Decision-focused learning: integrating downstream combinatorics in ML

Bistra Dilkina
Associate Professor of Computer Science
Co-Director of USC Center on AI in Society
University of Southern California

February 25th, 2021

IPAM-UCLA
Workshop on Deep Learning and Combinatorial Optimization
Augment discrete optimization algorithms with learning components

Learning methods that incorporate the combinatorial decisions they inform
The data-decisions pipeline

Many real-world applications of AI involve a common template:

[Horvitz and Mitchell 2010; Horvitz 2010]

Observe data $\rightarrow$ Predictions $\rightarrow$ Decisions

[Wilder et al., AAAI 2019]
Typical two-stage approach

Machine learning models
- Neural network
- Gaussian process
- Logistic regression
- Random forest

Goal: maximize accuracy

Optimization algorithms
- Greedy
- Local search
- Mixed-integer program
- LP relaxation

Goal: maximize decision quality

[Wilder et al., AAAI 2019]
Google maps

Data → Predictive model → Predicted travel times → Routing algorithm → Shortest path

[Wilder et al., AAAI 2019]
Two-stage training

Challenge

- Maximizing accuracy ≠ maximizing decision quality
- “All models are wrong, some are useful”
- Two-stage training doesn’t align with end goal

Update model to make predictions closer to actual delays
Key idea:

**Decision-focused learning**

Automatically shape the ML model’s loss by incorporating the combinatorial optimization problem into the training loop


Decision-focused learning

Objective function $f(x, \theta)$

- $x \in \{0, 1\}^n$ are the **discrete** decision variables
- $\theta$ are unknown parameters (i.e. the coefficients in the objective e.g., true travel times)

Idea: **Take derivative of decision objective w.r.t. ML model weights**, train model via gradient descent (e.g. similar approach for convex opt. [Donti et al '17])

$$\frac{\partial \text{obj}(\text{decision})}{\partial \text{weights}} = \frac{\partial \text{prediction}}{\partial \text{weights}} \frac{\partial \text{decision}}{\partial \text{prediction}} \frac{\partial \text{obj}(\text{decision})}{\partial \text{decision}}$$

Data $\rightarrow$ Predictive model $\rightarrow$ Predicted delays $\rightarrow$ Optimization algorithm $\rightarrow$ Shortest path

[Wilder et al., AAAI 2019]
Approach

• **Challenge:** the optimization problem is discrete!

• **Solution:** relax to continuous problem, differentiate, round

\[
\begin{align*}
\text{Discrete} & : \quad \max_{x \in X} f(x, \theta) \\
\text{Continuous} & : \quad \max_{x \in \text{conv}(X)} F(x, \theta)
\end{align*}
\]

- \( x = \) binary decision
- \( x = \) fractional decision
- \( F = \) continuous objective

• How to compute \( \frac{dx^*}{d\theta} \)?

• Idea: (locally) optimal continuous solution must satisfy 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.

• Differentiate those equations at optimum (e.g. convex opt. [Donti, Amos, and Kolter 2017])

[Wilder, Dilkina, Tambe, AAAI 2019]

[Wilder et al., AAAI 2019]
Linear programs

Model exactly **combinatorial** problems like bipartite matching, shortest path, mincut, etc.

Or correspond to a relaxation of other **combinatorial** problems

Standard form:

$$
\max_{x} \theta^T x \\
A x \leq b
$$

• $\frac{dx^*}{d\theta}$ doesn’t exist!

• Solution: add a regularizer to smooth things out

$$
\max_{x} \theta^T x - \gamma \|x\|_2^2 \\
A x \leq b
$$

• Now, Hessian is $\nabla^2_{x} f(x, \theta) = -2\gamma I < 0$

• Provably (a) differentiable and (b) close to original LP

[Wilder et al., AAAI 2019]
Results

• Combinatorial problems: encoded as LP, e.g. **bipartite maximum matching**
• Combinatorial problems: submodular maximization, e.g. **influence maximization, budget allocation, diverse recommendation**

• Decision-focused has consistently better solution quality
  • **15-70% improvement in solution over 2-Stage**, across three domains

