JIGSAW Tutorial

Contents

Chapter 1

Introduction

Here we demonstrate the use of JIGSAW with an example. Training and running JIGSAW differs from traditional *ab initio* gene prediction programs since it does not rely on the genomic sequence alone. In addition to the genomic sequence, JIGSAW must access output from the user's gene structure annotation pipeline. There are many different ways to create an annotation pipeline and this means you may have to do some tweaking of the scripts to integrate JIGSAW into your system. This example demonstrates one way to use JIGSAW using the scripts provided with our distribution (but by no means is this the only way). Feel free to modify or change the code to accommodate your needs.

In this tutorial you should find 4 data directories. Each directory contains a genomic sequence and gene prediction evidence with coordinates mapping to the sequence. We will use directories: 30373, 33025 and 42492 for training and run JIGSAW on the data in directory 60001. To make for a fast download the data provided in this tutorial is limited to a very small subset of a much larger example. The user is cautioned that the predictions generated from this example will not be meaningful, but it allows the user to quickly go through the step by step procedure of using JIGSAW. After completing this tutorial, you may want to play around with the larger dataset, which includes pre-generated JIGSAW predictions and can be use to make comparisons to your own newly generated predictions[1].

Next we go over the steps for training, the first two steps are already set up for this example.

[1] ftp://ftp.cbcb.umd.edu/pub/software/jigsaw/data/arab_pipeline_output.tar.gz

Chapter 2

Training JIGSAW

2.1 Create a training directory

The training directory will contain the decision trees and related prediction files, along with a file containing the list of evidence you use for annotating your sequence, and a file containing a list of directories where the data is located. For this example I typed "mkdir *my_train_dir*" to create a training directory (directory should already come with this tutorial).

2.2 Place two files in the training directory.

(for this tutorial, the files are already located in the *my_train_dir* directory):

1. *my_dir_list.txt* - contains the list of directories with the data you will use to train JIGSAW. Each line in the file should contain either the full path name of a directory or the relative path name with respect to the current training directory (or where you plan to run the training script). Each directory should contain a fasta formatted genomic sequence and the files from the various prediction programs, which contain coordinates mapping to the genomic sequence.

   (a) *my_evidence_list.txt* - This file contains a template listing the evidence files you expect to find and use in each of the directories specified in the *my_dir_list.txt* file. In this example, the *my_evidence_list.txt* contains several lines of information:

   Line 1: "curation.cd default curation" says that we expect to see a file called "curation.cd" in each directory (from *my_dir_list.txt*) that provides the coordinates of the known gene models used for training (indicated by the keyword "curation"). The file format is "default".

   IMPORTANT CHANGE IN VERSION 2.1.3: Partial gene models can be used to train JIGSAW, however, some substantial number of the example genes must still include start and stop codons. If partial gene models are used then the exon type must be provided (initial, internal, single or terminal) in the file containing the training gene coordinates. Otherwise, JIGSAW will assume that the first and last exons of the model correspond to the "true" first and last exons of the model.

   The "curation" line is required for training, the remaining lines in the *my_evidence_list.txt* file are used to predict gene structures. (NOTE: JIGSAW assumes that the filenames specified in the *my_evidence_list.txt* file are prefixed with the directory name they are located in. For example, the *my_evidence_list.txt* file specifies a "genemarkHMM.gp" file and JIGSAW expects to find the filename 30373.genemarkHMM.gp in the 30373 directory, 42492.genemarkHMM.gp in the 42492 directory, etc. (Attaching the prefix to the filenames helps avoid mixing up what files correspond to what genomic sequence.)

   Line 2: "glimmerA.gp default geneprediction acc don coding start stop" says that we expect to see a file called "glimmerA.gp" in each directory, which contains the coordinates of the gene

models predicted by some program (called GlimmerM). In this example, the default file format is used and the type "geneprediction" indicates the prediction type and all 5 prediction models are used (acceptor, donor, protein-coding, start codon and stop codon). Lines 3-7 list the remaining evidence used in the example.

