vladaburian (vladaburian@gmail.com)

# Argumentation

I haven't utilized any advanced machine learning approaches/algorithms because the problem seemed to me as being quite straightforward. It's just an evaluation of measurements which should be directly and strongly related to the property being measured. I would not expect any non-obvious hidden dependencies/errors to be discovered and learnt using machine learning.

That being said, I have seen many cases in the training dataset that I have no explanation for. Quite often measurements were very different from ground-truth values and seemed totally unrelated. Maybe it's only just because of some complex hidden interference, maybe there is physical explanation for that or maybe it's a human error. Anyway I would investigate the training set and the measurement process more deeply and try to understand what's actually going on before taking ML approach.

Problem description has suggested that k-means algorithm used in reference solution has high bias. So I have used EM (expectation-maximization) algorithm with GMM (Gaussian mixture model), which should perform better in terms or bias and capability to detect overlapped peaks. In the end I would say some bias is not really the problem. It's more of a problem to detect peaks at all (or to decide they are indistinguishable) and to assign them correct genes (33% vs 67% peak). I should have spend more time especially with the latter.

# Algorithm

The core of my solution is EM algorithm.

## EM

I have reimplemented EM with some tweeks:

- It's constrained to 2 Gaussians/peaks.
- Overall probability/size of peaks is not being estimated by the algorithm but it's constrained to be 33% and 67% because that's our a priori knowledge about the data.
- I have added 3rd class - noise. Every sample is soft-assigned to peak A or B or to noise. I assume that there is a priori probability (e.g. 5%) that a sample is a noise. This has effect during probability normalization step - samples which have low a priori probability being from peak A or B (they are far from their means) are assigned to noise with high probability. I.e. it causes EM to put less importance on outliers which would normally affect computation of mean and standard deviation a lot.
- Normally Gaussians would be initialized randomly. But I initialize means to 10th and 90th percentile and standard deviation to cca 20% of range. And I try to put 33%

peak first and then 67% peak first. So I run estimation twice with two initial conditions and then I select the one with lower BIC (Bayesian information criterion).

In practise it's necessary to account for case of a single peak. One could compare single-peak and two-peak solution using BIC, but it performed badly. I have decided to ignore two detected peaks in case they are too close relative to their standard deviations:

$$\left|\mu_0 - \mu_1\right| \;/\; (\sigma_1 + \sigma_2) < threshold$$

In that case I simply take median of all samples and ignore detected Gaussians. (Threshold was hand-tuned)

# Overall description

Measurements are evaluated in parallel - single unit of work is evaluation of all experiments with the same barcode (experiments with the same pair of genes). Process to evaluate single barcode is following.

1. All samples are transformed to square roots. (And final results are squared again.) This has yield small improvement.
2. Detect peaks (GMM) across all samples and all experiments from particular barcode. I.e. detect overall gene's distribution. It's done using the EM algorithm described previously with *'noise'* set to 5% and *'threshold'* to 2 - parameter were hand tuned. Results from this step are later used to correct an assignment of 2 genes to detected peaks.
3. Every experiment (well) is evaluated using exactly the same EM algorithm. The result are values A and B - means of 2 detected peaks. Value A should correspond to gene with 33% of samples and value B to gene with 67% of samples. But sometimes this is wrongly detected by EM itself because there is not so many samples to clearly distinguish both genes.
   So values A and B might be exchanged (corrected) based on overall GMM from previous step 2 - it was computed from all samples so we can be pretty confident that both peaks were correctly assigned both genes.
   So I compute values of PDF (probability density function) of A and B being from 33% or 67% percent peak. And if both values A and B have higher probability of being from exactly the other gene (peak), they are exchanged. Mathematically the condition for exchange is:

   $$(P_{33}(A) < P_{67}(A)) \;\&\; (P_{67}(B) < P_{33}(B))$$

   In case that overall peaks were not distinguished (they overlap too much probably), correction step is skipped.
4. That's all

# Deployment

Solution is fully dockerized - to build docker image, run following from the project directory:

```
$ docker build -t gene .
```

To evaluate measured data, run following:

```
$ docker run --rm -it \
     -v $(pwd)/input:/input -v $(pwd)/output:/output \
     gene --dspath /input/$PLATE --out /output --plate $PLATE
```

Where `$PLATE` is a name of test case.

As all (few) parameters were hand-tuned, there is no training phase. Table with relation between barcodes and gene pairs is in `gene/conf.py`.

Solution is written in python with EM algorithm and input files parser written in c++ and wrapped as python module for speed up.

# QA

- **Country of residence:** Czech Republic
- **Highest academic degree achieved:** Master's degree
- **What was your major:** Cybernetics and Robotics

**Main motivation for competing in this challenge:**
I found it to be an interesting problem to solve. And it's related to real world and physics which I especially like.

**How difficult/enjoyable the problem and actual competition of this MM was relative to standard MMs on Topcoder?**
I'm not regular Topcoder user so I don't really have anything to compare with. But I welcome dockerization and online scoring, because it gives us early feedback and freedom to use whatever language/tools. I would also welcome more detailed scoring output - like timing and precision and overall score or so. That would clarify some things.

I would not be afraid to make whole build and run logs available. I think it's a must - you can't keep the output in secret and troubleshoot user's submissions manually. Concerning testing data, I don't see their possible leakage via logs as a problem. First - they are quite useless without ground-truth counterpart. Second - why not to make them public (including ground-truth) anyway? I don't see any harm in doing so.

I would personally be interested to see provisional testing datasets. I was tuning my solution and was observing improvement in precision of cca 10% locally with the same time of execution, but have not seen any distinguishable change in online score. I have tried totally different approach which also performed better locally and again I got roughly the same score online. I wonder what's the cause. So more detailed report and public provision dataset would be helpful.

And last: Prizes are quite high, but amount of time needed to solve such problems is also high and especially risk of earning/winning literally nothing is considerable. It would be nice if solutions (or reports at least) of e.g. N first competitors were shared among them. I would love to learn what are other approaches to the problem and how they perform. It would also be a nice compensation of undertaken risk. And it could potentially led to better overall results in other challenges.