A unified multitask architecture for predicting local protein properties

Yanjun Qi[1], Merja Oja [2*], Jason Weston[3], Wiliam Stafford Noble[2,*]
1 Machine Learning Department, NEC Labs America, Princeton, NJ, USA
2 Department of Genome Sciences, University of Washington, Seattle, WA, USA
3 Google, New York City, NY, USA
∗ E-mail: noble@gs.washington.edu

Abstract

A variety of functionally important protein properties, such as secondary structure, transmembrane topology and solvent accessibility, can be encoded as a labeling of amino acids. Indeed, the prediction of such properties from the primary amino acid sequence is one of the core projects of computational biology. Accordingly, a panoply of approaches have been developed for predicting such properties; however, most such approaches focus on solving a single task at a time. Motivated by recent, successful work in natural language processing, we propose to use *multitask learning* to train a single, joint model that exploits the dependencies among these various labeling tasks.

We describe a deep neural network architecture that, given a protein sequence, outputs a host of predicted local properties, including secondary structure, solvent accesibility, transmembrane topology, signal peptides and DNA-binding residues. The network is trained jointly on all these tasks in a supervised fashion, augmented with a novel form of semi-supervised learning in which the model is trained to distinguish between local patterns from natural and synthetic protein sequences. The task-independent architecture of the network obviates the need for task-specific feature engineering. We demonstrate that, for all of the tasks that we considered, our approach leads to statistically significant improvements in performance, relative to a single task neural network approach, and that the resulting model achieves state-of-the-art performance.

Introduction

Proteins participate in every major biological process within every living cell. Therefore, elucidating protein function is a central endeavor of molecular biology. In this work, we focus on predicting local functional properties, which can be summarized as a labeling of amino acids. Many important functional properties can be described in this fashion, including secondary structure, solvent accesibility, transmembrane topology and the locations of signal peptides, DNA-binding residues, protein-binding residues and coiled-coil regions.

Our work is motivated, in part, by recent, successful work in the field of natural language processing [1]. Analogous to functional labeling of amino acids, natural language can be annotated with tags indicating synonymous pairs of words, parts of speech, larger syntactic entities, named entities, semantic roles, etc. These labelings exhibit strong dependencies across tasks. Accordingly, Collobert and Weston [1] demonstrated that a unified neural network architecture, trained simultaneously on a collection of related tasks, provides more accurate labelings than a network trained only on a single task. Their study thus demonstrates the power of *multitask learning*, which has been the subject of much recent work in machine learning [2]. Furthermore, essential to the success of the Collobert and Weston system is the use of a *deep neural network* [3, 4], which is able to learn a hierarchy of features that are relevant to the tasks at hand given very basic inputs. The deep network employs different layers to represent cross-task and task-specific information. Thus, the deep multitask architecture makes it possible to avoid the challenging process of incorporating hand-engineered features specific to each task. Finally, a critical piece of the

[*]Currently at VTT Technical Research Centre of Finland, Espoo, Finland

Collobert and Weston methodology is the use of a so-called *language modeling* task [1], in which the network learns to discriminate between genuine natural language sentences and synthetically generated sentences. Our work makes use of all three of these components—multitask learning, deep learning and an analog of the language model—to predict local protein properties.

Currently, most approaches to predict local protein properties focus on one task at a time. Perhaps the most well-studied such problems are the canonical secondary structure prediction problem [5] and the related task of predicting transmembrane protein topology [6]. Other tasks include identifying signal peptides, predicting DNA-binding residues, identifying coiled-coil regions, predicting relative or absolute solvent accessibility, etc. However, like the natural language processing tasks mentioned above, all of these protein labeling tasks exhibit strong inter-task dependencies. For example, transmembrane protein topology and secondary structure prediction are closely linked, because most transmembrane protein segments are alpha helices. Similarly, signal peptide prediction can be viewed as prediction of a particular type of transmembrane segment. DNA-binding and protein-binding residues may also share similar structural features, since both must be exposed on the surface of the protein. Some work makes use of these dependencies in a pipelined fashion; for example, [7] use secondary structure and solvent accessibility predictions as input to a DNA-binding residue predictor. A drawback to the pipelining approach is that errors from one classifier get propagated to downstream classifiers [8]. A more elegant and robust approach is to use only the primary amino acid inputs, combine the prediction problems, and learn the tasks simultaneously using multitask learning. This strategy also avoids extracting task-specific sets of features.

In this work, we define a unified architecture for prediction of local protein properties by training a deep neural network in a multitask fashion. Specifically, our network learns simultaneously to predict solvent accessibility, transmembrane topology and two secondary structure alphabets, and to identify DNA-binding residues, protein-binding residues, signal peptides and coiled-coil regions. We also include a semi-supervised learning task, which we call *natural protein modeling*, to learn features representing local amino acid patterns in naturally occurring protein sequences. We evaluate this architecture using benchmark datasets for each task. The results show that multitask learning improves performance on nearly every task, and adding the semi-supervised natural task helps in nearly every case. Furthermore, adding both multitask learning and the natural protein task makes our architecture achieve state-of-art performance on almost all tasks.

Methods

Prediction tasks

A variety of local protein properties can be represented as predicting labels of amino acids. In this work, we predict ten such labelings, each of which is described below. Specific details regarding the data sets employed in this study are provided in the supplement.

Secondary structure A protein's *secondary structure* is a useful intermediate between the relatively easy-to-ascertain primary amino acid sequence and the difficult-to-obtain three-dimensional structure. The secondary structure specifies the general three-dimensional form of local segments of a protein. The most commonly observed local structures are α-helices, β-sheets and loops. In the secondary structure prediction task we aim to predict each residue's secondary structure label. Knowing a protein's secondary structure may yield insight into the protein's functional class, suggest boundaries between domains, or aid in inferring the protein's 3D structure.

