Ensemble prediction of protein secondary structure through feature-based accuracy estimation

Spencer Krieger[*] and John Kececioglu

Department of Computer Science, The University of Arizona, Tucson, Arizona, USA

[*]Corresponding author. Email: `skrieger@arizona.edu`

Abstract

Protein secondary structure prediction is a fundamental task in computational biology, basic to many bioinformatics workflows, with a diverse collection of tools currently available. An approach from machine learning with the potential to capitalize on such a collection is *ensemble prediction*, which runs multiple predictors and combines their predictions into a single one, output by the ensemble.

We conduct a thorough study of eight different approaches to ensemble secondary structure prediction, several of which are novel, and show we can indeed obtain an ensemble method that significantly exceeds the accuracy of individual state-of-the-art tools. The best approaches build on a recent technique known as *feature-based accuracy estimation*, which estimates the unknown true accuracy of a prediction, here using features of both the prediction output and information internal to the prediction method. In particular, a *hybrid approach* to ensemble prediction that leverages accuracy estimation is now the most accurate method currently available: on average over standard CASP and PDB benchmarks, it exceeds the state-of-the-art Q_3 accuracy for 3-state prediction by nearly 4%, and exceeds the state-of-the-art Q_8 accuracy for 8-state prediction by more than 8%.

An implementation of our hybrid approach to ensemble protein secondary structure prediction in a new tool we call `Ssylla`, together with accuracy estimators for the best prediction methods in widespread use, is available at `http://ssylla.cs.arizona.edu`.

Author summary

The biological function of a protein molecule is largely determined by the three-dimensional shape that it folds up into in the cell. A simplified representation of this folded shape known as the protein's secondary structure classifies the amino acids of the molecule into a discrete set of structural states. The structure of most proteins is currently unknown, as it is difficult to determine experimentally; instead, their structure is often predicted computationally from the known sequence of amino acids along the molecule. This task of protein secondary structure prediction is challenging, and has led to a multitude of computational methods. Since these competing methods possess unique strengths, developing a computational approach that can harness their complementary strengths to yield a superior prediction would be advantageous.

Ensemble prediction combines the multiple outputs from a collection of prediction methods into a single prediction output by the ensemble. We develop several new approaches to ensemble prediction using the recent technique of feature-based accuracy estimation, which estimates the unknown true accuracy of a method's prediction, and compare these novel ensemble variants to state of the art individual methods and conventional ensemble approaches. The best ensemble variant, called hybrid selection, leverages accuracy estimation to significantly surpass the state of the art. Furthermore, this new approach to ensemble prediction is general, and is applicable beyond protein secondary structure to diverse prediction tasks in both computational biology and other fields.

Introduction

Protein secondary structure prediction is fundamental in computational biology and bioinformatics, and has many applications, such as enhancing the accuracy of protein tertiary structure prediction [1], solvent accessibility prediction [2], and protein multiple sequence alignment [3, 4].

In general, the prediction task is: given the *primary sequence* S for a protein of length n over amino acid alphabet Σ, to predict the corresponding *secondary structure sequence* P of length n for the folded protein, which specifies the discrete secondary structure state of the residue associated with the amino acid at each position in S.

In the standard 8-*state model* (Kabsch and Sander [5]), structure sequence P is over the alphabet $\Gamma_8 := \{\mathtt{B}, \mathtt{C}, \mathtt{E}, \mathtt{G}, \mathtt{H}, \mathtt{I}, \mathtt{S}, \mathtt{T}\}$, where symbol `B` denotes isolated β-bridge, `E` is extended β-strand; `G` is 3_{10}-helix, `H` is α-helix, `I` is π-helix; `C` is coil, `S` is bend, and `T` is bonded turn.

More typically, in the reduced 3-*state model*, structure sequence P is over the alphabet $\Gamma_3 := \{\alpha, \beta, \gamma\}$, where usually $\alpha = \{\mathtt{G}, \mathtt{H}, \mathtt{I}\}$ (class α-helix), $\beta = \{\mathtt{B}, \mathtt{E}\}$ (class β-strand), and $\gamma = \{\mathtt{C}, \mathtt{S}, \mathtt{T}\}$ (class coil, the "other" class that contains everything else).

Predicting from primary sequence S the correct secondary structure sequence P that encodes the protein's unknown folded structure under models Γ_3 or Γ_8 is challenging, and has a long history of development of many computational approaches.

Related work

A plethora of tools are currently available for protein secondary structure prediction. Well over 250 methods have been published since the 1970s (see [6]), and there are many tools in widespread use today. `PSIPRED` (Jones [7]) was among the first to leverage a `PSI-BLAST` [8] protein sequence database homology search coupled with a neural network, while `JPred` (Drozdetskiy, Cole, Proctor, and Barton [9]) incorporates a `HMMer` [10] search as well. `SSpro` (Pollastri, Przybylski, Rost, and Baldi [11]) searches a smaller template database to find homologous proteins with known secondary structure, relying on a `PSI-BLAST` search and neural network when template matches are not found. `Porter` (Mirabello and Pollastri [12]), `DeepCNF` (Wang, Peng, Ma, and Xu [13]), and `MUFOLD` (Fang, Shang, and Xu [14]) each use variations on deep neural networks, including convolutional neural networks, conditional neural fields, and inception-inside-inception networks, while `PSRSM` (Ma, Liu, and Cheng [15]) uses a semi-random subspace variant of support vector machines—all in conjunction with `PSI-BLAST`. `Nnessy` (Krieger and Kececioglu [16, 17]) is unique in that it forgoes a time-intensive `PSI-BLAST` search, instead replacing it with metric-space nearest neighbor search over a template database of short protein words, coupled with a fast dynamic programming algorithm for finding an optimal physically-valid prediction of maximum likelihood.

Ensemble methods from machine learning (see [18]) are an established means of harnessing such a collection of prediction tools. The main techniques for *training* an ensemble of predictors from the same model class are boosting, which reweights training examples sequentially to form linked predictors; and bagging, which resamples the training set in parallel to form independent predictors. Given an ensemble of predictors, the main techniques for *combining* their predictions into one that is output are averaging, often over real-valued output vectors; voting, including weights; and stacking, which trains a higher-level predictor to map the many predictions in the ensemble to one that is output.

While most of the commonly-used tools for protein secondary structure prediction apply bagging internally to train a uniform collection of classifiers all coming from the same model class, the thrust of our work here is how to best utilize a diverse collection of external prediction tools coming from distinct or even arbitrary model classes that we cannot control or train ourselves. Accordingly, our emphasis is on how to optimally combine the predictions from such tools into the most accurate one output.

To briefly survey the literature on ensemble secondary structure prediction, many methods employ bagging to train a *homogeneous* collection of classifiers (typically either neural networks or support vector machines) that they then combine by stacking a final classifier of the same type [19–24]. Only a few methods combine a *heterogeneous* collection of secondary structure predictors [25–30]. Guermeur, Geourjon, Gallinari, and Deléage [25] apply multivariate linear regression to secondary structure class scores output by the predictors. Selbig, Mevissen, and Lengauer [26] build decision trees to combine predictions. Ouali and King [31] use linear or quadratic discrimination (and neural networks) to combine classifiers. Guermeur, Pollastri, Elisseeff, Zelus, Paugam-Moisy, and Baldi [27] train a multi-class support vector machine on predictor class score vectors, while Liu, Carbonell, Klein-Seetharaman, and Gopalakrishnan [32] employ conditional random fields on such vectors. Bouziane, Messabih, and Chouarfia [28, 29] combine class membership probability vectors by a variety of arithmetic functions, while Aydin and Uzut [30] directly average them. Notably, all of these heterogeneous ensemble methods for secondary structure prediction that perform the following comparisons achieve only a very minor improvement over simple unweighted voting (an order of magnitude less than 1% increase in Q_3 accuracy), and a fairly modest improvement over the best single predictor (around a 1% increase in Q_3 accuracy).

In contrast, we combine ensemble predictions by leveraging the recent technique of *feature-based accuracy estimation* (DeBlasio and Kececioglu [33–35]), originally developed in the context of parameter advising for protein multiple sequence alignment [36, 37]. This technique estimates the unknown true accuracy of a prediction using efficiently-computable real-valued feature functions evaluated on the prediction, assuming features can be discovered whose values tend to be correlated with true accuracy. The form of accuracy estimation we develop here is somewhat different than [36], since good estimators on secondary structures will need to be functions of auxiliary information internal to the prediction method (whereas the original estimators in [36] were strictly functions of a prediction). As a consequence, we now have to learn separate estimators specific to each secondary structure prediction tool in our ensemble. We also construct our estimators somewhat differently, as they now require learning an intermediate transformation on feature values. Feature-based accuracy estimation leads to a large boost in ensemble prediction accuracy, both over the best single predictor and over simple voting, as we summarize next.

Our contributions

We present a new approach to ensemble protein secondary structure prediction that leverages feature-based accuracy estimation to significantly surpass the state-of-the-art for both 3- and 8-state prediction. In particular, this work makes the following contributions.

- Our best approach to ensemble prediction is *significantly more accurate* than any single prediction tool averaged over standard CASP and PDB benchmarks, exceeding the state-of-the-art Q_3 accuracy for 3-state structure prediction by nearly 4%, and exceeding the state-of-the-art Q_8 accuracy for 8-state prediction by more than 8%.

- Our new ensemble approach leverages *feature-based accuracy estimation*, which estimates the unknown true accuracy of a secondary structure prediction using easily-computed features of both the predicted structure and its associated prediction method. As part of this work we provide, for the first time, concrete accuracy estimators for *all* widely-used secondary structure prediction tools, for both Q_3 and Q_8 accuracy. The average error of these estimators is currently around 5-7% for Q_3 and around 6-11% for Q_8 accuracy.
- The error in our accuracy estimators has the potential to improve on discovering new informative features. We demonstrate through simulation that, even keeping current prediction tools unchanged, each successive *reduction in estimator error* yields a further boost in ensemble accuracy.
- Combining predictions through feature-based accuracy estimation is significantly *more accurate than simple approaches*: compared to uniform voting, it boosts the Q_3 ensemble accuracy by more than 2%, and Q_8 ensemble accuracy by nearly 5%.
- In contrast to voting-based approaches to ensemble prediction, which eventually degrade as more predictors are added, our new estimator-based approach on adding further predictors successively *improves or remains stable* in accuracy.

In brief, our ensemble approach is the most accurate method currently available for both 3- and 8-state prediction, on all standard benchmarks.

Lastly we emphasize that the general estimator-based approach to ensemble prediction leveraged here has *broad applicability*, and applies to *any* ensemble prediction task where accuracy estimators can be constructed.

An implementation of our best approach to ensemble prediction in a new tool we call `Ssylla` (short for "protein secondary structure prediction by an ensemble leveraging accuracy estimation"), together with accuracy estimators for the best widely-used prediction tools, is available free for non-commercial use at `http://ssylla.cs.arizona.edu`.

Overview

The methods section next describes our general approach to ensemble prediction, including the ensemble variants we study, and how we estimate the unknown accuracy of predictions. The results section then presents experimental results comparing these ensemble variants to state-of-the-art tools for both 3- and 8-state structure prediction. The discussion section delves into their behavior, and through simulation examines how with further improvement in accuracy estimation they can reach the empirical limit on ensemble accuracy. We close with a summary and suggestions for further research.

