Combining Super-structuring and Abstraction on Sequence Classification

Adrian Silvescu
Yahoo! Labs
Sunnyvale, CA
silvescu@yahoo-inc.com

Cornelia Caragea
Computer Science Department
Iowa State University
cornelia@cs.iastate.edu

Vasant Honavar
Computer Science Department
Iowa State University
honavar@cs.iastate.edu

***Abstract*—We present an approach to adapting the data representation used by a learner on sequence classification tasks. Our approach that exploits the complementary strengths of super-structuring (constructing complex features by combining existing features) and abstraction (grouping of similar features to generate more abstract features), yields smaller and, at the same time, accurate models. Super-structuring provides a way to increase the predictive accuracy of the learned models by enriching the data representation (and hence, increases the complexity of the learned models) whereas abstraction helps reduce the number of model parameters by simplifying the data representation. The results of our experiments on two data sets drawn from macromolecular sequence classification applications show that adapting data representation by combining super-structuring and abstraction, makes it possible to construct predictive models that use significantly smaller number of features (by one to three orders of magnitude) than those that are obtained using super-structuring alone, without sacrificing predictive accuracy. Our experiments also show that simplifying data representation using abstraction yields better performing models than those obtained using feature selection.**

***Keywords*-super-structuring; abstraction; feature selection**

I. Introduction

Sequence classification arises in many real-world problems. For example, in computational biology, predicting protein function or subcellular localization can be formulated as sequence classification tasks, where the amino acid sequence of the protein is used to classify the protein in functional and localization classes.

Representational commitments, i.e., the choice of features or attributes that are used to describe the sequence data presented to a learner, and the level of detail at which they describe the data, can have a major impact on the difficulty of learning, and the accuracy, complexity, and comprehensibility of the learned predictive model [1]. The representation has to be rich enough to capture distinctions that are relevant from the standpoint of learning, but not so rich as to make the task of learning harder due to overfitting.

Sequence data contain intrinsic dependencies between their constituent elements. Given a sequence $\mathbf{x} = (x_0, \cdots, x_{t-1})$ over a finite alphabet $\mathcal{X}$, $\mathbf{x} \in \mathcal{X}^*$, the dependencies between neighboring elements can be modeled using *super-structuring*. Super-structuring involves generating all the contiguous (potentially overlapping) sub-sequences of a certain length k, $(x_{i-k}, \cdots, x_{i-1})$, $i = k, \cdots, t$, $k > 1$, called super-structures, or k-grams (see e.g., [2] for more details). Super-structuring aims to increase the richness of the representation [3]. While super-structuring provides a way to improve predictive accuracy, the number of parameters of the resulting models increases exponentially with k.

Traditional feature selection methods [3], [4] select a subset of the available features based on some chosen criteria. On a sequence classification task, selecting a subset of super-structures by feature selection can substantially reduce the number of model parameters. However, this approach of combining super-structuring and feature selection may not capture important sequence patterns (*motifs*, in the case of biological sequences) that are removed during the selection process.

Feature abstraction methods [5], [6], on the other hand, group "similar", existing features to generate more abstract features. These methods aim to control the level of detail in the data representation. We present an approach to adapting the data representation used by a learner that exploits the complementary strengths of super-structuring and feature abstraction. Specifically, we propose an algorithm to construct new features in a sequence classification scenario by combining *super-structuring* and *abstraction*. Super-structuring improves classification performance at the expense of increasing the model size, and abstraction reduces the model size and improves the statistical estimates of complex models (especially when data is sparse) by reducing the number of parameters to be estimated from data.

We evaluate our approach on two protein subcellular localization data sets. The problem of predicting subcellular protein localization is important in cell biology, because it can provide valuable information for predicting protein function and protein-protein interactions, among others.

The results of our experiments show that adapting data representation by combining super-structuring and abstraction makes it possible to construct predictive models that use significantly smaller number of features (by one to three orders of magnitude) than those that are obtained using super-structuring independently without sacrificing predictive accuracy. The results also show that, for the same number of features used, models obtained using abstraction outperform those obtained using feature selection.

