TaskMet: Task-Driven Metric Learning for Model Learning

Dishank Bansal [1] **Ricky Chen** [1] **Mustafa Mukadam** [1] **Brandon Amos** [1]

Abstract

Deep learning models are often used with some downstream task. Models solely trained to achieve accurate predictions may struggle to perform well on the desired downstream tasks. We propose using the task's loss to learn a metric which parameterizes a loss to train the model. This approach does not alter the optimal prediction model itself, but rather changes the model learning to emphasize the information important for the downstream task. This enables us to achieve the best of both worlds: a prediction model trained in the original prediction space while also being valuable for the desired downstream task. We validate our approach through experiments conducted in two main settings: 1) decision-focused model learning scenarios involving portfolio optimization and budget allocation, and 2) reinforcement learning in noisy environments with distracting states.

1. Introduction

Machine learning prediction models are typically trained to maximize the likelihood on a training dataset. While the models are capable of approximating the underlying data generating process to predict the output, they are prone to approximation errors due to limited training data and model capacity. These errors lead to suboptimal performance in downstream tasks where the models are used. *End-to-end task-based model learning methods* is an the area of machine learning research that uses information from the downstream task to improve the model's performance on that particular task. These methods work well for financial price predictions (Bengio, 1997; Elmachtoub & Grigas, 2022), inventory stock, demand, and price forecasting (Donti et al., 2017; Elmachtoub & Grigas, 2022; El Balghiti et al., 2019; Mandi et al., 2020; Liu et al., 2023), dynamics modeling for RL (Farahmand et al., 2017; Amos et al., 2018; Farahmand, 2018; Bhardwaj et al., 2020; Voelcker et al., 2022; Nikishin et al., 2022), and budget allocation, matching, and recommendation (Kang et al., 2019; Wilder et al., 2019; Shah et al., 2022). We notate the model's loss on the prediction problem as $\mathcal{L}_{\text{pred}}$ and the downstream task loss as $\mathcal{L}_{\text{task}}$.

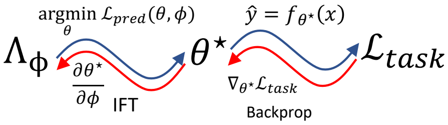
$\operatorname{argmin}_{\theta} \mathcal{L}_{pred}(\theta, \phi)$
$\hat{y} = f_{\theta^\star}(x)$
Λ_ϕ
$\theta^\star$
$\mathcal{L}_{task}$
$\frac{\partial \theta^\star}{\partial \phi}$
IFT
$\nabla_{\theta^\star} \mathcal{L}_{task}$
Backprop

Figure 1. TaskMet learns a metric for parameterized prediction loss using the gradient signal from a downstream task loss.

Running examples of predictions and tasks. We focus on the following examples in this paper: 1) the portfolio setting from Wilder et al. (2019), which predicts the expected returns from stocks for a financial portfolio. Here, the $\mathcal{L}_{\text{pred}}$ is the MSE and $\mathcal{L}_{\text{task}}$ is from the regret of running a portfolio optimization problem on the output; 2) the allocation setting from Wilder et al. (2019), which predicts the value of items that are being allocated, *e.g.* click-through-rates for recommender systems. Here, $\mathcal{L}_{\text{pred}}$ is the MSE and $\mathcal{L}_{\text{task}}$ measures the result of allocating the highest-value items. 3) the model-based reinforcement learning setting of learning the system dynamics from Nikishin et al. (2022). Here, $\mathcal{L}_{\text{pred}}$ is the MSE and the $\mathcal{L}_{\text{task}}$ measures how well the dynamics performs for downstream value predictions.

Limitations of task-based learning. Task-based model learning comes with the promise of being able to discover task-relevant features and data-samples on its own without the need of explicit inductive biases. The current trend for end-to-end model learning uses task loss along with the prediction loss to train the prediction models. Though easy to use, these methods may be limited by 1) the prediction overfitting to the particular task, rendering it unable to generalize; 2) the need of tuning weights to combine the task and prediction losses as in Eq. (1).

Our contributions. We present TaskMet — a task-driven end-to-end metric learning framework which is used for training prediction model, see Fig. 1. This enables more interpretable learning of the model using the metric compared to learning with combination of task loss and prediction loss. The learned metric can uncover underlying properties of the task that are useful for training the model, *e.g.* as in Figs. 2 and 4. Section 4 shows the empirical success of metric learning on decision focused model learning setting and MBRL with policy learning as the downstream task.

[1] Meta AI. Correspondence to: Dishank Bansal <dishankbansal09@gmail.com>.

Published at the Differentiable Almost Everything Workshop of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. July 2023. Copyright 2023 by the author(s).

2. Preliminaries and Background

We consider a prediction model $\hat{y} := f_\theta(x) : \mathcal{X} \to \mathcal{Y}$ that maps from $\mathcal{X}$ to $\mathcal{Y}$ and is parameterized by θ. The dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^N$ consists of N input-output $(x, y) \in \mathcal{X} \times \mathcal{Y}$ pairs. The prediction model has an associated prediction loss, $\mathcal{L}_{\text{pred}}$ and is used in conjunction with some downstream task that provides a task loss $\mathcal{L}_{\text{task}}$ characterizing how well the model performs on the task. The most relevant related work to ours includes the approaches of Bengio (1997); Donti et al. (2017); Farahmand et al. (2017); Kang et al. (2019); Wilder et al. (2019); Nikishin et al. (2022); Shah et al. (2022); Voelcker et al. (2022); Nikishin et al. (2022), which learn the optimal prediction model parameter θ to minimize the task loss $\mathcal{L}_{\text{task}}$:

$$\theta^\star := \arg\min_\theta \mathcal{L}_{\text{task}}(\theta) + \alpha \mathcal{L}_{\text{pred}}(\theta), \tag{1}$$

where α is a regularization parameter to weigh the prediction loss which is MSE error (Eq. (2)) in general. Alternatives to Eq. (1) include 1) *Smart, "Predict, then Optimize"* (SPO) methods (Elmachtoub & Grigas, 2022; El Balghiti et al., 2019; Mandi et al., 2020; Liu et al., 2023), which consider surrogates for when the derivative is undefined or uninformative, or 2) changing the prediction space from the original domain $\mathcal{X}$ into a latent domain with task information, *e.g.* task-specific latent dynamics for RL (Hafner et al., 2019b;a; Hansen et al., 2022). Sadana et al. (2023) provide a broader survey of the field of decision making under uncertainty.

3. TaskMet: Task-Driven Metric Learning

3.1. Metricized loss for training prediction models

A supervised prediction model f_θ parameterized by θ is often trained using the mean squared error (MSE)

$$\theta^\star := \arg\min_\theta \mathbb{E}_{x,y\sim\mathcal{D}}[(f_\theta(x) - y)^2]. \tag{2}$$

Equation (2) is equivalent to maximizing the data likelihood and is often optimized using a stochastic first-order method, *e.g.* as in Bottou et al. (2018). MSE error is indicative of the prediction performance of the model. We propose to use a parameterized variant of the MSE to turn it into a metricized loss:

$$\mathcal{L}_{\text{pred}}(\theta, \phi) := \mathbb{E}_{x,y\sim\mathcal{D}}[\|f_\theta(x) - y\|^2_{\Lambda_\phi(x)}], \tag{3}$$

where $\|x\|_M := (x^\top M x)^{1/2}$ is a Mahalanobis norm with positive semi-definite matrix M, *e.g.* as in Ghojogh et al. (2022). We consider a parametric metric Λ_ϕ with parameters ϕ and make it conditional on the feature x so it can learn the importance of the regression space from each part of the feature space. Learning θ using $\mathcal{L}_{\text{pred}}(\theta, \phi)$ makes the model parameters θ an implicit function of ϕ denoted as $\theta^\star(\phi)$, with the condition $\nabla_\theta \mathcal{L}_{\text{pred}}(\theta, \phi)|_{\theta=\theta^\star} = 0$.

Algorithm 1 TaskMet

```
Models: predictor $f_\theta$ and metric $\Lambda_\phi$
while unconverged do
  for $i$ in $1 \ldots K$ do
    $\theta \leftarrow$ update($\theta, \nabla_\theta \mathcal{L}_{\text{pred}}(\theta, \phi)$) // Eq. (3): fit $f$ to the current metric loss
  end for
  $\phi \leftarrow$ update($\phi, \nabla_\phi \mathcal{L}_{\text{task}}$) // Eq. (6): update $\Lambda$ with the task loss
end while
return optimal parameters $\theta, \phi$
```

Learning model parameters with a given metric can condition the learning process in the following ways: 1) *Relative importance of dimensions*: the metric allows for down- or up-weighting different dimensions of the prediction space by changing the diagonal entries of the metric. 2) *Correlation in the prediction space*: the quadratic nature of the loss with the metric allows the model to be aware of correlations between dimensions in the prediction space. 3) *Relative importance of samples*: heteroscedastic metrics $\Lambda(x)$ enable different samples to be weighted differently for the final expected cost over the dataset. Hence, learning model parameters with a metricized loss can be seen as conditioning the learning problem. The ability to learn the metric end-to-end enables the task to condition the learning of the model in any or all of the three ways described above. This approach offers an interpretable method for the task to guide the model learning, in contrast to relying solely on task gradients for learning model parameters, which may or may not align effectively with the given prediction task.

3.2. End-to-end metric learning for model learning

The key idea of the method is to learn a metric end-to-end with a given task, which is then used to train the prediction model as shown in Eq. (3). The learning problem of the metric and model parameters are

$$\phi^\star := \arg\min_\phi \mathcal{L}_{\text{task}}(\theta^\star(\phi)), \tag{4}$$

$$\text{s.t. } \theta^\star(\phi) = \arg\min_\theta \mathcal{L}_{\text{pred}}(\theta, \phi) \tag{5}$$

where ϕ and θ are (respectively) the metric and model parameters, $\mathcal{L}_{\text{pred}}$ is the metricized prediction loss (Eq. (3)) to train the prediction model, and $\mathcal{L}_{\text{task}}$ is the task loss defined by the task at hand (which could be another optimization problem, *e.g.* Eq. (8), or another learning task, *e.g.* Eq. (10).

Gradient-based learning. We learn the optimal metric $\Lambda_{\phi^\star}$ with the gradient of the task loss, *i.e.* $\nabla_\phi \mathcal{L}_{\text{task}}(\theta^\star(\phi))$. Using the chain rule and assuming we have the optimal $\theta^\star(\phi)$ for some metric parameterization ϕ, this derivative is

$$\nabla_\phi \mathcal{L}_{\text{task}}(\theta^\star(\phi)) = \nabla_\theta \mathcal{L}_{\text{task}}(\theta)\big|_{\theta=\theta^\star(\phi)} \cdot \frac{\partial \theta^\star(\phi)}{\partial \phi} \tag{6}$$

To calculate the term $\nabla_\phi \mathcal{L}_{\text{task}}(\theta^\star(\phi))$, we need to compute two gradient terms: $\nabla_\theta \mathcal{L}_{\text{task}}(\theta)\big|_{\theta=\theta^\star(\phi)}$ and $\partial \theta^\star(\phi)/\partial \phi$.