Methods

Ensemble prediction takes a collection of prediction methods, applies them to an input instance of the prediction problem, and combines their individual predictions into a single prediction that is output by the ensemble. We first describe the variants that we consider for combining the secondary structure predictions from the ensemble, several of which require an estimate of the accuracy of a prediction. We then explain how we estimate the unknown true accuracy of a secondary structure prediction by applying the recent technique of feature-based accuracy estimation.

Ensemble variants

In general, given an input protein sequence S and an ensemble $\mathcal{E} = \{M_1, \ldots, M_k\}$ of k protein secondary structure prediction methods M_i, we want to combine their individual structure predictions $P_1, \ldots, P_k$, where $P_i := M_i(S)$, into a single prediction $\mathcal{P}$ that is output by ensemble $\mathcal{E}$. Protein sequence S is a string of length n over the 20-letter amino acid alphabet, while predictions P_i and $\mathcal{P}$ are corresponding secondary structure strings of length n over the 3- or 8-symbol secondary structure alphabets Γ_3 or Γ_8, for respectively 3- or 8-state structure prediction.

Each position j in protein sequence S is referred to as a *residue*. In a structure prediction P, the corresponding symbol $P[j]$ at position j gives the predicted secondary structure state of the jth residue in S.

We consider two categories of ensemble variants. In *per-protein variants*, the prediction $\mathcal{P}$ output by ensemble $\mathcal{E}$ is one of the protein structure predictions computed by a method in the ensemble: specifically, $\mathcal{P} := P_i$ for some $i \in \{1, \ldots, k\}$. In *per-residue variants*, the ensemble prediction $\mathcal{P}$ may combine different predictions P_i at different residues j: formally, the ensemble's predicted structural state $\mathcal{P}[j] := P_{I(j)}[j]$, where at each residue j the function $I(j)$ can potentially select a different prediction P_i.

We describe per-residue variants first, and then per-protein variants.

Per-residue variants

The ensemble variants that operate per-residue use forms of weighted voting to choose the predicted structural state $\mathcal{P}[j]$. At each residue position j, and every structure state q in secondary structure alphabet Γ, state q at position j receives a *total vote*,

$$V_j(q) \quad := \sum_{1 \,\le\, i \,\le\, k} w(i, j, q)\,,$$

where weight function $w(i, j, q)$ gives the real-valued vote of method M_i at residue j for state q. (For most of the variants we consider, the vote of a method M_i at residue j is concentrated on the single state $P_i[j]$, as in practice many tools only ouput a single prediction P_i, but we also consider variants where the vote for a method at a residue is spread over all states $q \in \Gamma$.) Per-residue variants choose, for their ensemble prediction $\mathcal{P}$ at residue j, the state with the highest total vote,

$$\mathcal{P}[j] \quad := \quad \operatorname*{argmax}_{q \,\in\, \Gamma} V_j(q)\,.$$

These variants differ in how weight w is computed.

Uniform voting With *uniform voting*, each prediction receives a uniform weight,

$$w(i, j, q) \quad := \quad \begin{cases} 1, & \text{if } q = P_i[j]; \\ 0, & \text{otherwise.} \end{cases}$$

Effectively, every method gives the same weight to its predicted state $P_i[j]$.

While uniform voting is simple and is frequently used for ensemble prediction, a consequence of every method having the same weight—including weak predictors that can corrupt $V_j(q)$ with poor votes—is that eventually the accuracy of the ensemble tends to degrade as the number of methods k increases (as shown in our experiments later (Section “Effect of ensemble size”)).

Probability voting With *probability voting*, each method votes for the structure states at each residue with its class membership probabilities. More formally, we assume each prediction method M_i at residue j computes structure class membership probabilities, in particular, the probability $p_{ij}(q) \geq 0$ of predicting state q, where $\sum_{q \in \Gamma} p_{ij}(q) = 1$. Given these probabilities, the weight function is simply,

$$w(i,j,q) \quad := \quad p_{ij}(q)\,.$$

(In this variant, a method can vote for multiple structure states at a residue.) Some prediction tools do not output class membership probabilities, in which case we concentrate all of the probability mass on the predicted state: $p_{ij}(P_i[j]) = 1$ (which degenerates to uniform voting).

Probability voting is equivalent to uniformly averaging class membership probability vectors (or averaging the normalized class score vectors obtained for instance from neural networks), and is the approach perhaps most often used to combine classifiers that produce class membership probabilities; in the machine learning literature, it is sometimes called “soft voting” (see for instance [18, pp. 75–77]).

Confidence voting In *confidence voting*, we weight votes by the confidence values that many standard protein secondary structure prediction tools output at residues. At position j, we assume prediction method M_i provides a confidence value $C_i(j)$ for its predicted state $P_i[j]$, which reflects its degree of certainty in its prediction at that residue. The weight function is then simply confidence:

$$w(i,j,q) \quad := \quad \begin{cases} C_i(j), & \text{if } q = P_i[j]; \\ 0, & \text{otherwise.} \end{cases}$$

(Typically, confidence $C_i(j)$ is computed internally by method M_i from estimated structure class membership probabilities at residue j, by taking the difference between the highest and second-highest structure class probabilities, as described later (Section “Choosing features”). In practice, the confidence that a prediction tool outputs is often scaled and rounded to an integer in the range $[0, 9]$.) With confidence voting, a structural state that is chosen by methods with higher confidence receives a greater vote.

One aspect of confidence voting is that while the relative values for confidences may be meaningful across residues within the same method, their values are not always comparable between methods, so methods that are less objective about their certainty can still distort vote $V_j(q)$.

Estimator voting In *estimator voting*, we weight votes by an estimate of the accuracy of a method’s prediction at that residue. While in practice the true accuracy of a prediction is usually not known (since a protein’s ground-truth secondary structure is often unavailable), we can estimate its unknown true accuracy using features of both a structure prediction and its prediction method. The recent technique of feature-based *accuracy estimation* [34, 36], described later (Section “Accuracy estimation”), combines together efficiently-computable real-valued features that tend to be correlated with true accuracy to construct an accuracy estimator that is trained on benchmark data for which true accuracy is known. Given an accuracy estimator $E_i(j)$ with range $[0, 1]$ that returns an estimate of the true accuracy of prediction $P_i[j]$, the weight function is then simply,

$$w(i,j,q) \quad := \quad \begin{cases} E_i(j), & \text{if } q = P_i[j]; \\ 0, & \text{otherwise.} \end{cases}$$

(We note that for some secondary structure prediction tools, such as `JPred` [9], the best estimator we can construct gives a single estimated accuracy value for the entire prediction P_i, not per residue; in that case $E_i(j)$ is independent of j, and the method votes with the estimated accuracy of its whole prediction at every residue.)

While the burden is now on the construction of a good estimator, estimator voting offers many advantages: in contrast to both uniform and probability voting, it tends to be stable even when weak predictors are added to the ensemble, and in contrast to confidence voting, it puts the weights of votes on a comparable basis across methods.

Per-protein variants

The variants that operate per-protein evaluate a specific scoring function F on each of the protein structure predictions P_i returned by the k methods in the ensemble, and select the ensemble prediction $\mathcal{P}$ of highest score,

$$\mathcal{P} \;:=\; \operatorname*{argmax}_{Q \,\in\, \{P_1,\, \dots,\, P_k\}} F(Q)\,.$$

These variants differ by the scoring function F they use for selection.

Support selection In *support selection*, the scoring function on a protein structure prediction Q is the average across residues of the total vote for Q under uniform voting,

$$F(Q) \;:=\; \frac{1}{n} \sum_{1 \,\le\, j \,\le\, n} \Big| \big\{ i \,:\, P_i[j] = Q[j] \big\} \Big|\,.$$

Effectively, support selection chooses the prediction that has the most support from other predictions in the ensemble. While this is natural, support selection can suffer on hard instances of the prediction task where most predictors in the ensemble are inaccurate, which may cause the rare prediction that is accurate to receive little support.

Confidence selection In *confidence selection*, the scoring function on structure prediction P_i in the ensemble is the average confidence of the prediction,

$$F\big(P_i\big) \;:=\; \frac{1}{n} \sum_{1 \,\le\, j \,\le\, n} C_i(j)\,.$$

The predictor that is the most confident in its own prediction dominates the ensemble.

Estimator selection In *estimator selection*, the scoring function is the average estimated accuracy of a prediction,

$$F\big(P_i\big) \;:=\; \frac{1}{n} \sum_{1 \,\le\, j \,\le\, n} E_i(j)\,.$$

Ensemble prediction by estimator selection was originally introduced by DeBlasio and Kececioglu [35] (see also [36, Chapter 7]). Their context was ensemble multiple sequence alignment, which they viewed as an instance of parameter advising where the parameter is the choice of aligner.

Hybrid selection An exception to this general form is *hybrid selection* (introduced by Krieger and Kececioglu [16] under the name "method hybridization"), where the ensemble hybridizes two complementary methods—a core method M_{core} and an alternate method M_{alt}—using an accuracy estimator threshold τ. Informally, if the prediction from the core method has average estimated accuracy above the threshold, hybrid selection chooses the core prediction, otherwise it chooses the alternate prediction. In other words, this approach takes the core prediction if it is estimated to be sufficiently accurate, and if not, it falls back on the alternate prediction. (In general, the alternate method M_{alt} can itself be another ensemble method; if M_{alt} recursively uses hybrid selection again, this results in a kind of decision-tree approach.)

For the *core method*, in practice we use an approach that is template-based, such as `Nnessy` [16] or `SSpro` [11], which are very accurate when their template database finds sufficiently good matches to the input protein sequence (but can be inaccurate in the absence of good template matches). For the *alternate method*, we use an approach that is template-free, as they are generally more consistent (but do not achieve the upper accuracies of template-based approaches). Hybridizing a templated-based and a template-free method in this way, discerning when to switch between them by thresholding an accuracy estimator, yields an ensemble that enjoys the benefits of both: often attaining the peak accuracies of the template-based method while retaining the consistency of the template-free one.

More formally, a core prediction $P_{\mathrm{core}} := M_{\mathrm{core}}(S)$ is first computed. Then using the scoring function $F(P)$ for estimator selection, the average estimated accuracy for the core prediction $E_{\mathrm{core}} := F(P_{\mathrm{core}})$ is evaluated. If $E_{\mathrm{core}} \geq \tau$, the hybrid ensemble selects the core prediction, $\mathcal{P} := P_{\mathrm{core}}$. Otherwise, the hybrid computes the alternate prediction $P_{\mathrm{alt}} := M_{\mathrm{alt}}(S)$, which it then selects by $\mathcal{P} := P_{\mathrm{alt}}$.

As our results later show (Section "Prediction accuracy"), while per-residue variants may seem more powerful than per-protein variants due to their potential to combine the best parts of different predictions into one ensemble prediction, generally the per-protein variants are more accurate than their per-residue counterparts. This may be due to per-protein variants, in averaging their selection criteria across residues, tending to dampen the unavoidable noise present at individual residues. Furthermore, hybrid selection is surprisingly strong, achieving the highest accuracy of all these approaches, while using an ensemble of only two prediction methods.

