Learning Tree-Based Models with Manifold Regularization: Alternating Optimization Algorithms

Arman Zharmagambetov
Advisor: Miguel Á. Carreira-Perpiñán

Electrical Engineering and Computer Science
University of California, Merced

November 21, 2022

Outline

- Motivation
- Preliminaries: Tree Alternating Optimization (TAO) algorithm
- Semi-supervised learning with decision trees
  - Optimization problem and its reformulation
  - Proposed algorithm and practicalities
  - Experiments
- Dimensionality reduction with decision trees
  - Optimization problem and its reformulation
  - Proposed algorithm and practicalities
  - Experiments
- Conclusion

Motivation: manifold regularization

- Recall that a common objective in machine learning is to minimize a certain loss for the given training dataset $\mathcal{D}$ to train a target model f:

$$\min_f \; L\left(f, \mathcal{D}\right) + \alpha \|f\|_l + \gamma \|f\|_I$$

  where blue terms are optional regularizes: $\alpha\|f\|_l$ is commonly referred as generalized Tikhonov regularization (e.g. weight decay), $\gamma\|f\|_I$ is **manifold regularization**.
- At a high level, manifold regularization exploits the geometry of $\mathcal{D}$ and smoothness of f to constrain the model that should be learned.
- A common assumption is that similar instances have similar predictions.

Motivation: manifold regularization

Several examples:

- Semi supervised learning

$$E(\mathbf{\Theta}) = \sum_{n=1}^{l} L\left(f(\mathbf{x}_n; \mathbf{\Theta}), y_n\right) + \gamma \sum_{n,m=1}^{N} w_{nm}(f(\mathbf{x}_n; \mathbf{\Theta}) - f(\mathbf{x}_m; \mathbf{\Theta}))^2.$$

- Nonlinear dimensionality reduction

$$E(\mathbf{Z}) = \sum_{n,m=1}^{N} \left(w_{nm} \|\mathbf{z}_n - \mathbf{z}_m\|^2 + \alpha e^{-\|\mathbf{z}_n - \mathbf{z}_m\|^2} \right)$$

Why Decision Trees?

- **Widespread usage** – successfully used as a standalone predictor [6] or as a building block for popular ensemble frameworks: XGBoost [9], Random Forest [3], etc.
- Part of the winning solutions in kaggle competitions:

  `https://www.kaggle.com/code/sudalairajkumar/winning-solutions-of-kaggle-competitions/notebook`
- Numerous successful use cases:
  - COVID-19 spread prediction as a time series analysis

    `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9251895/`
  - Financial organizations: `https://www.youtube.com/watch?v=fiSfB74yvlk`
  - Recommendation systems (e.g. ranking problems)
  - ...

Why Decision Trees?

- **Interpretability** – input follows a unique root-to-leaf path:

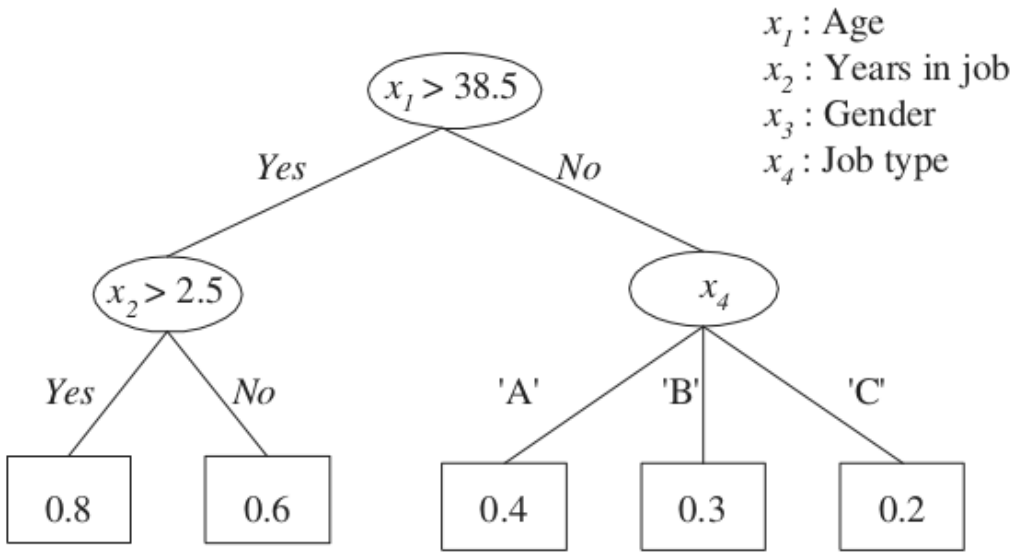


Figure: Example of a (hypothetical) decision tree (source: [1]).

Outline

- Motivation
- **Preliminaries: Tree Alternating Optimization (TAO) algorithm**
- Semi-supervised learning with decision trees
  - Optimization problem and its reformulation
  - Proposed algorithm and practicalities
  - Experiments
- Dimensionality reduction with decision trees
  - Optimization problem and its reformulation
  - Proposed algorithm and practicalities
  - Experiments
- Conclusion

Overview of Tree Alternating Optimization (TAO) algorithm

M. Carreira-Perpiñán, 2022; M. Carreira-Perpiñán and P. Tavallali, 2018

- Given a training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$
- $\mathbf{T}$: $\mathbb{R}^D \rightarrow \mathbb{R}$ – tree predictive mapping with parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{\text{nodes}}$
- Assume a tree structure $\mathbf{T}$ is given. Consider the problem:

$$E(\boldsymbol{\Theta}) = \sum_{n=1}^{N} L(\mathbf{y}_n, \mathbf{T}(\mathbf{x}_n; \boldsymbol{\Theta})) + \alpha \sum_{i \in \mathcal{N}} \phi(\boldsymbol{\theta}_i)$$

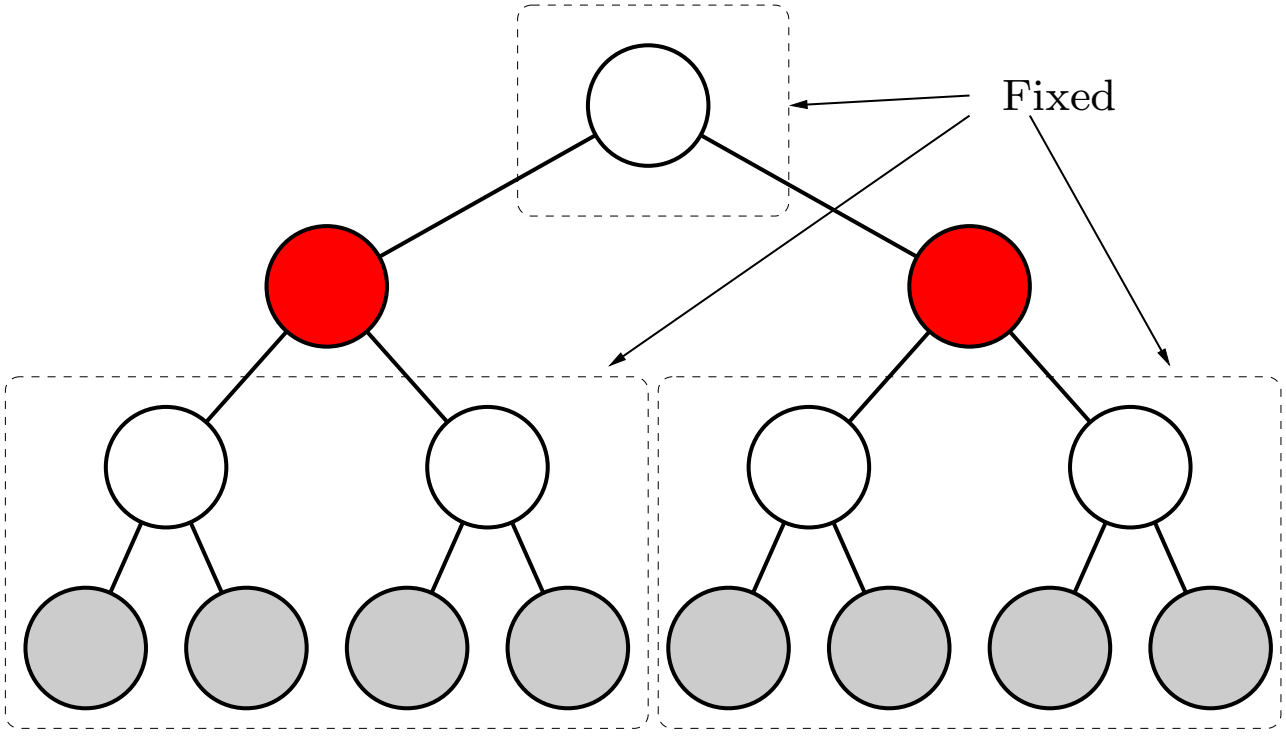
- Tree structure is fixed (as in neural nets) and we optimize over $\boldsymbol{\Theta}$. The problem is NP-hard [10]!
- **Separability condition** [6]: Consider any pair of nodes i and j. Fix the parameters of all other nodes ($\boldsymbol{\Theta}_{\text{rest}}$). If nodes i and j are not descendants of each other, then $E(\boldsymbol{\Theta})$ can be rewritten as:

$$E(\boldsymbol{\Theta}) = E_i(\boldsymbol{\theta}_i) + E_j(\boldsymbol{\theta}_j) + E_{\text{rest}}(\boldsymbol{\Theta}_{\text{rest}})$$

- i.e., non-descendant nodes can be optimized independently. Theorem extends beyond 2 nodes!