The former can be estimated in standard way since $\mathcal{L}_{\text{task}}(\theta)$ is an explicit function of θ. However, the latter cannot be directly calculated because $\theta^\star$ is a function of optimization problem which is multiple iterations of gradient descent, as shown in Eq. (5). Backpropping through multiple iterations of gradient descent can be computationally expensive, so we use the implicit function theorem on the first-order optimality condition of Eq. (5), *i.e.* $\frac{\partial \mathcal{L}_{\text{pred}}(\theta,\phi)}{\partial \theta} = 0$; see Appendix A for more details. $\nabla_\phi \mathcal{L}_{\text{task}}(\theta^\star(\phi))$ can be computed with

$$-\nabla_\theta \mathcal{L}_{\text{task}}(\theta) \cdot \left(\frac{\partial^2 \mathcal{L}_{\text{pred}}(\theta,\phi)}{\partial \theta^2}\right)^{-1} \cdot \frac{\partial^2 \mathcal{L}_{\text{pred}}(\theta,\phi)}{\partial \phi \partial \theta}\Bigg|_{\theta=\theta^\star(\phi)} \tag{7}$$

We follow Blondel et al. (2022) and compute the implicit derivative by using conjugate on the normal equations.

Discussion. Other related work on metric learning such as Hastie & Tibshirani (1995); Yang & Jin (2006); Weinberger & Tesauro (2007); Kulis et al. (2013); Hauberg et al. (2012); Kaya & Bilge (2019) often learns a non-Euclidean metric or distance that captures the geometry of the data and then solves a prediction task such as regression, clustering, or classification in that geometry. In contrast to these, in the task-based model learning, we propose that the downstream task (instead of the data alone) gives the relevant metric for the prediction, and that it is possible to use end-to-end learning as in Eq. (4) to obtain the task-based metric.

4. Experiments

We evaluate our method on standard decision-focused model learning and model-based reinforcement learning settings.

4.1. Decision Oriented Model Learning

Setup. We use three standard resource allocation tasks for comparing task-based learning methods (Shah et al., 2022; Wilder et al., 2019; Donti et al., 2017; Futoma et al., 2020). In this setting, resource prediction based on some input features constitutes prediction model, resource allocation constitutes downstream task which is characterized by $\mathcal{L}_{\text{task}}$ The prediction model's output is used in a downstream resource optimization. The settings are implemented exactly as in Shah et al. (2022) and have task losses defined by

$$\mathcal{L}_{\text{task}}(\theta) := \mathbb{E}_{x,y\sim\mathcal{D}}[g(z^\star(\hat{y}), y)] \tag{8}$$

where $z^\star(\hat{y}) := \arg\min_z g(z,\hat{y})$ and $g(z,y')$ is some combinatorial optimization objective over variable z parameterized by y'. The task loss $\mathcal{L}_{\text{task}}$ is the expected value of objective function with decision variable $z^\star(\hat{y})$ induced by the prediction model $\hat{y} = f_\theta(x)$ under the ground truth parameters y. We use corresponding surrogate losses to replicate the $z^\star(\hat{y})$ optimization problem as in Shah et al. (2022); Wilder et al. (2019); Xie et al. (2020) and differentiate through the surrogate using `cvxpylayers` (Agrawal et al., 2019). Appendix B.2 provides the exact formulations of $g(z,y)$ and description for the optimization problems. We use Eq. (4) and Eq. (5) to find optimal parameters.

Table 1. Comparison of the normalized test decision quality (0=random, 1=oracle) on the decision oriented learning problems. α is prediction loss weight in Eq. (1)

		Problems		
Method	α	Cubic	Budget	Portfolio
MSE		-0.96 ± 0.02	0.54 ± 0.17	0.33 ± 0.03
DFL	0	0.61 ± 0.74	0.91 ± 0.06	0.25 ± 0.02
DFL	10	0.62 ± 0.74	0.81 ± 0.11	0.34 ± 0.03
LODL	0	0.96 ± 0.005	0.84 ± 0.105	0.17 ± 0.05
LODL	10	-0.95 ± 0.005	0.58 ± 0.14	0.30 ± 0.03
TaskMet		0.96 ± 0.005	0.83 ± 0.12	0.33 ± 0.03

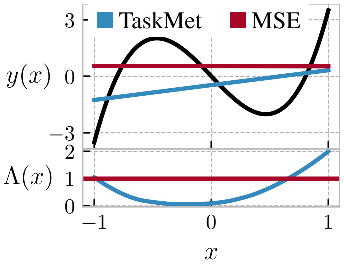

Figure 2. (Cubic problem) TaskMet learns a metric that prioritizes points that are the most important the downstream task. The euclidean metric (MSE) puts equal weight on all points and leads to a bad model with respect to the downstream task.

These settings evaluates the ability of TaskMet to capture the correlation between model predictions and differentiate between different data-points in accordance to their importance for the optimization problem. Hence, we consider a heteroscedastic metric i.e., $\Lambda_\phi(x)$.

Baselines. We compare our method with standard baselines such as MSE and DFL (Donti et al., 2017; Shah et al., 2022) along with more sophisticated method such as LODL (Shah et al., 2022). The description of baseline methods can be found in the Appendix B.2. We run our own experiments for LODL (Shah et al., 2022) using their public code.

