

Chapter 19

Entanglement and Differentiable Information Gain Maximization

A. Montillo, J. Tu, J. Shotton, J. Winn, J.E. Iglesias, D.N. Metaxas,
and A. Criminisi

Decision forests can be thought of as a flexible optimization toolbox with many avenues to alter or recombine the underlying architectural components and improve recognition accuracy and efficiency. In this chapter, we present two fundamental approaches for re-architecting decision forests that yield higher prediction accuracy and shortened decision time.

The first is entanglement, *i.e.* using the learned tree structure and intermediate probabilities computed in nodes closer to the root to affect the training of other nodes deeper in the trees. Unlike more conventional classifiers which assume that all data points (even those neighboring in space or time) are IID, the entanglement approach learns semantic correlation in non IID data. To demonstrate, we build an entangled decision forest (EDF) that exploits spatial correlation in human anatomy by simultaneously labeling voxels in computed tomography (CT) scans into 12 anatomical structures.

The second contribution is the formulation of information gain as a function that is differentiable with respect to the parameters of the split node weak learner. This provides increased confidence and accuracy of maximum margin boundary localization and reduces classification time by using a few, shallow trees. We further extend the method to incorporate training label confidence, when available, into the information gain maximization. Due to bagging and random feature subset selection, we can retain decision forest virtues such as resiliency to overfitting. To demon-

A. Montillo (✉) · J. Tu
General Electric Global Research, Niskayuna, NY 12309, USA

J. Shotton · J. Winn · A. Criminisi
Microsoft Research Ltd, Cambridge, UK

J.E. Iglesias
Massachusetts General Hospital, Harvard Medical School, Boston, MA, USA

D.N. Metaxas
Rutgers, Piscataway, NJ, USA

strate, we build a gradient ascent decision forest (GADF) that tracks visual objects in videos. For both approaches, superior accuracy and computational efficiency is shown in quantitative comparisons with state of the art algorithms.

19.1 Introduction

As discussed in Part I of this book, decision forests are a flexible framework for addressing diverse tasks, with many avenues to alter or recombine the underlying architectural components to improve accuracy and efficiency. In this chapter, we present two fundamental approaches for re-designing the decision forest. These lead to improved prediction accuracy, increased confidence and accuracy of maximum margin boundary localization, and reduced decision time and memory requirements for real world applications including semantic segmentation of 3D medical images and tracking objects in video.

19.2 Entangled Decision Forests

Our first approach, a re-architecting of decision forests, is the *entanglement* or sharing of information between the nodes in a decision forest. In entangled decision forests, the result of the binary tests applied at each tree node depends on the results of tests applied earlier during forest growth. This concept was first presented in [252] and later refined with context selectivity [250]. This chapter presents a more general exposition than reported previously, enabling the most broad interpretation and application.

Entanglement is *the use of the learned tree structure and intermediate probabilities associated with nodes in the higher levels of a tree to affect training of split nodes in deeper levels of the forest*. In its simplest incarnation one may think of entanglement as using the class posteriors of previously trained nodes as input feature into the training of subsequent nodes in the same tree.

A traditional assumption of many classifiers is that all data points (*e.g.* pixels in an image) are independent and identically distributed (IID). However, in many applications, this assumption is incorrect; many data points are in fact highly correlated and thus non IID. Entanglement automatically learns the semantic structural pattern of this correlation and encodes it in the features chosen during decision tree training. In practice, this correlation tends to occur over time, space or both. For example, in 3D medical image segmentation, human anatomy defines a canonical 3D configuration (correlation) over 3D space. In other cases, such as 4D medical scans, the correlation can be in both space and time (the fourth dimension). In entanglement, a tree node, j , at level, ℓ , in the forest is constructed by designing *entanglement* features that exploit the uncertain partial contextual information learned (or at test time, inferred) in a correlation neighborhood by the previous $\ell - 1$ levels of the forest (already trained). We call a forest that uses such features an entangled decision forest (EDF).

As an additional contribution, we randomly sample feature types and parameters from learned, non-uniform *proposal distributions* rather than from a uniform distribution used (implicitly) in previous decision forest research [5, 44, 77, 128, 212, 341, 411]. With this modification in place, the random draws from the proposal distribution select, with greater probability, the feature types and parameters that tend to be relevant for classification. As we will demonstrate, this allows for higher accuracy for the same number of features evaluated during training. Entanglement and learned proposal distributions allow faster training, and faster, more accurate prediction.

To illustrate entanglement, we discuss an example application where we wish to automatically segment a 3D Computed Tomography (CT) scan into its anatomical components such as the aorta, pelvis, and the lungs. We cast this task as a voxel classification problem which we solve via an EDF. In this case entanglement allows the class posteriors of voxels reaching nodes deep in the tree to depend directly from the intermediate posteriors attained higher up in the same tree. This improves accuracy and captures long-range *semantic* context. Previously, segmentation constraints in the form of semantic (*e.g.* anatomical) context have been applied, but these have required either a separate random field [342] or multi-pass processing [341, 375]; EDFs achieve this in one pass with no additional methods.

19.2.1 Entanglement Feature Design

We assume we are given a set, $\mathcal{S} = \{(\mathbf{v}, c)\}$, of voxels, $\mathbf{v} = (i, \mathbf{p})$, each consisting of its image intensity, i , (a measure of tissue density in the case of CT) voxel location \mathbf{p} and ground truth label, c . This set is formed from the collection of voxels from a group of training CT scans. Our goal is to infer the probability of each label for each voxel of unseen test scans.

Following the work in [78] we construct two types of long-range, context-aware feature. The first type captures “appearance context”, the latter are entangled and capture “semantic context”. See also Chap. 15. Details are explained next.

19.2.1.1 Appearance Features

Using the intensity image, J , we construct intensity features for each voxel \mathbf{v} that are spatially defined by (1) their position, \mathbf{p} , centered on the voxel to be labeled (Fig. 19.1a), and (2) one or two cuboidal probe regions, \mathbf{F}_1 and \mathbf{F}_2 , offset by displacement vectors, $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$, which can be up to 200 mm in each dimension (x, y, z). A probe region, $\mathbf{F}(\mathbf{q}; \mathbf{w})$, is the set of voxels within the region centered at \mathbf{q} with side lengths, \mathbf{w} . We construct two variants of intensity features. The first variant consists of the mean CT intensity at a probed region, \mathbf{F}_1 (Fig. 19.1a, left), while the second consists of the difference in the mean intensity of regions, \mathbf{F}_1 and

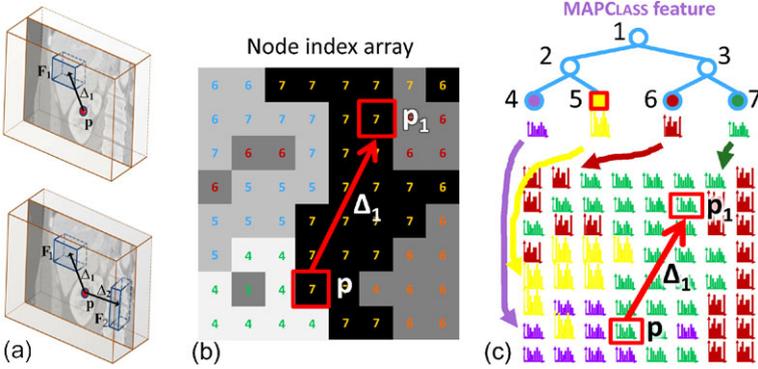


Fig. 19.1 Intensity and entanglement features. (a) Intensity features measure image information from regions offset from the reference voxel at \mathbf{p} . (b) MAPCLASS feature retrieves the label that the classifier currently predicts at location \mathbf{p}_1 offset from \mathbf{p} . We maintain a node index array which associates with each voxel the current tree node ID (represented by the number in each voxel). (c, top) The array allows to determine the current label posterior in the tree for the voxel at location \mathbf{p}_1 . (c, bottom) Conceptually, the tree induces a vector image of class posteriors which we use when designing MAPCLASS and TOPNCLASSES features