TAO: separability of nodes

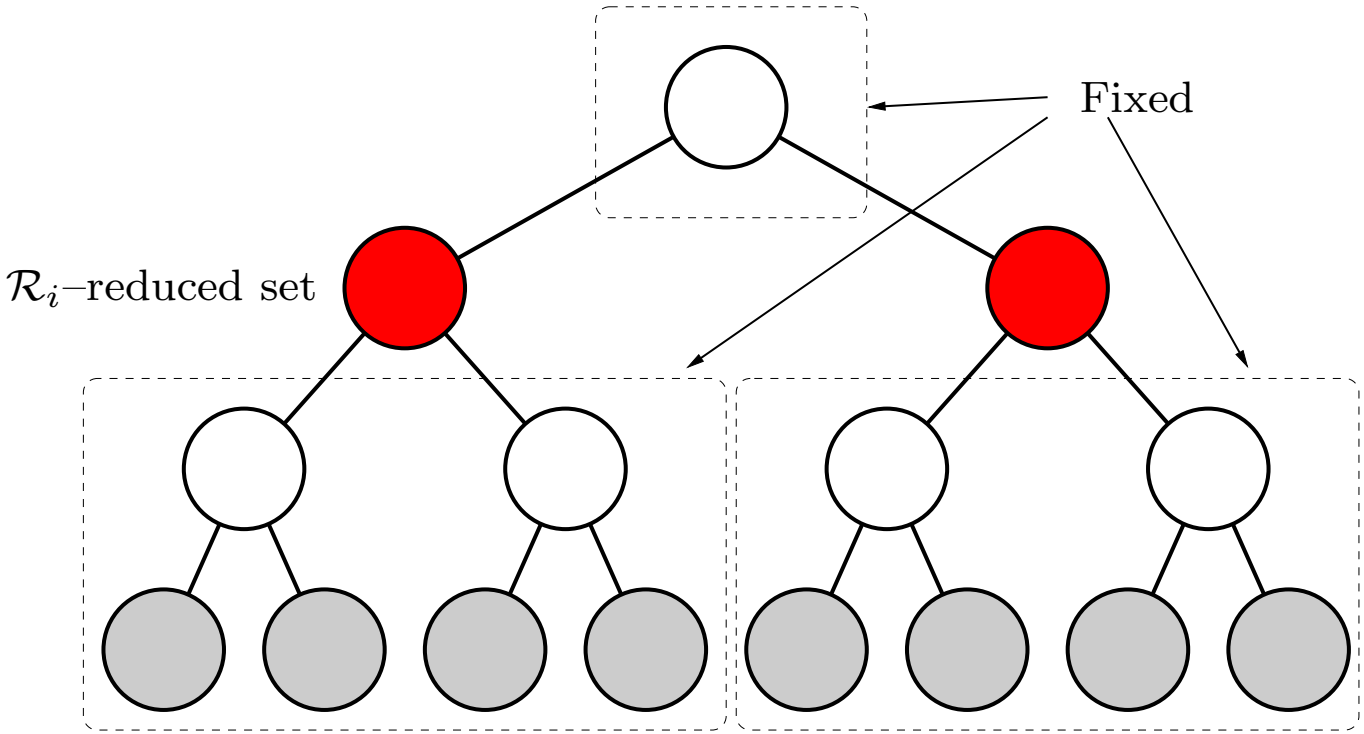
- Any set of non-descendant nodes of a tree can be optimized independently:



- Key idea: fix one part of the tree and optimize over another

TAO: separability of nodes

- Any set of non-descendant nodes of a tree can be optimized independently:



- Key idea: fix one part of the tree and optimize over another
- Initial $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{\text{nodes}}$ are random
- The reduced set $\mathcal{R}_i$ contains the training instances that reach node i.

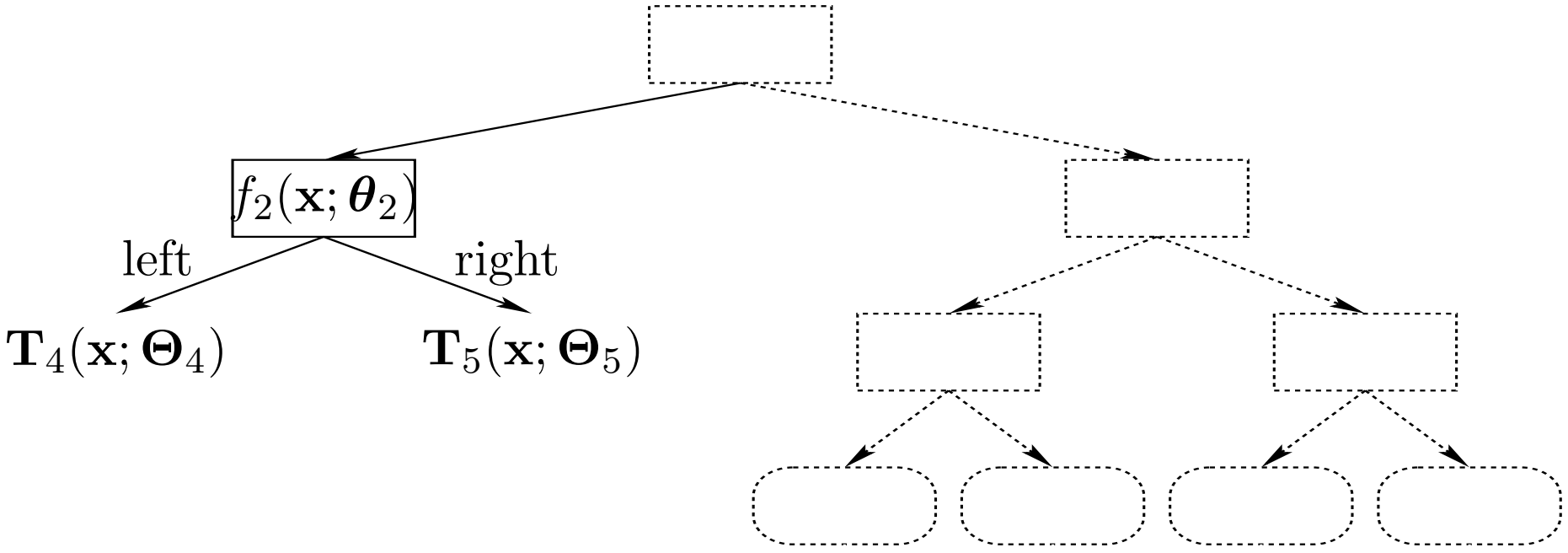
TAO: learning leaves

A set of non-descendant nodes are all the leaves. Learning the parameters of one leaf is given by the optimization of $E(\boldsymbol{\Theta})$ over $\boldsymbol{\theta}_i$:

$$\min_{\boldsymbol{\theta}_i} E_i(\boldsymbol{\theta}_i) = \sum_{n \in \mathcal{R}_i} L(\mathbf{y}_n, \mathbf{g}_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \alpha \, \phi_i(\boldsymbol{\theta}_i).$$

Each leaf i has a predictor function $\mathbf{g}_i(\mathbf{x}; \boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \mathbb{R}^K$ that produces the actual output. Therefore, solving the reduced problem over a leaf i amounts to fitting the leaf's predictor $\mathbf{g}_i$ to the instances in its reduced set to minimize the original loss (e.g. squared error).

TAO: learning internal nodes



$f_i(\mathbf{x};\boldsymbol{\theta}_i)$: $\mathbb{R}^D \rightarrow \{\texttt{left},\texttt{right}\}$ is a decision function in node i which sends instance $\mathbf{x}_n$ to the corresponding child of i.

Outline

- Motivation
- Preliminaries: Tree Alternating Optimization (TAO) algorithm
- **Semi-supervised learning with decision trees**
  - Optimization problem and its reformulation
  - Proposed algorithm and practicalities
  - Experiments
- Dimensionality reduction with decision trees
  - Optimization problem and its reformulation
  - Proposed algorithm and practicalities
  - Experiments
- Conclusion

SSL: motivation

- ML use is rapidly growing $\rightarrow$ as is the need for data labeling/annotation

SSL: motivation

- ML use is rapidly growing $\rightarrow$ as is the need for data labeling/annotation
- . . . but manually labeling data is expensive!