Algorithm 1 Feature Construction

Input: $\mathcal{D} = (\mathbf{x}_l, y_l)_{l=1,\cdots,N}$, $\mathbf{x}_l \in \mathcal{X}^*$, $y_l \in \mathcal{Y}$; k = length of super-structures; m = number of features to construct
Output: A transformed data set $\mathcal{D}_{\mathcal{S}+\mathcal{A}}$

$\mathcal{S} \leftarrow$ `Generate-Super-structures` $(\mathcal{D})$

$\mathcal{D}_{\mathcal{S}} \leftarrow$ `Encode` $(\mathcal{D}, \mathcal{S})$

$\mathcal{A} \leftarrow$ `Construct-Abstractions` $(\mathcal{D}_{\mathcal{S}}, m)$

$\mathcal{D}_{\mathcal{S}+\mathcal{A}} \leftarrow$ `Encode` $(\mathcal{D}_{\mathcal{S}}, \mathcal{A})$

II. Combining Super-structuring and Abstraction

Let $\mathcal{D} = (\mathbf{x}_l, y_l)_{l=1,\cdots,N}$ be a data set of sequences $\mathbf{x}_l$ over a finite alphabet $\mathcal{X}$ along with their associated class labels y_l from a finite set $\mathcal{Y}$. Our approach to feature construction that combines super-structuring and abstraction is illustrated in Algorithm 1. The *input* of the algorithm is: the data set $\mathcal{D}$; the length k of the super-structures; and a number m of features to construct. The *output* of the algorithm is a transformed data set $\mathcal{D}_{\mathcal{S}+\mathcal{A}}$ of instances encoded with the newly constructed features.

The algorithm starts by generating a set $\mathcal{S} = \{s_1, \cdots, s_n\}$ of (complex) unique features (i.e., super-structures or k-grams) from the sequences in $\mathcal{D}$. Specifically, the k-grams are substrings of length k over $\mathcal{X}$ that are generated by sliding a window of length k over sequences in $\mathcal{D}$. The cardinality of $\mathcal{S}$ is $|\mathcal{S}| = n \leq |\mathcal{X}|^k$ (note that if a k-gram does not appear in $\mathcal{D}$, it is not considered as a feature[1]). In the next step of the algorithm the original data set $\mathcal{D}$ is transformed into a data set $\mathcal{D}_{\mathcal{S}}$ as follows: each instance $(\mathbf{x}_l, y_l)$ in $\mathcal{D}$ is encoded by $([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)$ where $[s_i : \#s_i^l]$, $i = 1, \cdots, n$ represents frequency counts for the k-gram s_i in the sequence $\mathbf{x}_l$. This representation is known as "bag of features (k-grams)" [7].

With the transformed data set $\mathcal{D}_{\mathcal{S}}$ where each instance consists of a bag of k-grams over the set $\mathcal{S}$ and its associated class label, the algorithm proceeds to reduce the feature set $\mathcal{S}$ to the desired number of features m. The reduction is accomplished by constructing new features or *abstractions* over the k-grams inside the procedure `Construct-Abstractions`$(\mathcal{D}_{\mathcal{S}}, m)$. Specifically, the set of k-grams is partitioned into m non-overlapping sets $\mathcal{A} = \{a_1 : \mathcal{S}_1, \cdots, a_m : \mathcal{S}_m\}$ where a_i denotes the i-th abstraction and $\mathcal{S}_i$ denotes the subset of k-grams that are grouped together into this i-th abstraction. Thus, an abstraction is identified with the collection of k-grams/features that are grouped together. Note that $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_m = \mathcal{S}$ and $\forall 1 \leq i, j \leq m,\ \mathcal{S}_i \cap \mathcal{S}_j = \emptyset$.

[1]The number of unique k-grams is exponential in k. However, for large values of k, many of the k-grams may not appear in the data (and, consequently, their frequency counts would be zero). Note that the number of unique k-grams is bounded by the cardinality of the *multiset* of k-grams.

Algorithm 2 `Construct-Abstractions`

Input: $\mathcal{D}_{\mathcal{S}} = ([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)$, m = number of abstractions to construct
Output: A set of abstractions $\mathcal{A}$, such that $|\mathcal{A}| = m$

Initialize $\mathcal{A} = \{a_1 : \{s_1\}, \cdots, a_n : \{s_n\}\}$
for $u = n + 1$ **to** $2n - m$ **do**
  $(i_{min}, j_{min}) = \arg\min_{i,j} dist(a_i, a_j)$
  $a_u = a_{i_{min}} \cup a_{j_{min}}$
  $\mathcal{A} = \mathcal{A} \backslash \{a_{i_{min}}, a_{j_{min}}\} \cup \{a_u\}$
end for