\mathbf{F}_2 (Fig. 19.1a, right). Then split functions are defined from these as follows:

$$h_{\text{INTENSITY}}(\mathbf{v}, \theta_j) = [\bar{J}(\mathbf{F}_1(\mathbf{p} + \Delta_1)) > \tau], \quad (19.1)$$

$$h_{\text{INTENSITYDIFF}}(\mathbf{v}, \theta_j) = [\bar{J}(\mathbf{F}_1(\mathbf{p} + \Delta_1)) - \bar{J}(\mathbf{F}_2(\mathbf{p} + \Delta_2)) > \tau]. \quad (19.2)$$

During training, each type of split function is characterized for node j by the split parameters $\theta_j = (\phi, \tau)$. For $h_{\text{INTENSITY}}$, ϕ includes the parameters of \mathbf{F}_1 : the offset Δ_1 , the size \mathbf{w}_1 and an intensity threshold τ . For $h_{\text{INTENSITYDIFF}}$, ϕ includes the additional parameters Δ_2 and \mathbf{w}_2 . These parameters are sampled randomly during training for each split node. Once training has finished, the maximum information gain node test along with its optimal features are frozen and stored within the node for later use during testing.

19.2.1.2 Semantic Context Entanglement Features

We now describe an instance of our entanglement contribution. During testing on novel images, we exploit the confident voxel label predictions (peaked distributions) that can be found using early levels of the forest to aid the labeling of nearby voxels. This provides semantic context similar to auto-context [341, 375], but does so within a single forest. We define four types of long-range entanglement feature to help train the node currently being grown using knowledge learned in already trained nodes of the forest. Two features (MAPCLASS and TOPNCLASSES) are based on the posterior class distribution of the nodes corresponding to probed voxels, and two (NODEDESCENDANT and ANCESTORNODEPAIR) are based on the location of the nodes within the trees.

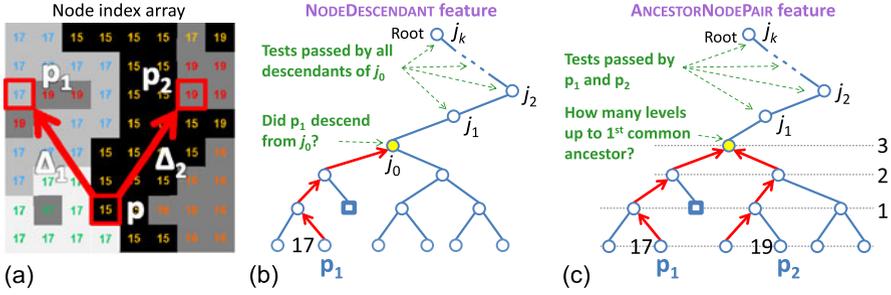


Fig. 19.2 Further entanglement features. (a) Node index array associates voxels with intensity and tree node indices (same format as Fig. 19.1b but for a deeper tree level). (b) NODEDESCENDANT feature tests whether probe voxel at \mathbf{p}_1 descends from a node (j_0 in this case). (c) ANCESTORNODEPAIR feature tests whether the nodes of voxels \mathbf{p}_1 and \mathbf{p}_2 have a common ancestor $< \tau$ levels away

MAPCLASS Entanglement Features As the name suggests, this type of feature uses the maximum a posteriori label of a neighboring voxel at \mathbf{p}_1 to reduce uncertainty about the label at \mathbf{p} (Fig. 19.1b). When such semantic context is helpful to classify the voxel at \mathbf{p} , the feature yields high information gain and may become the winning feature for the node during tree growth. The MAPCLASS split function tests whether the MAP class in the posterior of a probed voxel $\mathbf{p}_1 = \mathbf{p} + \Delta_1$ is equal to a particular class c^* :

$$h_{\text{MAPCLASS}}(\mathbf{v}, \theta_j) = \left[\arg \max_c p(c; j(\mathbf{p}_1)) = c^* \right]. \quad (19.3)$$

The parameter θ_j includes $\phi = (\Delta_1, c^*)$ while $p(c; j(\mathbf{p}_1))$ is the posterior class distribution of the node of \mathbf{p}_1 denoted $j(\mathbf{p}_1)$. This posterior can be retrieved from the tree because (1) we train and test voxels in breadth-first fashion, and (2) we maintain an association between voxels and the tree node ID at which they reside while moving down the tree. This association is a node index array (Fig. 19.1b).

TOPNCLASSES Entanglement Features Similarly we define features, called TOPNCLASSES, where $N \in \{2, 3, 4\}$, that generalize the MAPCLASS feature. A TOPNCLASSES feature tests whether a particular class c^* is in the top N classes of the posterior class distribution of the probe voxel at $\mathbf{p}_1 = \mathbf{p} + \Delta_1$. The split function, with parameter θ_j including $\phi = (\Delta_1, N, c^*)$ is defined as

$$h_{\text{TOPNCLASSES}}(\mathbf{v}, \theta_j) = \left[c^* \in \text{top } N \text{ classes of } p(c; j(\mathbf{p}_1)) \right]. \quad (19.4)$$

NODEDESCENDANT Entanglement Features This type of feature tests whether a region near voxel \mathbf{p} has a particular appearance. The neighboring region is centered at voxel \mathbf{p}_1 (Fig. 19.2a, b). The split test is whether the node currently corresponding to \mathbf{p}_1 descends from a particular tree node, j_0 . If it does, then we know \mathbf{p}_1 has satisfied the appearance tests at nodes ($j_1 \dots j_k$) above j_0 in the tree in a particular way to arrive at j_0 .

ANCESTORNODEPAIR Entanglement Features This type of feature tests whether two regions near voxel \mathbf{p} have passed similar appearance and semantic tests. The neighboring regions are centered at voxels \mathbf{p}_1 and \mathbf{p}_2 (Fig. 19.2a). The split test is whether the nodes currently corresponding to \mathbf{p}_1 and \mathbf{p}_2 have their first common ancestor $< \tau$ tree levels above the current level (Fig. 19.2c). The threshold controls the required degree of similarity: the lower τ , the greater the required appearance and context similarity needed to pass the test, because the lower τ , the larger the number of tests with identical outcomes above the common ancestor.

19.2.2 Guiding Feature Selection by Learned Proposal Distributions

This section describes the use of learned proposal distributions. These distributions aim to match the feature types and their parameters proposed at each tree node during training to those that have proven to be most useful for classification in a previous training run. The decision forest still chooses the winning feature, but each node chooses from features sets that are likely to be useful based on prior experience. Specifically, we train an initial decision forest, F_{temp} , on our training data, using a uniform proposal distribution. We then record (as histograms) the distribution of accepted feature parameters and feature types across all tree nodes in the forest. F_{temp} is then discarded, and we then use parameter distributions as the proposal distributions in a subsequent training of the next decision forest. While this requires additional training, it imposes no time penalty for prediction. This process could be repeated, though in practice even just one iteration has proven sufficient for a substantial improvement in accuracy (*e.g.* $> 5\%$).