Since the advent of the first automated secondary structure prediction method 22 years ago [5], dozens of subsequent methods have been described in the scientific literature. These include methods that employ neural networks [5, 9–13] and probability models such as hidden Markov models [14] and dynamic Bayesian networks [15].

Transmembrane topology and signal peptide prediction Complementary to prediction of protein secondary structure is the prediction of transmembrane topology. The most common type of transmembrane protein consists of a series of α-helices that span the membrane, interleaved with loops that extend out of the membrane. The labeling task consists of identifying these membrane-spanning segments and then specifying whether each loop is inside or outside of the membrane; hence, transmembrane predictors employ a three-letter alphabet.

Transmembrane proteins are of particular interest for two reasons. First, because transmembrane proteins cannot be crystallized (due to the presence of the membrane), their 3D structure cannot be easily determined. Second, because of their importance in communicating across membranes, more than half of all drug targets are transmembrane proteins, even though fewer than 10% of all proteins are transmembrane proteins.

A closely related task to the prediction of transmembrane protein topology is the prediction of signal peptides, which are short (3–60 amino acids) peptide chains that direct newly translated proteins to their final destinations in the cell. Early methods predicted signal peptides [16, 17] and transmembrane protein topology separately [6]. More recent work suggests that the two tasks can be solved more effectively using a joint predictor [18, 19].

Solvent accessibility The solvent accesibility prediction task involves distinguishing between amino acids that are accessible to water versus amino acids that are buried inside the protein. Hence, the task involves a two-letter alphabet. Defining this alphabet requires setting a threshold, which can be done either on an absolute scale or relative to the protein in which the amino acid resides.

Early methods for solvent accessibility prediction used neural networks [10, 20, 21] or support vector machines [22]. [23] showed that a simple baseline predictor performs rather well relative to more sophisticated methods, and [24] made a consensus predictor by combining predictions from three different methods.

Coiled coil regions A coiled coil is a protein structural motif, in which α-helices are coiled together like the strands of a rope. Coiled coils usually contain a repeated pattern, `hpphppp`, of hydrophobic (`h`) and polar (`p`) amino-acid residues, referred to as a heptad repeat. The positions in the heptad repeat are usually labeled `abcdefg`, where `a` and `d` are the hydrophobic positions, often being occupied by isoleucine, leucine or valine. Folding a sequence with this repeating pattern into an alpha-helical secondary structure causes the hydrophobic residues to be presented as a "stripe" that coils gently around the helix in left-handed fashion, forming an amphipathic structure. Coiled coil domains function in the stabilization of tertriary and quaternary structure of proteins. Many coiled coil proteins are involved in protein-protein interactions and have important biological functions, such as protein trafficking, signalling and regulation of gene expression.

The first method for predicting coiled coil regions used position specific scoring matrices to score sequence windows [25]. Subsequent methods achieved improved accuracy by including correlations among residues [26–28] or by using hidden Markov models [29]. Currently, the best performing coiled coil predictor is an HMM that uses evolutionary information [30].

DNA binding The final two prediction tasks involve identifying amino acids that interact with, respectively, DNA molecules or other proteins. Detection of DNA-binding sites in proteins is critical for targeting gene regulation and manipulation. Thousands of proteins are known to bind to DNA; however, for most of these proteins the mechanism of action and the residues that bind to DNA, i.e. the binding sites, are not known. Experimental identification of binding sites requires expensive and laborious methods such as mutagenesis and binding assays. If the 3D structure of a protein is known, then it is often possible to predict DNA-binding sites *in silico*. However, for most proteins, such knowledge is not available.

Several methods have been developed to predict DNA-binding residues from the primary amino acid sequence. [31] described predictors based on sequence composition and predicted solvent accessibility. Later, some of the same authors [32] used profiles of homologous proteins to achieve more accurate prediction of DNA-binding sites. More recently, [7] used three types of inputs—PSI-BLAST profiles, predicted secondary structure, and predicted solvent accessibility—to train a support vector machine DNA-binding predictor.

Protein binding residues The protein-binding prediction task is analogous to the prediction of DNA-binding residues, but focused on binding sites for protein-protein interactions rather than protein-DNA interactions. Identifying these sites from the primary amino acid sequence is critical to understanding protein function, because so many proteins carry out their functions as part of multi-protein complexes.

Several studies attempted to address the sequence-based interaction site prediction problem. Pazos et al. [33] use multiple sequence alignment to detect correlated changes in a group of interacting protein domains for predicting contacting pairs of protein residues. Gallet et al. [34] analyze hydrophobicity patterns and amino acid distributions in known interaction sites to identify linear stretches of sequences. Yan et al. [35] apply support vector machines to predict protein binding sites with features extracted from sequence neighbors for each target residue. Liang et al. [36] predict interface residues using an empirical score function that is a linear combination of the energy score, interface propensity and residue conservation score. And Ofran et al. [37] employ a neural network approach using PSI-BLAST profile features to identify interaction sites directly from sequences.

Deep neural network for each task

We introduce a deep neural network architecture for protein labeling tasks. The input sequence is fed through several layers of feature extraction, and features relevant to the task are learned automatically by backpropagation in deep layers of the network. The general deep network architecture, which is suitable for all our prediction tasks, is summarized in Figure 1. The network is characterized by two specialized layers—an amino acid feature extraction layer and a sequential feature extraction layer—followed by a series of classical neural network layers.

Amino acid feature extraction The first layer extracts features for each amino acid by mapping amino acids to real-valued vectors. We use two distinct types of features—a learned embedding into a d-dimensional feature space, and a 20-dimensional feature representation produced by PSI-BLAST [38]—which are concatenated to produce the output of the layer.

