Learning meets Sequencing: a Generality Framework for Read-Sets

Filip Železný[1], Karel Jalovec[1], and Jakub Tolar[2]

[1] Czech Technical University in Prague, Faculty of Electrical Engineering
Department of Computer Science
{zelezny,jalovkar}@fel.cvut.cz
[2] University of Minnesota, Stem Cell Institute
tolar003@umn.edu

Abstract. We address a task of supervised learning from data generated by biological sequencing technologies. We aim to learn directly from raw sequencing data (sets of short reads), thereby avoiding the two-step approach where an combinatorial algorithm would first be employed to assemble longer strings from the input reads, and then a learning algorithm would be applied on the assembled strings. We show that our approach leads to a learning scenario involving joint generalization and specialization of read-sets resembling similar operations conducted on clauses in inductive logic programming. We outline the learning framework and prove some properties of its ingredients, however, we also leave some of the important questions unanswered.

Keywords: Biological Sequencing, Classification, Generality Order

1 Introduction

In biology, *sequencing* means reading the sequence of elements that constitute a polymer, such as the DNA, the carrier of genetic information. In the case of DNA, each element (called nucleotide) is one of 4 basic kinds denoted as a, g, c, t. The *human genome project* [1] completed in 2003 was a prime example of sequencing, resulting in the identification of almost the entire genomic sequence (over 3 billion symbols) of a single human. Presently, the cost of sequencing has reduced significantly due to *new generation sequencing* (NGS) technologies. NGS allow to conduct wider studies such as the *1000 Genomes* project [5], and potentially to deploy sequencing even in clinical practise.

Technologically, sequencing becomes difficult as the length of the read sequence grows. The common principle of NGS is that only very short substrings (10's to 100's of symbols) are read at random positions of the sequence of interest. It is usually required that the number of such read substrings, called *reads*, is such that with high probability each position in the sequence is contained in multiple reads. The complete sequence is then determined by combinatorial assembly of the substrings guided by their suffix-prefix overlaps. For example, the

reads {aggc, tgga, gct} would be assembled into aggctgga. Short reads imply low cost of wet-lab sequencing traded off with high computational cost of assembly.

While the assembly task may have a simple basic definition (e.g. find the shortest string subsuming all the given reads), practical algorithms [2] involve a number of heuristics to cope with read errors (misread symbols), non-determinism at each assembly step (multiple overlapping reads), etc. Often it is impossible to produce a single assembled string, and a set of disconnected assembled strings (called contigs) is achieved instead.

Here we consider a problem merging the assembly task with the classification learning task. In particular, we consider a set of labeled individuals, each described by a set of reads. For simplicity we only consider the binary scenario involving the *positive* and *negative* class labels, which may respectively correspond e.g. to disease cases and controls. We look for a discriminative string, i.e., one *consistent* with the reads of each positive individual and not consistent with any of the negative individuals. We call a string consistent with a read-set if the former can be assembled from the latter (we do not require that the string subsumes all the reads). The reason to search such a discriminative string is that it may correspond to a part of the genome responsible for the phenotype manifested by the positive individuals, e.g. by containing a specific set of polymorphisms, mutations, etc. There are about 5400 known mutation-based diseases and about 30 million people suffer from these in Europe only [4].

The stated problem could be solved by first employing an existing assembly algorithm on each read-set, and then employing an existing learning algorithm to find discriminative substrings in the assembled strings (inheriting the class labels from the original read-sets). However, we follow here V. Vapnik's maxim *to solve a problem, do not solve a more difficult problem as an intermediate step.* Hence we want to search discriminative substrings directly from the read-sets.

In the rest of the paper we propose a framework in which such a learning task could be conducted and explore the options for designing its basic ingredients (generality, specialization, generalization). To this end, we exploit analogies to some well established concepts from inductive logic programming (ILP) [3]. Outlining such a framework is the main contribution of this work-in-progress paper. We do not yet provide answers (i.e., algorithms) to the two most important problems of the framework, that is, how to compute the defined specialization and generalization operators. We merely show that some trivial solutions are not valid.

In illustrative examples, we will use arbitrary symbols in strings rather than adhering to the DNA-specific alphabet.

2 An Outline of the Approach

As conventional in symbolic machine learning, we will work with ground data instances, and with structures which *cover* (*subsume*, are *consistent with*) some of the ground instances. In particular, the ground instances are strings, and the structures are read-sets. A read-set covers a string if the string can be assembled

from the read-set. For example, string ababab is covered by read-set {ab, ba, ca}. We will call a read set *less general* than another read-set if the former covers no more strings than than the latter.