Results. Table 1 presents a summary of the performance of different methods on all the tasks. Each problem poses unique challenges for the methods. The *cubic* setting suffers from severe approximation errors, hence the learning method needs to prioritize the accuracy of higher utility points over the accuracy of lower utility points. The MSE method performs the worst as it lacks task information and only care about prediction error. DFL with $\alpha = 0$ performs better than MSE, but it can get trapped in local optima, leading to higher variance in the problem (Shah et al., 2022). LODL ($\alpha = 0$) performs among the highest in this problem since it uses metricized loss as well learned for each point. TaskMet performs as good as LODL as it can capture the relative importance of higher utility points versus lower utility points using the learned metric, resulting in more accurate predictions for those points (see Fig. 2). In *budget allocation*, DFL (with $\alpha = 0$) performs the best, since it is solely optimizer over $\mathcal{L}_{\text{task}}$, but on the other hand it has 10 orders of larger prediction error as shown in Table 2 indicating that the model is overfit to the task, LODL ($\alpha = 0$) suffers from the same problem. TaskMet has the 2nd best Decision Quality without overfitting on the task i.e., low prediction error. In *Portfolio Optimization*, the decision quality correlates highly with the model accuracy/prediction error as in this setting the optimization problem mostly depends upon the accurate prediction of the stocks. This is the reason that MSE, DFL ($\alpha = 10$) performs the best, but DFL ($\alpha = 0$) performs the worst, since it has solely being trained on $\mathcal{L}_{\text{task}}$ without any $\mathcal{L}_{\text{pred}}$. As shown in Table 1 and Table 2, TaskMet is the only method that consistently performs well across all the problem setting, this is due to the ability of metric to infer problem specific features without manual tuning unlike other methods.

4.2. Model Based Reinforcement Learning

Setup. Given the current state s_t and control a_t at a timestep t of a discrete-time MDP, the *dynamics model* predicts the next state transition, *i.e.* $\hat{s}_{t+1} := f_\theta(s_t, a_t)$. The prediction loss is commonly the squared error loss $\mathbb{E}_{s_t,a_t,s_{t+1}} \|s_{t+1} - f_\theta(s_t, a_t)\|_2^2$, and the downstream task is to find the optimal Q-value/policy. Nikishin et al. (2022) introduces idea of *optimal model design* (OMD) to learn the dynamics model end-to-end with the policy objective via implicit differentiation. Let $Q_\omega(s, a)$ be the action-conditional value function parameterized by ω. The Q network is trained to minimize the Bellman error induced by the model f_θ:

$$\mathcal{L}_Q(\omega, \theta) := \mathbb{E}_{s,a}[Q_w(s, a) - \mathrm{B}^\theta Q_{\bar{w}}(s, a)]^2, \tag{9}$$

where $\bar{\omega}$ is moving average of ω and B^θ is the model-induced Bellman operator $\mathrm{B}^\theta Q_{\bar{w}}(s, a) := r_\theta(s, a) + \gamma\, \mathbb{E}_{p_\theta(s,a,s')}[\log \sum_{a'} \exp Q(s', a')]$ Q-network optimality defines ω as implicit function of model parameters θ as $\omega^\star(\theta) = \arg\min_\omega \mathcal{L}_Q(\omega, \theta) \implies \frac{\partial \mathcal{L}_Q(\omega,\theta)}{\partial \omega} = 0$. Now we have task loss which is optimized to find optimal Q-values:

$$\mathcal{L}_{\text{task}}(\omega^\star(\theta)) := \mathbb{E}_{s,a}[Q_{\omega^\star(\theta)}(s, a) - \mathrm{B}Q_{\bar{\omega}}(s, a)]^2 \tag{10}$$

where the Bellman operator induced by ground-truth trajectory and reward is $\mathrm{B}Q_{\bar{\omega}}(s, a) := r(s, a) + \gamma\, \mathbb{E}_{s,a,s'} \log \sum_{a'} \exp Q_{\bar{\omega}}(s', a')$.

OMD setup. OMD end-to-end optimizes the model for the task loss, *i.e.* $\theta^\star = \arg\min_\theta \mathcal{L}_{\text{task}}(\omega^\star(\theta))$.

TaskMet setup. For metric learning, we extend OMD to learn a metric using task gradients, to train the model parameters, see Fig. 5. Metric learning just adds one more level of optimization to OMD and results in the *tri-level* problem

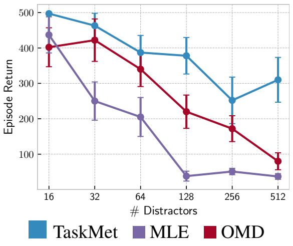

Figure 3. Results on the cartpole with distracting states (Nikishin et al., 2022).

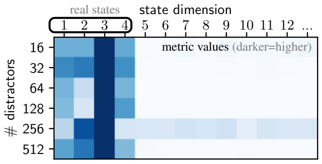

Figure 4. Our learned metric successfully distinguishes the real states from the distracting states, *i.e.* the real states take a higher metric value.

$$\begin{aligned}
\phi^\star &= \arg\min_{\phi} \;\; \mathcal{L}_{\text{task}}(\omega^\star) \\
\text{subject to} \;\; & \omega^\star(\theta^\star) = \arg\min_{\omega} \mathcal{L}_Q(\omega, \theta^\star) \\
& \theta^\star(\phi) = \arg\min_{\theta} \mathcal{L}_{\text{pred}}(\phi, \theta)
\end{aligned} \tag{11}$$

where $\mathcal{L}_{\text{task}}(\omega^\star)$ and $\mathcal{L}_Q(\omega, \theta^\star)$ are defined in Eq. (10) and Eq. (9), respectively, and $\mathcal{L}_{\text{pred}}(\phi, \theta) = \mathbb{E}_{s_t,a_t,s_{t+1}} \|s_{t+1} - f_\theta(s_t, a_t)\|^2_{\Lambda_\phi(s_t)}$ is the metricized prediction loss in Eq. (3). Appendix B.3 describes how we compute gradients.