The first feature extraction module projects each amino acid into a d-dimensional feature space, where d is a hyperparameter; i.e., d is not subject to optimization. For simplicity and efficiency, this layer treats amino acids as indices into a finite dictionary $\mathcal{D}$. Each amino acid $i \in \mathcal{D}$ is then embedded into the feature space using a $d \times |\mathcal{D}|$ lookup table W, such that the ith column of W (denoted W_i) is the vector corresponding to amino acid i. Thus, in the first layer of our architecture an input sequence $\{s_1, s_2, \ldots s_n\}$ is transformed into a series of real valued vectors $\{W_{s_1}, W_{s_2}, \ldots W_{s_n}\}$. The parameters of the lookup table W are learned automatically as part of the neural network training. This type of feature extraction—called a "local encoding of amino acids"—was originally proposed by Riis et al. [39]. The encoding weights W are randomly initialized and then learned by a generalization of back-propagation. The resulting encoding is optimal in the sense that it optimizes the objective cost on the training set for the specific network and the specific task.

The second feature extraction module in the first layer extracts information from an alignment of homologous proteins identified by the PSI-BLAST algorithm. Each length-M query sequence is searched using PSI-BLAST against the NCBI nonredundant protein sequence database, yielding a $20 \times M$ position-specific scoring matrix. Each element in the matrix represents the log-likelihood of a particular residue substitution at that position in the template. The profile matrix elements (typically in the range $[-7, 7]$) are scaled to the required range $[0, 1]$ by using the following scaling function [40]:

$$f(x) = \begin{cases} 0.0 & \text{if } x \leq -5 \\ 0.5 + 0.1x & \text{if } -5 < x < 5 \\ 1.0 & \text{if } x \geq 5 \end{cases} \tag{1}$$

where x is the value from profile matrix. Each column of the rescaled matrix comprises a 20-dimensional PSI-BLAST feature vector for the corresponding amino acid.

Sequential feature extraction layer To facilitate identification of local sequence structure, the second layer performs a sliding window operation on the sequence. As illustrated in Figure 1, the second layer aggregates the output of the first layer into blocks corresponding to a fixed window size k. Figure 1 shows an example using a window size of $k = 7$, so each block contains information about the current amino acid as well as the three flanking amino acids on either side. Altogether, because the output of the first layer has $d + 20$ dimensions, the output of the second layer has $k(d + 20)$ dimensions.

Classical neural network layers The remaining layers comprise a standard, fully connected multi-layer perceptron network with L layers of hidden units. Each hidden layer learns to map its input to a hidden feature space, and the last output layer then learns the mapping from the hidden space to the output class label space. In Figure 1, the sequential feature extraction layer induces a large effective feature space, where examples correspond to each possible length-k sequence of amino acids. Thus, the job of the hidden layers is to map this high dimensional input space to a lower dimensional feature space and then to look for hyperplanes that separate examples with different amino acid labels in the output layer.

In practice, the output of the ℓ^{th} layer $\boldsymbol{o}^{\ell}$ containing h_ℓ hidden units is computed with

$$\boldsymbol{o}^{\boldsymbol{\ell}} = \text{HardTanh}(\boldsymbol{L}^{\boldsymbol{\ell}} \cdot \boldsymbol{o}^{\boldsymbol{\ell-1}}) \tag{2}$$

where the matrix of parameters $\boldsymbol{L}^{\boldsymbol{\ell}} \in \mathbb{R}^{\mathrm{h}_\ell \times \mathrm{h}_{\ell-1}}$ is trained by backpropagation. The transfer function is defined as:

$$\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases} . \tag{3}$$

The size of the last (parametric linear) layer's output $\boldsymbol{o}^L$ is the number of classes considered in the prediction task. This layer is followed by a softmax layer which ensures that the outputs are positive and sum to 1, allowing us to interpret the outputs of the neural network as probabilities for each class. The i^{th} output is given by

$$\tilde{P}_i = \frac{exp\{o_i^L\}}{\sum_j exp\{o_j^L\}} \tag{4}$$

In this work, for simplicity, we restrict the classical neural network to one single hidden linear layer and one ouput linear layer.

Assuming that we are given a set of training examples $\{(\boldsymbol{x}_n, y_n)\}_{n=1...N}$, where $\boldsymbol{x}_n$ represents a local short window of amino acids and y_n a label, the whole network is then trained to minimize the negative log-likelihood, i.e. $E(\Theta) = \sum_{n=1}^{N} E(\Theta, \boldsymbol{x}_n, y_n)$ over the data with respect to Θ: all parameters of the network. Specifically,

$$E(\Theta) = -\sum_{n=1}^{N} logP_\theta(y_n|\boldsymbol{x}_n) = -\sum_{n=1}^{N} log\tilde{P}_{\theta,y_n}(\boldsymbol{x}_n) \tag{5}$$

Stochastic gradient descent optimization is used for the above training. Random examples $(\boldsymbol{x}, y)$ are sampled from the training set and then a gradient descent step is applied to update network parameter Θ as follows:

$$\Theta \leftarrow \Theta - \eta \frac{\partial E(\Theta, \boldsymbol{x}, y)}{\partial \Theta} \tag{6}$$

where η is a learning rate parameter.

Multitasking with weight sharing of deep neural networks

The neural network architecture displayed in Figure 1 can be adapted in various ways to perform multitask learning. In this work, we used the multitask architecture shown in Figure 2, in which the top-most layers of the network are shared across multiple tasks, and only the very last layers of the network are task specific.

Assuming we have T related tasks, the “weight sharing” strategy implies that the parameters for the top-most layers of the network are shared between tasks; i.e., the network includes parameters

$$\Theta_t = \{W, \boldsymbol{L^1}, \boldsymbol{L^2}, ..., \boldsymbol{L^{L-1}}, \boldsymbol{L_t^L}\} \tag{7}$$

for each task t. With this setup, the neural network automatically learns an embedding that generalizes across tasks in the first layers of the network, and learns features specific for the desired tasks in the deep layers of the network.

