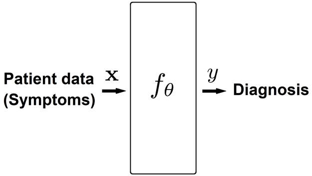
Efficient learning of neighbor representations for boundary trees and forests

Tharindu Adikari, Stark C. Draper

Department of Electrical and Computer Engineering
University of Toronto

CISS, 2019

Introduction

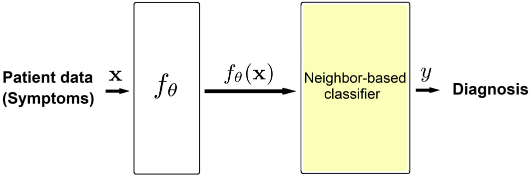


- In real world use cases

  non-quantitative factors are involved.

  black-box classifiers lack **interepretability**.

Introduction



- In real world use cases
  - **non-quantitative** factors are involved.
  - black-box classifiers lack **interepretability**.
- Neighbor-based classification:
  - Things that appear similar are likely similar.
  - Provides a natural reasoning for classifier decisions.
- Neighbor-based methods can
  - justify the decision process.
  - provide similar examples to a given query.

Approach

With neighbor-based a query is

- *not* constant-time like neural network classifier.
- linear in the size of training dataset in worst-case.

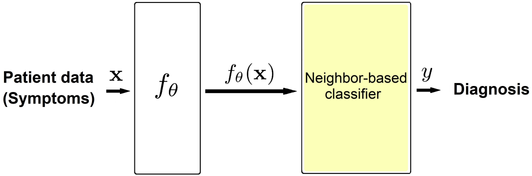


Figure: Use a cascade of neural network and a neighbor-based classifier.

Approach

With neighbor-based a query is

- *not* constant-time like neural network classifier.
- linear in the size of training dataset in worst-case.

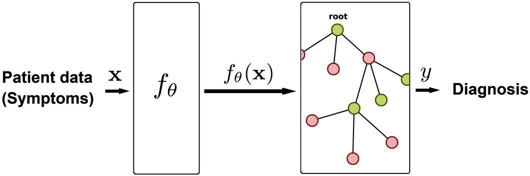


Figure: Use a cascade of neural network and a neighbor-based classifier.

Approach

With neighbor-based a query is

- *not* constant-time like neural network classifier.
- linear in the size of training dataset in worst-case.

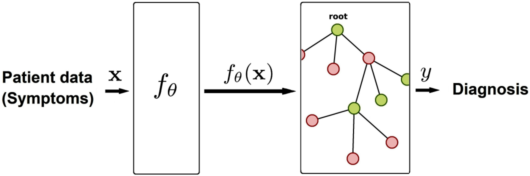


Figure: Use a cascade of neural network and a neighbor-based classifier.

Notation:

- $[K] = \{1, \ldots, K\}$ for any positive integer K.
- $\mathcal{D} = \{(\mathbf{x}_n, y_n) \mid \mathbf{x}_n \in \mathbb{R}^D, y_n \in [C], n \in [N]\}$ is a given dataset.

Boundary tree (BT) algorithm[1]

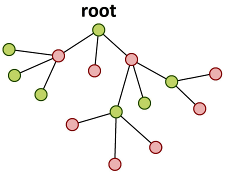


Figure: A BT with data points belonging to 2 classes.

- First, BT is built using $(\mathbf{x}_i, y_i) \in \mathcal{D}$.
- Nodes represent $(\mathbf{x}_i, y_i)$ - data, label pairs.
- Offers approximate nearest neighbor search (ANN).

[1] Charles Mathy et al. "The Boundary Forest Algorithm for Online Supervised and Unsupervised Learning". In: *AAAI Conf. Artificial Intelligence*. 2015.

