Indecisive trees for classification and prediction of knee osteoarthritis

DOI:
10.1007/978-3-319-67389-9_33

Document Version
Accepted author manuscript

Indecisive Trees for Classification and Prediction of Knee Osteoarthritis

Luca Minciullo[1], Paul A. Bromiley[1], David T. Felson[2] and Timothy F. Cootes[1]

[1] The University of Manchester
[2] ARUK Epidemiology Unit, The University of Manchester

Abstract. Random forests are widely used for classification and regression tasks in medical image analysis. Each tree in the forest contains binary decision nodes that choose whether a sample should be passed to one of two child nodes. We demonstrate that replacing this with something less decisive, where some samples may go to both child nodes, can improve performance for both individual trees and whole forests. Introducing a soft decision at each node means that a sample may propagate to multiple leaves. The tree output should thus be a weighted sum of the individual leaf values. We show how the leaves can be optimised to improve performance and how backpropagation can be used to optimise the parameters of the decision functions at each node. Finally, we show that the new method outperforms an equivalent random forest on a disease classification and prediction task.

Keywords: Random Forests, Decision Trees, Optimisation

1 Introduction

Decision trees, particularly in the form of Random Forests, are widely used in medical image analysis for tasks such as landmark localisation [5], segmentation [8] and classification [6, 7]. In most cases each tree uses hard decision nodes, a threshold on a feature response derived from the input, in which a sample is channelled either to the left or right child node. Thus one input sample ends up at exactly one leaf, which holds the output for the tree.

A natural extension is to replace this binary decision with something softer, so that a sample can go down both branches, but with different weights or probabilities depending on the feature response at the node. An early example of this approach was "Fuzzy Decision Trees" [9] in which a sigmoidal function was used to assign a weight to be passed down each child branch. The approach was extended in [4], where a forest of such trees was integrated into a deep network allowing end-to-end training. However, one problem with using a sigmoidal transfer function is that every input effectively ends up at every leaf of the tree with a non-zero weight, though at most leaves the weight may be very close to zero. This is potentially very inefficient for deep trees.

In this paper we introduce trees in which only samples near the decision boundary are propagated to both children; most samples only go to one child.

This is equivalent to using a simple, sloped step function to compute the weights. Each input is then propagated to a relatively small number of leaves. This allows the use of deep trees whilst retaining most of the efficiency of binary decision trees. We describe the approach in detail, including a greedy method for training a tree. As with fuzzy trees, both the values stored at the leaf nodes and the parameters of the transfer functions can be optimised using either closed form or gradient descent approaches, leading to better performance than that from the greedy training. We demonstrate that replacing random forests with these more indecisive trees leads to improvements in overall performance on a classification task. We show the improvement in performance of our methodology on Osteoarthritis (OA) classification and prediction tasks. OA is the most common form of arthritis, affecting millions of people around the world, the chance of developing the disease being particularly high in older people. The most common signs of OA are osteophytes, bony spurs that grow on the bones of the spine or around the joints, joint space narrowing (JSN), and calcium deposits. We train our trees to use features that measure the shape and appearance of the knee in radiographs, to classify OA status and predict who is at risk of developing the disease.

2 Background

Random Forests are a very successful machine learning ensemble model, where each of the sub-models is a binary decision tree. The randomness comes from two main sources. First, each of the decision trees is trained on a different sample of the original dataset, obtained by generating multiple bootstrap samples. Second, the optimal split is found by considering only a random subset of the features appearing in the data.

Ren *et al.* [8] showed how the leaves of a forest could be mutually optimised to give better performance than that of a forest with independent trees. Fuzzy decision trees, which can be optimised by a backpropagation-like algorithm, were introduced in [9]. They proposed training a tree in the normal way, then replacing the binary decision threshold with a sigmoidal function to indicate branch membership, the parameters of which could then be optimised. Kontschieder *et al.* [4] extended this idea to full decision forests, using a sigmoidal decision function. They too used a stochastic gradient descent approach to optimise the parameters of the decision nodes and the leaves. The decisions at each node were based on the output of one node of a deep convolutional network, making the entire system amenable to end-to-end training.

