Faster Matchings via Learned Duals

Michael Dinitz[1], Sungjin Im[2], Thomas Lavastida[3], Benjamin Moseley[3], Sergei Vassilvitskii[4]

[1]Johns Hopkins University, [2]UC Merced, [3]Carnegie Mellon University, [4]Google Research

Learning Augmented Algorithms

- Recently there has been significant interest in incorporating machine learning into the design of algorithms
- Predominantly applied to online algorithms
- Potential to speed up combinatorial optimization?

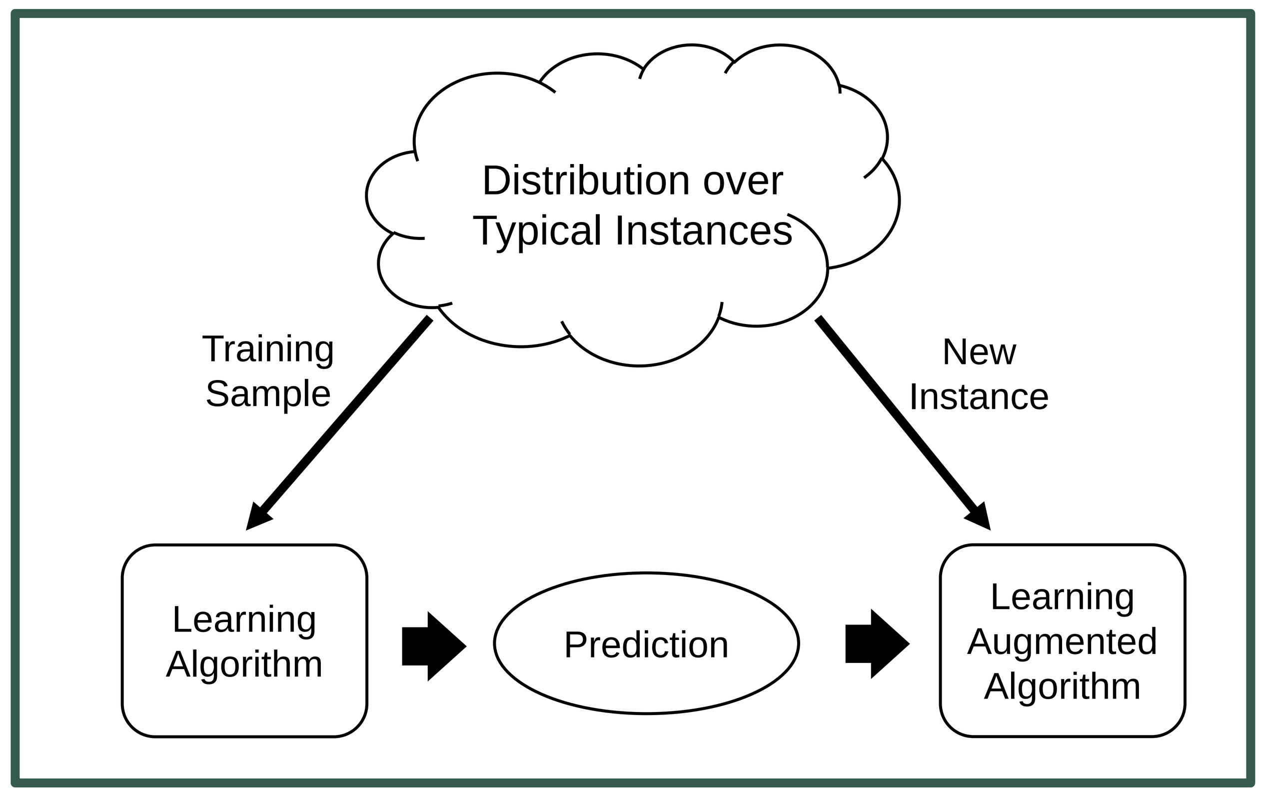

Figure 1: Illustration of model for learning augmented algorithms

Minimum Weight Perfect Matching

- Fundamental combinatorial optimization problem
- Natural place to explore learning augmented algorithms for running time
- Bipartite graph $G = (V, E)$ w/ weights $c \in \mathbb{Z}^E$, $|V| = n, |E| = m$
- Goal: find a perfect matching M of minimum total weight
- Known that natural linear program is exact

$$(P) \quad \begin{aligned} \min \sum_{e \in E} c_e x_e & \\ \sum_{e \in \delta(i)} x_e = 1, & \forall i \in V \\ x_e \geq 0, & \forall e \in E \end{aligned}$$