SSL: motivation

- ML use is rapidly growing $\rightarrow$ as is the need for data labeling/annotation
- . . . but manually labeling data is expensive!
- Unlabeled data are usually cheap and easy to get
- Unlabeled data contain useful information that can improve our model

SSL: motivation

- ML use is rapidly growing $\rightarrow$ as is the need for data labeling/annotation
- . . . but manually labeling data is expensive!
- Unlabeled data are usually cheap and easy to get
- Unlabeled data contain useful information that can improve our model
- **Semi-supervised learning** (SSL) seeks to train a machine learning model by leveraging a small percentage of labeled data and much larger sample of unlabeled data.

LapTAO: problem formulation

- Consider dataset $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u$: $\mathcal{D}_l = \{\mathbf{x}_n, y_n\}_{n=1}^{l}$ is the labeled data, $\mathcal{D}_u = \{\mathbf{x}_n\}_{n=l+1}^{N}$ is unlabeled data, and $l << N$.

LapTAO: problem formulation

- Consider dataset $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u$: $\mathcal{D}_l = \{\mathbf{x}_n, y_n\}_{n=1}^{l}$ is the labeled data, $\mathcal{D}_u = \{\mathbf{x}_n\}_{n=l+1}^{N}$ is unlabeled data, and $l << N$.
- T: $\mathbb{R}^D \rightarrow \mathbb{R}$ – tree predictive mapping with parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{\text{nodes}}$

LapTAO: problem formulation

- Consider dataset $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u$: $\mathcal{D}_l = \{\mathbf{x}_n, y_n\}_{n=1}^{l}$ is the labeled data, $\mathcal{D}_u = \{\mathbf{x}_n\}_{n=l+1}^{N}$ is unlabeled data, and $l << N$.
- T: $\mathbb{R}^D \rightarrow \mathbb{R}$ – tree predictive mapping with parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{\text{nodes}}$
- Then, our goal is to minimize the following regularized objective:

$$E(\boldsymbol{\Theta}) = \sum_{n=1}^{l} \left(T(\mathbf{x}_n; \boldsymbol{\Theta}) - y_n\right)^2 + \alpha\, \phi(\boldsymbol{\Theta}) + \gamma \sum_{n,m=1}^{N} w_{nm} \left(T(\mathbf{x}_n; \boldsymbol{\Theta}) - T(\mathbf{x}_m; \boldsymbol{\Theta})\right)^2$$

- w_{nm} are the elements of the similarity matrix $\mathbf{W}$ (i.e., neighborhood graph, affinity matrix) [2, 13]
- $\phi(\cdot)$ is the regularization penalty such as $\|\cdot\|_1$
- γ,α are regularization hyperparameters
- **Non-differentiable and non-convex problem** due to $T(\cdot)$!

LapTAO: constrained reformulation

- We apply the idea of MAC [7]: introduce a new variable z for each training instance and consider the constrained problem:

$$\min_{z_1,\ldots,z_N,\Theta} \sum_{n=1}^{l} (z_n - y_n)^2 + \alpha\ \phi(\Theta) + \gamma \sum_{n,m=1}^{N} w_{nm}(z_n - z_m)^2$$
$$\text{s.t.} \quad z_n = T(\mathbf{x}_n; \Theta) \quad n = 1, \ldots, N.$$

LapTAO: constrained reformulation

- We apply the idea of MAC [7]: introduce a new variable z for each training instance and consider the constrained problem:

$$\min_{z_1,\dots,z_N,\boldsymbol{\Theta}} \sum_{n=1}^{l} (z_n - y_n)^2 + \alpha\ \phi(\boldsymbol{\Theta}) + \gamma \sum_{n,m=1}^{N} w_{nm}(z_n - z_m)^2$$
$$\text{s.t.} \quad z_n = T(\mathbf{x}_n; \boldsymbol{\Theta}) \quad n = 1, \dots, N.$$

- Denote $\mathbf{y} = [y_1, y_2, \dots, y_l, 0, 0, \dots]^T \in \mathbb{R}^N$, as the augmented ground truth vector
- Introduce diag matrix $\mathbf{J} = \text{diag}(1, \dots, 1, 0, \dots, 0) \in \mathbb{R}^{N \times N}$ with the first l diagonal entries equal to 1 and the rest 0
- Denote $\mathbf{z} = [z_1, \dots, z_N]^T$ and $\mathbf{t}(\mathbf{X}; \boldsymbol{\Theta}) = [T(\mathbf{x}_1; \boldsymbol{\Theta}), \dots, T(\mathbf{x}_N; \boldsymbol{\Theta})]^T$ where $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$
- Introduce graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$ where diagonal matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ with entries $d_{nn} = \sum_{m=1}^{N} w_{nm}$

LapTAO: constrained reformulation

- We apply the idea of MAC [7]: introduce a new variable z for each training instance and consider the constrained problem:

$$\min_{z_1,\ldots,z_N,\boldsymbol{\Theta}} \sum_{n=1}^{l} (z_n - y_n)^2 + \alpha\ \phi(\boldsymbol{\Theta}) + \gamma \sum_{n,m=1}^{N} w_{nm}(z_n - z_m)^2$$
$$\text{s.t.} \quad z_n = T(\mathbf{x}_n; \boldsymbol{\Theta}) \quad n = 1, \ldots, N.$$

- Can be rewritten as:

$$\min_{\mathbf{z},\boldsymbol{\Theta}} (\mathbf{z} - \mathbf{y})^T \mathbf{J}\ (\mathbf{z} - \mathbf{y}) + \alpha\ \phi(\boldsymbol{\Theta}) + \gamma\ \mathbf{z}^T \mathbf{L}\ \mathbf{z}$$
$$\text{s.t.} \quad \mathbf{z} = \mathbf{t}(\mathbf{X}; \boldsymbol{\Theta}).$$

LapTAO: solution

- Apply *augmented Lagrangian* [11] which defines a new, unconstrained optimization problem:

$$\min_{\mathbf{z},\boldsymbol{\Theta}} (\mathbf{z}-\mathbf{y})^T \mathbf{J}\, (\mathbf{z}-\mathbf{y}) + \alpha\, \phi(\boldsymbol{\Theta}) + \gamma\, \mathbf{z}^T \mathbf{L}\, \mathbf{z} - \boldsymbol{\lambda}^T (\mathbf{z}-\mathbf{t}(\mathbf{X};\boldsymbol{\Theta})) + \mu \|\mathbf{z}-\mathbf{t}(\mathbf{X};\boldsymbol{\Theta})\|^2$$

- $\boldsymbol{\lambda} \in \mathbb{R}^N$ are the estimates of Lagrange multipliers. Optimizing this for fixed $\mu > 0$ produces the sequence of $(\mathbf{z}_\mu, \mathbf{t}_\mu(\mathbf{X};\boldsymbol{\Theta}))$ and as $\mu \to \infty$, we force the minimizer to be in the feasible region for the constrained problem.

LapTAO: solution

- Apply *augmented Lagrangian* [11] which defines a new, unconstrained optimization problem:

$$\min_{\mathbf{z},\boldsymbol{\Theta}} (\mathbf{z}-\mathbf{y})^T \mathbf{J} (\mathbf{z}-\mathbf{y}) + \alpha \phi(\boldsymbol{\Theta}) + \gamma \mathbf{z}^T \mathbf{L} \mathbf{z} - \boldsymbol{\lambda}^T (\mathbf{z}-\mathbf{t}(\mathbf{X};\boldsymbol{\Theta})) + \mu \|\mathbf{z}-\mathbf{t}(\mathbf{X};\boldsymbol{\Theta})\|^2$$

- $\boldsymbol{\lambda} \in \mathbb{R}^N$ are the estimates of Lagrange multipliers. Optimizing this for fixed $\mu > 0$ produces the sequence of $(\mathbf{z}_\mu, \mathbf{t}_\mu(\mathbf{X};\boldsymbol{\Theta}))$ and as $\mu \to \infty$, we force the minimizer to be in the feasible region for the constrained problem.
- Finally, we apply alternating optimization to minimize above objective over:
  - $\mathbf{z}$ a.k.a "Label–step"
  - $\mathbf{t}(\mathbf{X};\boldsymbol{\Theta})$ a.k.a "Tree–step"

Label–step: optimizing over $\mathbf{z}$ given fixed $\mathbf{t}(\mathbf{X};\boldsymbol{\Theta})$

- The objective is a **quadratic** function and minimizer is obtained by solving the linear system:

$$\min_{\mathbf{z}} (\mathbf{z}-\mathbf{y})^T\mathbf{J}\ (\mathbf{z}-\mathbf{y}) + \gamma\ \mathbf{z}^T\mathbf{L}\ \mathbf{z} - \boldsymbol{\lambda}^T(\mathbf{z}-\mathbf{t}(\mathbf{X};\boldsymbol{\Theta})) + \mu\|\mathbf{z}-\mathbf{t}(\mathbf{X};\boldsymbol{\Theta})\|^2 \Rightarrow$$

$$\mathbf{A}\mathbf{z} = \mathbf{J}\mathbf{y} + \mu\mathbf{t}(\mathbf{X};\boldsymbol{\Theta}) + \frac{1}{2}\boldsymbol{\lambda}$$