Boundary tree (BT) algorithm[1]

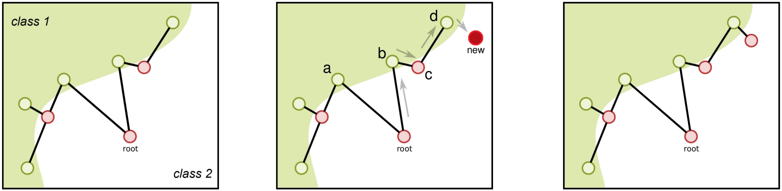


Figure: A BT with data points belonging to 2 classes.

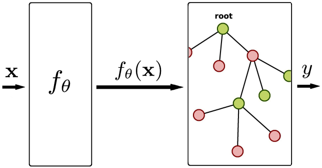
- First, BT is built using $(\mathbf{x}_i, y_i) \in \mathcal{D}$.
- Nodes represent $(\mathbf{x}_i, y_i)$ - data, label pairs.
- Offers approximate nearest neighbor search (ANN).

- Given $\mathbf{x}$, traverse BT searching for ANN in local neighborhoods.
- Local neighborhood: A node and its children.
- Use $\|\cdot\|_2$, the L_2 norm to measure distances.

[1] Charles Mathy et al. "The Boundary Forest Algorithm for Online Supervised and Unsupervised Learning". In: *AAAI Conf. Artificial Intelligence*. 2015.

Differentiable boundary trees[2] (DBTs)

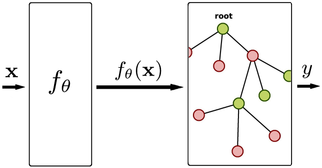
- Assert a neural network for $f_\theta : \mathbb{R}^D \to \mathbb{R}^m$.
- Build a BT with $f_\theta(\cdot)$ transformed data.
- Nodes represent $(f_\theta(\mathbf{x}_i), y_i)$ data, label pairs.



[2] Daniel Zoran, Balaji Lakshminarayanan, and Charles Blundell. "Learning Deep Nearest Neighbor Representations Using Differentiable Boundary Trees". In: *arXiv preprint arXiv:1702.08833* (2017).

Differentiable boundary trees[2] (DBTs)

- Assert a neural network for $f_\theta : \mathbb{R}^D \to \mathbb{R}^m$.
- Build a BT with $f_\theta(\cdot)$ transformed data.
- Nodes represent $(f_\theta(\mathbf{x}_i), y_i)$ data, label pairs.



- Minimize training loss $\mathcal{L}_{\text{dbt}}$ w.r.t. θ using gradient descent.
- Tree traversal is not a differentiable operation.
- (Minimize $\mathcal{L}_{\text{dbt}}$ over θ with BT fixed) $\longleftrightarrow$ (Re-build BT with θ fixed)

[2] Daniel Zoran, Balaji Lakshminarayanan, and Charles Blundell. "Learning Deep Nearest Neighbor Representations Using Differentiable Boundary Trees". In: *arXiv preprint arXiv:1702.08833* (2017).

Issues of DBT algorithm

(1) Only local neighborhood contributes to training.

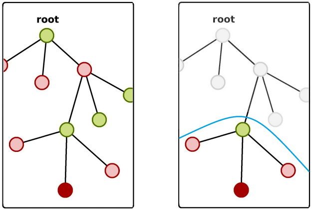


Figure: Dark red point is the closest to a training point $\mathbf{x}$. DBT training only considers the points below blue curve. Others do not contribute to training.

Issues of DBT algorithm

(1) Only local neighborhood contributes to training.

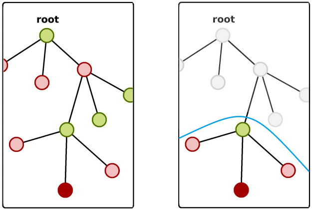


Figure: Dark red point is the closest to a training point $\mathbf{x}$. DBT training only considers the points below blue curve. Others do not contribute to training.

(2) **Batch-implementation** of DBT algorithm is **hard**.

Modern software/hardware tools rely on batch-implementation.

Tree traversal cannot be implemented as a batch operation.

- Is using a **tree** in **training** (not in testing/deploying) necessary?