Training in the multitask setting is accomplished by minimizing an objective function that is the sum of the objectives from each task, where each task is given equal weight. That is, we optimize:

$$\sum_{t=1}^{T} \sum_{n_t=1}^{N_t} E_t(\Theta_t, \boldsymbol{x}_{n_t}, y_{n_t}) \tag{8}$$

assuming each task has a training set $\{(\boldsymbol{x}_{n_t}, y_{n_t})\}_{n_t=1...N_t}$. The tasks share a common feature input, and the weight sharing among Θ_t makes the optimization of different tasks dependent. When training by stochastic gradient descent, this amounts to interleaving the stochastic updates for each of the related tasks. That is, the procedure iteratively carries out the following three steps:

1. select a task at random,
2. select a random training example for this task, and
3. compute the gradients of the neural network attributed to this task with respect to this example and update the parameters.

Because some of the parameters are shared between the tasks, the tasks influence each other during training. The training procedure continues until the per-residue error becomes stable on a held-out validation set (one fifth of each training set was held out for this purpose) or reaches a specified maximum number of iterations. In practice, we found that 100–150 iterations are sufficient for the networks trained here.

It is worth noting that labeled data for training each task can come from completely different data sets. However, when the sizes of the training sets for the different tasks are very different, then the above procedure does not work well because the network does not train enough on the "larger" tasks. To address this problem, we employ a pre-training strategy, where the larger tasks (i.e. four large task including "ss", "dssp", "saa" and "sar" in Table 1) are trained jointly prior to multitasking of all tasks. This pre-training procedure ensures that the large tasks reach a stable area of the parameter space before the full multitasking involving all tasks, which strategy is a common practice in the "deep learning" commnity. As pointed out by [41], the pre-training could guide learning towards better generalizations from the training data set, and provides a regularization effect.

The natural protein task: feature learning with unlabeled protein sequences

Labeling a data set can be expensive, especially when doing so requires expensive and time-consuming laboratory experiments. Consequently, the ability to leverage unlabeled data to improve a predictive model is a compelling goal. We now present a semi-supervised task to model the local patterns of amino acid contexts that occur in natural protein sequences.

This "natural protein" task is motivated by results from the natural language processing community. In that context, researchers noticed that for part-of-speech or other semantic tagging tasks, words that are semantically similar can often be exchanged with no impact on the labeling. For example, in a sentence like "the cat sat on the mat" one can replace "cat" with nouns such as "dog", "man" or "patient" with no change in the part-of-speech tagging. Collobert and Weston [1] therefore included in their multitask learning system a task that forces two sentences with the same semantic labels to have similar representations in the shared layers of neural network, and vice versa. Training for this task is achieved by assigning a positive label to genuine fragments of natural language, and negative labels to fragments that have been synthetically generated. Essentially, this task involves learning to predict whether the given text sequence exists naturally in the English languague.

Motivated by this language model, we propose an auxiliary task aiming to model the local patterns of amino acids that naturally occur in protein sequences. This is achieved by learning to predict whether the given protein segment exists in real protein sequences. Accordingly, all length-k windows from SwissProt version 54.7 are labeled as positive examples, and negative fragments are generated by randomly substituting the middle amino acid in each window, as illustrated in Figure 3. Because the training set for this task is extremely large, we train the natural protein task separately from the other tasks. Also, the network architecture used in this task is slightly different from that shown in Figure 1; here we do not use the PSI-BLAST feature encodings (see Figure 4). Other components of the network are the same and explained in the previous sections. As for the other tasks, the amino acid embeddings and the parameters of the subsequent neural network layers are all automatically trained by backpropagation. The difference is that here the model is trained with a ranking-type cost (with margin):

$$\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{D}} \max\left(0,\, 1 - f(s) + f(s^a)\right) \tag{9}$$

where $\mathcal{S}$ is the set of windows of amino acid segments, $\mathcal{D}$ is the vocabulary of amino acids, $f(\cdot)$ represents the output of the neural network architecture, and s^a is a window where the middle amino acid has been replaced by a random amino acid a. Essentially, we are learning the network weights to rank positive protein segments above synthetic segments. The training is carried out using stochastic gradient descent, which samples the cost online with respect to (s, a). As in the natural language setting, the end goal for this training procedure is not the solution to the classification task itself, but the embedding of amino acids into a semantically meaningful, d-dimensional space. The real-valued vectors representing the amino acids comprise the columns of the lookup table W in the amino acid feature extraction layer of the network. Thus, to combine the natural protein task with other tasks, we initialize the amino acid embedding lookup table W in the feature extraction layer (Figure 1) with the embedding weights learned during training of the natural protein task. This step is closely related to the "language model" in natural language processing, where language modeling aims to learn the joint probability function of sequences of words.

Results

Data Sets

The collection of data sets tested in this paper is summarized in Table 1 and is publicly available at *http://noble.gs.washington.edu/proj/multitask*. Our multitask learning data set includes three protein secondary structure prediction tasks. The first is a standard benchmark, CB513 [42], consisting of 513 unrelated proteins with known 3D structure. To create the other two secondary structure prediction tasks, we used 11 795 chains from the DSSP database [43] downloaded on February 1, 2008. We also considered two variants of the secondary structure prediction task, one task using the full 8-letter alphabet and one task using the reduced, 3-letter alphabet. For the signal peptide (SP) and transmembrane (TM) topology prediction tasks, we define two tasks: a five-letter SP+TM topology prediction task, as well as a signal peptide detection task. We used the DSSP to define two solvent accessibility data sets, absolute accessibility and relative accessibility, in which the accessibility is scaled relative to the maximum per-protein accessibility. For the coiled coil prediction task, we use a previously described data set containing 776 proteins [28]. For the prediction of DNA binding residues, we use a data set from [7], consisting of 693 DNA-binding proteins. The supplement provides details of each data set, as well as descriptions of the performance metrics that we used.

Experimental Setup

This work is predicated on a three-fold hypothesis, namely, that we can improve our ability to predict various protein labeling tasks by (1) learning an amino acid embedding, (2) using multitask learning and (3) including the "natural protein" task in our multitask learning. Accordingly, we systematically tested variants of our learning approach, with the goal of testing each of these hypotheses.

