Highlights and alerts

scribe here an approach which may provide a framework for relevant and informative ICU alerts. ICU alerts are both not sufficiently relevant (false alerts) and not sufficiently informative (trivial alerts). ICU staff tends to ignore alerts because the alert often either does not require an action, or the action will be taken anyway, with or without the alert.

We introduce distinction between *highlights* and *alerts*. A *highlight* is a short timespan with observations which may require attention. The computer finds highlights but does not force attention to highlights per se. An *alert* is the *decision* that a highlight does require human attention. Based on this decision, the computer may light a bulb, play a sound, or draw attention in another way.

Many — most — highlights may be irrelevant. It is important though that:

- Highlights are more often relevant than a random timespan.
- It is easier to decide on a highlight's relevance than on a random timespan.
- Highlights are hard to spot. One needs a tool to find and investigate them.

Most alerts, however, must be *relevant*, *actionable*, and *non-trivial*.

- Relevant — related to a change in condition which requires attention.
- Actionable — the change in condition must be easily deduced from the information supplied with the alert.
- Non-trivial — the action following the alert would not be taken if the alert were not issued.

A model of early deterioration signs

In what follows, we assume the model of deterioration signs shown in Figure 1. A single concept is shown for simplicity and may be extended to multiple concepts.

There are three areas of interest (labeled I, II and III in the figure) in deterioration process which require attention.

I corresponds to a short-term perturbation in the patient's state, after which the patient transitions to another stable state, which may even be indistinguishable from the state before **I**. When observing the patient either before or after **I**, but excluding **I**, no signs of deterioration can be observed. However, the instability which happens in **I**, along with the state change, is the first early sign of ongoing deterioration.

II is where the patient's state begins to drift towards the deteriorated state. The patient's condition is still within acceptable limits, however *the trend* points in the direction of deterioration. By observing the patient in **II** along a contiguous

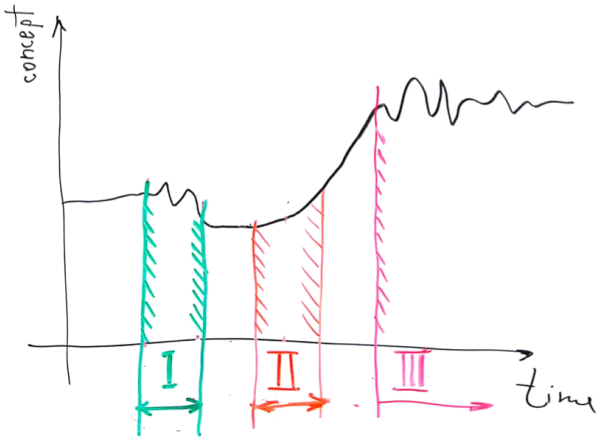

Figure 1: Deterioration model

relatively short timespan a doctor or an algorithm may conclude that there are signs of deterioration based on patient dynamics.

III is the deteriorated state. The patient's condition is beyond the acceptable limits, and momentary observation of the patient sufficient to conclude that the patient has deteriorated.

We accept that **II** can happen shortly or even immediately after **I**, but assume that **I** (that is a sharp short-term change) happens in most cases, and propose to alert on the risk of deterioration based on detection of **I**. Not any short-term instability is an early sign of deterioration, and it is the role of a filtering algorithm is to find those instabilities (highlights) which are likely to be stage **I** of a deterioration, and thus justify an alert.

Learning: two-stage versus end-to-end

Modern machine learning advocates end-to-end learning — that is, we get what we observe and optimize for the desired answer. Applied to our problem of alerting on early deterioration signs, this would mean learning to decide where **I** is based on raw input (such as combination of vital signs, lab tests, and any other information about the patient).

We advocate here an approach which seemingly deviates from the end-to-end principle: we propose to find *change points* (highlights), and then to *classify* the change points as either deserving an alert or safe. By breaking the learning process into stages, we risk getting inferior results because what we learn as highlights may not necessarily be the best alert candidates.

However, we argue that the proposed highlight/alert split is well justified. This

is because the ultimate problem we want to solve is a classification problem — whether the probability of a given timespan to contain an early deterioration sign is high enough to issue an alert. To solve this problem using the tools of supervised machine learning, we need to obtain a not too skewed dataset with binary or probabilistic labels of every entry (a short timespan, e.g. 15 minutes) as either worthy or unworthy of an alert.

Note that if we consider any timespan for labeling, we get a highly skewed (unbalanced) dataset, because most timespans should not cause an alert. In addition, a labeling function (implemented formally as a computer algorithm or informally as the expert's intuition) should address both whether there are possible signs of deterioration in the current timespan and that there is a related deterioration in the future. While the latter question (future deterioration) can be answered based on the available records, the former question needs to take into account the history of observations and involves computations which should apparently be based on the model of the time process describing the patient.

