Spalter: A Meta Machine Learning Approach to Distinguish True DNA Variants from Sequencing Artefacts

Till Hartmann
Genome Informatics, Institute of Human Genetics, University of Duisburg-Essen, University Hospital Essen, 45122 Essen, Germany
Till.Hartmann@uni-due.de
https://orcid.org/0000-0002-6993-347X

Sven Rahmann
Genome Informatics, Institute of Human Genetics, University of Duisburg-Essen, University Hospital Essen, 45122 Essen, Germany
Bioinformatics, Computer Science XI, TU Dortmund, Dortmund, Germany
Sven.Rahmann@uni-due.de
https://orcid.org/0000-0002-8536-6065

Abstract

Being able to distinguish between true DNA variants and technical sequencing artefacts is a fundamental task in whole genome, exome or targeted gene analysis. Variant calling tools provide diagnostic parameters, such as strand bias or an aggregated overall quality for each called variant, to help users make an informed choice about which variants to accept or discard. Having several such quality indicators poses a problem for the users of variant callers because they need to set or adjust thresholds for each such indicator. Alternatively, machine learning methods can be used to train a classifier based on these indicators. This approach needs large sets of labeled training data, which is not easily available.

The new approach presented here relies on the idea that a true DNA variant exists independently of technical features of the read in which it appears (e.g. base quality, strand, position in the read). Therefore the *nucleotide separability* classification problem – predicting the nucleotide state of each read in a given pileup based on technical features only – should be near impossible to solve for true variants. Nucleotide separability, i.e. achievable classification accuracy, can either be used to distinguish between true variants and technical artefacts directly, using a thresholding approach, or it can be used as a meta-feature to train a separability-based classifier. This article explores both possibilities with promising results, showing accuracies around 90%.

2012 ACM Subject Classification Applied computing → Sequencing and genotyping technologies, Computing methodologies → Unsupervised learning, Computing methodologies → Supervised learning by classification

Keywords and phrases variant calling, sequencing error, technical artefact, meta machine learning, classification

Digital Object Identifier 10.4230/LIPIcs.WABI.2018.13

1 Introduction

Apparent differences between a reference genome and a sequenced sample may either be due to biological variance or to technical artefacts (e.g. caused by flawed library preparation or sequencing machine bias). For downstream analysis, e.g., finding disease-related genes, or personal disease risk assessment, it is important to be able to distinguish between these two

kinds of causes. Therefore, variant callers, such as the one in GATK [3], include measures, such as strand bias or overall variant quality, to assess the likelihood of a technical sequencing or processing artefact, and conversely, to provide a measure of confidence for each detected variant. For this, variant callers typically rely on assumptions about error distributions or require prior or expert knowledge [1]. Since these assumptions may not hold for every sequencing and processing pipeline or no prior knowledge may be available, we develop a different approach. For simplicity, our presentation focuses on single nucleotide variants (SNVs). In principle, it can be applied to longer variants (e.g. short indels) at the expense of a more complex technical implementation.

The idea is that a true DNA variant exists independently of technical features of the read where it appears, e.g. base quality, position in the read, mapping quality or strand of the read. Therefore the classification problem to predict the nucleotide state of each read in a given pileup based on only technical features should be unlearnable for a true variant. In other words, high classification accuracy on the *nucleotide separability* classification problem on a pileup suggests that the called variant is a technical artefact. Separability alone can be used to decide between a true DNA variant and a technical artefact. This yields a method that does not need labeled training data with known true variants. If ground truth information about which pileups exhibit true variants (and which exhibit technical artefacts) is available, it is possible to train a machine learning model on the separability values, resulting in a meta machine learning task. We here explore both possibilities.