Note that the assumed input training data are read-sets rather than strings. That is, they are not expressed on the ground instance level but rather in terms of the entities that are structured by generality. Interestingly, this is analogical to the ILP setting known as *learning from entailment* [3] where the training instances are clauses rather than interpretations (ground possible worlds).

Given these basic concepts, we will now motivate two learning scenarios involving joint specialization (generalization, respectively) of read-sets.

As already mentioned in the introduction, the sequencing of an individual ideally results in a full coverage of the individual, i.e. each position of the read sequence is contained at least in one of the reads. The first learning scenario assumes that this is the case. In such a case we would like to determine the set S_p of all strings, which can be assembled from each of the read-sets pertaining to the positive individuals. This is because $S = S_p \setminus S_n$ (where S_n is the result of the analogical procedure conducted on the negative individuals) are exactly the discriminative strings.

A simple idea to achieve this would be to generate the *extension* (set of all covered strings) of each input read-set and compute the intersection of these extensions. However, the extensions may (and typically will) be infinite (recall that a single read may appear multiple times in the assembled string). We thus need to work on the "lifted" read-set level; in particular, we want to infer a read-set S generating only strings that can be generated from each of the input read-sets. Among such read-sets S (trivially including $S = \emptyset$) we are interested in the most general ones so that we 'lose' as few as possible discriminative strings (e.g. $S = \emptyset$ is so over-specialized that it does not cover any string). As conventional in inductive logic programming, we want to develop a pair-wise specialization operator whose iterative application would lead to such the joint specialization of the entire set of read-sets.

In the complementary scenario, we cannot assume that all positions of each individual's sequence are covered by at least one read. Here, the set of 'suspect' strings will be obviously larger. In particular, we want to determine the set of all strings which can be assembled from at least one (rather than each) of the constituent read-sets. Analogically to the reasoning above, this will lead to the application of a pair-wise generalization operator.

3 The Generality Relation

We now revisit in more detail the *generality* relation. We first define it for single strings and only then for read-sets.[3]

[3] Since reads are strings, we could simply talk of string-sets rather than read-sets. We keep the latter term to distinguish read-sets from their *extensions* which will be defined in turn and which also are sets of strings but treated differently than read-sets.

By $s \sqsubseteq s'$ we denote that string s is a substring of s', e.g. $\mathsf{bc} \sqsubseteq \mathsf{abcd}$ but not $\mathsf{bd} \sqsubseteq \mathsf{abcd}$. It is obvious that $\sqsubseteq$ is transitive: $s_1 \sqsubseteq s_2 \sqsubseteq s_3 \Rightarrow s_1 \sqsubseteq s_3$. Informally, the substring relation instantiates the *cover* relation that many symbolic learning algorithms are based on. A string should be defined as *more general* than another one if the latter covers (has as substrings) no more strings than the latter. Thus, string s_1 is said to be *less general than* s_2, denoted $s_1 \preccurlyeq s_2$, iff for each s, $s \sqsubseteq s_1$ it holds $s \sqsubseteq s_2$.

It is easy to observe that relations $\sqsubseteq$ and $\preccurlyeq$ are equivalent:

- $s_1 \preccurlyeq s_2 \Rightarrow s_1 \sqsubseteq s_2$: Since $s_1 \sqsubseteq s_1$, also $s_1 \sqsubseteq s_2$ from the definition of $\preccurlyeq$.
- $s_1 \sqsubseteq s_2 \Rightarrow s_1 \preccurlyeq s_2$: From transitivity of $\sqsubseteq$, for any $s \sqsubseteq s_1 \sqsubseteq s_2$, it holds $s \sqsubseteq s_2$.

That is, a string is less general than another string iff it is a substring of the latter. This equivalency will cease when we refine the generality relation in the less trivial case of read-sets.

We first extend the substring relation to the read-set level. Let S be a set of strings. Define $s \sqsubseteq S$ iff there is an $s' \in S$ such that $s \sqsubseteq s'$, and define $S_1 \sqsubseteq S_2$ iff for each $s \in S_1$ it holds $s \sqsubseteq S_2$. For completeness, define also $S \sqsubseteq s'$ iff $S \sqsubseteq \{s'\}$. Note that relation $\sqsubseteq$ is still transitive on sets: assume $S_1 \sqsubseteq S_2 \sqsubseteq S_3$, then for each $s_1 \in S_1$ there is $s_2 \in S_2$ such that $s_1 \sqsubseteq s_2$ and $s_3 \in S_3$ such that $s_2 \sqsubseteq s_3$ and thus $s_1 \sqsubseteq s_3$.