The learned displacements tends to be Gaussian distributed and centered on the reference voxel (Fig. 19.3 top row). Acceptance distributions of the remaining parameters, such as the thresholds τ or the choice of the MAPCLASS class c^* , also have non-uniform distributions (Fig. 19.3 bottom row). Similarly, the distribution of feature types for each tree level is learned. Drawing feature types from this distribution can also improve classifier accuracy. Figure 19.4a shows how the ratio of feature types varies with tree depth. As the tree is grown, entanglement features increasingly dominate the scene over the more conventional intensity features. The entangled features used by the nodes in the lower part of the tree exploit semantic context and neighborhood consistency inferred from appearance features of earlier levels.

19.2.3 Results

We evaluate our EDF model on the task of segmenting a database of 250 varying field of view CT scans. Each voxel in each CT scans needs be assigned one of 12 class labels from the following set of anatomical structures of interest {heart, liver,

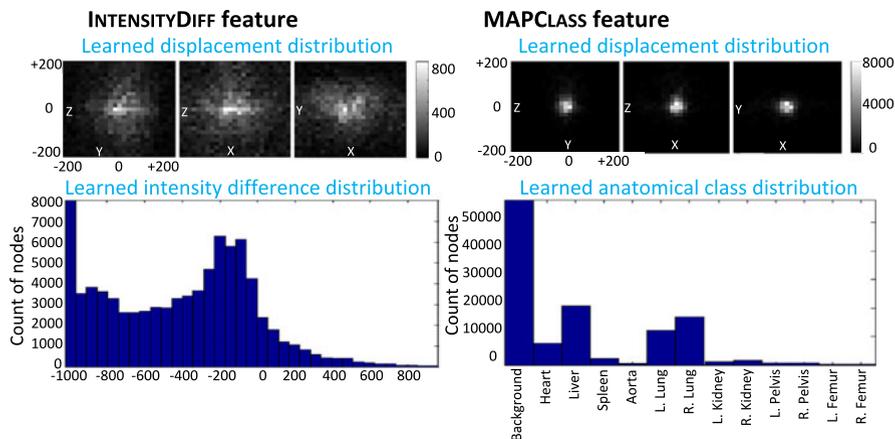


Fig. 19.3 Learned parameter distributions are clearly non-uniform. (Left) Learned displacement and anatomical class distributions for MAPCLASS feature. (Right) Displacement and intensity difference distributions for INTENSITYDIFF feature

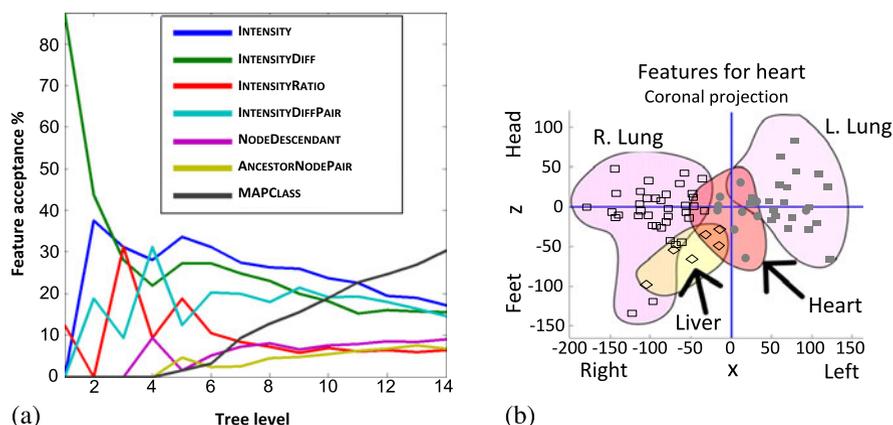


Fig. 19.4 An EDF reveals how and what it has learned. (a) Learned relative proportion of feature types chosen at each level of forest growth. (b) Location and organ class of the top 50 features used to identify heart voxels. The hand-drawn regions here group these locations for different MAPCLASS classes c^*

spleen, aorta, l./r. lung, l./r. femur, l./r. pelvis, l./r. kidney} or the background class. This database has been designed to include wide variations in patient health status, field of view and scan protocol. We randomly selected 200 volumes for training and 50 for testing.

Qualitative Results The EDF achieves a visually accurate segmentation of organs throughout the 50 test volumes. Example segmentations are shown in Fig. 19.5a where the first column is the ground truth segmentation, and the sec-

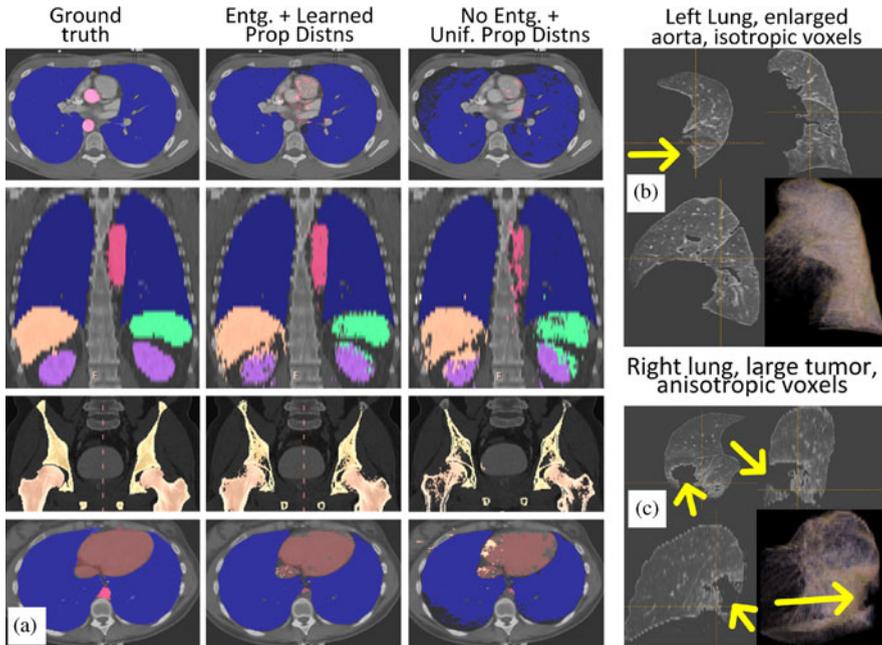


Fig. 19.5 Qualitative segmentation results. (a) The use of entanglement and learned proposal distributions (*column 2*) improves accuracy compared to not using them (*column 3*). The *rows* show four different subjects. (b) EDF segmented left lung distorted by enlarged aorta; volume rendering in *lower right*. (c) EDF accurately segments a right lung despite a severe tumor

ond column is the EDF result. We see good agreement for the lungs (blue), liver (orange), spleen (green), kidneys (purple), femur (tan), and heart (dark brown). Column 3 shows the result using our decision forest without entanglement and with uniform proposal distributions. Entanglement with proposal distributions noticeably improves the lungs, aorta (red), kidneys, spleen, femur, and heart.

The algorithm handles many complexities commonly found in the clinic. For example, our algorithm correctly segmented the lung despite the case of a severely enlarged aorta (Fig. 19.5b) and another with a tumor (Fig. 19.5c).

