DeepSV: a deep learning based variant scorer

Félix Raimundo
Department of Computer Science
École Polytechnique
Palaiseau, France
`felix,raimundo@inria.fr`

February 7, 2018

Abstract

In this paper, we present DeepSV, a deep learning based method for classifying potential genetic variants into true variants or non false positive.

Indeed in the task of variant calling, the general pipeline consists of two major tasks, first identifying regions that are likely to contain a variant (candidates), and the second is to classify these candidates as either real variants or false positives.

We identified a region of fixed size that contains enough signal to make classification with deep learning method.

1 Properties of the data

Property 1 The distance between two reads of a pair, called insert size, follows a normal distribution.

Property 2 The length of the reads, called read size, follows a normal distribution.

1.1 Types of signals

There are basically 4 signals used for structural variant calling:

- Assembly based, which are rare outside of long read technologies
- Read count, rarely used alone and only works for imbalanced variants. It basically consists of looking at the number of reads at a particular location, more reads implies duplication, less reads implies deletion.
- Split reads, use unmapped reads and try to map them around suspected variants (by splitting them in two or three if needed).

- Paired reads, use pairs of mapped reads. By looking at the distance between them and their orientation we can guess when there are likely SVs

In the case of deepSV we plan on using PR and RC signals for scoring the variants, SR will only be used for breakpoint location.

1.2 Anomalous paired reads

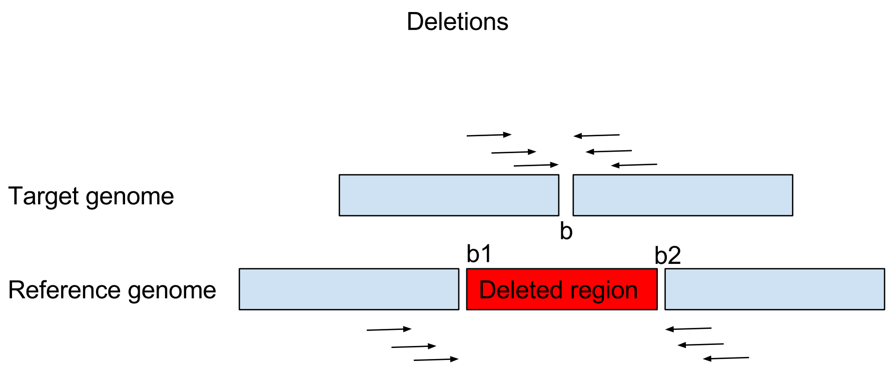
Anomalous read pairs (ARP) are read pairs that do not have the normal orientation or distance. These reads need to happen around a breakpoint, and have one read on each side.

Indeed, if one read was on the breakpoint, it could not be mapped (it can technically be only a little bit on the breakpoint, depending on the mapping quality).

Respectively, if both reads were on a side of a breakpoint, they would be mapped normaly and would thus not be caught as anomalous.

Lemna 1 With ARPs there is, by construction, a breakpoint between the two reads.

1.2.1 Deletion



Here reads around the breakpoint b will be mapped farther apart than expected on the reference genome.

Lemna 2 Once again by definition of ARP, there is one read on each side of the breakpoint b, thus all reads in ARPs will be at most `read_size + insert_size` away from the breakpoint, in the target genome.

Remembering properties 1 and 2, this tells us that we can have a boundary of fixed size in which we can expect to catch all ARPs around a breakpoint.

Lemna 3 Respectively, should we have a group of ARPs supporting the same deletion we can say with a known confidence interval, that most ARPs to the left of the breakpoint b1 will be contained within `insert_size + read_size` of the beginning of the leftmost supporting read. The same follows for the reads on the right of the breakpoint b2.

1.2.2 Inversion

```
target: ->->->->->->->-> b1 <-<-<-<-<-<-<- b2 ->->->->->->->->
ref:    ->->->->->->->->    ->->->->->->->    ->->->->->->->->

where b1 and b2 are the two breakpoints
```

Here ARPs are pairs where one read is in the inversed region and the other is in the regular one (i.e. either one read left of b1 or one read right of b2). Indeed reads between b1 and b2 will be mapped properly (because BWA is not sensitive to the order of the reads).

Reads in the inversed region will be mapped in the wrong direction, and farther away than expected (as the region is inversed they will be moved closer to the other breakpoint).

We can see that if one read is left of b1, the other one will be mapped close to b2 inside the inversed region and in the wrong direction, and that if one read is to the right of b2, the other one will be mapped close to b1 in the inversed region and in the wrong direction too.

We can also see by using properties 1 and 2, that reads outside the inversed region will be at most `insert_size + read_size` away from the breakpoints and that the same is true for reads inside the inversed region.

Lemna 4 ARPs supporting an inversion will be distributed around the two breakpoints within `2*(insert_size + read_size)`.

Lemna 3 Respectively, should we have a group of ARPs supporting the same inversion we can say with a known confidence interval, that most ARPs around the left breakpoint will be contained within `2 * (insert_size + read_size)` of the beginning of the leftmost supporting read. The same is trivially true for the right breakpoint.

1.2.3 Tandem Duplication

```
target: -------- b1 +++++++ b +++++++ b1 ----------
ref:    --------    d1   +++++++   d2    ----------
```

Here ARPs, are pairs that have one read on each side of breakpoint b, indeed read pairs fully contained between b1 and b or b and b2, will be mapped properly on the duplicated region.

Lemna 6 ARPs are aligned in the wrong direction (`<- ->` instead of `-> <-` for example), and inside the duplicated region, one within `insert_size + read_size` to the right of d1, the other within `insert_size + read_size` left of d2. The read count in the duplicated region will be about the double of outside the duplicated region.

Lemna 7 It follows that should a region be duplicated more than once, the ARPs will be distributed the same way as with tandem duplication (except for higher read count and more ARPs).

1.3 Conclusion

Theorem 1 Putting all the lemnas together we can conclude that given a set of structural variants, there are two windows of size `2 * (insert_size + read_size)` each, centered on the two breakpoints in the reference genome that contains most of the ARPs that will support it.

Theorem 2 Given a type of variant and a set of supporting ARPs, we can construct two windows of size `2 * (insert_size + read_size)` that will contain the breakpoints in the reference genome.

Note For tandem CNVs (and deletions) windows of size `insert_size + read_size` would have sufficed to contain all the ARPs, however they would not have contained reads outside of the duplicated area (inside the deleted area), which would make us lose the RC information. Taking larger windows also allows us to have a single window size for all our variants, and thus using the same model for all the variants.

Note All the window sizes given here assumed that the reads and inserts were of the same size, when we know that they are variable, adding a few standards deviations will allow us to catch the read pairs with higher insert size or read size.