Stochastic trees, forests, wavelets & neural networks

Starting point

This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Deep Neural Decision Forests

Peter Kontschieder[1] Madalina Fiterau[*,2] Antonio Criminisi[1] Samuel Rota Bulò[1,3]

Microsoft Research[1] Cambridge, UK
Carnegie Mellon University[2] Pittsburgh, PA
Fondazione Bruno Kessler[3] Trento, Italy

Motivation

- DL architectures need to be a.e. differentiable with respect to the given loss function and their parameters.
- ML classifiers typically used at the end of a DL pipeline are indeed differentiable with respect to their input and weights: logistic regression (binary classification), soft-max (classification), linear regression (regression), etc.
- Such classifiers are ‘weak’. So why does DL work? The whole network works towards ‘cooking’ a good last feature space for these classifiers.

- Standard Decision trees, forests are ‘strong classifiers’ but non-parametric and thus not differentiable. They use deterministic routing. Let’s replace with ‘soft’ stochastic routing…

A ‘Soft’ decision tree

- Recall, we like to use a uniform approach to regression / classification. So, we may assume we have a vector valued function/dataset.
- Let’s use a logistic regression model at each node to create ‘soft’ stochastic routing at the node Ω

$$h_{\Omega}(x) = h_{\theta(\Omega)}(x) := \frac{1}{1+e^{-\left(\langle \beta(\Omega), x\rangle + \beta_0(\Omega)\right)}}.$$

- Observe there is an underlying hyper-plane subdivision associated with the ‘soft’ decision function h_{Ω}.
- The parameters of the tree are $\Theta = \{\theta(\Omega) : \Omega \in \mathcal{T}\}$
- We could use other more complex routing at each node (e.g. small NN).

- At any node Ω, a point $x \in \mathbb{R}^n$ is routed to a
  - right child Ω' with probability $p_{\Omega,\Omega'}(x) := h_\Omega(x)$
  - left child Ω'' with probability $p_{\Omega,\Omega''}(x) := 1 - h_\Omega(x)$
- The probability of x reaching any node $\tilde{\Omega}$ is

$$p_{\tilde{\Omega}}(x|\Theta) := \prod_{\Omega \text{ is ancestor of } \tilde{\Omega}} p_{\Omega,\Omega'}(x)$$

- Obviously, at any level $\mathcal{T}_l$

$$\sum_{\Omega \in \mathcal{T}_l} p_\Omega(x|\Theta) = 1, \quad \forall x \in \mathbb{R}^n.$$

- Vector-valued regression function on the tree with m levels

$$\tilde{f}_{\mathcal{T}}(x) = \sum_{\Omega \in \mathcal{T}_m} \vec{E}_\Omega p_\Omega(x|\Theta)$$

- One can conjecture higher order ‘local’ approximations at leaves

$$\tilde{f}_{\mathcal{T}}(x) = \sum_{\Omega \in \mathcal{T}_m} \vec{E}_{\Omega}(x)\, p_{\Omega}(x|\Theta).$$

- The forest (as usual) aggregates the inference of ‘soft’ trees

$$\tilde{f}_{\mathcal{F}}(x) = \frac{1}{J}\sum_{j=1}^{J} \tilde{f}_j(x)$$

- Classification? One can replace the soft-max layer by the normalization of $\tilde{f}(x) = \left(\tilde{f}_1(x), \ldots, \tilde{f}_L(x)\right)$

$$\overline{f}(x) = \frac{1}{\sum_{i=1}^{L} e^{\tilde{f}_i(x)}} \left(e^{\tilde{f}_1(x)}, \ldots, e^{\tilde{f}_L(x)}\right).$$

Training

1. Random initialization of Θ (along with the parameters of the NN component if exists).
2. For each epoch:
   a. Computation of the leaf values $\left\{\vec{E}_{\Omega}\right\}_{\Omega\in\mathcal{T}_m}$ using the entire training set (least squares, negative log likelihood) with fixed Θ.
   b. For each mini-batch of training set:
      Update Θ using SGD

Stochastic Wavelets

The value at an inner node $\Omega \in \mathcal{T}_{l-1}$, with children $\Omega', \Omega'' \in \mathcal{T}_l$, $\Omega' \bigcup \Omega'' = \Omega$ is defined bottom-up recursively by

$$\vec{E}_\Omega(x) := p_{\Omega'|\Omega}(x)\vec{E}_{\Omega'}(x) + p_{\Omega''|\Omega}(x)\vec{E}_{\Omega''}(x).$$

If we define the stochastic wavelets by

$$\begin{aligned}\psi_{\Omega'}(x) &:= p_{\Omega'}(x)\vec{E}_{\Omega'}(x) - p_{\Omega'|\Omega}(x)\,p_\Omega(x)\vec{E}_\Omega(x) \\ &= p_{\Omega'}(x)\left(\vec{E}_{\Omega'}(x) - \vec{E}_\Omega(x)\right),\end{aligned}$$

we get

$$\sum_{\Omega' \in \mathcal{T}_l} p_{\Omega'}(x)\vec{E}_{\Omega'}(x) - \sum_{\Omega \in \mathcal{T}_{l-1}} p_\Omega(x)\vec{E}_\Omega(x) = \sum_{\Omega' \in \mathcal{T}_l} \psi_{\Omega'}(x),$$

Let’s see that

$$\sum_{\Omega' \in \mathcal{T}_l} p_{\Omega'}(x)\vec{E}_{\Omega'}(x) - \sum_{\Omega \in \mathcal{T}_{l-1}} p_{\Omega}(x)\vec{E}_{\Omega}(x)$$

$$= \sum_{\substack{\Omega \in \mathcal{T}_{l-1} \\ \Omega', \Omega'' \text{child of } \Omega}} p_{\Omega'}(x)\vec{E}_{\Omega'}(x) + p_{\Omega''}(x)\vec{E}_{\Omega''}(x) - \left(p_{\Omega'}(x) + p_{\Omega''}(x)\right)\vec{E}_{\Omega}(x)$$

$$= \sum_{\substack{\Omega \in \mathcal{T}_{l-1} \\ \Omega', \Omega'' \text{child of } \Omega}} p_{\Omega'}(x)\left(\vec{E}_{\Omega'}(x) - \vec{E}_{\Omega}(x)\right) + p_{\Omega''}(x)\left(\vec{E}_{\Omega''}(x) - \vec{E}_{\Omega}(x)\right)$$

$$= \sum_{\Omega' \in \mathcal{T}_l} \psi_{\Omega'}(x)$$

The telescopic sum

$$\tilde{f}(x) = \sum_{\Omega \in \mathcal{T}_m} p_\Omega(x) \vec{E}_\Omega(x)$$

$$= \left(\sum_{\Omega \in \mathcal{T}_m} p_\Omega(x) \vec{E}_\Omega(x) - \sum_{\Omega \in \mathcal{T}_{m-1}} p_\Omega(x) \vec{E}_\Omega(x) \right)$$

$$+ \left(\sum_{\Omega \in \mathcal{T}_{m-1}} p_\Omega(x) \vec{E}_\Omega(x) - \sum_{\Omega \in \mathcal{T}_{m-2}} p_\Omega(x) \vec{E}_\Omega(x) \right)$$

$$+ \cdots + \vec{E}_{[0,1]^n}(x),$$

gives the stochastic wavelet decomposition

$$\tilde{f}(x) = \sum_{\Omega \in \mathcal{T}} \psi_\Omega(x), \qquad \psi_{[0,1]^n} := \vec{E}_{[0,1]^n}(x).$$