Results. We replicated experiments from Nikishin et al. (2022) on the Cartpole environment. The first experiment involved state distractions, where the state of the agent was augmented with noisy and uninformative values to increase dimensionality. In this setting, we considered an unconditional diagonal metric of dimension n, which is the dimension of the state space, *i.e.* $\Lambda_\phi := \text{diag}(\phi)$, where $\phi \in \mathbb{R}^n$. As shown in Fig. 3, the MLE method performed the worst across different numbers of distracting states, as it allocated its capacity to learn distracting states as well. TaskMet outperformed the other methods in all scenarios. The superior performance of TaskMet with distracting states can be attributed to the metric's ability to explicitly distinguish informative states from noise states using the task loss and then train the model using the given metric, as shown in Fig. 4. The learned metric in Fig. 4 assigned the highest weight to the third dimension of the state space, which corresponds to the pole angle — the most indicative dimension for the reward. This shows that the metric can differentiate state dimensions based on their importance to the task. Appendix B.3 also shows experiments with a limited capacity.

5. Conclusions

This paper addresses the challenge of combining task and prediction losses in task-based model learning. This paper introduces the concept of task-driven metric learning, which integrates the task loss into a parameterized prediction loss. This approach enables end-to-end learning of metrics to train prediction models, allowing the models to focus on task-relevant features and dimensions in the prediction space.

References

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019. Cited on page 3.

Amos, B., Jimenez, I., Sacks, J., Boots, B., and Kolter, J. Z. Differentiable mpc for end-to-end planning and control. *Advances in neural information processing systems*, 31, 2018. Cited on page 1.

Bengio, Y. Using a financial training criterion rather than a prediction criterion. *International journal of neural systems*, 8(04):433–443, 1997. Cited on pages 1 and 2.

Bhardwaj, M., Boots, B., and Mukadam, M. Differentiable gaussian process motion planning. In *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 10598–10604. IEEE, 2020. Cited on page 1.

Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-López, F., Pedregosa, F., and Vert, J.-P. Efficient and modular implicit differentiation. *Advances in neural information processing systems*, 35:5230–5242, 2022. Cited on page 3.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018. Cited on page 2.

Dini, U. *Analisi infinitesimale*. Lithografia Gorani, 1878. Cited on page 7.

Dontchev, A. L. and Rockafellar, R. T. *Implicit functions and solution mappings*, volume 543. Springer, 2009. Cited on page 7.

Donti, P., Amos, B., and Kolter, J. Z. Task-based end-to-end model learning in stochastic optimization. *Advances in neural information processing systems*, 30, 2017. Cited on pages 1, 2, and 3.

El Balghiti, O., Elmachtoub, A. N., Grigas, P., and Tewari, A. Generalization bounds in the predict-then-optimize framework. *Advances in neural information processing systems*, 32, 2019. Cited on pages 1 and 2.

Elmachtoub, A. N. and Grigas, P. Smart "predict, then optimize". *Management Science*, 68(1):9–26, 2022. Cited on pages 1 and 2.

Farahmand, A.-m. Iterative value-aware model learning. *Advances in Neural Information Processing Systems*, 31, 2018. Cited on page 1.

Farahmand, A.-m., Barreto, A., and Nikovski, D. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pp. 1486–1494. PMLR, 2017. Cited on pages 1 and 2.

Futoma, J., Hughes, M. C., and Doshi-Velez, F. Popcorn: Partially observed prediction constrained reinforcement learning. *arXiv preprint arXiv:2001.04032*, 2020. Cited on page 3.

Ghojogh, B., Ghodsi, A., Karray, F., and Crowley, M. Spectral, probabilistic, and deep metric learning: Tutorial and survey. *arXiv preprint arXiv:2201.09267*, 2022. Cited on page 2.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a. Cited on page 2.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b. Cited on page 2.

Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022. Cited on page 2.

Hastie, T. and Tibshirani, R. Discriminant adaptive nearest neighbor classification and regression. *Advances in neural information processing systems*, 8, 1995. Cited on page 3.

Hauberg, S., Freifeld, O., and Black, M. A geometric take on metric learning. *Advances in Neural Information Processing Systems*, 25, 2012. Cited on page 3.

Kang, B., Liu, Z., Wang, X., Yu, F., Feng, J., and Darrell, T. Few-shot object detection via feature reweighting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8420–8429, 2019. Cited on pages 1 and 2.

Kaya, M. and Bilge, H. Ş. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019. Cited on page 3.

Kulis, B. et al. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013. Cited on page 3.

Liu, M., Grigas, P., Liu, H., and Shen, Z.-J. M. Active learning in the predict-then-optimize framework: A margin-based approach. *arXiv preprint arXiv:2305.06584*, 2023. Cited on pages 1 and 2.

Mandi, J., Stuckey, P. J., Guns, T., et al. Smart predict-and-optimize for hard combinatorial optimization problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 1603–1610, 2020. Cited on pages 1 and 2.

Markowitz, H. M. and Todd, G. P. *Mean-variance analysis in portfolio choice and capital markets*, volume 66. John Wiley & Sons, 2000. Cited on page 7.

Michaud, R. O. The markowitz optimization enigma: Is 'optimized' optimal? *Financial analysts journal*, 45(1): 31–42, 1989. Cited on page 7.

Nikishin, E., Abachi, R., Agarwal, R., and Bacon, P.-L. Control-oriented model-based reinforcement learning with implicit differentiation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7886–7894, 2022. Cited on pages 1, 2, 4, 8, 9, and 10.

