bio1NFORM
The integrated informatics news source

Georgia Tech Improves Gene-Prediction Software to Tackle Tricky Fungal Genomes

October 3, 2008

This article has been updated from a previous version to clarify the publication date of the Genome Research paper and the development history of GeneMark.hmm.

Mark Borodovsky
Director, Center for Bioinformatics and Computational Genomics
Georgia Tech

Mark Borodovsky, director of Georgia Tech's Center for Bioinformatics and Computational Genomics, and his colleagues have recently developed a self-training algorithm to predict protein-coding genes in fungal DNA.

Borodovsky and his colleagues describe the tool, called GeneMark.hmm-ES (BP), in an advance online paper slated for publication in the December issue of *Genome Research*.

While the previous version of GeneMark.hmm has been shown to work well with eukaryotic genomes like *Arabidopsis thaliana*, *Caenorhabditis elegans* and *Drosophila melanogaster*, it was not accurate enough for fungal genomes, which have genes that are more complex due to additional splicing signals, called branch point sites, located in fungal introns.

"To better reflect features of fungal gene organization, we enhanced the intron submodel to accommodate sequences with and without branch point sites," Borodovsky and colleagues write in the paper.

Researchers at the Department of Energy's Joint Genome Institute and the Broad Institute have already begun using the new tool to annotate more than 20 novel fungal genomes.

BioInform recently spoke with Borodovsky about the updated software. Below is an edited transcript of that conversation.

What makes gene prediction so hard?

Bacterial and archaeal genomes have continuous genetic code from [the] start to [the] finish of a gene. In bacteria this is pretty straightforward although it is still not simple to find the starts and ends of gene in bacterial DNA. With eukaryotic genomes like *Drosophila* or *C. elegans*, they have genes that are interrupted by non-coding introns. So that is the first level of complexity. You have to find introns, connect the exons, and make the continuous genetic code message.

Today it is more or less easy to sequence DNA but if we could sequence at the RNA level, it would be much easier to find genes. The cell does part of the work for you — it connects all the exons, [and] you get directly to RNA molecules. That would save a lot of effort but today it is still more difficult than sequencing DNA. To find this additional information, we need to develop methods, do experiments, and extract RNAs.

The whole complement of genes encoded in genomic DNA is a base level, so there [are] no more genes than that. If you are able to decipher genomic DNA, that is all that is possible to see in this genome. On the RNA level there is always a level of uncertainty; you may only have 10 or 20 percent of the genes. … In a way, being able to sequence RNA is a kind of a competitive method to my method conceptually. At this time and for perhaps a long time, it is a fantasy to think you can get the whole complement that way. You would need all genes expressed and transcribed into RNAs, be able to transform all the RNA back to DNA … without losing anything.

The way [the new algorithm works] is by bypassing many obstacles people have when they try to analyze the genomic sequence in a traditional way. We are able to bring experimentalists to predict genes *ab initio*. This step is very important now in my opinion and will remain important for a long time.

So the tool is applicable to other eukaryotes?

As far as *ab initio* gene prediction in DNA sequences [are concerned], perhaps the easy part is to predict genes in bacteria where genes have this continuous simple structure. There are some problems, yes, but we have more problems in eukaryotic species. We developed an algorithm to predict genes in prokaryotes. It was the first version of GeneMark and it was used to annotate genomes for *Haemophilus influenzae*, and the first completely sequenced archaea, *Methanococcus jannaschii* in 1995 and 1996.

Then came the fungal genome, yeast, and [that] was similar to prokaryotic genomes because many genes had no introns — they were continuous DNA sequence with 5,500 genes and only about 250 genes had introns. Gene prediction in yeast was possible using the traditional GeneMark algorithm.

Generally there are only about 100 sequenced eukaryotic genomes and about 1,000 sequenced bacterial and archaeal genomes. With the plans to sequence about 1,000 new eukaryotic genomes, comes increased attention for eukaryotic genomes.

So how does *ab initio* aspect of gene prediction work?

