Neural decision trees

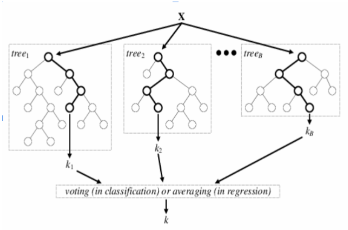
Team 12

ML, Skoltech

2019

Decision Trees & Random Forests

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

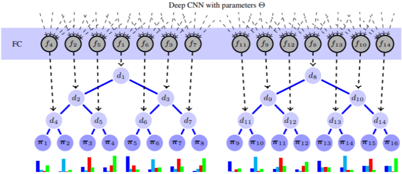
	Decision trees	Neural Networks
Easily interpretable	✓	✗
Model functions diversity	Only axis parallel splits	Arbitrary functions, Complex structures
Time complexity	Reasonably fast	Comparably slow, long training
Online learning	✗	✓
Model parameters	Only a few	Up to millions (hidden layers, number of units)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

Neural Decision trees

- Instead of using weak learners as base classifiers, we can make use of the features learned by neural network

Neural Decision trees

- 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



$$d_i(x) = g(f_j(x)), \quad p(x, \pi_i) = \prod_{n \in \{d_1, \ldots, \pi_i\}} p_{route}(n)$$

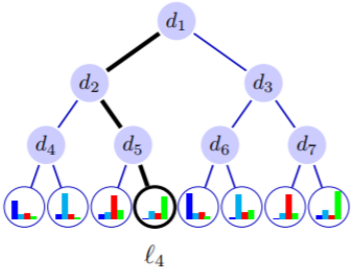
Neural Decision trees

$$\mathbb{P}_T[y|\boldsymbol{x},\Theta,\boldsymbol{\pi}] = \sum_{\ell\in\mathcal{L}} \pi_{\ell y}\mu_\ell(\boldsymbol{x}|\Theta)$$

$$\mu_\ell(\boldsymbol{x}|\Theta) = \prod_{n\in\mathcal{N}} d_n(\boldsymbol{x};\Theta)^{\mathbb{1}_{\ell\swarrow n}} \bar{d}_n(\boldsymbol{x};\Theta)^{\mathbb{1}_{n\searrow\ell}}$$

$$d_n(\boldsymbol{x};\Theta) = \sigma(f_n(\boldsymbol{x};\Theta))$$

$$\mathbb{P}_\mathcal{F}[y|\boldsymbol{x}] = \frac{1}{\mathsf{k}}\sum_{h=1}^{\mathsf{k}} \mathbb{P}_{T_h}[y|\boldsymbol{x}]\,,$$



Loss function and node output

- he 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 are

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

Trainig algorithm

Require: $\mathcal{T}$: training set, nEpochs
1. Initialize Θ randomly
2. **for all** $i \in \{1, \ldots, \mathrm{nEpochs}\}$ **do**
3.   Compute π using formula above
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
Ability to parallelize	✓	✗	✗
Feature learning	✗	✓	✓
Gradient Descent applicable	✗	✓	✓
Loss	NP hard to build optimal tree	Non convex	Non convex

Skoltech
Skolkovo Institute of Science and Technology

Results

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