Sadana, U., Chenreddy, A., Delage, E., Forel, A., Frejinger, E., and Vidal, T. A survey of contextual optimization methods for decision making under uncertainty, 2023. Cited on page 2.

Shah, S., Wang, K., Wilder, B., Perrault, A., and Tambe, M. Decision-focused learning without decision-making: Learning locally optimized decision losses. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=eN2lQxjWL05. Cited on pages 1, 2, 3, 7, 8, and 9.

Voelcker, C., Liao, V., Garg, A., and Farahmand, A.-m. Value gradient weighted model-based reinforcement learning. *arXiv preprint arXiv:2204.01464*, 2022. Cited on pages 1 and 2.

Weinberger, K. Q. and Tesauro, G. Metric learning for kernel regression. In *Artificial intelligence and statistics*, pp. 612–619. PMLR, 2007. Cited on page 3.

Wilder, B., Dilkina, B., and Tambe, M. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1658–1665, 2019. Cited on pages 1, 2, and 3.

Xie, Y., Dai, H., Chen, M., Dai, B., Zhao, T., Zha, H., Wei, W., and Pfister, T. Differentiable top-k with optimal transport. *Advances in Neural Information Processing Systems*, 33:20520–20531, 2020. Cited on page 3.

Yang, L. and Jin, R. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2(2):4, 2006. Cited on page 3.

A. The implicit function theorem

We used the implicit function theorem to compute the derivative of the prediction model with respect to the metric's parameters in Eq. (7). For completeness, this section briefly presents the standard implicit function theorem, *cf.* Dini (1878) and Dontchev & Rockafellar (2009, Theorem 1B.1):

Theorem 1 (Implicit Function Theorem)**.** Implicit Function Theorem*: Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be a continuous differentiable function, and let $x^\star, y^\star$ be a point satisfying $f(x^\star, y^\star) = 0$. If the Jacobian $\frac{\partial(f(x^\star,y^\star)}{\partial y} \neq 0$, then there exists an open set around $(x^\star, y^\star)$ and a unique continuously differentiable function g such that $y^\star = g(x^\star)$ and $f(x, g(x)) = 0$. Additionally, the following relation holds:*

$$\frac{\partial g(x)}{\partial x} = -\left(\frac{\partial f(x, y^\star)}{\partial y}\right) \frac{\partial f(x, y^\star)}{\partial x}|_{y^\star = g(x)} \tag{12}$$

B. Experimental Details

B.1. Metric Parameterization

We parameterize the metric using a neural network with parameters ϕ, denoted as $\Lambda_\phi := L_\phi^\top L_\phi$, where L_ϕ is an $n \times n$ matrix, where n is the dimension of the prediction space. This particular factorization constraint ensures that the metric matrix is positive semi-definite, which is crucial for it to be considered a valid metric. To initialize the neural network parameters, we set them to initialize the matrix closer to the identity matrix $\mathbb{I}$, representing the Euclidean metric. The learned metric can be conditional on the input, denoted as $\Lambda_\phi(x)$, or unconditional, represented as Λ_ϕ, depending on the problem's structure.

B.2. Decision Oriented Model Learning

Description of the resource allocation problems (Shah et al., 2022) used to benchmark our method:

- **Cubic** This setting evaluate methods under model mismatch scenario where the model being learned suffer with severe approximation error. In this task, it is important for methods to model the data well around the points that are more critical for the downstream tasks. *Prediction Model*: A linear prediction model $f_\theta(x) := \theta x$ is learned for the problem where the ground truth data is generated by cubic function i.e. $\mathcal{D} := \{x_i, y_i\}_{i=1}^N \, y_i = 10x_i^3 - 6.5x_i, x_i \in U[-1, 1]$. *Downstream task*: Choose top $B = 1$ out of $M = 50$ resources $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_M]$, $z^\star(\hat{\mathbf{y}}) := \arg\max_i \hat{\mathbf{y}}$

- **Budget Allocation**: Choose top $B = 2$ websites to advertise based on Click-through-rates (CTRs) predictions of K users on M websites. *Prediction Model*: $\hat{\mathbf{y}}_m = f_\theta(x_m)$ where x_m is given features of m^{th} website and $\hat{\mathbf{y}}_m = [\hat{y}_{m,1}, \ldots, \hat{y}_{m,K}]$ is the predicted CTRs for m^{th} website for all K users. *Downstream task*: Determine $B = 2$ websites such that expected number of users that click on the ad at least once is maximized i.e., $z^\star(\hat{\mathbf{y}}_m) = \arg\max_z \sum_{j=0}^K (1 - \prod_{i=0}^M z_i \cdot \hat{y}_{ij})$ where $z_i \in \{0, 1\}$.

- **Portfolio Optimization**: The task is to choose a distribution over M stocks in Markowitz portfolio optimization (Markowitz & Todd, 2000; Michaud, 1989) that maximize the expected return under the risk penalty. *Prediction Model*: Given the historical data x_m about a stock m, predict the future stock price $\hat{y}_m$. Combining prediction over M stocks to get $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_M]$. *Downstream Task*: Given the correlation matrix Q of the stocks, choose a distribution over stocks $\mathbf{z}^\star(\hat{\mathbf{y}}) = \arg\max_{\mathbf{z}} \mathbf{z}^\top \hat{\mathbf{y}} - \lambda \mathbf{z}^\top Q \mathbf{z}$ s. t. $\sum_{i=0}^M z_i \leq 1 \quad$ and $\quad 0 \leq z_i \leq 1, \forall i$

Description of different baselines considered for benchmarking:

- **MSE**: Prediction model trained with standard MSE loss, $\theta^\star = \arg\min_\theta \mathbb{E}_{x,y\sim\mathcal{D}}[(f_\theta(x) - y)^2]$. This method doesn't use any task information.

