Nanopore Results

Gregor Sturm

April 22, 2016

This report is a short outline of my work on Nanopore Basecalling from 2015-06 until 2016-01. It is the outcome of a student project at the Institute of Bioinformatics and Systems Biology (IBIS) of the Helmholtz-Zentrum München. Here, I presents the major results of this project.

Availability Analysis are available as iPython-Notebooks in the github repository `https://github.com/grst/nanopore`. The code of the novel python basecaller is available in the repository `https://github.com/grst/nanopore_pkg`.

Contact gregor.sturm@cs.tum.edu

1 Introduction to Nanopore Basecalling

The MinION by Oxford Nanopore Technologies (ONT) is a 'fourth-generation' sequencing device not larger than a portable hard drive. The device is connected with a customary PC and interacts with the Company's *Metrichor* cloud for *basecalling*, i.e. translation of the raw signal into the DNA sequence (Feng, Zhang, et al. 2015).

The functioning and usage of the device is fairly straight-forward: After DNA purification and preparation, the sample is put on a flow-cell. Every flow-cell consists of an "upper" and a "lower" chamber and is separated into 512 compartments, each of which is equipped with a protein pore connecting the two chambers. After a voltage is applied on the membrane, the negative charge of the DNA molecules is exploited to pull the them through the pore from the upper into the lower chamber. It is assumed, that six nucleotides are in the pore at the same time. Depending on the exact sequence of these six nucleotides, more or less cations can pass the pore in the opposite direction, which can be measured as the electric current.[1]

Translating these signals into the corresponding nucleotide sequence is data-intensive machine learning task which is commonly solved using Hidden Markov Models (HMMs) (Timp, Comer, and Aksimentiev 2012). Unfortunately, the Metrichor basecaller is closed source and works as a black box. Uploading data to the analysis cloud is not an option when (i) dealing with critical patient data and (ii) sequencing at places with an

[1] Information collated from various posts in the Nanoporetech wiki and Feng, Zhang, et al. 2015

unstable Internet connection. Moreover, an open source basecaller would be a valuable contribution to the field, as it can easily be applied and improved by other research groups.

I address this with the python package `npcaller`, an open source implementation of a HMM-based basecaller for the MinION.

All data used in this work is based on the SQK-MAP006 sequencing kit. ONT has recently announced a new flow cell and basecaller (see section 5). It is likely that `npcaller` will not work with the future version of MinION.

2 Overview over the ONT Basecalling Pipeline

2.1 File formats

All data generated during the MinION basecalling pipeline is stored in `fast5` files, which are built upon the `HDF5`[2] data storage format. `HDF5` contains a directory-like hierarchical structure. Every directory can have attributes and/or contain data-tables. The HDF group provides software for viewing and manipulating `HDF5`-files[3]. Moreover, there is the python library `h5py` for programmatic interaction with the contents[4]

2.2 The Pipeline

The Metrichor Basecalling Pipeline consists of the following major steps[5]. Check figure 1 for an illustration of the data.

1. **Raw data.** Sensors measure the electric current over the pore at a fixed sample-rate, which is 3000 Hz for SQK-MAP006.

2. **Event calling.** This step serves for reducing the amount of data. Consecutive samples with similar mean are combined into one *event* or *squiggle*. An event is a tuple `(start_time, length, mean, stdv)`. The resulting set of squiggles is often referred to as *Squiggle space*.

3. **(1D) Base calling.** The events are converted to a nucleotide sequence using a HMM (Timp, Comer, and Aksimentiev 2012). Other than in the paper, by now the MinION uses 6-mers. The HMM has therefore $2^6 = 4096$ states.

4. **2D base calling.** To improve accuracy, the MinION supports 2D basecalling. Here, forward and reverse strand of the DNA are ligated together using a hairpin adapter. Like that, the same sequence is read twice, once in sense and once in anti-sense direction. In 2D base calling, the two 1D reads are aligned and uncertainities resolved.

[2] `https://www.hdfgroup.org/HDF5/`
[3] `https://www.hdfgroup.org/products/java/release/download.html`
[4] `http://www.h5py.org/`
[5] Reverse engineering of *Metrichor* data files and posts in the internal wiki

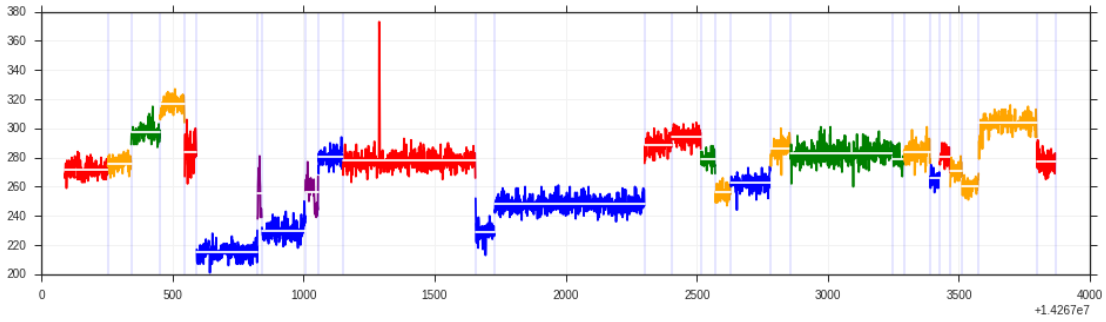

Figure 1: Exemplary raw Data divided into events. The measured current (y) is plotted against the time (x). Each section separated by two vertical lines represents one *event*, consisting of many individual current samples measured by the sensor. The white, horizontal lines represent the mean of the respective event.

