results are also achieved by model RT (n is 89%, and p is 75.5%). BB also proves effective for the medium and hard cases (achieving the best and second best results in these categories). Overall, model RT and BB win in this setting, mainly due to the fact that models that focus more on the balancing the data before training (BB, RT, and XGBoost-us) can better address class imbalance in data distribution.

Table 3: n: proportion of true positives, p: purity, hit: number of true pure basins detected, pr: precision, r: recall, f: f-measure, acc: accuracy

	Train, test, # pure basins: test	XGBoost	XGBoost – us	XGBoost -us-avg	BB	RT
		n, p, hits	n, p, hits	n, p, hits	n, p, hits	n, p, hits
		pr, r, f, acc	pr, r, f, acc	pr, r, f, acc	pr, r, f, acc	pr, r, f, acc
easy cases: 1ail, 1dtdb, 1wapa, 1dtja, 1tig train size: 17924 test size: 5975	Train: 75% test: 25% # pure basins in test: 88	97.9, 40.6, 59	**89, 89**, 79	88.2, 44.5, 70	98.8, 40.2, 73	**89, 75.5**, 37
		0.11, 0.67, 0.18, 0.91	0.06, 0.9, 0.11, 0.78	0.05, 0.8, 0.1, 0.78	0.1, 0.83, 0.19, 0.89	0.44, 0.42, 0.43, 0.98
Medium cases: 1hz6a,1c8ca,2ci2, 1bq9, 1fwp,1sap, 1hhp Train size: 30097 Test size: 10033	Train: 75% test: 25% # pure basins in test: 390	53.9, 11.7, 209	55, 12.8, 230	**73.8, 16.7**, 301	**35.1, 19.7**, 137	25, 11.3, 135
		0.11, 0.54, 0.19, 0.82	0.12, 0.59, 0.2, 0.81	0.13, 0.8, 0.22, 0.78	0.17, 0.35, 0.23, 0.91	0.14, 0.35, 0.2, 0.89
Hard cases: 2h5nd, 1aoy, 2ezk, 1cc5, 1isua, 1aly Train size: 29889 Test size: 9964	Train: 75% test: 25% # pure basins in test: 390	59.3, 11.1, 149	61.7, 10.4, 148	63.5, 10.6, 152	**49, 12.4**, 138	27.3, **12.8**, 91
		0.05, 0.7, 0.09, 0.69	0.05, 0.69, 0.09, 0.68	0.05, 0.71, 0.09, 0.68	0.06, 0.64, 0.11, 0.77	0.07, 0.42, 0.12, 0.86

Table 4 shows the classification results in setting 4. In this setting, we combine all the targets of the same category, leaving out only one target of that category for testing, and training on the rest. Table 4 emphasizes the observations revealed by Table 3; when the dataset size is considerably big, balancing the dataset before helps classification models achieve better p and n. Specifically, similar to results related above, BB and RT consistently obtain the best and second best results in n and p for all the cases. They provide purity as high as 85.5% (Row 4) and n as high as 99.3% (row 2). The lowest purity that these two models obtain is 13.3% and p of 22.4% (hard case, last row), which is not a surprise considering the sub-par quality of the decoys in the target 1aoy, and comparable with the results obtained by unsupervised learning (related above).

Table 4: n: proportion of true positives, p: purity, hit: number of true pure basins detected, pr: precision, r: recall, f: f-measure, acc: accuracy