- **DFL**: Prediction model trained with combination of $\mathcal{L}_{\text{task}}$ and $\mathcal{L}_{\text{pred}}$ as shown in Eq. (1).

- **LODL**: Shah et al. (2022) learn parametric loss for each point in the training data to approximate the $\mathcal{L}_{\text{task}}$ around that point i.e., $LODL_{\psi_n}(\hat{y}_n) \approx \mathcal{L}_{\text{task}}(\hat{y}_n)\forall n$. They create a dataset of $\{(\hat{y}_n, \mathcal{L}_{\text{task}}(\hat{y}_n))\}$ for $\hat{y}_n$ sampled around the y_n. After this they learn the LODL loss for each point as follows - $\psi_n^\star = \arg\min_{\psi_n} \frac{1}{K} \sum_{k=1}^K (LODL_{\psi_n}(y_n^k) - \mathcal{L}_{\text{task}}(y_n^k))^2$. We chose the variant of their method which was closest to ours, where $LODL_{\psi_n}(\hat{y}) = (\hat{y} - y)^\top \psi_n (\hat{y} - y)$ where ψ_n

is a learned symmetric Positive semidefinite (PSD) matrix. LODL also uses Eq. (1) to learn the model parameters, but using $LODL_{\psi_n}(\hat{y}_n) \approx \mathcal{L}_{\text{task}}(\hat{y}_n)$

Table 2. Prediction Error of different methods

Method	Problems		
	Cubic	Budget Allocation	Portfolio Optimization
MSE	2.30 ± 0.03	$4.32e^{-4} \pm 2.35e^{-4}$	$4.03e^{-4} \pm 0.24e^{-4}$
DFL ($\alpha = 0$)	2.89 ± 0.32	$7.17e^{-3} \pm 5.83e^{-3}$	0.826 ± 0.081
DFL ($\alpha = 10$)	2.41 ± 0.05	$8.09e^{-4} \pm 12.07e^{-4}$	$5.18e^{-4} \pm 0.46e^{-4}$
LODL-Quadratic ($\alpha = 0$)	2.88 ± 0.030	$3.59e{-}3 \pm 1.29e^{-3}$	$5.56e^{-3} \pm 9.95e^{-4}$
LODL-Quadratic($\alpha = 10$)	2.29 ± 0.19	$5.05e^{-4} \pm 1.88e^{-4}$	$4.31e^{-4} \pm 0.31e^{-4}$
TaskMet	2.89 ± 0.03	$9.74e^{-4} \pm 13.79e^{-4}$	$4.69e^{-4} \pm 0.56e^{-4}$

B.3. Model-based reinforcement learning

Following is the derivation of final gradient to learn ϕ from Eq. (11). Using the implicit function theorem and using it on Eq. (11), we get the following

$$\nabla_\phi \mathcal{L}_{\text{task}} = \nabla_\omega \mathcal{L}_{\text{task}}(\omega^\star) \cdot \frac{\partial \omega^\star}{\partial \theta^\star} \cdot \frac{\partial \theta^\star}{\partial \phi} \tag{13}$$

$$= \nabla_\omega \mathcal{L}_{\text{task}}(\omega^\star) \cdot \left(\frac{\partial \mathcal{L}(\omega, \theta^\star)}{\partial^2 \omega} \right)^{-1} \cdot \left. \frac{\partial \mathcal{L}(\omega, \theta^\star)}{\partial \theta \partial \omega} \right|_{\omega = \omega^\star(\theta^\star)} \cdot \left(\frac{\partial \mathcal{L}_{\text{pred}}(\theta, \phi)}{\partial^2 \theta} \right)^{-1} \cdot \left. \frac{\partial \mathcal{L}_{\text{pred}}(\theta, \phi)}{\partial \phi \partial \theta} \right|_{\theta = \theta^\star(\phi)} \tag{14}$$

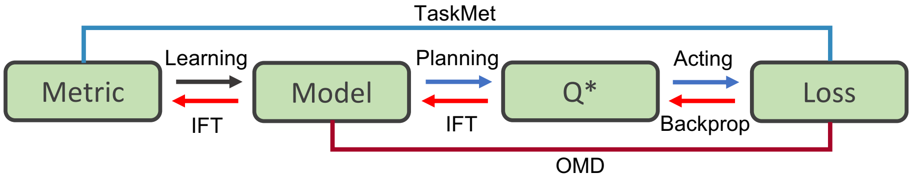

Figure 5. OMD (Nikishin et al., 2022) uses planning task loss to learn the model parameters using implicit gradients. TaskMet add one more optimization step over OMD and instead of learning the model parameters using task loss, we learn the metric which then is used to learn model parameters.

We also consider a setting with reduced model capacity, where the network is under-parametrized, forcing the model to prioritize how it allocates its capacity. In this scenario, we employ a full conditional metric, denoted as $\Lambda_\phi = \Lambda_\phi(x)$, which enables the metric to weigh dimensions and state-action pairs differently. We conducted the experiment using a model size of 3 hidden units in the layer. As depicted in Fig. 6, TaskMet achieves a better return on evaluation compared to MLE and OMD. Additionally, it is evident that TaskMet achieves a lower MSE on the model predictions compared to OMD, indicating that learning with the metric also contributes to a better dynamics model.

C. Implementation Details

C.1. Source code

Upon request, we will provide an anonymized version of our code in the rebuttal.