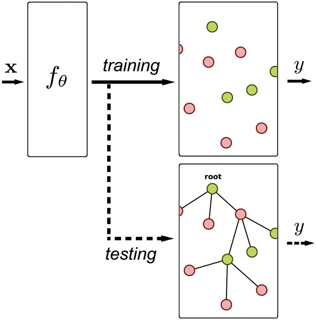
  Size of BT is limited by number of training points.

  Number of nodes in the tree is already small (typically < 100).

Proposing *Boundary Sets* and *Differentiable Boundary Sets*

Boundary set (BS):

- Follow boundary tree algorithm.
- Accumulate data in a **set**, rather than a tree.



Differentiable boundary set (DBS):

- All data points in the **set** contribute in optimization.
- Efficient **batch-implementation** is possible with existing tools.

Experiments

In each dataset,

- 10 image categories.
- $28 \times 28 = 784$ pixel images.
- 60,000 training examples.
- 10,000 test examples.

Digit-MNIST

Fashion-MNIST

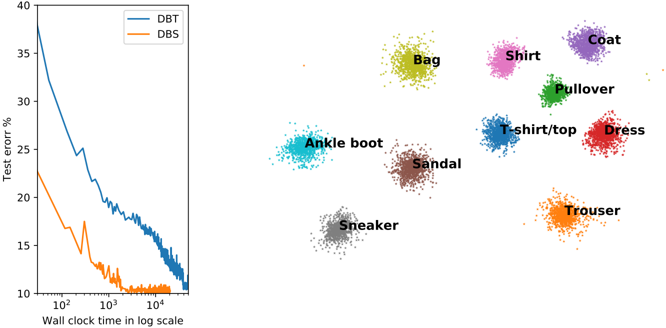
$f_\theta(\cdot)$ model architecture for DBT and DBS:

- $784 \xrightarrow{\text{relu}} 400 \xrightarrow{\text{relu}} 400 \xrightarrow{\text{identity}} 20$

Comparison with vanilla neural network (VNet) classifier.

- $784 \xrightarrow{\text{relu}} 400 \xrightarrow{\text{relu}} 400 \xrightarrow{\text{identity}} 20 \xrightarrow{\text{softmax}} 10$

DBS is much faster than DBT



(a) (b)

Figure: Experimental results with Fashion-MNIST training data.

(a) Comparison of training time for DBT and DBS.
(b) Learning 2-d representations by setting output dimension of $f_\theta(\cdot) = 2$.

Comparing test errors

Table: Test error comparison of DBT, DBS and VNet.

Model	Digit-MNIST		Fashion-MNIST	
	Test error %	# of nodes	Test error %	# of nodes
DBT	2.23	220	14.2	505
DBS	**1.52**	**29**	**10.3**	**26**
VNet	1.48	-	9.8	-

- # of nodes: Number of data points stored in the BT.
- DBS is the best performing in neighbor-based category.

Conclusion and future work

Proposed an algorithm that

- learns representations efficiently for neighbor-based classification.
- improves the accuracy and data representability of DBT.
- is easy to implement on modern machine learning tools.

Open for exploration:

- An adaptive classifier without the need for re-training.
- Two-stage training process that preserves privacy.

Thank you.

Questions?

References

- Mathy, Charles et al. "The Boundary Forest Algorithm for Online Supervised and Unsupervised Learning". In: *AAAI Conf. Artificial Intelligence*. 2015.
- Zoran, Daniel, Balaji Lakshminarayanan, and Charles Blundell. "Learning Deep Nearest Neighbor Representations Using Differentiable Boundary Trees". In: *arXiv preprint arXiv:1702.08833* (2017).
- Van Durme, Benjamin and Ashwin Lall. "Online Generation of Locality Sensitive Hash Signatures". In: *ACL Conf. Short Papers*. 2010.
- Gionis, Aristides, P. Indyk, and R. Motwani. "Similarity search in high dimensions via hashing". In: *Vldb*. 1999.