For this tutorial, we prepend the filenames with the name of the directory they are located in. This is the default convention for JIGSAW scripts.

2.3 Run the training procedure.

Make sure that the programs *train_jigsaw.pl*, jigsaw and *mktree* are in a directory in your $path (these programs are provided with JIGSAW distribution). From the training directory (*my_train_dir*) type the following:

train_jigsaw.pl -l my_dir_list.txt -e my_evidence_list.txt -f seq -t

Options passed to the *train_jigsaw.pl* script include:

-t : creates simpler decision trees, which are fairly fast to generate. This option may work just as well as the slower "oblique" splits (which we prefer for final usage) - and for this example we choose the faster "axis parallel" splits. See the OC1 software included with JIGSAW distribution for a more detailed discussion.

-f seq : Indicates the filename of the fasta formated genomic sequence in each directory. (Note that by default the file is prefixed with the directory name.)

-l my_dir_list.txt : filename with the list of directories to process.

-e my_evidence_list.txt : filename with list of evidence to use.

The *train_jigsaw.pl* script is a perl wrapper script, which calls runs the binary jigsaw on each directory. An important option not previously mentioned is to specify how much sequence to examine surrounding each known gene. The presumption is that the input genes do not all occur continuously on the same sequence. The -q option specifies the number of bases to use in the upstream and downstream region of each training gene. The number is currently hard coded to 50 in the train_jigsaw.pl wrapper script, but can easily be changed or made an input parameter. Currently for training, all input data is assumed to be sorted in ascending order, prior to input to train_jigsaw.pl.

After the *train_jigsaw.pl* script is complete, the decision trees should be located in the *my_train_dir* and JIGSAW is ready to run.

Chapter 3

Running JIGSAW

After training is complete we can run JIGSAW using the *run_jigsaw.pl* script. When running JIGSAW, it uses an evidence list file, nearly identical the one described for the training procedure. Before version 3.2.0, when running JIGSAW, it ignores the "curation" line in the evidence list file, and therefore the exact same evidence list file used in training can also be used at runtime. In subsequent versions, the "curation" file can be optionally used to tell JIGSAW where to make predictions in a long sequence. It may be useful in cases where you have test genes for a chromosome, but do not want to run the program on the entire sequence. The -q option is used to specify the number of bases to use surrounding each gene.

To prepare running JIGSAW we will first make a "runtime" version of the evidence list file by copying my_evidence_list.txt to a new file, let's call it my_evidence_list_run.txt and remove the "curation" line from the new file.

We will run JIGSAW from the root directory of the tutorial (type "cd .." if you are in the training directory). In the root directory there is a *run_list.txt* file, which contains the list of directories we want to run JIGSAW on. In this example there is a single directory, 60001.

Make sure *run_jigsaw.pl* is in your search path and type:

run_jigsaw.pl -f seq -e my_train_dir/my_evidence_list_run.txt -d my_train_dir -l run_list.txt -o jigsaw_output.gff

The options: -f, -e, and -l are the same as in the training script. The -d option tells JIGSAW where the decision tree directory is located. When running JIGSAW it is a good idea to start with the same evidence list file as the one used in training, because JIGSAW needs to read the evidence in the same order as it was trained on, and it expects to access the same evidence in training and at runtime.

JIGSAW predictions are placed in a gff formated file called "directoryname".jigsaw.gff in each directory specified in the run_list.txt file (e.g. 60001.jigsaw.gff). Note that the predictions produced for this example are not designed to be used since only a limited amount of training data is available in the example.

IMPORTANT - When running JIGSAW, the only change that can be made to the evidence list file created during training (e.g. my_train_dir/my_evidence_list.txt) is the file naming conventions. You can alter the file names and the text file formats, but lines can not be removed! Each line is associated with a particular piece of evidence, a particular gene finder for example. Even if you do not have this evidence, it must remain listed in the evidence list file. If a particular piece of evidence is not available (either an empty file or a non existent file), JIGSAW will still run but it needs to know that the particular type of evidence is not missing.