Our work, called `spalter`, differs from related approaches that also use machine learning techniques to classify pileups, like SNPSVM [4] or DeepVariant [5], in that it does not necessarily need labeled training data. `Spalter` employs classification methods for computing separability values, which *can*, but do not need to be used for training another classifier using labeled target data (true variant vs. technical artefact). If such is not available, a thresholding rule can be applied on separability at the cost of accuracy. In contrast, SNPSVM learns only the latter classifier in conjunction with 'hand-crafted' features (such as 'base quality mean' or 'read position mean'). DeepVariant encodes information in RGBA images (in newer versions, data is encoded in multichannel tensors instead), which are then used to train a convolutional neural network (CNN). Training the CNN is computationally expensive, especially without dedicated graphic processing units (GPUs) or tensor processing units (TPUs). `Spalter`, in contrast, is neither restricted to a single model architecture nor does it require extensive CPU or memory resources.

2 Idea

For an arbitrary site in the reference sequence, let $\mathcal{B}$ be the multiset of bases in that site's pileup, and let s be the base in the reference sequence at that site. If some bases in $\mathcal{B}$ disagree with s, i.e. there is a certain (minimum) fraction of bases which are not equal to s, we ask whether this is due to technical artefacts or biological variation.

However, there is more information available for each base of a pileup than just the nucleotide represented, and we gather and store the purely *technical* information that should be independent of the presence or absence of a biological variant, such as the *base quality* of a base (assigned by the sequencer), the *strand* of the read, or whether the read is the first or second of a read pair.

If, for a given pileup, it is possible to *separate* bases that match the respective reference base from those that do not by making use of the aforementioned technical features and simple classifiers, we are inclined to believe that a technical artefact is present (depending

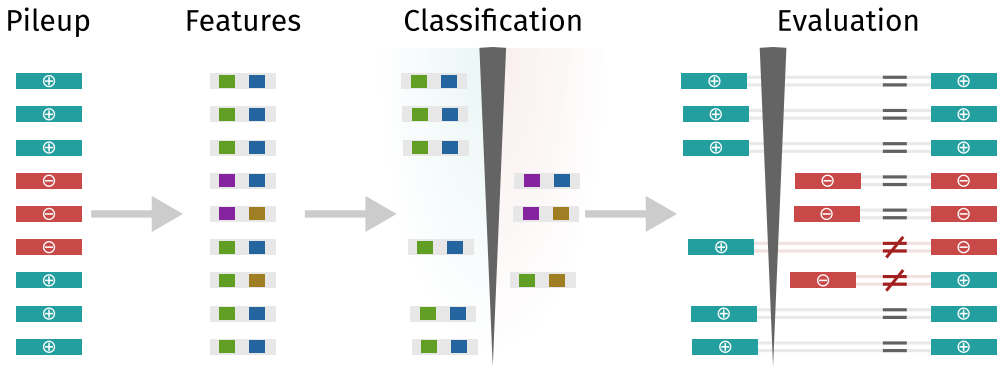

Figure 1 Computing separability profiles. For a given pileup, labels ⊕ (base matches reference base) and ⊖ (base does not match reference base) are determined for every base (rectangle) in the pileup (stack of rectangles). Also, for each base a technical feature vector (grey rectangle with colored squares, one square for each feature) is derived from its read and accompanying auxiliary data (such as base quality or read strand). Using these labels and feature vectors, a simple classification model is trained and its training accuracy is evaluated and interpreted as *separability*.

on the degree to which separation was successful).

The reasoning behind this assumption is that the sequencing process is consistent regardless of whether samples contain SNVs or not. This implies that it should not be possible to tell reference from alternative bases by looking at technical features only. Hence, if it is indeed possible to tell them apart, a pattern was recognized which is most likely explained by technical biases during some stage of the sequencing process.

For a given pileup's bases $\mathcal{B}$, we derive base labels $\oplus$ if the base matches the reference base s and $\ominus$ if it does not. We then train a classifier to predict the base label only from the technical features (which should work badly for true biological variants). The *training accuracy* of this classifier is called *separability* of $\mathcal{B}$ (Figure 1).

A low separability suggests that there is a true variant in $\mathcal{B}$. As a first step, the separability of a pileup can thus be used *directly* for decision making (true variant or technical artefact), or, in a further step, as a feature for training a classifier based on separability. The first step does not need labeled training examples (in the sense of manually annotated 'true SNV' and 'technical artefact' labels; of course, a supervised classification problem is solved based on known nucleotide status for each read). The second step does need labeled training examples and uses (meta-)features (separability) from the classification problem of the first step. We hence call `spalter` a pseudo-supervised meta machine learning approach.