Training models and size	Test size and model, # pure basins: train, test	XGBoost	XGBoost -us	XGBoost -us-avg	BB	RT
		n, p, hits	n, p, hits	n, p, hits	n, p, hits	n, p, hits
		pr, r, f, acc	pr, r, f, acc	pr, r, f, acc	pr, r, f, acc	pr, r, f, acc
Train size: 17896, easy: 1dtdb, 1wapa, 1dtja, 1tig	Test size: 6003, easy: 1ail, # pure basins: 299, 46	95, 22.6, 33	95.6, 20.5, 35	96, 13.5, 32	**94.5, 26.4**, 35	**92.8, 27.6**, 25
		0.12, 0.72, 0.2, 0.96	0.1, 0.76, 0.17, 0.94	0.03, 0.7, 0.05, 0.79	0.12, 0.76, 0.21, 0.96	0.2, 0.54, 0.3, 0.98
Train size: 20458, easy: 1ail, 1wapa, 1dtja, 1tig	Test size: 3441, easy: 1dtdb, # pure basins: 332, 13	99.8, 47.3, 13	**99.4, 68.6**, 12	99.6, 40.4, 12	99.5, 63.8, 12	**99.3, 73.9**, 11
		0.02, 1.0, 0.04, 0.84	0.06, 0.92, 0.11, 0.94	0.01, 0.95, 0.02, 0.62	0.05, 0.92, 0.1, 0.94	0.08, 0.85, 0.15, 0.96
Train size: 39854, medium: 1c8ca, 2ci2, 1bq9, 1fwp, 1sap, 1hhp	Test size: 276: medium: 1hz6a, # pure basins: 1613, 5	57.7, 44.5, 2	59.9, 31.4, 2	83, 15.4, 3	**55.5, 85.5**, 1	**55.5, 72.8**, 1
		0.5, 0.4, 0.44, 0.98	0.09, 1.0, 0.14, 0.91	0.04, 0.76, 0.08, 0.69	1.0, 0.2, 0.33, 0.99	0.5, 0.2, 0.3, 0.98
Train size: 34372, medium: 1c8ca,2ci2, 1bq9,1fwp , 1sap, 1hz6a	Test size: 5718, medium: 1hhp, # pure basins: 1551, 67	**55.4, 28.7**, 20	78.6, 12.6, 37	65.9, 24.1, 28	**46, 43**, 12	51.7, 16.2, 11
		0.19, 0.29, 0.23, 0.98	0.1, 0.55, 0.16, 0.93	0.09, 0.42, 0.14, 0.82	0.2, 0.18, 0.19, 0.98	0.08, 0.16, 0.11, 0.97
Train size: 35729, hard: 2h5nd, 2ezk, 1cc5, 1isua, 1aly	Test size: 4763, hard: 1aoy, # pure basins: 607, 228	59.5, 9.6, 151	**70.6, 10.2**, 161	**70.2, 9.9**, 176	22.4, **13.3**, 50	7.7, 11.6, 14
		0.04, 0.66, 0.08, 0.30	0.05, 0.71, 0.09, 0.28	0.05, 0.77, 0.08, 0.21	0.07, 0.22, 0.11, 0.83	0.06, 0.06, 0.06, 0.91

Table 3 emphasizes the indications revealed by Table 2, when the datasize is considerably big, balancing the dataset to some extent before training the ML model helps in achieving better p and n. Models BB and RT consistently obtained best and second best results in terms of n and p for all the cases. They provide purity as high as high as 85.5 (row 4) and proportion of true positives (n) as high as 99.3% (row 2). The lowest purity that these two models obtain is 13.3% with n of 22.4% (hard case, last row), which is not a surprise considering the category of the target. The highest purity of the same target (1aoy)

Table 5 shows the results of the regression models. In this setting, the goal is to predict the purity of the target test case. Table 5 shows the effectiveness of the boosting technique, revealed by XGBoost. In most of the cases, the lowest RMSE and MAE are obtained by XGBoost. The lowest MAE (0.0092) results on the easy cases (train: 1dtdb, test: 1wapa), which reiterates the fact that these targets are easy for prediction due to the quality of Rosetta-generated decoys for them. Although

XGBoost provides superior results, SVR proves its utility in medium cases by providing better MAE.