Accuracy estimation

The ensemble variants of estimator voting, estimator selection, and hybrid selection, which as we will see later (Section "Prediction accuracy") are among the best ensemble approaches for secondary structure prediction, all require every method in the ensemble to have an associated *accuracy estimator* that estimates the unknown true accuracy of the method's prediction. Of the many software tools available for protein secondary structure prediction, only `Nnessy` [16, 17] currently provides an accuracy estimator. We now describe how we construct, for the first time, accuracy estimators for other state-of-the-art secondary structure tools, using the recent technique of *feature-based accuracy estimation* [36]. This new technique was originally developed for estimating the unknown true accuracy of protein multiple sequence alignments [33, 34, 37]. How we apply this technique to protein secondary structure predictions is somewhat different, as in this new context we need an estimate of the accuracy of the prediction at each residue of the protein (as opposed to a single estimate for an entire alignment), and to get a good estimator we will need to leverage additional information that is internal to the prediction method (as opposed to solely basing the estimate on information that is contained in the external structure prediction itself). A consequence of the novel approach described below is that we also obtain a new estimator for `Nnessy` [17] that further improves upon its original accuracy [16].

The technique of feature-based accuracy estimation learns an estimator that takes real-valued features of a prediction and maps them to an accuracy estimate (here an estimated Q_3 or Q_8 accuracy value). The key is having easily-computed *features* of a prediction that tend to be correlated with true accuracy. (Such features may be challenging to discover.) The final *estimator* is obtained from an affine combination of transformed feature-values that is fitted on training data consisting of benchmark predictions with associated true-accuracy values (which requires knowing a ground-truth secondary structure for the benchmark's proteins). We next describe the features we use for our accuracy estimators, and then how we fit an estimator to these transformed features.

Choosing features

A feature of a protein secondary structure prediction is a real-valued function f of the structure prediction P and the prediction method M, which we write as $f(P, M)$. As ultimately a feature will be mapped by an estimator to an accuracy value, which is in the range $[0, 1]$, it is desirable for the feature function f to ideally have a bounded range. For the feature f to be useful for accuracy estimation, it should be efficiently computable, and the value of $f(P, M)$ for a given method M should be correlated (positively or negatively) with the true accuracy of prediction P.

To estimate the accuracy of a protein secondary structure prediction, we make use of two distinct features, both of which are not strictly functions of a prediction P but also require additional information from the method M: (1) *residue confidence*, which is a quantity that many prediction tools already output at each residue (or which can be computed from other information they output such as structure class membership probabilities for residues), that reflects the tool's degree of certainty in its prediction at that residue; and (2) *template similarity*, which for template-based prediction tools is a measure of the similarity of the template database matches to the input protein amino-acid sequence S around the given residue position (which for evaluation requires having access to the source code for the template-based tool). Both of these features tend to be correlated with true accuracy: as a method's confidence increases in its predicted structure states at residues, the accuracy of the protein's structure prediction tends to increase, and as the template database matches become more similar to the amino acid sequence surrounding a residue, the secondary structure sequence associated with the template match tends to better agree with the correct secondary structure around that residue for the input protein.

We also considered other features besides these two (for example, the entropy of the prediction's estimated state probabilities at residues), but only found confidence and similarity to have sufficiently strong correlation with true accuracy.

Residue confidence Tools that provide confidence values at residues for their structure prediction typically output an integer from $\{0, 1, \ldots, 9\}$, where a larger value reflects greater confidence in the residue's predicted state. Tools that do not directly provide confidences fortunately instead often output estimated structure class membership probabilities at residues. For residue position j, and each possible structure state $q \in \Gamma$, such tools give an estimated probability $p_j(q)$ that residue j is in state q, where $\sum_q p_j(q) = 1$. Let $s := \mathrm{argmax}_q \; p_j(q)$ be the highest-probability state at residue j. A confidence value can then be derived from these estimated structure probabilities by computing the difference $p_j(s) - \max_{q \neq s} p_j(q)$ between the highest and second-highest probabilities (and for compatibility with standard tool confidences, scaling this difference by a factor of 10). Note that confidence essentially reflects how unambiguous is the state of highest estimated structure probability.

We denote the residue *confidence feature function* by f_{con}, where $f_{\mathrm{con}}(P, M)$ either returns a vector of confidence values for corresponding residue positions, or a single aggregate confidence value for the entire prediction P.

Template similarity For template-based tools, we measure the similarity of the template database match to the query sequence coming from the input protein around the residue position. The similarity between these two portions of amino acid sequence from the template match and the query sequence is either measured by average percent identity (the fraction of amino acids that agree) or average word distance (the average substitution dissimilarity score between the corresponding amino acids).

We denote the template *similarity feature function* by f_{sim}, which either returns a vector of similarity values for residues, or an aggregate similarity value for the entire prediction.

Fitting an estimator

Our accuracy estimators evaluate their estimate by a two-step process:

(1) for each feature function f_i used by the estimator, the individual feature f_i is first *mapped* via a transformation into an initial accuracy estimator $A_i(P, M)$; and

(2) the initial estimators A_i are then linearly *combined* into the final accuracy estimator $\mathcal{A}$,

$$\mathcal{A}(P, M) \ := \ c_0 \ + \ \sum_i c_i \, A_i(P, M) \, ,$$

where coefficients $c_0, c_1, \ldots$ specify the affine combination.

For feature functions f_{con} and f_{sim}, we have initial estimators A_{con} and A_{sim}.

For the first step, we consider two ways of mapping a feature f_i into an initial accuracy estimator A_i, both of which use a piecewise-linear transform function. With *per-residue mapping*, we first transform feature f at a residue into an accuracy estimate at that residue, and then average these individual residue accuracy estimates across the protein, to obtain an initial accuracy estimate for the whole prediction. With *per-protein mapping*, we first average the feature f for a residue across the entire protein, and then transform this single average feature value for the entire protein into an initial accuracy estimate for the whole prediction.

For per-residue mapping, we obtain the piecewise-linear transform function as follows. On training data consisting of predictions for which we know their true accuracy, we evaluate the feature function at all residues of the training proteins, sort these residues by their feature value, and for each residue, take a window of the closest residues in feature value centered at that residue, and evaluate the empirical accuracy over the residues in the window by counting how many of them were correctly predicted divided by the window size. This process yields data where we have pairs consisting of the feature value for residues together with associated empirical accuracies for residues. We then fit a piecewise-linear function to this data, to obtain a transform function that maps a residue feature value to an estimated accuracy. The initial estimator evaluates this transform at all residues of the protein, and averages the mapped estimated accuracies across the prediction. (This process is later shown in Fig 5.)

For per-protein mapping, the piecewise-linear transform function is obtained similarly, except it is fit to data over proteins, where the pair of values for a protein is its average feature value across all residues, and the true accuracy of the prediction for the protein.

Generally, we find that per-residue mapping yields an initial accuracy estimator that has a stronger correlation with true accuracy, and hence a better final accuracy estimator. However, for the confidence feature, tools that directly provide confidences give values with a coarse integer resolution of $\{0, \ldots, 9\}$, which is not well-suited for the per-residue mapping process, so for confidence we rely on per-protein mapping.

For the second step of combining initial estimators, we find the coefficients c_i for final estimator $\mathcal{A}$ through linear regression on training data consisting of predictions (from which we compute initial estimator values) and their associated true accuracy.

This estimator fitting process is shown concretely later (Section "Construction of accuracy estimators").

Results

We now describe our experimental setup, compare the prediction accuracies of ensemble variants to state-of-the-art tools, enumerate which tools comprise the best ensembles, and examine the error in accuracy estimators.

Datasets, parameter tuning, and implementation

To evaluate ensemble variants and prediction tools, we used both CASP benchmark datasets [38–41], and datasets extracted from PDB, the Protein Data Bank [42]. For the CASP datasets (named `CASP10` through `CASP13`), all proteins were used. For the PDB datasets, which are organized by year from 2016 to 2019 (named `PDB2016` through `PDB2019`), a random subset of 100 proteins deposited into PDB that year were used, except for `PDB2019` which contains all proteins deposited between 1 January 2019 and 15 May 2019. From all datasets, we filtered out proteins shorter than length 23, or whose sequences contained more than five ambiguous amino acids.

Below are brief summary statistics for each dataset, giving their number of proteins and average protein length in amino acids.

	`CASP10`	`CASP11`	`CASP12`	`CASP13`	`PDB2016`	`PDB2017`	`PDB2018`	`PDB2019`
proteins	129	105	55	49	100	100	100	292
length	177	183	188	280	318	318	211	281

We formed ensembles from the following state-of-the-art tools (listed along with their version number): `DeepCNF` 1.02 [13], `JPred` 4 [9], `MUFOLD` 3 [14], `Nnessy` 1.0.2 [16, 17], `Porter` 5 [12], `PSIPRED` 4.02 [7], and `SSpro` 5.2 [11]. (`PSRSM` [15], one of the most accurate tools for 3-state prediction, was not included as it outputs neither state probabilities nor confidences, which our ensembles require.)

For the tool `Nnessy`, which is template-based, when evaluating it on a `CASP` or `PDB` dataset its template database was constructed only over proteins deposited into PDB prior to the date of the CASP competition, or prior to the year of the `PDB` dataset. (While `SSpro` is also template-based, we did not modify the template database used by its software; this tends to inflate its prediction accuracy on `CASP` and `PDB` datasets whose years fall within those of its template database construction.)

Accuracy estimators for these tools were fit using the proteins from all `CASP` and `PDB` datasets. We did not perform leave-one-out cross validation when fitting estimators on proteins: the regression for an estimator fits at most four parameters, so removing one protein (out of nearly a thousand) would have a negligible effect on an estimator.

Source code (in Python) for the highest-accuracy ensemble variant—namely hybrid selection—in a new tool we call `Ssylla` [43], together with accuracy estimators for all of the above prediction tools, as well as these `CASP` and `PDB` datasets, are available at `http://ssylla.cs.arizona.edu`.

Prediction accuracy

We next report prediction accuracies on these `CASP` and `PDB` benchmarks.

To measure the accuracy of 3- or 8-state secondary structure prediction on a benchmark dataset we use, respectively, *standard Q_3 or Q_8 accuracy*: the percentage of residues in a protein where the predicted state, under model Γ_3 or Γ_8, matches the correct state from the gold-standard structure for the benchmark protein, averaged over all the proteins in the dataset.

For each ensemble variant, we report the Q_3 or Q_8 accuracy of the *optimal ensemble* for that variant. More precisely, over all the tools that perform 3- or 8-state structure prediction, for each variant we found the subset of tools whose ensemble had the highest Q_3 or Q_8 accuracy averaged over all `CASP` and `PDB` datasets (where each dataset is weighted equally when averaging), by exhaustively enumerating all subsets of tools. (As a point of comparison, we also compute the specific ensembles that are optimal for each particular dataset, though our main focus is on the general ensemble that has optimal accuracy averaged equally across datasets.)

We note that tools have differing requirements for the proteins on which they can predict secondary structure (in terms of the minimum length of a protein sequence, and the maximum number of ambiguous amino acids they allow). The accuracy we report for a tool is over the subset of proteins in a dataset on which that tool can make predictions. Similarly, the accuracy we report for an ensemble predicting on a protein is over the subset of tools in the ensemble that can make predictions on that protein.