Quantitative Impact of Each Contribution For a quantitative analysis we measured segmentation accuracy across all 50 test scans using the average class Jaccard similarity coefficient [100]. The metric is the ratio of the intersection size (ground truth and predicted labels) divided by the size of their union. While EDF achieves $> 97\%$ average voxel accuracy throughout our database, we use the Jaccard metric because we feel it is a more reliable metric of segmentation accuracy.

To understand the impact of using the acceptance distributions as proposal distributions (Sect. 19.2.2), we trained the decision forest in four different ways: (1) using uniform feature type and uniform feature parameter distributions for baseline performance (light blue curve, Fig. 19.6a), (2) using learned feature type distribution

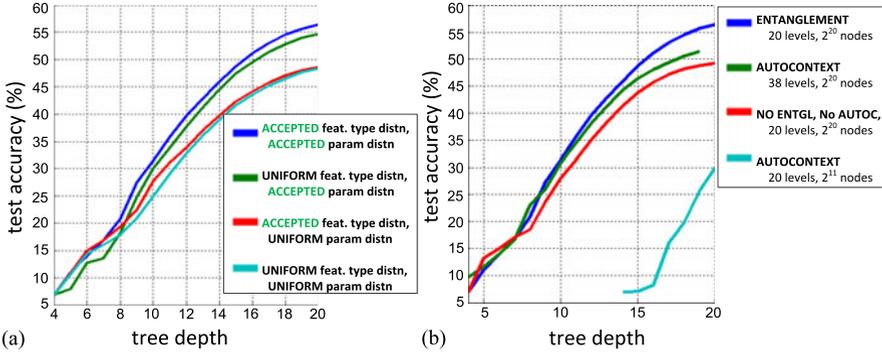


Fig. 19.6 Quantitative impact of each contribution. **(a)** Learning proposal distributions for both feature types and feature parameters increases accuracy. **(b)** Entanglement (*dark blue*) provides greater accuracy and prediction speed than auto-context (*green*). Note: the *green* curve should properly be plotted at depths 20–38, but for ease of comparison we plot it at depths 1–19

with uniform feature parameter distributions (red), (3) using uniform feature type distributions with learned feature parameter distributions (green), (4) using learned feature type and learned parameters distributions (dark blue). Learning only the feature type distribution yields a negligible improvement to baseline (red vs. light blue). Learning feature parameter distribution boosts accuracy significantly (green vs. red). Learning both yields the best performance boosting accuracy over baseline by 8 %.

We compared our method to auto-context [341, 375] by conducting four experiments. First, we trained our decision forest 20 levels deep without entanglement and without auto-context for a baseline (red, Fig. 19.6b). Second, we trained a two-round, auto-context decision forest (ADF) using 10 levels in each round (light blue). Third, we trained another ADF, but this time with an equal *modeling capacity* to the baseline by using two rounds with 19 levels each (green). Fourth, we trained the proposed EDF method as a single, 20 level deep forest using entanglement (dark blue curve). We find considerably better accuracy using the EDF method (dark blue vs. green). In addition to beating the performance of ADF, it reduces the prediction time by 47 % since the EDF requires 18 fewer levels (20 vs. 38).

Efficiency Considerations With a parallel implementation, EDF segments volumes ($512 \times 512 \times 424$) in just 12 seconds per volume using an 8 core Xeon 2.4 GHz computer with 16 GB RAM. This speed is equal to or better than state of the art single organ methods [419], yet we segment *multiple* (12) organs simultaneously.

Inspecting the Chosen Features Figure 19.4b shows how the MAPCLASS feature learns to segment a heart voxel located at the cross-hair. To find the top contributing semantic context features, we express information gain as a sum of the information gain from each class:

$$I(\mathcal{S}, \theta) = \sum_{c \in \mathcal{C}} \left(-p(c|\mathcal{S}) \log p(c|\mathcal{S}) + \sum_{i \in \{L, R\}} \left(\frac{|S^i|}{|\mathcal{S}|} p(c|S^i) \log p(c|S^i) \right) \right), \quad (19.5)$$

where \mathcal{S} is the set of voxels being split into partitions \mathcal{S}^L and \mathcal{S}^R , and c is the index over classes. We can then readily rank the learned node features based on how much they contributed to classifying the voxels of a given class (*e.g.* heart) by increasing the information gain for that class.

19.3 Differentiable Information Gain Maximization

In Sect. 19.2 we simultaneously increased classification accuracy and reduced decision time using entanglement which propagates knowledge from one part of the forest to another. In this section we achieve a similar result using a complementary approach. This second approach optimizes training by applying gradient ascent to differentiable information gain. Finding the optimum parameters for the split tests has traditionally [302] been achieved via exhaustive discrete search to find those parameters which maximize information gain. For a given computational budget, exhaustive search is limited to a small region of parameter space or a coarse quantization of a wider region.

By making information gain differentiable, we can directly find the optimal data partition for the given input subset and node feature subset. This produces more compact decision trees which in turn reduces classification (test) time and memory requirements. Through the use of random input (bagging) and feature subset selection, decision forest virtues including independent trees and resiliency to overfitting can be retained.

Using non-differentiable information gain, an optimal solution can be found by simulated annealing techniques [158, 260], though this can be computationally impractical in high dimensional feature spaces. Alternative discriminative criteria could be optimized, such as LDA [246], SVM [383], or boosting techniques [374, 413], but these may not provide the optimal data partitions when the data distributions are from many classes.

As discussed in Sect. 3.3, binary tests based on parameterized functionals (such as hyperplanes or conic sections) are stronger learner models than coordinate aligned split functions, and can have greater generalization capabilities. We show below that our differentiable information gain forest both accepts these more powerful split functions and improves their generalization power to approximate the maximum margin decision boundary (see also [80] for a detailed discussion about maximum margin behavior in decision forests). Like [173], we impose a soft split using a sigmoid function; however, we explicitly present the derivation absent in [173], and extend it to include label confidence when training data include label uncertainty. We also incorporate hyperplane and non-linear split tests in the gradient ascent framework.

We call the resulting classifier a gradient ascent decision forest (GADF) and illustrate its power in both synthetic and real-world examples. First, we investigate the impact of the GADF for a synthetic 2D classification task, and demonstrate that gradient ascent reduces classification time, increases prediction accuracy and increases

confidence in the estimation of the maximum margin decision boundary, compared to the standard decision forest. Second, we implement a GADF to solve classification problems in several application domains including mass spectrometry, biomechanics, botany, image classification and 1D signal processing. We demonstrate how the GADF approach increases prediction accuracy across this application spectrum. Third, we cast visual object tracking as an iterative classification task and train a gradient ascent classifier to perform object tracking in public PET videos. We show how the approach avoids tracker drift and handles severe occlusions better than state of the art trackers.

19.3.1 Formulating Differentiable Information Gain

Given the labeled dataset $\mathcal{S} = \{(\mathbf{v}, c)\}$ of size N , where \mathbf{v} is the feature data of dimension d , and $c = \zeta(\mathbf{v})$ is the ground truth class label of \mathbf{v} (with $c \in \{1, \dots, C\}$), the Shannon entropy of the class distribution can be computed as

$$H(\mathcal{S}) = - \sum_{c=1}^C p(c|\mathcal{S}) \log p(c|\mathcal{S}), \quad (19.6)$$

where

$$p(c|\mathcal{S}) = \frac{\sum_{\mathbf{v}} [\zeta(\mathbf{v}) - c]}{N} \quad (19.7)$$

defines the data class distribution and $[\cdot]$ is the indicator function.

