Task-based End-to-end Model Learning in Stochastic Optimization

Priya L. Donti,*[1,2] Brandon Amos[1], and J. Zico Kolter[1]

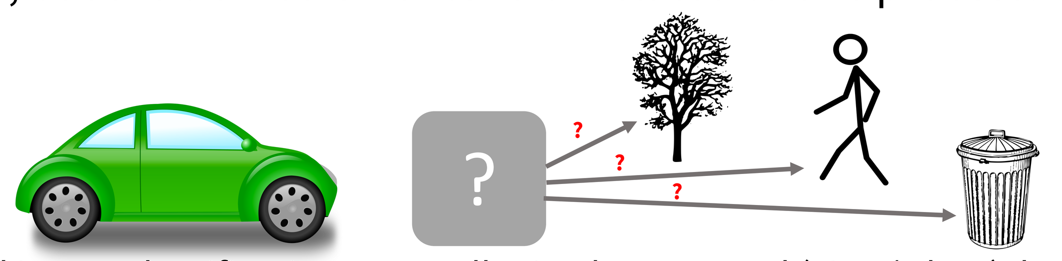
[1] School of Computer Science [2] Dept. of Engineering & Public Policy (Carnegie Mellon University)

* pdonti@cmu.edu https://github.com/locuslab/e2e-model-learning

We propose a task-based approach for learning probabilistic ML models in the loop of stochastic optimization.

Introduction

Predictive algorithms often operate within some larger process, but are trained on criteria unrelated to this process.



Standard image classification treats all mistakes as equal (via 0/1 loss), but the wrong *kind* of mistake could lead to undesirable driving behavior.

We train a model not (solely) for predictive accuracy, but to minimize the task-based objective we ultimately care about.

Related Work

Bengio [1] uses task-based learning in a deterministic setting by tuning a financial price prediction model based on returns from a hedging strategy that employs it. We extend this work to a stochastic optimization setting, and propose a general procedure for task-based learning in this domain.

Setting: Stochastic Optimization

Stochastic optimization makes decisions under uncertainty by optimizing objectives governed by a random process [2].

Given: Input-output pairs $(x, y) \sim \mathcal{D}$ for real, unknown $\mathcal{D}$
Output: "Optimal" actions z, by optimizing task cost f via:

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \mathbf{E}_{x,y\sim\mathcal{D}}[f(x,y,z)] \\ \text{subject to} \quad & \mathbf{E}_{x,y\sim\mathcal{D}}[g_i(x,y,z)] \le 0, \quad i = 1,\dots,n_{ineq} \\ & h_i(z) = 0, \quad i = 1,\dots,n_{eq} \end{aligned}$$

E.g.: x = pixels, y = segmentation map, z = vehicle path, f = driving quality, g, h = constraints in physical environment

Knowing $\mathcal{D}$ would enable us to choose truly optimal $z^\star$, but in reality we don't know $\mathcal{D}$… so we turn to machine learning.

Standard ML Approaches

Standard approaches to ML in stochastic optimization are:

1) Traditional model learning: Model conditional distribution $y|x$ by learning distribution parameters θ.

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} -\log p(y^{(i)}|x^{(i)};\theta).$$

Drawback: Model bias in (common) non-realizable case.

2) Model-free policy optimization: Map directly from inputs x to actions z. Forgo learning model of y.

Drawback: Data-inefficient.

We offer an intermediate approach where we both learn a model of y AND adjust model parameters with respect to z.

Our Method

Our model-based approach incorporates knowledge of the final task.

We provide a general framework for *adjusting model parameters* in stochastic optimization to *optimize closed-loop performance* of the resulting system.

Our method chooses parameters θ for $y|x$ to minimize *task loss*:

$$\underset{\theta}{\text{minimize}}\ L(\theta) = \mathbf{E}_{x,y\sim\mathcal{D}}[f(x,y,z^\star(x;\theta))]$$

where $z^\star(x;\theta)$ are the optimal actions *w.r.t. our predictions*, i.e.

$$z^\star(x;\theta) = \underset{z}{\text{argmin}}\ \mathbf{E}_{y\sim p(y|x;\theta)}[f(x,y,z^\star(x;\theta))] \qquad (*)$$

(with constraints omitted above for simplicity of illustration).