3 Method

To disambiguate between the different kinds of features used throughout the paper, features associated with bases and reads (such as base quality or read strand) are called *technical* features, while features constructed as explained in subsection 3.1 are called *meta features* or *separability profile*.

Algorithm 1 Building a separability profile for a single pileup `p`, using a set of classifiers $\mathcal{C}$ and a set of feature combinations $\mathcal{F}$. A $\mathcal{C} \times \mathcal{F}$ matrix `S` with separability values is returned.

```
1  function separability_profile(p, 𝒞, ℱ):
2      S = empty real 𝒞 × ℱ matrix
3      ℓ = [base == reference_base for base in p.bases]
4      f = extract_features(p)          # technical features
5      for (C, F) in 𝒞 × ℱ:             # for each combination (each F ∈ 2^f)
6          M = C.train(F,ℓ)             # train model M
7          ℓ′ = M.predict(F)            # predict labels ℓ′
8          s = rel_accuracy(ℓ, ℓ′)      # evaluate training accuracy (see text)
9          S[C,F] = s                   # store s in meta feature matrix
10     return S
```

3.1 Computing the separability profile

Since the idea is to discern between SNVs and technical artefacts on the assumption that they behave differently with respect to separability, the first step is to build a separability profile[1] for each pileup. Algorithm 1 outlines the routine for building a separability profile for a single pileup `p` for a given set $\mathcal{C}$ is a set of classifiers and a set $\mathcal{F}$ of feature combinations from a basic set f of technical features.

We assume that the given pileup `p` has a given minimum alternative allele fraction (default: ≥ 0.1) and a given minimum coverage (default: ≥ 10)[2]. A label vector ℓ is built by comparing every base in the pileup to the reference base at the pileup's position (line 3). By default, the following four technical features are extracted *for each base* in the pileup (line 4), resulting in a technical feature matrix of dimension number of reads in `p` times number of features:

1. Read pair index (binary): Is the read containing the base the first or the second of its read pair?
2. Read strand (binary): Is the read on the forward or the reverse strand?
3. Neighbour base quality (continuous): Average base quality of the respective base and its direct neighbours (from the same read).
4. End proximity (continuous): Distance of the base to either end of its read, transformed to be in $[0, 1]$. The closer the base to the center of its read, the lower the end proximity and vice versa.

In order to build a separability *profile*, we train one model (lines 6–9) for each combination of classifier and feature subset (line 5) in order to obtain information about the difficulty of separation (a) with a limited set of features and (b) with different classifiers. Note that increasing the number of technical features or increasing the number of classifiers, while increasing the level of detail of separability profiles, will not change the general appearance and behaviour of separability profiles; they will always exhibit a gradual increase of accuracy with both increasing number of technical features used for training and classifier complexity, as can be seen in Figure 2. Calculating separability is straightforward: After training a model (line 6), its prediction (line 7) is used to calculate *relative* training accuracy $s = \frac{a-p}{1-p}$, where a is the model's accuracy (fraction of correct predictions) and p is the accuracy of a

[1] Initially, we assumed that a single separability value might suffice; however, the resulting accuracy was not satisfactory.

[2] Since we are training classifiers, these defaults ensure that there are at least 10 examples available for training, albeit with heavily skewed class sizes $(1 : 9)$.