Given a binary split function (weak learner) $h(\cdot, \cdot)$, we can partition the data into two subsets (see Sect. 3.2.3):

$$\mathcal{S}^L = \{\mathcal{S}|h = 1\} = \{(\mathbf{v}, c) | \mathbf{v} \in \mathcal{S}, h(\mathbf{v}; \boldsymbol{\theta}) = 1\} \quad (19.8)$$

of size N^L and

$$\mathcal{S}^R = \{\mathcal{S}|h = 0\} = \{(\mathbf{v}, c) | \mathbf{v} \in \mathcal{S}, \bar{h}(\mathbf{v}; \boldsymbol{\theta}) = 1 - h(\mathbf{v}; \boldsymbol{\theta}) = 1\} \quad (19.9)$$

of size $N^R = N - N^L$. The information gain defines the entropy change before and after the split h is applied:

$$I(\mathcal{S}|h) = H(\mathcal{S}) - H(\mathcal{S}|h), \quad (19.10)$$

where the entropy after partitioning is computed as

$$H(\mathcal{S}|h) = \frac{N^L}{N} H(\mathcal{S}^L) + \frac{N^R}{N} H(\mathcal{S}^R). \quad (19.11)$$

Information gain is a typical measure for selecting discriminative weak learners in decision tree training as described in Chap. 3. However, the information gain formulation is not differentiable w.r.t. the parameters $\boldsymbol{\theta}$ of h , making analytical optimization problematic.

To make $I(\mathcal{S}|h)$ in (19.10) differentiable w.r.t. the binary test h , we first define the split function h for data point \mathbf{v} as a parameterized functional:

$$h_{\psi}(\mathbf{v}; \boldsymbol{\theta}) = \begin{cases} 0, & \psi(\mathbf{v}; \boldsymbol{\theta}) < 0, \\ 1, & \psi(\mathbf{v}; \boldsymbol{\theta}) \geq 0, \end{cases} \quad (19.12)$$

where $\psi(\mathbf{v}; \boldsymbol{\theta})$ is the geometric split function of feature space with parameter set $\boldsymbol{\theta}$. The partition occurs at the boundary $\psi(\mathbf{v}; \boldsymbol{\theta}) = 0$.

We then define partition integrals for each class for all data w.r.t. h as follows:

$$U_c^{\mathcal{S}}(h) = \sum_{\mathbf{v}} h(\mathbf{v}; \boldsymbol{\theta}) [\zeta(\mathbf{v}) - c], \quad c \in \{1, \dots, C\}, \quad (19.13)$$

$$U^{\mathcal{S}}(h) = \sum_{\mathbf{v}} h(\mathbf{v}; \boldsymbol{\theta}), \quad (19.14)$$

where $h(\mathbf{v}; \boldsymbol{\theta})$ can be replaced with $\bar{h}(\mathbf{v}; \boldsymbol{\theta}) = 1 - h(\mathbf{v}; \boldsymbol{\theta})$ as needed.

We can then define $N^{\text{L}} = U^{\mathcal{S}}(h)$, $N^{\text{R}} = U^{\mathcal{S}}(\bar{h})$, $N = N^{\text{L}} + N^{\text{R}}$, $p_c(\mathcal{S}^h) = \frac{U_c^{\mathcal{S}}(h)}{U^{\mathcal{S}}(h)}$ and $p_c(\mathcal{S}^{\bar{h}}) = \frac{U_c^{\mathcal{S}}(\bar{h})}{U^{\mathcal{S}}(\bar{h})}$, and the entropy after partition by h is

$$\begin{aligned} H(\mathcal{S}|h) = & -\frac{1}{N} \left(\sum_c U_c^{\mathcal{S}}(h) \log U_c^{\mathcal{S}}(h) - U^{\mathcal{S}}(h) \log U^{\mathcal{S}}(h) \right. \\ & \left. + \sum_c U_c^{\mathcal{S}}(\bar{h}) \log U_c^{\mathcal{S}}(\bar{h}) - U^{\mathcal{S}}(\bar{h}) \log U^{\mathcal{S}}(\bar{h}) \right). \end{aligned} \quad (19.15)$$

Using the chain rule, the derivative of information gain w.r.t. $\boldsymbol{\theta}$ is

$$\begin{aligned} \frac{\partial I}{\partial \boldsymbol{\theta}} = & -\frac{\partial H(\mathcal{S}|h)}{\partial \boldsymbol{\theta}} \\ = & \frac{1}{N} \left(\sum_c U_c^{\mathcal{S}}(h) (\log U_c^{\mathcal{S}}(h) + 1) - U^{\mathcal{S}}(h) (\log U^{\mathcal{S}}(h) + 1) \right. \\ & \left. + \sum_c U_c^{\mathcal{S}}(\bar{h}) (\log U_c^{\mathcal{S}}(\bar{h}) + 1) - U^{\mathcal{S}}(\bar{h}) (\log U^{\mathcal{S}}(\bar{h}) + 1) \right), \end{aligned} \quad (19.16)$$

where $U_c^{\mathcal{S}}(h) = \sum_{\mathbf{v}} \frac{\partial h(\mathbf{v}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} [\zeta(\mathbf{v}) - c]$, $c \in \{1, \dots, C\}$, and $U^{\mathcal{S}}(h) = \sum_{\mathbf{v}} \frac{\partial h(\mathbf{v}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$.

Information gain is not differentiable w.r.t. the binary test parameter $\boldsymbol{\theta}$ because $h_{\psi}(\mathbf{v}; \boldsymbol{\theta})$ in (19.12) is not differentiable. To make it differentiable, we approximate the weak learner h by a sigmoid function $h_{\psi}(\mathbf{v}; \boldsymbol{\theta}) = 1/(1 + e^{-\frac{\psi(\mathbf{v}; \boldsymbol{\theta})}{\sigma}})$ to get:

$$\frac{\partial h_{\psi}(\mathbf{v}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{\sigma} h_{\psi}(\mathbf{v}; \boldsymbol{\theta}) (1 - h_{\psi}(\mathbf{v}; \boldsymbol{\theta})) \frac{\partial \psi(\mathbf{v}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \quad (19.17)$$

Combining (19.16) and (19.17) allows us to compute the derivative of information gain w.r.t. the binary test function parameter $\boldsymbol{\theta}$ using the chain rule. The split

function $\psi(\mathbf{v}; \theta)$ can be designed according to the purpose of information gain optimization. It is worth noting that the parameter σ defines the fidelity of the binary test, and controls the smoothness of the information gain surface in the decision boundary parametric space. One may apply annealing to σ when doing gradient ascent (*i.e.* letting $\sigma \rightarrow 0$) so that the chance that the optimization reaches global maxima can be increased.

