Machine Learning under a Modern Optimization Lens

Dimitris Bertsimas

Operations Research Center
Massachusetts Institute of Technology

January 2018

Outline

1. **Motivation**
2. **Sparse High Dimensional Regression: Exact Scalable Algorithms and Phase Transitions** with Bart van Parys, under review *Annals of Statistics*, 2016.
3. **Sparse Classification: a discrete optimization perspective** with Jean Pauphilet and Bart van Parys, under review *JMLR*, 2017.
4. **Optimal Trees** with Jack Dunn, *Machine Learning,* 106(7), 1039-1082, 2017.
5. **Conclusions**

Motivation

- Some of the central problems in Machine Learning (ML)/ Statistics (S) / (regression, classification and estimation) have been addressed using heuristic methods; (Lasso for best subset regression or CART for optimal classification).
- This implies that we do not really know if we have indeed solved these problems.
- While convex optimization (CO) has had impact in ML/S: Compressed Sensing, Matrix Completion, Mixed integer optimization (MIO) and Robust Optimization (RO) are relatively unknown in ML/S.
- ML/S considers MIO problems to be intractable.
- Yet MIO, RO, CO have advanced very significantly.

Progress of MIO

- Speed up between CPLEX 1.2 (1991) and CPLEX 11 (2007): 29,000 times
- Gurobi 1.0 (2009) comparable to CPLEX 11
- Speed up between Gurobi 1.0 and Gurobi 6.5 (2015): 48.7 times
- Total speedup 1991-2015: 1,400,000 times
- A MIO that would have taken 16 days to solve 25 years ago can now be solved on the same 25-year-old computer in less than one second.
- Hardware speed: 93.0 PFlop/s in 2016 vs 59.7 GFlop/s in 1993 1,600,000 times
- Total Speedup: **2.2 Trillion times!**
- A MIO that would have taken 71,000 years to solve 25 years ago can now be solved in a modern computer in less than one second.

Remarks on Complexity

- A 2.2 Trillion speed up forces us to reconsider what is tractable.
- A problem is tractable if it can be solved for sizes and in times that are appropriate for the application.
- Asymptotic polynomial solvability or NP-hardness is not relevant under this definition.

Research Objectives

- To demonstrate that using modern optimization (MIO, RO, CO) optimal solutions to large scale instances in ML/S
  - can be found in seconds.
  - can be certified to be optimal in minutes.
  - outperform classical heuristic approaches in out of sample experiments involving real and synthetic data.

- To bring closer ML/S to Optimization.

- To affect the teaching of ML/S. In the Fall 2017 and Spring 2018 I am teaching a MS and doctoral class at MIT on the topic of this lecture.

Sparse Linear Regression, B.+van Parys, 2016

- Problem with reguralization

$$\begin{aligned} \min_{w} \quad & \frac{1}{2\gamma}\|w\|_2^2 + \frac{1}{2}\|Y - Xw\|_2^2 \\ \text{s.t.} \quad & \|w\|_0 \le k, \end{aligned}$$

- We rewrite $(w_s)_i = s_i w_i,\ s_i \in \{0,1\}$.
- $\mathrm{S}_k^p := \left\{s \in \{0,1\}^p \ : \ \mathbf{1}^\top s \le k\right\}$

$$\min_{s \in \mathrm{S}_k^p} \left[\min_{w_s \in \Re^k} \ \frac{1}{2\gamma}\|w_s\|_2^2 + \frac{1}{2}\|Y - X_s w_s\|_2^2\right].$$

- Solution:

$$\begin{aligned} \min \quad & c(s) = \frac{1}{2} Y^\top \left(\mathbb{I}_n + \gamma \textstyle\sum_{j \in [p]} s_j K_j\right)^{-1} Y \\ \text{s.t.} \quad & s \in \mathrm{S}_k^p, \end{aligned}$$

- $K_j := X_j X_j^\top$.
- Binary convex optimization problem.

A Cutting Plane Algorithm

- Input: $Y \in \Re^n$, $X \in \Re^{n\times p}$ and $k \in [1, p]$.
- Output: $s^\star \in \mathrm{S}_k^p$ and $w^\star \in \Re^p$.
- $s_1 \leftarrow$ warm start
  $\eta_1 \leftarrow 0$
  $t \leftarrow 1$
- While $\eta_t < c(s_t)$
  - $s_{t+1},\ \eta_{t+1} \leftarrow \arg\min_{s,\,\eta} \{ \eta \in \Re_+ \text{ s.t. } s \in \mathrm{S}_k^p,\ \ \eta \geq c(s_t) + \nabla c(s_t)(s - s_t),\ \forall i \in [t]\}$
  - $t \leftarrow t + 1$
- $s^\star \leftarrow s_t$
- $w^\star \leftarrow 0, \qquad w^\star_{s^\star} \leftarrow \left(\mathbb{I}_p/\gamma + X_{s^\star}^\top X_{s^\star}\right)^{-1} X_{s^\star}^\top Y$

Scalability

Cutting plane algorithm is faster than Lasso.

