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. Example: Standard image classification treats all mistakes as equal (via e.g. 0/1 loss). However, making the wrong kind of mistake could lead to egregious autonomous driving behavior.

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

**Setting: Stochastic Optimization**

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

**Technical Challenge: Argmin Differentiation**

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

$$\frac{\Delta L}{\Delta \theta} = \frac{\Delta L}{\Delta z} \frac{\Delta z^*}{\Delta \theta} = \frac{\Delta \arg\min_{z} E_{g}(g(z)|\theta)}{\Delta \theta}.$$

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 $(x, y, v)$ is a primal-dual optimal point if it satisfies

$$\begin{align*}
E_{g}(g) &\le b, \\
\lambda &\ge 0, \\
\lambda \cdot E_{g}(g) &\le 0, \\
\n\n\end{align*}$$

where expectations are over $y = p(y|\cdot ; g)$, $g$ 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 $z^*(\cdot; \theta)$ via a solution for fast argmin differentiation in QPs [3] and then taking derivatives through the quadratic approximation at this optimum.

**Our Model**

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 $\theta$ for $f(x, y)$ to minimize loss:

$$\min_{\theta} L(\theta) = E_{g}(g(z)|\theta)$$

where $z^*(\cdot; \theta)$ are the optimal actions w.r.t. our predictions.

(with constraints omitted above for simplicity of illustration).

**Algorithm**

Input: Example $x, y \sim \mathcal{D}$

Initialize: $\theta$ // initial distribution parameters

for $t = 1, \ldots, T$ do

compute $z^*(\cdot; \theta)$ via Equation (*) (with constraints)

initialize $\alpha_t$ // learning rate

// Step in violated constraint or objective

if $3 \leq t$, $g(z^*(\cdot; \theta)) > 0$ then

update $\theta \leftarrow \theta - \alpha_t E_z f(x, z^*(\cdot; \theta))$

else

update $\theta \leftarrow \theta - \alpha_t E_z f(x, z^*(\cdot; \theta))$

end if

end for

**Our Method**

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.

### Experiments

- **Inventory stock problem:** Order quantity $z$ of a product to minimize costs over stochastic demand $y$. min $E_z f_{stock}(y, z) = E_z \left[ x_0 + \frac{1}{2} q_0 x_2 + c_0 (z - y)^2 + \frac{1}{2} q_0 (y - z)^2 \right]$

### Standard ML Approaches

**1)** Traditional model learning: Model conditional distribution $y|\cdot$. by learning distribution parameters $\theta$, minimize $\sum_{i=1} \log p(y|\cdot; \theta)$.

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

**Conclusion**

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 methods and a "black-box" policy-optimizing methods with respect to task cost.

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