OLAC - Online Learning at Cost

a. **Identify the problem & business case**

Data that changes over time can be an issue in regards to classification tasks in machine learning. Especially if new characteristics emerge within the same class. An example of this is machine learning applied to fraud detection in financial institutions. New kinds of fraud appear over time, as new ways to 'cheat the system' are invented, especially if current ways are being successfully detected or stopped.
A problem for ML is that the flexibility of most algorithms is not strong enough to keep up with these new types of the target-class appearing over time. Retraining is the a common way of dealing with these changes. However, successfully retraining your model to detect new types of your target-class highly depends on these new types of being labelled. Retrieving new labels is an expensive exercise as cases need to be investigated by the human in the loop. In addition, there is a risk of of introducing bias in your model by only investigating the most likely cases produced by the model.

In this project we are investigating retraining strategies for models in production models in a cost-effective way, applied to toy-datasets, with a basis in fraud detection as a use case. We will do this by comparing 'traditional' deep learning vs Online learning models. The goal of the project is to research optimal settings for labelling new data and providing feedback to a trained model, provided that we are trying balance the cost of obtaining new labels, with the cost of model decay over time.

b. **Describe the data-driven artifact**
We will produce a Proof of Concept demonstrating our progress towards an automated solution for preventing model decay on dynamic data. In particular we expect to produce a series of demonstrative Jupyter Notebooks with explorations of the results, as well as code snippets that could be compiled into a software package (or incorporated into existing packages) at a later date, e.g. as part of a follow-up project.

c. **Data**
    i. *Process that produced the data*
    The data we want to use will be a simulated set that represents the use cases described in the problem. We cope with two types of data we might use in this project. Our main type of data will be simulated data. This data is simulated according to the following determined characteristics:
        1. Has to be dynamic in nature (new data comes in over time)
        2. Data has only x and y coordinates as characteristics

3. Data with new ‘characteristics’ will appear over time. In the toy problem, which we will mainly use, this means that over time new clusters will appear in the xy space.
4. Noise will be simulated in the way that the clusters have a certain width. The density of the cluster will generally be given by a gaussian distribution.
5. (optional) we will sample from a existing dataset as input for the algorithm. This will only be done if there is time left.

So in other words the data will either be completely simulated, or based on existing data where portions of this data will be introduced over time. Because of the fact that it is a simulation the data will not have missing values or significant error that we have to fix or cope with.

ii. *Benefits of using this (type of) data (structured, unstructured, simulated)*
The benefits of using simulated data is that we can control when new features are introduced, and monitor exactly how the algorithms react to these changes. This will allow us to create a good baseline and see how different retraining or online learning strategies affect the performance of our model.

iii. Describe the datasets
For description of the datasets we will potentially use for de simulated data described in point c.i. See our github: https://github.com/RUrlus/olac

iv. Data Quality
Data quality issues will be addressed in an ad-hoc manner as they arise during the initial data explorations. We are using most of the time our own simulated data or simulated data from kaggle (last is the optional part).

v. Selections on data
To simulate data of a dynamic nature while trying to retain a realistic dataset, we may make use of data masking to distort the statistics of our train and test set. Specifically, one option is to cluster a static dataset and then return the rows in a streaming fashion, sampling from the different clusters with dynamic probability that varies in time. This will ensure that future data does not necessarily resemble past data, while still utilizing available real(istic) static data.

vi. For the data model see the input of a single line from the data generators below. This is the data we use for our toy problem. The data is streaming so will come in one by one over time for 100s to 1000 records per second. The data is fed into the algorithm directly. The generated data is not by default not saved but generated for each simulation again because it has to be streaming.

x	y	label

2.1819	3.3122	0 or 1

Data dictionary:

Field name	Data type	Allow empty records	Description
x	float	No	x coordinate in xy space
y	float	No	y coordinate in xy space
label	Int {0,1}	No	Label for the classifier can be either 1 or 0

d. Project steps

i. Schematic overview

1. Define the use case and get an overview of what we want to present in the end by asking the following things to ourselves:
    I. What does the real data look like?
    II. What are the real problems with dynamic datasets and online learning?
    III. What would a preliminary demo design look like?
    IV. Etc.
2. Research what the general shape of the data should be and how it is mapped to the real life problems.
3. Model the case where the problems (degradation of the model) actually occurs
4. Create the toy data we need for the problem, the data will be created in the order that it starts simplistic and made more realistic on the fly.
5. In parallel part of the team shall start with the setup for:
    I. The online learning algorithms
    II. Retraining the algorithms
    III. Learning at cost case

    We will make use of deep learning algorithms to classify the data points. Thus for example in case of fraud the labels are is a certain transaction fraudulent or not. Because of the fact that the dataset is dynamic we will either need to retrain or perform online learning. Retraining or performing online learning have each their own costs. Also the fact that data has to be newly labeled will also have a price.
6. Merge the two parts of the algorithm so that we can compare the whole result and find if there are improvements.