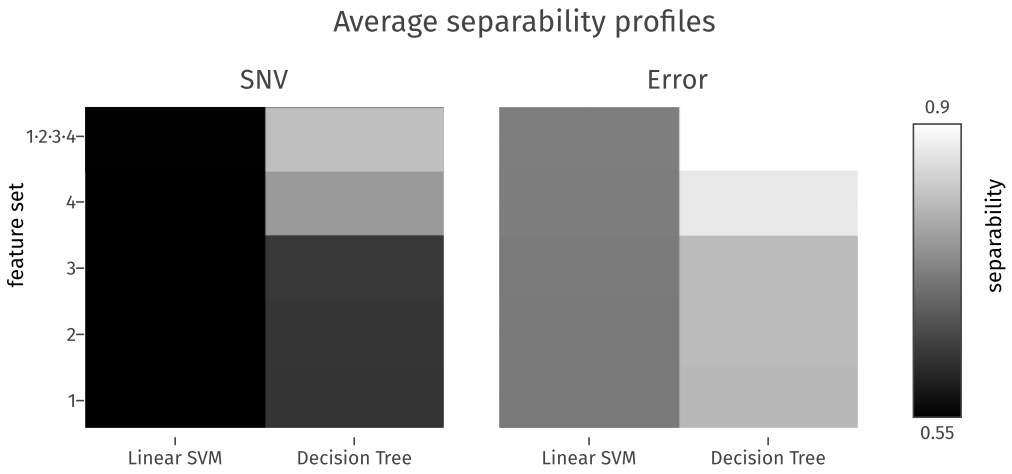

Figure 2 Average separability profiles for 2327321 SNV and 5618509 erroneous sites, using 2 classifiers and $4 + 1$ different feature combinations (numbers correspond to those of features listed in subsection 3.1). Each cell corresponds to the training accuracy of one classifier (x-axis) trained with a certain subset of features (y-axis). On average, separability is higher for erroneous sites than for SNV sites.

'largest class' classifier (line 8), i.e. an accuracy of p is trivial to achieve. Finally, the resulting relative accuracy value – called *separability* – is stored in the separability profile `S` (line 9).

3.2 Distinguishing between errors and SNVs

Having built a separability profile for each pileup, the task is to utilise these to distinguish between sites with technical artefacts and sites with true SNVs.

One way to do this is to use a thresholding approach: Summarise separability profiles (per pileup) for example by averaging the values to a single mean separability value and apply a threshold to the averages. Any site whose mean separability is below the threshold is deemed a likely true variant site, while values above the threshold imply a site with technical sequencing artefacts (the easier it is to separate disagreeing bases in a pileup, the more likely becomes a technical artefact).

Another approach lies in using separability profiles as features to train a classification model; this necessitates labeled data, i.e. information about whether a pileup/site is deemed a true SNV site or not.

Both approaches have been implemented in `spalter`. For the first approach (classification on mean separability values with a single threshold), the threshold defaults to 0.8; for the second approach (classification on separability profiles with an arbitrary supervised classifier), the classifier defaults to a random forest consisting of 19 trees with a maximum depth of 4 (but can be set to an arbitrary supervised classifier). We provide pre-trained default models, which have been evaluated as described in section 4.

4 Evaluation

In order to evaluate `spalter`, we used datasets from CHM-eval/syndip [2], a benchmark specifically designed for small variant callers. Among other things, it consists of the *de*

Table 1 Different separability thresholds for different bins of read depths lead to an overall increase in accuracy compared to the accuracy achieved using a single global threshold.

read depth	20–24	25–29	30–34	35–39	40–44	45–49	50–54	55–59
separability threshold	0.2	0.81	0.82	0.82	0.82	0.82	0.81	0.80
accuracy	92.9%	89.6%	91.8%	92.0%	94.7%	95.1%	96.0%	95.8%

novo assembly of PacBio sequences of two different complete hydatidiform mole (CHM) cell lines (i.e. completely homozygous) as well as Illumina HiSeq-X10 reads of a 1:1 mixture of the DNA of the two cell lines. We applied different methods to the CHM-eval dataset with known SNV sites from `full.37m.vcf.gz` (indels excluded). We also used the NA12878 dataset from Genome In A Bottle to be able to compare spalter with SNPSVM. Note that we could not successfully run SNPSVM on either dataset[3], which is why we use the results given in [4] instead, where a dataset was obtained based on the NA12878 exome (i.e. in-house sequencing on an Illumina HiSeq 2000, variants for training determined using GATK's best practices guidelines and filtering these with data from the 1000 Genomes project).

Separability profiles were computed using different subsets of the technical features (every single feature and all features) listed in subsection 3.1. The classifiers used for the nucleotide separability problem were logistic regression (using the stochastic gradient descent implementation in Rust, available from the `rustlearn` crate[4]), linear support vector machines, radial basis function support vector machines with default parameters and decision trees with a maximum depth of 4 (always using their `rustlearn` implementations).