Appendix

Neighbor-based classification

Exact nearest neighbors:

- k-nearest neighbors
- Computational complexity: $\mathcal{O}(ND)$
- Logistic regression: $\mathcal{O}(1)$

$\mathcal{O}(\cdot)$ is the big O notation for computational complexity.

Neighbor-based classification

Exact nearest neighbors:

- k-nearest neighbors
- Computational complexity: $\mathcal{O}(ND)$
- Logistic regression: $\mathcal{O}(1)$

$\mathcal{O}(\cdot)$ is the big O notation for computational complexity.

Approximate nearest neighbors (ANNs):

- Tree-based: Organize data in a tree structure.
- Hashing-based: Computes low dimensional hash values.
- Computational complexity: Sub-linear

Hashing-based methods

Locality-sensitive hashing[4]

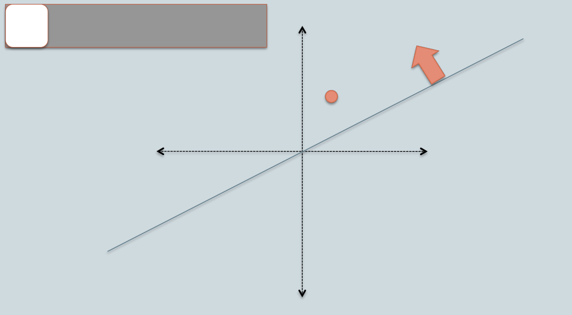


Figure: Intuition behind locality sensitive hashing[3].

[3] Van Durme and Lall, "Online Generation of Locality Sensitive Hash Signatures".

[4] Aristides Gionis, P. Indyk, and R. Motwani. "Similarity search in high dimensions via hashing". In: *Vldb*. 1999.

Hashing-based methods

Locality-sensitive hashing[4]

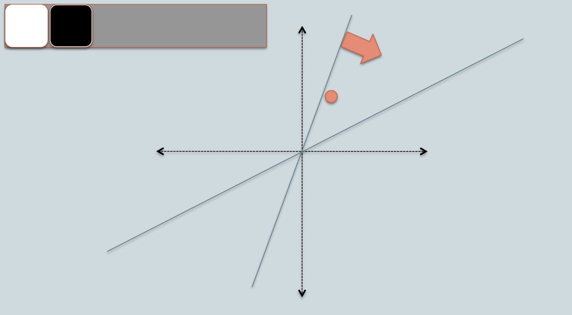


Figure: Intuition behind locality sensitive hashing[3].

[3] Van Durme and Lall, "Online Generation of Locality Sensitive Hash Signatures".

[4] Aristides Gionis, P. Indyk, and R. Motwani. "Similarity search in high dimensions via hashing". In: *Vldb*. 1999.

Hashing-based methods

Locality-sensitive hashing[4]

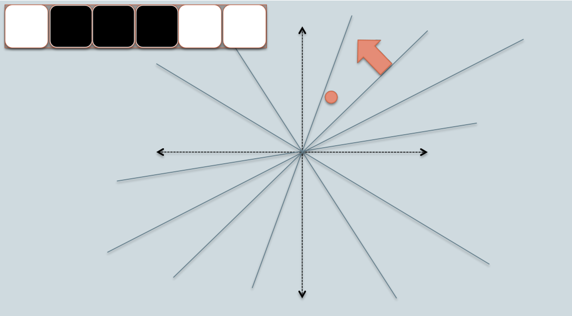


Figure: Intuition behind locality sensitive hashing[3].

[3]Van Durme and Lall, "Online Generation of Locality Sensitive Hash Signatures".

[4]Aristides Gionis, P. Indyk, and R. Motwani. "Similarity search in high dimensions via hashing". In: *Vldb*. 1999.

