SCHOOL OF DATA ANALYSIS

# Clustering with gradient descent

D. Derkach[3][4] N. Kazeev[3] R. Neychev[2][3] A. Panin[3] F. Ratnikov[3][4] I. Trofimov[1] M. Vesterinen[5] A. Ustyuzhanin[2][3][4]

[1] Yandex Data Factory [2] MIPT [3] Yandex School of Data Analysis [4] HSE [5] Ruprecht-Karls-Universitaet Physikalisches Institut

# Clustering

> Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). (Wikipedia)

> There is no objectively "correct" clustering algorithm, but as it was noted, "clustering is in the eye of the beholder." [1]

[1] Estivill-Castro, Vladimir (20 June 2002). "Why so many clustering algorithms — A Position Paper". ACM SIGKDD Explorations Newsletter.

# Clustering

> Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). (Wikipedia)

> There is no objectively "correct" clustering algorithm, but as it was noted, "clustering is in the eye of the beholder." [1]

> But what if you know what makes a good cluster for your problem?

[1] Estivill-Castro, Vladimir (20 June 2002). "Why so many clustering algorithms — A Position Paper". ACM SIGKDD Explorations Newsletter.

# Clustering → continuous optimization

# Clustering → continuous optimization

- Write down your loss as a function of the clusters assignments

# Clustering $\rightarrow$ continuous optimization

- Write down your loss as a function of the clusters assignments
- Assume each object has a probability to be in each cluster, write down the expectation of the loss as a function of the probabilities

# Clustering → continuous optimization

- Write down your loss as a function of the clusters assignments
- Assume each object has a probability to be in each cluster, write down the expectation of the loss as a function of the probabilities
- If calculating the precise expectation is difficult, use an approximation. Make sure it's continuous and evaluates correctly for integer probabilities.

# Solving the boundary conditions

$L_{ls}$ are probabilities for the $l$-th object to be in the $s$-th cluster. And they have boundary conditions:

- $L_{ls} \in [0, 1]$
- An object must be on average assigned to a cluster, so $\sum_s L_{ls} = 1$

# Solving the boundary conditions

$L_{ls}$ are probabilities for the $l$-th object to be in the $s$-th cluster. And they have boundary conditions:

- $L_{ls} \in [0, 1]$
- An object must be on average assigned to a cluster, so $\sum_s L_{ls} = 1$

Let's redefine $L_{ls}$ through a parameter:

$$L_{ls} = \frac{e^{A_{ls}}}{\sum_s e^{A_{ls}}}. \tag{1}$$

This way $A_{ls}$ can have any value. This trick is from deep learning and is called softmax.

# Optimizing [Theano ad]

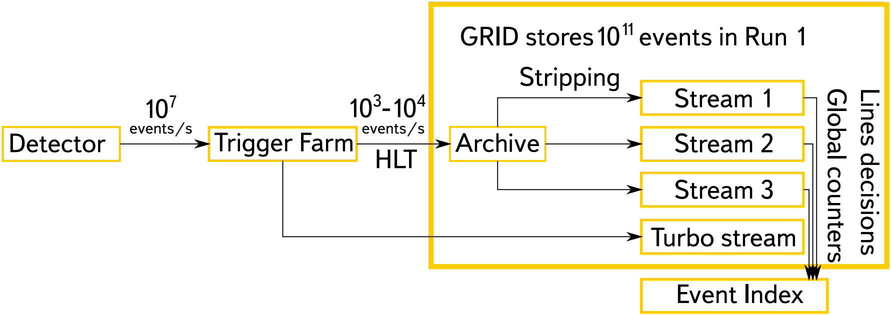
`http://deeplearning.net/software/theano/`

> Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

- Easy-to-use Python interface
- Fast evaluation: the expressions are put into C code and compiled
- Transparent GPU support. Even faster evaluation.
- Symbolic differentiation — Theano does your derivatives

# LHCb streams

# LHCb processing pipeline



Event Index: https://twiki.cern.ch/twiki/bin/view/LHCb/EventIndex

# Streams

- Event is selected if it passes at least one HLT line
- Lines are grouped in streams
- Part of the event (up to the whole, depends of the lines) is copied to all the streams its lines belong

# Streams are a trade-off

- Due to data format allowing only sequential access, tasks can only be launched on the entire streams and have to read unneeded events - the smaller the streams, the faster they would run.
- Events are duplicated - more streams means using more space.
- There is a limit on the number of streams from the computing side.
- The boundary cases are: all in one stream (best space) and each line in its own stream (best time)

# Quality metric

Assuming each stream would be read as many times as lines are in it and each reading would take $N_{\text{events in stream}}$. The total reading time would be proportional to:

$$T = \sum_{\text{stream}} N_{\text{events in stream}} \cdot N_{\text{lines in stream}} \tag{2}$$

# Continuous loss

Let $E_{el} \in \{0, 1\}$ be the indicator whether event $e$ was selected by line $l$.