		Exact T [s]			Lasso T [s]		
		$n = 10\text{k}$	$n = 20\text{k}$	$n = 100\text{k}$	$n = 10\text{k}$	$n = 20\text{k}$	$n = 100\text{k}$
$k = 10$	$p = 50\text{k}$	21.2	34.4	310.4	69.5	140.1	431.3
	$p = 100\text{k}$	33.4	66.0	528.7	146.0	322.7	884.5
	$p = 200\text{k}$	61.5	114.9	NA	279.7	566.9	NA
$k = 20$	$p = 50\text{k}$	15.6	38.3	311.7	107.1	142.2	467.5
	$p = 100\text{k}$	29.2	62.7	525.0	216.7	332.5	988.0
	$p = 200\text{k}$	55.3	130.6	NA	353.3	649.8	NA
$k = 30$	$p = 50\text{k}$	31.4	52.0	306.4	99.4	220.2	475.5
	$p = 100\text{k}$	49.7	101.0	491.2	318.4	420.9	911.1
	$p = 200\text{k}$	81.4	185.2	NA	480.3	884.0	NA

Phase Transitions

- $Y = Xw_{\mathrm{true}} + E$ where E is zero mean noise uncorrelated with the signal Xw_{true}.

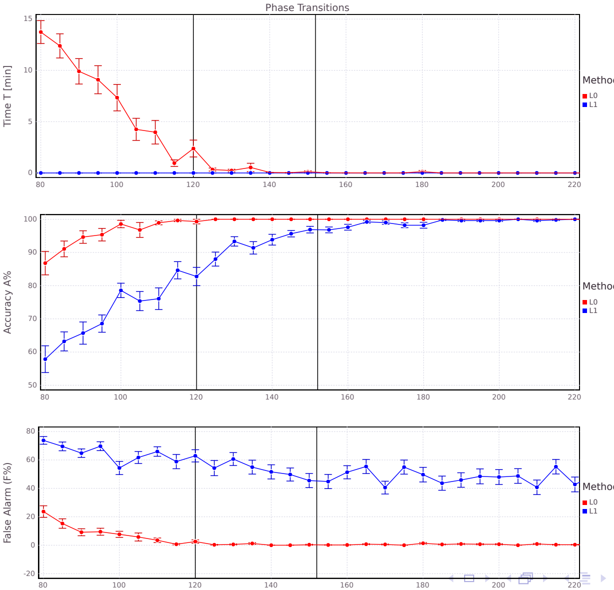
- Accuracy and false alarm rate of a certain solution $w^\star$

$$A\% := 100 \times \frac{|\mathrm{supp}(w_{\mathrm{true}}) \cap \mathrm{supp}(w^\star)|}{k}$$

$$F\% := 100 \times \frac{|\mathrm{supp}(w^\star) \setminus \mathrm{supp}(w_{\mathrm{true}})|}{|\mathrm{supp}(w^\star)|}.$$

- Perfect support recovery occurs only then when $w^\star$ tells the whole truth ($A\% = 100$) and nothing but the truth ($F\% = 0$).

Phase Transitions



Phase Transitions

Phase transition happens at

$$n > n^\star = \frac{2k \log p}{\log\left(\frac{2k}{\sigma^2} + 1\right)}.$$

Remark on Complexity

- Traditional complexity theory suggests that the difficulty of a problem increases with dimension.
- Sparse regression problem has the property that for small number of samples n, the dual approach takes a large amount of time to solve the problem, but most importantly the optimal solution does not recover the true signal.
- However, for a large number of samples n, dual approach solves the problem extremely fast and recovers 100% of the support of the true regressor w_{true}.

Sparse Classification, B.+Pauphilet+ Bart van Parys

- $\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} \sum_{i=1}^n \ell(y_i, w^T x_i + b) + \frac{1}{2\gamma}\|w\|_2^2$,
- Problem equivalent to

$$\min_{s\in S_k^p} c(s),$$

where for any $s \in \{0,1\}^p$,

$$c(s) := -\sum_{i=1}^n \hat{\ell}(y_i, \alpha_i) - \frac{\gamma}{2}\sum_{j=1}^n s_j \alpha^T X_j X_j^T \alpha \;\; \text{s.t.} \;\; \mathbf{e}^T \alpha = 0.$$

- $\hat{\ell}(y, \alpha) := \max_{u\in\mathbb{R}} u\alpha - \ell(y, u)$ is the *Fenchel conjugate* of the loss function ℓ.
- $c(s)$ is convex over $[0,1]^p$.