When using a sigmoidal function for branch membership, every sample ends up being propagated to every leaf of the tree, even though at some leaves the membership value may be very small. This may lead to inefficiencies for deep trees. To overcome this we use a ramp function for the membership propagation

$$\pi(\mathbf{x}; t_0, t_1) = \begin{cases} 0 & \text{if} \quad f(\mathbf{x}) \leq t_0 \\ \frac{(f(\mathbf{x})-t_0)}{(t_1-t_0)} & \text{if} \quad t_0 < f(\mathbf{x}) < t_1 \\ 1 & \text{if} \quad f(\mathbf{x}) \geq t_1 \end{cases} \tag{1}$$

where $f(\mathbf{x})$ is a feature derived from the input $\mathbf{x}$ and $t_0 < t_1$ are two thresholds defining the ramp function. Thus, if the membership for either branch is zero, we do not need to propagate down that branch. During training we choose the thresholds so that a given proportion of the training samples are in the ambiguous region (see below).

2.1 Evaluating the result from a tree

An indecisive tree is a collection of decision nodes and leaf nodes. Each decision node has two child nodes (left and right), a function that computes a scalar feature value from the input $f(\mathbf{x})$, and two threshold values defining the transfer function, t_0, t_1. Each leaf node contains an output value. When an input, $\mathbf{x}$, is evaluated with the tree, the output is a set of leaf values and associated weights, $\mathcal{S} = \{(\mathbf{v}_i, w_i)\}$. Starting at the root node, we propagate an input through the nodes, exploring only the branches with non-zero weights. Each node either adds its value (if it is a leaf) to a set of outputs, or it propagates the input and weight to one or both of its child nodes. This can be computed with a recursive function, starting at the root node with a unit weight: $\mathcal{S}$ =EVALUATE(root,$(\mathbf{x}, 1.0), \{\})$. The function is defined as follows:

1: **function** EVALUATE(node,$(\mathbf{x}, w), \mathcal{S})$
2: **if** $node.isLeaf$ **then**
3: $\mathcal{S} \leftarrow \{\mathcal{S}, (node.value, w)\}$
4: **else**
5: $\mu = \pi(node.f(\mathbf{x}), node.t_0, node.t_1)$
6: $w_L = (1 - \mu)w$
7: $w_R = \mu w$
8: **if** $w_L < w_R$ **then**
9: if $(w_L < w_t)$ then $w_L \leftarrow 0, w_R \leftarrow w$
10: **else**
11: if $(w_R < w_t)$ then $w_R \leftarrow 0, w_L \leftarrow w$
12: **end if**
13: if $(w_L > 0)$ $\mathcal{S}$ ←EVALUATE(node.leftChild,$(\mathbf{x}, w_L), \mathcal{S})$
14: if $(w_R > 0)$ $\mathcal{S}$ ←EVALUATE(node.rightChild,$(\mathbf{x}, w_R), \mathcal{S})$
15: **end if**
16: **return** $\mathcal{S}$
17: **end function**

The tests in lines 8-12 allow a threshold (w_t) to be enforced on the smallest allowable weight. If a split would cause the weight propagated to one child node to fall below the threshold, then that child node is ignored and all the weight is passed to the other child. Setting $w_t > 0$ ensures that no leaf is reached with a weight lower than w_t, and focuses processing on the branches with higher weights. It thus also limits the maximum number of leaves that can be returned to w_t^{-1}. The output of the tree can then be computed from $\mathcal{S}$ as the weighted sum of the leaf outputs, $\mathbf{v} = \sum_i w_i \mathbf{v}_i$.

3 Training and Optimising Indecisive Trees

In a similar way to training a normal decision tree, an indecisive tree is trained using a greedy recursive algorithm in which each node finds a feature and threshold to split the data arriving at it so as to minimise a cost function. During training a sample consists of a triplet, $(\mathbf{x}, \mathbf{y}, w)$, containing the input vector, the target output and a weight. To train a node, we consider the set of n samples $\mathcal{D}$ arriving from the parent node. To evaluate a particular choice of feature, $f(\mathbf{x})$, and thresholds t_0, t_1, we compute the sets of data $\mathcal{D}_L$ and $\mathcal{D}_R$ that would be propagated to the child nodes, and the cost function

$$C(f, t_0, t_1) = C(\mathcal{D}_L) + C(\mathcal{D}_R) \tag{2}$$

The cost $C(\mathcal{D})$ depends on the task. For instance, for regression, it can be the sum-of-squared differences. A random selection of features and possible thresholds is evaluated, and those giving the lowest cost retained.