Hashing-based methods

Locality-sensitive hashing[4]

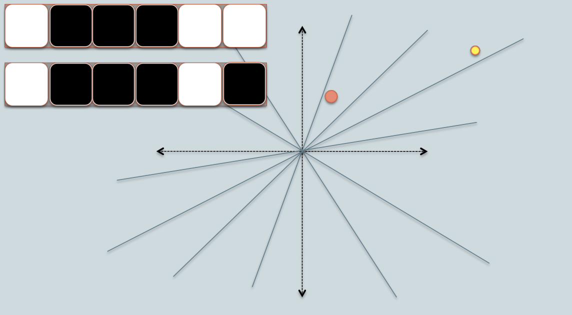


Figure: Intuition behind locality sensitive hashing[3].

[3] Van Durme and Lall, "Online Generation of Locality Sensitive Hash Signatures".

[4] Aristides Gionis, P. Indyk, and R. Motwani. "Similarity search in high dimensions via hashing". In: *Vldb*. 1999.

Building and querying a BT

- Local neighborhood: A node and its children.
- Querying BT for the ANN of $\mathbf{x}$:
  1. Traverse BT, searching for ANN in local neighborhoods.
- Training BT with a new data point $\mathbf{x}$:
  1. Traverse BT, searching for ANN in local neighborhoods.
  2. Add as a child **only** if the ANN is of a different class.

(Hence 'Boundary tree')

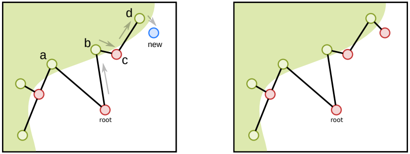
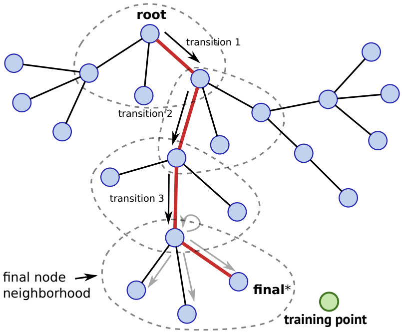


Figure: A BT with 2-d data belonging to 2 classes. New data point $\mathbf{x}$ shown in blue.

Probabilistic model for DBT traversal

- $(\mathbf{x}_r, y_r) \in \mathcal{D}$: training point
- $\mathsf{p}(i)$: parent node index of node-i
- $\mathcal{W}(i)$: index set of siblings of node-i
- $\mathcal{V}$: indexes in traversal path
- s: index of final node



$$\log \Pr{}^*(y_r = c|\mathbf{x}_r) = \left[\sum_{i \in \mathcal{V} \setminus s} \log \Pr(\mathsf{p}(i) \to i|r) \right] + \log \left[\sum_{i \in \mathcal{W}(s) \cup \{s\}} \Pr(\mathsf{p}(i) \to i|r) \mathbb{1}[y_i = c] \right]$$

Gradient descent with DBTs

- Unnormalized log soft-probabilities

$$\log \Pr^*(y_r = c|\mathbf{x}_r) = \left[\sum_{i \in \mathcal{V}\setminus s} \log \Pr(\mathsf{p}(i) \to i|r)\right] + \log\left[\sum_{i \in \mathcal{W}(s)\cup\{s\}} \Pr(\mathsf{p}(i) \to i|r)\mathbb{1}[y_i = c]\right]$$

- Normalized soft-probabilities

$$\Pr(y_r = c|\mathbf{x}_r) = \frac{\Pr^*(y_r = c|\mathbf{x}_r)}{\sum_{c' \in [C]} \Pr^*(y_r = c'|\mathbf{x}_r)}$$

- Cross entropy loss

$$\mathcal{L}_{\mathsf{dbt}} = -\sum_{c \in [C]} \mathbb{1}[y_r = c] \log \Pr(y_r = c|\mathbf{x}_r)$$

- Minimize $\mathcal{L}_{\mathsf{dbt}}$ with BT fixed $\longleftrightarrow$ Re-build BT with θ fixed.
- Use **final** DBT classifier at the test time.

Effectiveness of using the points near *boundary*

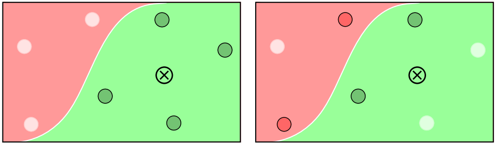


Figure: New training data point ⊗, and existing data points in the BS.