7. Create the final pitch of our results including live demo

ii. Planned interactions with the sponsor
    1. Erik Postma: Weekly update calls and periodic face-to-face meetings for more in-depth reviews and discussions.
    2. Max Baak: Sporadic input and validation.

iii. Potential issues
    1. **Scope creep** - The problem space is big enough that it could fill multiple PhDs so we should be careful to prevent digging in to much in a particular step given the allotted time.
       Preventative measure:
       By strongly scoping the problem space beforehand and solidifying this in a set of assumptions we can prevent creep.
    2. **Circular dependencies** - The way the subsections of the problem are intertwined could cause circular dependencies in the work, where the project members are dependent on work of others.
       Preventative measure:
       This will be addressed by creating a minimal viable pipeline between the various elements which will reduces the chance of blockages.
    3. **Highly abstract project** - Although not inherently an issue it would be preferable if we can strongly link it to practical use cases.

This results in the following week by week planning.
The project time we have for this project to implement the practical part is 16 weeks starting at the 22nd of june till the final presentation on the 5th of october:

Week nr	Subject	Expected impediments
0	All preparations (start report and getting feedback on project setup) and brainstorming with Eric Postma	Matching schedules between us and Eric.
1	Setting up structure, git, docs etc.	None expected
2	Development of Toy problem: - Define the key aspects of the real world use cases. - What are the main aspects we want to show in this problem? → how a dynamic dataset is in need of a dynamic model which needs a continuous stream of newly investigated data	None expected, might take up some time

	points. Where the investigation of data points is an expensive exercise. - Define simplest toy problem: → simulated dataset which is dynamic in time Define elements we need for demo/pitch thus: - Visualization - Tryout ML model - First look at online learning algorithms	
3	Development of Toy problem: - Make data generators for the model: - Popping clusters: Clusters that pop up at a random position at a random moment in time - Roving balls: Two round disc like clusters which rotate around a central point. - Satelites: Main large centroid cluster where at the edge randomly cluster appear and disappear. - (Optional later on if time left, resample out of existing dataset) Create Demo and pitch: - Finish visualization and think of general story for the demo	Might take up more time than initially estimated. DEMO Could fail or not working. Maybe we want a back-up in GIFs or similar.
:	Finishing development of Toy Problems and implement them with the demo and visualization. Pitch week! Will get feedback on the pitch	None expected, is partially buffer time for covering up delays which might be caused by unexpected events in the previous weeks.
5	Start investigation deep learning and online learning models. Start at research of how to measure the dynamicality of a dataset	Conceptually will this be a challenge the master this in a short period of time. This might take up some more time
6	Continue investigation and research: start with first static deep learning classifiers. And test Kallman filters	Same as above
7	Define general outline Pipeline schedule from data intake to decision making (learning at cost) to the online/batch learning models	The pipeline will be the spine of the

		total model/data driven artifact. So there is a need to have this done as best as possible.
8	Finish general setup of Pipeline so that models/settings are easily implemented or swapped out Investigate how to train a deep learning algorithm in batch/online mode Create strategie method to identify which point to investigate and which not (Clustering/ Entropy Multi Armed Bandit)	Online learning with deep learning is non standard
9	Vacation period	People will go on vacation so will not be able to work on the project
10	Vacation period	People will go on vacation so will not be able to work on the project
11	Create part of demo Implement/further develop label buying strategie with the clustering and multi armed bandit. Documentation	
12	Implementation of online deep learning algorithm Documentation/reporting	Get to understand conversion between sklearn result classes and keras result class, might take some time to align them
13	Implementation of online deep learning Implementation of learning at cost strategies Documentation/reporting	Get to understand conversion between sklearn result classes and keras result class, might take some time to align them

14	Hand in Final Draft version Final documentation and reporting	None expected
15	At JADS in Den Bosch Final adjustments for the final demo	None expected
16	Final presentation	None expected

e. **Models and algorithms**

i. *Data transformations*

The biggest transformation that we will have to make is to process our data line for line like a stream, although the data is provided as a batch. We will also likely use cluster-based masking to warp the statistics of the dataset through time, so that the training data is displaying substantially different statistics from the stream of new data points.

Additionally, the datasteam (resulting from our *data generators*) will be sent through the *Pipeline* and transforms each point in the sense that the *model* makes a prediction for that point, which is attached to the original point and handed as output.

ii. *Model/algorithm selection*

We will be comparing two classes of algorithms: batch-trained vs online. We expect that online algorithms will be more flexible for statistically dynamic data, but they will likely bring their own challenges too (discovering this is part of the project). Traditional batch-trained models (classification or outlier detection) are more familiar territory, but often make the assumption that the dataset is stationary.

Within these classes we'll be examining the *sklearn* models that have both online and offline training capabilities. Examples we have examined:

1. MLPClassifier: Multi Layer Perceptron Classifier
2. SGDClassifier: Stochastic Gradient Descent
3. PassiveAggressiveClassfier (Only applicable to online learning)

In the end we used the MLPClassifier throughout the comparrissions of the different labelleling strategies.

We will also be using traditional optimization methods to try to balance the cost of acquiring new labelled samples with maintaining good model performance.

iii. *Computation*

Since we have experience with Docker it seems feasible (given time/demand) to dockerize whatever model architecture we come up with. By deploying the model

as a service behind a load balancer we could run multiple instances in parallel, all sharing the same set of learned parameters and updating them as needed. Realistically, this will probably be out of scope for this project.

iv. *Performance measurement*

The goal of our model is to find an optimization between the costs of training, retraining, labeling and the performance of the classification of Fraud or not. The performance of our model will be measured in these two quantities: cost and how well the labeling of the model performs. This will be done by using the metrics of precision and the cost function.