Algorithm

input: $\mathcal{D}$ *// ability to sample from true, unknown distribution*
initialize: θ *// initial distribution parameters*

for $t = 1, \dots T$ **do**
  sample $(x, y) \sim \mathcal{D}$
  compute $z^\star(x;\theta)$ via Equation (*) (with constraints)
  // step in violated constraint or objective
  if $\exists i$ s.t. $g_i(x,y,z^\star(x;\theta)) > 0$ **then**
    update θ **with** $\nabla_\theta g_i(x,y,z^\star(x;\theta))$
  else
    update θ **with** $\nabla_\theta f(x,y,z^\star(x;\theta))$
  end if
end for

Technical Challenge: Argmin Differentiation

The gradient of the objective depends on the argmin result $z^\star(x;\theta)$:

$$\frac{\delta L}{\delta \theta} = \frac{\delta L}{\delta z^\star}\frac{\delta z^\star}{\delta \theta} = \frac{\delta L}{\delta z^\star}\frac{\delta\ \underset{z}{\text{argmin}}\ \mathbf{E}_{y\sim p(y|x;\theta)}[f(x,y,z^\star(x;\theta))]}{\delta\theta}.$$

To obtain the gradient, we write the KKT optimality conditions of (*). Assuming convexity allows us to replace the general equality constraints $h(z) = 0$ with the linear constraints $Az = b$.

A point (z, λ, ν) is a primal-dual optimal point if it satisfies

$$\begin{aligned} \mathbf{E}g(z) &\le 0 \\ Az &= b \\ \lambda &\ge 0 \\ \lambda \circ \mathbf{E}g(z) &= 0 \\ \nabla_z \mathbf{E}\, f(z) + \lambda^T \nabla_z \mathbf{E}\, g(z) + A^T\nu &= 0 \end{aligned}$$

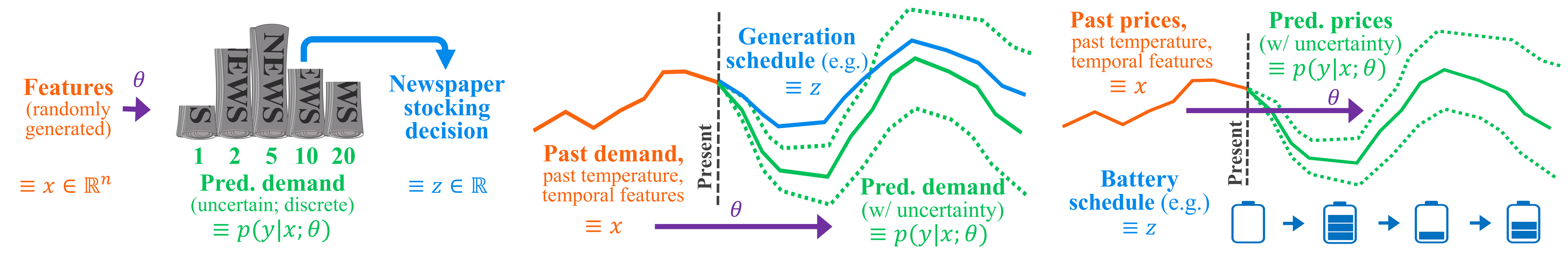
where expectations are over $y \sim p(y|x;\theta)$, g is the vector of all inequality constraints, and the dependence on x and y is via f and g.

Differentiating these equations and applying the implicit function yields linear equations we can solve to get the necessary Jacobians.

In practice, we use SQP to solve (*), finding $z^\star(x;\theta)$ via a solution for fast argmin differentiation in QPs [3] and then taking derivatives through the quadratic approximation at this optimum.