Implementing $f_\theta(\cdot)$ with neural networks

Feed-forward neural network comprising of L-layers.

- $\mathbf{x}^{(0)}$: input vector
- $W^{(l)}$ and $\mathbf{b}^{(l)}$ are learnable variables
- $\Phi^{(l)}$: activation function of l-th layer
- l-th layer output $(1 \leq l \leq L)$, $\mathbf{x}^{(l)} = \Phi^{(l)}(W^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)})$

- Parameter set $\theta = \{W^{(l)}, \mathbf{b}^{(l)}\}_{1 \leq l \leq L}$

$f_\theta(\cdot) : \mathbb{R}^D \rightarrow \mathbb{R}^m$ is implemented with

- $\Phi^{(L)}$ as the *identity* function.
- $\Phi^{(l)} = \text{relu}(\cdot) = \max(0, \cdot)$ for $1 \leq l < L$.

DBS algorithm

Algorithm 1: DBS training algorithm

input: $\mathcal{D}, N_\mathsf{b}, \sigma$
randomly initialize θ, parameters of $f_\theta(\cdot)$;
while *not reached maximum number of epochs* **do**
    shuffle elements and partition $\mathcal{D}$ to obtain subsets of size $(N_\mathsf{b}+1)$;
    foreach *subset* $\bar{\mathcal{D}}$ **do**
        $\mathcal{U} \leftarrow \{(f_\theta(\mathbf{x}_n), y_n) \mid (\mathbf{x}_n, y_n) \in \bar{\mathcal{D}}\}$;
        $\mathcal{U}_\mathsf{b} \leftarrow$ first N_b elements of $\mathcal{U}$;
        $\mathcal{S} \leftarrow$ boundary set computed using elements of $\mathcal{U}_\mathsf{b}$;
        $(f_\theta(\mathbf{x}_r), y_r) \leftarrow$ last element of $\mathcal{U}$;
        $\mathbf{d} \leftarrow$ row vector consisting Euclidean distances between each data point in $\mathcal{S}$ and $\mathbf{x}_r$;
        $\mathbf{w} \leftarrow$ softmax function applied on $\frac{-\mathbf{d}}{\sigma}$ i.e., $\mathbf{w}(i) = \frac{\exp(-\mathbf{d}(i)/\sigma)}{\sum_{j\in[|\mathcal{S}|]} \exp(-\mathbf{d}(j)/\sigma)}$ for $i \in [|\mathcal{S}|]$;
        $Y \leftarrow |\mathcal{S}| \times C$ matrix where rows are the one-hot label encodings of elements of $\mathcal{S}$;
        $\hat{\mathbf{y}} \leftarrow \mathbf{w}Y$ where $\hat{\mathbf{y}}(c) = \Pr(y_r = c | \mathbf{x}_r, \mathcal{S}, \theta)$ for $c \in [C]$;
        $\mathcal{L}_\mathsf{dbs} \leftarrow$ cross-entropy loss calculated with $\hat{\mathbf{y}}$ and y_r;
        Compute $\nabla_\theta \mathcal{L}_\mathsf{dbs}$ and take one step to minimize $\mathcal{L}_\mathsf{dbs}$;
    end
end

Model architecture of $f_\theta(\cdot)$

Differentiable boundary trees (DBT) and differentiable boundary sets (DBS)

- $f_\theta(\cdot) : 784 \xrightarrow{\text{relu}} 400 \xrightarrow{\text{relu}} 400 \xrightarrow{\text{identity}} 20$

In DBT, compute gradients by

- DBT-v1: only considering the new training data point.
- DBT-v2: considering new training data point and existing points in BT.

Comparison with vanilla neural network (VNet) classifier.

- $784 \xrightarrow{\text{relu}} 400 \xrightarrow{\text{relu}} 400 \xrightarrow{\text{identity}} 20 \xrightarrow{\text{softmax}} 10$