Finally, the algorithm transforms the data set $\mathcal{D}_{\mathcal{S}}$ into another data set $\mathcal{D}_{\mathcal{S}+\mathcal{A}}$ where each instance is now represented by m features (aside from the class label) as follows: given the set of m abstractions $\mathcal{A} = \{a_1 : \mathcal{S}_1, \cdots, a_m : \mathcal{S}_m\}$, where $\mathcal{S}_i \subseteq \mathcal{S}, \forall 1 \leq i \leq m$, an instance $([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)$ from $\mathcal{D}_{\mathcal{S}}$ is transformed into an instance $([a_1 : \#a_1^l], \cdots, [a_m : \#a_m^l]\}, y_l)$ in $\mathcal{D}_{\mathcal{S}+\mathcal{A}}$, where:

$$\#a_i^l = \sum_{s_q \in \mathcal{S}_i} \#s_q^l,\ \forall 1 \leq i \leq m$$

The advantage of this approach is that the resulting representation of sequences can be used with any learning algorithm. In this study we used Naive Bayes Multinomial [8] and Support Vector Machines [9] with linear kernel. In the next subsection we present the `Construct-Abstractions`$(\mathcal{D}_{\mathcal{S}}, m)$ procedure of Algorithm 1.

A. Constructing Abstractions

The procedure for constructing abstractions is illustrated in Algorithm 2. The *input* of the algorithm is a data set where each instance consists of a bag of k-grams and a class label, $\mathcal{D}_{\mathcal{S}} = ([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)_{l=1,\cdots,N}$, and a number $m \leq n$ that represents the reduced number of abstractions that is desired. The *output* is a set of abstractions $\mathcal{A}$ over the k-grams such that $|\mathcal{A}| = m$.

The algorithm, that is a *greedy* agglomerative procedure, starts by initializing the set of abstractions $\mathcal{A}$ such that each abstraction a_i corresponds to a k-gram s_i in $\mathcal{S}$. Next, the algorithm recursively groups pairs of abstractions until m abstractions are obtained. More precisely, $n - m$ times the algorithm searches for the most "similar" two abstractions $a_{i_{min}}$ and $a_{j_{min}}$, adds a new abstraction a_u to the set of abstractions $\mathcal{A}$ by taking the union of their k-grams, and removes $a_{i_{min}}$ and $a_{j_{min}}$ from $\mathcal{A}$. Finally, after $n - m$ such steps the algorithm returns m abstractions as the final result.

In order to complete the description of Algorithm 2, we need to define the similarity between two abstractions. Our general criterion for establishing similarity between items and, thus, deriving useful abstractions is based on the following observation: *similar items occur within similar contexts.* In our case, we define the context of a k-gram as

the probability of the class variable given the corresponding k-gram (see below). Since an abstraction is a set of k-grams, the class context of an abstraction will be obtained by aggregating the class contexts of its elements. Next, we define a distance $dist(a_i, a_j)$ between the class contexts of two abstractions a_i and a_j and identify the most "similar" abstractions as those that have the smallest distance between their class contexts.

Class Context for Abstractions. Given a transformed data set $\mathcal{D}_\mathcal{S} = ([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)_{l=1,\cdots,N}$ of $\mathcal{D}$, we define the *class context* of a k-gram s_i in $\mathcal{S}$ with respect to $\mathcal{D}$ as follows:

$$CContext_\mathcal{D}(s_i) := [p(Y|s_i), \#s_i]$$

$$= \left[\left[\frac{\#[s_i, y_j]}{\sum_{y_j \in \mathcal{Y}} \#[s_i, y_j]} \right]_{y_j \in \mathcal{Y}}, \sum_{y_j \in \mathcal{Y}} \#[s_i, y_j] \right]$$

That is, the class context of a k-gram s_i is the conditional distribution of the class variable Y given the k-gram s_i, $p(Y|s_i)$ estimated from $\mathcal{D}$, along with the frequency counts of the k-gram s_i in $\mathcal{D}$, $\#s_i$. The variable Y takes values $y_j \in \mathcal{Y}$. The counts $\#[s_i, y_j]$ can be computed from $\mathcal{D}_\mathcal{S}$ by summing $\#s_i^l$ over all instances that belong to $y_j \in \mathcal{Y}$.