Since finding the optimal pair of thresholds can be computationally expensive, we use the following approach. For each input $(\mathbf{x}_i, \mathbf{y}_i, w_i)$ we compute the feature value $f_i = f(\mathbf{x}_i)$, then rank the samples using this value. Let $(\mathbf{x}_j, \mathbf{y}_j, w_j)$ be the j^{th} sample in this ranked list. By computing running sums through this ranked data we can efficiently locate the index, k, for the hard split leading to the lowest total cost (all samples $j \leq k$ are sent to one child, all $j > k$ to the other). We then introduce an ambiguous region to include a proportion of approximately $r \in [0, 1]$ of the samples by setting $j0 = \max(1, k - 0.5rn)$, $j1 = \min(n, k + 0.5rn)$, and selecting $t_0 = f_{j0}$, $t_1 = f_{j1}$.

Since the samples in the ambiguous region will go to both children, the total number of samples propagated from nodes at depth d will be approximately $n_0.(1 + r)^d$, where n_0 is the original number of training examples, though it should be remembered that the total weight for each of the original samples will always sum to unity. In order to avoid propagating large numbers of samples with small weights, we use the same technique as described above (Sec. 2.1). If a sample weight would fall below w_t when propagated to one child node, we ignore that child and propagate all the sample weight to the other child. Decision nodes are added in a recursive manner until a suitable stopping condition (a maximum depth, minimum number of samples or measure of spread) is reached. The values at the leaf nodes can then be set to the weighted mean of the samples reaching that node, for instance for regression, the value

$$\mathbf{t} = \left(\sum w_i \mathbf{y}_i\right) / \left(\sum w_i\right) \tag{3}$$

3.1 Optimising the leaf values

A tree with vector output can be expressed as a function of input $\mathbf{x}$

$$\mathbf{y} = \mathbf{V}\mathbf{w}(\mathbf{x}) \tag{4}$$

where $\mathbf{V}$ is a matrix whose columns are all the leaf vectors, and $\mathbf{w}(\mathbf{x})$ is the sparse vector of weights returned by the tree, which selects the leaves to which $\mathbf{x}$ in propagated. Thus the outputs corresponding to the training inputs can be expressed as

$$\mathbf{Y} = \mathbf{VW} \tag{5}$$

where $\mathbf{Y} = (\mathbf{y}_1|...|\mathbf{y}_n)$ and $\mathbf{W} = (\mathbf{w}(\mathbf{x}_1)|...|\mathbf{w}(\mathbf{x}_n))$ is a sparse matrix.

For regression, as in [8], the leaf values can be found by minimising

$$Q(\mathbf{V}) = ||\mathbf{VW} - \mathbf{Y}||_2 + \alpha||\mathbf{V}||_2 \tag{6}$$

where α is an optional ridge regression regularisation function. Since $\mathbf{W}$ is sparse the solution can be found efficiently with conjugate gradient descent.

3.2 Optimising the decision nodes

If the leaf values are fixed, each decision node only affects the final output through the way it changes the weights on the samples passing through it. As in [9, 4] we can use a gradient descent-based backpropagation algorithm to optimise the parameters. However, in our case, since each sample only passes through a small subset of nodes, this can be significantly more efficient, as we only have to compute values at the nodes visited. The cost function to be minimised is of the form

$$Q_T(\theta; \{(\mathbf{x}_i, \mathbf{y}_i)\}) = \sum_i Q(\mathbf{Vw}(\mathbf{x}_i, \theta), \mathbf{y}_i) \tag{7}$$

where θ are the parameters affecting the weights and $Q(\mathbf{t}, \mathbf{y})$ is the cost function comparing the output of the tree, $\mathbf{t} = \mathbf{Vw}(\mathbf{x}, \theta)$ with the target output $\mathbf{y}$.