Table 5: Comparison of regression models alpng RMSE and MAE

Train and test models (datasize)	SVR	XGBoost
Training: Easy cases, test: Easy and medium cases		
Train: easy, 1dtdb, 60% (2064) Test: easy, 1dtdb, 40% (1377)	RMSE: **0.0757** MAE: **0.0352**	RMSE: 0.1175 MAE: 0.0771
Train: easy, 1dtdb (3441) Test: easy, 1wapa (5398)	RMSE: 0.0734 MAE: 0.0187	RMSE: **0.0562** MAE: **0.0092**
Train: easy, 1dtdb (3441) Test: easy, 1dtja (3863)	RMSE: 0.2002 MAE: 0.0564	RMSE: **0.1957** MAE: **0.0487**
Train: easy, 1dtdb (3441) Test: medium, 1c8ca (6435)	RMSE: 0.2148 MAE: 0.0780	RMSE: **0.2097** MAE: **0.0749**
Training: Medium case, test: easy and medium cases		
Train: medium, 1c8ca, 60% (3861) Test: medium, 1c8ca, 40% (2574)	RMSE: 0.2122 MAE: **0.0750**	RMSE: **0.1953** MAE: 0.1078
Train: medium, 1c8ca (6435) Test: medium, 1fwp (8571)	RMSE: 0.2036 MAE: 0.0789	RMSE: **0.1716** MAE: **0.0750**
Train: medium, 1c8ca (6435) test: medium, 1hz6a (276)	RMSE: 0.1700 MAE: **0.0545**	RMSE: **0.1525** MAE: 0.0814
Train: medium, 1c8ca (6435) test: medium, 1sap (5096)	RMSE: 0.1073 MAE: **0.0403**	RMSE: **0.1070** MAE: 0.0915
Train: medium, 1c8ca (6435) test: easy, 1dtdb (3441)	RMSE: 0.1034 MAE: **0.0317**	RMSE: **0.0911** MAE: 0.0626

Altogether, the results reported in the Tables 2-5 suggest the promise that supervised learning methods hold in basin-based decoy selection. In many cases, the results are comparable with those obtained via unsupervised learning.

Notes

The findings we relate in this paper show the utility of leveraging the organization of protein structure spaces or energy landscapes to learn which structural states are relevant for function. In particular, better results are obtained in the strict context of decoy selection when leveraging the organization of the energy landscape rather than ignoring energy and conducting all analysis on the structure space. In addition to evaluating performance in the context of unsupervised learning, this paper evaluates several supervised learning methods under the umbrella of both classification and regression. Though a very challenging setting due to the class imbalance distribution and the varying quality of decoys, the presented results show that in many cases, supervised learning methods achieve the same performance as unsupervised learning ones. It should be noted that the supervised learning methods we adopt here to assess their utility in decoy selection are pretty simple. However, even these simple methods are able to provide us with competitive results, suggesting further research in this direction is promising.

The work opens several lines of enquiry. For instance, the decoys in identified communities or basins can be further assessed by different scoring functions for indicators of nativeness. Community detection itself can be improved and lifted from the structure space to the energy landscape, by integrating energy in the construction of the nngraph. While the presented work focuses on an application in template-free protein structure prediction, the work may be useful in other settings where an organization of uncomplexed or complexed molecular structure data promises to reveal functionally-relevant structural states captured *in silico*.

References

[1] Akhter N, Shehu A (2017) From extraction of local structures of protein energy landscapes to improved decoy selection in template-free protein structure prediction. Molecules 23(1):216

[2] Bastian M, Heymann S, Jacomy M (2009) Gephi: an open source software for exploring and manipulating networks. In: Intl AAAI Conf on Weblogs and Social Media, AAS, pp 361–362

[3] Berman HM, Henrick K, Nakamura H (2003) Announcing the worldwide Protein Data Bank. Nat Struct Biol 10(12):980–980

[4] Boehr DD, Wright PE (2008) How do proteins interact? Science 320(5882):1429–1430

