Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization

Bryan Wilder, Bistra Dilkina, Milind Tambe
Center for Artificial Intelligence in Society, University of Southern California
{bwilder, dilkina, tambe}@usc.edu

Abstract

Creating impact in real-world settings requires artificial intelligence techniques to span the full pipeline from data, to predictive models, to decisions. These components are typically approached separately: a machine learning model is first trained via a measure of predictive accuracy, and then its predictions are used as input into an optimization algorithm which produces a decision. However, the loss function used to train the model may easily be misaligned with the end goal, which is to make the best decisions possible. Hand-tuning the loss function to align with optimization is a difficult and error-prone process (which is often skipped entirely).

We focus on combinatorial optimization problems and introduce a general framework for decision-focused learning, where the machine learning model is directly trained in conjunction with the optimization algorithm to produce high-quality decisions. Technically, our contribution is a means of integrating common classes of discrete optimization problems into deep learning or other predictive models, which are typically trained via gradient descent. The main idea is to use a continuous relaxation of the discrete problem to propagate gradients through the optimization procedure. We instantiate this framework for two broad classes of combinatorial problems: linear programs and submodular maximization. Experimental results across a variety of domains show that decision-focused learning often leads to improved optimization performance compared to traditional methods. We find that standard measures of accuracy are not a reliable proxy for a predictive model's utility in optimization, and our method's ability to specify the true goal as the model's training objective yields substantial dividends across a range of decision problems.

Introduction

The goal in many real-world applications of artificial intelligence is to create a pipeline from data, to predictive models, to decisions. Together, these steps enable a form of evidence-based decision making which has transformative potential across domains such as healthcare, scientific discovery, transportation, and more (Horvitz and Mitchell 2010; Horvitz 2010). This pipeline requires two technical components: machine learning models and optimization algorithms. Machine learning models use the data to predict unknown quantities; optimization algorithms use these predictions to arrive at a decision which maximizes some objective. Our concern here is combinatorial optimization, which is ubiquitous in real-world applications of artificial intelligence, ranging from matching applicants to public housing to selecting a subset of movies to recommend. We focus on common classes of combinatorial problems which have well-structured continuous relaxations, e.g., linear programs and submodular maximization. A vast literature has been devoted to combinatorial optimization (Korte et al. 2012). Importantly though, optimization is often insufficient without the broader pipeline because the objective function is unknown and must predicted via machine learning.

While machine learning has witnessed incredible growth in recent years, the two pieces of the pipeline are treated entirely separately by typical training approaches. That is, a system designer will first train a predictive model using some standard measure of accuracy, e.g., mean squared error for a regression problem. Then, the model's predictions are given as input to the optimization algorithm to produce a decision. Such *two-stage* approaches are extremely common across many domains (Wang et al. 2006; Fang et al. 2016; Mukhopadhyay et al. 2017; Xue et al. 2016). This process is justified when the predictive model is perfect, or near-so, since completely accurate predictions also produce the best decisions. However, in complex learning tasks, all models will make errors and the training process implicitly trades off where these errors will occur. When prediction and optimization are separate, this tradeoff is divorced from the goal of the broader pipeline: to make the best decision possible.

We propose a *decision-focused learning* framework which melds the data-decisions pipeline by integrating prediction and optimization into a single end-to-end system. That is, the predictive model is trained using the quality of the decisions which it induces via the optimization algorithm. Similar ideas have recently been explored in the context of convex optimization (Donti, Amos, and Kolter 2017), but to our knowledge ours is the first attempt to train machine learning systems for performance on *combinatorial* decision-making problems. Combinatorial settings raise new technical challenges because the optimization problem is discrete. However, machine learning systems (e.g., deep neural networks) are often trained via gradient descent.

Our first contribution is a general framework for training

machine learning models via their performance on combinatorial problems. The starting point is to relax the combinatorial problem to a continuous one. Then, we analytically differentiate the optimal solution to the continuous problem as a function of the model's predictions. This allows us to train using a continuous proxy for the discrete problem. At test time, we round the continuous solution to a discrete point.

Our second contribution is to instantiate this framework for two broad classes of combinatorial problems: linear programs and submodular maximization problems. Linear programming encapsulates a number of classical problems such as shortest path, maximum flow, and bipartite matching. Submodular maximization, which reflects the intuitive phenomena of diminishing returns, is also ubiquitous; applications range from social networks (Kempe, Kleinberg, and Tardos 2003) to recommendation systems (Viappiani and Boutilier 2010). In each case, we resolve a set of technical challenges to produce well-structured relaxations which can be efficiently differentiated through.

Finally, we give an extensive empirical investigation, comparing decision-focused and traditional methods on a series of domains. Decision-focused methods often improve performance for the pipeline as a whole (i.e., decision quality) despite worse predictive accuracy according to standard measures. Intuitively, the predictive models trained via our approach focus specifically on qualities which are important for making good decisions. By contrast, more generic methods produce predictions where error is distributed in ways which are not aligned with the underlying task.

Problem description

We consider combinatorial optimization problems of the form $\max_{x \in \mathcal{X}} f(x, \theta)$, where $\mathcal{X}$ is a discrete set enumerating the feasible decisions. Without loss of generality, $\mathcal{X} \subseteq \{0,1\}^n$ and the decision variable x is a binary vector. The objective f depends on a parameter $\theta \in \Theta$. If θ were known exactly, a wide range of existing techniques could be used to solve the problem. In this paper, we consider the challenging (but prevalent) case where θ is unknown and must be inferred from data. For instance, in bipartite matching, x represents whether each pair of nodes were matched and θ contains the reward for matching each pair. In many applications, these affinities are learned from historical data.