C.2. Decision Oriented Model Learning

We replicated our experiments using the codebase provided by Shah et al. (2022), which can be found at github. To ensure consistency, we used the same hyperparameters as mentioned in the code or article for the baselines. Our metric learning

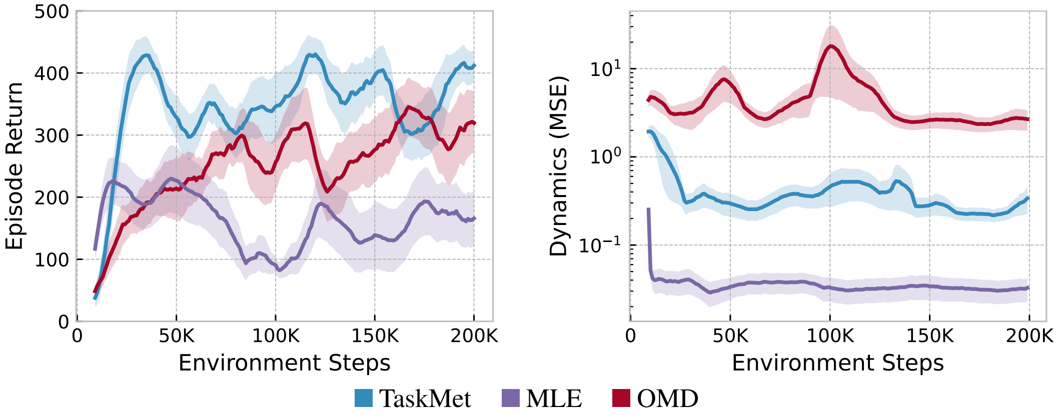

Figure 6. Results on cartpole with a reduced model capacity from Nikishin et al. (2022).

pipeline was added on top of their code, and thus we focused on tuning hyperparameters related to metric learning. The metric is parameterized as $\Lambda_\phi(x) = L_\phi(x)L_\phi^\top(x) + \epsilon_\phi \mathbb{I}_{n\times n}$, where ϵ_ϕ is a learnable parameter that explicitly controls the amount of Euclidean metric in the predicted metric. This helps ensure the stability of metric learning. We initialize the parameters in such a way that the predicted metric is close to the Euclidean metric. For each outer loop of metric update, we perform K inner updates to train the predictor. Following the methodology of Shah et al. (2022), we conducted 50 runs with different seeds for each of the experiments, where each method was evaluated on 10 different datasets, with 5 different seeds used for each dataset.

Table 3. Hyper-parameters for Decision Oriented Learning Experiments

Hyper-Parameter	Values
Λ_ϕ learning rate	10^{-3}
Λ_ϕ hidden layer sizes	`[200]`
Warmup steps	500
Inner Iterations (K)	100
Implicit derivative batchsize	10
Implicit derivative solver	Conjugate gradient on the normal equations (5 iterations)

C.3. Model-Based Reinforcement Learning

We consider the work of Nikishin et al. (2022) as the baseline for replicating the experiments, and we build upon their source code. Our metric learning is just one additional step to their method. We adopt exact same hyperparameters as their for dynamics learning and Action-Value function learning. We focus on exploring and tuning the hyper-parameters specific to the metric learning component of the method.

Table 4. Hyper-parameters for the CartPole experiments

Hyper-Parameter	Values
Λ_ϕ learning rate	10^{-3}
Λ_ϕ hidden layer sizes	`[32, 32]`
Warmup steps	5000
Inner iterations (K)	1
Implicit derivative batchsize	256
Implicit derivative solver	Conjugate gradient on the normal equations (10 iterations)

For the state distractor experiments, we parameterize the metric as an unconditional diagonal matrix, denoted as $\Lambda_\phi = \text{diag}(\phi)$ where $\phi \in \mathbb{R}^n$ and n is the dimension of the state space. In addition, we also consider a hyper-parameter of *metric parameterization*, for which we either take normalize or unnormalized metric. When we refer to normalizing the metric, we mean constraining the norm of the ϕ vector to be equal to the L2 norm of an euclidean metric which is used by MSE method. This constraint the family of learnable metrics. To achieve this, we set $\phi := \frac{\phi}{\|\phi\|_2}\sqrt{n}$, ensuring $\|\phi\|_2 = \|\mathbb{I}_{n\times n}\|_2 = \sqrt{n}$. We also used L1 regularization on the metric output, to induce sparsity in the metric. We sweep over three values of the regularization coefficient - $[0.0, 0.001, 0.1]$. We ran a sweep over the 6 combinations of hyperparameters - [unnormalized, normalized] × $[0.0, 0.001, 0.1]$ and choose the best hyper-parameter combination for each of the experiment. All the number reported in the experiments are calculated over 10 random seeds.

Our metric learning approach uses two implicit gradient steps. Firstly, we take the implicit derivative through action-value network parameters, approximating the inverse hessian to the identity, similar to Nikishin et al. (2022). Secondly, for the step through dynamics network parameters, we calculate the exact implicit derivative.

D. Training time analysis

We also measure the training time of our method compared to MLE-based training, as using implicit derivatives may introduce additional computational overhead. It is important to note that the evaluation time is not affected by the use of metric learning, as the metric is only employed during the training phase. Consequently, the evaluation runtime of the agent remains the same for both methods.

Table 5. Time (sec) for one update step of the agent

Method	Time (s)
MLE	0.0240
TaskMet	0.0243

We measure the time for single update step of agent. The code is written in JAX and evaluated on the computer with GPU — Quadro GV100 and CPU — Intel Xeon Gold 6230 @ 2.10GHz. As shown in Table 5, learning a metric using implicit derivative to train the predictor takes negligible extra time compared to MLE method.