Hence, by splitting the labeling/detection process into highlight detection and filtering/augmentation of alerts, we essentially both

- Automate a part of labeling function which examines a time span for possible signs of deterioration.
- During classification, prune (reject) samples which will certainly get the negative predicted label.

So, the proposed highlight/alert split is actually an end-to-end supervised learning approach with automatically computed labeling function (which is itself learned using unsupervised learning techniques, more on this later).

Locating highlights

To remind, we propose a two stage approach. The first stage is to use temporal data (monitor) to detect highlights. The second stage is:

- During learning, label each highlight and train a model to distinguish between relevant and irrelevant highlights.
- During detection, classify each highlight using the pre-trained model, augment highlight which cause alerts with additional information and raise the alerts.

Hence the first stage is to detect highlights. We use a machine learning model (time series predictor based on RNN architecture, described in another note) to predict the mean and variance of each temporal concept out of a small set of frequently taken measurements (the current setup uses 'HR', 'InvBPSys', 'InvBPDias', 'SpO2', and 'RRtotal'). Using this prediction model, we can tell at every point time what we expect the measurements to be based on the past

history. If the actual measurements deviate significantly from the anticipated one, we observe a *change point*.

The simplest approach would be to compute the probability of actual measurements relative to predictions made from the past (such as 15 minutes ago). Then, if the total probability of a 15 minute timespan is below a certain threshold, mark the timespan as a highlight. This is indeed the way we marked highlights initially. However, this approach turned out to be quite noisy and required heuristic smoothing and cutting off 'clear outliers' for reliable detection.

A more robust approach is based on the observation that short-term prediction follow the trend change better than long-term prediction. When the time series is stable, predictions from 1 minute and 15 minute in the past should be close. However, when there is a change, 1 minute predictions will follow the change better than 15 minute predictions, which will be more conservative. We use this observation to detect highlights — we compute the discrepancy between 1 minute and 15 minute predictions and use the maximum discrepancy over the timespa (KL divergence under assumption of component-wise normality is computed, in closed form, to estimate the discrepancy). This proved to be a sufficiently stable and robust detection criterion.

Highlight features

After highlights are detected, they must be passed to a classifier. To be classified efficiently they must be augmented with information sufficient to characterize them. Additional information not present in the time series can be added, but essential temporal features can be characterized by the time series model itself. The model maintains an internal multidimensional state vector, updated at every time step. The vector does not have a simple interpretation, however it fully summarizes the past history at every time step for the model. Hence, the basic feature vector of a highlight is concatenation of the internal state vectors at the beginning and at the end of the timespan.

Deciding when a highlight is worth an alert

Given a highlight represented as the feature vector, we need to solve a classification problem of whether to issue an alert for the highlight. To train the classifier, we need to obtain a label. Since we already concluded that the timespan contains possible signs of deterioration, we only need the label to specify whether there is a relevant deterioration after the highlight.

Such labeling can be accomplished by expert-provided functions (rules) which are applied automatically during training (dataset preparation) time. A simple version of the labeling function is just a check whether any deterioration is

recorded after the timespan — in the absence of a better one, this function is used in the empirical evaluation and gives reasonable results.

Experimental results

To evaluate the approach, we trained an unsupervised temporal model for highlight detection on 5 monitor features using the Ichilov dataset. We computed highlights on all stays 3 to 14 days in length which gave us 78173 15 minute highlights.

Further on, we used the labeling function which returns 1 if there is a deterioration tag after the highlight (at any time) and 0 otherwise. This gave us 31997 (41%) positively labeled highlights.

Highlight classifer

Based on this labeling, we trained a classifier. Logistic regression gives reasonable results, however the results here are for a three layer fully connected network. The confusion matrix and the F1 score on the test set are as follows:

Table 1: Confusion matrix, F1 score: 0.63

	0	1
0	0.38	0.21
1	0.12	0.28

In words, there are 3 false alerts for each 4 true alerts, and 30% true alerts are missed, given the labeling. By altering the class weights during training, the ratios can be adjusted.

We verified that there is no obvious label leakage — the classifier does not learn to distinguish between stays with and without tags (actually, the classifier accuracy is higher when trained and tested only on stays with tags) and there is no implicit time counter — splitting stays at random points and running the model on partial stays does not affect the accuracy, beyond the computation noise.

Example stays

Two stays are shown below as examples of highlight detection and alert filtering.

For each stay, the upper plot shows the KL divergence between 15 min and 1 min predictions along the bottom edge, passed through the *sigmoid* function to bound it between 0 and 1. The threshold for detecting highlights was set to 0.5,

rather arbitrarily. Red peaks correspond to raised alerts, grey peaks correspond to highlights which did not result in an alert.

The lower plot shows cumulative tag, highlight, and alert counts for each stay — every time there is a tag, an alert or a highlight, correspondingly, the value is increased by one. A perfect detection algorithm would result in the cumulative alert plot which closely resembles the cumulative tag plot. Normalized time series of monitor concepts are shown in the background.