Gradient at leaf nodes: The contribution to the output from a single leaf node is given by $w\mathbf{v}$, where w is the weight of the sample arriving at the leaf. For one leaf,

$$\frac{dQ}{dw} = \frac{dQ}{d\mathbf{t}}\frac{d\mathbf{t}}{dw} = \mathbf{v}^T\frac{dQ}{d\mathbf{t}} \tag{8}$$

Gradient at decision nodes: At a decision node, the weights passed to the output nodes are given by

$$\begin{pmatrix} w_L \\ w_R \end{pmatrix} = w \begin{pmatrix} 1 - \pi(f, t_0, t_1) \\ \pi(f, t_0, t_1) \end{pmatrix} \tag{9}$$

If the parameters at the decision nodes are θ, then

$$\begin{aligned} \tfrac{dQ}{d\theta} &= \tfrac{dw_R}{d\theta}\tfrac{dQ}{dw_R} + \tfrac{dw_L}{d\theta}\tfrac{dQ}{dw_L} \\ &= w\tfrac{d\pi}{d\theta}\tfrac{dQ}{dw_R} - w\tfrac{d\pi}{d\theta}\tfrac{dQ}{dw_L} \\ &= w\tfrac{d\pi}{d\theta}\left(\tfrac{dQ}{dw_R} - \tfrac{dQ}{dw_L}\right) \end{aligned} \tag{10}$$

Similarly

$$\frac{dQ}{dw} = \pi(\theta)\frac{dQ}{dw_R} + (1 - \pi(\theta))\frac{dQ}{dw_L} \tag{11}$$

a) Forward pass b) Backward pass

Fig. 1: During the forward pass (root to leaves), weights are calculated. During the backward pass (leaves to root), gradients are calculated.

During the backward pass we use (10) to compute the gradient w.r.t. the thresholds t_0 and t_1. In the experiments below we keep the features fixed, but it would also be possible to compute gradients of any parameters of the features.

We use the following algorithm to update the parameters of each node (t_0, t_1):

function UPDATENODES($\mathcal{X} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$)
  for all $\mathbf{x}$ in $\mathcal{X}$ **do**
    Feed $\mathbf{x}$ forward through tree to calculate weights
    Visit each node in reverse depth order - compute gradients
    Update estimate of mean gradient over batch
  end for
  Update parameters using mean gradient
end function

In the following the parameter update is made using a momentum term, but something more sophisticated could be used.

4 Experiments

Here we focus on two classification tasks related to knee osteoarthritis. The features used were shape, texture and appearance parameters extracted from lateral knee radiographic images (Fig. 2). Those features were obtained by first building a statistical appearance model [1] of the knee. This model was a PCA-based combination of statistical shape and texture models and was built on fully automated annotations found using a 3-stage Constrained Local Model [2].

Data. The images were taken from the Multicentre Osteoarthritis Study (MOST) dataset [3]. MOST is a longitudinal prospective study that collected data from 3026 participants with a 7-year follow-up. Lateral radiographs have been collected at each time-point for both knees and a KL (Kellgren-Lawrence) grade assessing the severity of the disease was assigned to each knee. For our binary classification tasks the grades were split into two groups: non-OA, KL (0,1), and the OA group, KL(2-4). The first task was OA classification, where the goal was to distinguish patients from the two groups, and used 8606 OA ($KL \geq 2$) and 10604 non-OA images. In the second task we considered 3478 baseline images with no OA ($KL \leq 1$) and aimed to discriminate those that would develop OA within 84 months from those that would not.

Knee OA classification tasks. We compared the performance of our Indecisive Forest (IF) with a standard Random Forest (RF) in 5-fold CV ex-

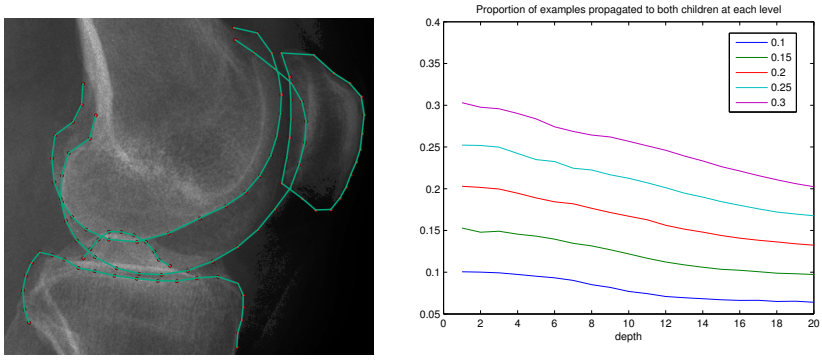

Fig. 2: An example of the landmark points used to build the appearance model (Left). Proportion of examples within the indecisive window at each level for different choices for window width. (Right)