Label–step: optimizing over $\mathbf{z}$ given fixed $\mathbf{t}(\mathbf{X};\boldsymbol{\Theta})$

- The objective is a **quadratic** function and minimizer is obtained by solving the linear system:

$$\min_{\mathbf{z}} (\mathbf{z}-\mathbf{y})^T\mathbf{J}\ (\mathbf{z}-\mathbf{y}) + \gamma\ \mathbf{z}^T\mathbf{L}\ \mathbf{z} - \boldsymbol{\lambda}^T(\mathbf{z}-\mathbf{t}(\mathbf{X};\boldsymbol{\Theta})) + \mu\|\mathbf{z}-\mathbf{t}(\mathbf{X};\boldsymbol{\Theta})\|^2 \Rightarrow$$

$$\mathbf{A}\mathbf{z} = \mathbf{J}\mathbf{y} + \mu\mathbf{t}(\mathbf{X};\boldsymbol{\Theta}) + \frac{1}{2}\boldsymbol{\lambda}$$

- $\mathbf{A} = \mathbf{J} + \mu\mathbf{I} + \gamma\mathbf{L}$ is a positive definite matrix. Moreover, $\mathbf{A}$ is a **sparse matrix** if graph Laplacian $\mathbf{L}$ is sparse.
- The "label–step" can be interpreted as "approximating" the labels (for $\mathcal{D}_u$) using the graph Laplacian and predictions obtained from the current tree (i.e., label smoothing or label propagation)

Tree–step: optimizing over $\boldsymbol{\Theta}$ given fixed $\mathbf{z}$

- The problem reduces to a regression fit of a tree:

$$\min_{\boldsymbol{\Theta}} \ \mu\|\mathbf{z} - \mathbf{t}(\mathbf{X};\boldsymbol{\Theta})\|^2 + \alpha\ \phi(\boldsymbol{\Theta}) - \boldsymbol{\lambda}^T(\mathbf{z} - \mathbf{t}(\mathbf{X};\boldsymbol{\Theta})) \Leftrightarrow$$

$$\min_{\boldsymbol{\Theta}} \left\| \left(\mathbf{z} - \frac{1}{2\mu}\boldsymbol{\lambda}\right) - \mathbf{t}(\mathbf{X};\boldsymbol{\Theta}) \right\|^2 + \frac{\alpha}{\mu}\phi(\boldsymbol{\Theta}).$$

- using $(\mathbf{z} - \frac{1}{2\mu}\boldsymbol{\lambda})$ as labels

Tree–step: optimizing over $\boldsymbol{\Theta}$ given fixed $\mathbf{z}$

- The problem reduces to a regression fit of a tree:

$$\min_{\boldsymbol{\Theta}} \ \mu\|\mathbf{z} - \mathbf{t}(\mathbf{X}; \boldsymbol{\Theta})\|^2 + \alpha\ \phi(\boldsymbol{\Theta}) - \boldsymbol{\lambda}^T(\mathbf{z} - \mathbf{t}(\mathbf{X}; \boldsymbol{\Theta})) \Leftrightarrow$$

$$\min_{\boldsymbol{\Theta}} \left\| \left(\mathbf{z} - \frac{1}{2\mu}\boldsymbol{\lambda} \right) - \mathbf{t}(\mathbf{X}; \boldsymbol{\Theta}) \right\|^2 + \frac{\alpha}{\mu}\phi(\boldsymbol{\Theta}).$$

- using $(\mathbf{z} - \frac{1}{2\mu}\boldsymbol{\lambda})$ as labels
- Intuitively, this step can be understood as fitting a tree with the current estimates of the labels

Tree–step: optimizing over $\boldsymbol{\Theta}$ given fixed $\mathbf{z}$

- The problem reduces to a regression fit of a tree:

$$\min_{\boldsymbol{\Theta}} \ \mu\|\mathbf{z} - \mathbf{t}(\mathbf{X};\boldsymbol{\Theta})\|^2 + \alpha\ \phi(\boldsymbol{\Theta}) - \boldsymbol{\lambda}^T(\mathbf{z} - \mathbf{t}(\mathbf{X};\boldsymbol{\Theta})) \Leftrightarrow$$

$$\min_{\boldsymbol{\Theta}} \left\| \left(\mathbf{z} - \frac{1}{2\mu}\boldsymbol{\lambda}\right) - \mathbf{t}(\mathbf{X};\boldsymbol{\Theta}) \right\|^2 + \frac{\alpha}{\mu}\phi(\boldsymbol{\Theta}).$$

- using $(\mathbf{z} - \frac{1}{2\mu}\boldsymbol{\lambda})$ as labels
- Intuitively, this step can be understood as fitting a tree with the current estimates of the labels
- Potentially, any decision tree learning algorithm can be applied: CART [4], C5.0 [12], etc.
- We solve this using Tree Alternating Optimization (TAO) algorithm: supports warm start, guarantees monotonic decrease of the above objective (for convergence) [14].

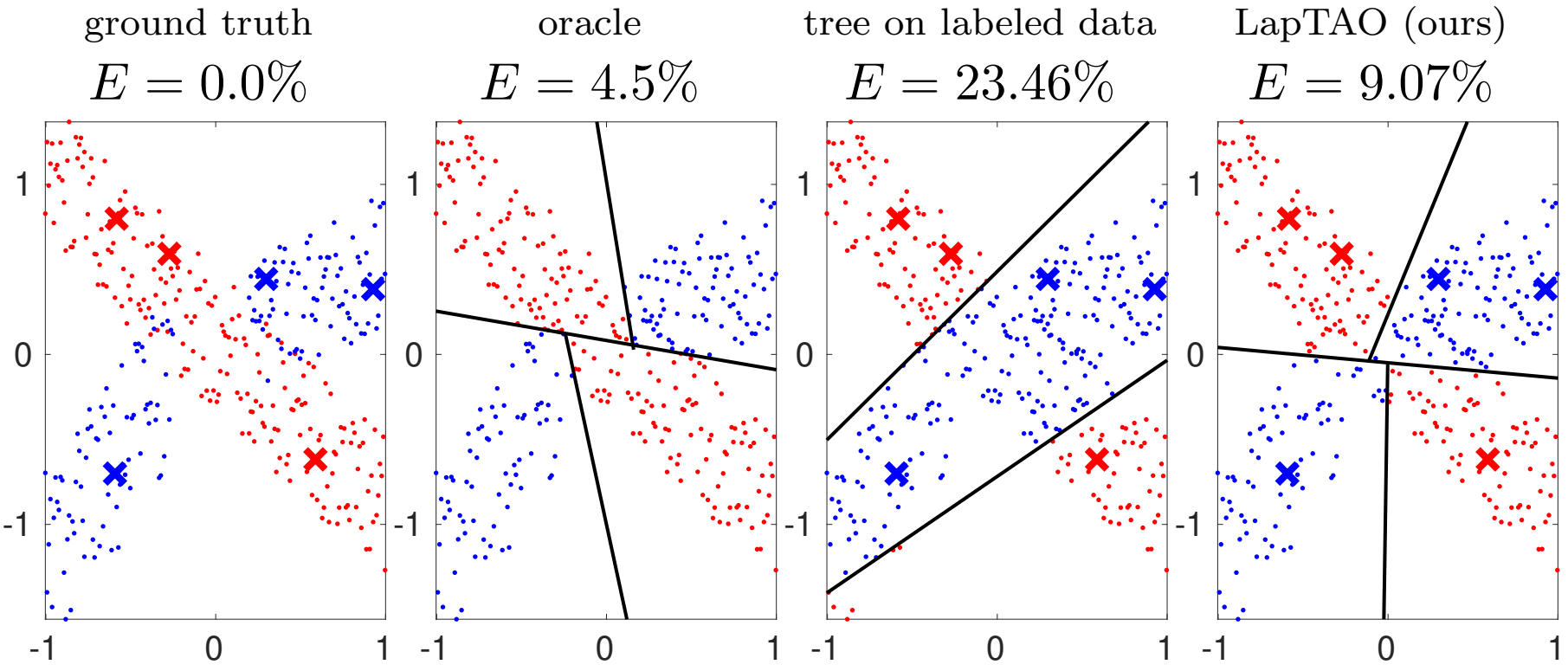
LapTAO: the final algorithm

input labeled set $\mathcal{D}_l = \{\mathbf{x}_n, y_n\}_{n=1}^{l}$
unlabeled set $\mathcal{D}_u = \{\mathbf{x}_n\}_{n=l+1}^{N}$;
penalty parameters: α, γ;
μ schedule: $\mu_0, \ldots, \mu_{\max}$;
graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$;
initialization:
$\boldsymbol{\lambda} \leftarrow \mathbf{0}$ (initialize Lagrange multipliers);
$\mathbf{z}_0 \leftarrow$ solve the "Label–step" with $\mu = 0$;
$\mathbf{t}(\cdot; \boldsymbol{\Theta}) \leftarrow$ fit a tree to $(\{\mathbf{x}_n\}_{n=1}^{N}, \mathbf{z}_0)$;
for $\mu = \mu_0 < \mu_1 < \mu_2 < \cdots < \mu_{\max}$;
"Label–step": $\mathbf{z} \leftarrow$ solve the linear system;
"Tree–step": $\mathbf{t}(\cdot; \boldsymbol{\Theta}) \leftarrow$ use TAO to fit the tree;
Lagrange multipliers step: $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \mu(\mathbf{z} - \mathbf{t}(\cdot; \boldsymbol{\Theta}))$;
end for
return $\mathbf{t}(\cdot; \boldsymbol{\Theta})$