We first report these accuracies for individual tools compared to the most accurate ensemble variant, and then report accuracies for all ensemble variants described earlier (Section “Ensemble variants”).

Comparing the best ensemble to individual tools The best ensemble method significantly outperforms *all* individual tools, consistently across the `CASP` and `PDB` datasets, for both 3- and 8-state structure prediction.

Tables 1 and 2 show the mean and standard deviation of respectively the Q_3 and Q_8 accuracies for:

- individual prediction tools,
- the *specific* ensembles that are optimal for each dataset, and
- the *general* ensemble that is optimal when averaged across all datasets.

(These dataset-specific ensembles provide an upper bound on the best-possible accuracy achieveable on a given dataset by a general ensemble.) Next we dive into the results in these tables.

The best ensemble for both 3- and 8-state prediction, when averaging over all datasets, uses *hybrid selection.* This optimal general ensemble forms a hybrid of `Nnessy` as the core method with `Porter` as the alternate method. Surprisingly, this `Nnessy`/`Porter`-hybrid, which exceeds all other ensemble variants, outperforms in particular every hybrid selection ensemble where either `Nnessy` or `SSpro` as the core method are hybridized with an *ensemble* of other tools, considering all ensemble variants as the alternate method. This is due to error in the accuracy estimators, which for instance causes lower-accuracy alternate predictions to be chosen by the `Nnessy`/ensemble-hybrid, compared to `Porter`'s higher-accuracy alternate predictions in the `Nnessy`/`Porter`-hybrid.[1]

Tables 1 and 2 highlight in bold for each dataset the best Q_3 and Q_8 accuracy achieved by a single tool (where the best tool varies per dataset), as well as the accuracy achieved on that dataset by the best general ensemble (that has highest

[1]This behavior occurs especially on `CASP12` and `CASP13`, where `Porter` has high accuracy while other tools have unusually low accuracy (including `MUFOLD`, as explained later). Specifically, for 3-state prediction, `Porter` alone achieves an average accuracy of 84.0%, while the best ensemble with which to hybridize `Nnessy` (that excludes `Nnessy`) has a lower accuracy of 83.6%. Similarly, for 8-state prediction, `Porter`'s average accuracy is 71.2%, while the best non-`Nnessy` ensemble only gets 70.5%.

Table 1. Three-State Accuracies on CASP and PDB Datasets

	Q_3 accuracy (%)								
Method	CASP10	CASP11	CASP12	CASP13	PDB2016	PDB2017	PDB2018	PDB2019	average
JPred	77.7 ± 7.9	77.5 ± 7.0	75.3 ± 9.5	75.4 ± 9.2	78.5 ± 6.6	77.7 ± 7.0	78.1 ± 7.4	77.1 ± 7.6	77.2
Nnessy	75.5 ± 18.5	74.5 ± 17.0	67.9 ± 15.8	75.7 ± 16.3	87.3 ± 12.0	**84.9** ± 12.7	**87.4** ± 10.5	**86.3** ± 10.7	79.9
DeepCNF	83.3 ± 7.7	82.4 ± 6.0	81.1 ± 8.2	78.0 ± 8.6	81.7 ± 10.7	81.9 ± 7.4	83.0 ± 7.7	81.0 ± 8.9	81.6
PSIPRED	**85.0** ± 7.4	83.8 ± 5.8	80.2 ± 9.9	80.0 ± 8.1	84.2 ± 6.6	82.8 ± 7.3	80.1 ± 8.8	79.8 ± 8.5	82.0
SSpro	84.7 ± 11.5	**85.2** ± 10.4	74.6 ± 10.7	76.2 ± 9.7	**87.5** ± 7.5	84.5 ± 9.3	84.6 ± 9.7	82.1 ± 12.9	82.4
Porter	84.7 ± 4.7	83.8 ± 4.7	**83.0** ± 6.4	**81.8** ± 4.6	85.1 ± 4.2	84.6 ± 3.7	84.8 ± 4.7	84.3 ± 4.1	**84.0**
MUFOLD[⋆]	86.6 ± 8.1	85.4 ± 6.5	81.9 ± 9.1	83.6 ± 5.4	86.7 ± 5.7	84.9 ± 7.9	85.2 ± 7.8	83.4 ± 8.6	(84.7)
Ensemble, specific[†]	(89.8 ± 8.6)	(89.4 ± 7.9)	(84.9 ± 7.5)	(85.5 ± 6.6)	(90.1 ± 10.3)	(88.6 ± 7.8)	(89.6 ± 6.5)	(88.2 ± 7.9)	—
Ensemble, general[‡]	**89.3** ± 8.4	**88.4** ± 8.1	**84.9** ± 7.5	**85.5** ± 6.6	**89.7** ± 6.5	**88.5** ± 7.9	**89.4** ± 6.9	**87.8** ± 8.6	**87.9**

[⋆]MUFOLD predicts on the most restricted subset of each dataset, and is not the most accurate tool on its subset.
[†]"Ensemble, specific" lists accuracies for the different, specific ensembles that achieve optimum accuracy on each dataset.
[‡]"Ensemble, general" gives the accuracy of the single, general ensemble that achieves optimal accuracy averaged across datasets.

Table 2. Eight-State Accuracies on CASP and PDB Datasets

	Q_8 accuracy (%)								
Method	CASP10	CASP11	CASP12	CASP13	PDB2016	PDB2017	PDB2018	PDB2019	average
DeepCNF	72.7 ± 10.8	72.7 ± 8.0	68.4 ± 13.2	64.3 ± 11.4	71.7 ± 12.0	71.6 ± 9.5	73.7 ± 10.2	68.4 ± 11.8	70.4
Porter	74.5 ± 7.7	74.5 ± 6.3	**72.6** ± 9.5	**68.8** ± 9.2	75.2 ± 6.4	73.7 ± 6.6	73.7 ± 7.6	72.4 ± 6.8	71.2
SSpro	**76.6** ± 15.9	**77.3** ± 14.8	62.0 ± 14.7	59.4 ± 16.6	80.3 ± 12.7	75.5 ± 14.3	75.9 ± 15.1	73.4 ± 18.7	72.6
Nnessy	67.4 ± 22.4	67.2 ± 21.3	55.5 ± 21.8	67.0 ± 21.2	**82.5** ± 15.3	**80.1** ± 17.7	**82.8** ± 15.7	**82.4** ± 15.6	**73.1**
MUFOLD[⋆]	76.1 ± 11.4	75.9 ± 8.3	70.7 ± 12.9	67.2 ± 11.8	76.6 ± 9.0	74.2 ± 9.3	74.7 ± 10.9	71.3 ± 11.2	(73.3)
Ensemble, specific[†]	(82.2 ± 13.5)	(83.4 ± 11.8)	(75.0 ± 11.4)	(75.3 ± 14.7)	(85.4 ± 11.8)	(84.1 ± 11.6)	(84.6 ± 10.6)	(84.0 ± 12.9)	—
Ensemble, general[‡]	**81.6** ± 14.2	**82.3** ± 12.6	**75.0** ± 11.4	**74.6** ± 11.6	**85.4** ± 11.8	**84.1** ± 11.6	**84.6** ± 10.6	**84.0** ± 12.9	**81.5**

[⋆]MUFOLD predicts on the most restricted subset of each dataset, and is not the most accurate tool on its subset.
[†]"Ensemble, specific" lists accuracies for the different, specific ensembles that achieve optimum accuracy on each dataset.
[‡]"Ensemble, general" gives the accuracy of the single, general ensemble that achieves optimal accuracy averaged across datasets.

accuracy averaged over all datasets); the final column highlights in bold the highest average accuracy (weighting datasets equally) of both the best tool and ensemble. The *best tool* overall for 3-state prediction is `Porter`, and for 8-state prediction is `Nnessy`.[2] The *best ensemble*, on the other hand (which again is the `Nnessy`/`Porter`-hybrid both for 3- and 8-state prediction), consistently outperforms all tools on every dataset, and significantly improves upon the overall accuracy of the best individual tool: for 3-state prediction by nearly 4%, and for 8-state prediction by more than 8%.

We also mention that the best *general* ensemble comes remarkably close to the accuracy upper-bound given by the best *specific* ensemble for each dataset: on some datasets they agree, while on most they differ in Q_3 and Q_8 accuracy by much less than 1%.

Obviously, 8-state prediction is harder than 3-state prediction, both for individual tools and the best ensemble. (Datasets from `CASP` also tend to be harder than from `PDB`.) The variance in accuracy of a method on a dataset is larger as well for 8-state than 3-state prediction.

The template-based tools `SSpro` and `Nnessy` notably have the most inconsistent performance across datasets. For `SSpro`, its template database is drawn from proteins submitted to `PDB` before 2017, and this may explain its markedly higher accuracy on `CASP10`, `CASP11`, and `PDB2016` than other datasets (as its database likely contains structure templates similar to their proteins). For `Nnessy`, which is markedly better on `PDB` datasets compared to `CASP`, recall that when it is evaluated on a given dataset, its template database is deliberately formed only over proteins deposited into `PDB` *prior* to the date of the given `PDB` or `CASP` dataset. Even so, the `PDB` dataset for a particular year—which might arguably better-reflect actual bioinformatics practice for that year than the corresponding `CASP` dataset—could contain proteins similar to those deposited in previous years. Furthermore, `CASP` datasets are intentionally designed to contain proteins that differ from prior `PDB` entries, making them especially challenging for template-based tools.

Given that the `Nnessy`/`Porter`-hybrid had the highest 3- and 8-state average accuracy, we further evaluated `Nnessy`, `Porter`, and their hybrid on the most recent `CASP` dataset, `CASP14` [44]. The 3- and 8-state accuracies on `CASP14` for `Porter` are respectively 82.0% and 70.2%; for `Nnessy`, they are 77.1% and 66.7%. When hybridized together, their hybrid's accuracy rises to respectively 83.5% and 71.5%. These accuracies on `CASP14`, and the boost from hybridizing `Nnessy` with `Porter`, are consistent with their performance trend on the other `CASP` datasets.

Comparing ensemble variants In brief, as noted before the *best ensemble variant* for both 3- and 8-state structure prediction, that achieves the highest accuracy averaged across all datasets, is *hybrid selection*. Variants attaining the top accuracy on individual datasets are most frequently hybrid selection and estimator selection, with per-protein selection variants generally tending to outperform per-residue voting variants.

Tables 3 and 4 show in full detail the mean and standard deviation of the Q_3 and Q_8 accuracies of all ensemble variants described earlier (Section "Ensemble variants") on all datasets. The accuracy shown on a given dataset for a given variant is again for the *optimal ensemble* employing that variant, which achieves the highest accuracy averaged across all datasets (found by exhaustive enumeration of all subsets of tools).

[2] The average accuracy for `MUFOLD`, which appears to be the highest in both tables, is in parentheses for the following reasons. `MUFOLD` has the most severe input requirements of any tool: it disallows *any* ambiguous amino acids in a protein sequence. Consequently, these tabulated `MUFOLD` accuracies are over the most restricted subset of the data. Since predicting the secondary structure of proteins that do contain ambiguous amino acids is generally harder, the 3- and 8-state `MUFOLD` accuracies reflect prediction over a comparatively easier subset of proteins. Moreover, when evaluated on this easier subset, other tools actually achieve higher accuracy than `MUFOLD`.

