QPalma Documentation

July 2009

1 Intro

QPALMA is an alignment tool targeted to align spliced reads produced by "Next Generation" sequencing platforms such as *Illumina Genome Analyzer* or *454*. The basic idea is to use an extended Smith-Waterman algorithm for local alignments that uses the base quality information of the reads directly in the alignment step. Optimal alignment parameters i.e. scoring matrices are inferred using a machine learning technique similar to *Support Vector Machines*. For further details on *QPALMA* itself consult the paper [1]. For details about the learning method see [2].

2 Installation

The following installation notes assume that you are working in a *Linux*/*NIX environment. *Windows* or *Mac OS* versions do not exist and are not planned. *QPALMA* is written in C++ and Python and tested with the *gcc* compiler suite and python2.5.

2.1 Dependencies

QPALMA was designed to be as self-contained as possible. However in order to install *QPALMA* completely you will need:

- *SWIG*, the simple wrapper interface generator,

→ http://www.swig.org

- *numpy*, a python package for numeric computations,

→ http://numpy.scipy.org/

For training *QPALMA* you need *one* of the following optimization toolkits:

- CVXOPT (http://abel.ee.ucla.edu/cvxopt, Free)
- MOSEK (http://www.mosek.com, Commercial, Free student licenses available)
- CPLEX (http://www.ilog.com/products/cplex, Commercial)

2.2 Step by step installation guide

1. Run the setup_qpalma.sh script and follow the instructions given

In order to check your installation you can run the scripts:

- ./examples/run_example.sh
- ./examples/run_example2.sh

3 Working with *QPALMA*

In order to work with *QPALMA* you need the following data:

- Reads you want to align together with their seed positions,
- genomic sequences,
- splice site scores predicted for the genomic sequences.

Note: To obtain splice site scores we use the predictions of another tool of our group [3]. However this tool is not available yet but you can use any splice site prediction tool you prefer as long as you stick to the data format described below.
A typical run consists of

1. Find the seeds for your reads using another method (NOT VIA *QPALMA*). For example GenomeMapper [4].
2. Set your parameters (location of splice site score files etc...) configuration files. *QPALMA* uses one central configuration file. The parameters you can set in this file are described below.
3. Train a parameter vector (basically this is the scoring matrix for alignments) once for your organism.
4. (Optional: Try to filter reads that mapped unspliced but are spliced, see [1] for explanation)
5. Predict read alignments using the trained parameters.

When working with *QPALMA* you will use three commands:

- qpalma_train
- qpalma_filter
- qpalma_align

3.1 *QPALMA* Commands

For training *QPALMA* you need a configuration file, a file contraining a set of spliced reads in bed format (described below) and a result directory. The syntax is then:

qpalma_train <config file> <read_data.bed file> <result directory>

Once the training is finished the result directory will contain a parameter file called *parameters_final.pickle*. You can use this parameter then in the following steps.
The filtering step is started via the following command:

qpalma_filter <config file> <read_data.bed file> <result directory> <parameter file>

This reads are split up into two sets of putatively spliced and unspliced reads. Both files are stored in the result directory.
Alignments can be calculated via:

qpalma_align <config file> <read_data.bed file> <result directory>

Currently *QPALMA* supports the blat alignment format. *QPALMA* can be easily extended to support other alignment information formats. See the file *OutputFormat.py* in the QPalma module directory for details.

3.2 The configuration file

In order to run *QPALMA* successfully you will need to create a configuration file. All valid parameters for *QPALMA* are summarized below. You can also look at the example_configuration.conf file which is included in the distribution.
The parameters do not have to be in a particular order. In order to simplify explanations we have grouped them according to their particular semantics.
Via the variable "result_dir" you can specify where all of QPalma's data should reside. The first group consists of general settings for *QPALMA* :

- **perform_checks**- Enables some checks and debugging output.
- **platform** - IGA or 454 for Illumina Genome Analyzer or Roche's 454.
- **result_dir** - This is the toplevel result directory.

