MIT Sloan School of Management
Optimization Methods (Fall 2023)

Optimization-aware active learning

Giorgio Demarchi
giodem@mit.edu

Sara Pasquino
pasquino@mit.edu

Abstract

We develop a framework to be leveraged in the scenario where the inputs to an optimization problem are fueled by a machine learning model that is undergoing active learning. In such context, it is possible to leverage duality information to select a batch of to-be-labelled datapoints with awareness of the impact on the optimization problem, which is considered the end goal. We prove the added value of such method and we validate the approach empirically by testing it on a specific task. We find that the optimization aware points selection heuristic can help the optimization problem get quicker to the optimal objective value as high as 4 times the alternatives. As a holistic approach in the scenario, we compare the performance of different methods individually, as well as the combinations of them. Finally, we propose extensions and possible future work to generalize the framework.

(LateX template borrowed from NeurIPS 2019)

Contents

1 Background and Motivation

The two traditional approaches to pool-based active learning are the uncertainty-based method and feature distribution-based method.

Uncertainty-based active learning selects data points to label based on an acquisition function, which returns the set of points on which the model is most uncertain about. Given a model M, and inputs $x \in D_{pool}$, the acquisition function $a(x, M)$ decides what to query next as $x^* = \text{argmax}_{x \in D_{pool}} a(x, M)$. The most common acquisition function, intended to capture the points where the model is most uncertain about, is the predictive entropy. Feature distribution-based active learning selects to-be-labelled datapoints based on low similarity with the already labelled data points. It chooses the datapoints that contain more novel information for the model to be trained on.

The uncertainty approach identifies challenging data but may overlook correlations, while the feature-based approach considers feature relationships for representative data points but is less effective with small query batches. A hybrid approach combines these, selecting points based on a score that balances the two methods using parameter α:

$$Score(x, \theta) = H[y|x, \theta] - \alpha * Sim(x, S)$$

We develop a new method that can be leveraged, alone or jointly with the aforementioned, in a scenario where the classification model output is an input to an optimization model. In this context, we have the opportunity to leverage dual information of the solution of the optimization problem, to select points that have a significant impact on the optimal solution. In the appendix, we take it a step further and overcome one limitation of the other two approaches: we propose a methodology to select the initial training batch, as opposed to randomly pick it.

2 Methodology

2.1 Problem Setting

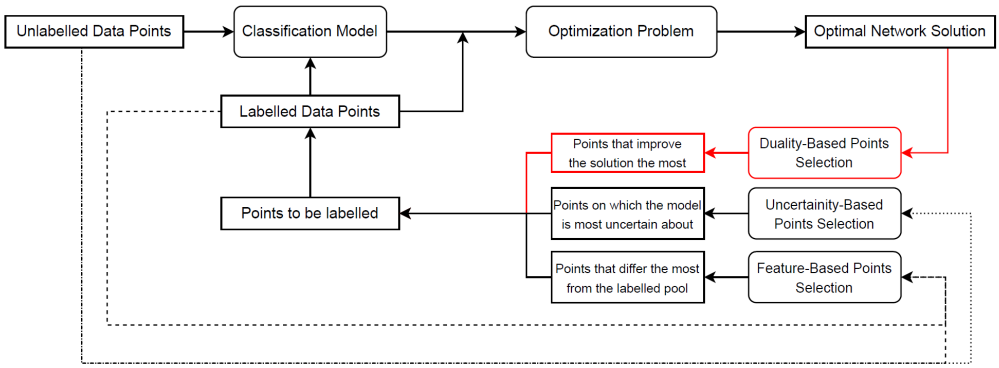

Figure 1: Diagram of problem setting

The problem setting used for this project involves several components.

A binary *classification model* undergoing active learning. For the scope of this project, we have constructed a Logistic Regression model, but the methodology extends to all types of classification models. Being the task supervised in nature, it is trained on the labelled data available, which is a subset of the entire data pool. Once the model is trained, it is run on the entire unlabelled pool and it outputs a binary vector $\hat{\mathbf{z}}$ containing the predictions.