Figure 2:Primal LP for minimum weight perfect matching

$$(D) \quad \begin{aligned} \max \sum_{i \in V} y_i & \\ y_i + y_j \leq c_{ij} & \\ \forall ij \in E & \end{aligned}$$

Figure 3: Dual LP for minimum weight perfect matching

- Classic Hungarian algorithm solves efficiently in practice
- Faster methods known in theory
- Main question: What to predict and how to measure loss?

Framework: Warm Start Primal-Dual

- Idea: Predict optimal dual variables
- Check optimality with single call to max cardinality matching
- Would like our prediction $\hat{y}$ to be "close" to optimal y^*
- Need to handle 3 challenges:
  1. Feasibility – $\hat{y}$ may not be feasible for new instance
  2. Optimization – Should exploit any "closeness" of $\hat{y}$ to y^*
  3. Learning – Constructing $\hat{y}$ should use past data efficiently

Restoring Feasibility + Optimization

- Suppose that $\hat{y}$ infeasible for new instance
- Make feasible while retaining dual objective value
- Model as linear program with variables δ_i for $i \in V$ representing decrease to $\hat{y}_i$, i.e., insist $\hat{y} - \delta$ is feasible
- Let $r_{ij} = \max\{\hat{y}_i + \hat{y}_j - c_{ij}, 0\}$

$$(P_F) \quad \begin{aligned} \min \sum_{i \in V} \delta_i & \\ \delta_i + \delta_j \geq r_{ij} & \ \forall ij \in E \\ \delta_i \geq 0, & \forall i \in V \end{aligned}$$

Figure 4: Primal LP for feasibility problem

$$(D_F) \quad \begin{aligned} \max \sum_{e \in E} r_e \gamma_e & \\ \sum_{e \in \delta(i)} \gamma_e \leq 1, & \forall i \in V \\ \gamma_e \geq 0, & \forall e \in E \end{aligned}$$

Figure 5: Dual LP for feasibility problem

Theorem 1: There is a linear time 2-approximation alg. for P_F.

- Once we have a feasible dual solution, we consider the classic Hungarian algorithm to reach optimality

Theorem 2: Given any feasible dual $\hat{y}$, the Hungarian algorithm has running time $O(m\sqrt{n}\|y^* - \hat{y}\|_1)$.

- Gives us a loss function to use when learning: $\|y^* - \hat{y}\|_1$.
- Restoring feasibility from an arbitrary prediction only loses $O(1)$ factors to this loss
- Still strongly polynomial time even when predictions are bad

Learning Initial Duals

- Formulate as a PAC learning problem
- Let D be an unknown distribution on weight vectors c
- Goal: For any $\epsilon, \rho > 0$, use small number of samples from D to learn duals $\hat{y}$ such that:

$$\mathbb{E}_{c\sim D}[\|y^*(c) - \hat{y}\|_1] \leq \min_{y} \mathbb{E}_{c\sim D}[\|y^*(c) - y\|_1] + \epsilon$$

  with probability $1 - \rho$.
- $y^*(c)$ is an optimal dual solution for weights c

Theorem 3: When $c_{ij} \in [-C, C]$ w.p. 1, then $\tilde{O}\left(n^3 C^2 \epsilon^{-2} \log\frac{1}{\rho}\right)$ samples are sufficient to learn $\hat{y}$.

- Follows from a pseudo-dimension argument

Experiments

- Goal is to confirm theory – show that when there is something to learn, learned duals outperform the Hungarian algorithm
- Construct distributions over instances using datasets taken from UCI Machine Learning repository
- Assume data belongs to $\mathbb{R}^d$, construct distribution as follows:
  1. Randomly partition data into two sets L, R
  2. Run k-means clustering on each of L and R
  3. To sample an instance, sample one point from each cluster
  4. Set c_{ij} to be the distance between the points sampled from cluster i and cluster j on each side, respectively
- We set $k = 500$ and use 20 samples for learning initial duals
- Report the average runtime on 10 test instances.