Table 3. Three-State Accuracies of Optimized Ensemble Variants on `CASP` and `PDB` Datasets

	Q_3 accuracy (%)								
Method	`CASP10`	`CASP11`	`CASP12`	`CASP13`	`PDB2016`	`PDB2017`	`PDB2018`	`PDB2019`	average
Per-residue voting									
Probability	86.4 ± 7.2	85.8 ± 5.7	81.7 ± 9.0	83.3 ± 6.9	89.0 ± 9.9	88.1 ± 7.5	88.1 ± 6.7	83.0 ± 9.3	85.6
Uniform	86.8 ± 7.2	86.0 ± 6.5	80.5 ± 9.0	82.3 ± 7.6	88.5 ± 9.6	87.1 ± 7.3	88.2 ± 6.9	85.1 ± 11.2	85.6
Confidence	87.7 ± 7.8	87.4 ± 6.1	82.0 ± 9.4	84.0 ± 8.3	89.9 ± 9.8	88.7 ± 7.8	89.2 ± 6.7	87.2 ± 9.1	87.0
Estimator	87.9 ± 7.7	87.5 ± 6.3	81.6 ± 9.5	83.6 ± 7.8	**90.1** ± 10.3	88.6 ± 7.9	89.4 ± 6.9	87.4 ± 9.2	87.0
Per-protein selection									
Support	86.8 ± 8.3	86.2 ± 7.1	81.4 ± 9.4	81.5 ± 9.0	89.2 ± 9.8	87.6 ± 7.9	88.3 ± 7.3	86.1 ± 9.7	85.9
Confidence	87.1 ± 6.5	83.7 ± 6.3	82.5 ± 8.3	81.8 ± 7.5	88.2 ± 11.5	88.0 ± 9.4	89.0 ± 7.1	87.5 ± 9.0	86.0
Estimator	89.1 ± 7.6	87.6 ± 7.2	83.2 ± 9.0	84.8 ± 8.2	89.5 ± 10.9	**89.0** ± 8.1	**89.9** ± 7.0	87.4 ± 10.3	87.6
Hybrid	**89.3** ± 8.4	**88.4** ± 8.1	**84.9** ± 7.5	**85.5** ± 6.6	89.7 ± 6.5	88.5 ± 7.9	89.4 ± 6.9	**87.8** ± 8.6	**87.9**

Table 4. Eight-State Accuracies of Optimized Ensemble Variants on `CASP` and `PDB` Datasets

	Q_8 accuracy (%)								
Method	`CASP10`	`CASP11`	`CASP12`	`CASP13`	`PDB2016`	`PDB2017`	`PDB2018`	`PDB2019`	average
Per-residue voting									
Probability	74.4 ± 12.5	77.0 ± 9.3	67.1 ± 16.2	69.7 ± 11.8	79.8 ± 12.0	78.9 ± 9.9	80.0 ± 9.7	76.8 ± 11.8	75.8
Uniform	77.0 ± 11.9	77.7 ± 10.2	66.2 ± 15.2	67.7 ± 12.6	83.4 ± 13.4	80.4 ± 12.8	80.9 ± 12.4	79.3 ± 15.7	76.6
Confidence	80.5 ± 10.6	80.0 ± 9.6	69.7 ± 14.6	71.7 ± 15.5	84.7 ± 12.1	82.3 ± 12.3	83.7 ± 11.0	78.3 ± 21.4	78.9
Estimator	79.9 ± 12.0	80.4 ± 10.3	67.8 ± 15.5	71.5 ± 12.4	84.3 ± 13.5	81.6 ± 12.8	82.8 ± 11.7	80.9 ± 14.9	78.7
Per-protein selection									
Support	77.6 ± 13.1	76.3 ± 11.5	66.8 ± 14.6	65.1 ± 16.1	82.3 ± 12.2	78.5 ± 12.7	79.8 ± 11.9	77.5 ± 15.6	75.5
Confidence	77.6 ± 10.6	76.1 ± 9.0	72.4 ± 14.1	71.5 ± 16.0	84.0 ± 12.9	82.2 ± 12.7	83.2 ± 11.1	83.4 ± 13.1	78.8
Estimator	**82.2** ± 13.5	**83.4** ± 11.8	70.4 ± 16.6	**75.2** ± 15.7	84.5 ± 13.5	81.8 ± 14.8	82.8 ± 14.4	82.2 ± 15.4	80.3
Hybrid	81.6 ± 14.2	82.3 ± 12.6	**75.0** ± 11.4	74.6 ± 11.6	**85.4** ± 11.8	**84.1** ± 11.6	**84.6** ± 10.6	**84.0** ± 12.9	**81.5**

Table 5. Composition of Optimal Ensembles for Three- and Eight-State Prediction

3-state	
Variant	Ensemble
Probability voting	{MUFOLD, Nnessy, PSIPRED, SSpro}
Uniform voting Confidence selection	{Nnessy, Porter, SSpro}
Confidence voting Estimator voting Support selection	{MUFOLD, Nnessy, Porter, SSpro}
Estimator selection	{DeepCNF, MUFOLD, Nnessy, Porter, PSIPRED}
Hybrid selection	(Nnessy, Porter)

8-state	
Variant	Ensemble
Support selection	{DeepCNF, Nnessy, Porter, SSpro}
Probability voting	{DeepCNF, MUFOLD, Nnessy, SSpro}
Uniform voting	{Nnessy, Porter, SSpro}
Confidence voting Estimator voting Confidence selection Estimator selection	{Nnessy, Porter}
Hybrid selection	(Nnessy, Porter)

Highlighted in bold in each column is the variant achieving the highest accuracy on that specific dataset, as well as on average across all datasets.

To contrast variants that use *per-protein* selection versus *per-residue* voting, for 3-state prediction, estimator selection is better than every per-residue variant on all datasets except PDB2016. For 8-state prediction, hybrid selection is better than all per-residue variants on all datasets. That per-protein variants tend to outperform per-residue variants is likely due to the inherent noise at individual residues in the state probabilities, confidence values, and accuracy estimates used by per-residue voting, which may be getting dampened when aggregating across an entire protein with per-protein selection. The separation in prediction accuracy among variants that weight votes by uniform, confidence, and accuracy-estimate values is also more pronounced between per-protein variants than per-residue variants, and is further amplified in 8-state versus 3-state prediction.

Interestingly, probability voting—perhaps the most widely-used approach for ensemble prediction in the machine learning literature (equivalent to averaging class membership probabilities or class score vectors across the ensemble)—is the *worst ensemble variant* for 3-state prediction, averaged across datasets, and nearly the worst for 8-state prediction. (Even so, probability voting does outperform all individual tools.)

The new ensemble approach of hybrid selection exceeds probability voting on average by more than 2% in Q_3 accuracy, and more than 5% in Q_8 accuracy.

Composition of optimal ensembles

For both 3- and 8-state prediction, the best ensemble composition differs among ensemble variants. Unsurprisingly, per-residue voting and support variants suffer when low-accuracy predictions are added to the ensemble. For 3-state prediction, two of the lowest-accuracy methods, DeepCNF and JPred, are excluded from the best support and per-residue ensembles. This trend is mirrored in the 8-state ensembles as well, where the addition of DeepCNF and SSpro to the per-residue and support ensembles lowers their accuracy.

The best 3- and 8-state hybrid ensemble is a hybrid between only Nnessy and Porter, due to the estimator selection ensemble excluding Nnessy having lower accuracy than Porter alone, where the ensemble achieves 70.5% Q_8 (83.6% Q_3) accuracy compared to 71.2% Q_8 (84.0% Q_3) accuracy for Porter alone. This accuracy difference is probably due to the accuracy drop of most tools on CASP12 and CASP13, resulting in an overestimation of accuracy, which lowers the accuracy of any ensemble using accuracy estimation.

For 3-state selection ensembles, the estimator selection ensemble excludes JPred and the confidence selection ensemble excludes PSIPRED, JPred, and MUFOLD. The 8-state estimator selection and confidence selection ensembles exclude DeepCNF, SSpro and MUFOLD. For the confidence ensembles, the lack of normalization between tools may cause lower accuracy compared to the estimator ensembles, which fits a linear regression converting confidence to estimated accuracy.

Error in accuracy estimators

Hybrid selection and the estimator variants (whether per-protein or per-residue), use accuracy estimators, which take features from the output of the prediction tools to estimate the true accuracy of a prediction. If these accuracy estimators had zero error, they would achieve the oracle accuracy shown later (Section "Limit on ensemble accuracy"). The accuracy estimators have the following error:

	MUFOLD	PSIPRED	Porter	DeepCNF	JPred	Nnessy	SSpro
3-state	4.3 ± 5.2	4.4 ± 4.5	4.4 ± 5.0	4.8 ± 5.7	5.0 ± 4.5	5.4 ± 5.2	7.1 ± 7.0
8-state	6.0 ± 6.4	—	5.7 ± 6.0	6.1 ± 6.5	—	7.7 ± 7.8	11.0 ± 9.3

where the error measured is mean absolute error. Tools with lower variance in accuracy tend to have lower estimator error. Template-based tools like SSpro and Nnessy have high estimator error because of their high prediction accuracy variance caused by the presence and absence of template data for a prediction. Fig 1 shows a histogram of estimated accuracy minus true accuracy for each tool over the CASP and PDB datasets. These error values are roughly normally distributed. We present later (Section "Effect of improving accuracy estimators") a simulation of how estimator selection ensemble accuracy improves as estimator error decreases.

Discussion

We now discuss the behavior of our ensemble variants, their cardinality, and a limit on their accuracy. We also discuss how to construct an accuracy estimator and the theoretical effects of their improvement.

Visualizing ensemble behavior

To better understand the estimator selection variant and how it behaves, we visualized the ensemble method in two ways.