An *optimization model* in the form of a network flow (transportation) cost minimization problem, that takes the vector $\hat{\mathbf{z}}$ as input and uses it to consider the availability of destination nodes.

The pseudo formulation is:

$$\min \quad \sum_i \sum_j c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \quad [...] \tag{2}$$

$$\sum_j x_{ij} \leq \hat{z}_i C_i \quad \forall i \tag{3}$$

$$x_{ij} \geq 0 \quad \forall (i,j) \tag{4}$$

Where (2) encompasses all classical constraints associated with a network flow optimization problem, such as capacity, demand and supply constraints.
Constraint (3) is the additional constraint we implement in our model, which restricts the inflow into destination nodes i through the use of "big-M"-like modelling, according to the previously classified vector $\hat{\mathbf{z}}$. Important to note that $\hat{z}$ is not a variable for the optimization model, making it a linear problem possessing duality properties that will be leveraged in the framework. The optimization problem is solved at each iteration to find the optimal cost and flow matrix X.

These two models are connected by a feedback loop where active learning takes place. At each iteration, an additional to-be-labelled batch is selected. To do so, we propose a new *optimization-aware active learning heuristic*, to be applied at each iteration along with the two existing methodologies outlined in the background section (uncertainty-based and feature distribution-based active learning).

In theory, the expected behavior of each method is the following:

- The *uncertainty-based active learning* model improves out-of-sample accuracy of the ML classifier, by allowing it to train on "hard" points at each iteration.
- The *feature selection-based active learning* model controls for overfitting, allowing the model to train on a diverse pool of new points at each iteration.
- The *optimization-aware active learning* heuristic improves the performance on the optimization model, pushing it closer to the real framework at each iteration.

2.2 Optimization-aware selection heuristic

Consider the feedback loop iteration $k \neq 0$, where the ML classifier has trained on a starting labelled batch B_k, and the optimization framework is solved for the classifier output $\hat{\mathbf{z}}_\mathbf{k}$. Next, consider the split the classified vector $\hat{\mathbf{z}}_k$ into two sets:

$$O = \{\hat{z}_i \in \hat{\mathbf{z}}_\mathbf{k} \mid \hat{z}_i = 0\}$$
$$I = \{\hat{z}_i \in \hat{\mathbf{z}}_\mathbf{k} \mid \hat{z}_i = 1\}$$

The optimal solution gives us the dual value (shadow prices) for the set of constraints in (3) corresponding to $z_i \in O$. We can rank the vector of shadow prices in ascending order, and define a parameter $\bar{p}$ equal to some i^{th} percentile of the shadow price vector. The choice of $\bar{p}$ has to be coherent with the step size, i.e. the number of points to label at each iteration. We define set U as:

$$U = \{z_i \in O \mid p_i \geq \bar{p}\} \tag{5}$$

Secondly, take the set of constraints in (3) corresponding to $z_i \in I$ and compute the total inflow at each node $q_i = \sum_j x_{ij}$. Rank the vector of flows in ascending order, and define a parameter $\bar{q}$ equal to some i^{st} percentile of the cost vector. Again, the choice of $\bar{p}$ has to be coherent with the step size. Define set L as:

$$L = \{z_i \in I \mid q_i \geq \bar{q}\} \tag{6}$$

Finally, join set L and set U, label the respective data points and add them to the labelled pool to obtain B_{k+1}.

The procedure allows to extract those points that, if misclassified, would take us away from the full information solution y the most. Indeed, set U collects all those $z_i = 0$ that correspond to the highest

gain in terms of objective value if they were to be classified as 1. Thus, on those indices, we want the ML classifier to perform very well, in order to minimize, at each iteration, the distance between our network formulation $\hat{y}$ and the real formulation y. On the other hand, set L collects all those $z_i = 1$ where the solution network inflow of x_{ij} is highest. Thus, if these nodes that the ML model classified as 1 were really 0, this would take us further and further away from the real solution and framework y. Then, again, we want the ML classifier to perform really well on these points.