3.1 Running JIGSAW with the linear combiner option (training not required)

To use JIGSAW's linear combiner option, changes are made to the command line options used for the statistical version. The run_jigsaw.pl script uses the "–lin" option, which passes the "-l" flag to the jigsaw binary. To evidence list file format used with the linear combiner option differs from evidence list file format used in the trained version of JIGSAW.

The linear combiner evidence list file format requires a numerical weight to be associated with each evidence source's prediction for each of the six gene feature types (acceptor, donor, coding, start, stop and intron). An example of the evidence list file format is given in my_train_dir/my_evidence_list_wghts.txt. In this case all evidence sources are assigned an equal weight of 1 except the splice site prediction program, which is given less weight (0.5). The weight assigned for a given gene feature follows the gene feature name. It is possible for one evidence source to have its constiuent gene feature predictions differently weighted, for example, the "start" codon predictions could be asigned a different weight than the "stop" codon predictions.

The "-d" flag is optional and specifies where the required "param.txt" file is found. When the "-d" flag is not used, the "param.txt" file is assumed to be in the current directory. The command line example to run is:

run_jigsaw.pl –lin -f seq -e my_train_dir/my_evidence_list_wghts.txt -d my_train_dir -l run_list.txt -o jigsaw_output_linoption.gff

The output is placed in the file: 60001/60001.jigsaw_output_linoption.gff

3.2 Evaluating and debugging JIGSAW results

JIGSAW accuracy is dependent on the accuracy of the predictions fed to the program from gene finders and sequence alignments. Therefore, it is important to be aware of the accuracy of the input fed to JIGSAW, and ensure that JIGSAW is correctly reading the file formats. In the tutorial directory there are two directories, *eval* and *prog_ eval*. The *eval* directory contains (unfortunately - crudely written) perl scripts to report the baseline accuracy of the input gene finders. We can test the scripts out to asses the accuracy in the test directory *60001* and see if our JIGSAW predictions are at least showing some improvement over the input gene finders as we would expect.

1. cd into eval and type:

- report.sh "glimmerA.gp gp2gff.pl"
- report.sh "genscan+.gp gp2gff.pl"
- report.sh "genemarkHMM.gp gp2gff.pl"
- report.sh jigsaw_output.gff

The shell script "report.sh" launches a series of perl scripts to compute performance statistics on the various gene finders. The evaluation scripts assume the gene finder output is in gff format, therefore, for the gene finders not in gff format, the conversion script, "gp2gff.pl" is used (additional conversion scripts are included for convenience). JIGSAW output is in gff format, and therefore does not require the use of a conversion script.

1. From the eval directory cd into *../prog_ eval* and type: 'more *.report | ../eval/preport.pl' to display the results.

3.3 Debugging

An additional useful tool, is the -x option, which requires a filename parameter telling JIGSAW where to place the output showing the data JIGSAW uses to make predictions. To test this option, type the command from the root tutorial directory:

jigsaw -f 60001/60001.seq -e 60001/60001.my_evidence_list.txt -d my_train_dir -m jigsaw_debug.gff -x debug.matrix

Now open the debug.matrix. You'll see the sequence divided in to non-overlapping intervals marked by the string: "RegionPrediction either X Y size: Z..." where X and Y are the beginning and ending indices into a portion of the genomic sequence and Z counts the number of evidence sources aligned to the sequence which overlap the region from position X to Y. Next to these values are the independent probabilities of

each gene feature occurring in this subsequence for both strands beginning first with the "positive" strand followed by the "negative" strand. Below this comes the line "Available Evidence Sets:". This shows how JIGSAW encodes the evidence into numerical feature vectors for the interval from X to Y. Below this, the raw inputs from the evidence sources are shown. Additional information showing how the gene structure predictions are made are also shown. Looking at this data matrix can be useful in checking that JIGSAW is storing the data you expect it to store. JIGSAW's file reading ability is somewhat limited so it is possible that small changes in file formats will result in the data being read incorrectly. The data matrix file can be useful in determining whether file reading errors occurred.