More generally, we define the class context of an abstraction $a_i = \{s_{i_1,\ldots,s_{i_q}}\}$ as follows:

$$CContext_\mathcal{D}(\{s_{i_1,\cdots,s_{i_q}}\}) := \left[\sum_{r=1}^{q} \pi_r p(Y|s_{i_r}), \sum_{r=1}^{q} \#s_{i_r} \right]$$

$$\text{where } \pi_r := \frac{\#s_{i_r}}{\sum_{r=1}^{q} \#s_{i_r}}$$

If we "abstract out" the difference between all the k-grams $\{s_{i_1,\ldots,s_{i_q}}\}$ in the abstraction a_i and replace each of their occurrences in $\mathcal{D}$ by a_i, then $\#a_i = \sum_{r=1}^{q} \#s_{i_r}$ and $\#[a_i, y_j] = \sum_{r=1}^{q} \#[s_{i_r}, y_j]$. Furthermore, for any $y_j \in \mathcal{Y}$:

$$p(y_j|a_i) = \frac{\#[a_i, y_j]}{\#a_i} = \frac{\sum_{r=1}^{q} \#[s_{i_r}, y_j]}{\sum_{r=1}^{q} \#s_{i_r}}$$

$$= \sum_{r=1}^{q} \frac{\#s_{i_r}}{\sum_{r=1}^{q} \#s_{i_r}} \frac{\#[s_{i_r}, y_j]}{\#s_{i_r}} = \sum_{r=1}^{q} \pi_r p(y_j|s_{i_r})$$

Hence, we have shown that:

$$CContext_\mathcal{D}(a_i) = [p(Y|a_i), \#a_i]$$

$$= \left[\sum_{r=1}^{q} \pi_r p(Y|s_{i_r}), \sum_{r=1}^{q} \#s_{i_r} \right] = CContext_\mathcal{D}(\{s_{i_1,\cdots,s_{i_q}}\})$$

This is a natural definition of an abstraction based on "abstracting out" the difference between the k-grams $\{s_{i_1,\ldots,s_{i_q}}\}$ and considering them as being a single feature a_i. Note that we have also shown that $\sum_{r=1}^{q} \pi_r p(Y|s_{i_r})$ is actually a probability distribution over $\mathcal{Y}$ because it is the same as $p(Y|a_i)$. Next, we define a distance between the class contexts of two abstractions, motivated by ideas from information theory [10].

Distance Between Abstractions. Our goal is to obtain compact, yet accurate models. Hence, if we denote by A a random variable that takes values in the set of abstractions $\mathcal{A} = \{a_1, \cdots, a_m\}$, our goal reduces to constructing the set $\mathcal{A}$ of abstractions such that the dependency between A and the class variable Y is preserved as much as possible. One way to measure the dependency between two variables is to use mutual information [10]. Hence, we want to construct $\mathcal{A}$ such that the reduction in the mutual information between A and the class variable Y, $I(A, Y)$, is minimized at each step of Algorithm 2.

The reduction in the mutual information between A and Y due to a single merge of Algorithm 2 can be calculated as follows: Let $\mathcal{A}_m$ and $\mathcal{A}_{m-1}$ denote the sets of abstractions corresponding to two consecutive steps of Algorithm 2. Let $\{a_i, a_j\} \rightarrow a_u$ denote the merge that produced $\mathcal{A}_{m-1}$ from $\mathcal{A}_m$. Furthermore, let π_i and π_j denote the prior probabilities of a_i and a_j in the set a_u, i.e. $\pi_i = \frac{p(a_i)}{p(a_i)+p(a_j)}$ and $\pi_j = \frac{p(a_j)}{p(a_i)+p(a_j)}$.

Proposition 1: *The reduction in the mutual information between A and Y, due to the above merge is given by $\delta I(\{a_i, a_j\}, a_u) = (p(a_i) + p(a_j)) \cdot JS_{\pi_i,\pi_j}(p(Y|a_i), p(Y|a_j)) \geq 0$, where $JS_{\pi_i,\pi_j}(p(Y|a_i), p(Y|a_j))$ represents the weighted Jensen-Shannon divergence between two probability distributions $p(Y|a_i)$ and $p(Y|a_j)$ with weights π_i and π_j, respectively. (We omit the proof due to lack of space.)*

The weighted Jensen-Shannon divergence between two probability distributions p_i and p_j with weights π_i and π_j, is given by: $JS_{\pi_i,\pi_j}(p_i, p_j) = \pi_i KL(p_i||\bar{p}) + \pi_j KL(p_j||\bar{p})$, where $\bar{p} = \pi_i p_i + \pi_j p_j$, and $KL(p_i||\bar{p})$ represents the Kullback-Leibler divergence between p_i and $\bar{p}$. The weighted Jensen-Shannon divergence is a generalization of the Jensen-Shannon divergence between two probability distributions that allows for an asymmetry in the consideration of the two elements (in the case of standard Jensen-Shannon divergence $w_1 = w_2 = \frac{1}{2}$) [11].