Fig 2 shows the proteins from the CASP datasets sorted by Q_3 accuracy of the estimator selection ensemble. The rank of the protein in this sorted order is given on the horizontal axis. On the vertical axis, the Q_3 accuracy of each individual tool is given, so that for each protein, a marker is present for each individual tool. The solid black curve intersects the prediction output by the estimator selection ensemble for each

[Figure is in file Fig1.tiff]

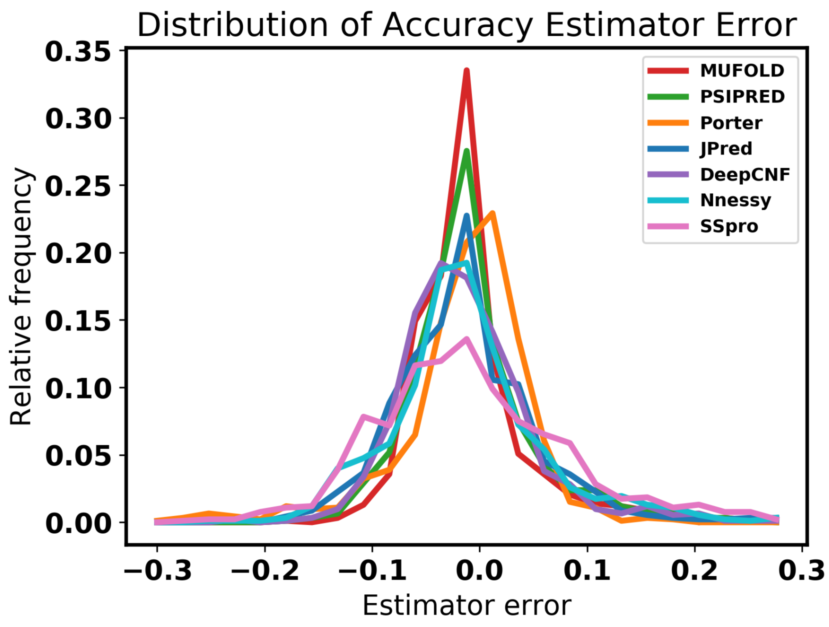


Fig 1. Distribution of error in three-state accuracy estimators. For each tool a histogram is plotted of the difference between their accuracy estimate and true accuracy, over the `CASP` and `PDB` datasets.

protein. The average accuracies of the oracle (a theoretical ensemble method where the highest-accuracy prediction is chosen for each protein), estimator selection ensemble, and the best tool (`Porter`) are given by the dashed and dotted horizontal lines. In this plot, any marker above the curve represents a suboptimal choice made by the ensemble method, caused by accuracy estimator error, where markers below the curve show predictions with lower accuracy than the ensemble. The high prediction accuracy variance of `Nnessy` and `SSpro` is visible by the spread of the green triangles and blue inverted triangles in the plot. The highest-accuracy predictions are almost exclusively from `Nnessy`, as shown by the black curve intersecting mostly green triangles for roughly the last 100 proteins.

To show the contribution of each tool to the estimator selection ensemble, we plotted histograms of each tool's true Q_8 accuracy on evaluation proteins and compared their histograms to the Q_8 accuracy of the ensemble. Fig 3 shows these histograms. Along the horizontal axis, the true Q_8 accuracy is given, where the vertical axis gives the relative frequency of proteins with that accuracy. The colored curves show the histograms of the individual tools and the thicker black curve shows the histogram of the estimator selection ensemble. The ensemble curve has peaks at roughly the same positions as `Porter` and `Nnessy`, showing that they contribute a significant amount of predictions to the ensemble.

Effect of ensemble size

To explore the effects of ensemble cardinality on ensemble accuracy, we plotted the accuracy of the ensemble variants as their cardinality increases. Fig 4 gives the accuracy of each ensemble on the vertical axis of a given cardinality on the horizontal axis. For each cardinality, all possible ensembles were considered and only the highest accuracy

[Figure is in file Fig2.tiff]

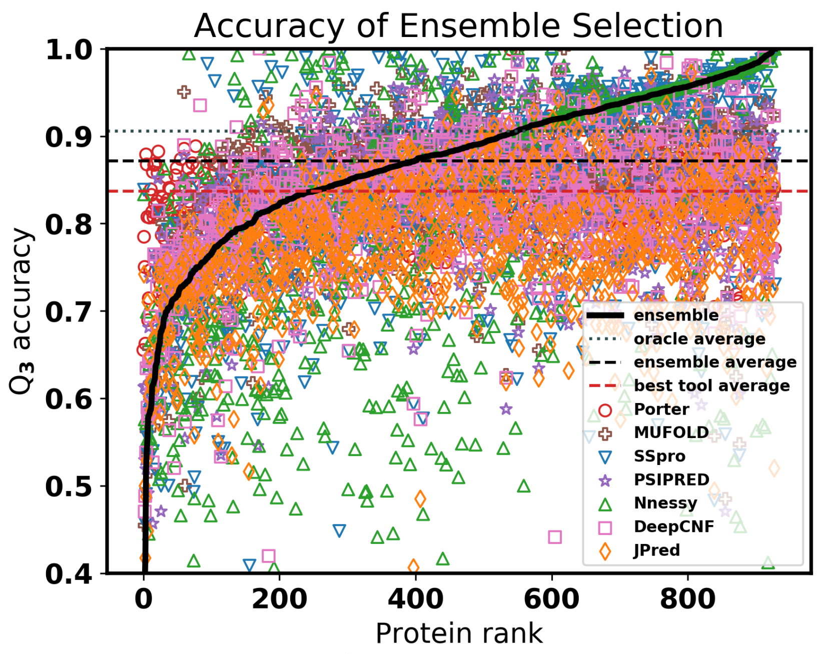


Fig 2. Three-state accuracy of ensemble selection. For `CASP` proteins ranked by ensemble prediction accuracy, the plot shows the Q_3 accuracy of the estimator selection ensemble and individual tools, as well as average accuracies for the oracle, ensemble, and best single tool, which is `Porter`.

[Figure is in file Fig3.tiff]

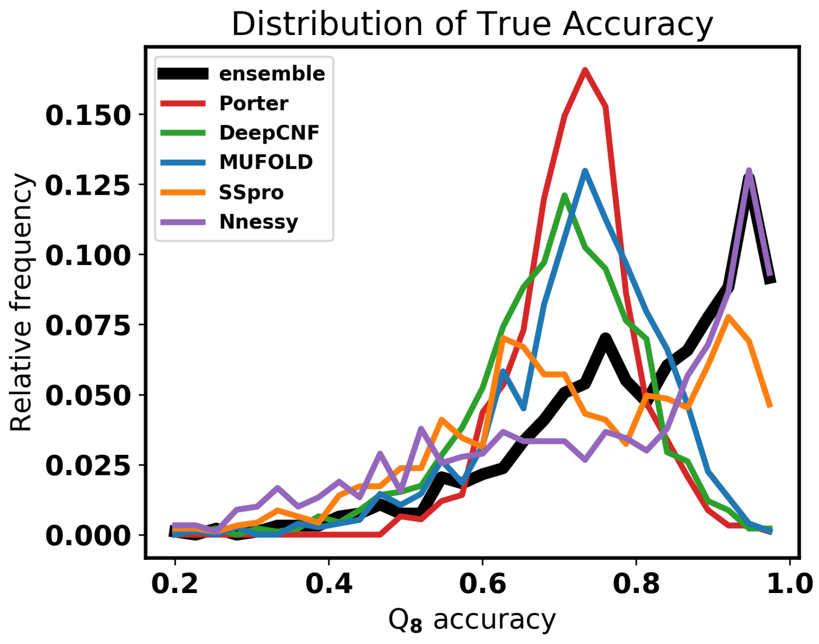


Fig 3. Distribution of eight-state accuracies. For each tool and the estimator selection ensemble, a histogram is plotted of their Q_8 accuracy, over the `CASP` and `PDB` datasets.

[Figure is in file Fig4.tiff]

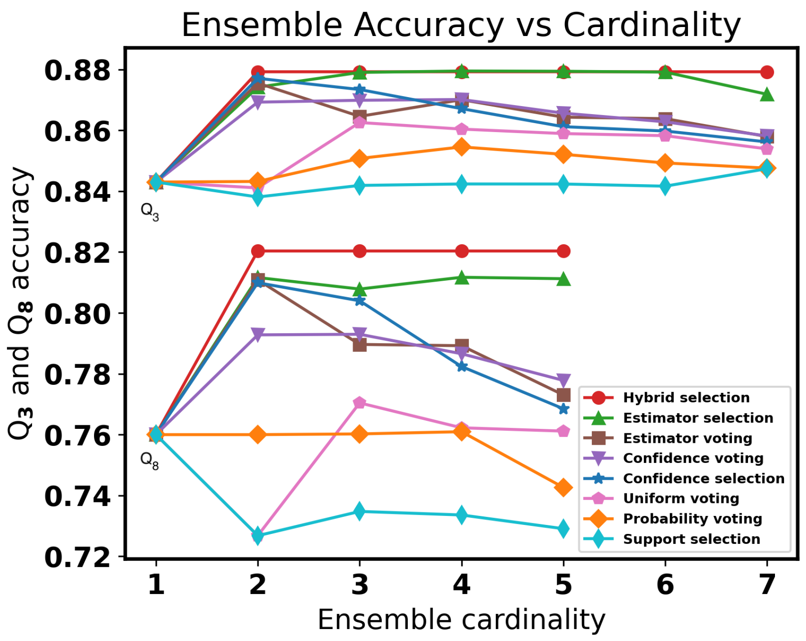


Fig 4. Accuracy of ensemble variants versus ensemble cardinality. The top curves show the average Q_3 accuracy of ensemble variants, while the bottom curves show their average Q_8 accuracy, over the `CASP` and `PDB` datasets.

for a given variant is shown. Unsurprisingly, for most ensemble variants, the tools were added to the highest-accuracy ensemble in order of average accuracy. Voting-based methods, especially uniform voting and probability voting, suffer from the addition of lower-accuracy predictions to the ensemble. This is reflected by the downward slopes of the curves after a certain cardinality (around 4 for Q_3 and around 3 for Q_8). Per-protein selection methods based on accuracy estimation tend to have more stable accuracies as more tools are added to the ensemble. With perfect accuracy estimators, adding more prediction methods to the ensemble could only improve the ensemble's accuracy, as lower-accuracy predictions would never be selected by the ensemble. The accuracy of uniform voting tends to improve on odd numbered cardinality and decline on even-numbered cardinalities probably due to fewer arbitrarily-broken ties of secondary structure states in ensembles with odd-numbered cardinality.

Limit on ensemble accuracy

We can define a conceptual *oracle* ensemble that always combines predictions in the most advantageous way using their true accuracy, which provides an empirical limit on the actual accuracy that can be attained by any real ensemble. The per-protein oracle accuracy for a `CASP` or `PDB` protein is the highest accuracy of any tool's prediction. For each protein the per-residue oracle outputs the fraction of residues for which at least one prediction method predicted the correct state. The following table gives these accuracies and the difference Δ between the oracle and the best ensemble variant (either per-protein or per-residue).

	CASP		PDB		average	
	(Oracle)	Δ	(Oracle)	Δ	(Oracle)	Δ
3-state, per-protein	89.9%	-2.9%	91.3%	-2.3%	90.6%	-2.7%
3-state, per-residue	96.1%	-10.8%	96.7%	-7.8%	96.4%	-9.4%
8-state, per-protein	82.2%	-3.8%	87.0%	-2.5%	84.6%	-3.1%
8-state, per-residue	88.9%	-13.4%	92.3%	-10.1%	90.6%	-11.7%

The per-residue oracles achieve much higher accuracy than the per-protein oracles, but surprisingly do not reach 100% accuracy for 3-state prediction, even though 7 tools were used in the oracle. For certain residues in the proteins from the CASP and PDB datasets, no tool predicts the correct secondary structure state.

Even though the two-tool hybrid of Nnessy and Porter outperforms the estimator selection ensemble, the oracle accuracy of the estimator selection ensemble far surpasses the oracle accuracy of the hybrid, which is further discussed later (Section "Effect of improving accuracy estimators"). The hybrid method actually approaches and in some cases achieves the two-tool oracle accuracy for certain CASP and PDB datasets [16]. As accuracy estimators improve, the estimator selection ensemble is likely to surpass the accuracy of hybrid selection.