Problems

Method	Loss $\ell(y,u)$	Fenchel conjugate $\hat{\ell}(y,\alpha)$
Logistic loss	$\log\left(1+e^{-yu}\right)$	$\begin{cases}(1+y\alpha)\log(1+y\alpha)-y\alpha\log(-y\alpha), & y\alpha \in \\ +\infty, & \text{other}\end{cases}$
1-norm SVM	$\max(0,1-yu)$	$\begin{cases} y\alpha, & \text{if } y\alpha\in[-1,0], \\ +\infty, & \text{otherwise.}\end{cases}$
2-norm SVM	$\frac{1}{2}\max(0,1-yu)^2$	$\begin{cases}\frac{1}{2}\alpha^2+y\alpha, & \text{if } y\alpha \leqslant 0, \\ +\infty, & \text{otherwise.}\end{cases}$

Cutting-plane procedure

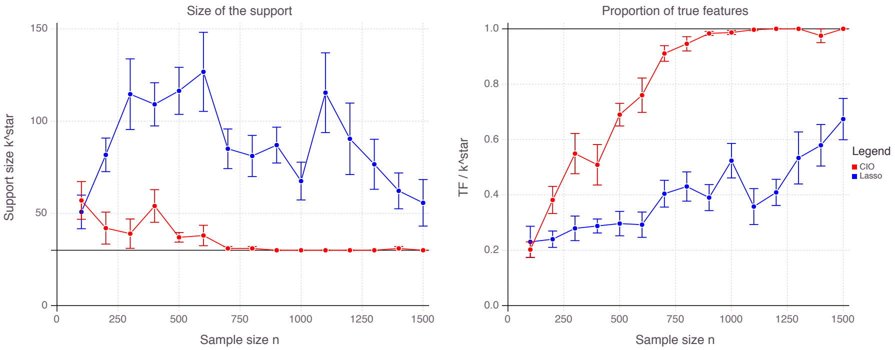
- $\dfrac{\partial c}{\partial s_j}(s) = -\dfrac{\gamma}{2}\alpha^*(s)^T X_j X_j^T \alpha^*(s).$
- Outer-approximation algorithm
- $X \in \mathbb{R}^{n\times p}$, $Y \in \{-1,1\}^p$, $k \in \{1,...,p\}$
- $s_1 \leftarrow$ warm-start, $\eta_1 \leftarrow 0$, $t \leftarrow 1$
- Repeat $s_{t+1}, \eta_{t+1} \leftarrow$ $\text{argmin}_{s,\eta} \left\{ \eta \; : s \in S_k^p, \eta \geqslant c(s_i) + \nabla c(s_i)\dot{(}s - s_i)\forall i = 1,...,t \right\}$
- $t \leftarrow t+1$
- Until $\eta_t < c(s_t)$

- Algorithm converges to an optimal solution in a finite number of iterations.

- Algorithm scales to $n, p = 10,000$,

Sparse logistic regression, $p = 1000$

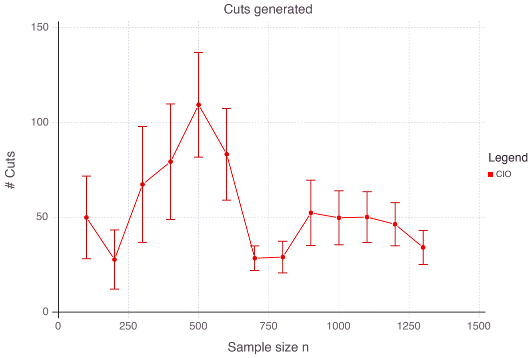
Optimal sparsity $k^\star$ (left) and proportion of true features $TF/k^\star$ (right) as n increases,



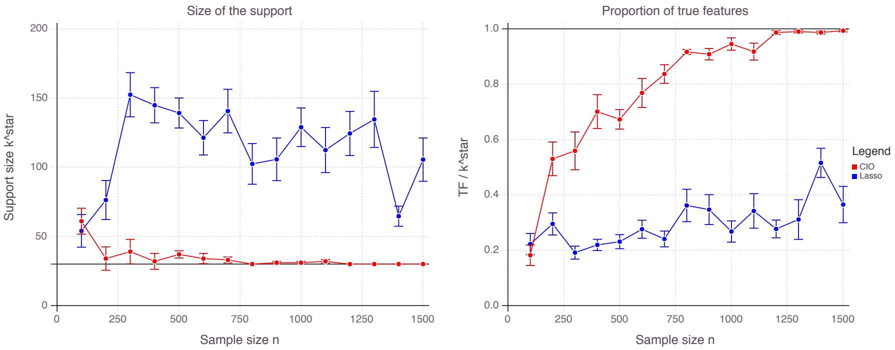
Real world data sets

			Sparsity k		AUC	
Data set	n	p	Sparse	Lasso	Sparse	Lasso
Lung cancer	$1,145$	$14,858$	50	171	0.9816	0.9865

Sparse SVM, $p = 1000$



Optimal sparsity $k^\star$ (left) and proportion of true features $TF/k^\star$ (right) as n increases,