Hence, we define the distance between two abstractions a_i and a_j, denoted by $dist(a_i, a_j)$, as follows:

$$dist(a_i, a_j) = \delta I(\{a_i, a_j\}, a_u) \text{ where } a_u = \{a_i \cup a_j\}$$

$$= (p(a_i) + p(a_j)) \cdot JS(CContext_\mathcal{D}(a_i), CContext_\mathcal{D}(a_j))$$

III. Feature Selection

In this section we discuss an alternative approach, i.e., feature selection, to reducing the feature set $\mathcal{S}$ of superstructures to a desired number of features m. Feature selection is performed by selecting a subset $\mathcal{F}$ of features (superstructures or k-grams) from the entire set $\mathcal{S}$, $\mathcal{F} \subseteq \mathcal{S}$ such that $|\mathcal{F}| = m$ and $m \leq n$. The features are ranked according to

a scoring function $Score$ and the top m best ranked features are selected.

We used mutual information [10] between the probability of the class variable $p(Y)$ and the probability of the feature s_i, which measures how dependent the two variables are. More exactly, let $\mathcal{D}_{\mathcal{S}} = ([s_1 : \#s_1^l], ..., [s_n : \#s_n^l], y_l)_{l=1,\cdots,N}$ be a transformed data set of $\mathcal{D}$, let Y be the class variable that takes values $y_j \in \mathcal{Y}$, and let $s_i \in \mathcal{S}$ be a feature. We define a scoring function $Score$ of the feature s_i as follows:

$$Score_{\mathcal{D}}(s_i) = KL(p(s_i, Y)||p(s_i)p(Y))$$

where $p(s_i, Y)$ is estimated from counts gathered from $\mathcal{D}$ and $p(s_i)$ and $p(Y)$ are obtained by marginalization from $p(s_i, Y)$. As before, these counts can be computed from $\mathcal{D}_{\mathcal{S}}$. The $Score_{\mathcal{D}}(s_i)$ is also known as information gain because $KL(p(s_i, Y)||p(s_i)p(Y)) = H(p(Y)) - H(p(Y|s_i)|p(Y))$ where $H(p(Y)) = -\sum_{y_j} p(y_j) \log p(y_j)$ is the Shannon's entropy.

The data set $\mathcal{D}_{\mathcal{S}}$ is transformed into another data set $\mathcal{D}_{\mathcal{S}+\mathcal{F}}$ where each instance is represented by m features (besides the label) as follows: given the selected set of features $\mathcal{F} = \{s_{i_1}, \cdots, s_{i_m}\}$, an instance $([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)$ from $\mathcal{D}_{\mathcal{S}}$ is transformed into an instance $([s_{i_1} : \#s_{i_1}^l], ..., [s_{i_m} : \#s_{i_m}^l], y_l)$ in $\mathcal{D}_{\mathcal{S}+\mathcal{F}}$.

IV. Experiments and Results

Our experiments compare Naïve Bayes (NB) and Support Vector Machine (SVM) classifiers trained using the combination of super-structuring and abstraction as the feature representation with their corresponding counterparts trained using a unigram representation of sequences, super-structuring representation, and super-structuring followed by feature selection representation (detailed below). We show results of these comparisons on two protein subcellular localization data sets: **plant** and **non-plant**[2] introduced in [12]. For both data sets we report the average classification accuracy obtained in a 5-fold cross-validation experiment.

The feature representations used to train the NB and SVM classifiers are as follows:

- unigram: a bag of letters representation of protein sequences, no super-structuring, abstraction or feature selection (UNIGRAM);
- super-structuring: a bag of k-grams ($k = 3$) representation of protein sequences (SS);
- super-structuring and feature selection: a bag of m k-grams ($k = 3$) chosen using feature selection from the bag of k-grams obtained by super-structuring (See Section III for details) (SS+FSEL);
- super-structuring and abstraction: a bag of m abstractions over k-grams ($k = 3$) obtained using the combination of super-structuring and abstraction (See Section II for details) (SS+ABS).

