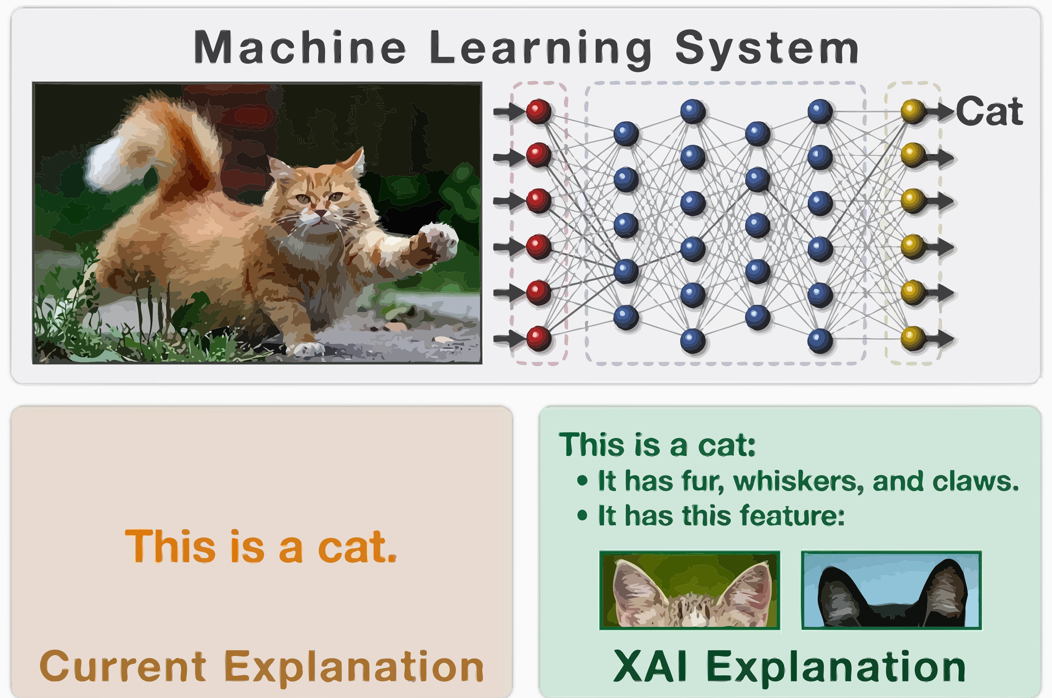
Using MaxSAT for Efficient Explanations of Tree Ensembles

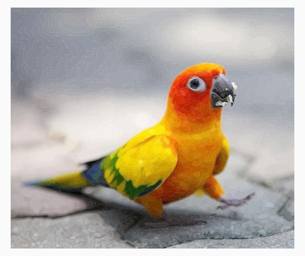
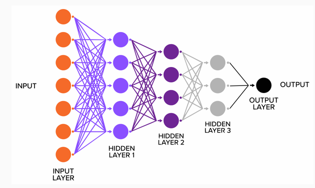
Alexey Ignatiev[1], Yacine Izza[2], Peter J. Stuckey[1], Joao Marques-Silva[3]

[1]Monash University, Melbourne, Australia [2]University of Toulouse, France [3]IRIT, CNRS, Toulouse, France