The specific function of the cost is part of the research but we estimate that it will consist of 3 terms: Cost of retraining or additive training, cost of obtaining new labels, cost of bad performance of the model.

v. *Testing & validation*

To test our setup we will need some sort of model experiment scenario consisting of the following phases:

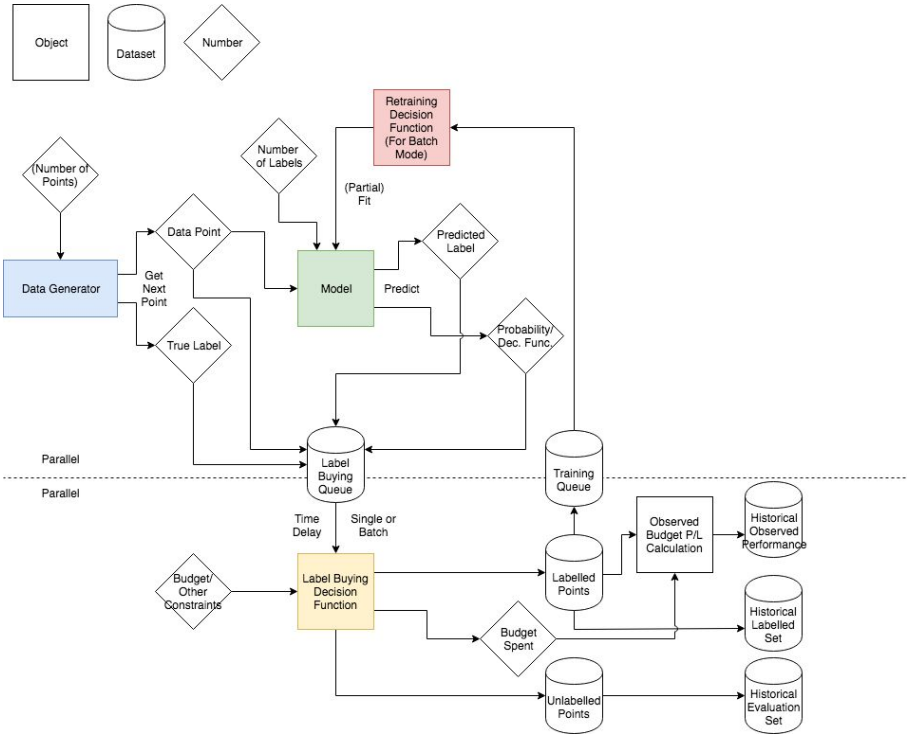
1. (Optional) Initial training phase on “historical” data
2. Performance monitoring phase on data that happens "now"
3. Adjusting the model phase, retraining or online training
4. Optimizing phase we optimize the best settings for the lowest cost and the highest performance of the model

vi. *Feature engineering & selection*

The features we use will be dummy features. We will start to extract a subselection of the features from real fraud cases. We will start with the feature that fraud occurs in different clusters, we will add also in the time dependency. So in plain words, x and y component and a time component.

f. Results:

Our data driven artifact consists mostly of code and notebooks. These notebooks run a pipeline to evaluate incoming data points, classify them, make a decision which points to investigate/buy the labels from, and then retrain on these new points. See the figure below for the schematic overview of the pipeline.



The pipeline consists of 4 major parts. The Data Generator (or real dataset), model, retraining decision function and the label buying decision function. The retraining decision function will be the switch between, retraining on batch mode or online learning (batch mode with batch size of 1 data point). When the data has passed through the pipeline it will be stored/ kept track of. Because of the fact there is a continuous stream of data which comes in, is the performance of the model continuously evaluated.

Learning at Cost:

An optimal labeller needs to solve three major challenges:

1. Detect both gradual and sudden changes in the data in manner that limits false positives whilst remaining sensitive to gradual changes.
2. If a change has been detected the observations prior to the breaking change should be at the very least be given a lower importance to those of the new state. This requires determining/estimating when the change started.
3. Taking into consideration the above and the current state of knowledge regarding the observations choose observations for label buying that maximise the long-term return.

Change detection:

Change detection is an active research field, particularly for streaming data, that has quite a bit of room for further improvement. Most state of the art methodologies have significant limitations in some regard and can be highly domain specific.
To tackle the first two issues the K-ADWIN algorithm [1] was chosen as it has been shown to work well for problems where the data is dynamic and can be separated in some dimension.
The K-ADWIN algorithm consists of three parts:

1. An estimator; adaptive Kalman filter [2]
2. A change-detector; Cumulative Sum (CUSUM) [3, 4]
3. Memory

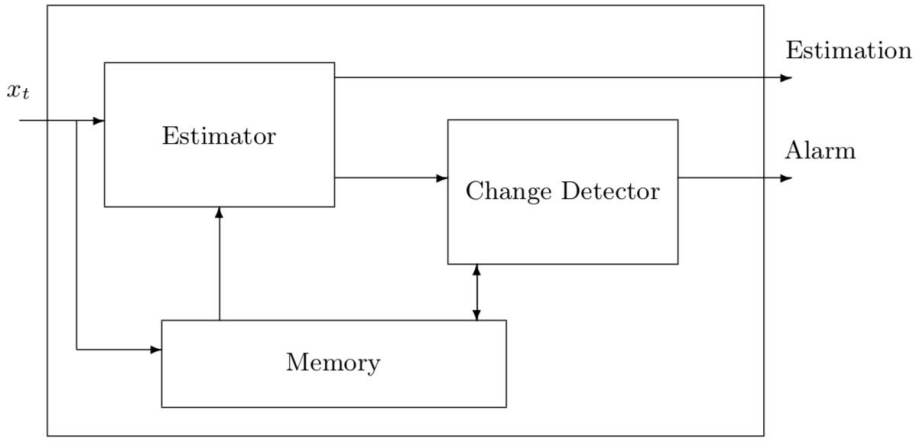


Fig. 1. General Framework

The observations are stored in memory which consists of a set number of blocks of a certain size, if the latest block in full the oldest block is cast away. Next, the observations are run through the estimator. A good estimator should normally distributed residuals with mean zero and a set standard deviation. Knowing the expected distribution of the estimator allows one to detect changes relative quickly.