Specifically, the decision maker observes a feature vector $y \in \mathcal{Y}$ which is correlated with θ. This introduces a learning problem which must be solved prior to optimization. As in classical supervised learning, we formally model y and θ as drawn from a joint distribution P. Our algorithm will observe training instances $(y_1, \theta_1)...(y_N, \theta_N)$ drawn iid from P. At test time, we are give a feature vector y corresponding to an *unobserved* θ. Our algorithm will use y to predict a parameter value $\hat{\theta}$. Then, we will solve the optimization problem $\max_x f(x, \hat{\theta})$ to obtain a decision x^*. Our utility is the objective value that x^* obtains with respect to the *true but unknown* parameter θ, $f(x^*, \theta)$.

Let $m : \mathcal{Y} \to \Theta$ denote a model mapping observed features to parameters. Our goal is to (using the training data) find a model m which maximizes expected performance on the underlying optimization task. Define $x^*(\theta) = \arg\max_{x \in \mathcal{X}} f(x, \theta)$ to be the optimal x for a given θ. The end goal of the data-decisions pipeline is to maximize

$$\mathop{\mathbb{E}}_{y,\theta \sim P} [f(x^*(m(y)), \theta)] \tag{1}$$

The classical approach to this problem is a *two-stage* method which first learns a model using a task-agnostic loss function (e.g., mean squared error) and then uses the learned model to solve the optimization problem. The model class will have its own parameterization, which we denote by $m(y, \omega)$. For instance, the model class could consist of deep neural networks where ω denotes the weights. The two-stage approach first solves the problem $\min_\omega \mathbb{E}_{y,\theta \sim P} [\mathcal{L}(\theta, m(y, \omega))]$, where $\mathcal{L}$ is a loss function. Such a loss function measures the overall "accuracy" of the model's predictions but does not specifically consider how m will fare when used for decision making. The question we address is whether it is possible to do better by specifically training the model to perform well on the decision problem.

Previous work

There is a growing body of research at the interface of machine learning and discrete optimization (Vinyals, Fortunato, and Jaitly 2015; Bertsimas and Dunn 2017; Khalil et al. 2017b; 2017a). However, previous work largely focuses on either using discrete optimization to find an accuracy-maximizing predictive model or using machine learning to speed up optimization algorithms. Here, we pursue a deeper synthesis; to our knowledge, this work is the first to train predictive models using combinatorial optimization performance with the goal of improving decision making.

The closest work to ours in motivation is (Donti, Amos, and Kolter 2017), who study task-based convex optimization. Their aim is to optimize a convex function which depends on a learned parameter. As in their work, we use the idea of differentiating through the KKT conditions. However, their focus is entirely on continuous problems. Our discrete setting raises new technical challenges, highlighted below. Elmachtoub and Grigas (2017) also propose a means of integrating prediction and optimization; however, their method applies strictly to linear optimization and focuses on linear predictive models while our framework applies to nonlinear problems with more general models (e.g., neural networks). Finally, some work has noted that two-stage methods lead to poor optimization performance in specific domains (Beygelzimer and Langford 2009; Ford et al. 2015).

Our work is also related to recent research in structured prediction (Belanger, Yang, and McCallum 2017; Tu and Gimpel 2018; Niculae et al. 2018; Djolonga and Krause 2017). which aims to make a prediction lying in a discrete set. This is fundamentally different than our setting since their goal is to *predict* an external quantity, not to *optimize* and find the best decision possible. However, structured prediction sometimes integrates a discrete optimization problem as a module within a larger neural network. The closest such work technically to ours is (Tschiatschek, Sahin, and Krause 2018), who design a differentiable algorithm for

submodular maximization in order to predict choices made by users. Their approach is to introduce noise into the standard greedy algorithm, making the probability of outputting a given set differentiable. There are two key differences between our approaches. First, their approach does not apply to the decision-focused setting because it maximizes the likelihood of a *fixed* set but cannot optimize for finding the best set. Second, exactly computing gradients for their algorithm requires marginalizing over the $k!$ possible permutations of the items, forcing a heuristic approximation to the gradient. Our approach allows closed-form differentiation.

Some deep learning architectures differentiate through gradient descent steps, related to our approach in the submodular setting. Typically, previous approaches explicitly unroll T iterations of gradient descent in the computational graph (Domke 2012). However, this approach is usually employed for *unconstrained* problems where each iteration is a simple gradient step. By contrast, our combinatorial problems are constrained, requiring a projection step to enforce feasibility. Unrolling the projection step may be difficult, and would incur a large computational cost. We instead exploit the fact that gradient ascent converges to a local optimum and analytically differentiate via the KKT conditions.

General framework

Our goal is to integrate combinatorial optimization into the loop of gradient-based training. That is, we aim to directly train the predictive model m by running gradient steps on the objective in Equation 1, which integrates both prediction and optimization. The immediate difficulty is the dependence on $x^*(m(y, \omega))$. This term is problematic for two reasons. First, it is a discrete quantity since x^* is a decision from a binary set. This immediately renders the output nondifferentiable with respect to the model parameters ω. Second, even if x^* were continuous, it is still defined as the solution to an optimization problem, so calculating a gradient requires us to differentiate through the argmax operation.

We resolve both difficulties by considering a continuous relaxation of the combinatorial decision problem. We show that for a broad class of combinatorial problems, there are appropriate continuous relaxations such that we can analytically obtain derivatives of the continuous optimizer with respect to the model parameters. This allows us to train any differentiable predictive model via gradient descent on a continuous surrogate to Equation 1. At test time, we solve the true discrete problem by rounding the continuous point.