3 The Basecalling Package

I developed the python package `npcaller` which provides alternative basecalling for the MinION. The software consists of three major parts:

1. Model generation
2. Basecalling
3. Validation.

The software depends on the python packages `h5py`, `ghmm`, `numpy`, `pandas` and `pysam` which are all available trough the python package index[6] and can be installed using `pip install`. Additionally, the package uses the following command line software, which has to be available on the machine:

- **graphmap** (Sovic, Sikic, et al. 2015) is a novel, specialized aligner for long, error-prone reads as they are generated by nanopore sequencing devices and is available through github[7].
- **samtools** (Li, Handsaker, et al. 2009) is a commonly used tool for handling alignment files. It is available on `http://samtools.sourceforge.net/`
- **poretools** (Loman and Quinlan 2014) is a toolkit for analyzing nanopore sequencing data. It is available on github[8] or through `pip install poretools`. Poretools is only required for the validation part of the software.

[6] `https://pypi.python.org/pypi`
[7] `https://github.com/isovic/graphmap`
[8] `https://github.com/arq5x/poretools`

3.1 Documentation

This documentation aims at explaining what's going on under the hood. For description of the program calls, see the help pages of the scripts themselves.

3.1.1 Model generation

In the first step, one needs to retrieve the parameters (i.e. mean and standard deviation for each kmer) for the HMM. Template and complement DNA strands show a different base calling behaviour, therefore two models are generated. The package provides two scripts for the model generation:

- `model_from_fast5.py`: The parameters used by *Metrichor* are saved as metadata in the fast5-files. This script extracts this data and saves it as a model.
- `model_from_alignment.py`: This script computes the parameters by aligning a set of basecalled (e.g. by *Metrichor*) fast5-files to a reference genome. For every kmer $\in \{\text{AAAAAA}, \ldots, \text{TTTTTT}\}$, the mean and standard deviation of the current among all occurrences in the reference genome is calculated. As this script involves the aligning of many reads, it is quite computationally expensive and provides the option to run multithreaded.

3.1.2 Basecalling

`basecall.py` takes a list of (uncalled) fast5 files and model files as input and writes the resulting nucleotide sequence to a fasta file. Template and complement model files can be specified separately. If only one of them is specified, only only the corresponding sequence will be called. As this process is CPU-intensive, the script provides the option to run multithreaded.

3.1.3 Validation

`validate.py` comes with four sub-programmes:

- `randomize` takes a fasta file as input and mutates all nucleotides assuming a uniform distribution. The original sequence lengths are kept. This is useful to compare the performance of the parser to a completely random model.
- `align` uses `graphmap` to align the reads in a fasta file to the reference genome.
- `filter` is a simple tool that filters fasta files for entries which contain certain keywords. For the evaluation, I use this tool to split the fasta files in template and complement.
- `stats` takes a reference genome and an alignment file (e.g. generated by the `align` command) and creates a table with simple statistics. These include the number of mapped reads, the number of reads aligned with a significant E-Value, the overall

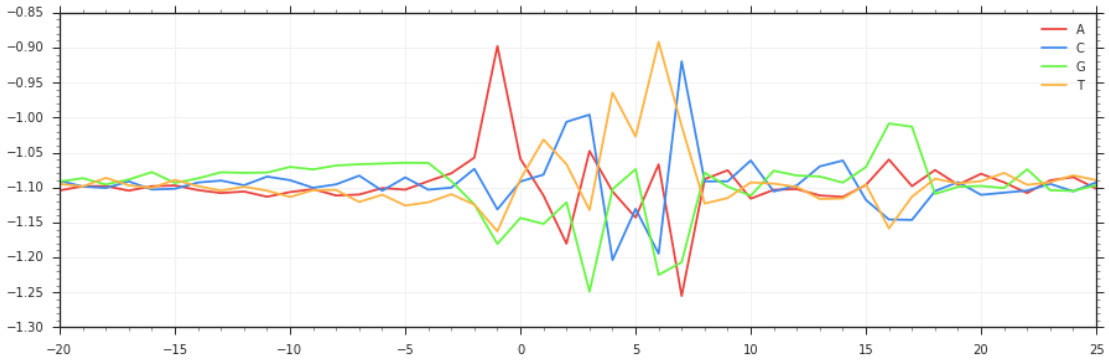

Figure 2: Long range signal context of the template strand. The median deviation of the model (y-axis) is plotted against the occurence of a certain nucleotide at a specific distance from the kmer (x-axis). The region from 0 to 6 refers to the actual 6-mer, which is taken into account by the model.

alignment score, the overall editdistance to the reference and the count of SNPs, insertions and deletions.