$$E\left(N_{\text{lines in stream}}\right) = \sum_{l} L_{ls} \tag{3}$$

$$E\left(N_{\text{events in stream } s}\right) = \sum_{e} \left(1 - \prod_{l} \left(1 - E_{el} L_{ls}\right)\right) \tag{4}$$

# The approximated T

$$\tilde{T} = \sum_s E\left(N_{\text{lines in stream } s}\right) \cdot E\left(N_{\text{events in stream } s}\right)$$

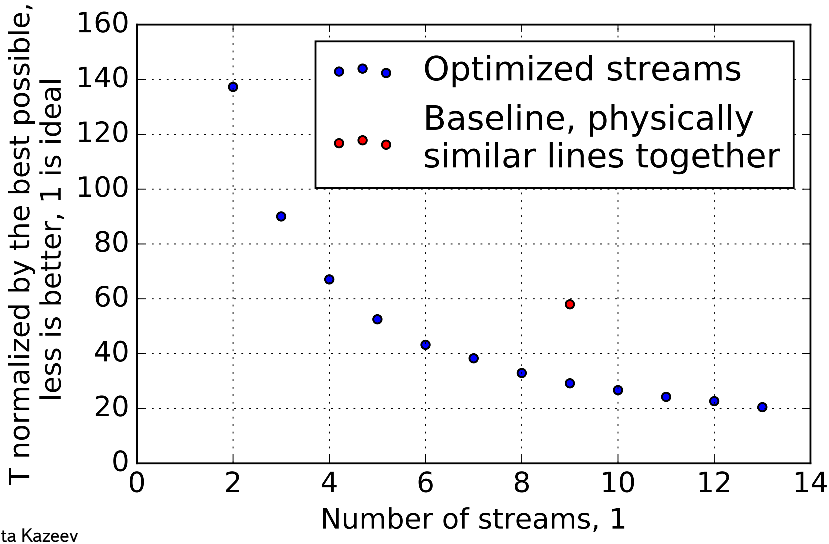
$$= \sum_s \left[ \sum_l L_{ls} \cdot \sum_e \left( 1 - \prod_l \left(1 - E_{el} L_{ls}\right) \right) \right]$$

In general $\tilde{T} \neq E(T)$. However if all the assignments are definite $L_{ls} \in \{0, 1\}, \tilde{T} = T$.

# Stochastic gradient optimization

- In the loss there is a sum over all the events $\sum_e$.
- Evaluating over all of them is too CPU consuming.
- Solution: don't evaluate the sum over all the events, use stochastic gradient optimization.
- Split events into batches, calculate the gradient on it, make a descent step.
- Take the next batch, repeat until convergence.

# Optimization results

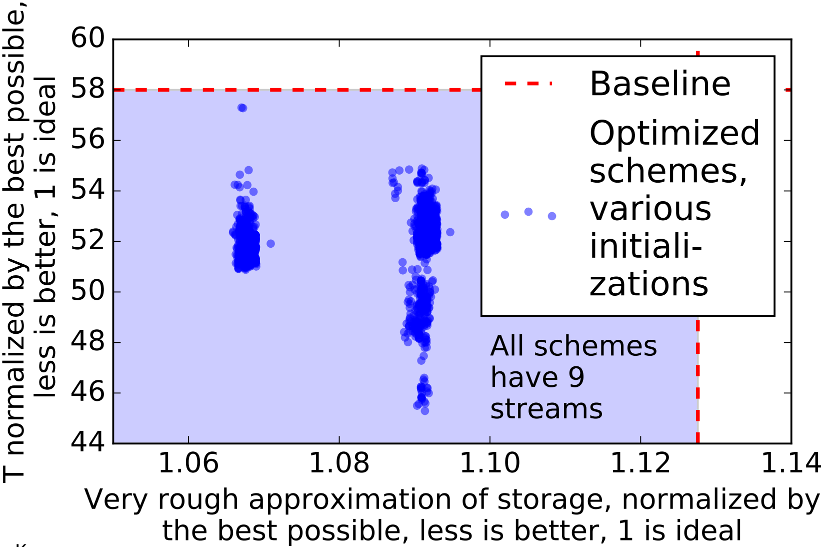


# More constraints

Some lines should go together as they are often used together. It was accommodated by calling such groups modules and adjusting $E$ and $L$: $\tilde{E}_{em}$ as indicator whether in module $m$ any line has selected event $e$ and $\tilde{L}_{ms}$ as the probability of module $m$ to be in stream $s$ and $M_m$ as the number of lines in a module.

$$\tilde{\tilde{T}} = \sum_s \left[ \sum_m \left( M_m \tilde{L}_{ms} \right) \cdot \sum_e \left( 1 - \prod_m \left( 1 - \tilde{E}_{el} \tilde{L}_{ms} \right) \right) \right] \tag{5}$$

# Constrained optimization result



# Summary

- The method allows to solve the clustering problem for a wide range of loss functions.
- The method was tested on a sample from 2016 LHCb HLT data and potentially allows to decrease the IO time of the analysis jobs by 20% compared to the baseline while maintaining analysis and computing constraints.