[2]Available at http://www.cbs.dtu.dk/services/TargetP/datasets/datasets.php

For **plant** and **non-plant** data sets the number of 3-grams is 8293 and 8404, respectively. Note that these represent the numbers of 3-grams that occur in the data. For both data sets, the number of unigrams is 22 ($|\mathcal{X}| = 20$ amino acids plus two special characters inserted at the beginning and at the end of each sequence).

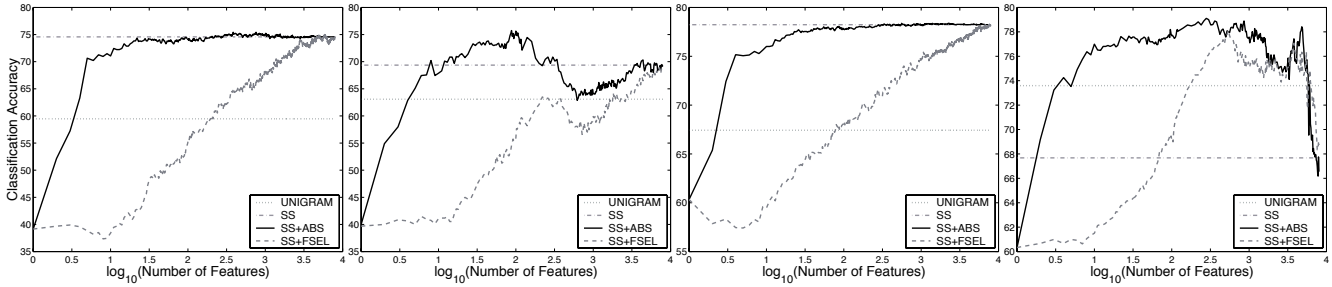
Figures 1a and 1b show the results of the comparison of SS+ABS with UNIGRAM, SS, and SS+FSEL on the **plant** data set using NB and SVM with linear kernel, respectively. Figures 1c and 1d show similar results on the **non-plant** data set. The average classification accuracy is shown as a function of the number of features used in the classification model, ranging from 1 to $n = |\mathcal{S}|$ (the number of unique k-grams). The x axis of Figure 1 shows the number of features on a logarithmic scale.

Comparison of SS+ABS with UNIGRAM. Figure 1 shows that, for any choice of the number of features, classifiers trained using SS+ABS outperform those trained using UNIGRAM. For example, on the **plant** data set, with 22 features, NB and SVM achieve 74.36% and 70.74% accuracy (respectively) using SS+ABS, as compared to 59.46% and 63.08% (respectively) obtained using UNIGRAM. Similarly, on the **non-plant** data set, with 22 features, NB and SVM achieve 77.24% and 77.61% accuracy (respectively) using SS+ABS, whereas NB and SVM achieve 67.42% and 73.59% (respectively) using UNIGRAM.

Comparison of SS+ABS with SS. As can be seen in Figure 1, SS+ABS matches the performance of SS with substantially smaller number of features. Specifically, the performance of SS-based NB and SVM classifiers trained using more than 8000 3-grams is matched by SS+ABS-based classifiers trained using only 22 and 8 features (respectively) on the **plant** data set, and only 270 and 2 features (respectively) on the **non-plant** data set. We conclude that SS+ABS can match the performance of SS using substantially (by one to three orders of magnitude) more compact classifiers.

The performance of SVM on SS is worse than that of NB on SS on both data sets, and also worse than that of SVM on UNIGRAM (Figure 1). This could be due to *overfitting* (see [13] for a theoretical analysis of overfitting for the SVM algorithm). It is interesting to note that, for many choices of the number of features, SS+ABS substantially outperforms SS in the case of SVM on both data sets (Figures 1b and 1d). Thus, SS+ABS can help avoid overfitting by providing more robust estimates of model parameters.

Comparison of SS+ABS with SS+FSEL. As can be seen from Figure 1, SS+ABS outperforms SS+FSEL over a broad range of choices for the number of features. For example, with only 10 features on the **plant** data set, NB using SS+ABS achieves an accuracy of 71.06% as compared to NB using SS+FSEL which achieves an accuracy of only 38.40%. On the same data set with only 10 features, SVM using SS+ABS achieves an accuracy of 67.87% as compared to 41.06% achieved by SVM using SS+FSEL.