<table>
<thead>
<tr>
<th>$k$</th>
<th>Budget allocation</th>
<th>Matching</th>
<th>Diverse recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>49.18 ± 0.24</td>
<td>2.50 ± 0.56</td>
<td>15.81 ± 0.50</td>
</tr>
<tr>
<td>10</td>
<td>72.62 ± 0.33</td>
<td>6.15 ± 0.38</td>
<td>29.81 ± 0.85</td>
</tr>
<tr>
<td>20</td>
<td>98.95 ± 0.46</td>
<td>2.99 ± 0.76</td>
<td>52.43 ± 1.23</td>
</tr>
</tbody>
</table>

• But typically much less accurate (wrt AUC, MSE etc.)

[Wilder et al., AAAI 2019]
Application: Tuberculosis treatment

- Follow-on work improving treatment in Indian TB system
- In collaboration with Everwell (NGO)
- **Predict** if patients will miss daily dose
- **Optimize** health worker visits subject to knapsack constraints (LP)

- More in our paper


[Killian et al, KDD 2019]
Application: Tuberculosis treatment

Less “accurate”, but +15% successful interventions!

[Killian et al, KDD 2019]
Decision Focused Learning for Mixed Integer Programming (MIP) problems

• MIPs capture many combinatorial problems that do not have a nice relaxation-based algorithm
• ..and we know how to differentiate through LP optimization

• Idea: cutting planes for MIP results in LP with added cuts
• Differentiate through Cutting-plane-generated LP for training

• At test time, obtain predictions and solve MIP with Branch-and-Bound

Domains

- **Portfolio Optimization**
  - Predict *monthly rate of return* (% return)
  - Optimize *monthly return* for portfolio
  - Limiting risk, sector exposure, transactions...
  - Data: SP500 (USA), DAX (Germany)

- **Diverse Bipartite Matching**
  - Predict *match success probability*
  - Optimize *total number of successful matches*
  - Matching constraints: each node matched at most once
  - Ensure min % of proposed matches are different/same type
  - Data: CORA citation network, nodes = papers, edges = citations

- **Energy Production Knapsack**
  - Predict *energy prices*
  - Optimize *total revenue*
  - Limit on number of time periods we can generate energy
  - Data: ICON Energy Scheduling Challenge

[ Ferber et al, AAAI 2020 ]
Results: decision quality at test time

Objective: monthly % increase for portfolio optimization (SP500 and DAX), number of pairs successfully matched for Matching (CORA), and value of items for Knapsack (Energy).

<table>
<thead>
<tr>
<th></th>
<th>SP500</th>
<th>DAX</th>
<th>Matching</th>
<th>Knapsack</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPaaL</td>
<td>2.79 ± 0.17</td>
<td>5.70 ± 0.68</td>
<td>4.80 ± 0.71</td>
<td>507.70 ± 0.471</td>
</tr>
<tr>
<td>MIPaaL-Warm</td>
<td>1.09 ± 0.18</td>
<td>0.68 ± 1.01</td>
<td>2.14 ± 0.51</td>
<td>499.60 ± 0.566</td>
</tr>
<tr>
<td>MIPaaL-Hybrid</td>
<td>1.08 ± 0.15</td>
<td>0.74 ± 1.10</td>
<td>3.21 ± 0.73</td>
<td>503.36 ± 0.578</td>
</tr>
<tr>
<td>MIPaaL-1000</td>
<td>2.60 ± 0.16</td>
<td>4.39 ± 0.66</td>
<td>3.45 ± 0.71</td>
<td>506.34 ± 0.662</td>
</tr>
<tr>
<td>MIPaaL-100</td>
<td>1.25 ± 0.14</td>
<td>0.35 ± 0.63</td>
<td>2.57 ± 0.54</td>
<td>505.99 ± 0.621</td>
</tr>
<tr>
<td>RootLP (Wilder et al. 2019)</td>
<td>1.97 ± 0.17</td>
<td>-1.97 ± 0.69</td>
<td>3.17 ± 0.60</td>
<td>501.58 ± 0.662</td>
</tr>
<tr>
<td>TwoStage</td>
<td>1.19 ± 0.15</td>
<td>0.70 ± 1.46</td>
<td>3.42 ± 0.78</td>
<td>501.49 ± 0.523</td>
</tr>
</tbody>
</table>

- MIPaaL gives **2x monthly returns on SP500 and 8x on DAX**
- MIPaaL **improves the objective by 40.3% and 1.2% for Matching and Knapsack respectively.**
- MIPaaL outperforms all other variants considered.