Phase Transitions

Phase transition happens at

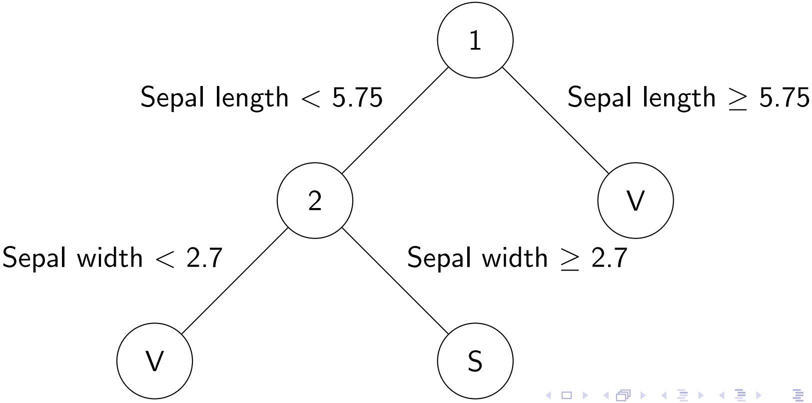
$$C\left(1 + C'\frac{\sigma^2}{k}\right) k \log(p-k)$$

Classification

- Classification is a key problem in Machine Learning
  - Given training data $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$, we want to learn a function for predicting y based on $\mathbf{x}$
  - $\mathbf{x}_i \in \mathbb{R}^p$ are the features of the data
  - $y_i \in \{-1, +1\}$ are the labels $\implies$ binary classification
- **Example:** Iris dataset from UCI Machine Learning Repository
  - 150 iris flowers of three different types
  - Four measurements for each flower: petal width/height and sepal width/height
  - Task: Predict the iris type using the measurements

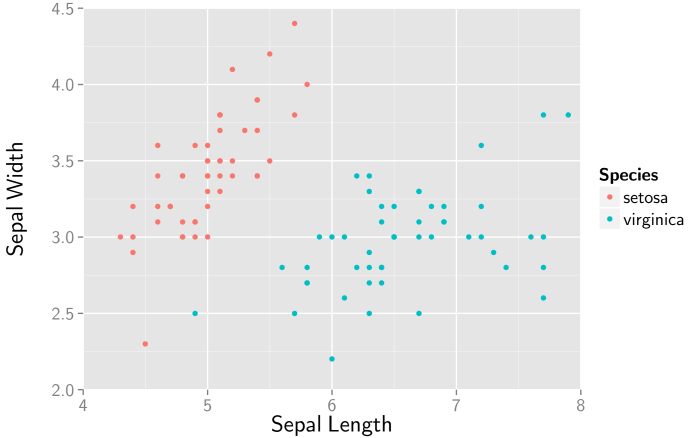
Decision Trees

- Decision tree methods are a popular and successful method for classification
  - Create a recursive partitioning of the features to classify points
  - CART (Breiman et al, 1984) is the state-of-the-art method in this area
  - Widespread use in academia and industry ($\sim$ 33000 citations!)



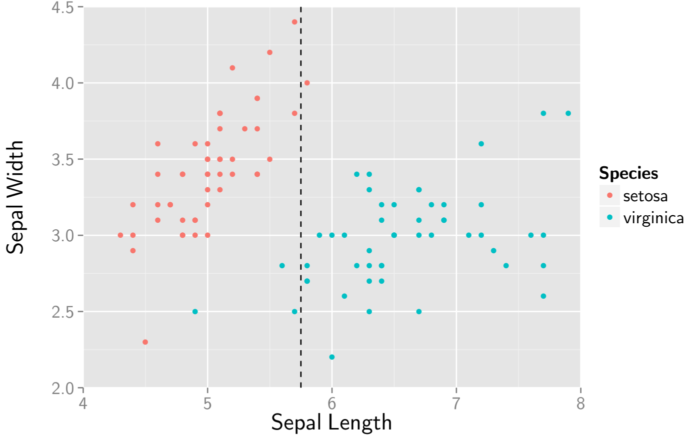
How does CART work?

- Each split branches on the value of on a single feature
- Greedy approach to partitioning—make a locally optimal split then recurse on both children



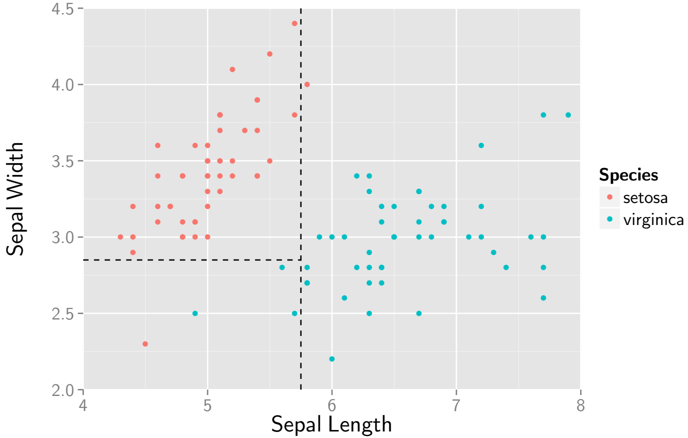
How does CART work?