Experiments

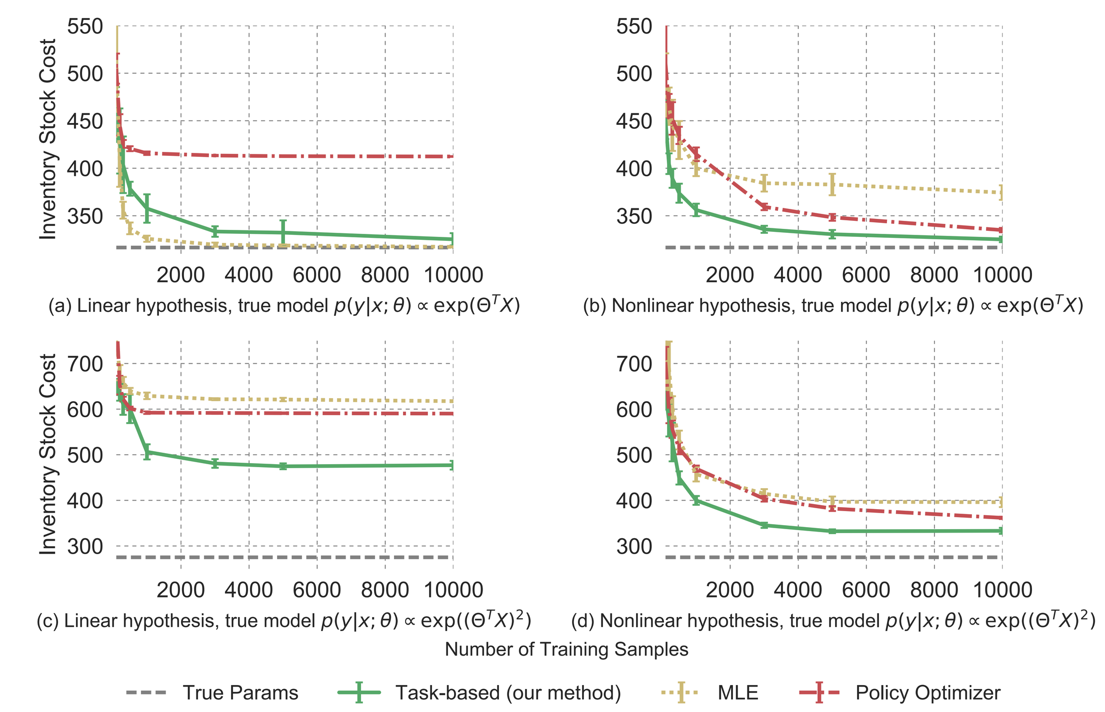
We outperform both traditional model learning and model-free policy optimization in terms of task cost, the objective of actual interest in the closed-loop system.



(a) Inventory stock problem (b) Load forecasting/generator scheduling (c) Price forecasting/battery arbitrage

Inventory stock problem: Order quantity z of a product to minimize costs over stochastic demand y.

$$\underset{z\in\mathbb{R}}{\text{minimize}}\ \mathbf{E}_y[f_{stock}(y,z)] = \mathbf{E}_y\left[c_0 z + \frac{1}{2}q_0 z^2 + c_b[y-z]_+ + \frac{1}{2}q_b([y-z]_+)^2 + c_h[z-y]_+ + \frac{1}{2}q_h([z-y]_+)^2\right]$$