[Ferber et al, AAAI 2020]
Transfer Learning

- Learn on one distribution of assets (30\textsuperscript{a} SP assets) and test on another (30\textsuperscript{b} other SP assets and 30 DAX assets), keeping the MIP size the same
- Learn on one size of MIPs (number of assets available, 30 SP) and test on larger MIPs (with more assets to choose from 50-500 SP)

<table>
<thead>
<tr>
<th>Decision Quality</th>
<th>SP-30\textsuperscript{b}</th>
<th>DAX</th>
<th>SP-50</th>
<th>SP-100</th>
<th>SP-200</th>
<th>SP500</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPaaL</td>
<td>2.02 ± 0.48</td>
<td>2.77 ± 0.40</td>
<td>1.93 ± 0.13</td>
<td>2.27 ± 0.11</td>
<td>2.17 ± 0.48</td>
<td>2.26 ± 0.37</td>
</tr>
<tr>
<td>RootLP</td>
<td>1.81 ± 0.44</td>
<td>1.74 ± 0.43</td>
<td>1.50 ± 0.09</td>
<td>1.58 ± 0.08</td>
<td>1.82 ± 0.41</td>
<td>1.90 ± 0.29</td>
</tr>
<tr>
<td>TwoStage</td>
<td>0.71 ± 0.04</td>
<td>0.82 ± 0.54</td>
<td>1.58 ± 0.13</td>
<td>1.22 ± 0.09</td>
<td>1.50 ± 0.58</td>
<td>1.11 ± 0.35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ML Loss</th>
<th>SP-30\textsuperscript{b}</th>
<th>DAX</th>
<th>SP-50</th>
<th>SP-100</th>
<th>SP-200</th>
<th>SP500</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPaaL</td>
<td>4.81 ± 8.59</td>
<td>4.59 ± 8.80</td>
<td>5.42 ± 3.16</td>
<td>5.42 ± 2.37</td>
<td>5.25 ± 1.83</td>
<td>5.43 ± 1.67</td>
</tr>
<tr>
<td>RootLP</td>
<td>5.14 ± 1.02</td>
<td>5.39 ± 1.04</td>
<td>4.73 ± 3.17</td>
<td>4.88 ± 2.58</td>
<td>4.81 ± 1.91</td>
<td>4.83 ± 1.56</td>
</tr>
<tr>
<td>TwoStage</td>
<td>0.08 ± 0.05</td>
<td>0.07 ± 0.03</td>
<td>0.08 ± 0.02</td>
<td>0.07 ± 0.01</td>
<td>0.08 ± 0.01</td>
<td>0.08 ± 0.01</td>
</tr>
</tbody>
</table>
Decision-Focused Learning

- No need to silo out ML vs Optimization tasks
- When data is scarce, we want predictions to be accurate where it matters most for decision making
- Marrying predictive and prescriptive tasks in a unified end-to-end system

Related Frameworks:
- Empirical decision model learning (M Lombardi, M Milano, A Bartolini, Artificial Intelligence 2017)
- “Predict-and-optimize” framework and its variants (Elmachtoub & Grigas, 2017; Demirovic et al., 2019; Mandi et al., AAAI 2020)
- Blackbox differentiation of combinatorial solvers (Vlastelica et al, ICLR 2020; Rolínek et al, ECCV 2020; Paulus et al NeurIPS 2020 LMCA Workshop)
Relax + differentiate

Forward pass: run a solver

Backward pass: sensitivity analysis via KKT conditions

Convex QPs [Amos and Kolter 2017, Donti et al 2017]
Linear and submodular programs [Wilder, Dilkina, Tambe 2019]
MAXSAT (via SDP relaxation) [Wang, Donti, Wilder, Kolter 2019]
MIPs [Ferber, Wilder, Dilkina, Tambe 2020]

Some problems don’t have good relaxations
Slow to solve continuous optimization problem
Slow to backprop through – $O(n^3)$
An Alternative Approach

• Learn a **representation** that maps the original problem to a simpler (efficiently differentiable) **proxy problem**.