3 Experiments

3.1 Data

We have tested our hypothesis on a fictional scenario with synthetically generated datasets. Although the context of the problem was not critical for our analysis, we choose to simulate a problem for the relocation of in danger species to more suitable habitats and more specifically Red Pandas (because they are cute). We have meticulously constructed the datasets, deliberately incorporating noise, multiple complexities, and correlations to ensure they closely mirror real-world scenarios, which is vital for the integrity of our experiments.

Specifically, we generated one dataset for the classification task and one for the optimization task. The former has a total of 10000 rows and the columns are listed in appendix 2. For our second dataset, we designed the architecture of a transportation network, specifying the configurations, costs, and capacities associated with origin points, transit nodes, and end destinations. The total destinations in this network correspond to the row count of the classification task. As a matter of fact, the classification tasks predicts suitable destinations for the relocation of animals. The code to generate both datasets, useful to understand the complexities that we have introduced, is reported in the appendix.

3.2 Results and discussion

First we have established the full information baseline, i.e. the performance of the classification model and the optimal solution of the optimization problem if all labels are given. Active learning was then simulated through a series of iterative cycles, each time supplying the classification model with an incremental addition of 50 labeled data points. With an assumed total labeling capacity of 1,000 data points, this approach required executing the training of the classification model and resolving the optimization problem upwards of 50 times for each method evaluated.

Evaluating the performance of different active learning frameworks only by looking at the classification accuracy is a an approach that is blind to the ultimate goal of solving the optimization model. Therefore, to effectively evaluate different methodologies, it's imperative not only to measure the machine learning model's accuracy but also to determine the extent to which the solution at each iteration i approximates the solution derived under full information conditions.

We consider four performance metrics:

- **Out of sample classification accuracy**. This is the accuracy of the machine learning model as measured only on the unlabelled data points. This metric is the most important to assess the effectiveness of the active learning on the classification. It is important to note that in a real world scenario this accuracy is not visible to the analyst, as there are no labels to assess the out of sample performance.
- **Validation classification accuracy**. This is the "visible" accuracy, i.e. the accuracy as measured by splitting the labelled pool in train and validation sets. It is materially important in a real world scenario because it can be the case where a model performs very well on out-of-sample, but poorly on the validation set, or vice-versa.
- **Layout factor**. This measures the extent to which the solution at each iteration i approximates the solution derived under full information conditions. Specifically, the layout factor considers the optimal solution variables values X, and is proportional to the 1-norm of the difference between the value at iteration i and the value at full information. The norm is

scaled by dividing for the total supply.

$$LF_i = \frac{\|\hat{X}_i - X_{FI}\|_1}{2 \cdot S} \tag{7}$$

- **Objective factor**. We also consider how far the intermediate network solution is in terms of objective function.

$$OF_i = |1 - \frac{Z^*_{FI}}{\hat{Z}^*_i}| \tag{8}$$

The value of these four metrics as a function of the labelled pool size (measured over multiple random seeds) gives valuable insights on the effectiveness of different active learning methods on the synthesized dataset.