Construction of accuracy estimators

To construct an accuracy estimator, we fit features from a prediction tool to true accuracy, as described earlier (Section "Fitting an estimator"). We mapped the confidence given by a tool (JPred and Porter) or calculated by us (Section "Choosing features") to true accuracy using linear regression. Because we fit only two parameters, the slope and intercept of the linear regression, we did not perform leave-one-out cross validation to fit this linear regression as omitting one of almost 1000 proteins would not have meaningfully changed the fit, so there was no danger of overfitting the estimator line. For the template-based tools Nnessy and SSpro, additional information from their template database searches were used. For SSpro, the average percent identity of template database searches was used. For Nnessy, we use the percent identity of the input protein to the nearest neighbors found in the template database, as explained in the next paragraph.

Fig 5 shows the two features, confidence and percent identity of template database words, used in Nnessy's accuracy estimator, along with their linear fit. For confidence (Fig 5a), the average protein confidence is mapped to an accuracy estimate using the shown piecewise-linear fit. For percent identity (Fig 5b), each residue's percent identity is mapped to an accuracy estimate, which is then averaged over the protein. The feature is given on the horizontal axis in both plots, and Nnessy's true Q_3 accuracy for that protein (confidence) or the 300 residues closest to that feature value (percent identity) is given on the vertical axis. In Fig 5a, each blue circle and green triangle respectively corresponds to a CASP or PDB protein. In Fig 5b, each circle represents a residue from the CASP and PDB proteins, but has been sparsified by a factor of 250. The dashed lines give the piecewise-linear fit, calculated by linear regression for each piece.

Fig 6 shows the true versus estimated accuracy using these two features independently and then combined. Each blue circle and green triangle represents a CASP or PDB protein, respectively, the same as Fig 5. Its estimated accuracy is given on the horizontal axis and its true Q_3 accuracy is given on the vertical axis. The identity (where true equals estimated accuracy) is plotted as a dotted line. The closer to the identity, the lower the error of the accuracy estimator for a protein.

Figs 5a and 6a combine to show the quality of confidence as an estimation feature. The flatter region in Fig 5a corresponds to the vertical swath of green triangles around 0.9 estimated accuracy in Fig 6a. This trend is also visible with the flatter section of Figs 5b and 6b.

When the two features are combined in Fig 6c, the noise present using only confidence or percent identity cancels out for many of the points, resulting in an accuracy estimator with less error. This approach could be employed for the accuracy estimators for other tools by using some feature of the output of their neural network architectures to estimate accuracy.

Effect of improving accuracy estimators

To quantify the effect of improving the accuracy estimators on the estimator selection ensemble's accuracy, we simulated decreasing accuracy estimator error. We fit a normal distribution over the error values for each prediction method to get a mean and standard deviation of the error in each method. We changed the mean to 0 and for each prediction, computed a simulated estimated accuracy by sampling error randomly from the fitted normal distribution and adding that to the true accuracy of the prediction. This gives us an accuracy estimate with the expected error for a given accuracy estimator We then decreased the standard deviation of the distribution in 10% increments, simulating a better accuracy estimator, until the standard deviation was 0 (giving the oracle accuracy), shown in Fig 7. We also show the effect of increasing the standard deviation, simulating the accuracy estimator worsening. The dashed line toward the top of the plot is the ensemble oracle accuracy, where the dashed line near the bottom is the `Nnessy-Porter` hybrid oracle accuracy (the most-accurate ensemble variant). Each data point is shown as a box spanning the 1st and 3rd quartiles of the 10 simulations that were performed with the given standard deviation ratio, where the top and bottom lines give the maximum and minimum values from the runs. This shows the sensitivity of the hybrid and estimator selection ensembles to accuracy estimator error.

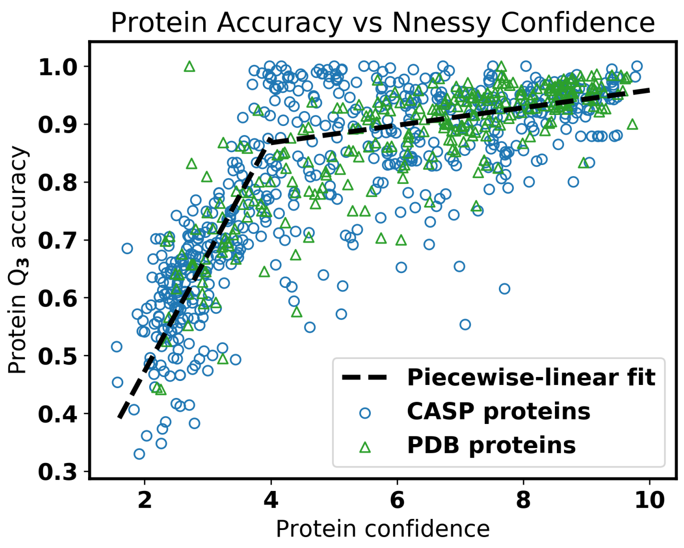
The simulation assumes that the mean estimated error is 0, which is close to the actual mean error of the accuracy estimator for each tool (these histograms can be seen in Fig 1). We also assume that the error is uncorrelated to the accuracy of the prediction. If this assumption were true, then simulating the observed distribution should give us the true ensemble accuracy, which is not the case. Regardless, this simulation is useful to show how quickly we expect the true ensemble accuracy to change as accuracy estimators improve.

In practice, incorporating the confidence feature in addition to the similarity feature in `Nnessy`'s accuracy estimator gave a 19% reduction in the standard deviation of the normal fit to the error. This trend is likely to also hold for accuracy estimators of other tools.

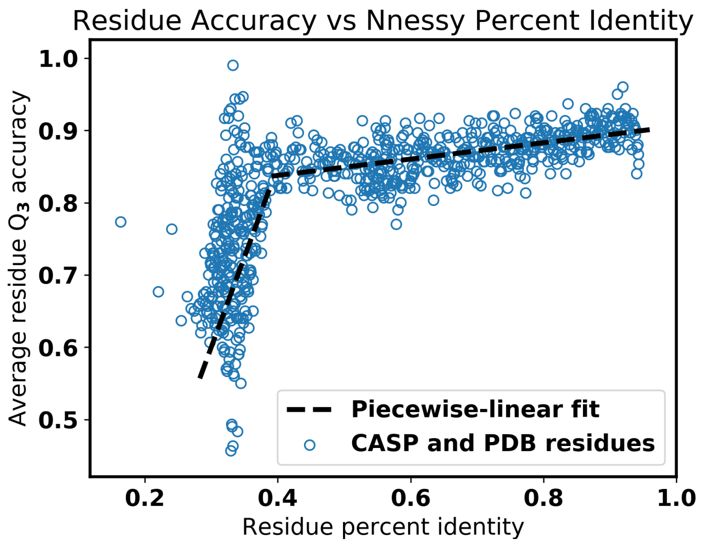
Summary

We have presented several new approaches to ensemble protein secondary structure prediction that significantly exceed the *state-of-the-art accuracy* for 3- and 8-state structure prediction. These approaches leverage feature-based *accuracy estimation* in a variety of ways to select a prediction from the ensemble of high estimated accuracy. Surprisingly, the approach of *hybrid selection*, which is one of the simplest, is also among the most accurate, while using an ensemble with only two methods (and requiring an estimator for only one). Simulations suggest that an alternate approach of *estimator selection*, which theoretically has greater room for improvement (though it requires accuracy estimators for all methods), has the potential—through future discovery of new features that further reduce estimator error—to eventually surpass hybrid selection, and could become the approach of choice.

[Figure is in file Fig5a.tiff]



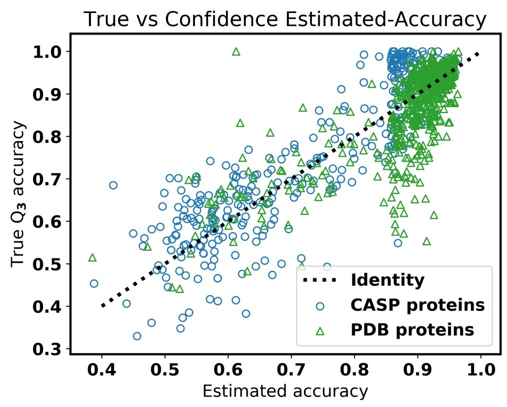
(a) Protein accuracy versus `Nnessy` confidence.
[Figure is in file Fig5b.tiff]



(b) Residue accuracy versus `Nnessy` percent identity.

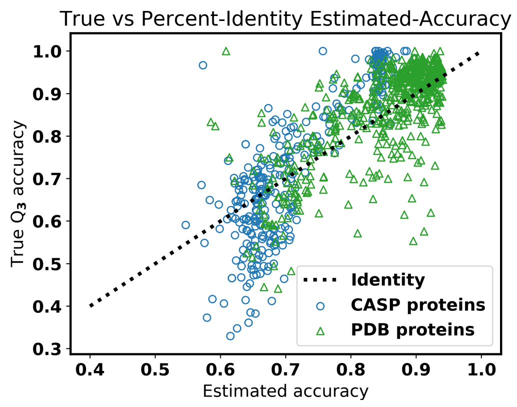
Fig 5. True accuracy versus `Nnessy` estimator features. The scatterplots show Q_3 accuracy versus `Nnessy` feature values for (a) protein confidence, and (b) residue percent identity. The dashed lines show the initial accuracy estimator for each feature.

[Figure is in file Fig6a.tiff]



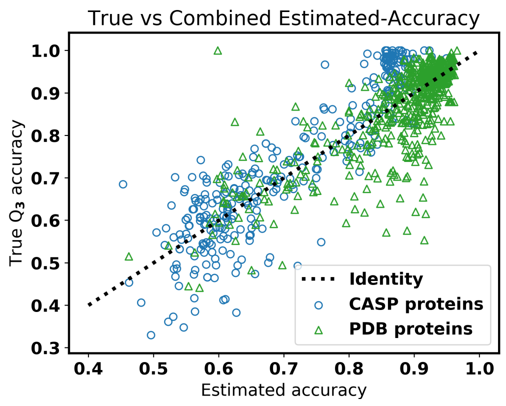
(a) Initial estimator using confidence.

[Figure is in file Fig6b.tiff]



(b) Initial estimator using percent identity.

[Figure is in file Fig6c.tiff]



(c) Final combined estimator.

Fig 6. True versus estimated accuracy of `Nnessy` estimators. The scatterplots show Q_3 accuracy versus estimated accuracy over `CASP` and `PDB` proteins, for the estimators using (a) protein confidence, (b) residue percent identity, and (c) a combination of the two features. For reference, the line is the identity function, where true and estimated accuracy are equal.