eXplainable AI



Why? Status Quo

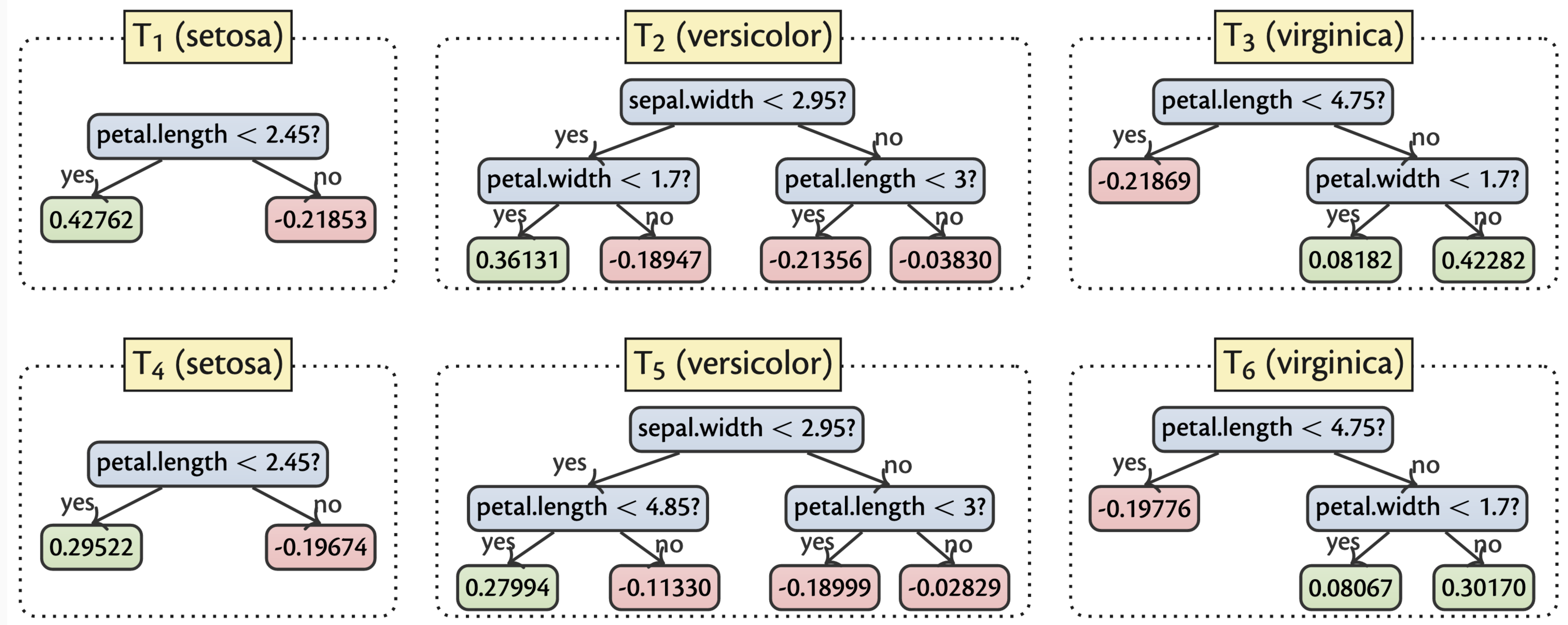
	A parrot	Machine learning algorithm
		
Learns random phrases	✓	✓
Doesn't understand s**t about what it learns	✓	✓
Occasionally speaks nonsense	✓	✓

Abductive Explanations and Boosted Trees

classifier $\tau : \mathbb{F} \rightarrow \mathcal{K}$,
instance $\mathbf{v}$ **s.t.** $\tau(\mathbf{v}) = c$

abductive explanation $\mathcal{X}$:
$\forall(\mathbf{x} \in \mathbb{F}).\bigwedge_{j\in\mathcal{X}}(x_j = v_j) \rightarrow (\tau(\mathbf{x}) = c)$

- $w(\mathbf{x}, c) = \sum_{j\in\{0,\ldots,n-1\}} \mathcal{T}_{Kj+c}(\mathbf{x}),\ c \in [K]$
- $\tau(\mathbf{x}) = \arg\max_{c\in[K]} w(\mathbf{x}, c)$



Idea and Contributions

SMT

- reach logic, can handle *linear constraints*
- directly reason about:

$$\mathcal{H} \wedge \left(\sum_\iota \geq \sum_i\right)$$

- high decimal point precision

MaxSAT

maximize $\sum_\iota - \sum_i$
subject to $\mathcal{H}$

- pure propositional logic
  - prediction as objective function
  - weighted soft clauses keep precision
  - core-guided MaxSAT
- incremental MaxSAT calls
  - *MiniSat-like assumptions interface!*
  - unsatisfiable core *reuse*
- novel weight stratification
- early termination

Encoding BT Operation

$\mathcal{F}$ — **set of features** $\quad \forall_{j\in\mathcal{F}}\ D_j$ — **domain of feature** j $\quad \mathfrak{E}$ — **tree ensemble**

$$\mathcal{H} = \bigwedge_{j\in\mathcal{F}} \mathcal{H}_{D_j} \wedge \bigwedge_{T_i\in\mathfrak{E}} \mathcal{H}_{T_i}$$