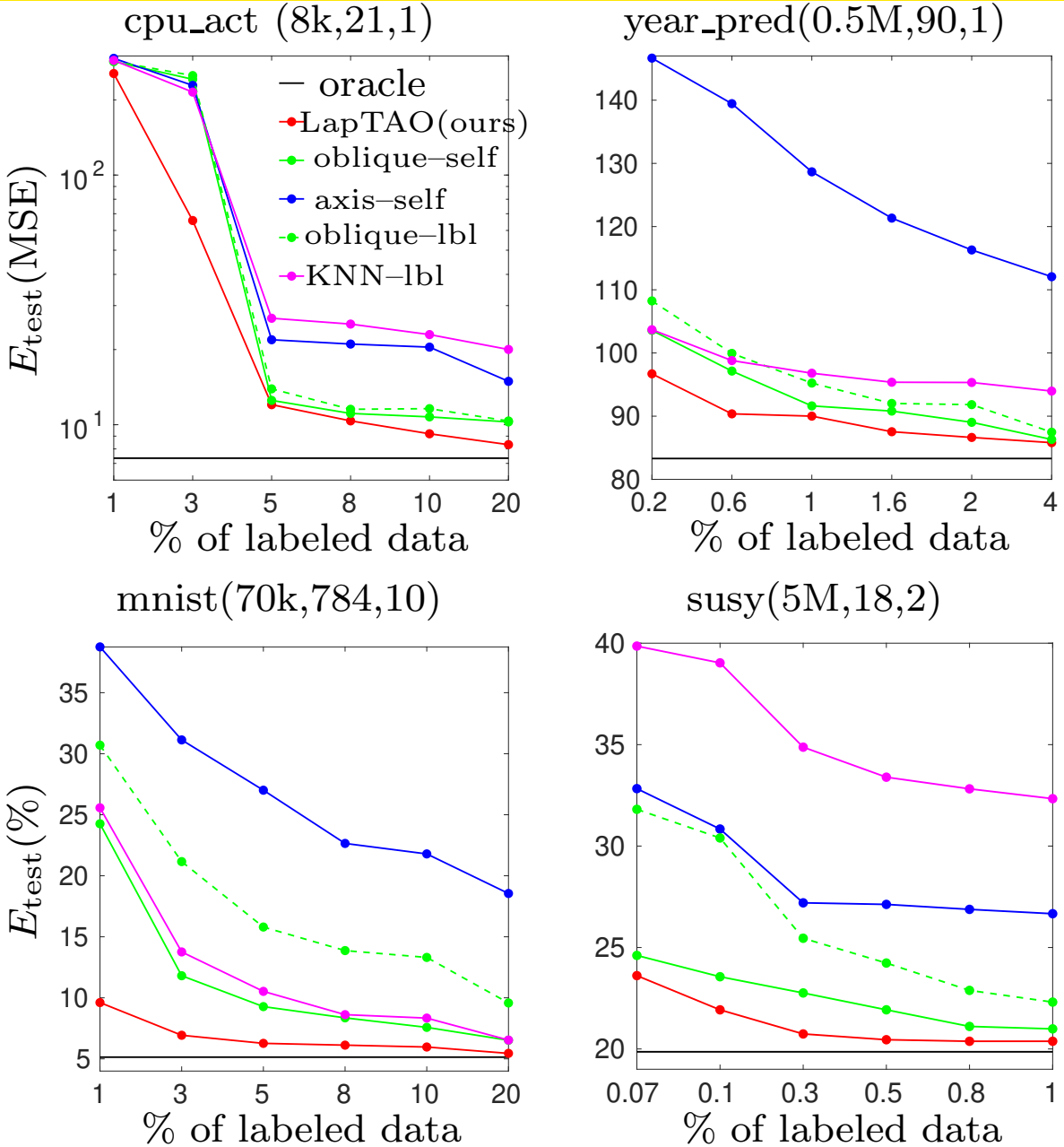
Experiments: setup

- As our main model, we consider oblique trees, having hyperplane decision functions "go to right if $\boldsymbol{\theta}_i^T \mathbf{x} \geq 0$". In this case, TAO uses LIBLINEAR to solve the linear binary classification problem at each decision node.
- Each leaf i outputs a constant value c_i
- We apply ℓ_1 penalty on tree node parameters to encourage sparsity and hyperparameter α controls the sparsity level.
- We use the fixed validation set (1% of train data) to explore hyperparameters: γ, α, etc.
- # of TAO iterations = 15, # of LapTAO iterations = 20.
- $\mu_0 = 0.001$ multiplied by 1.5 after each LapTAO iteration.
- $\mathbf{W}$ is obtained from Entropic affinities with perplexity of K.