(a) **plant** (NB) (b) **plant** (SVM) (c) **non-plant** (NB) (d) **non-plant** (SVM)

Figure 1: Comparison of super-structuring and abstraction (SS+ABS) with super-structuring alone (SS), super-structuring and feature selection (SS+FSEL) and UNIGRAM on the **plant** and **non-plant** data sets using Naïve Bayes (NB), (a) and (c) respectively, and Support Vector Machines (SVM) with linear kernel (b) and (d), respectively. The plots show the accuracy as a function of the number of features used in the classification model, ranging from 1 to $\approx 8,000$ on both data sets. The x axis shows the number of features on a logarithmic scale.

V. Summary and Discussion

We have presented an approach to feature construction in a sequence classification scenario to trade off the predictive accuracy of the learned model against its complexity. Our approach combines *super-structuring* with *abstraction*. Super-structuring involves generating all the contiguous (potentially overlapping) subsequences of a certain length k, i.e. k-grams. Abstraction involves constructing more abstract features by grouping "similar" features.

The results of our experiments on two protein subcellular localization data sets show that adapting data representation by combining super-structuring and abstraction makes it possible to construct predictive models that use significantly smaller number of features (by one to three orders of magnitude) than those obtained using super-structuring alone (whose size grows exponentially with k). Moreover, the performance of such models is similar and, in some cases, better compared to the performance of models that use only super-structuring. In this case, abstraction can be seen as a *regularizer*. The results also show that simplifying data representation using abstraction yields better performing models than those obtained by feature selection.

A. Related Work

Segal et al. [14] and McCallum et al. [15] have used abstraction hierarchies over *classes* to improve classification accuracy. Zhang et al. [5] have used abstraction hierarchies over nominal variables to build compact yet accurate classifiers. In contrast to these methods, we have learned abstraction hierarchies from sequence data, that is a type of topologically constrained data. Our abstraction hierarchies group "similar" k-grams based on the class conditional distributions that they induce, $p(Y|k\text{-}gram)$.

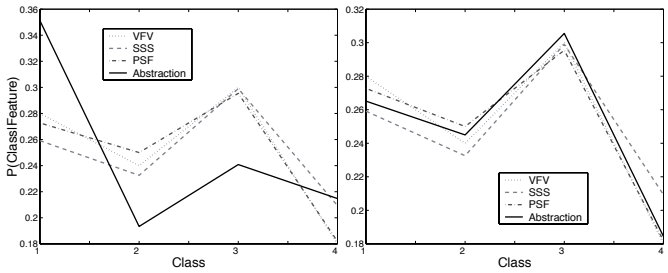
Our work is similar in spirit to the *agglomerative information bottleneck* (AIB) method of Slonim and Tishby [16]. However, AIB requires an iterative process to estimate parameters for the *bottleneck variables*, whereas our method requires only a simple weighted aggregation of the existing parameters.

Feature construction methods aim to increase the richness of the data representation (see [3] for a survey). Super-structuring corresponds to considering higher order moments [17] in the particular case of multivariate statistics. Feature selection (or extraction) methods that select a subset of the available features based on some chosen criteria [18] (see [19] for a survey) and abstraction methods that group *similar* features to generate more abstract features [6], [16], [20] aim to control the level of detail in the representation. With the exception of Jonyer et al. [21] who have described an algorithm for Context-Free Graph Grammar Induction that combines in effect super-structuring and abstraction (although not explicitly, and not aimed at predicting a class variable), there has been relatively limited exploration of techniques that combine super-structuring with abstraction or feature selection in adapting the data representation used for training predictive models to obtain accurate and comprehensible classifiers.

B. Discussion

Although we have focused on super-structuring in the case of sequence data to enrich the data representation, and hence increase the performance of learned models, in general, any topology that reflects the interactions between elements of structured data can be used to guide super-structuring. For example, in the case of image data, this involves generating features from spatially contiguous elements of an image.