More specifically, we relax the discrete constraint $x \in \mathcal{X}$ to the continuous one $x \in conv(\mathcal{X})$ where $conv$ denotes the convex hull. Let $x(\theta) = \arg\max_{x \in conv(\mathcal{X})} f(x, \theta)$ denote the optimal solution to the continuous problem. To train our predictive model, we would like to compute gradients of the whole-pipeline objective given by Equation 1, replacing the discrete quantity x^* with the continuous x. We can obtain a stochastic gradient estimate by sampling a single (y, θ) from the training data. On this sample, the chain rule gives

$$\frac{df(x(\hat{\theta}), \theta)}{d\omega} = \frac{df(x(\hat{\theta}), \theta)}{dx(\hat{\theta})} \frac{dx(\hat{\theta})}{d\hat{\theta}} \frac{d\hat{\theta}}{d\omega}$$

The first term is just the gradient of the objective with respect to the decision variable x, and the last term is the gradient of the model's predictions with respect to its own internal parameterization.

The key is computing the middle term, which measures how the optimal decision changes with respect to the prediction $\hat{\theta}$. For continuous problems, the optimal continuous decision x must satisfy the KKT conditions (which are sufficient for convex problems). The KKT conditions define a system of linear equations based on the gradients of the objective and constraints around the optimal point. Is is known that by applying the implicit function theorem, we can differentiate the solution to this linear system (Gould et al. 2016; Donti, Amos, and Kolter 2017). In more detail, recall that our continuous problem is over $conv(\mathcal{X})$, the convex hull of the discrete feasible solutions. This set is a polytope, which can be represented via linear equalities as the set $\{x : Ax \leq b\}$ for some matrix A and vector b. Let (x, λ) be pair of primal and dual variables which satisfy the KKT conditions. Then differentiating the conditions yields that

$$\begin{bmatrix} \nabla_x^2 f(x, \theta) & A^T \\ diag(\lambda)A & diag(Ax - b) \end{bmatrix} \begin{bmatrix} \frac{dx}{d\theta} \\ \frac{d\lambda}{d\theta} \end{bmatrix} = \begin{bmatrix} \frac{d\nabla_x f(x,\theta)}{d\theta} \\ 0 \end{bmatrix} \quad (2)$$

By solving this system of linear equations, we can obtain the desired term $\frac{dx}{d\theta}$. However, the above approach is a general framework; our main technical contribution is to instantiate it for specific classes of combinatorial problems. Specifically, we need (1) an appropriate continuous relaxation, along with a means of solving the continuous optimization problem and (2) efficient access to the terms in Equation 2 which are needed for the backward pass (i.e., gradient computation). We provide both ingredients for two broad classes of problems: linear programming and submodular maximization. In each setting, the high-level challenge is to ensure that the continuous relaxation is differentiable, a feature not satisfied by naive alternatives. We also show how to efficiently compute terms needed for the backward pass, especially for the more intricate submodular case.

Linear programming

The first setting that we consider is combinatorial problems which can be expressed as a linear program with equality and inequality constraints in the form

$$\max \theta^T x \text{ s.t. } Ax = b,\ Gx \leq h \quad (3)$$

Example problems include shortest path, maximum flow, bipartite matching, and a range of other domains. For instance, in a shortest path problem θ contains the cost for traversing each edge, and we are interested in problems where the true costs are unknown and must be predicted. Since the LP can be regarded as a continuous problem (it just happens that the optimal solutions in these example domains are integral), we could attempt to apply Equation 2 and differentiate the solution. This approach runs into an immediate difficulty: the optimal solution to an LP may not be differentiable (or even continuous) with respect to θ. This

is because the optimal solution may "jump" to a different vertex. Formally, the left-hand side matrix in Equation 2 becomes singular since $\nabla_x^2 f(x,\theta)$ is always zero. We resolve this challenge by instead solving the regularized problem

$$\max \theta^T x - \gamma||x||_2^2 \text{ s.t. } Ax = b,\ Gx \leq h \qquad (4)$$

which introduces a penalty proportional to the squared norm of the decision vector. This transforms the LP into a strongly concave quadratic program (QP). The Hessian is given by $\nabla_x^2 f(x,\theta) = -2\gamma I$ (where I is the identity matrix), which renders the solution differentiable under mild conditions (see supplement for proof):

Theorem 1. *Let $x(\theta)$ denote the optimal solution of Problem 4. Provided that the problem is feasible and all rows of A are linearly independent, $x(\theta)$ is differentiable with respect to θ almost everywhere. If A has linearly dependent rows, removing these rows yields an equivalent problem which is differentiable almost everywhere. Wherever $x(\theta)$ is differentiable, it satisfies the conditions in Equation 2.*

Moreover, we can control the loss that regularization can cause on the original, linear problem:

Theorem 2. *Define $D = \max_{x,y \in conv(\mathcal{X})} ||x-y||^2$ as the squared diameter of the feasible set and OPT to be the optimal value for Problem 3. We have $\theta^\top x(\theta) \geq OPT - \gamma D$.*

Together, these results give us a differentiable surrogate which still enjoys an approximation guarantee relative to the integral problem. Computing the backward pass via Equation 2 is now straightforward since all the relevant terms are easily available. Since $\nabla_x \theta^\top x = \theta$, we have $\frac{d\nabla_x f(x,\theta)}{d\theta} = I$. All other terms are easily computed from the optimal primal-dual pair (x, λ) which is output by standard QP solvers. We can also leverage a recent QP solver (Amos and Kolter 2017) which maintains a factorization of the KKT matrix for a faster backward pass. At test time, we simply set $\gamma = 0$ to produce an integral decision.