[5] Boehr DD, Nussinov R, Wright PE (2009) The role of dynamic conformational ensembles in biomolecular recognition. Nature Chem Biol 5(11):789–796

[6] Bryngelson JD, Onuchic JN, Socci ND, Wolynes PG (1995) Funnels, pathways, and the energy landscape of protein folding: a synthesis. Proteins: Struct Funct Genet 21(3):167–195

[7] Cao R, Wang Z, Wang Y, Cheng J (2014) Smoq: a tool for predicting the absolute residue-specific quality of a single protein model with support vector machines. BMC bioinformatics 15(1):120

[8] Cazals F, Dreyfus T (2017) The structural bioinformatics library: modeling in biomolecular science and beyond. Bioinformatics 33(7):997–1004

[9] Chatterjee S, Ghosh S, Vishveshwara S (2013) Network properties of decoys and casp predicted models: a comparison with native protein structures. Molecular BioSystems 9(7):1774–1788

[10] Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, ACM, pp 785–794

[11] Clausen R, Shehu A (2014) A multiscale hybrid evolutionary algorithm to obtain sample-based representations of multi-basin protein energy landscapes. In: ACM Conf on Bioinf and Comp Biol (BCB), Newport Beach, CA, pp 269–278

[12] Frauenfelder H, Sligar SG, Wolynes PG (1991) The energy landscapes and motion on proteins. Science 254(5038):1598–1603

[13] Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Annals of statistics pp 1189–1232

[14] Ginalski K, Elofsson A, Fischer D, Rychlewski L (2003) 3D-Jury: a simple approach to improve protein structure predictions. Bioinformatics 19(8):1015–1018

[15] Guan W, Ozakin A, Gray A, et al (2011) Learning protein folding energy functions. In: Intl Conf Data Mining, IEEE, pp 1062–1067

[16] He Z, Alazmi M, Zhang J, Xu D (2013) Protein structural model selection by combining consensus and single scoring methods. PloS one 8(9):e74,006

[17] Jacomy M, Venturini T, Heymann S, Bastian M (2014) ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. PLoS ONE 9(6):e98,679

[18] Jing X, Wang K, Lu R, Dong Q (2016) Sorting protein decoys by machine-learning-to-rank. Scientific Reports 6:31,571

[19] Kabsch W (1976) A solution for the best rotation to relate two sets of vectors. Acta Cryst A32:922–923

[20] Kryshtafovych A, Fidelis K, Tramontano A (2011) Evaluation of model quality predictions in CASP9. Proteins 79(Suppl 10):91–106

[21] Kryshtafovych A, Barbato A, Fidelis K, Monastyrskyy B, Schwede T, Tramontano A (2014) Assessment of the assessment: evaluation of the model quality estimates in CASP10. Proteins 82(Suppl 2):112–126

[22] Leaver-Fay A, et al (2011) ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. Methods Enzymol 487:545–574

[23] Liu T, Wang Y, Eickholt J, Wang Z (2016) Benchmarking deep networks for predicting residue-specific quality of individual protein models in casp11. Scientific reports 6:19,301

[24] Lorenzen S, Zhang Y (2007) Identification of near-native structures by clustering protein docking conformations. PROTEINS: Structure, Function, and Bioinformatics 68(1):187–194

[25] Manavalan B, Lee J, Lee J (2014) Random forest-based protein model quality assessment (rfmqa) using structural features and potential energy terms. PloS one 9(9):e106,542

[26] Maximova T, Moffatt R, Ma B, Nussinov R, Shehu A (2016) Principles and overview of sampling methods for modeling macromolecular structure and dynamics. PLoS Comp Biol 12(4):e1004,619

[27] Mirzaei S, Sidi T, Keasar C, Crivelli S (2016) Purely structural protein scoring functions using support vector machine and ensemble learning. IEEE/ACM transactions on computational biology and bioinformatics