Soft Label Decision Forests If each training data point \mathbf{v} also includes a (training) class label probability measure $q_c(\mathbf{v})$, then we define a confidence score $\gamma(\mathbf{v}) \in [0, 1]$ as a function of the label log-likelihood $l(\mathbf{v}) = \log(\frac{q_c(\mathbf{v})}{1-q_c(\mathbf{v})})$ as follows:

$$\gamma(\mathbf{v}) = \frac{2}{1 + e^{-\frac{\sqrt{|l(\mathbf{v})|} - t_l}{\sigma_l}}}. \quad (19.18)$$

The intuition is that the label is less confident if the class probability ratio is too close to 1 (and thus $\sqrt{|l(\mathbf{v})|}$ approaches zero), and σ_l controls how sensitive the confidence score is to the log-likelihood-ratio score. Such class label probability measures occur naturally in tasks such as video processing. In this case an on-line model learning may be applied per frame. Given the classification model trained on-line using the previous frames, the new observations in the current frame may be labeled with likelihood confidence (soft labels), and become the training data for the on-line model updating for the current frame. In [52], such ‘soft label decision forests’ are used for on-line tracking in videos where labels are quantized into a histogram and a standard node training procedure is applied.

We provide an analytic solution for the soft label decision forests learning problem. By modeling the label confidence measures based on how much the label likelihood deviates from the decision threshold, we derive a differentiable information gain formulation weighted by the label confidence. Our gradient ascent optimization technique can then be applied to find the optimal data split based on the information gain criteria with respect to the known class labels. Specifically, to optimize the information gain with emphasis on data \mathbf{v} labeled with high confidence, we can simply derive the differentiable information gain by weighting the terms in (19.13) with the confidence measure:

$$U_c^{\mathcal{S}}(h) = \sum_{\mathbf{v}} h(\mathbf{v}; \theta) [\zeta(\mathbf{v}) - c] \gamma(\mathbf{v}), \quad c \in \{1, \dots, C\}, \quad (19.19)$$

$$U^{\mathcal{S}}(h) = \sum_{\mathbf{v} \in \mathcal{S}} h(\mathbf{v}; \theta) \gamma(\mathbf{v}). \quad (19.20)$$

19.3.2 Split Function Design and Gradient Ascent Optimization

In training a classification forest, we solve for a decision boundary, $\psi_{\text{DF}}(\mathcal{S}; \theta_{\text{DF}})$, optimally partitioning the data \mathcal{S} with maximal information gain:

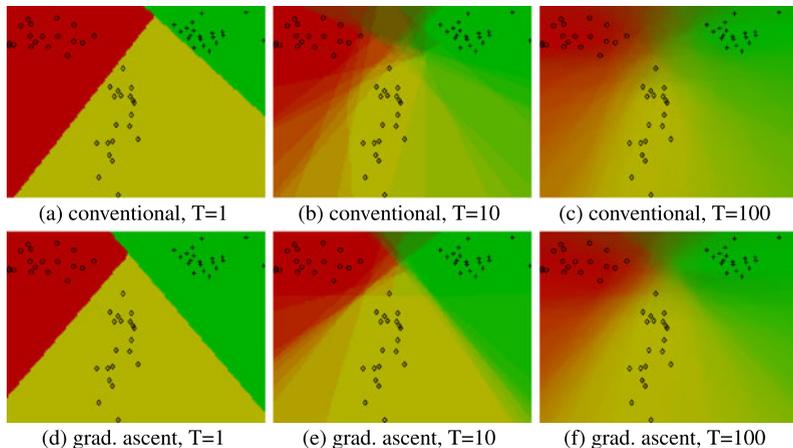


Fig. 19.7 Gradient ascent can improve the location and confidence in the maximum margin decision boundary and reduce classification time. *Top row: discrete optimization* using: (a) 1 tree, (b) 10 trees, (c) 100 trees. *Bottom row: discrete optimization followed by gradient ascent optimization* using: (d) 1 tree, (e) 10 trees, (f) 100 trees

$$\theta_{\text{DF}}^* = \arg \max_{\theta_{\text{DF}}} I(\mathcal{S} | h(\mathbf{v}; \theta_{\text{DF}})), \quad (19.21)$$

where θ_{DF} is the concatenated vector of the binary test parameters θ at each tree node.

The classic decision forest [44] uses a univariate split test, which consists of a threshold τ of the k th feature element of \mathbf{v} . We can denote this partitioning boundary function as $\psi = g_0(\mathbf{v}; \theta) = v_k - \tau$ with $\theta = \theta_0 = \{k, \tau\}$. Such a boundary is well suited for fast discrete search via maximizing I . However, the boundary coordinate alignment in feature space can require many binary tests to make the joint decision boundary $\psi_{\text{DF}}(\mathcal{S}; \theta_{\text{DF}})$ approximate the maximum margin class separation boundary (as discussed in Chap. 3). An alternative is to approximate the maximal margin decision boundary using far fewer but stronger weak learner models, such as hyperplanes or conic functions, and this is especially true if differentiable information gain is used. This is vital since maximum margin fidelity is likely to endow the forest with superior generalization properties.

To illustrate, Fig. 19.7 shows synthetic 2D data points from three classes (red, yellow, green) and the resulting partitions of a 2D feature space (x_1, x_2) by different decision forests architectures. The top row shows the results of applying a conventional classification forest. The bottom row shows the results (for corresponding forest size) of a classification forest using our differentiable information gain. While it has been shown in [80] (and in Chap. 4) that when a large number of trees are used the maximum margin class boundaries can be found, in practice, a small number of trees is typically preferred for either classification runtime efficiency or memory constraints. A comparison between the two rows in Fig. 19.7 shows that our new formulation of information gain allows forests to approximate maximum margin

behavior accurately even with just a few trees. In fact, in each column, showing the results on various number of trees, we see an improvement in the maximum margin boundary.

In detail, Fig. 19.7a shows the result of a decision forest with 1 tree and oriented-line weak learners. We observe that the decision boundary does not approximate well the maximum margin decision boundary. Averaging the output of 10 trees, Fig. 19.7b, starts to improve the location of the class boundary. Using 100 trees Fig. 19.7c provides a reasonable approximation to the maximum margin location and a smooth transition class posterior.

Using gradient ascent optimization yields improved location of the class boundary even for just one tree (Fig. 19.7d). Here, the method is initialized with the result from Fig. 19.7a. Figure 19.7e shows the result when the output from 10 gradient ascent trained trees are averaged. Compared to Fig. 19.7b we can see the confidence in the correct maximum margin boundary location is improved and a smoother posterior. Similarly, when the output from 100 gradient ascent trained trees are averaged in Fig. 19.7f, an improvement in the confidence of the correct maximum margin decision boundary is still observed.

The improvement in maximum margin fidelity obtained by using GADF can provide additional generalization when training data are limited, which is often the case in practice. The use of fewer trees also substantially speeds up classification time, since each gradient ascent trained tree does not require additional time to test yet provides increased accuracy. For example, the gradient ascent based result in Fig. 19.7e has similar maximum margin fidelity to the non-gradient ascent result in Fig. 19.7c yet requires 10 times fewer trees. In additional 2D synthetic tests we have also observed a large capture range (basin of convergence) for both oriented hyperplane and conic weak learners when using GADF.