Submodular maximization

We consider problems where the underlying objective to maximize a set function $f : 2^V \to R$, where V is a ground set of items. A set function is *submodular* if for any $A \subseteq B$ and any $v \in V \backslash B$, $f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$. We will restrict our consideration to submodular functions which are *monotone* ($f(A \cup \{v\}) - f(A) \geq 0\ \forall A, v$) and *normalized* $f(\emptyset) = 0$. This class of functions contains many combinatorial problems which have been considered in machine learning and artificial intelligence (e.g., influence maximization, facility location, diverse subset selection, etc.). We focus on the cardinality-constrained optimization problem $\max_{|S| \leq k} f(S)$, though our framework easily accommodates more general matroid constraints.

Continuous relaxation: We employ the canonical continuous relaxation for submodular set functions, which associates each set function f with its *multilinear extension* F (Calinescu et al. 2011). We can view a set function as defined on the domain $\{0,1\}^{|V|}$, where each element is an indicator vector which the items contained in the set. The extension F is a continuous function defined on the hypercube $[0,1]^{|V|}$. We interpret a given fraction vector $x \in [0,1]^{|V|}$ as giving the marginal probability that each item is included in the set. $F(x)$ is the expected value of $f(S)$ when each item i is included in S independently with probability x_i. In other words, $F(x) = \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{i \notin S} 1 - x_i$. While this definition sums over exponentially many terms, arbitrarily close approximations can be obtained via random sampling. Further, closed forms are available for many cases of interest (Iyer, Jegelka, and Bilmes 2014). Importantly, well-known rounding algorithms (Calinescu et al. 2011) can convert a fractional point x to a set S satisfying $\mathbb{E}[f(S)] \geq F(x)$; i.e., the rounding is lossless.

As a proxy for the discrete problem $\max_{|S| \leq k} f(S)$, we can instead solve $\max_{x \in conv(\mathcal{X})} F(x)$, where $\mathcal{X} = \{x \in \{0,1\}^{|V|} : \sum_i x_i \leq k\}$. Unfortunately, F is not in general concave. Nevertheless, many first-order algorithms still obtain a constant factor approximation. For instance, a variant of the Frank-Wolfe algorithm solves the continuous maximization problem with the optimal approximation ratio of $(1 - 1/e)$ (Calinescu et al. 2011; Bian et al. 2017).

However, non-concavity complicates the problem of differentiating through the continuous optimization problem. Any polynomial-time algorithm can only be guaranteed to output a *local* optimum, which need not be unique (compared to strongly convex problems, where there is a single global optimum). Consequently, the algorithm used to select $x(\theta)$ might return a *different* local optimum under an infinitesimal change to θ. For instance, the Frank-Wolfe algorithm (the most common algorithm for continuous submodular maximization) solves a linear optimization problem at each step. Since (as noted above), the solution to a linear problem may be discontinuous in θ, this could render the output of the optimization problem nondifferentiable.

We resolve this difficulty through a careful choice of optimization algorithm for the forward pass. Specifically, we use apply projected stochastic gradient ascent (SGA), which has recently been shown to obtain a $\frac{1}{2}$-approximation for continuous submodular maximization (Hassani, Soltanolkotabi, and Karbasi 2017). Although SGA is only guaranteed to find a local optimum, each iteration applies purely differentiable computations (a gradient step and projection onto the set $conv(\mathcal{X})$), and so the final output after T iterations will be differentiable as well. Provided that T is sufficiently large, this output will converge to a local optimum, which must satisfy the KKT conditions. Hence, we can apply our general approach to the local optimum returned by SGA. The following theorem shows that the local optima of the multilinear extension are differentiable:

Theorem 3. *Suppose that x^* is a local maximum of the multilinear extension, i.e,., $\nabla_x F(x^*, \theta) = 0$ and $\nabla_x^2 F(x^*, \theta) \succ 0$. Then, there exists a neighborhood $\mathcal{I}$ around x^* such that the maximizer of $F(\cdot, \theta)$ within $\mathcal{I} \cap conv(\mathcal{X})$ is differentiable almost everywhere as a function of θ, with $\frac{dx(\theta)}{d\theta}$ satisfying the conditions in Equation 2.*

We remark that Theorem 3 requires a local maximum, while gradient ascent may in theory find saddle points. However, recent work shows that random perturbations ensure that gradient ascent quickly escapes saddle points and finds

an approximate local optimum (Jin et al. 2017).

Efficient backward pass: We now show how the terms needed to compute gradients via Equation 2 can be efficiently obtained. In particular, we need access to the optimal dual variable λ as well as the term $\frac{d\nabla_x F(x,\theta)}{d\theta}$. These were easy to obtain in the LP setting but the submodular setting requires some additional analysis. Nevertheless, we show that both can be obtained efficiently.

Optimal dual variables: SGA only produces the optimal primal variable x, not the corresponding dual variable λ which is required to solve Equation 2 in the backward pass. We show that for cardinality-constrained problems, we can obtain the optimal dual variables analytically given a primal solution x. Let λ_i^L be the dual variable associated with the constraint $x_i \geq 0$, λ_i^U with $x_i \leq 1$ and λ^S with $\sum_i x_i \leq k$. By differentiating the Lagrangian, any optimum satisfies

$$\nabla_{x_i} f(x) - \lambda_i^L + \lambda_i^U + \lambda_i^S = 0 \quad \forall i$$

where complementary slackness requires that $\lambda_i^L = 0$ if $x_i > 0$ and $\lambda_i^U = 0$ if $x_i < 1$. Further, it is easy to see that for all i with $0 < x_i < 1$, $\nabla_{x_i} f(x)$ must be equal. Otherwise, x could not be (locally) optimal since we could increase the objective by finding a pair i, j with $\nabla_{x_i} f(x) > \nabla_{x_j} f(x)$, increasing x_i, and decreasing x_j. Let ∇_* denote the shared gradient value for fractional entries. We can solve the above equation and express the optimal dual variables as

$$\lambda^S = -\nabla_*, \quad \lambda_i^L = \lambda^S - \nabla_{x_i} f, \quad \lambda_i^U = \nabla_{x_i} f - \lambda^S$$

where the expressions for λ_i^L and λ_i^U apply only when $x_i = 0$ and $x_i = 1$ respectively (otherwise, complementary slackness requires these variables be set to 0).