Our learning framework requires the specification of a variety of hyperparameters. These include the size k of the sequence window, the size h of the hidden layer, and the learning rate η. We considered $k \in \{11, 13, 15\}$, $h \in \{60, 75, 85, 100, 120\}$, and $\eta \in \{0.001, 0.003, 0.005, 0.01, 0.03\}$. For the size of the embedding vector d in the general embedding layer case or in the natural protein task, we tried $\{10, 15, 25\}$. For the natural protein task, the unlabeled corpus was split into one training (70%) and one validation set. The best parameters found for the natural protein task were $k = 13$, $h = 100$ and $d = 15$. For the other tasks when training separately, we optimized each task's performance through a grid search of parameter combinations. When training jointly on multiple tasks, a grid search on parameters was also performed for the performance optimization (on average). In general, the learning rate η does not vary much across different tasks: for instance, the best learning rates for the tasks dssp, ssp, sar, saa, cc, sp, dna (using the abbreviations from Table 1) are all roughly 0.005. The window size $k = 13$ and the embedding size $d = 15$ gave overall best results. Not surprisingly, the optimal number of hidden units differs depending on the inputs. For instance, when using only the PSI-BLAST feature representation for amino acids, $h = 75$ roughly performs the best for all tasks. After adding the amino acid embedding to the feature extraction layer, $h = 85$ works better. Then, when training a joint model for multitasking several tasks, we found that $h = 120$ gives good performance overall.

A learned amino acid embedding

We begin by evaluating the utility of including an amino acid embedding into the amino acid feature extraction layer of the network. The first two columns of Table 2 provide direct evidence for the utility of the learned embedding. These columns compare the performance of single-task neural networks trained using only the PSI-BLAST embedding or trained using a combination of the PSI-BLAST and learned amino acid embeddings. Considering only the amino acid-level accuracy, including the embedding improves the network's performance on 7 out of 10 tasks. Furthermore, when we consider adding the amino acid embedding in the context of multitask learning (comparing columns "Multi" and "Multi-Embed"), we observe an improvement in 9 out of 10 tasks. The largest improvement is observed for relative solvent accessibility prediction, which improves by 1.8% (from 79.2% to 81.0%) in the multitask setting.

Thus far, these observations are only qualitative. However, to avoid problems with multiple testing correction, we chose not to perform a statistical test comparing the performance of each algorithm with and without the embedding layer. Instead, we perform at the end a statistical test jointly with respect to all three of our hypotheses.

Multitask learning

Next, we evaluate the performance improvement to be gained by performing multitask learning when using just the PSI-BLAST features. Columns 1 and 3 ("Single" and "Multi") in Table 2 compare the performance of networks trained one task at a time, versus networks trained in a multitask fashion. For the multitask learning, we experimented with various joint training schemes, and we settled upon the following. First, we pre-train a joint network for four "larger" tasks—dssp, ss, sar, saa. We then use the resulting learned embedding as a starting point for joint learning of the dna, cc, ppi, sp and tm tasks, with joint multitask learning of the four "larger" tasks. For task cb513ss, to avoid overfitting between the task "ss" and "cb513ss", we train a separate joint network from the task sar and saa, then multitask the joint model with cb513ss.

In all 10 cases, multitask learning improves the amino acid level accuracy. Not surprisingly, the performance difference is most dramatic for tasks with small training sets. For example, the performance on the DNA binding task, which has a data set of 693 proteins, jumps from 82.41% to 85.28%, and the prediction of coiled coil regions, with a data set of 765 proteins, improves from 88.61% to 93.08%. The secondary structure prediction tasks show how multitask learning helps with small data sets: for the small, CB513 data set, the accuracy improves by 3.7%, whereas for the larger secondary structure prediction benchmark, accuracy increases by 1.4%. Similar conclusions can be drawn by comparing the "Embed" and "Multi-Embed" columns: in all 10 cases, multitask learning improves the amino acid-level accuracy.

Natural protein prediction

Finally, we evaluate the utility of the natural protein prediction task. The results in Table 2 confirm that multitasking with this natural protein task is an effective strategy to improve deep neural network training. Comparing column "Embed" to "NP", we see that the performance improves in all cases. The benefit of the natural protein task is more apparent in conjunction with multitask learning, because the latter needs to handle much more complicated cases and to search in a larger parameter space, where a better starting position alleviates the difficulties associated with small training sets. Comparing the "Multi-Embed" column with "All3," we see that adding the natural protein task improves the amino acid level accuracy for all ten tasks. In general, however, adding the natural protein prediction task is not as beneficial as adding multitasking. This observation implies that inter-task dependencies provide more information than the contraints introduced via the natural protein embedding.

Furthermore, Figure 5 provides qualitative evidence that this embedding is helpful. Here, we used principal component analysis to project a learned, 15-dimensional amino acid embedding down to two dimensions for the purposes of visualization. The figure shows that amino acids with similar physical and chemical properties are embedded closely to one another. For instance, we observe clustered groups of hydrophilic (DEKQN) and hydrophobic (LMIVC) amino acids. We also observe that pairs of amino acids that are close in the embedding tend to have high BLOSUM62 scores [44], indicating that they can readily substitute for one another in real protein sequences. Specifically, we calculate the k-nearest neighbors for each amino acid, first based on our learned embedding and then based on BLOSUM62. We found that, with $k = 3$, 62% of the amino acid neighbors identified by one method were also identified by the other. This result suggests that the learned embedding is closely related to BLOSUM62, even though it is learned purely from unlabeled protein sequences without any additional information.

To better understand the impact of the natural protein task, we also evaluated our system without the PSI-BLAST embedding module, but initializing the amino acid embedding with the embedding layer from the natural protein task. These results are reported in Table 2 in the column labeled “NP only”. The “NP only” network performs worse than “Single” (which uses a randomly initialized amino acid embedding plus the PSI-BLAST embedding) in 9 out of 10 tasks; however, combining the natural language task with the PSI-BLAST embedding (i.e. “NP”) makes better predictions than “Single” in 8 out of 10 tasks. Thus, the embedding learned from the natural protein task is complementary to the PSI-BLAST features.