With this motivation for differentiable information gain, we derive the gradient of two useful binary tests for gradient ascent as follows:

Hyperplane Partition Binary test functionals in the form of hyperplanes for the training of decision forest can be defined as $\psi(\mathbf{v}; \boldsymbol{\theta}) = g_1(\mathbf{v}; \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix}$ where $\boldsymbol{\theta} = \theta_1 = [\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(d+1)}] \in \mathbb{R}^{d+1}$ may be directly incorporated into the proposed gradient ascent information gain optimization where we can show that

$$\frac{\partial \psi}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix}. \quad (19.22)$$

Hyper-Ellipsoid Partition Binary tests with hyper-ellipsoid split functionals in \mathbb{R}^d can be defined as $\psi(\mathbf{v}; \boldsymbol{\theta}) = g_2(\mathbf{v}; \boldsymbol{\theta}) = 1 - (\mathbf{v} - \mathbf{v}_0)^\top \mathbf{Q}(\mathbf{v} - \mathbf{v}_0)$ where \mathbf{v}_0 is the ellipsoid center, and \mathbf{Q} defines the semi-axes lengths and orientations. To incorporate into our gradient ascent information gain optimization, we have

$$\frac{\partial \psi}{\partial \mathbf{v}_0} = -2(\mathbf{v} - \mathbf{v}_0)^\top \mathbf{Q} \quad \text{and} \quad \frac{\partial \psi}{\partial \mathbf{Q}} = (\mathbf{v} - \mathbf{v}_0)(\mathbf{v} - \mathbf{v}_0)^\top \quad (19.23)$$

with the optimization subject to $\mathbf{Q} > 0$.

To train each node using gradient ascent information gain optimization, an initial estimate of the split function parameters can be chosen by random guess or discrete optimization in discrete parameter space. For example, for hyperplanes we find the best simple binary test with parameter $\theta_0^* = \{k^*, \tau^*\}$ by discrete search and then set the initial guess $\theta_1 = [0, \dots, 0, \theta_1^{(k^*)} = 1, 0, \dots, -\tau^*]^\top$. Given the formulation and initial guess, we can conveniently implement the gradient ascent optimization by adopting existing off-the-shelf gradient ascent optimization toolboxes (*i.e.* `fminunc()` in Matlab with default options).

19.3.3 Object Tracking via Information Gain Maximization

Reliable visual tracking of a target object is difficult as many confusing factors need to be addressed including: occlusions, distractions from background clutter, and object appearance variations. We cast object tracking as an iterative classification problem [9, 11, 139, 178], and model the on-line appearance model update and tracking as a sequential process of information gain maximization that partitions the pixels in feature space (image features) and in image space (incorporating pixel coordinates as additional features) iteratively. Various decision forest based visual trackers have been proposed in the literature. A popular approach is to use classification forests to construct an appearance likelihood model that is updated on-line in the current frame and evaluated for the next frame. Tracking is achieved by finding the maxima of the confidence map for the next frame by picking the centroid. As shown in [317], the on-line decision forest model based visual tracker consistently outperformed that based on an on-line Adaboost model. In [120], the concept of Hough forests is proposed. With Hough forests, the target center is detected and tracked by the fusion of generalized Hough transforms that are based on the codebook classification of local image patches. Also, Chap. 12 and Chap. 16 provide further forest-based video tracking algorithms.

Our approach is substantially different from the previous approaches. We consider the tracking as an *information gain maximization process* (Gain-Max tracking) in pixel XY-coordinate space. By parameterizing the target shape with an ellipsoid, the tracking of the target location and scale can be achieved by maximizing the differentiable information gain via gradient ascent techniques. We note that while we train a forest for each frame, the amount of data is small and thus a forest can be trained quickly. Realtime computation can be achieved by sacrificing some model optimality (*i.e.* limiting the number of trees), or by adopting on-line decision forest learning techniques [317].

Given an image J , we define a region of interest as $\Omega(J)$ and denote the target region as $\Omega^+(J)$ and background region $\Omega^-(J)$ such that $\Omega(J) = \Omega^+(J) \cup \Omega^-(J)$. The pixel feature vector at location \mathbf{p} is a d dimensional vector denoted $J(\mathbf{p}) \in \mathbb{R}^d$ where J has channels for textures, RGB colors, gradients, and wavelets. We denote target and background pixel labels as $\{F, B\}$. We use information gain maximization in two ways: first, to learn to discriminate foreground from background based

on pixel appearance (feature space) and second, to track the target in the pixel XY-coordinate space (image space). These are explained next.

GainMax in Feature Space: Updating the Appearance Model To discriminate foreground from background pixels, we train a two-category pixel classifier that assigns pixels with a label from $\{F, B\}$. When information gain maximization is achieved, solving (19.21) the classifier learns the image features that best separate the training data: $\mathcal{S}^J = \{(J(\mathbf{p}), \zeta(\mathbf{p})) \mid \mathbf{p} \in \Omega(J), J(\mathbf{p}) \in \mathbb{R}^d, \zeta(\mathbf{p}) \in \{F, B\}\}$ into foreground and background. The features are computed directly from the image while the target and background labels come from a prior frame (the initial frame is assumed manually labeled). For example, we can obtain the label of the pixel at location \mathbf{p} as

$$\zeta(\mathbf{p}) = \begin{cases} F & \text{if } \mathbf{p} \in \Omega^+(J), \\ B & \text{if } \mathbf{p} \in \Omega^-(J). \end{cases} \quad (19.24)$$

GainMax in Image Space: Tracking the Target Further optimization of the foreground and background is possible if we take into consideration each pixel's XY-coordinates in image space in addition to the pixel foreground and background labels output from the previous step's two-category classifier. We denote such a training dataset as $\mathcal{S}^{\Omega(J)} = \{(\mathbf{p}, \zeta_{DF}(J(\mathbf{p}))) \mid \mathbf{p} \in \Omega(J)\}$ where $\{\zeta_{DF}(J(\mathbf{p})) \in \{F, B\} \mid \mathbf{p} \in \Omega(J)\}$. To solve, we find the optimal partition boundary $h_\psi(\mathbf{p}; \theta^*)$ that achieves maximal information gain. Intuitively, the optimal split function $\psi(\mathbf{p}; \theta^*) = 0$ should match the target region boundary. The solution can again be found by solving (19.21) using gradient ascent, but with the target boundary function being parameterized based on the predefined target shape model, *i.e.* a 2D ellipsoid. As $\{\zeta_{DF}(J(\mathbf{p}))\}$ is estimated by the on-line trained classification forest in feature space with probability $q_\zeta(J(\mathbf{p}))$, we can perform tracking by optimizing the confidence weighted information gain formulation derived in (19.15) and (19.19).

19.3.4 Results

19.3.4.1 Classification of Public Machine Learning Datasets

To evaluate the gradient ascent decision forest (GADF), we compare its performance to those of commonly used classifiers including a reference standard Adaboost implementation and a decision forest with oblique hyperplanes. We denote these classifiers as follows:

Adaboost A standard Adaboost classifier that uses axis-aligned stumps as decision functions. This is used as a baseline reference for comparison.

StumpDF A standard decision forest classifier with an oblique hyperplane for the binary test. The optimal binary test is searched by randomly drawing 20 hyperplane samples in the feature space.

Table 19.1 Comparison of classification equal-error rate for **Adaboost**, **StumpDF** and **GADF** on public datasets

Dataset Name	#sample	# fea.	#Train:#Test	Adaboost	StumpDF	GADF
<i>Arcene</i>	200	10000	1:1	0.25	0.318	0.240
<i>Vertebral Column</i>	310	6	3:7	0.157	0.175	0.170
<i>Iris</i>	150	2	1:4	0.281	0.010	0.000
<i>Cardiotocography</i>	2126	23	3:7	0.021	0.022	0.019
<i>Breast Cancer Wisconsin</i>	569	32	1:1	0.056	0.048	0.043

GADF Similar to the **StumpDF**, but gradient ascent information gain optimization is also used during training, using the hyperplane with best performance of the randomly drawn 20 planes. Assuming the data are always normalized into standard deviation along each dimension, we do gradient ascent information gain optimization by gradually reducing the annealing parameter σ starting from $0.03^{\frac{1}{\sqrt{d}}}$ (where d is the feature dimension of the data) with multiplicative scaling factor 0.7. The annealing stops when the optimization converges.