Let us briefly digress to explore a simple way of defining the generality relation on read-sets. This relation would be just as for strings, except using the extended substring relation $\sqsubseteq$, i.e.: $S_1 \preccurlyeq S_2$ iff for each $S \sqsubseteq S_1$ it holds $S \sqsubseteq S_2$. With this definition, relations $\sqsubseteq$ and $\preccurlyeq$ would still be equivalent even on the read-set level. This can be seen by verifying that the two inverse implications proven above for the string level are still valid on the read-set level (i.e. by replacing s, s_1, s_2 by S, S_1, S_2 respectively). Given this observation, we have good reason to abandon relation $\preccurlyeq$ due to its equivalency to the (extended) substring relation $\sqsubseteq$.

Now realize that the prescribed learning task actually calls for a definition of read-set generality other than the substring relation. Indeed, a read-set should be called less general than another read-set if all strings that can be *assembled* from the former can also be *assembled* from the latter.

To this end, define $s_1 \pm s_2$ to be the concatenation only for strings with non empty suffix-prefix overlap, so e.g. $\mathsf{abcd} \pm \mathsf{cdef} = \mathsf{abcdef}$, but $\mathsf{ab} \pm \mathsf{cd}$ is not defined. Then define $s \lhd S$ (s *assembled from* S) if

$$s \sqsubseteq s_1 \pm s_2 \pm \ldots \pm s_k,\ s_1 \ldots s_k \in S$$

Note that this definition allows to form loops, e.g. $\mathsf{abababa} \lhd \{\mathsf{ab}, \mathsf{ba}\}$. Furthermore denote the *extension* of read-set S as $Ext(S) = \{s | s \lhd S\}$. It is obvious that extensions are monotonic in the sense that $S_1 \subseteq S_2 \Rightarrow Ext(S_1) \subseteq Ext(S_2)$. We have not yet resolved whether extensions are closed under concatenation, i.e. for any read-set S, $s_1 \in Ext(S) \wedge s_2 \in Ext(S) \Rightarrow s_1 \pm s_2 \in Ext(S)$.

We can now redefine the generality relation for read-sets: S_1 is *less general than* S_2, denoted $S_1 \preceq S_2$, iff for each $s \lhd S_1$ it holds $s \lhd S_2$, i.e. $Ext(S_1) \subseteq Ext(S_2)$.

While $\sqsubseteq$ and $\preccurlyeq$ were shown to be equivalent, $\sqsubseteq$ and $\preceq$ are different relations:

- $\preceq$ does not imply $\sqsubseteq$. For example, consider that $\{\mathsf{abc}\} \preceq \{\mathsf{ab}, \mathsf{bc}\}$ since the only strings $s \lhd \{\mathsf{abc}\}$ are substrings of abc, but $\mathsf{abc} = \mathsf{ab} \pm \mathsf{bc}$ so also $s \lhd \{\mathsf{ab}, \mathsf{bc}\}$. However, it clearly is not the case that $\{\mathsf{abc}\} \sqsubseteq \{\mathsf{ab}, \mathsf{bc}\}$.
- $\sqsubseteq$ does not imply $\preceq$. For example $\{\mathsf{ab}, \mathsf{ba}\} \sqsubseteq \{\mathsf{aba}\}$ but $\{\mathsf{ab}, \mathsf{ba}\} \not\preceq \{\mathsf{aba}\}$ since $\mathsf{bab} \lhd \{\mathsf{ab}, \mathsf{ba}\}$ and not $\mathsf{bab} \lhd \{\mathsf{aba}\}$.

4 Most General Specialization

Given read-sets S_1, S_2, define their *most general specialization* $Mgs(S_1, S_2)$ as a read-set S such that $S \preceq S_1$ and $S \preceq S_2$ and there is no S' such that $S' \preceq S_1$ and $S' \preceq S_2$ and at the same time $S \preceq S'$ and $S' \not\preceq S$. We do not know yet an algorithm for computing $Mgs(S_1, S_2)$ except for the trivial case where $S_1 \preceq S_2$ and obviously $Mgs(S_1, S_2) = S_1$, and the inverse case. In the general case of incomparable read-sets ($S_1 \not\preceq S_2$, $S_2 \not\preceq S_1$), the following example would tempt one to say that $Ext(S_1 \cap S_2) = Ext(S_1) \cap Ext(S_2)$ and thus simply $Mgs(S_1, S_2) = S_1 \cap S_2$:

$$\begin{aligned} S_1 = \{\mathsf{ab}, \mathsf{bc}\} \qquad & Ext(S_1) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{ab}, \mathsf{bc}, \mathsf{abc}\} \\ S_1 = \{\mathsf{bc}, \mathsf{cd}\} \qquad & Ext(S_2) = \{\mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{bc}, \mathsf{cd}, \mathsf{bcd}\} \\ S = S_1 \cap S_2 = \{\mathsf{bc}\} \; & Ext(S) = \{\mathsf{b}, \mathsf{c}, \mathsf{bc}\} = Ext(S_1) \cap Ext(S_2) \end{aligned}$$