The K-ADWIN implementation in [1] uses CUSUM for the change detection on the residuals of the Kalman filter. As this combination of algorithms has been shown to work well the same set-up was used. The original CUSUM algorithm enabled change detection and was later extended to include change point determination. When the cumulative sum of the changes passes the user-set threshold an alarm is raised. Upon the sounding of the alarm the memory will be flushed up to change point.

K-ADWIN ensures that only the relevant observations are considered. The downside of this approach can be that sudden but short lasting breaks results in the discard of only temporarily non-representative observations.

Entropy Multi-armed bandits

Having dealt with the dynamicity of the data the next challenge is develop the strategy that optimises the long term gain from buying labels under uncertainty. Under the assumption that the data is separable in at least one dimension we can view this problem as a multi-armed bandit problem. Multi-armed bandits are named as the problem originated from the issue of selecting the most profitable slot machine.

Slot machines can be seen as having a unknown distributions at time zero. In order to determine which slot machine yields the highest return one needs to explore the various slot machines and pull the arms. A number of strategies exist but in general ‘epsilon greedy’ is a good start. For each point in time the epsilon, a value between 0 and 1, is determined. When using the epsilon greedy strategy the epsilon is a function of the number of iterations that have passed scaled by a decay rate.

At each iteration the epsilon value is compared to a sample from a standard uniform. When the epsilon is higher than the sample value we explore the arms. Exploration of the arms can be done is various ways but commonly an arm is chosen randomly.
If the epsilon value is lower than the sample value we exploit the arm that has the highest expected return at the time of choosing.

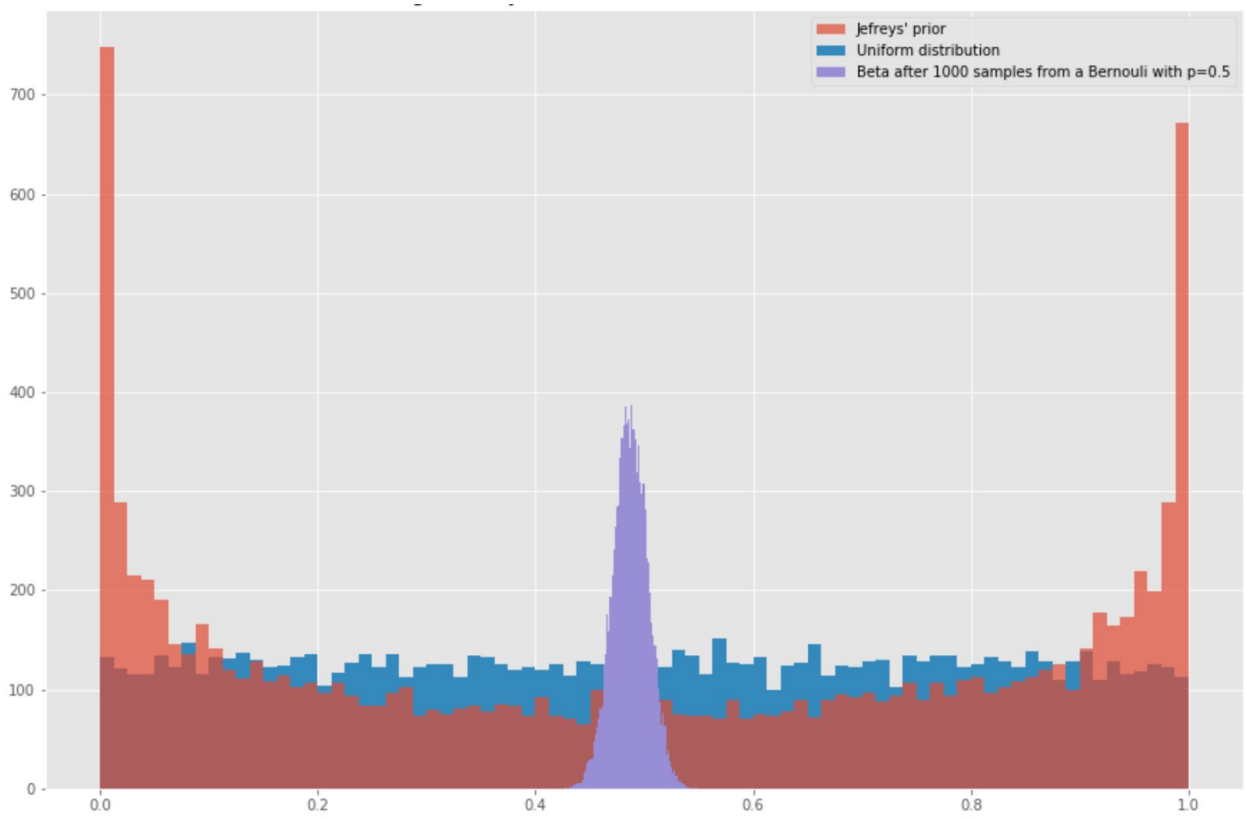
As one might imagine the epsilon will start at one and will asymptotically approach zero as more iterations have passed. As such at the beginning exploration is very likely to happen whilst at the end exploitation is almost certain, this strategy works well when the distributions governing the slot machine is static over time. If this assumption is violated the strategy is not as effective. One could reset epsilon every time the memory is flushed to combat this effect but we can do better by taking into account how much we know of the distributions.