periments. A parameter sweep suggested that a good choice for the parameter responsible for the width of indecision window was $r = 0.3$. In addition, we applied the tuning algorithm described above to optimise the IF, and evaluated the performance. We report the area under the ROC curve to evaluate each of the models in Table 1. This shows that for both classification tasks the IF performed better than a standard Random Forest, with an improvement of at least 2% for both classification and prediction. The optimisation improved the results for the OA classification task, while the OA prediction performance did not change significantly. Our results on both tasks achieved the state-of-the-art on the MOST dataset using only lateral knee radiographs (compared to [7]).

Figure 2 (Right) shows that the proportion of examples within the indecisive region increases when the window width increases and decreases linearly as examples go deeper in the trees.

Timings. The average time to train a standard tree on the prediction dataset was 9.3s, compared to 94.9s for each indecisive tree. The average tree optimisation time depended on the dataset and the parameter choice, ranging from 3s to 2 minutes. There was little difference in time taken when applying the trees.

	OA Classification	OA Prediction
Baseline Forest	86.35 ± 0.99	59.03 ± 1.20
IF	87.61 ± 0.94	$\mathbf{61.11 \pm 1.79}$
OIF	$\mathbf{88.15 \pm 0.91}$	59.11 ± 2.01

Table 1: AUC for the two knee OA tasks: comparing a standard Random Forest with both an Indecisive Forest (IF) and an Optimised Indecisive Forest (OIF).

5 Discussion and Conclusions

We have presented an improvement on the standard random forest that uses a ramp function with an ambiguous region to train and test decision trees. We showed improved performance, compared to a standard Random Forest, on two OA-related classification tasks. The combined leaf and node optimisation further improved the results on one of the tasks. The indecisive forests take longer to train and optimise. Pilot experiments on regression tasks have shown small but encouraging improvements, something that we will explore in future work.

6 Acknowledgments

The research leading to this results has received funding from EPSRC Centre for Doctoral Training grant 1512584. This publication also presents independent research supported by the Health Innovation Challenge Fund (grant no. HICF-R7-414/WT100936), a parallel funding partnership between the Department of Health and Wellcome Trust, and by the NIHR Invention for Innovation (i4i) programme (grant no. II-LB_0216-20009). The views expressed are those of the authors and not necessarily those of the NHS, NIHR, the Department of Health or Wellcome Trust.

References

1. T. F. Cootes, C. J. Taylor, et al. Statistical models of appearance for computer vision, 2004.
2. D. Cristinacce and T. F. Cootes. Feature detection and tracking with constrained local models. In *BMVC*, 2006.
3. D. Felson, J. Niu, T. Neogi, J. Goggins, M. Nevitt, F. Roemer, J. Torner, C. Lewis, A. Guermazi, and M. I. Group. Synovitis and the risk of knee osteoarthritis: the most study. *Osteoarthritis and Cartilage*, 24(3):458–464, 2016.
4. P. Kontschieder, M. Fiterau, A. Criminisi, and S. Bulo. Deep neural decision forests. In *International Conference on Computer Vision*, 2015.
5. C. Lindner, P. Bromiley, M. Ionita, and T. Cootes. Robust and accurate shape model matching using random forest regression-voting. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 37(9):1862–1874, 2015.
6. L. Minciullo and T. F. Cootes. Fully automated shape analysis for detection of osteoarthritis from lateral knee radiographs. In *ICPR*, 2016.
7. L. Minciullo, J. Thomson, and T. F. Cootes. Combination of lateral and pa view radiographs to study development of knee oa and associated pain. In *SPIE Medical Imaging*, pages 1013411–1013411. International Society for Optics and Photonics, 2017.
8. S. Ren, X. Cao, Y. Wei, and J. Sun. Global refinement of random forest. In *Computer Vision and Pattern Recognition*, 2015.
9. A. Suarez and J. Lutsko. Globally optimal fuzzy decision trees for classification and regression. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(12):1297–1311, 1999.