To Call or not to Call?
Using ML Prediction APIs more Accurately and Economically

Lingjiao Chen [1] **Matei Zaharia** [1] **James Zou** [1]

Abstract

Prediction APIs offered for a fee are a fast-growing industry and an important part of machine learning as a service. While many such services are available, the heterogeneity in their price and performance makes it challenging for users to decide which API or combination of APIs to use for their own data and budget. In this paper, we take a first step towards addressing this challenge by proposing FrugalML, a principled framework that jointly learns the strength and weakness of each API on different data, and performs an efficient optimization to automatically identify the best sequential strategy to adaptively use the available APIs within a budget constraint. Preliminary experiments using ML APIs from Google, Microsoft and Face++ for a facial emotion recognition task show that FrugalML typically leads to more than 50% cost reduction while matching the accuracy of the best single API.

1. Introduction

Machine learning as a service (MLaaS) is a rapidly growing industry. For example, one could use Google prediction API (Goo) to classify an image for $0.0015 or to classify the sentiment of a text passage for $0.00025. MLaaS services are appealing because using such APIs reduces the need to develop one's own ML models. The MLaaS market size was estimated at $1 billion in 2019, and it is expected to grow to $8.4 billion by 2025 (MLa).

Third-party ML APIs come with their own challenges, however. A major challenge is that different companies charge different amounts for similar tasks. For example, for image classification, Face++ charges $0.0005 per image (Fac), which is 67% cheaper than Google (Goo), while Microsoft charges $0.0010 (Mic). Moreover, the prediction APIs of different providers perform better or worse on different types of inputs. For example, accuracy disparities in gender classification were observed for different skin colors (Buolamwini & Gebru). The heterogeneity in their performance and price makes it challenging to decide which API or combination of APIs to use for users' own data and budget.

Our Contributions. In this paper, we take a first step towards this challenge, by proposing FrugalML, a principled framework to pick appropriate APIs depending on users' data and budget constraint. FrugalML first learns the strength and weakness of each API jointly, and then performs an efficient optimization to identify the best adaptive strategy given the user's budget constraint. FrugalML leverages the modular nature of APIs by designing adaptive strategies that call APIs sequentially. For example, we might first send an input to API A. If A returns the label "dog" with high confidence—and we know A is accurate for dogs—then we stop and report "dog". But if A returns "hare" with lower confidence, and we have learned that A is less accurate for "hare", then we adaptively select a second API B to make additional assessment. FrugalML optimizes such adaptive strategies to substantially improve prediction performance over simpler approaches such as model cascades with a fixed quality threshold (Figure 1). Adaptive strategies are challenging to learn and optimize, as the choice of the 2$^{\text{nd}}$ predictor could depend on the prediction and confidence of the first API, and because FrugalML may need to allocate different fractions of its budget to predictions for different classes. We show that under quite general conditions, there is natural sparsity in this problem that we can leverage to make FrugalML efficient (Chen et al., 2020). Preliminary experiments on real world APIs provided by companies like Google, Microsoft, and Face++, have shown that FrugalML typically leads to more than 50% cost savings while matching the accuracy of the best commercial APIs.

2. Preliminaries and Related Work

MLaaS Market. With the growing importance of MLaaS APIs (Fac; Goo; Mic), existing research has largely focused on evaluating individual API for their performance (Yao

[1]Stanford University. Correspondence to: Lingjiao Chen <lingjiao@stanford.edu>.

Proceedings of the 37th International Conference on Machine Learning, EcoPaDL workshop, Vienna, Austria, PMLR 119, 2020.