Discrete distributions have a Shannon entropy value, how much information the next observation is likely to give us, that is bounded between 0 and 1. Higher entropy values indicate higher uncertainty about the value of the next sample and as such one should focus on exploration when entropy is high and focus on exploitation when it is low.

The probability of drawing a fraudulent from a set of observations can be modelled as a Bernoulli distribution, a Binomial distribution where n=1. A Bernoulli distribution is discrete and therefore has a defined Shannon entropy.
However, at the beginning we have no observations that allow us to fit a distribution. This can be solved by taking a Bayesian approach and setting a prior for the Bernoulli distribution. A prior can be seen as the belief on has regarding the distribution. In the case of the Bernoulli distribution we want to form a belief regarding its parameter `p`, the probability of drawing fraud from the distribution.

A Beta distribution is a conjugate prior of the Bernoulli distribution, the distributions are from the same family of distributions, and as such can be used as the prior for `p`. In the case of the Beta and the Bernoulli a compound distribution of the two exists, the Beta-Bernoulli distribution where the mean of the Beta distribution is the estimator of the `p` for the Bernoulli.

In turn we can update the Beta distribution with the values we observe, `alpha` is equal to the initial alpha plus the number of frauds we have seen. Beta is set to its initial value plus the number of samples we have drawn that were not frauds.
We use Jeffreys prior, the most uninformative prior with `alpha` = `beta` = 0.5,, as our prior at time zero. A Beta distribution with parameters 0.5, 0.5 looks as following:



The red distribution represent our prior distribution whereas the purple distribution represent the posterior after drawing 1000 sample from a Bernoulli distribution with true probability `p=0.5`. As one can observe the more samples we have seen the stronger our belief about the range in which we expect `p` to fall becomes narrower. In this case the mean of the beta is very close to the actual `p` value of the Bernoulli distribution from which we drew the samples.

With the above in mind it's an intuitive substitute for the epsilon value as the uncertainty we have about `p` will asymptotically approach zero as the iterations approach infinity, much like epsilon in the epsilon greedy strategy.

A slight issue arises at this point, Shannon entropy is not defined for continuous functions and moreover, ideally we would like to consider both the uncertainty we have about the Beta and

Bernoulli distribution. Although a month or two ago a paper was published that defines the entropy of the compound Beta-Binomial distribution the resulting equation is not easily usable to say the least. Although cross-entropy for mixed distributions is a work in progress the resulting equations are not much more usable than the equation for the compound distribution.

With no easy solution available in the given time the choice was made to model the entropy as a function of the the Beta's differential distribution remapped to [0,1] and the Bernoulli's entropy. Please see the notebook Cluster_probability_entropy for more details.
A paper [5] from March this year takes a similar approach in using the information value to govern the optimisation of the MAB problem and shows it to leads to 'a regret that is logarithmic with respect to the number of arm pulls'.

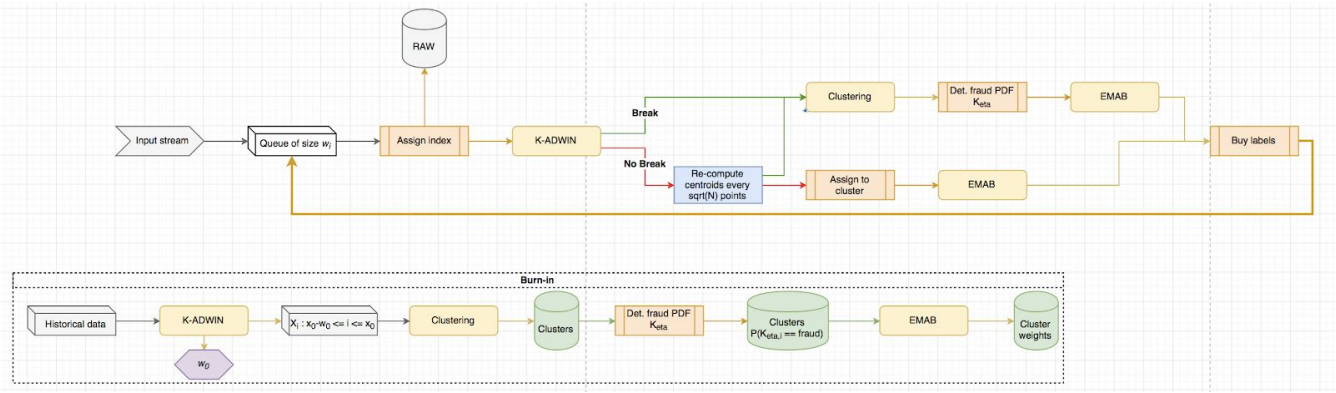
Having defined a entropy measure, although a rough approximation of the true value, that behaves much like epsilon in the epsilon greedy strategy we have one step left in LaC pipeline, grouping the observations into arms.