In a first step, we use *only decisions based on thresholded average separability values*. Average separability values are computed as the average over the different feature combinations and the decision tree classifier as shown in Figure 2 (right column of panels). With a threshold of 0.8 (which is not an optimised threshold), an accuracy of 89% is obtained for the CHM-eval dataset, while the accuracy for the GIAB dataset is 87.2%. For CHM-eval, in detail, we discover 85.2% of the true SNVs, and 91.7% of our SNV calls are correct; we discover 93.6% of the true technical artefacts, and 88.4% of our artefact calls are correct with this threshold. For GIAB, we discover 65.3% of the true SNVs, and 95.3% of our SNV calls are correct; we discover 98.4% of the true technical artefacts, and 84.8% of our artefact calls are correct with this threshold.

Since SNPSVM did not finish successfully on either dataset, the comparison between the recall and precision values for SNPSVM reported in [4] and the values for spalter reported above is not a strong one, especially considering that the number of variants examined is roughly $4 \cdot 10^4$ for SNPSVM and $3 \cdot 10^6$ for spalter. Still, performance on the CHM-eval dataset is comparable to SNPSVM's performance, which achieved comparable recall with higher precision. We note, however, that SNPSVM needs labeled training data to achieve this performance, while our results are based on simple separability thresholding and an educated guess for the threshold. Note that it may well be possible to estimate a better threshold from the resulting average separability distribution automatically.

The accuracy can be improved if different (optimal) thresholds are chosen based on the pileup's read depth (coverage) and labeled training data, as shown in Table 1.

Similarly, higher accuracies can be obtained if separability is used as a feature in a meta-classification task: The classifiers used for the decision problem (true variant / technical

[3] This is probably due to a 0/1 indexing error in its source.
[4] see `https://maciejkula.github.io/rustlearn/`

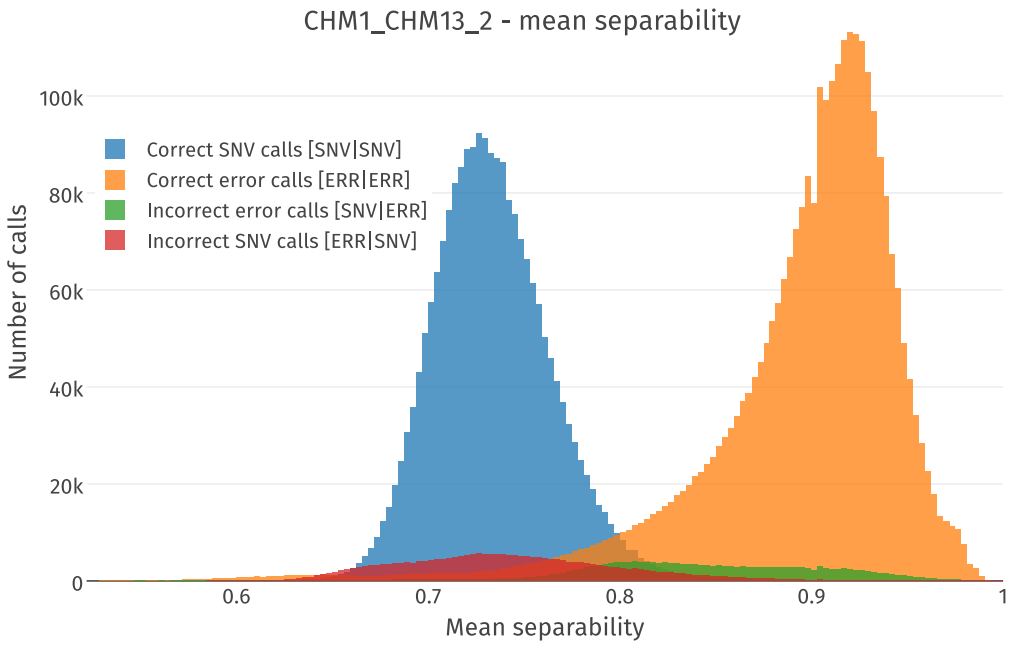

Figure 3 Histograms of mean separability values for dataset CHM1_CHM13_2. Correctly called SNVs (blue) and errors (orange) are clearly distinct, with the latter exhibiting higher mean separability values. Incorrectly called SNV sites (red) have a mean separability distribution with its mean close to that of the distribution of correct SNV calls, while incorrectly called errors (green) occur most often around the intersection of correct SNV and error calls.