Experiments: toy 2D

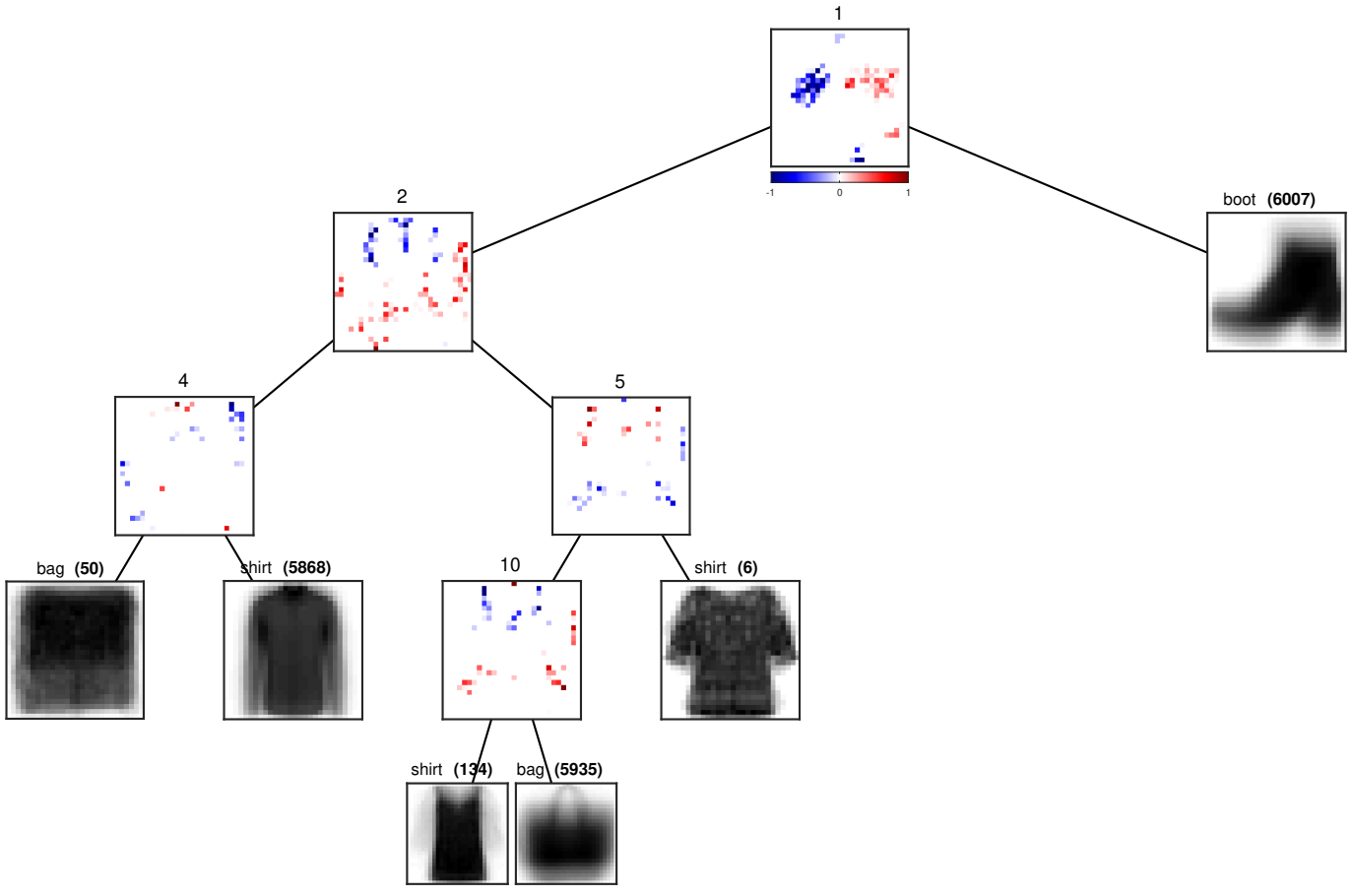


Experiments: performance



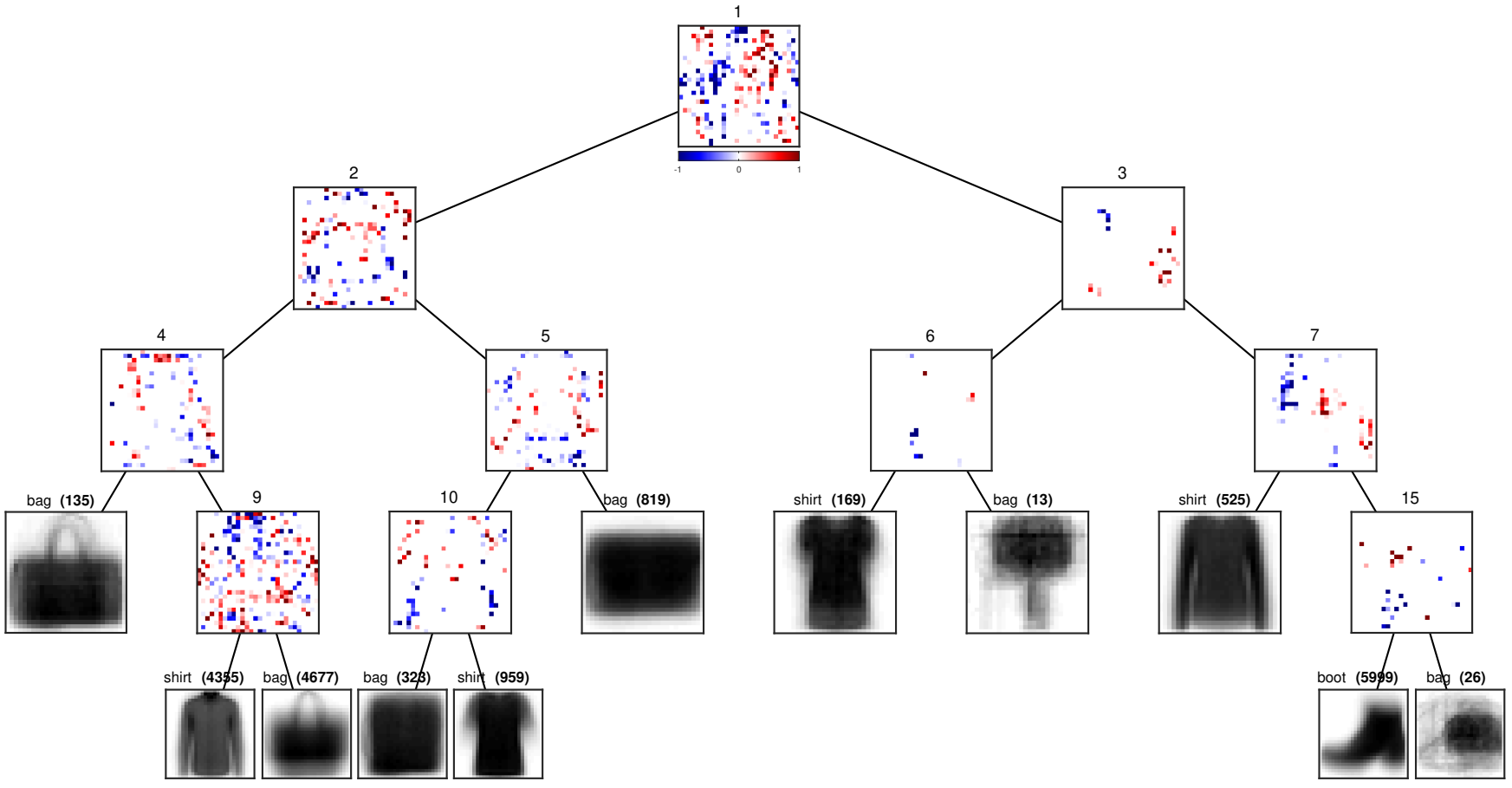
Experiments: interpretability

An example tree with $\alpha = 10$ and $E_{\text{test}} = 3.9\%$.



Experiments: interpretability

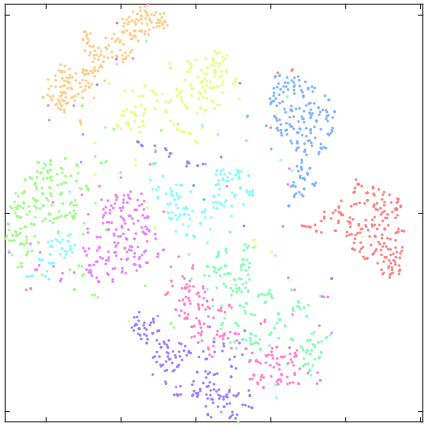
An example tree with $\alpha = 1$ and $E_{\text{test}} = 2.1\%$.



Outline

- Motivation
- Preliminaries: Tree Alternating Optimization (TAO) algorithm
- Semi-supervised learning with decision trees
  - Optimization problem and its reformulation
  - Proposed algorithm and practicalities
  - Experiments
- **Dimensionality reduction with decision trees**
  - Optimization problem and its reformulation
  - Proposed algorithm and practicalities
  - Experiments
- Conclusion

Overview



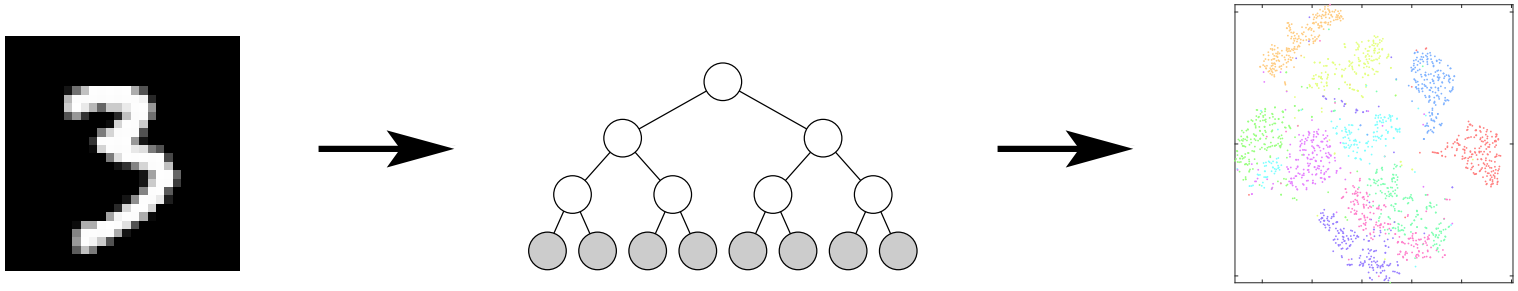
- Nonlinear embeddings (NLE), such as t-SNE, are widely used DR methods.
- Recall that such DR methods do not naturally define an out-of-sample mapping, rather they directly learn a low-dimensional projection for each training point.
- We consider the problem of learning interpretable out-of-sample mappings for NLE.

Overview

Why interpreting a projection mapping matters?

- Low-dimensional embeddings may not be a faithful projection of the original, high-dimensional data:
  1. The result depends in an obscure way on the objective function and on hyperparameters;
  2. The resulting embeddings may give a misleading view of the data, e.g. t-SNE has a strong tendency to find clusters where none exist [5];
- Augmenting the t-SNE embedding with an interpretable out-of-sample mapping allows one to understand how the high-dimensional input instances are projected to the embedding and understand whether that makes sense.

What mapping should we use?



- We argue for the use of sparse oblique decision trees as an out-of-sample mapping;
- Trees are considered to be interpretable models;
- Sparse oblique trees strike a good tradeoff between accuracy and interpretability which can be controlled via a hyperparameter.
- They can make full use of any and all features of an instance.

Jointly learning an optimal tree and embedding

Consider the elastic embedding objective function:

$$E(\mathbf{Z}) = \sum_{n,m=1}^{N} \left(w_{nm} \|\mathbf{z}_n - \mathbf{z}_m\|^2 + \alpha e^{-\|\mathbf{z}_n - \mathbf{z}_m\|^2} \right)$$

Call the resulting embeddings $\mathbf{z}$ the free embedding. If we want an out-of-sample mapping $\mathbf{F}$ so we can project new points, then $\mathbf{z} = \mathbf{F}(\mathbf{x})$ by definition and we have a parametric embedding objective function:

$$E(\mathbf{F}) = \sum_{n,m=1}^{N} \left(w_{nm} \|\mathbf{F}(\mathbf{x}_n) - \mathbf{F}(\mathbf{x}_m)\|^2 + \alpha e^{-\|\mathbf{F}(\mathbf{x}_n) - \mathbf{F}(\mathbf{x}_m)\|^2} \right) + \lambda \phi(\mathbf{F})$$

Not easy to optimize since $\mathbf{F}$ is non-differentiable and non-convex mapping (a tree)!