- Each split branches on the value of on a single feature
- Greedy approach to partitioning—make a locally optimal split then recurse on both children



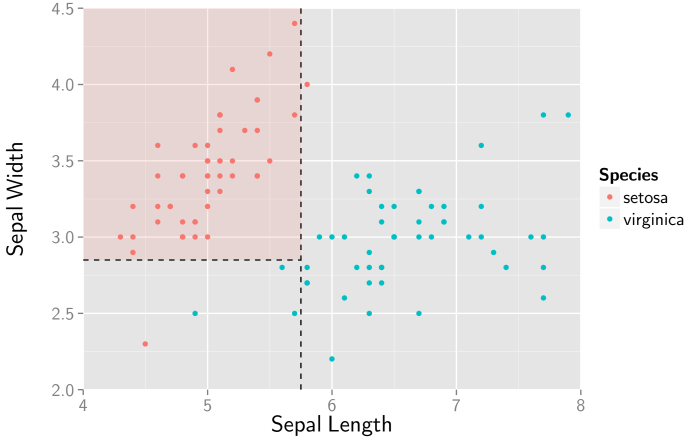
How does CART work?

- Each split branches on the value of on a single feature
- Greedy approach to partitioning—make a locally optimal split then recurse on both children



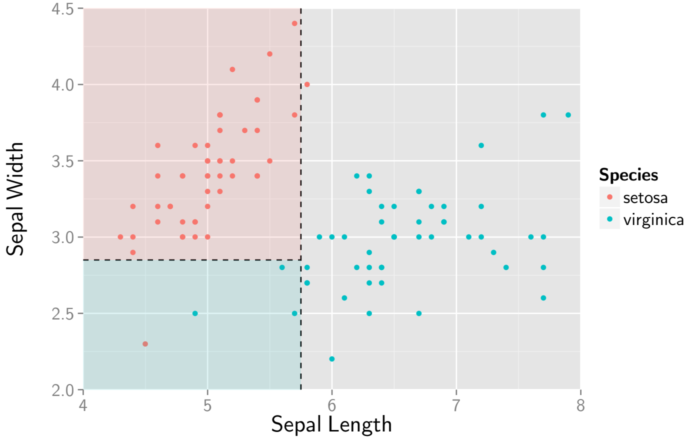
How does CART work?

- Each split branches on the value of on a single feature
- Greedy approach to partitioning—make a locally optimal split then recurse on both children



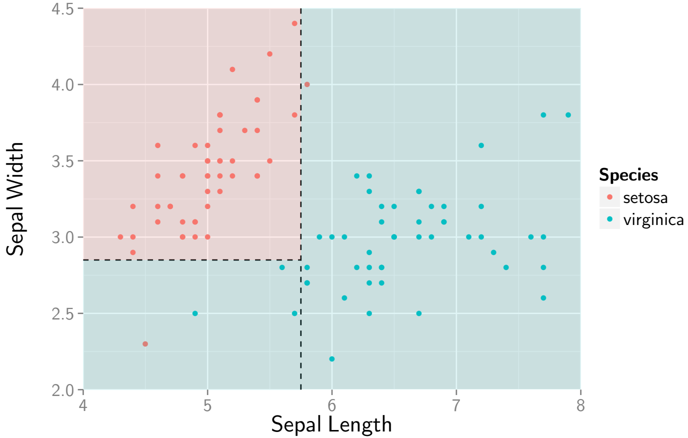
How does CART work?

- Each split branches on the value of on a single feature
- Greedy approach to partitioning—make a locally optimal split then recurse on both children



How does CART work?

- Each split branches on the value of on a single feature
- Greedy approach to partitioning—make a locally optimal split then recurse on both children



Observations

- CART is fundamentally greedy—it makes a series of locally optimal decisions, but the final tree could be far from optimal
  - Can we find **globally optimal** decision trees instead?
  - How far from optimality are trees created by current methods?

- There have been many attempts to find methods for globally optimal decision trees in the literature. Examples include:
  - Linear optimization (Bennett, 1992)
  - Continuous non-linear optimization (Bennett and Blue, 1996)
  - Dynamic programming (Cox Jr et al, 1989; Payne and Meisel 1977)
  - Genetic algorithms (Son, 1998)

- To date, there has not been a globally optimal decision tree method that is tractable and is able to scale to the typical problem sizes seen in classification

Breiman's take on globally optimal trees

> *Finally, another problem frequently mentioned (by others, not by us) is that the tree procedure is only one-step optimal and not overall optimal. ... If one could search all possible partitions ... the two results might be quite different.*
>
> *We do not address this problem. At this stage of computer technology, an overall optimal tree growing procedure does not appear feasible for any reasonably sized data set.*
>
> — Breiman et al. (1984)