Computing $\frac{\mathbf{d}}{\mathbf{d}\theta}\nabla_{\mathbf{x}}\mathbf{F}(\mathbf{x},\theta)$: We show that this term can be obtained in closed form for the case of probabilistic coverage functions, which includes many cases of practical interest (e.g. budget allocation, sensor placement, facility location, etc.). However, our framework can be applied to arbitrary submodular functions; we focus here on coverage functions just because they are particularly common in applications. A coverage function takes the following form. There a set of items U, and each $j \in U$ has a weight w_j. The algorithm can choose from a ground set V of actions. Each action a_i covers each item j independently with probability θ_{ij}. We consider the case where the probabilities θ are be unknown and must be predicted from data. For such problems, the multilinear extension has a closed form

$$F(x,\theta) = \sum_{j \in U} w_j \left(1 - \prod_{i \in V} 1 - x_{ij}\theta_{ij}\right)$$

and we can obtain the expression

$$\frac{d}{d\theta_{kj}} \nabla_{x_i} F(x,\theta) = \begin{cases} -\theta_{ij} x_k \prod_{\ell \neq i,k} 1 - x_\ell \theta_{\ell j} & \text{if } k \neq i \\ \prod_{k \neq i} 1 - x_k \theta_{kj} & \text{otherwise.} \end{cases}$$

Experiments

We conduct experiments across a variety of domains in order to compare our decision-focused learning approach with traditional two stage methods. We start out by describing the experimental setup for each domain. Then, we present results for the complete data-decisions pipeline in each domain (i.e., the final solution quality each method produces on the optimization problem). We find that decision-focused learning almost always outperforms two stage approaches. To investigate this phenomenon, we show more detailed results about what each model learns. Two stage approaches typically learn predictive models which are more accurate according to standard measures of machine learning accuracy. However, decision-focused methods learn qualities which are important for optimization performance even if this leads to lower accuracy in an overall sense.

Budget allocation: We start with a synthetic domain which allows us to illustrate how our methods differ from traditional approaches and explore when improved decision making is achievable. This example concerns budget allocation, a submodular maximization problem which models an advertiser's choice of how to divide a finite budget k between a set of channels. There is a set of customers R and the objective is $f(S) = \sum_{v \in R} 1 - \prod_{u \in S}(1 - \theta_{uv})$, where θ_{uv} is the probability that advertising on channel u will reach customer v. This is the expected number of customers reached. Variants on this problem have been the subject of a great deal of research (Alon, Gamzu, and Tennenholtz 2012; Soma et al. 2014; Miyauchi et al. 2015).

In our problem, the matrix θ is not known in advance and must be learned from data. The ground truth matrices were generated using the Yahoo webscope (Yahoo 2007) dataset which logs bids placed by advertisers on a set of phrases. In our problem, the phrases are channels and the accounts are customers. Each instance samples a random subset of 100 channels and 500 customers. For each edge (u, v) present in the dataset, we sample θ_{uv} uniformly at random in [0,0.2]. For each channel u, we generate a feature vector from that channel's row of the matrix, θ_u via complex nonlinear function. Specifically, θ_u is passed through a 5-layer neural network with random weight matrices and ReLU activations to obtain a feature vector y_u. The learning task is to reconstruct θ_u from y_u. The optimization task is to select k channels in order to maximize the number of customers reached.

Bipartite matching: This problem occurs in many domains; e.g., bipartite matching has been used to model the problem of a public housing programs matching housing resources to applicants (Benabbou et al. 2018) or platforms matching advertisers with users (Bellur and Kulkarni 2007). In each of these cases, the reward to matching any two nodes is not initially known, but is instead predicted from the features available for both parties. Bipartite matching can be formulated as a linear program, allowing us to apply our decision-focused approach. The learning problem is to use node features to predict whether each edge is present or absent (a classification problem). The optimization problem is to find a maximum matching in the predicted graph.

Our experiments use the cora dataset (Sen et al. 2008). The nodes are scientific papers and edges represent citation.

Table 1: Solution quality of each method for the full data-decisions pipeline.

	Budget allocation			Matching	Diverse recommendation		
$k =$	5	10	20	–	5	10	20
NN1-Decision	**49.18 ± 0.24**	**72.62 ± 0.33**	**98.95 ± 0.46**	2.50 ± 0.56	**15.81 ± 0.50**	**29.81 ± 0.85**	**52.43 ± 1.23**
NN2-Decision	44.35 ± 0.56	67.64 ± 0.62	93.59 ± 0.77	**6.15 ± 0.38**	13.34 ± 0.77	26.32 ± 1.38	47.79 ± 1.96
NN1-2Stage	32.13 ± 2.47	45.63 ± 3.76	61.88 ± 4.10	2.99 ± 0.76	4.08 ± 0.16	8.42 ± 0.29	19.16 ± 0.57
NN2-2Stage	9.69 ± 0.05	18.93 ± 0.10	36.16 ± 0.18	3.49 ± 0.32	11.63 ± 0.43	22.79 ± 0.66	42.37 ± 1.02
RF-2Stage	**48.81 ± 0.32**	**72.40 ± 0.43**	**98.82 ± 0.63**	3.66 ± 0.26	7.71 ± 0.18	15.73 ± 0.34	31.25 ± 0.64
Random	9.69 ± 0.04	18.92 ± 0.09	36.13 ± 0.14	2.45 ± 0.64	8.19 ± 0.19	16.15 ± 0.35	31.68 ± 0.71