$\mathcal{H}_{D_j}$ **encodes feature domain** D_j

- feature **threshold values** $s_{j,k}$ from $\mathfrak{E}$
- variable $o_{j,k} = 1$ iff $x_j < s_{j,k}$
- variable $l_{j,k} = 1$ iff $x_j \in [s_{j,k-1}, s_{j,k})$
- **order encoding** of D_j with $l_{j,k}$ and $o_{j,k}$
- instance is expressed by $l_{j,k}$-variables

$\mathcal{H}_{T_i}$ **encodes paths of** T_i

- $\mathcal{P}_i$ is the set of paths of T_i
- given a $P_r \in \mathcal{P}_i$, t_r denotes its leaf
- encode each path $P_r \in \mathcal{P}_i$:

$$\left(\bigwedge_{(x_j<s_{j,k})\in P_r} o_{j,k} \wedge \bigwedge_{(x_j\geq s_{j,k})\in P_r} \neg o_{j,k}\right) \leftrightarrow t_r$$

- $\sum_{P_r\in\mathcal{P}_i} t_r = 1$

"Optimizing" Model Predictions

AXp condition for $\mathfrak{X}$

$$\forall(\mathbf{x} \in \mathbb{F}).\bigwedge_{j\in\mathfrak{X}}(x_j = v_j) \rightarrow (\tau(\mathbf{x}) = c_i)$$

weighted sums $\sum_\iota$ and $\sum_i$

$\mathfrak{X}$ is not AXp if $\exists_{\mathbf{x}\in\mathbb{F}, c_\iota\neq c_i}.\ w_\iota(\mathbf{x}) \geq w_i(\mathbf{x})$

even to represent in propositional logic! — hard to reason about

$\mathfrak{X}$ is not AXp if $\mathcal{S}^*_{i,\iota} \geq 0$

maximize $\mathcal{S}_{i,\iota} = \sum_\iota - \sum_i$
subject to $\mathcal{H} \wedge \bigwedge_{j\in\mathfrak{X}} [\![x_j = v_j]\!]$

weighted soft clauses + propositional variables only

Computing an AXp

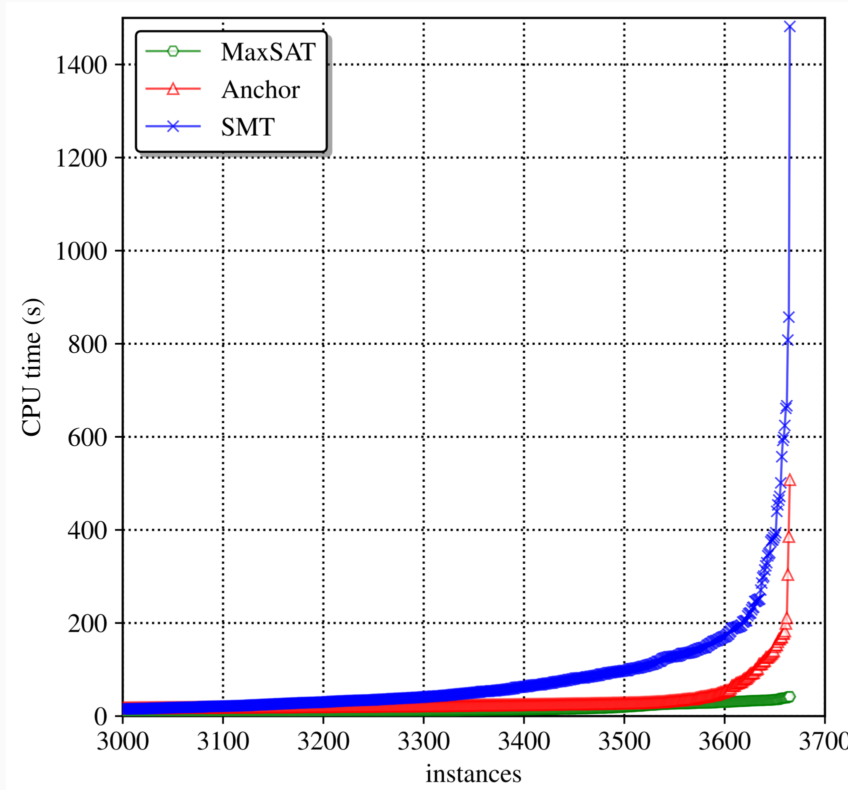
Function EntCheck($\langle\mathcal{H},\mathcal{S}\rangle$, $\mathbf{v}$, c_i, $\mathcal{X}$)
  Input: $\mathcal{H}$: Hard clauses, $\mathcal{S}$: Objective functions,
    $\mathbf{v}$: Input instance, c_i: Prediction, i.e. $\tau(\mathbf{v}) = c_i$,
    $\mathcal{X}$: Candidate explanation
  Output: true or false
  foreach $\mathcal{S}_{i,\iota} \in \mathcal{S}$: *# all relevant objective functions for c_i*
    $\mu \leftarrow$ MaxSAT($\mathcal{H} \wedge \bigwedge_{j\in\mathcal{X}} l[\![x_j = v_j]\!], \mathcal{S}_{i,\iota}$)
    if ObjValue(μ) ≥ 0: *# non-negative objective?*
      return false *# misclassification reached*
  return true *# $\mathcal{X}$ indeed entails prediction c_i*

