Instructions

Dependencies (You need to have the following installed)

- SpliceGrapher (available at: http://splicegrapher.sourceforge.net/)
- PyML (available at: http://pyml.sourceforge.net/)
- SciPy Stack (available at: http://www.scipy.org/install.html)

Follow the following steps to run the code and generate the results.

1. Once you have downloaded this code, copy the two data files to the same directory (where the code is). The two files are *"a_thaliana.fa"*, the whole genome sequences and *"TAIR10_GFF3_genes.gff"*, which is the gene annotation file.

2. Next, copy the folder which has all the splice graphs (which you get after you extract the *"Splicegraphs.tar.gz"* file). If you are copying the splice graphs, you can skip the 3rd step.

3. To generate the splice graphs for all the genes in the gene annotation file, you need SpliceGrapher's script named **gene_model_to_splicegraph.py**. The script is included in the code, however it can be found in the *scripts* directory inside the SpliceGrapher's downloaded directory, in case you want to run it from there. To run it, use the following command in the terminal:

```
> python gene_model_to_splicegraph.py -a -A -v -m TAIR10_GFF3_genes.gff -d SpliceGraphs_TAIR10_noDir
```

   where -a is for annotating the splice graphs, -A is to include all genes, coding and non-coding, -v is for verbose output, -m is for the input gene annotation file, and -d is to specify the output directory. Note that you MUST create the directory beforehand.

4. Once the splice graphs are generated, use the following command to extract the positive and negative examples for both the 3' and 5' AS cases.

```
> python extract_examples.py SpliceGraphs_TAIR10_noDir
```

   Where SpliceGraphs_TAIR10_noDir is the directory which have all the splice graphs (generated in step 3). The name can be different if your splice graphs are in some other directory. The output of this command will generate positive and negative examples for the two AS events (for all three locations, Exon, Intron, and Splice). The positive examples will be in the folders named *"Alt3"* and *"Alt5"* for the two corresponding AS types while the negative examples will be in the folder named *"Rest"*.

5. Next step is to generate the actual datasets by combining the positive and negative examples for each AS type. To do that, execute the following command:

```
> python generate_final_datasets.py
```

   This command will generate the three individual datasets (and their labels) for both the AS types. So for instance, *"Alt3_Intron3.txt"* will be the Intron dataset for Alt. 3 type and its labels will be in *"Labels_Alt3_Intron3.txt"*.

6. Next step is to run the actual experiments on the datasets using the Support Vector Machines. There are total **8** scripts that you can run to generate the results:
   - **experimentsAlt3.py** and **experimentsAlt5.py**
     - Train SVMs on each dataset individually (for Alt. 3 and Alt. 5 types respectively). In the three datasets, Intron and Exon are generated using Spectrum kernel while Splice is generated using WD kernel.
   - **experimentsAlt3_spectrum.py** and **experimentsAlt5_spectrum.py**
     - Train SVMs on each dataset individually (for Alt. 3 and Alt. 5 types respectively). All the three datasets (Intron, Exon, and Splice) are generated using the Spectrum kernel.
   - **experimentsAlt3Aggregate.py** and **experimentsAlt5Aggregate.py**
     - Train SVMs on the composite dataset generated by combining the three datasets (for Alt. 3 and Alt. 5 types respectively). In the three datasets, Intron and Exon are generated using Spectrum kernel while Splice is generated using WD kernel.
   - **experimentsAlt3Aggregate_spectrum.py** and **experimentsAlt5Aggreate_spectrum.py**
     - Train SVMs on the composite dataset generated by combining the three datasets (for Alt. 3 and Alt. 5 types respectively). All the three datasets (Intron, Exon, and Splice) are generated using the Spectrum kernel.

   Use the following command to run any of the 8 scripts:

```
> python scriptname.py
```

   Each script will store its results in a folder with a name indicating what the script is doing. For example, **experimentsAlt3_spectrum.py** creates a folder named *"Alt3_Individual_Spectrum_Results"* to store its results in. The results include a text file detailing the accuracy of classification in terms of unbalanced success rate, balanced success rate, and Area under the ROC curve. The results folder also include the ROC curves generated for every possible dataset using the three kernels (Linear, Gaussian, and Polynomial) attached to the classifier. To generate all the results, you need to run all the 8 scripts using the command mentioned above.

7. To extract the top positive and top negative examples of the classifier, for both Alt. 3' and Alt. 5' splicing types, run the following scripts using the command mentioned in step **6**.
   - **topexamplesAlt3.py**
   - **topexamplesAlt5.py**

   These scripts will store the results in separate folders. For instance, the **topexamplesAlt3.py** stores the results (sequences) in a folder named *"Alt3_Top_Positives"* for the top positive examples and *"Alt3_Top_Negatives"* for the top negative examples.

8. For the top **false** positive examples for both the Alt. 3' and 5' splicing examples, run the following scripts using the same command as mentioned above.
   - **TopFalsePositivesAlt3.py**
   - **TopFalsePositivesAlt5.py**

   These scripts will store the results in separate folders. For instance, the **TopFalsePositivesAlt5.py** stores the results (sequences) in a folder named *"Alt5_Top_False_Positives"*. The results are only for the Splice dataset since we can compare them to the the top positive examples generated in the previous step, using the weblogo tool.

9. To look for motifs in the top positive, top negative, and top false positive examples, you can use the **weblogo** tool (weblogo.berkeley.edu/logo.cgi). You just need to copy the sequences generated in the previous steps (**7** and **8**).