Ab initio means no other information is used, just genomic DNA. The algorithm may contain some parameters, thus making it a kind of supervised learning, since the parameters are defined by the algorithm developers.

There is another type of algorithm, which does not receive parameters from developers. It works on genomic sequence and in the process of analysis, it extracts parameters. It may work in iterations, first extract parameters which are improved in several steps to finally get to the 'best' possible parameters. The algorithm will run with these newly defined parameters and deliver gene prediction. This is the difference between supervised and unsupervised learning approaches.

How does your new algorithm teach itself what to look for?

A general version of GeneMark.hmm-ES worked pretty well for species such as *Drosophila* or *C. elegans* or *Arabidopsis*. Surprisingly enough, their genomes had different features from fungal genomes — the introns had simpler structures.

There are two signals that indicate where introns start and end. In fungal genomes, introns have three signals. The two signals are GT [guanine-thymine] and AG [adenine-guanine]. The third signal is the branch point, pronounced in fungal genomes, but [it] is difficult to find this signal. … We don't know where either introns are nor the branch point. That is the challenge.

The branch site location varies; it is upstream from the end of the intron, the acceptor site AG, but it can be 5, 15, or 20 nucleotides away. So the algorithm has to get the whole picture. It has a concept before it starts running and then materializes this concept. It takes the concept by making initial predictions, looking into genes with initially predicted introns. It selects those and then from that subset, it tries to make a model of branch sites, acceptor-donor sites, and uses those models in another prediction round to be even better at predicting genes.

One difficulty we met is that this strategy actually doesn't work very well. So we had to modify this strategy. In the beginning we just tried to find the beginning and end of the introns and repeated the iteration several times until we got a good set of predicted genes.

We used those to find introns, now more confidently predicted than in the beginning, and from these introns, we then extracted a branch point model. If we tried to do that from the very beginning, we didn't get good models. If we did several iterations before we started to work with branch point sites, predictions significantly improved.

So we work with a hidden Markov model, which has our prior knowledge about gene structure. The model knows there are exons, introns, and knows genomic DNA has to be divided into zones, that inside the genes there are several exons and introns, that exons should end up with donor and start with acceptor sites. It knows about branching sites in a general sense, a probabilistic sense. …The model is prepared to have a sequence of functional zones but it doesn't know where exactly these zones fall in this particular genomic sequence.

So you don't need a test set? Might some scientists say they don't want to trust the results?

Here is another dimension to the whole story. For an algorithm to have probabilistic parameters for exons and introns, developers must deliver these parameters through training, with an experimentally derived test set.

That set has to be big. In this set, everything has its own place, the start of a gene, the positions of the exons and introns are precisely specified. … You need 1,000 of such sequences with genes already mapped, then you can have parameters [that] will let you annotate thousands of other genes in this genome using the *ab initio* algorithm.

In our case we use unsupervised training; we don't need this information about RNA, or about 1,000 genes. We just take 10 megabases of sequence, without any knowledge about the sequence. It just has to be from the same species. Then when we run the algorithm and extract parameters so we can annotate the [sequence].

The iterations do two things. It performs unsupervised training and at the same time it delivers sequence annotation.

So this is where the algorithm helps in the annotation bottleneck?

Yes, this is a bottleneck because doing supervised training means you have to wait until you have information to use in the training set.

Can this method help other teams with gene annotation?

We started to collaborate with the Broad when the algorithm was still in development. We wanted to see if it worked well and if it really does speed up the transfer of knowledge from unannotated data to annotated genomes.

With our procedure an important question is, 'How do we know we are doing a good job [of gene prediction] or not?' And the answer is that if we have even a small set of genes, which are experimentally validated by, let's say, RNA sequencing. For training one needs 1,000 genes but for testing we may just need 100 genes, which is a much easier task.

If we have such a small but representative test set, we can check how good our predictions are. And this is what we show in the paper: if we take test sets and these are sequences with annotated genes, we do as well or better than what can be done by other algorithms in addition to the fact that they need supervised training and parameters.

[The new algorithm] is not only for fungal genomes, although at the moment I believe we have the best algorithm for fungal genomes.