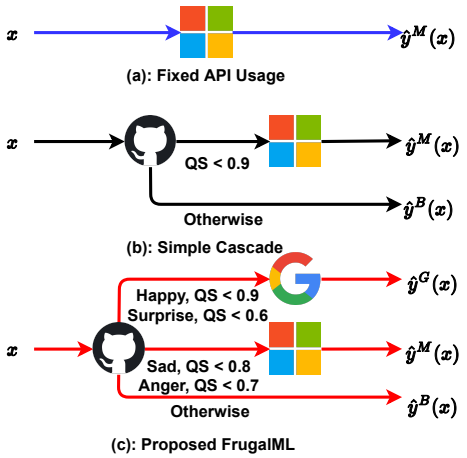

Figure 1. Comparison of different approaches to use ML APIs. Naively calling a fixed API in (a) results in a fixed cost and accuracy. The simple cascade in (b) uses the quality score (QS) from a low-cost open source model to decide whether to call an additional service. Our proposed FrugalML approach, in (c), exploits both the quality score and predicted label to select APIs.

et al.), robustness (Hosseini et al.), pricing (Chen et al.), and applications (Buolamwini & Gebru). On the other hand, FrugalML aims at finding strategies to select apt API from the MLaaS market to reduce costs and increase accuracy. In this paper, we consider a MLaaS market consisting of K different ML services which aim at a same (L-class) classification task. Given a data point x, the kth service returns a predicted label $y_k(x) \in [L]$ and its quality score $q_k(x) \in [0, 1]$. There is also a unit cost associated with each service. Let the vector $\mathbf{c} \in \mathbb{R}^K$ be the unit cost of all APIs. Then $\mathbf{c}_k = 0.01$ means that users need to pay 0.01 every time they call the kth service. Let $y(x)$ be x's true label, and $r^k(x) \triangleq \mathbb{1}_{y_k(x)=y(x)}$ the reward of using the k service on x.

Mixtures of Experts: A natural approach to exploiting multiple predictors is mixture of experts (Jordan & Jacobs, 1994), which uses a gate function to decide which expert to use. Substantial research has focused on developing gate function models such as SVMs (Collobert et al., 2002). However, directly applying mixture of experts for MLaaS would result in fixed cost and thus would not allow users to specify a budget constraint as in FrugalML.

Model Cascades: Cascades consisting of a sequence of models are useful to balance the quality and runtime of inference (Viola & Jones, 2001; Kang et al., 2017; Sun et al.; Xu et al., 2014). While model cascades use predicted quality score *alone* to avoid calling computationally expensive models, FrugalML' strategies can utilize both quality score and predicted class to select a downstream expensive add-on service. Designing such strategies requires solving a significantly harder optimization problem, e.g., choosing how to divide the budget between classes (§3), but also improves performance substantially over the simple cascade (§4).

3. FrugalML: a Formal Framework for Frugally Leveraging ML Services

In this section, we present FrugalML, a formal framework for API calling strategies to obtain accurate and cheap predictions from a MLaaS market. We generalize the scheme in Figure 1 (c) to K ML services and L label classes. Let a tuple $s \triangleq (\mathbf{p}^{[1]}, \mathbf{Q}, \mathbf{P}^{[2]})$ represent a calling strategy produced by FrugalML. Given an input data x, FrugalML first calls a *base service*, denoted by $A_s^{[1]}$, which with probability $\mathbf{p}_i^{[1]}$ is the ith service and returns quality score $q_i(x)$ and label $y_i(x)$. Let D_s be the indicator of whether the quality score is smaller than the threshold value $\mathbf{Q}_{i,y_i(x)}$. If $D_s = 1$, then FrugalML invokes an *add-on service*, denoted by $A_s^{[2]}$, with probability $\mathbf{P}^{[2]}_{i,y_i(x),j}$ being the jth service and producing $y_j(x)$ as the predicted label $\hat{y}^s(x)$. Otherwise, FrugalML simply returns label $\hat{y}^s(x) = y_i(x)$ from the base service. This process is summarized in Figure 2. Note that the strategy is adaptive: the choice of the add-on API can depend on the predicted label and quality score of the base model.