Clustering

The natural choice for this problem is clustering but no single clustering algorithm was found to perform as desired on the range of datasets as they exhibit heterogeneous characteristics that don't suit a single clustering algorithm. The algorithm DBSCAN, which clusters on the basis of the density of the observations, was found to work best for most sets but doesn't handle anomalies particular well. However, one can set to algorithm to create an outlier 'cluster'. The anomalous observations are grouped together in a sense but don't form a true cluster.

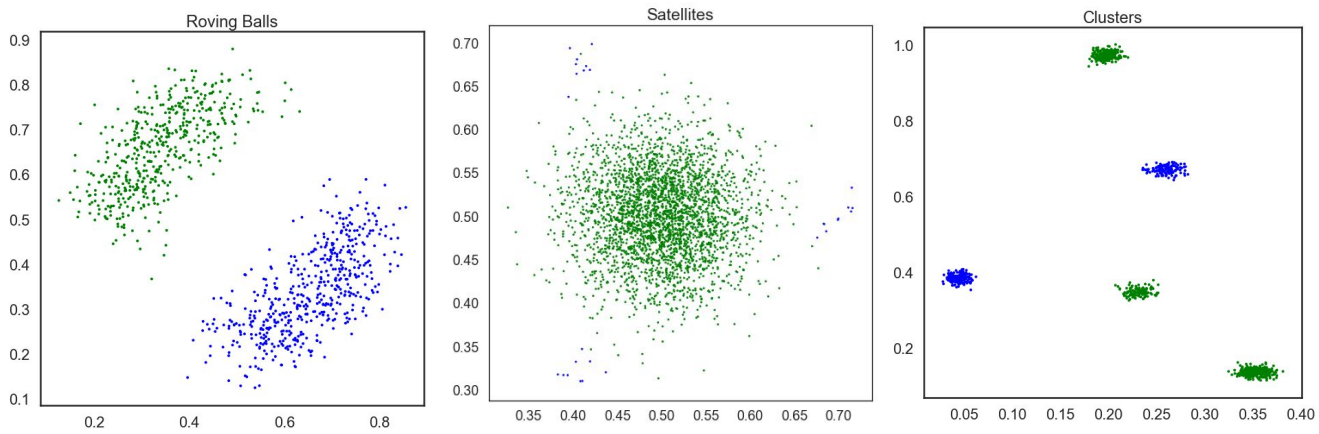
Where DBSCAN is strongest in clustering of larger groups of observations based on density, the Mean-shift algorithm is strong at determining clusters based on the n-dimensional radius set by the user. Hence we can stack the two clustering algorithms, we use DBSCAN to form the primary clusters and indicate the outliers. We then run Mean-shift over the group of outliers to split them into smaller clusters. The combined set of clusters form the arms for our multi-armed bandit problem.

The above three elements come together to form what we believe to be a strong approximation of the optimal Learning at Cost strategy. Unfortunately, we were not able to fully implement the combination of the elements into a unified working example. Please find a diagram of the work-in-progress pipeline.



Data Generators:

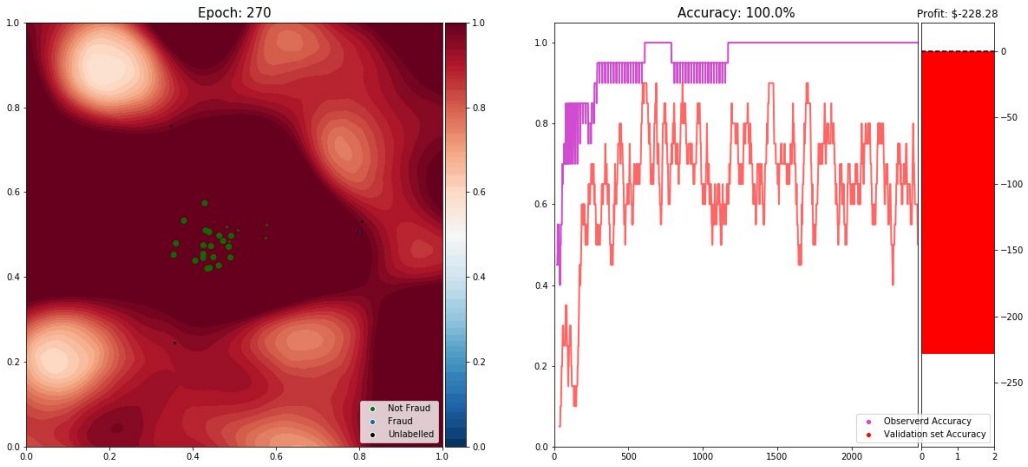
We developed three distinct data generators, each trying to emulate a dynamic dataset that can prove challenging for a ML model to predict and keep updating over time.



- *Roving Balls:*
  - This dataset consists of two clusters of points that revolve around a center point over time.
- *Clusters:*
  - Based on the input parameters different clusters appear in the space over time. This simulates the new types of the target-class appearing.
- *Satellites:*
  - Similar to the *clusters* dataset, but this one contains a majority class in the center, with target classes appearing along the perimeter over time.

