Neural decision trees

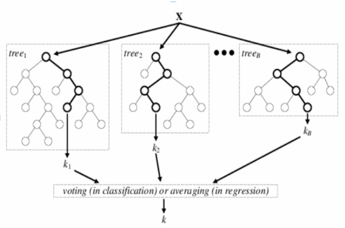
Team 12

ML, Skoltech

2019

Decision Trees & Random Forests

- Random forest - original idea by Tim Kam, Ho to implement stochastic discrimination (1995)
- Ensemble of decision trees: Wait! Decision trees are deterministic though!
- Main oracle: Bagging
- Widely used in practice



DTs: Training and testing

For each tree in ensemble we:

- Select a subset of samples and subset of features
- Create a tree with only 1 node
- Until stopping condition is achieved:
  - Make a split according to the criterion
- For each leaf node
  - Discrete: $p(c|v)$ $arg \max_c p(c|v)$
  - Continuous: mean or predefined func

Still, there is no use of differentiability. How can we convert DT learning to a differentiable one?

Pros and cons

	Decision trees	Neural Networks
Interpretability	✓	✗
Model functions diversity	Only axis parallel splits	Arbitrary functions, Complex structures
Time complexity	Reasonably fast	Comparably slow, long training
Online learning	✗	✓
Model parameters	Only a few	Up to millions (hidden layers, number of units)
Layout	Determinstic splits	Differentiable, stochastic, back-propagation compatible

Soft decision tree

- The input data is fed to the neural network
- The outputs of FC layer represent routing probabilities in each of the trees of the ensemble
- The assignment is random

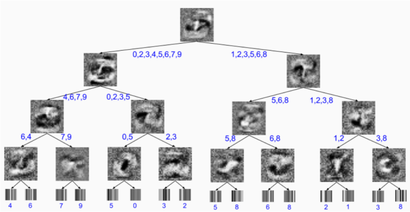
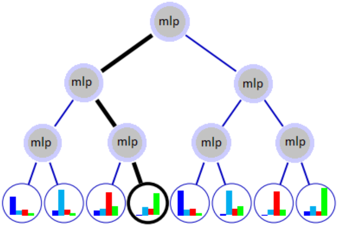


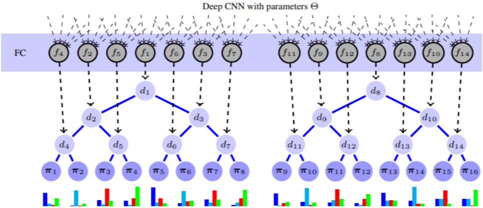
Figure: Soft decision tree trained on MNIST. The inner nodes are the learned filters, the leaves are visualizations of the learned probability distribution over classes.

MLP embedded decision tree

- each splitting node is an independent MLP allowing oblique decision functions
- similar mechanism to Hashing Neural Networks
- predictions can be given by:
  - using the distribution from the leaf with the greatest path probability
  - averaging the distributions over all the leaves, weighted by their respective path probabilities



Cross-dependent CNN Decision Forest



Prediction: $\mathbb{P}_T(y|x,\Theta,\pi) = \sum_l \pi_{l_y}\mu_l(x|\Theta)$, where

π_{l_y} is the probability of a sample reaching leaf l to take on class y,

$$\mu_l(x|\Theta) = \prod_{n\in\mathcal{N}} d_n(x;\Theta)^{\mathbb{1}_{l\swarrow n}}(1-d_n(x;\Theta)^{\mathbb{1}_{l\searrow n}}),$$

$$d_n(x;\Theta) = \sigma(f_n(x;\Theta)).$$

Prediction for forest: $\mathbb{P}_{\mathcal{F}}(y|x) = \frac{1}{k}\sum_{h=1}^{k} P_{T_h}(y|x)$.

Skoltech
Skolkovo Institute of Science and Technology

Loss function and node output

- The loss over decision nodes should be converted to a differentiable one

$$L(\Theta, \pi, x, y) = -\log\left(\mathbb{P}_T(y|x, \Theta, \pi)\right)$$

- Leaf nodes outputs update formula:

$$\pi_{l_y}^{(t+1)} = \frac{1}{Z_l^t} \sum_{x, y' \in \mathcal{T}} \frac{\mathbb{1}_{y=y'} \pi_{l_y}^{(t)} \mu_l(x|\Theta)}{\mathbb{P}_T(y|x, \Theta, \pi^{(t)})}$$

Training algorithm

Require: $\mathcal{T}$: training set, nEpochs

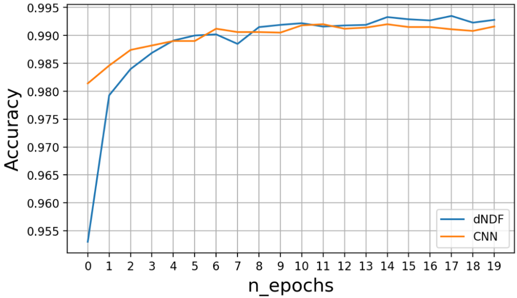
1. Initialize Θ randomly
2. **for all** $i \in \{1, \ldots, \mathrm{nEpochs}\}$ **do**
3.   Update π
4.   Break $\mathcal{T}$ into a set of mini-batches
5.   **for all** $\mathcal{B}$: mini-batch from $\mathcal{T}$ **do**
6.   Update Θ by SGD step
7.   **end for**
8. **end for**

Models comparison

	Decision Forest	Neural Network	Neural Decision Tree
Feature learning	✗	✓	✓
GD applied	✗	✓	✓
Loss	NP-hard to build optimal tree	Non-convex	Non-convex
Interpretable	✓	✗	Depends

Results on MNIST

Model	Score
Random Forest	82.4%
Decision Tree	67.4%
Extra Trees	51.2%
CNN	99.2%
Cross-dependent CNN (NDT)	99.3%
MLP embedded Decision tree	92.4%



References

1. R. Balestriero (2017). Neural Decision Trees, https://arxiv.org/abs/1702.07360
2. P. Kontschieder, M. Fiterau, A. Criminisi, S.R. Bulo (2015). Deep Neural Decision Forests, https://www.ijcai.org/Proceedings/16/Papers/628.pdf
3. N. Frosst, G. Hinton (2017). Distilling a Neural Network Into a Soft Decision Tree, https://arxiv.org/abs/1711.09784