The set of possible strategies can be parametrized as $S \triangleq \{(\mathbf{p}^{[1]}, \mathbf{Q}, \mathbf{P}^{[2]}) | \mathbf{p}^{[1]} \succcurlyeq \mathbf{0} \in \mathbb{R}^K, \mathbf{1}^T\mathbf{p}^{[1]} = 1, \mathbf{Q} \in \mathbb{R}^{K\times L}, \mathbf{0} \preccurlyeq \mathbf{Q} \preccurlyeq \mathbf{1}, \mathbf{P}^{[2]} \in \mathbb{R}^{K\times L\times K}, \mathbf{P}^{[2]} \succcurlyeq \mathbf{0}, \mathbf{1}^T\mathbf{P}^{[2]}_{k,\ell,\cdot} = 1\}$. Our goal is to choose the optimal strategy s^* that maximizes the expected accuracy while satisfies the user's budget constraint b. This is formally stated as below.

Definition 1. *Given a user budget b, the optimal Frugal*ML *strategy $s^* = (\mathbf{p}^{[1]*}, \mathbf{Q}^*, \mathbf{P}^{[2]*})$ is*

$$s^* \triangleq \arg\max_{s\in S} \mathbb{E}[r^s(x)] \text{ s.t. } \mathbb{E}[\eta^{[s]}(x, \mathbf{c})] \le b, \qquad (3.1)$$

where $r^s(x) \triangleq \mathbb{1}_{\hat{y}^s(x)=y(x)}$ is the reward and $\eta^{[s]}(x, \mathbf{c})$ the total cost of strategy s on x.

Remark 1. *The above definition can be generalized to wider settings. For example, instead of 0-1 loss, the reward can be negative square loss to handle regression tasks. We pick the concrete form for demonstration purposes. The cost of strategy s, $\eta^{[s]}(x, \mathbf{c})$, is the sum of all services called on x. For example, if service 1 and 2 are called for predicting x, then $\eta^{[s]}(x, \mathbf{c})$ becomes $\mathbf{c}_1 + \mathbf{c}_2$.*

Given the above formulation, a natural question is how to solve it efficiently. In the following, We first highlight an interesting property of the optimal strategy, *sparsity*, which inspires the design of the efficient solver, and then present the algorithm for the solver.

Sparsity Structure. We observe that one interesting property of problem 3.1: if it is feasible and has unique opti-

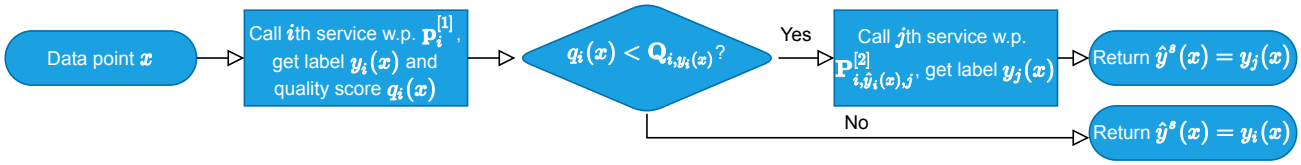

Figure 2. In FrugalML, a base service is first selected and called. If its quality score is smaller than the threshold for its predicted label, FrugalML chooses an add-on service to invoke and returns its prediction. Otherwise, the base service's prediction is returned.

Table 1. ML services price. Unit: USD/10,000 queries. A convolutional neural network (CNN) available from GitHub model is also used in addition to commercial APIs.

ML service	Price	ML service	Price
Google Vision (Goo)	15	MS Face (Mic)	10
GitHub (CNN) (FER)	0.001	Face++ (Fac)	5

mal solution, then its optimal solution must be sparse, i.e., $\|\mathbf{p}^{[1]*}\| \leq 2$. In other words, the optimal strategy should only choose the base service from at most two services (instead of K) in the MLaaS market.