Stay 'Ichilov_MICU_23879' has one deterioration tag. Alerts are raised before and in close proximity of the tag. Highlights are also detected along the whole stay duration till the very end, however the model properly detects and rejects most of them as irrelevant.

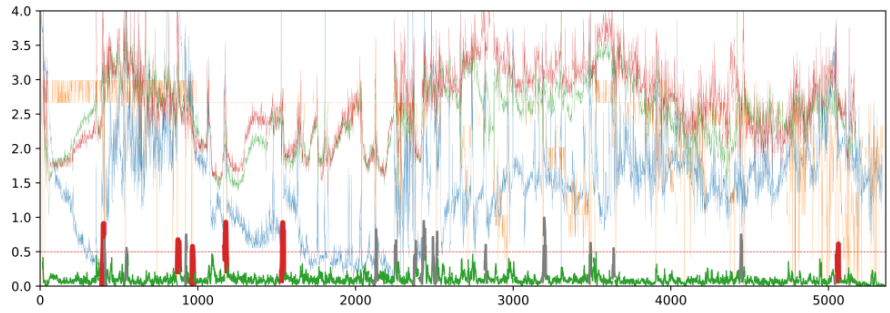

Figure 2: Ichilov_MICU_23879: Monitor and alerts

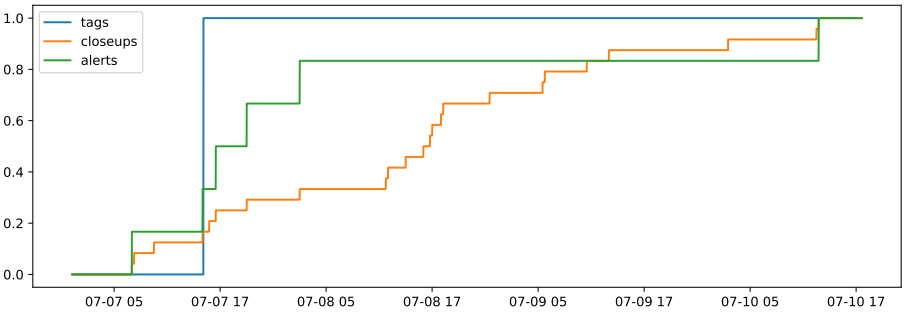

Figure 3: Ichilov_MICU_23879: Highlights, alerts, tags

Stay 'Ichilov_MICU_26088' does not have any deterioration tags. Despite the large number of detected highlights — the state is rather noisy — only two of them (one at the very beginning) are propagated to alerts, decreasing the false alert ratio roughly tenfold.

These are just two extreme examples. Stays with multiple tags are handled in a reasonable way too, however success of this approach depends on development of an informative labeling function.

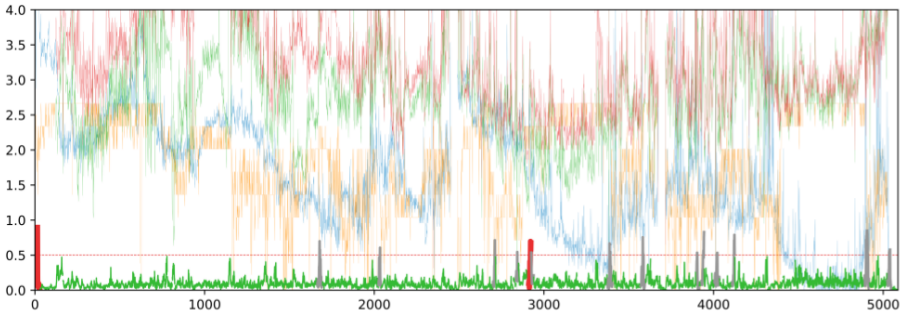

Figure 4: Ichilov_MICU_26088: Monitor and alerts

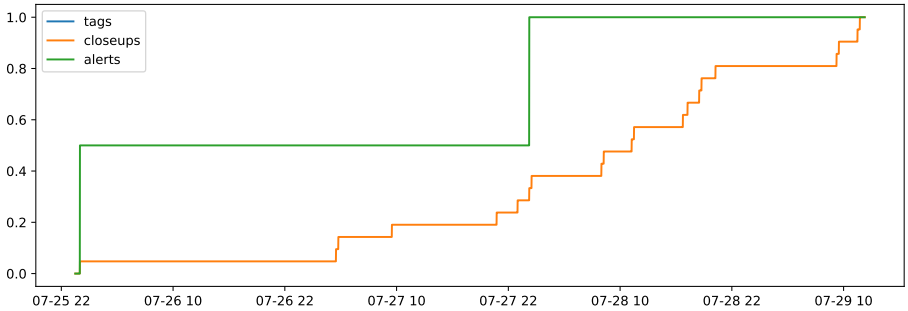

Figure 5: Ichilov_MICU_26088: Highlights, alerts, tags