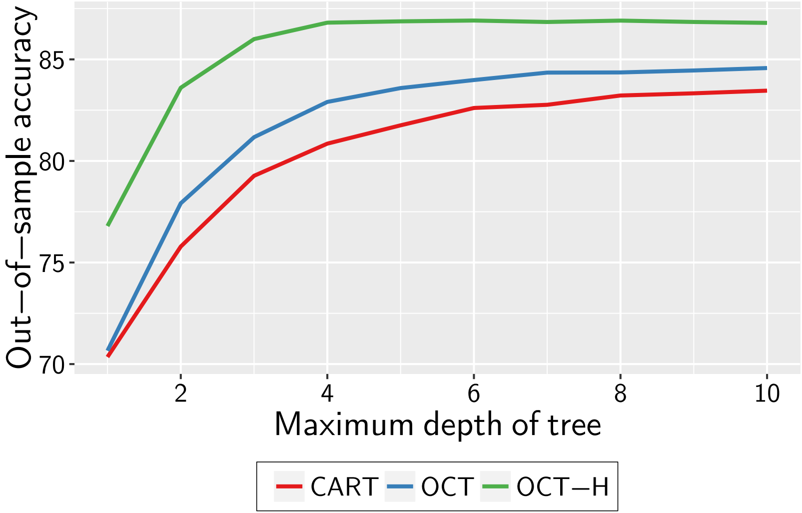
- CART's use of locally-optimal splits rather than globally-optimal was guided by practical limitations at the time.

Our Approach

- From: B.+Dunn, "Optimal Trees", *Machine Learning*, 2017.
- Use Mixed-Integer Optimization (MIO) to consider the entire decision tree problem at once and solve to obtain the Optimal Tree
- **Motivation:** MIO is the natural form for the Optimal Tree problem:
  - Decisions: Which variable to split on, which label to predict for a region
  - Outcomes: Which region a point ends up in, whether a point is correctly classified
- **Aspirations:** Driven by recent improvements in MIO we seek new decision trees that:
  - Are globally optimal (or near-optimal)
  - Provide a guarantee of optimality (or a measure of sub-optimality)
  - Can be found in times appropriate for the application

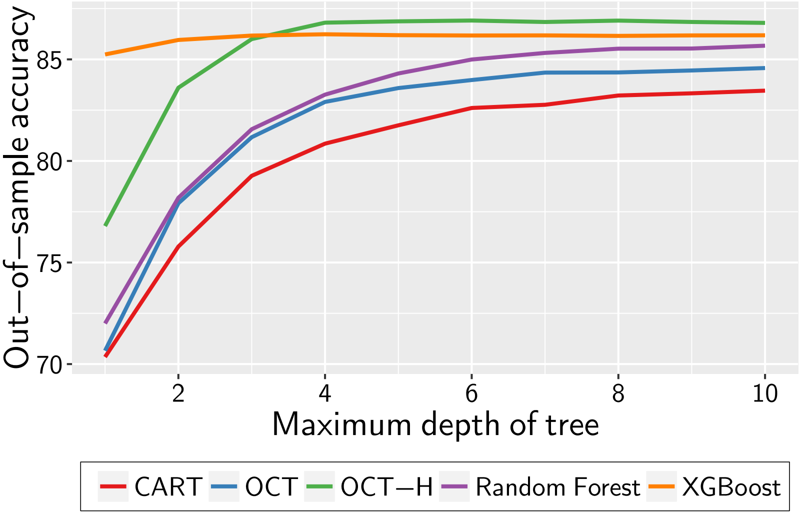
Performance of Optimal Classification Trees

- Average out-of-sample accuracy across 60 real-world datasets:



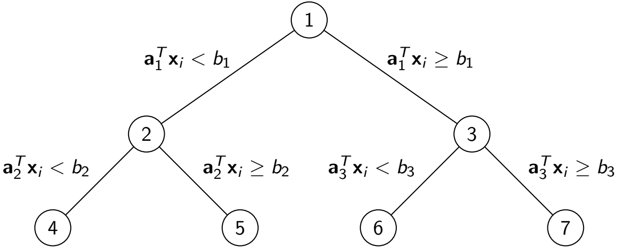
Performance of Optimal Classification Trees

- Average out-of-sample accuracy across 60 real-world datasets:



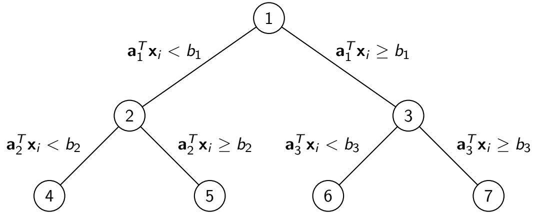
Forming the tree structure

- Given training data $(\mathbf{x}_i, y_i), i = 1, \ldots, n$
- Specify a maximum depth and form a complete tree of that depth



- Define sets for branches, $\mathcal{B} = \{1, 2, 3\}$, and leaves, $\mathcal{L} = \{4, 5, 6, 7\}$.
- Variables $\mathbf{a}_t$, b_t define the split at each branch node $t \in \mathcal{B}$
- We want splits that are parallel to the axes (one feature at a time)
  - Elements of $\mathbf{a}_t$ binary, and $\sum_{j=1}^{p} a_{jt} = 1$, so exactly one element is 1