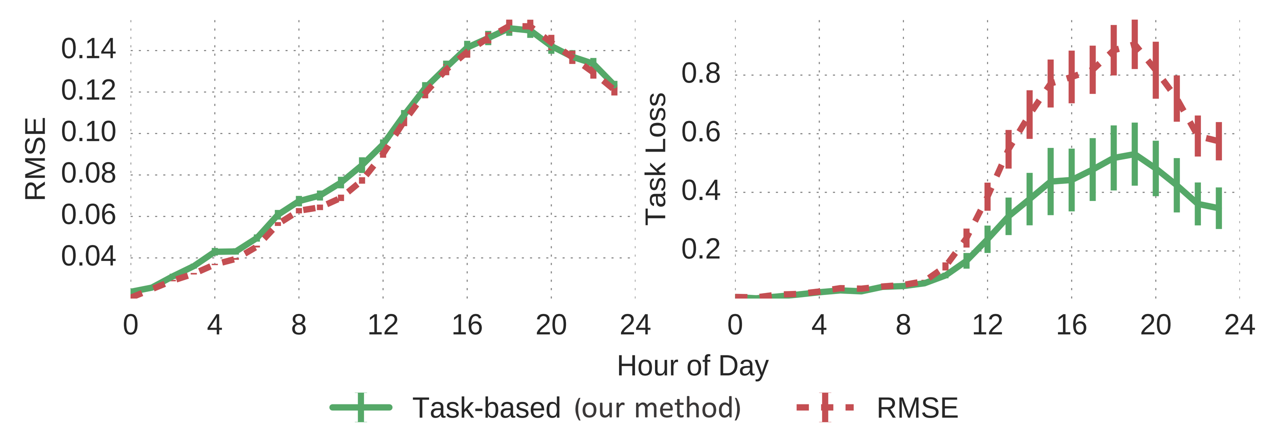


Our task-based model outperforms:

- Traditional model-based MLE in all but the realizable case, correcting for effects of model misspecification.
- Model-free policy optimizer, due to increased data efficiency.

Generator scheduling: Schedule electricity generation z to minimize costs over stochastic demand y.

$$\underset{z\in\mathbb{R}^{24}}{\text{minimize}}\ \mathbf{E}_y\left[\sum_{i=1}^{24} \gamma_s[y_i - z_i]_+ + \gamma_e[z_i - y_i]_+ + \frac{1}{2}\|z_i - y_i\|^2\right] \quad \text{subject to } |z_i - z_{i+1}| \le c_{\text{ramp}}.$$



While an RMSE-minimizing model produces "objectively" better predictions, our task-based model yields a **38.6% improvement** in task performance over the RMSE model.

Battery arbitrage: Schedule battery charge/discharge z to minimize costs over energy prices y.

$$\underset{z_{\text{in}}, z_{\text{out}}, z_{\text{state}}\in\mathbb{R}^{24}}{\text{minimize}}\ \mathbf{E}_y\left[\sum_{i=1}^{24} y_i(z_{\text{in}} - z_{\text{out}})_i + \lambda\left\|z_{\text{state}} - \frac{B}{2}\right\|^2 + \epsilon\|z_{\text{in}}\|^2 + \epsilon\|z_{\text{out}}\|^2\right]$$

subject to

$$\begin{aligned} & z_{\text{state},\,i+1} = z_{\text{state},\,i} - z_{\text{out},i} + \gamma_{\text{eff}}\, z_{\text{in},i} \\ & z_{\text{state},\,1} = B/2\ ,\ \ 0 \le z_{\text{state}} \le B \\ & 0 \le z_{\text{in}} \le c_{\text{in}}\ ,\ \ 0 \le z_{\text{out}} \le c_{\text{out}} \end{aligned}$$

Hyperparameters		Task cost		% improvement
λ	ϵ	RMSE net	Task-based net (our method)	
0.1	0.05	-1.45 ± 4.67	-2.92 ± 0.30	102
1	0.5	4.96 ± 4.85	2.28 ± 2.99	54
10	5	131.08 ± 144.86	95.88 ± 29.83	27
35	15	172.66 ± 7.38	169.84 ± 2.16	2

Here, y is inherently very stochastic, and our task-based net demonstrates more reliable performance than an RMSE-minimizing net.

Conclusions

We propose an end-to-end approach for learning machine learning models used within stochastic optimization. Our experiments indicate that our task-based model learning method outperforms both traditional MLE and "black-box" policy-optimizing methods with respect to task cost.

Future work includes an extension of this method to stochastic learning models with multiple rounds, and further to model predictive control and full reinforcement learning settings.

[1] Bengio, Y. (1997). Using a financial training criterion rather than a prediction criterion. *International Journal of Neural Systems*, *8*(04), 433-443.
[2] Shapiro, A., & Philpott, A. (2007). A tutorial on stochastic programming. *Manuscript. Available at* www2.isye.gatech.edu/ashapiro/publications.html.
[3] Amos, B. & Kolter, J.Z. (2017). OptNet: Differentiable Optimization as a Layer in Neural Networks. *Proceedings of the 34th International Conference on Machine Learning, in PMLR* 70:136-145

This work was supported by the NSF GRFP under Grant No. DGE1252522, as well as the Department of Energy Computational Science Graduate Fellowship.