• **Instantiation for a class of graph problems**: k-means clustering in embedding space.

Graph learning + graph optimization

[Wilder et al, NeurIPS 2019]
Problem classes

• Partition the nodes into K disjoint groups
  • Community detection, maxcut, …

• Select a subset of K nodes
  • Facility location, influence maximization, …

• Methods of choice are often combinatorial/discrete

Approach

• Observation: clustering nodes is a good proxy
  • Partitioning: correspond to well-connected subgroups
  • Facility location: put one facility in each community

• Observation: graph learning approaches already embed into $R^n$

[Wilder et al, NeurIPS 2019]
ClusterNet

One architecture and training process for all problems in these classes, which automatically learns a differentiable solver for a given problem

1. Embed nodes with GCN (Goal: train GCN to produce task-specific embeddings)
2. Run soft K-means on embeddings
3. Interpret clustering as optimization solution
4. Backpropagate optimization objective value

[Wilder et al, NeurIPS 2019]
Differentiable K-means

Forward pass

Update cluster centers

Softmax update to node assignments

\[ \mu_k = \frac{\sum_j r_{jk} x_j}{\sum_j r_{jk}} \]

\[ r_{jk} = \frac{\exp(-\beta ||x_j - \mu_k||)}{\sum_\ell \exp(-\beta ||x_j - \mu_\ell||)} \]

[Wilder et al, NeurIPS 2019]
Differentiable K-means

Backward pass

• Option 1: differentiate through the fixed-point condition
  \[ \mu^t = \mu^{t+1} \]
  • Prohibitively slow, memory-intensive

• Option 2: unroll the entire series of updates
  • Cost scales with # iterations
  • Have to stick to differentiable operations

• Option 3: get the solution, then unroll one update
  • Do anything to solve the forward pass
  • Linear time/memory, implemented in vanilla pytorch

Theorem [informal]: provided the clusters are sufficiently balanced and well-separated, the Option 3 approximate gradients converge exponentially quickly to the true ones.

[Wilder et al, NeurIPS 2019]
Example: community detection

Observe partial graph  
Predict unseen edges  
Find communities

\[
\max \text{ modularity} \quad \max_r \frac{1}{2m} \sum_{u,v \in V} \sum_{k=1}^{K} \left[ A_{u,v} - \frac{d_u d_v}{2m} \right] r_{uk} r_{vk}
\]

\[
r_{uk} \in \{0,1\} \quad \forall u \in V, \: k = 1 \ldots K
\]

\[
\sum_{k=1}^{K} r_{uk} = 1 \quad \forall u \in V
\]

• **Useful in scientific discovery** (social groups, functional modules in biological networks)

• In applications, **two-stage approach is common**:  
  [Yan & Gregory ’12, Burgess et al ‘16, Berlusconi et al ‘16, Tan et al ‘16, Bahulker et al ’18…]

[Wilder et al, NeurIPS 2019]
Experiments

• **Learning problem**: link prediction

• **Optimization**:
  • community detection
  • facility location problems

• **Train GCNs** as predictive component

• **Comparison**
  • Two stage: GCN + expert-designed algorithm (**2Stage**)
  • Pure end to end: Deep GCN to predict optimal solution (**e2e**)

• **ClusterNet**:
  • Community detection (use clusters as-is, measure modularity)
  • Facility location (one location in each cluster, measure max distance)

[Wilder et al, NeurIPS 2019]
Results: single-graph link prediction

Representative example from cora, citeseer, protein interaction, facebook, adolescent health networks

Community algos: CNM, Newman, SpectralClustering
Facility Locations algos: greedy, gonzalez2approx

[Wilder et al, NeurIPS 2019]
Results: generalization across graphs

ClusterNet learns generalizable strategies for optimization!

[Wilder et al, NeurIPS 2019]
Decoupled approaches (2-stage) and pure end-to-end methods miss out on useful structure.

Good decisions require integrating learning and optimization as in decision-focused learning.

Even simple optimization primitives (e.g. clustering) provide good inductive bias.