Jointly learning an optimal tree and embedding

- Solution: use the method of auxiliary coordinates (MAC) [7, 8]. Consider the following equivalent constrained problem with "auxiliary coordinates" $\mathbf{Z}$:

$$\min_{\mathbf{Z},\mathbf{F}} E(\mathbf{Z}) + \lambda\,\phi(\mathbf{F}) \quad \text{s.t.} \quad \mathbf{Z} = \mathbf{F}(\mathbf{X})$$

We solve this using a penalty method. We describe the quadratic penalty method for simplicity, but in the experiments we use the augmented Lagrangian. This defines a new, unconstrained objective function:

$$\min_{\mathbf{Z},\mathbf{F}} E(\mathbf{Z}) + \lambda\,\phi(\mathbf{F}) + \mu\|\mathbf{Z} - \mathbf{F}(\mathbf{X})\|^2. \tag{1}$$

Jointly learning an optimal tree and embedding

Finally, we optimize (1) by alternating optimization over $\mathbf{Z}$ and $\mathbf{F}$:

- Over $\mathbf{Z}$, eq. (1) is the original embedding objective E but with a quadratic regularization term on $\mathbf{Z}$:

$$\min_{\mathbf{Z}} E(\mathbf{Z}) + \mu \|\mathbf{Z} - \mathbf{F}(\mathbf{X})\|^2.$$

  Solution: off-the-shelf algorithm to optimize the original embedding (e.g. t-SNE) with a minor modification to handle the additional quadratic term.

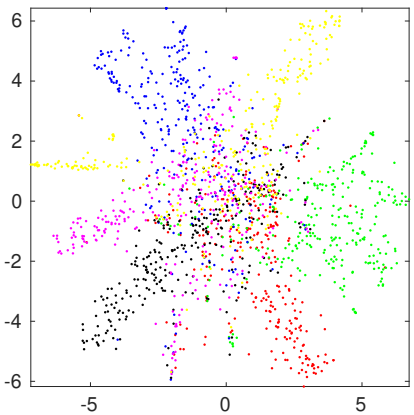
- Over $\mathbf{F}$, eq. (1) reduces to a regression fit of a tree which we solve using the Tree Alternating Optimization (TAO) [14]:

$$\min_{\mathbf{F}} \|\mathbf{Z} - \mathbf{F}(\mathbf{X})\|^2 + \frac{\lambda}{\mu}\phi(\mathbf{F})$$

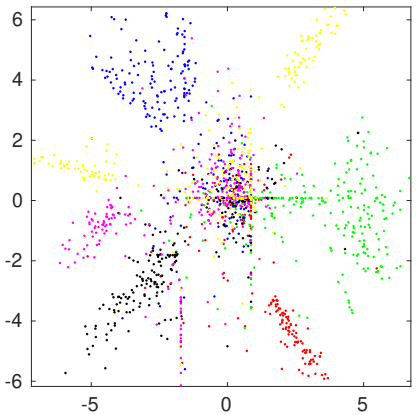
  The ability of the TAO algorithm to take an initial tree and improve over it is essential here to make sure that the step over $\mathbf{F}$ improves over the previous iteration, and to be able to use warm-start to speed up the computation.

Experiments

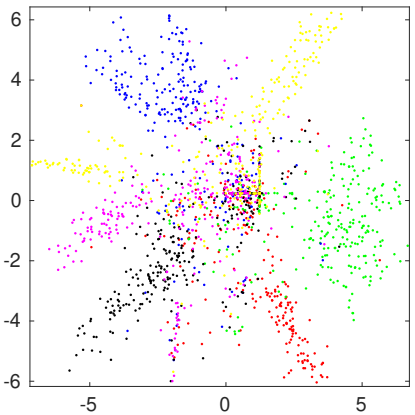
free embedding



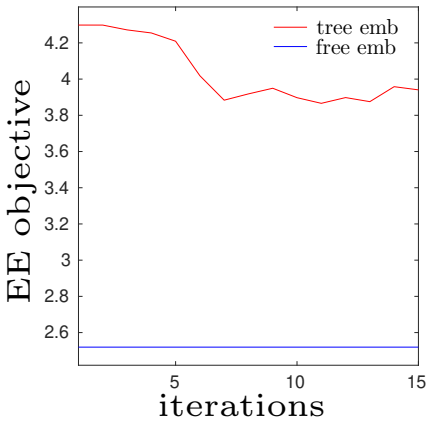
direct fit



tree embedding (ours)

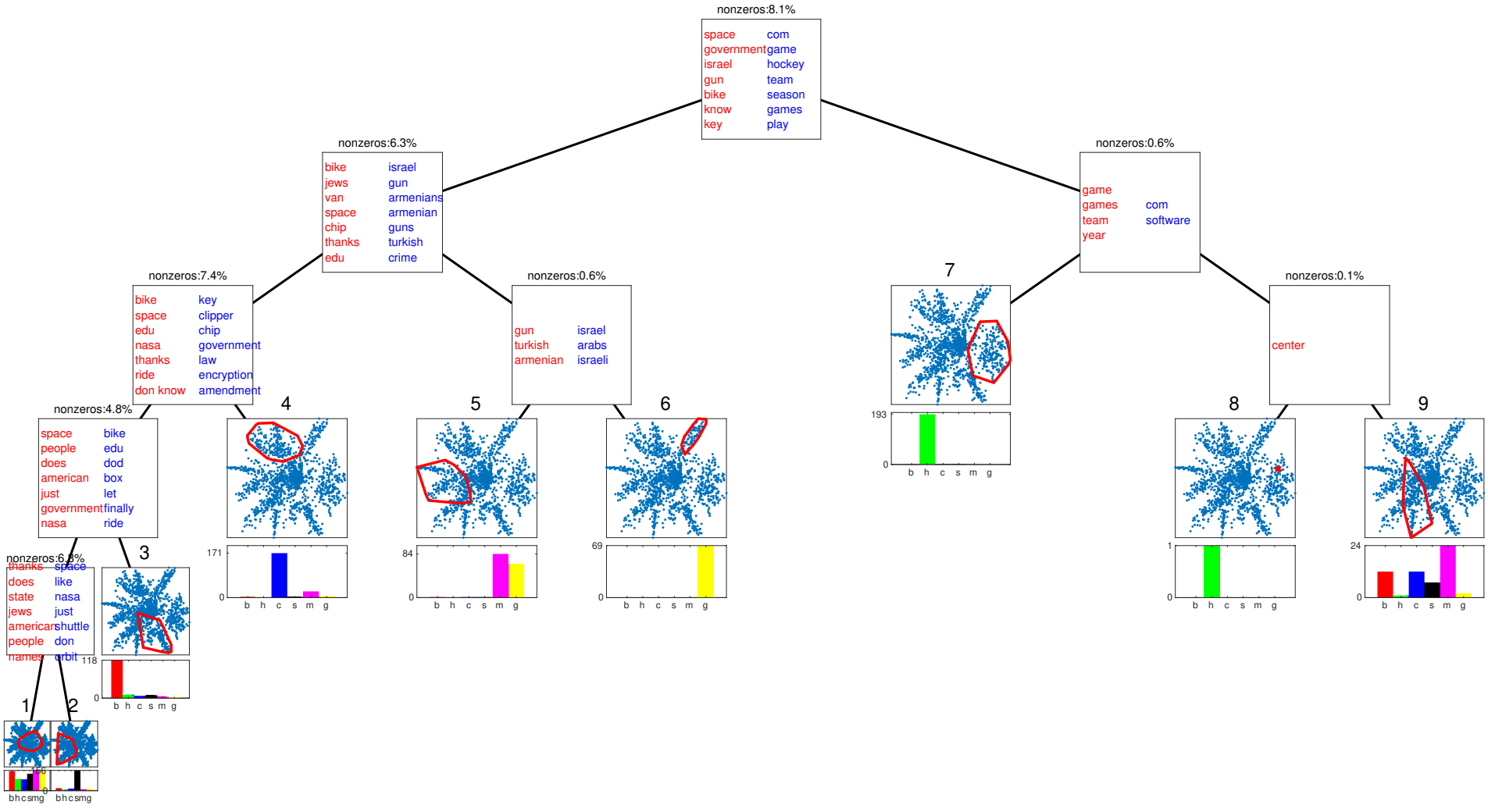


learning curves



- Results on 20-newsgroups dataset: 6 classes, tf-idf statistics on unigrams and bigrams as features (1000 features in total).
- We used elastic embedding to produce the free embedding.
- Direct fit trains an oblique tree (using TAO) directly to a free embedding, i.e. it uses free embedding as a label.
- The first iteration ($\mu = 0$) in learning curves (left plot) represents a direct fit. Our proposed approach (tree embedding) improves over this baseline (see iterations).

Experiments



Conclusion

- We have shown how to formulate a DT learning problem within manifold regularization framework
- This type of regularization appears in a range of machine learning problems
- The resulting problems are typically intractable to solve directly and we proposed efficient iterative algorithm to solve it.
- It is based on reformulation of the problem and decomposing it into two much simpler problems: fitting a tree and solving the step over coordinates (e.g. via linear system).
- Experimental results demonstrate that the algorithm can train accurate and interpretable decision trees in various scenarios.