Sketch of Solving Problem 3.1. This sparsity structure sheds light on solving problem 3.1 efficiently. In fact, the sparsity structure implies problem 3.1 becomes equivalent to a *master problem*

$$\max_{(i_1,i_2,p_1,p_2,b_1,b_2)\in C} p_1 g_{i_1}(b_1/p_1) + p_2 g_{i_2}(b_2/p_2) \quad (3.2)$$

where $C = \{(i_1,i_2,p_1,p_2,b_1,b_2) | i_1,i_2 \in [K], p_1,p_2 \geq 0, p_1+p_2 = 1, b_1,b_2 \geq 0, b_1+b_2 \leq b\}$, and $g_i(b')$ is the optimal value of the *subproblem*

$$\max_{\mathbf{Q},\mathbf{P}^{[2]}:s=(\mathbf{e}_i,\mathbf{Q},\mathbf{P}^{[2]})\in S} \mathbb{E}[r^s(x)\ s.t.\ \mathbb{E}[\eta^s(x)] \leq b' \quad (3.3)$$

Here, the master problem decides which two services (i_1,i_2) can be the base service, how often (p_1,p_2) they should be invoked, and how large budgets (b_1,b_2) are assigned, while for a fixed base service i and budget b', the subproblem maximizes the expected reward.

Hence, the solver to Problem 3.1 can be built based on the above decomposition. First, the conditional accuracy $\mathbb{E}[r_i(x)|D_s, A_s^{[i]} = j, y_j(x)]$ is estimated from the training data. Next for each i, we solve subproblem 3.3 for a few value of b' and then construct $g_i(\cdot)$ using linear interpolation. Finally, the master problem can be transformed into enumerating a few linear programmings due to the construction of $g_i(\cdot)$ and thus solved efficiently.

4. Preliminary Experiments

We compare the accuracy and incurred costs of FrugalML to that of real world ML services on a few datasets, which

Table 2. Datasets Statistics.

Datasets	# Data point
FER+ (Barsoum et al.)	6,358
RAFDB (Li et al.)	15,339
AFFECTNET (Mollahosseini et al., 2019)	287,401

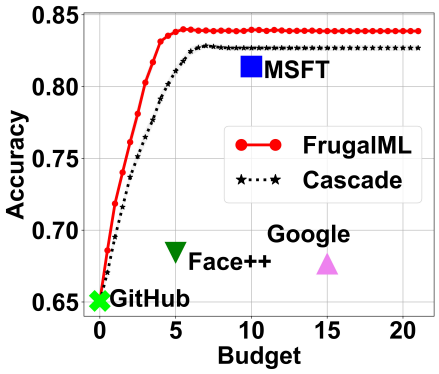

Figure 3. Accuracy and cost trade-offs on dataset FER+.

demonstrates the trade-offs between accuracy and cost as well as the cost savings achieved by FrugalML.

Tasks, ML services, and Datasets. We focus on a common computer vision task, facial emotion recognition, where the goal is to predict the emotion expressed in facial images. The ML services used as well as their prices are summarized in Table 1. We evaluate the performance of FrugalML as well as those APIs on three datasets, namely, FER+, RAFDB, and AFFECTNET. Originally FER+ contains training and testing partitions, and we only use its testing partition since the GitHub model was pretrained on its training partition. Both RAFDB and AFFECTNET contain images with basic emotions and component emotions, where we only use the formal ones. The statistics of those datasets are summarized in Table 2.

Accuracy and Cost Trade-offs on FER+. We start by measuring the accuracy and cost trade-offs achieved by FrugalML on FER+, shown in Figure 3. While using any single ML service incurs a fixed cost, FrugalML allows users to pick any point in its trade-off curve, offering substantial

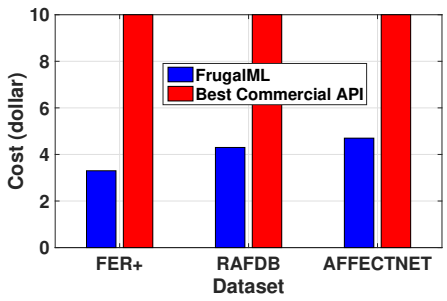

Figure 4. Cost of FrugalML while matching the accuracy of the best single commercial API. Across all datasets, FrugalML consistently leads to more than 50% cost savings.

flexibility. In addition to cost saving, FrugalML can achieve higher accuracy than the best commercial API (Microsoft) while matching its cost. Compared to the simple cascade approach, FrugalML consistently reaches a higher accuracy while using the same budget.