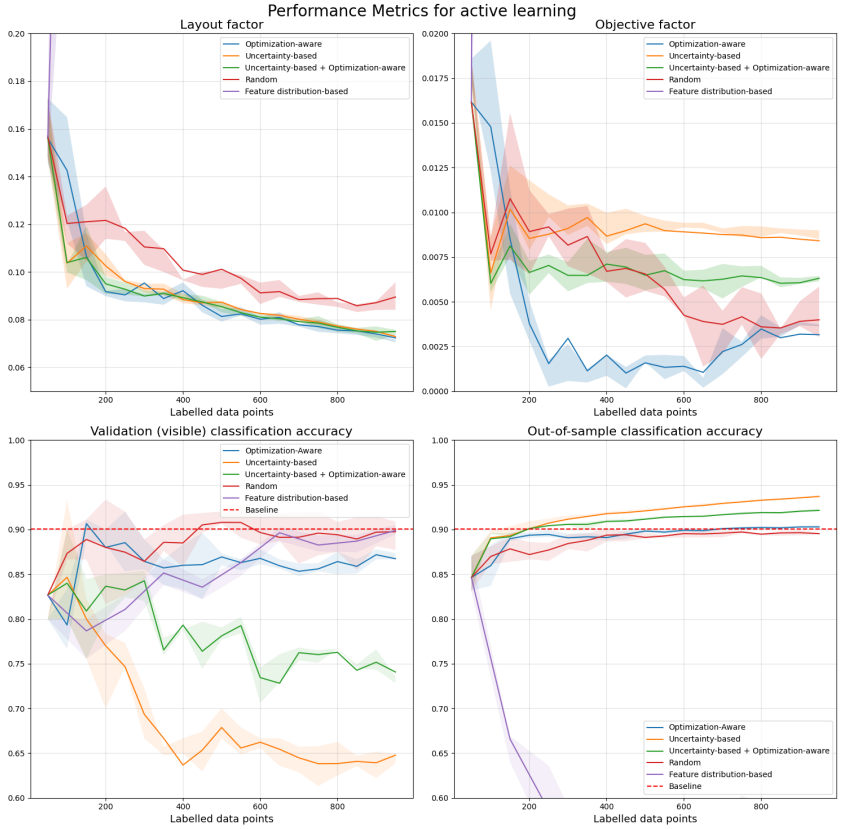

Figure 2: The four performance metrics for all methods tested

Random active learning was tested to have a more comprehensive understanding of the methods. In fact, if random works well enough, it might not be worth to spend time and energy on more sophisticated approaches. Indeed, the random approach proved to be a solid baseline, but not as good as the other approaches especially in terms of layout factor.

Feature distribution-based active learning behaved unexpectedly during our experiments. Specifically, it showed to have a very low out of sample accuracy, which lead the predictions to be so inaccurate that the optimization model becomes infeasible. When this is the case, "layout factor" and "objective factor" are set to one. The reason why this is the exhibited behavior is an open question, but we suspect being due to characteristics of the dataset. Hence, while we disregarded this method for the rest of this experiment, we acknowledge that this is not sufficient proof of bad quality of the method.

Uncertainty-based active learning excels in out-of-sample accuracy but falls short in validation accuracy, a result of focusing on "hard" (difficult to predict) data points. Notably, at 400 labeled examples, the validation accuracy is only 65% compared to 93% out-of-sample accuracy. This discrepancy might mislead analysts about the model's true effectiveness. While it significantly enhances the layout factor in optimization models, its performance on the objective factor is weaker than with random active learning. The interpretation of this is that the model is able to accurately predict most of the datapoints, but it fails on the most critical ones (those that impact the objective function the most, i.e. with high shadow price). This is something that is corrected by the optimization aware approach.

Optimization aware active learning exhibited the expected results: it proves to be the best method in terms of getting closer to the objective function and to the full information network layout. While not being as good as the uncertainty-based on out of sample accuracy, it is better than the random approach, and performs better in terms of validation (visible) accuracy. It is worth noticing that have not implemented the initial batch selection method outlined in append. We expect that by implementing it, the blue curves in the layout factors and objective factors would decrease to make the method even more effective on the first labelled set and even more appealing overall.

Finally, we test a **combination of uncertainty-based and optimization aware** points selection where, at each iteration, half points are labelled through one method and half with the other. This combination is intended to capture the best of both worlds: it maximizes performance on out of sample datapoint and it has an acceptable visible (validation) accuracy, while also closing the gap with the full information network solution faster than other alternatives.