Feature selection and abstraction simplify the data representation, and help maintain comprehensible models. While feature selection does that by removing redundant or irrelevant features, abstraction assumes that all features contain some information that is useful for making accurate predictions, and hence, groups sets of features to generate more



(a) 10 abstractions (b) 100 abstractions

Figure 2: Class distributions induced by one of the m abstractions and by three 3-grams, namely "VFV", "SSS", and "PSF", on the **plant** data set, where (a) $m = 10$ and (b) $m = 100$. The three 3-grams are initially sampled from the abstraction when $m = 100$. The number of classes is 4.

abstract features, based on the class distributions that they induce. Figure 2a shows the class distribution induced by one of the 10 abstractions, namely a_1, and the class distributions induced by three 3-grams sampled from a_1, namely "VFV", "SSS", "PSF", when abstraction is performed to reduce the number of features from $8,293$ to 10 on the **plant** data set. Note that the class distribution induced by the abstraction a_1 is very similar to those induced by the individual 3-grams (the average number of 3-grams per abstraction is 829). Hence, the individual 3-grams can be replaced by their abstraction a_1 without significant reduction in the accuracy of the resulting classifiers. In contrast, selecting a subset of 10 features by feature selection decreases classification performance (Figure 1b). When abstraction is performed to reduce the number of features from $8,293$ to 100 on the same data set, the class distributions induced by the same three 3-grams are more similar to the class distribution induced by their abstraction (Figure 2b) (the average number of 3-grams per abstraction is 83). We conclude that SS+ABS provides better estimates of the model parameters than SS+FSEL.

Some possible directions for further research include extensions of the methods developed here for sequence classification to settings where the data have a much richer structure (e.g., multi-modal data consisting of images and text, and combination of audio and visual data).

ACKNOWLEDGMENT

This research was funded in part by National Science Foundation grant number NSF 0711356 to Vasant Honavar.

REFERENCES

[1] L. Valiant, "A theory of the learnable." *Communications of ACM*, vol. 27, pp. 1134–1142, 1984.

[2] E. Charniak, *Statistical Language Learning, Cambridge: 1993*. MIT Press, 1993.

[3] H. Liu and H. Motoda, *Feature Extraction, Construction and Selection*. Springer, 1998.

[4] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.

[5] J. Zhang, D.-K. Kang, A. Silvescu, and V. Honavar, "Learning accurate and concise Naive Bayes classifiers from attribute value taxonomies and data," *Knowledge and Information Systems*, vol. 9, no. 2, pp. 157–179, 2006.

[6] D.-K. Kang, A. Silvescu, J. Zhang, and V. Honavar, "Generation of attribute value taxonomies from data for accurate and compact classifier construction." in *Proceedings of ICDM 2004, Brighton, UK*, 2004.

[7] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *AAAI-98 Workshop on "Learning for Text Categorization"*, 1998.

[8] T. M. Mitchell, *Machine Learning*. NY: McGraw-Hill, 1997.

[9] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition." *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[10] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* John Wiley, 1991.

[11] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, pp. 145–151, 1991.

[12] O. Emanuelsson, H. Nielsen, S. Brunak, and G. von Heijne, "Predicting subcellular localization of proteins based on their n-terminal amino acid sequence." *J. Mol. Biol.*, 2000.

[13] V. Vapnik, *Statistical Learning Theory.* John Wiley & Sons, N.Y., 1998.

[14] E. Segal, D. Koller, and D. Ormoneit, "Probabilistic abstraction hierarchies," in *Proceedings of NIPS 2001*, Vancouver, Canada, 2001, pp. 913–920.

[15] A. K. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng, "Improving text classification by shrinkage in a hierarchy of classes," in *Proceedings of ICML-98*, Madison, US, 1998.

[16] N. Slonim and N. Tishby, "Agglomerative Information Bottleneck," in *Proceedings of NIPS 1999*, pp. 617–623.

[17] K. V. Mardia, J. T. Kent, and J. Bibby, *Multivariate Analysis.* Academic Press, 1979.

[18] L.-P. Liu, Y. Yu, Y. Jiang, and Z.-H. Zhou, "TEFE: A time-efficient approach to feature extraction," in *Proceedings of ICDM-08*, 2008, pp. 423–432.

[19] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. Springer, 1998.

[20] D. Baker and A. McCallum, "Distributional clustering of words for text classification." in *Proc. of SIGIR-98*, 1998.

[21] I. Jonyer, L. Holder, and D. Cook, "MDL-based context-free graph grammar induction and applications," *International Journal of AI Tools*, vol. 13, pp. 45–64, 2004.