Viterbi post-processing

Thus far, our neural network framework uses a labeling-per-amino-acid strategy without exploiting dependencies among the targeted classes. This approach assumes that the label of each position in a protein sequence can be predicted independent of nearby positions in the sequence. Empirically, this assumption fails dramatically for many local structure alphabets. For instance, the repeated label patterns of `abcdefg` in coiled coil predictions exhibit strong inter-label dependencies. It is straightforward to handle these local dependencies by applying a Viterbi (dynamic programming) post-processing step on the label posteriors. Adding this postprocessing step on our multitasking deep network ouputs (the “All3+Vit” column in Table 2) improves the the performance on 7 out of 10 of our tasks.

Evaluation of the final system

The “All3+Vit” column in Table 2 represents the final performance of our multitask learning strategy. To evaluate the statistical significance of the difference between these results and the initial results provided by the single-task neural network, we performed a Z-test on each task. The result p-values are reported in Table 2. For almost all tasks, the “All3+Vit” setting is consistently better than the “Single” case, with most of the p-values smaller than 0.05. The only exception is the signal peptide task, assessed according to the protein level accuracy. The lack of statistical significance here is partly because the existing methods already achieve very good performance (99.47%) and partly because the protein-level metric provides fewer data points as input to the statistical test.

Relative to published, state-of-the-art prediction systems, our multitask, deep learning methodology fares quite well. For the secondary structure prediction task, our system achieved 80.33% amino acid level accuracy on the benchmark CB513 secondary structure prediction data set, which is slightly better than the state-of-the-art 80.0% [45]. For the signal peptide and transmembrane protein topology prediction tasks, our system outperforms Philius [19] on the same benchmark, though this improvement is partly because Philius does not make use of homology information. For prediction of coiled coil regions, our performance of 97.35% beats the best result (94%) on the same data set from [46] using the same evaluation setup. For the DNA binding task, our performance of 89.17% is slightly better than that of a previously described system [7].

In the protein-protein interaction task we achieve much higher per-residue accuracy (73.35%) than the ISIS system, which reports an accuracy of 68% [37]. Note, however, that this is an extremely unbalanced task, so a predictor that simply assigns all amino acids to the negative class would achieve an accuracy of 73%.

Discussion

We have described a multitask learning strategy for training a deep neural network architecture for the prediction of a variety of local protein properties. Our approach makes use of a learned embedding to share information across related tasks and uses the natural protein task to provide a good starting point for the learning of this embedding. We demonstrated that learning tasks simultaneously can improve generalization performance. In particular, when jointly trained with the natural protein task, our architecture achieved state-of-the-art performance in nearly all of the tasks that we considered.

We are not the first to suggest that multiple protein labeling tasks should be considered jointly. Many previous authors have combined tasks: transmembrane topology and signal peptide prediction [47], secondary structure and solvent accessibility [8], secondary structure, solvent accessibility and DNA-binding sites [7]. Nonetheless, we believe that the ability to train jointly on a large variety of tasks is novel and provides a flexible, robust prediction system that will be of great practical utility.

Our methodology could easily be adapted to additional tasks, such as the prediction of glycosylation sites, torsion angles, etc. The methodology could likely also benefit from further optimization. For example, for computational reasons, we have only performed joint training on a relatively small number of tasks. In principle, given sufficient computing power, a single, large network trained on all tasks may yield even better performance. In addition, it is possible to induce better pseudo-negative examples in the proposed natural protein task by adding biological knowledge, e.g., by simulating evolution. This, in turn, might increase our system's performance even more.

Acknowledgments

This work was funded by NIH awards R01 GM074257 and P41 RR0011823, and by the Academy of Finland and the Finnish Cultural Foundation.

References

1. Collobert R, Weston J (2008) A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the International Conference on Machine Learning.

2. Caruana R (1997) Multitask learning. Machine Learning 28: 41–75.

3. Bengio Y, Ducharme R, Vincent P (2000) A neural probabilistic language model. Journal of Machine Learning Research 3: 1137–1155.

4. Collobert R, Weston J (2007) Fast semantic extraction using a novel neural network architecture. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, Association for Computational Linguistics. pp. 25–32.

5. Qian N, Sejnowski TJ (1988) Predicting the secondary structure of globular proteins using neural network models. Journal of Molecular Biology 202: 865–884.

6. Krogh A, Larsson B, von Heijne G, Sonnhammer EL (2001) Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. Journal of Molecular Biology 305: 567–80.

7. Ofran Y, Rost B (2007) Prediction of DNA-binding residues from sequence. Bioinformatics 23: i347–i353.

8. Adamczak R, Porollo A, Meller J (2005) Combining prediction of secondary structure and solvent accessibility in proteins. Proteins 59: 467–475.

9. Rost B, Sander C (1993) Prediction of protein secondary structure at better than 70% accuracy. Journal of Molecular Biology 232: 584–99.

10. Cuff JA, Barton GJ (2000) Application of multiple sequence alignment profiles to improve protein secondary structure prediction. Proteins: Structure, Function, and Bioinformatics 40: 502–511.

11. Cole C, Barber JD, Barton GJ (2008) The Jpred 3 secondary structure prediction server. Nucleic Acids Research 36: W197–W201.

12. Jones DT (1999) Protein secondary structure prediction based on position-specific scoring matrices. Journal of Molecular Biology 292: 195–202.

13. Katzman S, Barrett C, Thiltgen G, Karchin R, Karplus K (2008) PREDICT-2ND: a tool for generalized protein local structure prediction. Bioinformatics 24: 2453-2459.

14. Chu W, Ghahramani Z, Podtelezhnikov A, Wild DL (2006) Bayesian segmental models with multiple sequence alignment profiles for protein secondary structure and contact map prediction. IEEE/ACM transactions on computational biology and bioinformatics 3: 98–113.