In order to get a more granular insight into the two highlighted active learning methods, we can have a more granular look at the points they are selecting at each iteration. Specifically, we can look at the points (destination nodes) that in the full information optimization solution are receiving a non-zero volume. The total number of such points is 4764, and Figure 3 shows that the optimization aware method is able to capture them around 1.8x faster than the uncertainty based active learning method. Indeed, when the labelled batch is made by 900 points, the former has captured on average 540 points in the final layout and the latter only 320.

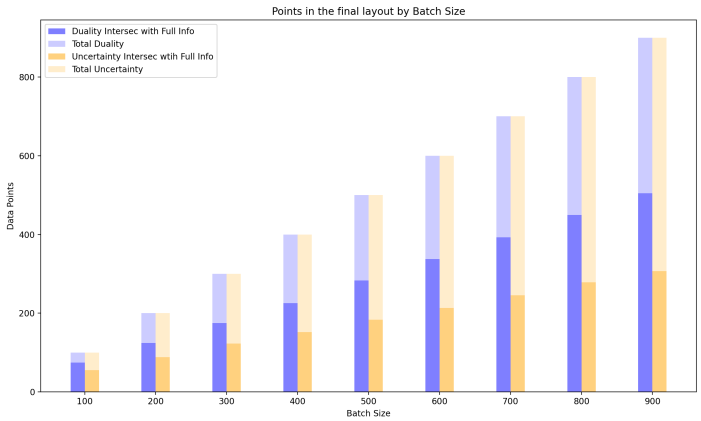

Figure 3: Intersection between final layout and intermediate layout

In conclusion, our experiments show that, under the specified problem settings, optimization aware active learning can significantly decrease the amount of resources needed to achieve the desired approximate solution to the optimization problem.

4 Appendix - Limitations and future work

Our work has opened the door to multiple further workstreams by raising questions related to the performance of the active learning models. Several routes can be explored, including:

- Testing the initial batch I_0 selection framework theorized below. We expect this application to shift all the performance metrics' curves, further improving the quality of the active learning, which at the moment relies on a random selection of I_0.
- We acknowledge the limitations of testing the framework on a single dataset, as the effectiveness of active learning methods can be dependent on the characteristics of the features. Testing on multiple types of dataset is important to assess the consistency of the framework's improvements showcased in this current analysis.
- Feature selection-based active learning performs very poorly in this framework, worsening out-of-sample accuracy at each iteration, thus making the optimization problem infeasible. Further investigation is thus required to assess the underlying causes, and potentially develop viable solutions.
- Testing traditional methods with several different activation functions and dissimilarity metrics to test the framework's robustness.

4.1 Phase 0: Finding an initial batch B_0

[Note: this is just an idea we are developing but that is not yet implemented/tested]

The goal of Phase 0 is to identify the best initial batch B_0 to label, in order to initialize the logistic regression model. The challenge of this step is given by the fact that there is not a solution of a previous iteration to rely on. Traditional active learning approaches are unable to tackle this problem and rely on a random first batch acquisition. In contrast, by leveraging the formulation of the optimization problem we can identify an optimal starting batch.

To find B_0, we leverage the idea behind the optimization-aware active learning methodology previously defined. We want to bound the objective value of the network flow problem below and above, and select said first batch to assure that, after the first iteration, the new solution of the network flow problem will fall within said bounded set.

Lower Bound
We construct a first artificial version of the network flow problem, $y_{\hat{ID}}$, where the vector $\mathbf{z}$ is such that $z_i = 1$ for all $i \in \{1, 2, \dots, n\}$. This represents the most relaxed version of our true problem y, thus the optimal cost resulting from solving this problem is, ceteris paribus, a lower bound to any other possible solution.
Let TP be the optimization network flow problem; then:

$$\hat{y}_{ID} = TP(z_i = 1 \quad \forall i) \tag{9}$$

Note that this solutions is infeasible with respect to the polyhedron corresponding to the true TP y, but we use it as a lower threshold for the real optimal cost.