Each node's feature vector indicating whether each word in a vocabulary appeared in the paper (there are 1433 such features). The overall graph has 2708 nodes. In order to construct instances for the decision problem, we partitioned the complete graph into 27 instances, each with 100 nodes, using metis (Karypis and Kumar 1998). We divided the nodes in each instance into the sides of a bipartite graph (of 50 nodes each) such that the number of edges crossing sides was maximized. The learning problem is much more challenging than before: unlike in budget allocation, the features do not contain enough information to reconstruct the citation network. However, a decision maker may still benefit from leveraging whatever signal is available.

Diverse recommendation: One application of submodular optimization is to select diverse sets of item, e.g. for recommendation systems or document summarization. Suppose that each item i is associated with a set of topics $t(i)$. Then, we aim to select a set of k items which collectively cover as many topics as possible: $f(S) = \left|\bigcup_{i \in S} t(i)\right|$. Such formulations have been used across recommendation systems (Ashkan et al. 2015), text summarization (Takamura and Okumura 2009), web search (Agrawal et al. 2009) and image segmentation (Prasad, Jegelka, and Batra 2014).

In many applications, the item-topic associations $t(i)$ are not known in advance. Hence, the learning task is to predict a binary matrix θ where θ_{ij} is 1 if item i covers topic j and 0 otherwise. The optimization task is to find a set of k items maximizing the number of topics covered according to θ. We consider a recommendation systems problem based on the Movielens dataset (GroupLens 2011) in which 2113 users rate 10197 movies (though not every user rated every movie). The items are the movies, while the topics are the top 500 actors. In our problem, the movie-actor assignments are unknown, and must be predicted only from user ratings. This is a *multilabel classification problem* where we attempt to predict which actors are associated with each movie. We randomly divided the movies into 101 problem instances, each with 100 movies. The feature matrix y contains the ratings given by each of the 2113 users for the 100 movies in the instance (with zeros where no rating is present).

Algorithms and experimental setup: In each domain, we randomly divided the instances into 80% training and 20% test. All results are averaged over 30 random splits. Our decision-focused framework was instantiated using feed-forward, fully connected neural networks as the underlying predictive model. All networks used ReLU activations. We experimented with networks with 1 layer, representing a restricted class of models, and 2-layer networks, where the hidden layer (of size 200) gives additional expressive power. We compared two training methods. First, the decision-focused approach proposed above. Second, a two stage approach that uses a machine learning loss function (mean squared error for regression tasks and cross-entropy loss for classification). *This allows us to isolate the impact of the training method since both use the same underlying architecture.* We experimented with additional layers but observed little benefit for either method. All networks were trained using Adam with learning rate 10^{-3}. We refer to the 1-layer decision focused network as *NN1-Decision* and the 1-layer two stage network as *NN1-2Stage* (with analogous names for the 2-layer networks). We also compared to a random forest ensemble of 100 decisions trees (*RF-2Stage*). Gradient-based training cannot be applied to random forests, so benchmark represents a strong predictive model which can be used by two stage approaches but not by our framework. Lastly, we show performance for a random decision.

Solution quality: Table 1 shows the solution quality that each approaches obtains on the full pipeline; i.e., the objective value of its decision evaluated using the true parameters. Each value is the mean (over the 30 iterations) and a bootstrapped 95% confidence interval. For the budget allocation and diverse recommendation tasks, we varied the budget k. The decision-focused methods obtain the highest-performance across the board, tied with random forests on the synthetic budget allocation task.

We now consider each individual domain, starting with budget allocation. Both decision-focused methods substantially outperform the two-stage neural networks, obtaining at least 37% greater objective value. This demonstrates that with fixed predictive architecture, decision-focused learning can greatly improve solution quality. NN1-Decision performs somewhat better than NN2-Decision, suggesting that the simpler class of models is easier to train. However, NN1-2Stage performs significantly worse than NN1-Decision, indicating that alignment between training and the decision problem is highly important for simple models to succeed. RF-2Stage performs essentially equivalently to NN1-Decision. This is potentially surprising since random forest are a much more expressive model class. As we will see later, much of the random forest's success is due to the fact that the features in this synthetic domain are very high-signal; indeed, they suffice for near-perfect reconstruction. The next two domains, both based on real data, explore low-signal settings where highly accurate recovery is impossible.

Table 2: Accuracy of each method according to standard measures.