[Figure is in file Fig7.tiff]

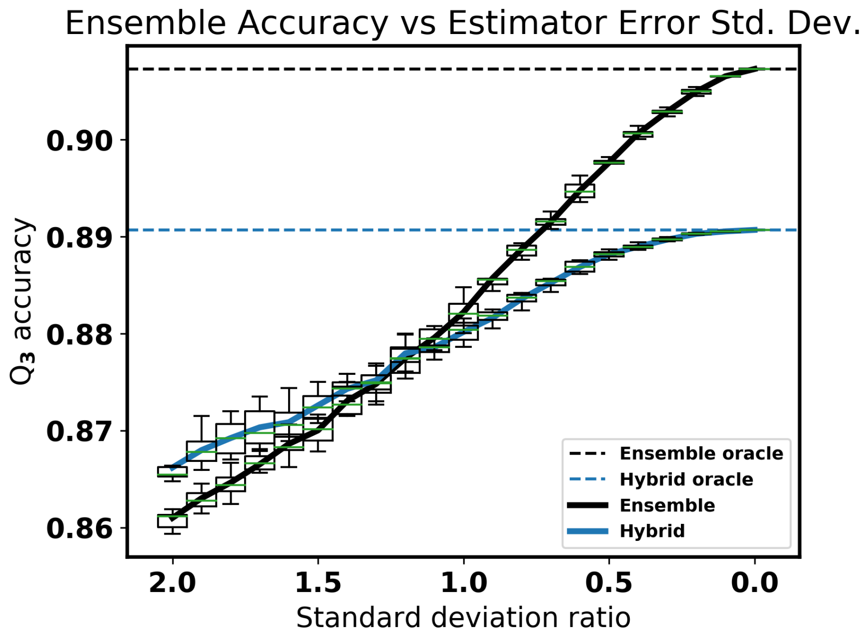


Fig 7. Simulated accuracy of the hybrid and estimator selection ensembles. The curves show the Q_3 accuracy of these variants when the standard deviation of accuracy estimator error is fixed at a given ratio.

Further research

We highlight a few promising directions for further research. As mentioned earlier, *hybrid selection* can be generalized to a recursive process where the alternate method again employs hybrid selection, now on a smaller ensemble, leading to a kind of decision tree that thresholds on accuracy estimates of the core methods. Recursive hybrid selection could be well-suited for an ensemble setting with independent tools that have high variance in true accuracy, yet moderate error in their accuracy estimators (where simply picking the prediction of highest estimated accuracy might not perform as well due to estimator error). Since *estimator selection* tends to behave robustly as predictors are added, it is ideal for ensembles that expand to incorporate future tools as new ones are developed. In general, the applicability of estimator selection is extremely broad, as it yields a natural ensemble method for any prediction task where accuracy estimators can be constructed.

Acknowledgments

This paper is an extended journal version of a prior conference paper [45].

Author contributions

Conceptualization Spencer Krieger and John Kececioglu
Data curation Spencer Krieger
Analysis Spencer Krieger and John Kececioglu
Investigation Spencer Krieger
Methodology Spencer Krieger and John Kececioglu

Software Spencer Krieger
Supervision John Kececioglu
Visualization Spencer Krieger and John Kececioglu
Writing Spencer Krieger and John Kececioglu

Competing interests The authors declare that no competing interests exist.

References

1. Dill KA, MacCallum JL. The protein-folding problem, 50 years on. Science. 2012;338(6110):1042–1046.
2. Adamczak R, Porollo A, Meller J. Accurate prediction of solvent accessibility using neural networks-based regression. Proteins. 2004;56(4):753–767.
3. Lu Y, Sze SH. Multiple sequence alignment based on profile alignment of intermediate sequences. Journal of Computational Biology. 2008;15(7):767–777.
4. Kececioglu J, Kim E, Wheeler T. Aligning protein sequences with predicted secondary structure. Journal of Computational Biology. 2010;17(3):561–580.
5. Kabsch W, Sander C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers. 1983;22:2577–2637.
6. Yang Y, Gao J, Wang J, Heffernan R, Hanson J, Paliwal K, et al. Sixty-five years of the long march in protein secondary structure prediction: the final stretch? Briefings in Bioinformatics. 2018;19(3):482–494.
7. Jones D. Protein secondary structure prediction based on position-specific scoring matrices. Journal of Molecular Biology. 1999;292:195–202.
8. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, et al. Gapped `BLAST` and `PSI-BLAST`: a new generation of protein database search programs. Nucleic Acids Research. 1997;25(17):3389–3402.
9. Drozdetskiy A, Cole C, Procter J, Barton GJ. `JPred4`: a protein secondary structure prediction server. Nucleic Acids Research. 2015;43(W1):W389–W394.
10. Finn RD, Clements J, Eddy SR. `HMMER` web server: interactive sequence similarity searching. Nucleic Acids Research. 2011;39(suppl2):W29–W37.
11. Pollastri G, Przybylski D, Rost B, Baldi P. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. Proteins: Structure, Function, and Bioinformatics. 2002;47(2):228–235.
12. Mirabello C, Pollastri G. `Porter`, `PaleAle` 4.0: high-accuracy prediction of protein secondary structure and relative solvent accessibility. Bioinformatics. 2013;29(16):2056–2058.
13. Wang S, Peng J, Ma J, Xu J. Protein secondary structure prediction using deep convolutional neural fields. Scientific Reports. 2016;6(18962).
14. Fang C, Shang Y, Xu D. `MUFOLD-SS`: new deep inception-inside-inception networks for protein secondary structure prediction. Proteins: Structure, Function, and Bioinformatics. 2018;86(5):592–598.
15. Ma Y, Liu Y, Cheng J. Protein secondary structure prediction based on data partition and semi-random subspace method. Scientific Reports. 2018;8(9856).
16. Krieger S, Kececioglu J. Boosting the accuracy of protein secondary structure prediction through nearest neighbor search and method hybridization. Bioinformatics, Proceedings of 28th ISCB Conference on Intelligent Systems for Molecular Biology (ISMB). 2020;36(Suppl_1):i317–i329.
17. Krieger S, Kececioglu J. `Nnessy`: nearest-neighbor-based secondary structure prediction without searching for homology, version 1.0.2; 2020. `http://nnessy.cs.arizona.edu`.

18. Zhou Z. Ensemble Methods: Foundations and Algorithms. Chapman and Hall; 2012.

19. Liu KH, Xia JF, Li X. Efficient ensemble schemes for protein secondary structure prediction. Protein and Peptide Letters. 2008;15(5):488–493.

20. Lin L, Yang S, Zuo R. Protein secondary structure prediction based on multi-SVM ensemble. In: Proceedings of IEEE International Conference on Intelligent Control and Information Processing (ICICIP); 2010. p. 356–358.

21. Yang Li, Shibuya T. `Malphite`: a convolutional neural network and ensemble learning based protein secondary structure predictor. In: Proceedings of IEEE International Conference on Bioinformatics and Biomedicine (BIBM); 2015. p. 1260–1266.

22. Hanson J, Paliwal K, Litfin T, Yang Y, Zhou Y. Improving prediction of protein secondary structure, backbone angles, solvent accessibility and contact numbers by using predicted contact maps and an ensemble of recurrent and residual convolutional neural networks. Bioinformatics. 2018;35(14):2403–2410.

23. Hu H, Li Z, Elofsson A, Xie S. A Bi-LSTM based ensemble algorithm for prediction of protein secondary structure. Applied Sciences. 2019;9(17):3538.

24. Xu G, Wang Q, Ma J. `OPUS-TASS`: a protein backbone torsion angles and secondary structure predictor based on ensemble neural networks. Bioinformatics. 2020;36(20):5021–5026.

25. Guermeur Y, Geourjon C, Gallinari P, Deléage G. Improved performance in protein secondary structure prediction by inhomogeneous score combination. Bioinformatics. 1999;15(5):413–421.

26. Selbig J, Mevissen T, Lengauer T. Decision-tree based formation of consensus protein secondary structure prediction. Bioinformatics. 1999;15(12):1039–1046.

27. Guermeur Y, Pollastri G, Elisseeff A, Zelus D, Paugam-Moisy H, Baldi P. Combining protein secondary structure prediction models with ensemble methods of optimal complexity. Neurocomputing. 2004;56:305–327.

28. Bouziane H, Messabih B, Chouarfia A. Profiles and majority voting-based ensemble method for protein secondary structure prediction. Evolutionary Bioinformatics. 2011;7:171–189.

29. Bouziane H, Messabih B, Chouarfia A. Effect of simple ensemble methods on protein secondary structure prediction. Soft Computing. 2015;19:1663–1678.

30. Aydin Z, Uzut OG. Combining classifiers for protein secondary structure prediction. In: Proceedings of 9th IEEE International Conference on Computational Intelligence and Communication Networks (CICN); 2017. p. 29–33.

31. Ouali M, King RD. Cascaded multiple classifiers for secondary structure prediction. Protein Science. 2000;9(6):1162–1176.

32. Liu Y, Carbonell J, Klein-Seetharaman J, Gopalakrishnan V. Comparison of probabilistic combination methods for protein secondary structure prediction. Bioinformatics. 2004;20(17):3099–3107.

33. DeBlasio DF, Wheeler TJ, Kececioglu JD. Estimating the accuracy of multiple alignments and its use in parameter advising. In: Proceedings of 16th Conference on Research in Computational Molecular Biology (RECOMB); 2012. p. 45–59.

34. Kececioglu J, DeBlasio D. Accuracy estimation and parameter advising for protein multiple sequence alignment. Journal of Computational Biology. 2013;20(4):259–279.

35. DeBlasio D, Kececioglu J. Ensemble multiple sequence alignment via advising. In: Proceedings of 6th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB); 2015. p. 452–461.

36. DeBlasio D, Kececioglu J. Parameter Advising for Multiple Sequence Alignment. vol. 26 of Computational Biology Series. Springer International; 2017.

37. DeBlasio D, Kececioglu J. `Facet`: feature-based accuracy estimation of protein multiple sequence alignments; 2014. `http://facet.cs.arizona.edu`.

38. Moult J, Fidelis K, Kryshtafovych A, Schwede T, Tramontano A. Critical assessment of methods of protein structure prediction (CASP) round X. Proteins: Structure, Function, and Bioinformatics. 2014;82:1–6.

39. Moult J, Fidelis K, Kryshtafovych A, Schwede T, Tramontano A. Critical assessment of methods of protein structure prediction: progress and new directions in round XI. Proteins: Structure, Function, and Bioinformatics. 2016;84:4–14.

40. Moult J, Fidelis K, Kryshtafovych A, Schwede T, Tramontano A. Critical assessment of methods of protein structure prediction (CASP) round XII. Proteins: Structure, Function, and Bioinformatics. 2018;86:7–15.

41. Kryshtafovych A, Schwede T, Topf M, Fidelis K, Moult J. Critical assessment of methods of protein structure prediction (CASP) round XIII. Proteins: Structure, Function, and Bioinformatics. 2019;87(12):1011–1020.

42. Berman H, Westbrook J, Feng Z, Gilliland G, Bhat T, Weissig H, et al. The Protein Data Bank. Nucleic Acids Research. 2000;28(1):235–242.

43. Krieger S, Kececioglu J. `Ssylla`: protein secondary structure prediction by an ensemble leveraging accuracy estimation, version 1.1; 2021. `http://ssylla.cs.arizona.edu`.

44. 14th Community Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction (`CASP14`); 2020. `https://predictioncenter.org/casp14/`.

45. Krieger S, Kececioglu J. Predicting protein secondary structure by an ensemble through feature-based accuracy estimation. In: Proceedings of 11th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB). 29; 2020. p. 1–10.