artefact) were random forests and decision trees (with maximum depth 4). Training sample size was varied between 10^i for $i \in \{4, 5, 6\}$, where 10^4 data points correspond to roughly 0.001% of available data. We found that both training and testing accuracy on CHM-eval were extremely consistent (for all tested combinations between 87.4% (lowest minus one standard deviation) and 92.9% (highest plus one standard deviation)), with the mean at 91.5%, which is higher than the overall accuracy using simple thresholding.

We conclude that (a) the introduced meta features are indeed useful for training classification models, and (b) a tiny fraction of the data is sufficient to train a suitable classification model.

To explicitly show that the claim 'error likelihood increases with separability' holds, we examined the empirical distributions of mean separability values as shown in Figure 3. Indeed, sites that exhibit technical artefacts tend to have higher mean separability values than true SNV sites.

5 Discussion

Using meta features (separability profiles) to train classifiers to distinguish technical artefacts from true SNVs leads to accurate models, especially given the fact that only abstract separability information is used for training. When using random labels for training instead, accuracy is close to 50%, which indicates that separability profiles do not facilitate overfitting. It remains to be determined if models derived from separability profiles are 'portable', i.e.

can be applied to different datasets successfully (without adjustments to the models).

The advantage of this approach is that separability as a single abstract notion can be used to distinguish between errors and SNVs, while the *only* assumption made is that errors grow increasingly more likely with increasing separability. This also entails that it is possible to increase classification accuracy by either making additional assumptions or manually adding features (such as alternative allele fraction). Here we wanted to show that separability in itself is a powerful feature and did not further explore this possibility.

Also, the extraction of meta features (computation of separability profiles) presented here is an elaborate way to automatically derive *constant size* feature matrices for any kind of data with varying item sizes (here: differing numbers of reads between pileups) and suspected intrinsic labels (here: base matches / does not match reference base), i.e. the approach is not restricted to this particular application in bioinformatics.

Another interesting avenue to explore is that, while there are (combinatorially) many different ways to assign labels to bases in a pileup, the intrinsic one (base matches / does not match the reference base) is the most 'natural'. So it may be viable to try to find errors not site-wise but read-wise by comparing separability regarding different label vectors (e.g. ones where a single base has its label flipped).

The source code of `spalter` can be obtained from `https://bitbucket.org/tillux/spalter/`.

References

1 Erik Garrison and Gabor Marth. Haplotype-based variant detection from short-read sequencing. *arXiv preprint arXiv:1207.3907*, 2012.

2 Heng Li, Jonathan M Bloom, Yossi Farjoun, Mark Fleharty, Laura D Gauthier, Benjamin Neale, and Daniel MacArthur. New synthetic-diploid benchmark for accurate variant calling evaluation. *bioRxiv*, 223297, 2017.

3 Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, and Mark A. DePristo. The genome analysis toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, 20(9):1297–1303, 2010. `doi:10.1101/gr.107524.110`.

4 Brendan D O'Fallon, Whitney Wooderchak-Donahue, and David K Crockett. A support vector machine for identification of single-nucleotide polymorphisms from next-generation sequencing data. *Bioinformatics*, 29(11):1361–1366, 2013.

5 Ryan Poplin, Pi-Chuan Chang, David Alexander, Scott Schwartz, Thomas Colthurst, Alexander Ku, Dan Newburger, Jojo Dijamco, Nam Nguyen, Pegah T. Afshar, Sam S. Gross, Lizzie Dorfman, Cory Y. McLean, and Mark A. DePristo. Creating a universal SNP and small indel variant caller with deep neural networks. *bioRxiv*, 2018. `doi:10.1101/092890`.