From this initial vector $\mathbf{z} = \mathbf{1}$, we want to extract the entries z_i that correspond to the highest possible mistake. Equivalently, we want to identify the $z_i = 1$ with the lowest associated objective cost c_i, as those are the nodes where the cost-minimizing model allocates the highest amount of flow.

Thus, if in the true model y those $z_i = 1$ were in fact $z_i = 0$, re-labelling them correctly in y_{UB} would grant the highest jump towards the true model y.

We thus rank the z_i by their associated cost in ascending order, consider as threshold $\hat{c}$ as the first percentile of the cost vector. Then define:

$$L = \{z_i \in \mathbf{z} \mid c_i \leq \bar{c}\} \tag{10}$$

Upper Bound
We now construct a second artificial version of the network flow problem, $y_{\hat{UB}}$, where we want to find the vector $\mathbf{z}$ that gives us an upper bound to the optimal cost. This is achieved by finding the vector $\mathbf{z}$ that yields the most constrained version of the problem.
Ideally, we would want a vector $\mathbf{z}$ such that $z_i = 0$ for all $i \in \{1, 2, \dots, n\}$, but under this constraint the problem $y_{\hat{UB}}$ would have no solution.
Thus, we minimize the number of 1 entries in the possible solution $\mathbf{z}$:

$$\hat{z}_{UB} = \text{argmin} \quad \sum_i z_i \tag{11}$$

$$\text{s.t.} \quad [...] \tag{12}$$

$$\sum_j x_{ij} \leq z_i M \quad \forall i \tag{13}$$

$$x_{ij} \geq 0 \quad \forall (i,j) \tag{14}$$

where constraint (10) includes all the logical constraints of TP. After that, we obtain the corresponding network by solving:

$$\hat{y}_{UB} = TP(\hat{z}_{UB}) \tag{15}$$

Note that there is no guarantee of feasibility of this second solution, but it just serves us as an upper bound to the real problem y. These two solutions give us a lower bound and an upper bound.

Once again, we want to extract from this vector $z_{\hat{UB}}$ the entries z_i associated with the highest possible mistake.

In this case, we compute the dual version of the problem, and extract the n shadow prices corresponding to constraint (5) in the above formulation. Then, we consider the z_i with the highest associated shadow price.

In this case, if in the true model y those $z_i = 0$ were in fact $z_i = 1$, re-labelling them correctly in y_{LB} would grant, once again, the highest jump towards the true model y.

Therefore, we rank the z_i by their associated shadow price in ascending order, and assign a threshold $\bar{p}$ equal to the 99^{th} percentile of the shadow price vector. Then define:

$$U = \{z_i \in \mathbf{z} \mid p_i \geq \bar{p}\} \tag{16}$$

Computing B_0
U and L are therefore sets of points z_i which we want to classify correctly, as they correspond to the highest impact on the objective cost in case of misclassification. Thus, we define:

$$B_0 = U \cup L \tag{17}$$

as the initial batch of points to be labelled.

5 Appendix - Datasets synthesis details

The classification dataset has 9 features and 10000 rows:

Parameter	**Generation Method**
Altitude	Generated from a uniform distribution ranging from 1200 to 3400 meters
Distance from Human Paths	Generated from an exponential distribution, with a mean distance of 500 meters
Livestock Density	Generated from a gamma distribution, shape parameter of 2, scale of 0.5
Vegetation Diversity Index	Generated from a uniform distribution between 0 and 1
Water Source Availability	Binary variable
Human Disturbance Index	Generated from a uniform distribution between 0 and 1
Slope	Generated from a uniform distribution between 0 and 30 degrees
Annual Rainfall	Generated from a normal distribution, mean 1500 mm, std dev 250 mm
Suitable (Target Column)	Generated with k-means clustering, 70% '0', 30% '1'