The second group includes parameters needed for accessing the sequence data and the splice site predictions:

- **genome_dir** - The location of the genomic sequence files.
- **acceptor_scores_loc** - The location of the acceptor scores files.
- **donor_scores_loc** - The location of the acceptor scores files.
- **genome_file_fmt** - A format string describing how valid file names are generated. See description below.
- **splice_score_file_fmt** - A format string describing how valid file names for the splice site scores are generated.
- **allowed_fragments** - A list of the form "[1,2,4]" describing the valid file name numbers. See description below.
- **half_window_size** - Given an alignment seed position for a given read we cut out the area [1500-seed_pos,seed_pos+1500].
- **output_format** - The output format can be either blat-like, ShoRe or mGene.
- **prb_offset** - We expect the quality values to be saved as ascii strings so the quality is calulated via ord(qchar)-prb_offset ($ord(\cdot)$ returns the numeric ascii value). For example an 'h' would correspond to quality 40.

The genomic sequence and the splice site scores file are accessed using the format strings given above. This works as follows: Suppose we have two chromosomes 1 & 4 we want to align to. The sequences are in flat files:

/home/fabio/genomic_sequence/chromosome_1.flat
/home/fabio/genomic_sequence/chromosome_4.flat

Then we set genome_dir to */home/fabio/genomic_sequence*, genome_file_fmt to "chromosome_%d.flat" and allowed_fragments to [1,4].
Parameters needed for training are:

- C - this is a parameter trading off between loss term and regularization (similar to C in SVM training).
- enable_quality_scores - You can enable or disable the use of quality scores via settings this parameter to True or False
- enable_splice_scores - You can enable or disable the use of quality scores via settings this parameter to True or False

- enable_intron_length - You can enable or disable the use of quality scores via settings this parameter to True or False
- optimizer - Either one of CVXOPT, MOSEK or CPLEX.
- numConstraintsPerRound - How many constraints per optimizer call should be generated (if unsure set to 50).

For training and prediction purposes you can set the number of support points for the piecewise linear functions of the scoring matrix. This effectively changes the total number of parameters of the model. That means that you can only predict with a parameter vector which was trained with the same settings. (We refer to piecewise linear functions as *plifs*.)

- numAccSuppPoints - The number of support points for the plif scoring the acceptor scores.
- numDonSuppPoints - The number of support points for the plif scoring the acceptor scores.
- numLengthSuppPoints - The number of support points for the plif scoring the acceptor scores.
- min_intron_len,max_intron_len - Set this to the range of minimum and maximum intron lengths.
- min_qual and max_qual - Set this to the range of possible quality values. (Illumina GA usually [-5,40]).
- min_svm_score and max_svm_score - As we work with normalized (in the range [0.0,1.0]) splice site scores this is usually 0.0 resp. 1.0.

The following last group of parameters includes "low-level" parameters. Just keep the default values of these parameters. Possible extensions of *QPALMA* may use them differently.

- remove_duplicate_scores
- print_matrix
- matchmatrixCols, matchmatrixRows
- numQualPlifs
- numQualSuppPoints
- totalQualSuppPoints
- anzpath
- iter_steps

4 File Formats / Specifications

This section introduces all formats and conventions that are assumed to be met by the users in order to make *QPALMA* work.

4.1 Format of the configuration file

The configuration file includes all settings *QPALMA* needs to perform an analysis. This includes paths to files where the raw data exists as well as settings which sequencing platform is being used, the number of cluster nodes to employ etc. .

Its values are in the form

$$\text{key} = \text{value}$$

and "#" for lines containing comments.