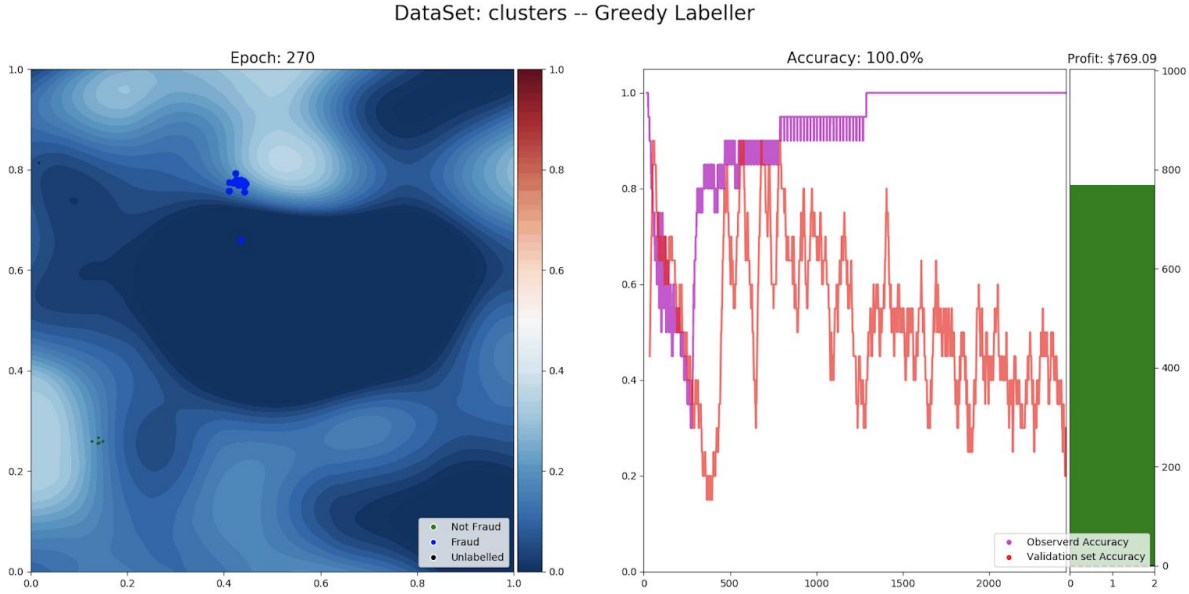
Final results:

The main problem is to find the balance between the accuracy and the profit one wants to make in the model buying algorithm. This has to be done otherwise one only gets short term solutions. Below we will first show what happens when you have an out of balance model. First we will focus on the accuracy. In the figure below we see that the model performs well on the accuracy but we will be instantly bankrupt.



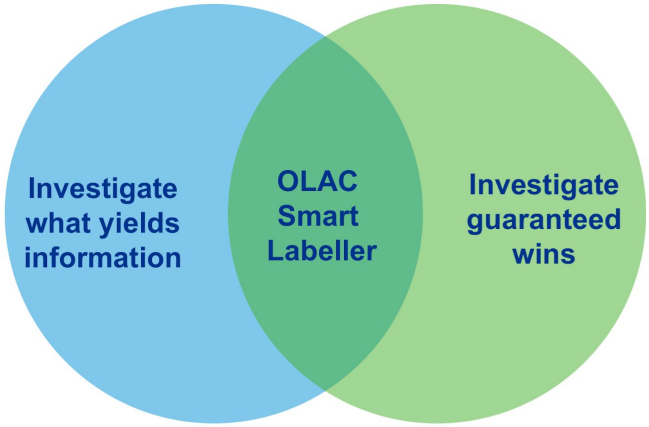
Labelling strategy: UncertaintyLabeller. Labels everything that it is the least sure about.
Dataset: Clusters

When we want to only focus on making a lot of profit and thus optimize the model to look at that only we see that the model only focuses on old known points but will never investigate newly appearing clusters which are also important. This will deliver short term profit but is not a durable solution.

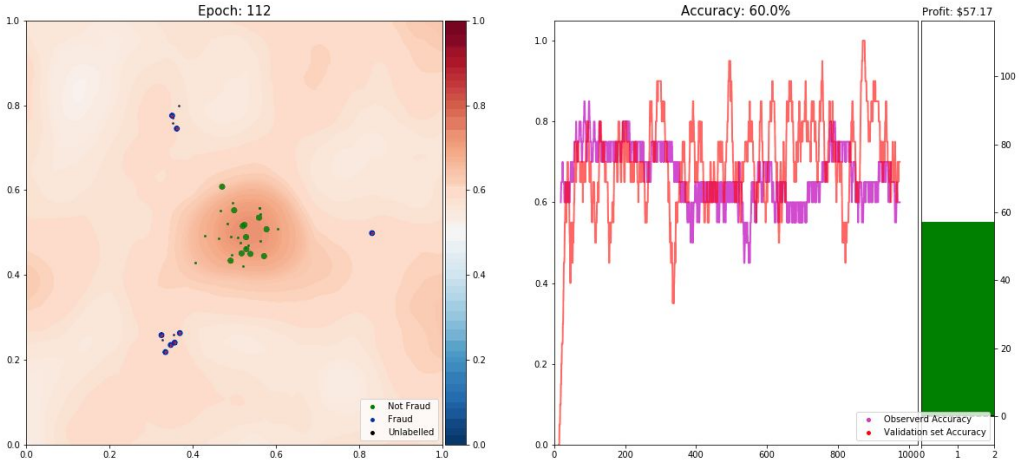


Labelelling strategy: Greedy Labeller; Labels everything it is the most sure about
Dataset: Satellites

So we want to find the balance between making profit and finding the accuracy as you can see below. For this reason we invented our own “smart” labeller. In general terms it looks how dynamic the data is and registers when a major change is happening. Combined this with an estimate of how much information will be gained when a point is investigated and decision is made. This optimum is shown in the figure below.