	Budget allocation	Matching		Diverse recommendation	
	MSE	CE	AUC	CE	AUC
NN1-Decision	0.8673e-02 ± 1.83e-04	0.994 ± 0.002	0.501 ± 0.011	1.053 ± 0.005	0.593 ± 0.003
NN2-Decision	1.7118e-02 ± 2.65e-04	0.689 ± 0.004	**0.560 ± 0.006**	1.004 ± 0.022	0.577 ± 0.008
NN1-2Stage	0.0501e-02 ± 2.67e-06	0.696 ± 0.001	0.499 ± 0.013	0.703 ± 0.001	0.389 ± 0.003
NN2-2Stage	0.0530e-02 ± 2.27e-06	**0.223 ± 0.005**	0.498 ± 0.007	0.690 ± 0.000	**0.674 ± 0.004**
RF-2Stage	**0.0354e-02 ± 4.17e-06**	0.693 ± 0.000	0.500 ± 0.000	**0.689 ± 0.000**	0.500 ± 0.000

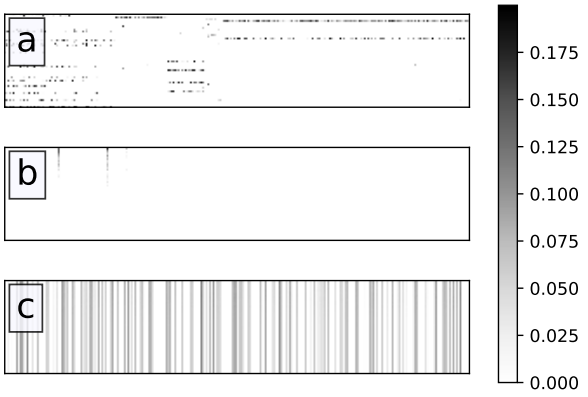

Figure 1: (a) ground truth (b) NN1-2Stage (c) NN1-Decision

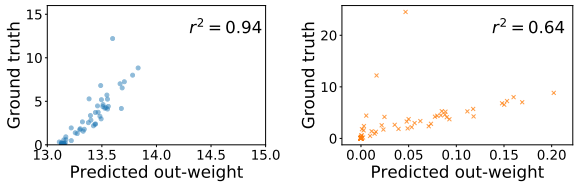

Figure 2: Left: our method's predicted total out-weight for each item. Right: predictions from two stage method.

In bipartite matching, NN2-Decision obtains the highest overall performance, making nearly *over 70% more matches* than the next best method (RF-2Stage, followed closely by NN2-2Stage). Both 1-layer models perform extremely poorly, indicating that the more complex learning problem requires a more expressive model class. However, the highly expressive RF-2Stage does only marginally better than NN2-2Stage, demonstrating the critical role of aligning training and decision making.

In the diverse recommendation domain, NN1-Decision has the best performance, followed closely by NN2-Decision. NN2-2Stage trails by 23%, and NN1-2Stage performs extremely poorly. This highlights the importance of the training method within the same class of models: NN1-Decision obtains approximately 2.7 times greater objective value than NN1-2Stage. RF-2Stage also performs poorly in this domain, and is seemingly unable to extract any signal which boosts decision quality above that of random.

Exploration of learned models: We start out by showing the accuracy of each method according to standard measures, summarized in Table 2. For classification domains (diverse recommendation, matching), we show cross-entropy loss (which is directly optimized by the two stage networks) and AUC. For regression (the budget allocation domain), we show mean squared error (MSE). For budget allocation and diverse recommendation, we fixed $k = 10$.

The two-stage methods are, in almost all cases, significantly more accurate than the decision-focused networks despite their worse solution quality. Moreoever, no accuracy measure is well-correlated with solution quality. On budget allocation, the two decision-focused networks have the worst MSE but the best solution quality. On bipartite matching, NN2-2Stage has better cross-entropy loss but much worse solution quality than NN2-Decision. On diverse recommendation, NN2-2Stage has the best AUC but worse solution quality than either decision-focused network.

This incongruity raises the question of what differentiates the predictive models learned via decision-focused training. We now show more a more detailed exploration of each model's predictions. Due to space constraints, we focus on the simpler case of the synthetic budget allocation task, comparing NN1-Decision and NN1-2Stage. However, the higher-level insights generalize across domains (see the supplement for more detailed visualizations).

Figure 1 shows each model's predictions on an example instance. Each heat map shows a predicted matrix θ, where dark entries correspond to a high prediction and light entries to low. The first matrix is the ground truth. The second matrix is the prediction made by NN1-2Stage, which matches the overall sparsity of the true θ but fails to recover almost all of the true connections. The last matrix corresponds to NN1-Decision and appears completely dissimilar to the ground truth. Nevertheless, these seemingly nonsensical predictions lead to the best quality decisions.

To investigate the connection between predictions and decision, Figure 2 aggregates each model's predictions at the channel level. Formally, we examine the predicted out-weight for each channel u, i.e., the sum of the row θ_u. This is a coarse measure of u's importance for the optimization problem; channels with connections to many customers are more likely to be good candidates for the optimal set. Surprisingly, NN1-Decision's predicted out-weights are extremely well correlated with the ground truth out-weights ($r^2 = 0.94$). However, the absolute magnitude of its predictions are skewed: the bulk of channels have low outweight (less than 1), but NN1-Decision's predictions are all at least 13. By contrast NN1-2Stage has poorer correlation, making it less useful for identifying the outliers which comprise the

optimal set. However, it better matches the values of low out-weight channels and hence attains better MSE. This illustrates how aligning the model's training with the optimization problem leads it to focus on qualities which are specifically important for decision making, even if this compromises accuracy elsewhere.

Acknowledgments: This work was supported by the Army Research Office (MURI W911NF1810208) and a National Science Foundation Graduate Research Fellowship.

References

Agrawal, R.; Gollapudi, S.; Halverson, A.; and Ieong, S. 2009. Diversifying search results. In *WSDM*, 5–14. ACM.

Alon, N.; Gamzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In *WWW*.

Amos, B., and Kolter, J. Z. 2017. Optnet: Differentiable optimization as a layer in neural networks. In *ICML*.

Ashkan, A.; Kveton, B.; Berkovsky, S.; and Wen, Z. 2015. Optimal greedy diversity for recommendation. In *IJCAI*, 1742–1748.

Belanger, D.; Yang, B.; and McCallum, A. 2017. End-to-end learning for structured prediction energy networks. In *ICML*.