Analysis of Cost Savings. We also evaluate how much cost can be saved by FrugalML to reach the highest accuracy produced by a single API on all the emotion recognition datasets. As shown in Table 4, FrugalML can typically save more than half of the cost. In fact, on datasets FER+, only 33% cost is needed to achieve the same accuracy as the best API (Microsoft API). This is likely because the base service's quality score is highly correlated to its prediction accuracy, and thus FrugalML only needs to call expensive services for a few difficult data points and relies on the (cheap) base services for the relatively easy data points.

5. Conclusion and Open Problems

In this work we proposed FrugalML, a formal framework for identifying the best strategy to call ML APIs given a user's budget. FrugalML identifies the optimal strategy that adaptively pick an API for depending on the users' budget constraint. Preliminary experiments demonstrate that FrugalML leads to significant cost reduction and accuracy improvement. Our research characterized the substantial heterogeneity in cost and performance across available ML APIs, which is useful in its own right and also leveraged by FrugalML. As for future work, we are working on more in-depth theoretical analysis of FrugalML. Extending FrugalML to produce calling strategies for ML tasks beyond classification (e.g., object detection and language translation) as well as comprehensive empirical evaluation is another interesting future direction. As a resource to stimulate further research in MLaaS, we also plan to release the dataset used to develop FrugalML.

References

Pretrained facial emotion model from GitHub. https://github.com/WuJie1010/Facial-Expression-Recognition.Pytorch. [Accessed March-2020].

Face++ Emotion API. https://www.faceplusplus.com/emotion-recognition/. [Accessed March-2020].

Google Vision API. https://cloud.google.com/vision. [Accessed March-2020].

Machine Learning as a Service Market Report . https://www.mordorintelligence.com/industry-reports/global-machine-learning-as-a-service-mlaas-market.

Microsoft computer vision API. https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision. [Accessed March-2020].

Barsoum, E., Zhang, C., Ferrer, C. C., and Zhang, Z. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *ICMI 2016*.

Buolamwini, J. and Gebru, T. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *FAT 2018*.

Chen, L., Koutris, P., and Kumar, A. Towards model-based pricing for machine learning in a data marketplace. In *SIGMOD 2019*.

Chen, L., Zaharia, M., and Zou, J. FrugalML: How to use ML prediction apis more accurately and cheaply. *CoRR*, abs/2006.07512, 2020.

Collobert, R., Bengio, S., and Bengio, Y. A parallel mixture of SVMs for very large scale problems. *Neural Computation*, 14 (5):1105–1114, 2002.

Hosseini, H., Xiao, B., and Poovendran, R. Google's cloud vision API is not robust to noise. In *ICMLA 2017*.

Jordan, M. I. and Jacobs, R. A. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

Kang, D., Emmons, J., Abuzaid, F., Bailis, P., and Zaharia, M. Noscope: Optimizing deep cnn-based queries over video streams at scale. *PVLDB*, 10(11):1586–1597, 2017.

Li, S., Deng, W., and Du, J. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *CVPR 2017*.

Mollahosseini, A., Hasani, B., and Mahoor, M. H. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Trans. Affect. Comput.*, 10(1):18–31, 2019.

Sun, Y., Wang, X., and Tang, X. Deep convolutional network cascade for facial point detection. In *CVPR 2013*.

Viola, P. and Jones, M. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.

Xu, Z. E. et al. Classifier cascades and trees for minimizing feature evaluation cost. *J. Mach. Learn. Res.*, 15(1):2113–2144, 2014.

Yao, Y., Xiao, Z., Wang, B., Viswanath, B., Zheng, H., and Zhao, B. Y. Complexity vs. performance: empirical analysis of machine learning as a service. In *IMC 2017*.