Function ExtractAXp($\mathfrak{E}$, $\mathbf{v}$, c_i)
  Input: $\mathfrak{E}$: TE computing $\tau(\mathbf{x})$, $\mathbf{v}$: Input instance,
    c_i: Prediction, i.e. $\tau(\mathbf{v}) = c_i$
  Output: $\mathcal{X}$: abductive explanation
  $\langle\mathcal{H},\mathcal{S}\rangle \leftarrow$ Encode($\mathfrak{E}$) *# MaxSAT encoding s.t. $\mathcal{S} \triangleq \cup\mathcal{S}_{i,\iota}$*
  $\mathcal{X} \leftarrow \mathcal{F}$ *# $\mathcal{X}$ is over-approximation*
  foreach $j \in \mathcal{X}$:
    if EntCheck($\langle\mathcal{H},\mathcal{S}\rangle$, $\mathbf{v}$, c_i, $\mathcal{X} \setminus \{j\}$): *# j unneeded?*
      $\mathcal{X} \leftarrow \mathcal{X} \setminus \{j\}$ *# If so, drop it*
  return $\mathcal{X}$ *# $\mathcal{X}$ is AXp*

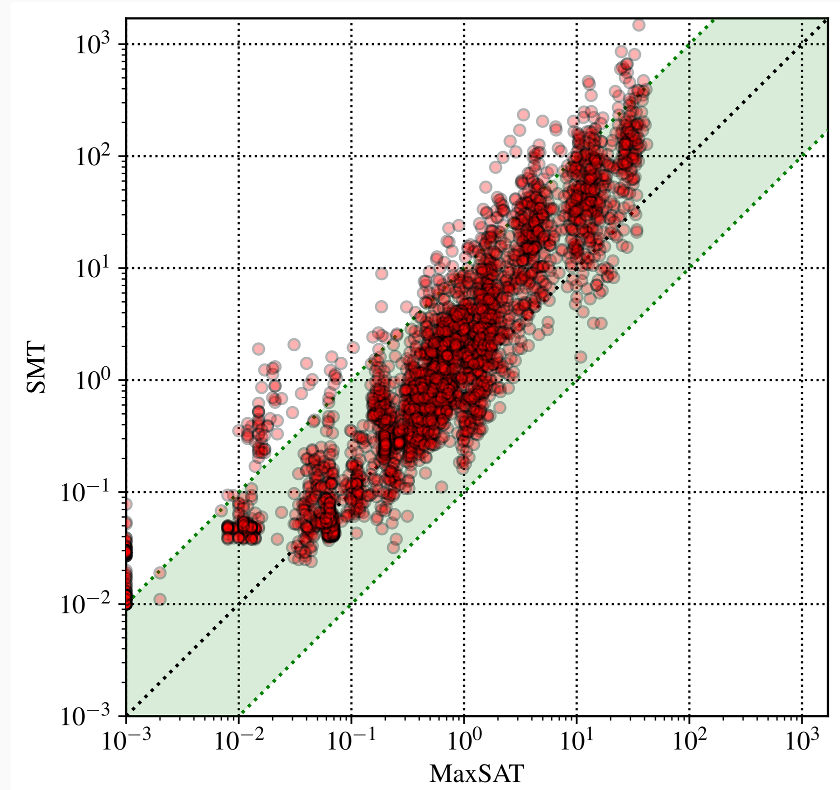
Incremental Core-Guided MaxSAT Solver

Function MaxSAT(ϕ, $\mathcal{A}$)
  Input: ϕ: Partial CNF formula,
    $\mathcal{A}$: Set of assumption literals