Allocating points to leaves



- Each point has to be assigned to a leaf $t \in \mathcal{L}$ according to the splits:
  - Binary variables $z_{it} = 1$ if point i assigned to leaf t, 0 otherwise
  - $\sum_{t \in \mathcal{L}} z_{it} = 1$ to ensure each point is assigned to a leaf
- Enforce splitting rules

$$\mathbf{a}_m^T\mathbf{x}_i + \epsilon \leq b_m + M\left(1 - z_{it}\right), \forall \text{ left-branch ancestors } m \text{ of } t$$
$$\mathbf{a}_m^T\mathbf{x}_i \geq b_m - M\left(1 - z_{it}\right), \forall \text{ right-branch ancestors } m \text{ of } t$$

Calculating misclassification

- For each leaf node, the best class to assign is the most common label among points assigned to that node
- Use N_{kt} to count the points of each label k in leaf t:

$$N_{kt} = \sum_{i:y_i=k} z_{it}$$

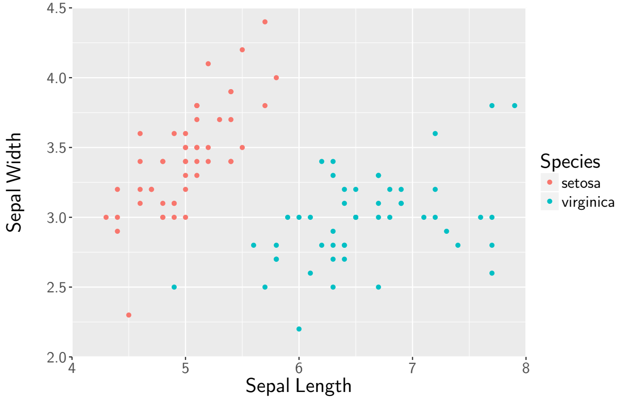
- Set $N_t = \sum_k N_{kt}$ to be the number of points in each leaf
- The misclassification error, L_t, is the number of points in the leaf that do not belong to the most common class

$$L_t = N_t - \max_k \{N_{kt}\} = \min_k \{N_t - N_{kt}\}$$

- Linearize with binary variables
- Objective is to minimize sum of misclassifications L_t

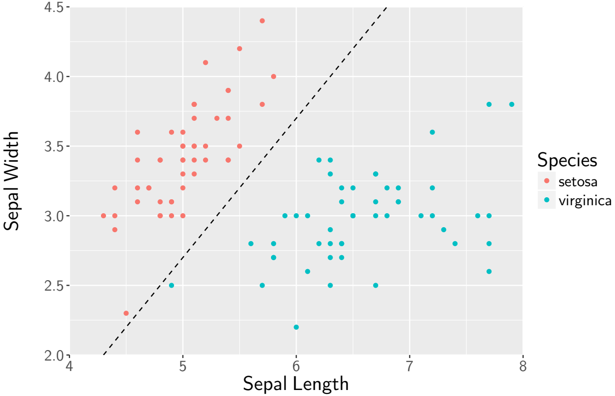
Extension—Hyperplane Splits

- CART and other methods require splits to be parallel to axes
- Using hyperplane splits can be more natural
  - Referred to as **oblique decision trees**
  - No good way to construct these in the literature: exponentially many hyperplanes to consider



Extension—Hyperplane Splits

- CART and other methods require splits to be parallel to axes
- Using hyperplane splits can be more natural
  - Referred to as **oblique decision trees**
  - No good way to construct these in the literature: exponentially many hyperplanes to consider



Modifying formulation to incorporate hyperplane splits

- Our previous constraints on the splits were:

$$\sum_{j=1}^{p} a_{jt} = 1, \quad \forall t \in \mathcal{B}$$

$$a_{jt} \in \{0,1\}, \quad j = 1, \ldots, p, \ \forall t \in \mathcal{B}$$

- We can simply relax integrality on the a_{jt} and replace sum with norm:

$$\|\mathbf{a}_t\|_1 \leq 1, \quad \forall t \in \mathcal{B}$$

- Choose 1-norm because we can linearize it easily

Local search heuristic for Optimal Trees

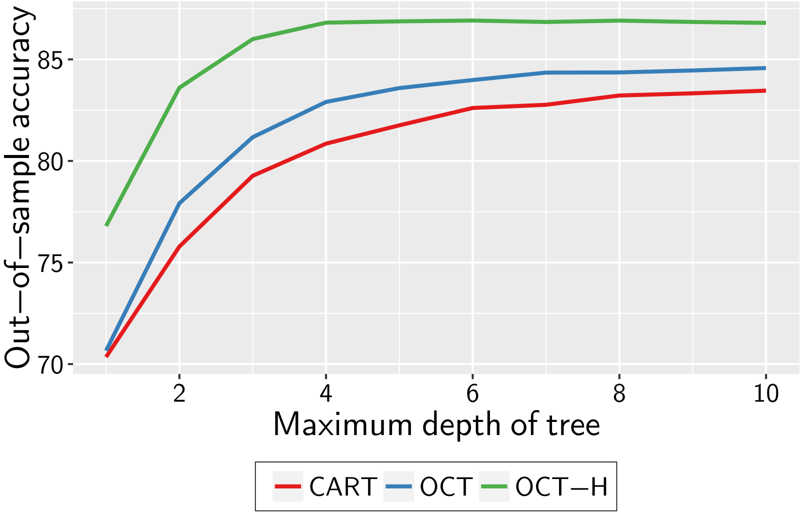
- We have developed a heuristic local search procedure with random restarts to efficiently optimize over this reduced search space
- Overview of one local search iteration:
  - Choose node in the tree at random
  - Re-optimize split at this node so it is locally optimal
    - If improved, exit and start local search iteration on new tree
  - If no split can be improved, terminate
- Repeat local search iterations until no improvements found
- Use different starting trees as random restarts

Real-world datasets

- 60 datasets from UCI Machine Learning Repository
  - Wide range of values for n (50–245,000), p (2–500) and K (2–10)
- For each dataset:
  - Split data into training and testing sets (75/25)
  - Train model on training and report error on test set
- Repeated five times for each dataset and results averaged
  - Minimizes the effect of any particular training/validation/test split
- Carried out for CART, OCT, and OCT-H
- State-of-the-art methods: Random Forest, XGBoost

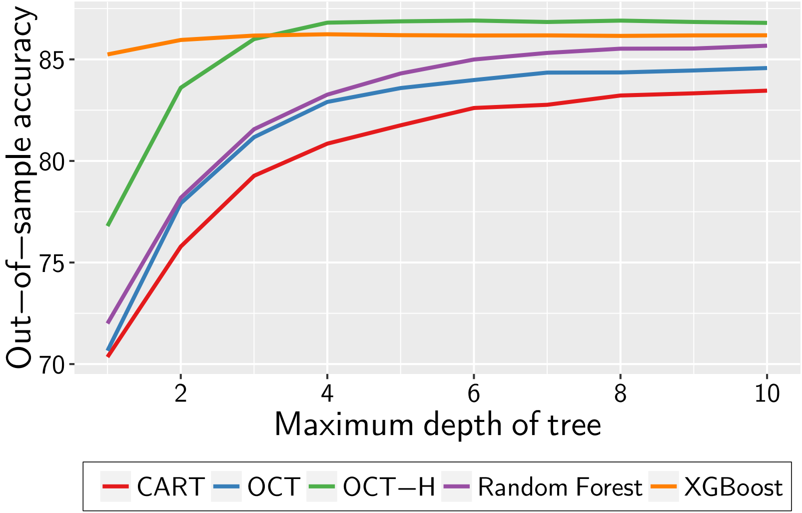
Optimal Classification Trees with Hyperplanes

- Average out-of-sample accuracy across all 60 datasets:



Optimal Classification Trees with Hyperplanes

- Average out-of-sample accuracy across all 60 datasets:



Improved triaging for children after head trauma, B.+Dunn+Trikalinos+Wang, 2017

- Most children with head injury have apparently minor trauma.
- Main challenge is to identify a clinically important traumatic brain injury (TBI) that necessitates immediate intervention.
- Computed tomography (CT) is the standard for rapid diagnosis of intracranial injury,
- A CT scan has more than 60 times the radiation of an X-ray.
- Pediatric Emergency Care Applied Research Network (PECARN), Lanchet, 2009 has developed and validated rules for triaging which children with head trauma to do CT scan based on CART.

Results

- Data EMR from 42412 children.
- We used OCT and compared with CART (PECARN study).
- Out of 8574 children < 2 years old, CART selected 4120 to do CT scans, identified 80 correctly and missed 1.
- Out of 8574 children < 2 years old, OCT selected 2380 to do CT scans, identified 80 correctly and missed 1.
- Out of 25355 children > 2 years old, CART selected 11866 to do CT scans, identified 237 correctly and missed 7.
- Out of 25355 children > 2 years old, OCT selected 9251 to do CT scans, identified 241 correctly and missed 3.

Prediction of mortality in ED

- Data: 380,000 emergency surgeries in 600 hospitals across the US from 2007-2014
- Task: Predicting risk of mortality within 30 days following surgery
- Use trees for interpretability so that we can help surgeons understand the risk factors
- Current state-of-the-art is CART: 83% AUC out-of-sample
- Optimal Trees: 92% AUC out-of-sample.

Conclusions

- Central problems in ML/S considered intractable a generation ago are now tractable via modern optimization.
- Given the astonishing developments in MIO, we need to revisit our core beliefs on computational complexity.
- ML/S traditionally linked to Probability; Especially in data rich environments, ML/S is more naturally linked to Optimization.
- Statistics departments need to rethink their educational offerings and the link to optimization.