Future work

- Semi-supervised learning for forests.
- Self-supervised representation learning with decision trees. The idea is to extract hierarchical representations of an input by stacking several layers of tree ensembles (forests).
- Theoretical properties of manifold regularization with decision trees.
  - Approximation guarantees
  - Convergence

Thank you!

References

[1] E. Alpaydın. *Introduction to Machine Learning*. Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, MA, third edition, 2014.

[2] M. Belkin. *Problems of Learning on Manifolds*. PhD thesis, Dept. of Mathematics,, Aug. 2003.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct. 2001.

[4] L. J. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, Calif., 1984.

[5] M. Á. Carreira-Perpiñán. The elastic embedding algorithm for dimensionality reduction. In J. Fürnkranz and T. Joachims, editors, *Proc. of the 27th Int. Conf. Machine Learning (ICML 2010)*, pages 167–174, Haifa, Israel, June 21–25 2010.

[6] M. Á. Carreira-Perpiñán and P. Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NEURIPS)*, volume 31, pages 1211–1221. MIT Press, Cambridge, MA, 2018.

[7] M. Á. Carreira-Perpiñán and W. Wang. Distributed optimization of deeply nested systems. arXiv:1212.5921, Dec. 24 2012.

[8] M. Á. Carreira-Perpiñán and W. Wang. Distributed optimization of deeply nested systems. In S. Kaski and J. Corander, editors, *Proc. of the 17th Int. Conf. Artificial Intelligence and Statistics (AISTATS 2014)*, pages 10–19, Reykjavik, Iceland, Apr. 22–25 2014.

[9] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proc. of the 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (SIGKDD 2016)*, pages 785–794, San Francisco, CA, Aug. 13–17 2016.

[10] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, May 1976.

[11] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, second edition, 2006.

[12] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

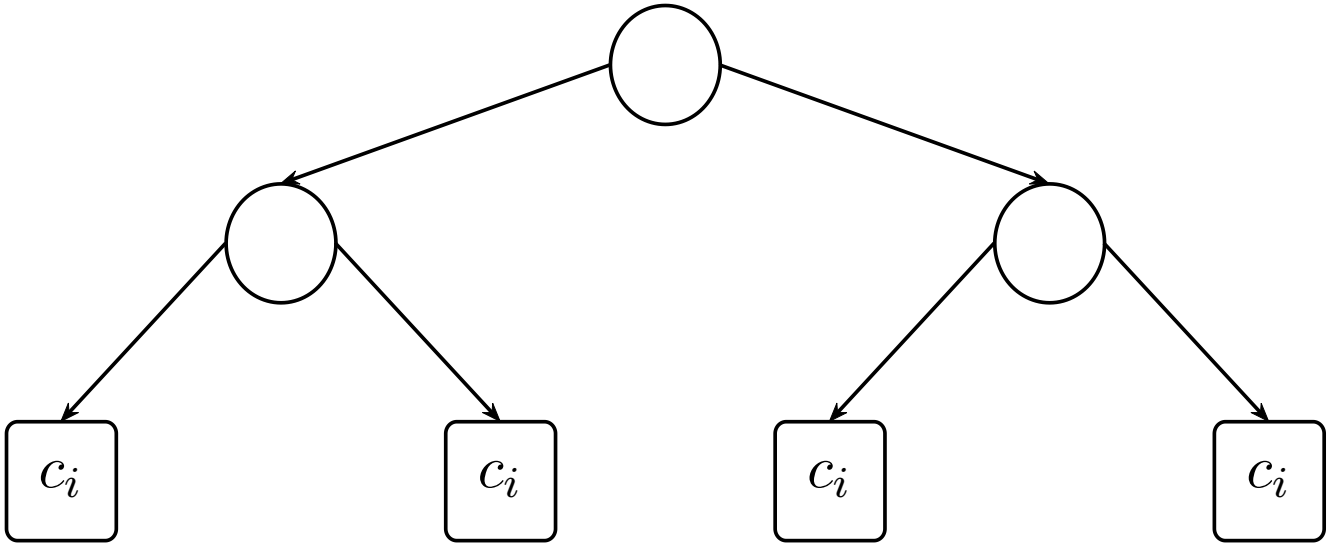
[13] M. Vladymyrov and M. Á. Carreira-Perpiñán. Entropic affinities: Properties and efficient numerical computation. In S. Dasgupta and D. McAllester, editors, *Proc. of the 30th Int. Conf. Machine Learning (ICML 2013)*, pages 477–485, Atlanta, GA, June 16–21 2013.

[14] A. Zharmagambetov and M. Á. Carreira-Perpiñán. Smaller, more accurate regression forests using tree alternating optimization. In H. Daumé III and A. Singh, editors, *Proc. of the 37th Int. Conf. Machine Learning (ICML 2020)*, pages 11398–11408, Online, July 13–18 2020.

Additional slides

Tree–step: optimizing over leaves

- Fix the tree structure as well as parameters in each decision node
- We assume each leaf i outputs a constant value c_i



Tree–step: optimizing over leaves

- Fix the tree structure as well as parameters in each decision node
- We assume each leaf i outputs a constant value c_i
- Tree prediction can be rewritten as: $T(\mathbf{x}) = \sum_{i=1}^{m} c_i \; b_i(\mathbf{x})$, where m is the number of leaves
- $b_i(\cdot) \in \{0, 1\}^m$ with single ith element equal to 1, indicating if $\mathbf{x}$ ends up in leaf i

Tree–step: optimizing over leaves

- Fix the tree structure as well as parameters in each decision node
- We assume each leaf i outputs a constant value c_i
- Tree prediction can be rewritten as: $T(\mathbf{x}) = \sum_{i=1}^{m} c_i \; b_i(\mathbf{x})$, where m is the number of leaves
- $b_i(\cdot) \in \{0,1\}^m$ with single ith element equal to 1, indicating if $\mathbf{x}$ ends up in leaf i
- Rewrite SSL objective over $\mathbf{c} = (c_1, \ldots, c_m)$ which has an exact solution given by another linear system:

$$\text{original obj:} \quad \min_{T} \sum_{n=1}^{l} \left(T(\mathbf{x}_n) - y_n\right)^2 + \gamma \sum_{n,m=1}^{N} w_{nm} \left(T(\mathbf{x}_n) - T(\mathbf{x}_m)\right)^2 \quad \Leftrightarrow$$

$$\text{leaves only:} \quad \min_{\mathbf{c}} \left(\mathbf{Bc} - \mathbf{y}\right)^T \mathbf{J} \left(\mathbf{Bc} - \mathbf{y}\right) + \gamma \, \mathbf{c}^T \mathbf{B}^T \mathbf{L} \mathbf{B} \mathbf{c} \quad \Rightarrow \quad \mathbf{Ac} = \mathbf{B}^T \mathbf{Jy}$$

- $\mathbf{B} = (b_i(\mathbf{x}_n)) \in \mathcal{R}^{N \times m}$ and can be precomputed since we fix the tree structure and parameters in all decision nodes.

Label–step: solving the linear system

- A reasonable choice: Conjugate Gradients (CG) since it leverages sparsity and supports warm start. **However, better solution exists for small–medium sized problems** (<30k data points)
- Observation: the matrix $\mathbf{A} = \mathbf{J} + \mu\mathbf{I} + \gamma\mathbf{L}$ is changed by adding $\mu\mathbf{I}$ at each iteration.
- Can we compute $\mathbf{A}^{-1}$ in $O(N^2)$ instead of $O(N^3)$?
- Denote $\mathbf{B} = \mathbf{J} + \gamma\mathbf{L}$ which is symmetric $\rightarrow$ calculate its eigendecomposition $\mathbf{B} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$
- Derivation of the inverse:

$$\mathbf{A}^{-1} = (\mu\mathbf{I}+\mathbf{B})^{-1} = (\mu\mathbf{I}+\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T)^{-1} = (\mathbf{Q}(\mu\mathbf{I}+\mathbf{\Lambda})\mathbf{Q}^T)^{-1} = \mathbf{Q}(\mu\mathbf{I}+\mathbf{\Lambda})^{-1}\mathbf{Q}^T$$

- Note that $\mu\mathbf{I} + \mathbf{\Lambda}$ is diagonal matrix and computing its inverse takes $O(N)$. Therefore, the overall computation is $O(N^2)$ at each iteration
- **But!** Precomputing decomposition for $\mathbf{B}$ is still $O(N^3)$ (and destroys the sparsity)

LapTAO: computational complexity

- Computational complexity
  - For solving large (sparse) linear system, CG iterates at most N times and each iteration takes $O(N^2)$. However, this is significantly cheaper than $O(N^3)$ with sparse matrices; < 30 seconds on the largest experiment we conducted (1GB of data).
  - Fitting an oblique tree with TAO is upper bounded by the tree depth times the cost of solving the logistic regression on the whole training set [6]