4.2 Training file format and internal representation

The read input files for *QPALMA* contain the read sequences with their quality as well as some information from the first mapping / seed region finding. We use the bed format `http://genome.ucsc.edu/FAQ/FAQformat#format1` as format for the input files. The bed format can be described as follows Each line corresponds to one short read. Each line consists of 12 tab-separated entries, namely:

1. *chrom* - The name of the chromosome (e.g. chr3)
2. *chromStart* - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. *chromEnd* - The ending position of the feature in the chromosome or scaffold. The chromEnd base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as chromStart=0, chromEnd=100, and span the bases numbered 0-99.
4. *name* - Defines the name of the BED line.
5. *score* - A score between 0 and 1000 (not used in *QPALMA*)
6. *strand* - Defines the strand - either '+' or '-'.
7. *thickStart* - (not used in *QPALMA*)
8. *thickEnd* - (not used in *QPALMA*)
9. *itemRgb* - An RGB value of the form R,G,B (e.g. 255,0,0) (not used in *QPALMA*)
10. *blockCount* - The number of blocks (exons) in the BED line.
11. *blockSizes* - A comma-separated list of the block sizes. The number of items in this list should correspond to blockCount.
12. *blockStarts* - A comma-separated list of block starts. All of the blockStart positions should be calculated relative to chromStart. The number of items in this list should correspond to blockCount.

Strand specific direction means that *QPALMA* assumes that the reads are already in their true orientation and the qualities as well.
Alignment information means that an alignment of a read to a genomic sequence. A mismatch is encoded as $[AG]$ if A is on the sequence and G on the read side. A gap is denoted by $[-X]$ resp. $[X-]$ denotinge a gap on the sequence resp. read side with $X \in A, C, G, T, N$.

4.3 Splice Scores

As mentioned before the splice site scores where generated using a tool described in [3]. If you would like to use your own splice site predictions you can create files according to the format *QPALMA* uses. This splice site scores format is described below:
For each canonical acceptor (AG) and donor site (GT/GC) *QPALMA* expects a score. The data is organized in files for each signal (acc/don) for each strand (+/-). The information on positions of canonical splice sites and the corresponding scores lies in separate files. Every chromosome or contig leads then to 8 files (acc/don, +/- and pos/score). The position and score files are raw binary files containing the orderd positions and the scores. The positions are stored as unsigned values and the scores as floats. Note that you have to be careful when working in an inhomogeneous cluster environment (endianness, data type size). The positions are 1-based and the assignment of positions and their scores is as follows: The acceptor score positions are the positions right after the AG and the donor score positions are the positions right on the G of the GT. For example:

...	3	4	5	6	7	8	9	10	11	...
...	w	g	t	x	y	z	a	g	v	...
...		0.2							0.3	...

We supply a script "spliceScoresConverter.py" (tools directory) for conversion of ascii to binary files. You can use this script as a template to make your own scoring information files.

4.4 Alignment result file

The result file for the blat-like output format consists of lines with the following columns:

- id - the unique read id of the input file
- chromosome/contig number
- strand
- start position of alignment in target
- qStart
- qEnd
- tStart
- tEnd
- number of exons
- qExonSizes
- qStarts
- qEnds
- tExonSizes
- tStarts
- tEnds
- number of matches in alignment
- number of gaps in alignment

5 Remarks

The *QPALMA* project is licensed under the GPL.
The official *QPALMA* project email address is:

qpalma@tuebingen.mpg.de

References

[1] De Bona F. and Ossowski S. and Schneeberger K. and G. Rätsch Optimal Spliced Alignment of Short Sequence Reads *ECCB 2008*

[2] Ioannis Tsochantaridis and Thomas Hofmann and Thorsten Joachims and Yasemin Altun Support Vector Machine Learning for Interdependent and Sturcutured Output Spaces *Proceedings of the 16th International Conference on Machine Learning*, 2004

[3] Sören Sonnenburg and Gabriele Schweikert and Petra Philips and Jonas Behr and Gunnar Rätsch Accurate splice site prediction using support vector machines *BMC Bioinformatics 2007, 8(Suppl 10):S7*, December 2007

[4] K. Schneeberger, S. Ossowski, and J. Hagmann, GenomeMapper Available under: http://www.1001genomes.org/downloads