We train both **StumpDF** and **GADF** with 10 trees and we train the trees by randomly sampling 90 % of the original training set. When training each tree node, we search for the optimal split parameters in a randomly sampled feature subspace with dimension number $\text{ceil}(\sqrt{d})$. We limit the maximal tree depth to 15. To evaluate the three methods, we compare their performance over a variety of different application domains using publicly available standard datasets used throughout the machine learning community [111, 377]. We select five datasets, including: *Arcene* (where the application is mass spectrometry), *Vertebral Column* (biomechanics), *Iris* (botany), *Cardiotocography* (1D signal processing), and *Breast Cancer Wisconsin* (cell image classification). We used the given train and test datasets when they are explicitly provided and divide the dataset into different ratios to evaluate generalization, as shown in Table 19.1, when they are not given. We also reduced *Iris* to only the first two features to make the test more challenging. The table summarizes the classification equal-error rates for the three methods, averaged over five experimental runs for each method. We observe that in nearly every case, **GADF** outperforms **StumpDF** as well as the reference standard **Adaboost**.

19.3.4.2 Object Tracking in Videos

In Sect. 19.3.3 we embedded GADF into a full-fledged tracking application. Here we compare its performance to the mean-shift tracker [69]. For the video data we used a standard object tracking PET video as the evaluation task [290]. We evaluated three variants of our GADF based tracker. For all our variants the tracker begins by learning a two-category foreground/background pixel classifier (an appearance model) using the manually delineated first frame. Characteristics of the trackers we compare are as follows:



Fig. 19.8 GainMax trackers using gradient ascent information gain maximization can handle distraction and occlusions well. For the mean-shift tracker (column 1), the red box is the tracking result; for GainMax trackers 0–2 (columns 2–4), red indicates pixels correctly labeled in the ellipsoid as target (true positive), blue indicate false positive pixels outside of the target ellipsoid

MeanShift The standard mean-shift tracker with histogram of size 9 by 9 by 9 bins in RGB space.

GainMax0 The appearance model from the first frame is reused on all frames. Gradient ascent is used to refine the tracking boundary in image space.

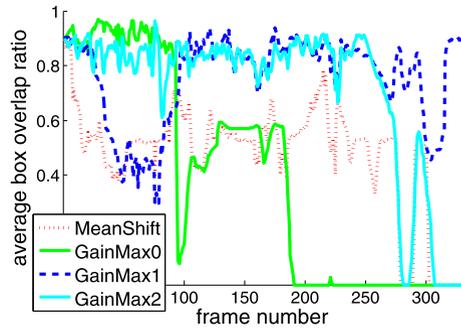
GainMax1 Training data are updated to use background pixels from the previous frame and target pixels from the first frame. Then the two-category pixel classifier is retrained and the tracking boundary is refined using gradient ascent for tracking in both image and pixel feature space.

GainMax2 Training data are updated to use the previous frame’s target and background pixels. Then the two-category pixel classifier is retrained and the tracking boundary is refined using gradient ascent for tracking in both image and pixel feature space.

For the three different variants of GainMax tracking, we fix the number of trees to 20, and the depth of the trees to be 10. We train the decision forest by randomly sampling 80 % of the pixels as training data for each tree, and randomly choose four features from $[r, g, b, \Delta X_x, \Delta X_y, \|\Delta X\|, \angle \Delta X]$ for the training of the binary test functionals.

Some qualitative results are shown in Fig. 19.8, in which a woman turns her head around in an office environment, and then a second person enters and occludes the

Fig. 19.9 The average tracking/ground truth box overlap ratio for the lady video



woman. Tracking is also challenging because the wall has nearly human skin color. We observe that **Meanshift** and **GainMax0** cannot handle the substantial variations in target appearance as they do not do model updating. They are eventually distracted and fail to track. **GainMax1** and **GainMax2** both maintain correct tracking of the lady even over the wall because their appearance models are updated on-line in the pixel feature space and learn to distinguish the face color from wall color. By comparison, **GainMax2** works better when there is no occlusions. However, it is easily distracted by the occlusion. However, **GainMax1** can resist this distraction successfully because it does not update the target pixel data, and can maintain tracking to the end of the video.

Given the ground truth target bounding box B_g and the tracking bounding box B_t in a frame, we can evaluate the tracking performance by their average box overlapping ratio w.r.t. the boxes: $R(B_g, B_t) = (\frac{\Omega(B_g \cap B_t)}{\Omega(B_g)} + \frac{\Omega(B_g \cap B_t)}{\Omega(B_t)})/2$. In Fig. 19.9 we plot the average overlap ratio of the four trackers on the *lady* video.

We can further summarize the tracker's accuracy by the percentage of frames in which $R(B_t, B_g) > 0.5$. Figure 19.10 summarizes the performance of the four evaluated trackers on three videos commonly used for evaluation in visual tracking. Due to substantial appearance variations, occlusions and background distractions, **Mean-shift** and **GainMax0** get distracted easily as they do not perform model updating. **GainMax2** can achieve high tracking accuracy when there is gradual appearance variations (as shown in Fig. 19.9), but fails to track when occlusions exist. Over all,

			
% ($R > 0.5$)	<i>lady</i> [135]	<i>view5</i> [291]	<i>mall</i> [57]
Meanshift	0.66	0.46	0.55
GainMax0	0.43	0.47	0.42
GainMax1	0.85	0.92	1.00
GainMax2	0.82	0.47	1.00

Fig. 19.10 Robustness comparison of the visual trackers by the percentage of frames with $R(B_t, B_g) > 0.5$

GainMax1 achieves the best robustness as it does model updating while avoiding drifting.

19.4 Discussion and Conclusion

This chapter has presented three complementary improvements to the decision forest framework presented in this book.

Entanglement propagates knowledge from one part of the forest to another which speeds learning, improves classifier generalization, and exploits long-range spatial correlations in the input data.

Differentiable information gain maximization allows the optimal data partitioning functional to be found directly through gradient ascent rather than through an exhaustive search over discrete functional parameter space.

Entanglement and differentiable information gain maximization enhance different aspects of decision forests: the use of semantic contextual features and the node optimization function, respectively; they are mutually compatible and may be combined to further enhance the forest accuracy.

The *learned proposal distributions* (Sect. 19.2.2) and differentiable information gain maximization both tackle the problem of node optimization in the presence of a high dimensional feature space. The former increases the effectiveness of brute-force feature search. The latter optimizes the information gain more directly. Since differentiable information gain requires initialization, these two methods can be combined effectively.

The fundamental enhancements presented here may be directly applied to improve results in other applications that use classification forests, including multiple sclerosis lesion segmentation [125], brain segmentation [411], myocardium delineation [212], and more generic object class segmentation tasks [342]. Here we have applied entanglement only to the task of anatomy segmentation, but it is a generic concept and may be adapted to exploit other correlations (*e.g.* over time or space). Likewise our differential information gain approach can form the basis for gradient ascent optimization with more complicated data partitioning functionals (*e.g.* differentiable shape models) based on a-priori heuristics for specific applications.