15. Yao XQ, Zhu H, She ZS (2008) A dynamic bayesian network approach to protein secondary structure prediction. BMC Bioinformatics 9.

16. Nielsen H, Engelbrecht J, Brunak S, von Heijne G (1997) Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. Protein Engineering 10: 1–6.

17. Nielsen H, Krogh A (1998) Prediction of signal peptides and signal anchors by a hidden Markov model. Proc Int Conf Intell Syst Mol Biol 6: 122-130.

18. Käll L, Krogh A, Sonnhammer ELL (2004) A combined transmembrane topology and signal peptide prediction method. Journal of Molecular Biology 338: 1027–1036.

19. Reynolds SM, Käll L, Riffle ME, Bilmes JA, Noble WS (2008) Transmembrane topology and signal peptide prediction using dynamic bayesian networks. PLoS Computational Biology 4: e1000213.

20. Rost B, Sander C (1994) Conservation and prediction of solvent accesibility in protein families. Proteins: Structure, Function and Genetics 20: 216–226.

21. Pollastri G, Baldi P, Fariselli P, Casadio R (2002) Prediction of coordination number and relative solvent accessibility in proteins. Proteins 47: 142–153.

22. Yuan Z, Burrage K, Mattick JS (2002) Prediction of protein solvent accessibility using support vector machines. Proteins: Structure, Function, and Bioinformatics 48: 566–570.

23. Richardson CJ, Barlow D (1999) The bottom line for prediction of residue solvent accessibility. Protein Engineering Design & Selection 12: 1051–1054.

24. Gianese G, Pascarella S (2006) A consensus procedure improving solvent accessibility prediction. Journal of Computational Chemistry 27: 621–626.

25. Lupas A, Van Dyke M, Stock J (1991) Predicting coiled coils from protein sequences. Science 252: 1162-4.

26. Berger B, Wilson D, Wolf E, Tonchev T, Milla M, et al. (1995) Predicting coiled coils by use of pairwise residue correlations. Proceedings of the National Academy of Sciences of the United States of America 92: 8259.

27. Wolf E, Kim P, Berger B (1997) MultiCoil: a program for predicting two-and three-stranded coiled coils. Protein Science 6: 1179.

28. McDonnell AV, Jiang T, Keating AE, Berger B (2006) Paircoil2: improved prediction of coiled coils from sequence. Bioinformatics 22: 356–358.

29. Delorenzi M, Speed T (2002) An HMM model for coiled-coil domains and a comparison with PSSM-based predictions. Bioinformatics 18: 617–625.

30. Bartoli L, Fariselli P, Krogh A, Casadio R (2009) CCHMM_PROF: a HMM-based coiled-coil predictor with evolutionary information. Bioinformatics 25: 2757–2763.

31. Ahmad S, Gromiha MM, Sarai A (2004) Analysis and prediction of DNA-binding proteins and their binding residues based on composition, sequence and structural information. Bioinformatics 20: 477–486.

32. Ahmad S, Sarai A (2005) PSSM-based prediction of DNA binding sites in proteins. BMC Bioinformatics 6: 33.

33. Pazos F, Helmer-Citterich M, Ausiello G, Valencia A (1997) Correlated mutations contain information about protein-protein interaction. Journal of Molecular Biology 271: 511–523.

34. Gallet X, Charloteaux B, Thomas A, Brasseur R (2000) A fast method to predict protein interaction sites from sequences. Journal of Molecular Biology 302: 917–926.

35. Yan C, Dobbs D, Honavar V (2004) A two-stage classifier for identification of protein-protein interface residues. Bioinformatics 20 Suppl 1: i371–i378.

36. Liang S, Zhang C, Liu S, Zhou Y (2006) Protein binding site prediction using an empirical scoring function. Nucleic Acids Research 34: 3698–3707.

37. Ofran Y, Rost B (2007) ISIS: interaction sites identified from sequence. Bioinformatics 23: e13–e16.

38. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, et al. (1997) Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. Nucleic Acids Research 25: 3389–3402.

39. Riis SK, Krogh A (1996) Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. Journal of Computational Biology 3: 163–183.

40. Kim H, Park H (2003) Protein secondary structure prediction based on an improved support vector machines approach. Protein Eng 16: 553–560.

41. Bengio Y, Glorot X (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of AISTATS 2010. volume 9, pp. 249-256.

42. Cuff JA, Barton GJ (1999) Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. Proteins 34: 508–519.

43. Kabsch W, Sander C (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 22: 2577-2637.

44. Henikoff S, Henikoff JG (1992) Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Sciences of the United States of America 89: 10915–10919.

45. Kountouris P, Hirst JD (2009) Prediction of backbone dihedral angles and protein secondary structure using support vector machines. BMC Bioinformatics 10.

46. Wishart DS, Arndt D, Berjanskii M, Guo AC, Shi Y, et al. (2008) PPT-DB: the protein property prediction and testing database. Nucleic Acids Research 36: D222–D229.

47. Käll L, Krogh A, Sonnhammer ELL (2007) Advantages of combined transmembrane topology and signal peptide prediction–the Phobius web server. Nucleic Acids Research 35: W429.

Figure 1. Deep neural network architecture. Given an input amino acid sequence, the neural network outputs a posterior distribution over the class labels for that amino acid. This general deep network architecture is suitable for all of our prediction tasks. The network is characterized by three parts: (1) an amino acid feature extraction layer, (2) a sequential feature extraction layer, and (3) a series of classical neural network layers. The first layer consists a PSI-BLAST feature module and an amino acid embedding module. With a sliding window input $\{s_1, s_2, \ldots s_k\}$ (here $k = 7$), the amino acid embedding module outputs a series of real valued vectors $\{W_{s_1}, W_{s_2}, \ldots W_{s_k}\}$. Similarly, the PSI-BLAST module derives k 20-dimensional PSI-BLAST feature vectors corresponding to the k amino acids. These vectors are then concatenated in the sequential extraction layer of the network. Finally, the derived vector is fed into the classical neural network layers. The final softmax layer allows us to interpret the outputs as probabilities for each class.