[28] Molloy K, Saleh S, Shehu A (2013) Probabilistic search and energy guidance for biased decoy sampling in ab-initio protein structure prediction. IEEE/ACM Trans Bioinf and Comp Biol 10(5):1162–1175

[29] Moult J, Fidelis K, Kryshtafovych A, Schwede T, Tramontano A (2014) Critical assessment of methods of protein structure prediction (CASP) – round X. Proteins: Struct Funct Bioinf 82:109–115

[30] Moult J, Fidelis K, Kryshtafovych A, Schwede T, A T (2017) Critical assessment of methods of protein structure prediction (CASP) - round XII. Proteins DOI 10.1002/prot.25415, in press

[31] Nguyen SP, Shang Y, Xu D (2014) Dl-pro: A novel deep learning method for protein model quality assessment. In: Neural Networks (IJCNN), 2014 International Joint Conference on, IEEE, pp 2071–2078

[32] Nussinov R, Wolynes PG (2014) A second molecular biology revolution? the energy landscapes of biomolecular function. Phys Chem Chem Phys 16(14):6321–6322

[33] Okazaki K, Koga N, Takada S, Onuchic JN, Wolynes PG (2006) Multiple-basin energy landscapes for large-amplitude conformational motions of proteins: Structure-based molecular dynamics simulations. Proc Natl Acad Sci USA 103(32):11,844–11,849

[34] Olson B, Shehu A (2013) Multi-objective stochastic search for sampling local minima in the protein energy surface. In: ACM Conf on Bioinf and Comp Biol (BCB), Washington, D. C., pp 430–439

[35] Pawlowski M, Kozlowski L, Kloczkowski A (2016) Mqapsingle: A quasi single-model approach for estimation of the quality of individual protein structure models. Proteins: Structure, Function, and Bioinformatics 84(8):1021–1028

[36] Samoilenko S (2008) Fitness landscapes of complex systems: Insights and implications on managing a conflict environment of organizations. Complexity & Organization 10(4):38–45

[37] Shehu A (2013) Probabilistic search and optimization for protein energy landscapes. In: Aluru S, Singh A (eds) Handbook of Computational Molecular Biology, Chapman & Hall/CRC Computer & Information Science Series

[38] Shehu A (2015) A review of evolutionary algorithms for computing functional conformations of protein molecules. In: Zhang W (ed) Computer-Aided Drug Discovery, Springer Methods in Pharmacology and Toxicology Series

[39] Shehu A, Plaku E (2016) A survey of omputational treatments of biomolecules by robotics-inspired methods modeling equilibrium structure and dynamics. J Artif Intel Res 597:509–572

[40] Shehu A, Clementi C, Kavraki LE (2007) Sampling conformation space to model equilibrium fluctuations in proteins. Algorithmica 48(4):303–327

[41] Tomek I (1976) Two modifications of CNN. IEEE Trans Systems, Man and Cybernetics 6:769–772

[42] Uziela K, Wallner B (2016) Proq2: estimation of model accuracy implemented in rosetta. Bioinformatics 32(9):1411–1413

[43] Wallner B, Elofsson A (2006) Identification of correct regions in protein models using structural, alignment, and consensus information. Protein Sci 15(4):900–913

[44] Xu D, Zhang Y (2012) Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field. Proteins: Struct Funct Bioinf 80(7):1715–1735, DOI 10.1002/prot.24065

[45] Yang J, Leskovec J (2012) Defining and evaluating network communities based on ground-truth. In: Intl Conf on Data Mining (ICDM), pp 745–754

[46] Yang Z, Algesheimer R, Tessone CJ (2016) A comparative analysis of community detection algorithms on artificial networks. Sci Reports 6:30,750

[47] Zhang Y, Skolnick J (2004) Spicker: A clustering approach to identify near-native protein folds. Journal of computational chemistry 25(6):865–871
[48] Zhou ZH (2012) Ensemble methods: foundations and algorithms. CRC press