Bellur, U., and Kulkarni, R. 2007. Improved matchmaking algorithm for semantic web services based on bipartite graph matching. In *ICWS*, 86–93. IEEE.

Benabbou, N.; Chakraborty, M.; Ho, X.-V.; Sliwinski, J.; and Zick, Y. 2018. Diversity constraints in public housing allocation. In *AAMAS*, 973–981.

Bertsimas, D., and Dunn, J. 2017. Optimal classification trees. *Machine Learning* 106(7):1039–1082.

Beygelzimer, A., and Langford, J. 2009. The offset tree for learning with partial labels. In *KDD*.

Bian, A. A.; Mirzasoleiman, B.; Buhmann, J. M.; and Krause, A. 2017. Guaranteed non-convex optimization: Submodular maximization over continuous domains. In *AISTATS*.

Calinescu, G.; Chekuri, C.; Pál, M.; and Vondrák, J. 2011. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing* 40(6):1740–1766.

Djolonga, J., and Krause, A. 2017. Differentiable learning of submodular models. In *NIPS*, 1013–1023.

Domke, J. 2012. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, 318–326.

Donti, P.; Amos, B.; and Kolter, J. Z. 2017. Task-based end-to-end model learning in stochastic optimization. In *NIPS*.

Elmachtoub, A. N., and Grigas, P. 2017. Smart "predict, then optimize". *arXiv preprint arXiv:1710.08005*.

Fang, F.; Nguyen, T. H.; Pickles, R.; Lam, W. Y.; Clements, G. R.; An, B.; Singh, A.; Tambe, M.; Lemieux, A.; et al. 2016. Deploying paws: Field optimization of the protection assistant for wildlife security. In *IAAI*, 3966–3973.

Ford, B.; Nguyen, T.; Tambe, M.; Sintov, N.; and Delle Fave, F. 2015. Beware the soothsayer: From attack prediction accuracy to predictive reliability in security games. In *International Conference on Decision and Game Theory for Security*.

Gould, S.; Fernando, B.; Cherian, A.; Anderson, P.; Cruz, R. S.; and Guo, E. 2016. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*.

GroupLens. 2011. Movielens dataset.

Hassani, H.; Soltanolkotabi, M.; and Karbasi, A. 2017. Gradient Methods for Submodular Maximization. In *NIPS*.

Horvitz, E., and Mitchell, T. 2010. From data to knowledge to action: A global enabler for the 21st century. *Computing Community Consortium* 1.

Horvitz, E. 2010. From data to predictions and decisions: Enabling evidence-based healthcare. *Computing Community Consortium* 6.

Iyer, R. K.; Jegelka, S.; and Bilmes, J. A. 2014. Monotone closure of relaxed constraints in submodular optimization. In *UAI*.

Jin, C.; Ge, R.; Netrapalli, P.; Kakade, S. M.; and Jordan, M. I. 2017. How to escape saddle points efficiently. In *ICML*.

Karypis, G., and Kumar, V. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing* 20(1):359–392.

Kempe, D.; Kleinberg, J.; and Tardos, É. 2003. Maximizing the Spread of Influence Through a Social Network. In *KDD*.

Khalil, E. B.; Dai, H.; Zhang, Y.; Dilkina, B.; and Song, L. 2017a. Learning combinatorial optimization algorithms over graphs. In *NIPS*.

Khalil, E. B.; Dilkina, B.; Nemhauser, G. L.; Ahmed, S.; and Shao, Y. 2017b. Learning to run heuristics in tree search. In *IJCAI*.

Korte, B.; Vygen, J.; Korte, B.; and Vygen, J. 2012. *Combinatorial optimization*, volume 2. Springer.

Miyauchi, A.; Iwamasa, Y.; Fukunaga, T.; and Kakimura, N. 2015. Threshold influence model for allocating advertising budgets. In *ICML*, 1395–1404.

Mukhopadhyay, A.; Vorobeychik, Y.; Dubey, A.; and Biswas, G. 2017. Prioritized allocation of emergency responders based on a continuous-time incident prediction model. In *AAMAS*, 168–177.

Niculae, V.; Martins, A. F.; Blondel, M.; and Cardie, C. 2018. Sparsemap: Differentiable sparse structured inference. In *ICML*.

Prasad, A.; Jegelka, S.; and Batra, D. 2014. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. In *NIPS*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine* 29(3):93.

Soma, T.; Kakimura, N.; Inaba, K.; and Kawarabayashi, K.-i. 2014. Optimal budget allocation: Theoretical guarantee and efficient algorithm. In *ICML*, 351–359.

Takamura, H., and Okumura, M. 2009. Text summarization model based on maximum coverage problem and its variant. In *EACL*.

Tschiatschek, S.; Sahin, A.; and Krause, A. 2018. Differentiable submodular maximization. In *IJCAI*.

Tu, L., and Gimpel, K. 2018. Learning approximate inference networks for structured prediction. In *ICLR*.

Viappiani, P., and Boutilier, C. 2010. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *NIPS*.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *NIPS*.

Wang, H.; Xie, H.; Qiu, L.; Yang, Y. R.; Zhang, Y.; and Greenberg, A. 2006. Cope: traffic engineering in dynamic networks. In *Sigcomm*, volume 6, 194.

Xue, Y.; Davies, I.; Fink, D.; Wood, C.; and Gomes, C. P. 2016. Avicaching: A two stage game for bias reduction in citizen science. In *AAMAS*, 776–785.

Yahoo. 2007. Yahoo! webscope dataset ydata-ysm-advertiser-bids-v1 0. `http://research.yahoo.com/Academic_Relations`.