With our smart labeler we are able to find an optimal balance between the profit and accuracy which is shown in the plot below:



Labelling strategy: ‘Smart’ Labeller
Dataset: Satellites

For more technical information about the “smart” Labeller or the Pipeline see the appendix or the docstrings.

Uncertainties of our approach:

The strength of our approach is that the investigation is done in a strongly controlled environment where for example the data quality and the noise are in control. This is also its weakness when it comes to the applicability in the real world. All the testing is done at by making use of toy problems which resemble parts of the real world but can never mirror the reality completely.

Due to the lack of time the fine tuning of the label buying algorithms are not completely finished this may compromise the performance of the model and final results. So this might be a good point to invest on in the near future or when the model is going to be (re-)used.

Computation time:

The computation of a single model/pipeline is in the order of minutes. This is pretty average. One might be able to speed things up by using multiprocessing. There are some buts here, namely the fact that we use a separated pipeline the communication between the two parts are more complicated than in normal pipelines. This is caused by the effects of queuing. To do this on a large scale on a real dataset we would advise to make use of a cluster to run the models. There is a pretty big bottle neck for when one wants to update the model, this can only be done at the moment single threaded.

We estimate that the impact of these limitations are not crucial. We suspect that the model/artifact is applicable in several use-cases but not in all real world problems.

Future steps:

Suggested future steps are:

- Putting this to a real world problem.
- Think of better label buying strategies
- Finish LAC ‘smart’ labeller and implement it as functionality in the pipeline.

g. **Ethics and legal**

We are going to design a toy problem so that will result in no issues with ethics and legal. No personal data will be used in the analysis so there are no issues with that.

h. We have to design our own problem and being able to map it to a real world problem. The process of distilling a real world problem in a toy problem and then modulate it will need a certain degree of creativity. The other part of the project which needs a significant portion of creativity is the need to integrate online learning with a learning at cost strategie. The largest part of creativity is needed to design the structure/pipeline of

the project. Thus in other words design how all the parts of the algorithm interact with each other.

i. **Contributions and learning goals of team member**s
    i. **Susanne**: Contributions: Programming experience in Python, machine learning and project management.
    Learning goals: Deep Learning techniques, online learning and visualization.
    ii. **John**: Contributions: technical/programming experience in Python, solid math background. Learning goals: Practical implementation of deep learning models, learning about new machine learning techniques
    iii. **Ralph**: Contributions: Programming experience in Python and Nim, know some things about maths and statistics.
    Learning goals: gain/improve knowledge about optimization methodologies, (extended) Kalman filters, Bayesian modelling, streaming/online learning.
    iv. **Bram**: Contributions: Programming experience in Python and C++, Experience with Monte-Carlo simulations, background in physics. Learning goals: Learn how to create and implement a deep learning network.

j. Gains from JADS

i. John: The afternoon sessions we had with Eric really helped build intuition for how neural networks work and for which type of problems they are strong/relevant, so I hope to be able to use these methods in future projects. I will also be remembering his tip for presentations to start with a “dummy demo” and then work toward making it real. I also very much appreciated the lecture of Dick den Hertog about linear programming and optimization.

ii. **Bram**: Gained insight in the cool world of deep learning networks. The tutorial sessions with Eric in the afternoon were a great help for that and are something I will not forget. The introduction weekend was pretty cool, i will not forget that because it were one of my first days at KPMG.

iii. **Ralph**: Because of the JADS project I gained a deeper insight into clustering algorithms, change/break detection and multi-armed bandit strategies.

iv. **Susanne**: The JADS project and the sessions with Eric have inspired me the most in the JADS postmaster. I learned a lot about deep learning networks, and the practical application of it in python using KERAS, tensorflow and sklearn. In practice I gained a lot of skills in object oriented programming, threading, and building visualisations in Python. In addition I enjoyed the classes held by the JADS professors, especially the lectures on visualisation and linear optimization.

References:

[1] Bifet A., Gavaldà R. (2006) Kalman Filters and Adaptive Windows for Learning in Data Streams. In: Todorovski L., Lavrač N., Jantke K.P. (eds) Discovery Science. DS 2006. Lecture Notes in Computer Science, vol 4265. Springer, Berlin, Heidelberg
[2] Rutan, Sarah C. "Adaptive kalman filtering." *Analytical Chemistry* 63.22 (1991): 1103A-1109A.
[3] Van Dobben de Bruyn, Cumulative Sum. "Cumulative sum tests: theory and practice." (1968).
[4] Basseville, Michèle, and Igor V. Nikiforov. *Detection of abrupt changes: theory and application*. Vol. 104. Englewood Cliffs: Prentice Hall, 1993.
[5] Sledge, I.J.; Príncipe, J.C. An Analysis of the Value of Information When Exploring Stochastic, Discrete Multi-Armed Bandits. *Entropy* **2018**, *20*, 155.