3.2 Evaluation

Table 1: Stats...

feature	Metrichor			npcaller			random model		
mapped reads / total reads	100	100	1.	100	100	1.	63	100	.63
significant reads / mapped reads	99	100	.99	96	100	.96	0	100	0
mapped nts / total nts	865397	970301	.89	847229	981932	.86	474349	981932	.48
editdistance / alignment length	311257	948375	.33	386821	945767	.41	304026	504365	.60
alignment score / alignment length	1802584	948375	1.90	1098801	945767	1.16	-93499	504365	< 0
SNPs / mapped nts	123375	865397	.14	153580	847229	.18	130710	474349	.28
insertions / mapped nts	104806	865397	.12	134611	847229	.16	143254	474349	.30
deletions / mapped nts	82978	865397	.10	98538	847229	.12	30016	474349	.6

4 Context Sensitive Basecalling using an Iterative Approach

As demonstrated in a Wiki-Post[9] and in my *IPython Notebooks* in `04_error_correction`, the 6-mer model does not model all information available from the data. Additional signals are detected about 15 nt downstream, e.g. the abundance of Guanine at positions +14 and +15 tend to lead to a higher current level for a given 6-mer in the pore. (see figure 2).

[9] `https://wiki.nanoporetech.com/display/DS/Spooky+action+at+a+distance+-+long+range+signal+context`

This information cannot be modeled using a HMM, as it requires information about nucleotides which are *downstream*, i.e. not called yet. As a workaround I tried an iterative basecalling model (illustrated in fig. 3):

1. The HMM predicts the sequence from a given squiggle space as usual.
2. An SVM corrects the squiggle data given the predicted sequence.
3. The HMM re-predicts the sequence from the corrected squiggle space.
4. continue with step 2 until satisfied.

4.1 training of the SVM

The training dataset was constructed from sequences successfully aligned to a reference genome with squiggle data available. For each position in the reference genome, I took only the *median deviation from the model* ($\text{mean}_{\text{kmer}} - \text{mean}_{\text{model}}$) to avoid a bias towards regions with higher coverage. The mean, stdv and the nucleotides ± 20 around the 6-kmer were used as features. I encoded nucleotides as binary vector, i.e. $A = [1, 0, 0, 0], C = [0, 1, 0, 0]$ and so on. I used the *median deviation from the model* as target value. The SVM achieved a pearson correlation coefficient (PCC) of 0.46 and a mean absolute error (MAE) of 0.62 in a 3-fold cross validation. For comparison: The mean *median deviation from the model* over all genomic positions is 0.71.

4.2 result

This approach lead to a significant increase in computation time, but did not improve the results (result not shown). This can either indicate insufficient performance of the SVM or the infeasibility of the entire approach.

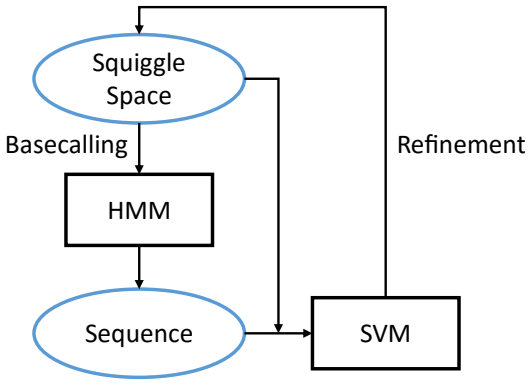

Figure 3: Architecture of the iterative base-calling approach

5 Future development

The nanopore sequencing is under very rapid development. Since the completion of this project, Oxford Nanopore has announced an open source version of their original Basecaller, developed a new generation of flow cells and a new kind of basecaller which is not based on HMMs any more and will also be released in an open source version. For more Information I recommend watching the Hangout by Oxford Nanopore CTO Clive Brown.[10]

References

Feng, Yanxiao et al. (2015). "Nanopore-based fourth-generation DNA sequencing technology". In: *Genomics, Proteomics Bioinforma.* 13.1, pp. 4–16. ISSN: 22103244. DOI: 10.1016/j.gpb.2015.01.009.

Li, Heng et al. (2009). "The Sequence Alignment/Map format and SAMtools". In: *Bioinformatics* 25.16, pp. 2078–2079. ISSN: 13674803. DOI: 10.1093/bioinformatics/btp352.

Loman, N. and a. Quinlan (2014). "Poretools: a toolkit for analyzing nanopore sequence data". In: *bioRxiv* 30.23, p. 007401. ISSN: 1367-4803. DOI: 10.1101/007401.

Sovic, Ivan et al. (2015). "Fast and sensitive mapping of error-prone nanopore sequencing reads with GraphMap". In: *bioRxiv*, p. 020719. ISSN: 2041-1723. DOI: 10.1101/020719.

Timp, Winston, Jeffrey Comer, and Aleksei Aksimentiev (2012). "DNA base-calling from a nanopore using a viterbi algorithm". In: *Biophys. J.* 102.10, pp. L37–L39. ISSN: 00063495. DOI: 10.1016/j.bpj.2012.04.009.

[10] Available in the Nanoporetech wiki https://wiki.nanoporetech.com/x/iD00Ag