# Detecting radiological threats in urban areas

by Pavel Kuzmin

pavloon@bk.ru

## Problem summary

The United States government routinely performs radiological search deployments to search for the presence of illicit nuclear materials (like highly enriched uranium and weapons grade plutonium) in a given area. The deployments can be intelligence driven, in support of law enforcement, and planned events such as the Super Bowl, presidential inaugurations or political conventions. In a typical deployment, radiation detection systems carried by human operators or mounted on vehicles move in a clearing pattern through the search area. Search teams rely on radiation detection algorithms, running on these systems in real time, to alert them to the presence of an illicit threat source. The detection and identification of sources is complicated by large variation of natural radiation background throughout a given search area and the potential presence of localized non-threat sources such as patients undergoing treatment with medical isotopes. As a result, detection algorithms must be carefully balanced between missing real sources and reporting too many false alarms. The purpose of this competition is to investigate new methodologies in detecting and identifying nuclear materials in a simplified search mission.

## Overview

For this competition, Monte Carlo particle transport models (see scale.ornl.gov) have been to create thousands of data files resembling typical radiological search data collected on urban streets in a mid-sized U.S. city. Each data file – a run – simulates the pulse trains of gamma-ray detection events (amount of energy deposited in the detector at a given time) received by a standard thallium-doped sodium iodide – NaI(Tl) – detector, specifically a 2”×4”×16” NaI(Tl) detector, driving along several city street blocks in a search vehicle. Gamma-rays produced from radioactive decay have a characteristic energy which can be used to identify the source isotope (https://en.wikipedia.org/wiki/Gamma_ray). All of the runs contain data on radiation-detector interaction events from the natural background sources (roadway, sidewalks, and buildings), and some of the runs also contain data arising from non-natural extraneous radiation sources.

The event data created for this competition is derived from a simplified radiation transport model – a street without parked cars, pedestrians, or other “clutter”, a constant search vehicle speed with no stoplights, and no vehicles surrounding the search vehicle. In fact, the search vehicle itself is not even in the model – the detector is moving down the street by itself, 1 meter off the ground. The energy resolution is 7.5% at 661 kilo-electronvolts (keV).This simple model provides a starting point for comparing detection algorithms at their most basic level.

## Objective

The runs are separated into two sets: a training set for which a file with the correct answers (source type, time at which the detector was closest to the source) is also provided, and a test set for which you, the competitor, will populate and submit an answer file for online scoring. For each run in the test set, you'll use your detection algorithm on the event data to determine

1. whether there is a non-natural extraneous source along the path (the detection component), and if so,
2. what type of source it is (identification) and
3. at what point the detector was closest to it during the run (location — which in this competition will be reported in seconds from the start of the run).

## Models description

### Feature extraction

Each Run file was aggregated to 1 Hz rate. Two kinds of features were produced during aggregation:

- Aggregation to energy channels (channel features). The energy was binned to the energy channels. The bandwidth is set to be 50 kEv (661 kEv*0.075 = 49.575 kEv). Only first 64 channels were taken (0-3.2 MEv). Higher energies are rare in data. Then calculated the number of events in each energy channel. So each run can be presented as a 64 channel signal with 1s sampling period.
- A number of statistics (statistics features) for time between events and event energy inside aggregation window (1s): mean,max,min,median,std,skew, count of events in a window. difference between energy mean and median,something like signal-noise ratio: the ratio of energy mean and energy std, and maximal signal-noise ratio.

### Validation scheme

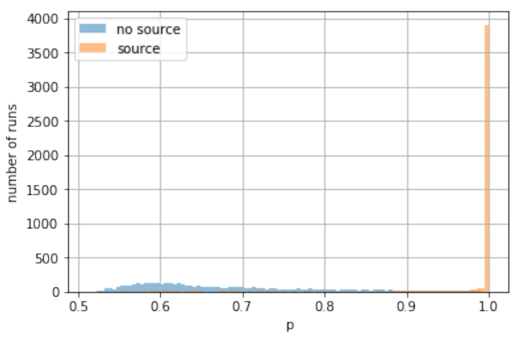
5 fold cross validation (CV) with stratification by source type was used during training and validation. The models trained in each split of CV are saved. Test predictions are done by averaging the results of such one type models. The random seed for CV was fixed and used the same each time.

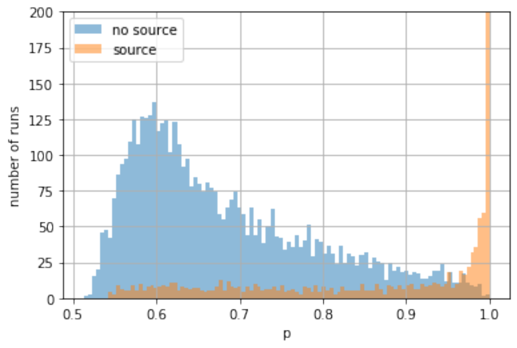
### First level models

Six first level models are trained in the following way.