  Output: μ: MaxSAT model
  $cost \leftarrow 0$ *# initially, cost is 0*
  $C \leftarrow$ ValidCores(ϕ, $\mathcal{A}$) *# get valid unsatisfiable cores*
  foreach $\kappa \in C$: *# iterate over known cores κ*
    $cost \leftarrow cost +$ CoreWt(κ) *# add its weight to cost*
    $\phi \leftarrow$ Process(ϕ, κ) *# process κ and update ϕ*
  while SAT(ϕ, $\mathcal{A}$) = false: *# iterate until ϕ gets satisfiable*
    $\kappa \leftarrow$ GetCore(ϕ) *# new unsatisfiable core*
    $cost \leftarrow cost +$ CoreWt(κ) *# add its weight to cost*
    $\phi \leftarrow$ Process(ϕ, κ) *# process κ and update ϕ*
    Record(ϕ, $\mathcal{A}$, κ) *# record κ for the future*
  return GetModel(ϕ) *# ϕ is now satisfiable*

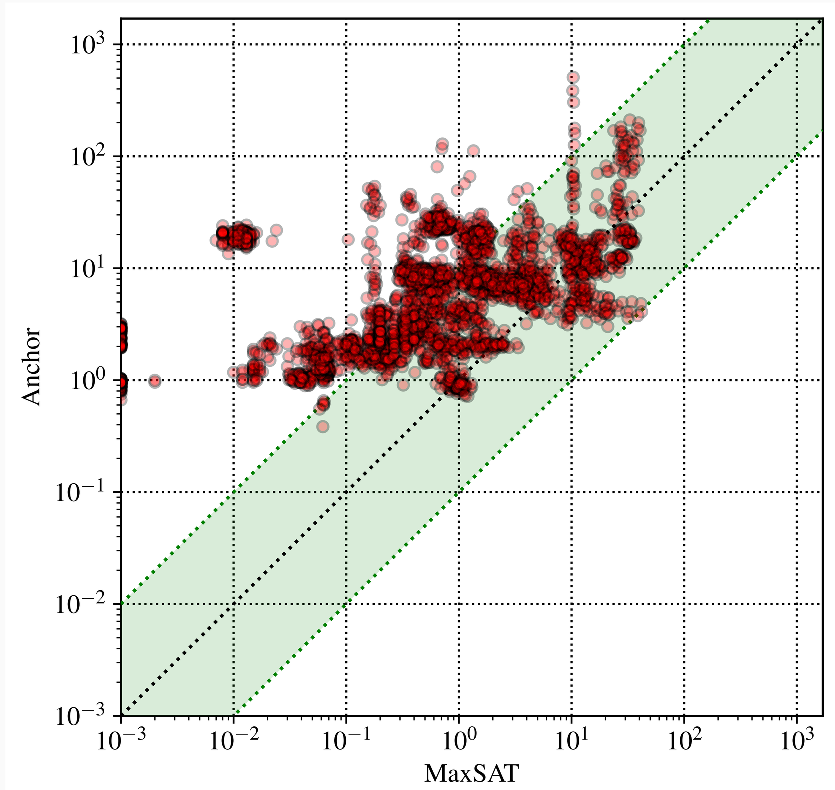
Experimental Results



all three competitors



MaxSAT vs. SMT



MaxSAT vs. Anchor

up to 2 orders of magnitude performance improvement

more robust than SMT and Anchor