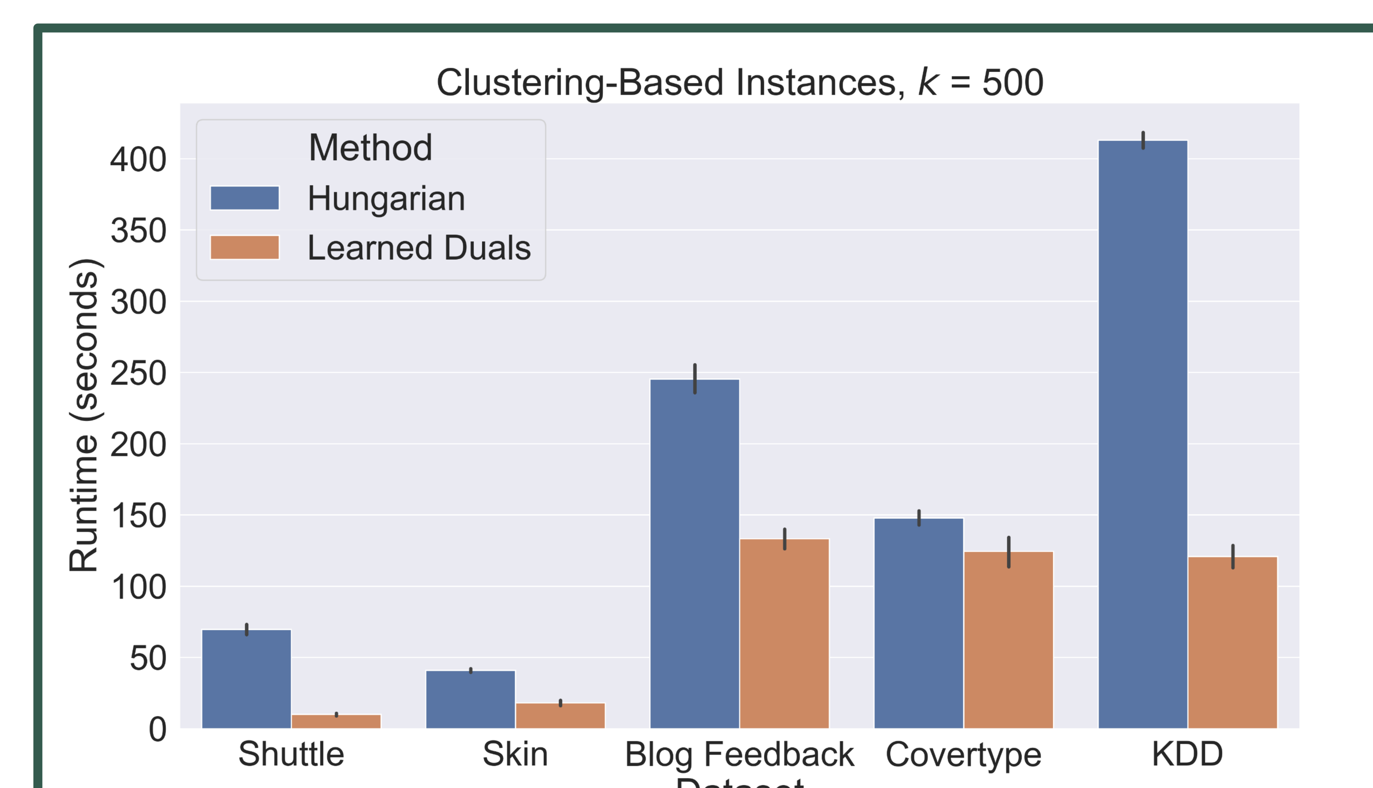

Figure 6: Experimental results on UCI datasets. Note that we typically see > 2X speedup when using learned duals over the Hungarian algorithm