One can assume that each second of each run is an absolutely independent observation (forget that we have a number of time-series). Then each observation is labeled as source type of run id it took from (1-6) or for binary classification 1/0 (source/no source) .

After that it is possible to train a classifier on a huge amount of points. While training auc metric was used as a score for binary classifiers and accuracy for multiclass. The score of these models is very low (~0.55 auc for example). But if we check the maximum of the classifier outputs  p for each run we'll find out that a huge part of runs with sources has one or more points where classifier is very confident that there is a source, while maximum confidence for no source is lower:





Even simple one model baseline can be done here. Choosing an appropriate threshold for p (about 0.97) can give about 0.93 accuracy for detection task on CV for train data. We can also mark the point with highest p as the closest to the source point.

So 4 models are trained to predict whether single point is from run with source or without:

- gradiend boosting model in light gbm package (lgb) on  channel features
- lgb on  statistics features
- multilayer perceptron (mlp) in keras package on  channel features
- mlp on  statistics features

Mlp models were trained with decreasing learning rate (like in cosine annealing) for 10 epochs  for each CV split.

Also two models for multi class classification were trained.

- lgb on  channel features
- mlp on  channel features

The runs with source only were used for training multiclass classifiers. Again simple baseline possible: just classifying the run to the class, which has greatest score p of the model during the run. The accuracy of such baseline is about 0.87 for lgb model.

## Second level models

Three different models were trained for the tasks. All of them are trained on channel features combined with the 1$^{st}$ level predictions. All the models were trained in the 5 fold CV.

### Detection

All the signals were combined with 1$^{st}$ level predictions and padded with zeros to the size of 1024All the data was scaled with min-max scaling. Then continuous wavelet transform was used for each channel with “mexican hat” mother wavelet to get 30 channels from each channel. To decrease the amount of data the maximum from these each 30 channels in each moment was used to recreate again 1 channel. (transformation: number of channels → 30*number of channels (each channel transformed) → number of channels (maximum in every 30 channels is taken))

The detection model is 1D convolution network with attention layer and one fully connected layer after attention.

The model was trained with binary focal loss. Focal loss is used to pay more attention to difficult for classification cases. While training snapshot ensembling was also used. Total number of epochs: 50, periods: 5. Cosine annealing for changing learning rate was used. So after CV 25 (5 splits*5 periods) models are presented. The predictions of models for a run are averaged and I used threshold 0.5 to classify if run with/without source. Even though we know that competition metric penalizes more for FP finding threshold for detection can be not a good idea. We have great imbalance of hard cases in train and test data. Leaderboard probing is possible but also can lead to the overfeat. So I just tried to build as accurate classifier as possible on available data.

## Finding closest point to the source

The training was done only on data with sources. Channel features were combined with 1st level predictions and min-max scaled. While training cropping (with probability 0.5) random size fragment with closest point to the source was used. Then each channel was standardized inside such run and transformed to the length of 1024 with interpolation. Such multichannel signals (cropped or not) then labeled with a 1-0 mask: a random region near the closest point to the target is labeled with 1.

The model has 1d U-net structure with binary focal loss. While training decreasing the learning rate on plateau is used and best model is saved. Each epoch the score was calculated in the following way.

Getting predicted closest time:

- get prediction mask for each sample a 1024 array for one signal.
- Assign 0 to all points with p (model score) less than 0.8*maximal p
- Calculate closest point: $t_{pred} = \frac{\sum t_i p_i}{\sum p_i}$ $t$ – means time 0 to 1024
- Recalculate to the seconds (remember we scaled our signal to 1024 points).
- Clip the time by 30 seconds.

Then score is calculated: -2 points for high difference to the ground truth ( I used max difference for this 4 seconds), +0...1 points by cosine law (1 in the center, zero at +- 4) for predicted time close to the ground truth. I have also divided this score by the number of points to get normalized score.

5 models for 5 CV splits were trained. During inference the predictions of these models (masks with p) are averaged and closest time is predicted according to the procedure like in training, but before clipping to 30 seconds all points that have predicted source time less then 26 are marked as no source points.

## Classification

For classification channel features were combined with 1st level predictions on channel features and aggregated by each run with different aggregation functions: mean, median, max, min, std,skewness. The training is done only on data with source presented. Trained model – gradient boosting over decision trees from light gbm package.

For inference the predictions of 5 models are averaged and argmax is used (usual multi class classification with softmax).

So final pipline: preprocessing→ generating 1st level predictions → detecting source→ calculating closest time → for non zero detected source classify it.

It’s worth noting that simple baseline with two lgb 1st level models can give the score of about 80-83 on public leader board without any heavy training