Figure 2. Multitask learning with weight sharing between multiple deep neural networks. In this figure, two related tasks are trained simultaneously using the network the architecture from Figure 1. Here only the very last layers of the network are task specific.

Figure 3. Training examples for the "natural protein" auxiliary task. The "natural protein" auxiliary task aiming to model the local patterns of amino acids that naturally occur in protein sequences. This figure shows a positive training example S (natural protein sequence) and the corresponding negative training example S^a. Both examples are fed into the network to compute the rank-type cost for training.

Figure 4. Network architecture for training the "natural protein" auxiliary task. The "natural protein" auxiliary task aiming to model the local patterns of amino acids that naturally occur in protein sequences. Using local windows in the unlabeled protein sequences as positive examples and randomly modified windows as negative examples, the network learns the feature representations for each amino acid. In contrast to the network illustrated in Figure 1, the network contains only the amino acid embedding module in the first layer of the network. The learned embedding is encoded into the real valued parameter matrix of the amino acid feature extraction layer.

Figure 5. A learned amino acid embedding. The figure shows an approximation of a 15-dimensional embedding of amino acids, learned by a neural network trained on the natural protein task. The projection to 2D is accomplished via principal component analysis.

Table 1. Summary of data sets

Name	Task	Prot Num	AA Num	CV	Composition (%)
ss	Secondary structure	11 765	2 518 596	5	41.7=C, 21.6=E, 36.7=H
cb513ss	Secondary structure	497	83 707	7	42.8=C, 22.7=E, 34.5=H
dssp	Secondary structure, DSSP	11 765	2 518 596	5	33.3=H,20.4=E, 20.1=L, 11.2=T, 9.5=S, 3.5=G, 1.1=B, 0.02=I
sar	Relative solvent accessibility	11 765	2 518 596	5	51.1=B, 48.9=A
saa	Absolute solvent accessibility	11 795	2 518 596	5	64.9=B, 35.1=A
dna	DNA binding	693	127 064	3	81.2=N, 18.8=B
sp	Signal peptide	2816	1 058 598	10	30.8=O, 4.0=S, 65.2=N
tm	Transmembrane topology	1457	460 780	10	82.1=O, 9.6=I, 7.5=M, 0.3=S, 0.1=R, 0.4=N
cc	Coiled coil	765	444 138	10	69.8=N, 4.3=each of a/b/c/d/e/f/g
ppi	Protein protein interaction	1129	188 676	3	73.4=P 26.6=N

For each data set, we list the number of protein sequences, the number of amino acids, the number of cross validation folds, and the proportion of amino acids assigned to each label.

Table 2. Comparison of learning strategies

Embedding?		✓		✓	*	*	*			
Multitask?			✓	✓			✓			
Natural protein?					✓	✓	✓			
Task	Single	Embed	Multi	Multi-Embed	NP	NP only	All3	All3+Vit	p-value	Previous
ss	0.7907	0.7964	0.8050	0.8130	0.7968	0.6766	**0.8174**	0.8141	1e-4	—
cb513ss	0.7610	0.7454	0.7976	0.8019	0.7479	0.6584	0.8020	**0.8033**	1e-3	0.800 [45]
dssp	0.6548	0.6625	0.6708	0.6810	0.6627	0.5426	0.6821	**0.6821**	1e-4	—
sar	0.7836	0.7979	0.7920	0.8100	0.7981	0.7306	0.8104	**0.8106**	1e-4	—
saa	0.8069	0.8128	0.8170	0.8256	0.8130	0.7419	**0.8263**	0.8262	1e-4	—
dna	0.8241	0.8222	0.8528	0.8702	0.8230	0.8113	0.8864	**0.8917**	1e-4	0.89 [7]
sp	0.8092	0.8069	0.8363	0.8392	0.8071	0.6944	0.8408	**0.9100**	1e-4	—
sp (prot)	0.9947	0.9947	0.9982	**0.9983**	0.9980	0.9981	0.9965	0.9977	5e-2	0.97 [19]
tm	0.8708	0.8754	0.8896	0.8931	0.8765	0.8582	0.8944	**0.9212**	1e-4	—
tm (seg)	0.9095	0.9691	0.9738	0.9825	0.9674	0.9272	**0.9837**	0.9653	1e-4	0.94 [19]
cc	0.8861	0.8988	0.9308	0.9421	0.9074	0.8725	0.9439	**0.9660**	1e-4	—
cc (seg)	0.9067	0.9188	0.9454	0.9555	0.9198	0.8972	0.9573	**0.9735**	1e-4	0.94 [46]
ppi	0.6983	0.7020	**0.7436**	0.7334	0.7111	0.7104	0.7375	0.7380	1e-4	0.68 [37]

The table lists, for each prediction task, the per-residue accuracy achieved via single-task training of the neural network with just the PSI-BLAST features ("Single"), single-task training that includes the amino acid embedding ("Embed"), multitask training just using the PSI-BLAST features ("Multi"), multitask training including the amino acid embedding ("Multi-Embed"), multitask training of one task along with the natural protein task ("NP"), multitask training without the PSI-BLAST embedding module but initializing the amino acid embedding by using the natural protein task ("NP only"), multitask training including the natural protein task ("All3"), "All3" with Viterbi post-processing ("All3+Vit") and a previously reported method ("Previous"). Each row corresponds to a single task. The p-value column indicates whether the difference between "Single" and "All3+Vit" is significant, according to a Z-test. Rows labeled "(prot)" or "(seg)" report the protein- or segment-level accuracy, rather than residue-level accuracy. For the "NP" setting, the "*" in the "Embedding?" row indicates that this network uses the pre-trained embedding layer from the natural protein task.