However, it turns out that in general, only $Ext(S_1 \cap S_2) \subseteq Ext(S_1) \cap Ext(S_2)$ but not the reverse way. Indeed, due to monotonicity of extensions mentioned in the previous section and the fact that $S_1 \subseteq S_1 \cap S_2$ it must be that $Ext(S_1 \cap S_2) \subseteq Ext(S_1)$. By swapping the variables we also have $Ext(S_1 \cap S_2) \subseteq Ext(S_2)$ and thus $Ext(S_1 \cap S_2) \subseteq Ext(S_1) \cap Ext(S_2)$. However, the inclusion is not symmetric; consider e.g. $Ext(\{\mathsf{ab}, \mathsf{ba}\} \cap \{\mathsf{aba}\}) = Ext(\emptyset) = \emptyset$, however $\mathsf{aba} \in Ext(\{\mathsf{ab}, \mathsf{ba}\}) \cap Ext(\{\mathsf{aba}\}) \neq \emptyset$.

Thus it is an open problem to design an algorithm for computing the most general specialization of two read-sets.

5 Least General Generalization

We are now interested in the *least general generalization* $Lgg(S_1, S_2)$ defined as a read-set S such that $S_1 \preceq S$ and $S_2 \preceq S$ and there is no read-set S' such that $S_1 \preceq S'$ and $S_2 \preceq S'$ and at the same time $S' \preceq S$ and $S \not\preceq S'$.

As in the previous section, the *Lgg* is trivial when the jointly generalized read-sets are comparable in generality. For incomparable read-sets, we first want

to inspect whether simply $Lgg(S_1, S_2) = S_1 \cup S_2$. Let for example

$$\begin{aligned} S_1 &= \{\mathsf{ab}, \mathsf{bc}\} & Ext(S_1) &= \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{ab}, \mathsf{bc}, \mathsf{abc}\} \\ S_2 &= \{\mathsf{bc}, \mathsf{cd}\} & Ext(S_2) &= \{\mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{bc}, \mathsf{cd}, \mathsf{bcd}\} \\ S = S_1 \cup S_2 &= \{\mathsf{ab}, \mathsf{bc}, \mathsf{cd}\} & Ext(S) &= \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{ab}, \mathsf{bc}, \mathsf{cd}, \mathsf{abc}, \mathsf{bcd}, \mathsf{abcd}\} \\ & & &= Ext(S_1) \cup Ext(S_2) \cup \{\mathsf{abcd}\} \end{aligned}$$

The natural question is whether there is a generalization whose extension would contain all the elements of $Ext(S)$ except for the 'extra' string abcd. Note that this could not be the case if extensions are closed under concatenation as discussed in Section 3. On the other hand, if it were the case, then certainly $Lgg(S_1, S_2) \neq S_1 \cup S_2$.

As in the case of *Mgs*, it is an open problem to design an algorithm for computing the least general generalization of two read-sets.

6 Conclusions and Future Work

We have shown that a well motivated problem combining the new-generation sequencing assembly task on one hand, and predictive classification on the other hand, leads to a structured learning formulation involving specialization and generalization operators akin to those known in inductive logic programming. We have provided arguments why these operators likely cannot be computed in a trivial way.

The first direction for future work is thus obvious: design algorithms for these operators, and use these to implement the desired learning algorithm.

The second direction is to explore whether read-sets could be superseded by more expressive formalisms, such as those based on graphs (which, after all, are the main language of state-of-the-art assembly algorithms [2]) or on regular expressions.

Acknowledgement. This work was supported by the Czech Science Foundation project number P202/12/2032.

References

1. Lander, E.: Initial impact of the sequencing of the human genome. Nature 470(7333), 187–97 (2011)
2. Miller, J., Koren, S., Sutton, G.: Assembly algorithms for next-generation sequencing data. Genomics 95(6) (2010)
3. Nienhuys-Cheng, S.H., de Wolf, R.: Foundations of Inductive Logic Programming. Springer (1997)
4. Schieppati, A., Henter, J., Daina, E., Aperia, A.: Why rare diseases are an important medical and social issue. Lancet 371(9629), 2039–41 (2008)
5. Shendure, J., Hanlee, J.: Next-generation DNA sequencing. Nature 26, 1135–45 (2008)