Revisiting Example Dependent Cost-Sensitive Learning with Decision Trees

Oisin Mac Aodha Gabriel J. Brostow
University College London
http://visual.cs.ucl.ac.uk/pubs/costSensitive

Abstract

Typical approaches to classification treat class labels as disjoint. For each training example, it is assumed that there is only one class label that correctly describes it, and that all other labels are equally bad. We know however, that good and bad labels are too simplistic in many scenarios, hurting accuracy. In the realm of example dependent cost-sensitive learning, each label is instead a vector representing a data point's affinity for each of the classes. At test time, our goal is not to minimize the misclassification rate, but to maximize that affinity. We propose a novel example dependent cost-sensitive impurity measure for decision trees. Our experiments show that this new impurity measure improves test performance while still retaining the fast test times of standard classification trees. We compare our approach to classification trees and other cost-sensitive methods on three computer vision problems, tracking, descriptor matching, and optical flow, and show improvements in all three domains.

1. Introduction

Given a set of training examples of the form $\{(\mathbf{x}^1, y^1), ..., (\mathbf{x}^N, y^N)\}$, where $\mathbf{x}^n$ is a D dimensional feature vector and $y^n \in \{1, ..., C\}$ is its corresponding class label, the test-time goal of classification is to label an unseen feature vector $\mathbf{x}^*$ with one of C discrete class labels. In most classification scenarios, the feature vectors are said to be disjoint, meaning each observation is assigned to one and only one class. The disjoint formulation of the classification problem is well suited to scenarios where each feature vector can be assigned a discrete class label, *e.g.* discriminating between two object categories: dog versus cat.

Cost-sensitive learning is concerned with the situation where the classification task may not be disjoint [9, 30, 6]. For example, when computing the optical flow field between a pair of images, we may have access to several different algorithms, each with differing strengths and weaknesses. For a given scene, we wish to use the algorithm that will produce the most accurate result. At test time, we aim to assign a *suitability probability* to each algorithm, representing our belief in their affinity for performing the given task. Each specialist (or in this case algorithm) is said to have a *task score*, a measure of their competence at performing that task, evaluated against known ground truth (available only at training time). A lone specialist might have a significantly higher task score in certain scenarios, while in others, multiple specialists could be comparably accurate. The key here is that we are not only interested in specialists that score well, but more importantly, the differences between them. More specifically, in cost-sensitive classification we are presented with a set of specialists $\mathcal{S}$, where $C = |\mathcal{S}|$, and a set of training examples $\{(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^N, \mathbf{y}^N)\}$, where $\mathbf{y}^n \in \mathbb{R}^C$ is our label vector. Each element of the label vector is a continuous value, $0 \leq y_c^n \leq 1$, representing specialist c's task score. Higher values of y_c^n indicate better accuracy for that data point $\mathbf{x}^n$. Traditional disjoint binary classification can then be seen as a special case, where $\mathbf{y}^n = (0, 1)$ or $(1, 0)$.

More concretely, for a given task instance $\mathbf{x}$ we wish to find the specialist $c \in \mathcal{S}$ that produces the maximum task score. Therefore we wish to minimize the loss function

$$\mathcal{L}_{\text{cs}}(c, f(c, \mathbf{x})) = 1 - f(c, \mathbf{x}), \tag{1}$$

where $f(c, \mathbf{x})$ is the task score for specialist c on task instance $\mathbf{x}$, with a best possible task score of 1, and 0 as the worst. Given a new $\mathbf{x}^*$ at test time, we wish to assign a proportionally higher suitability probability to specialists that give superior task scores.

We propose a novel impurity measure for decision trees, which takes task (*i.e.* cost) information into account when measuring the quality of candidate node splits. Unlike other tree based methods, we explicitly look at the difference between examples' costs at a node, and not just their total cost. Our experiments show how computer vision tasks such as tracking, descriptor matching and optical flow estimation can be posed as example dependent cost-sensitive learning problems. Our novel impurity measure has the benefits of higher accuracy at test time, simpler decision boundaries, and fast test time performance, at the expense of a moderate increase in training time.

2. Related Work

Cost-sensitive learning covers a broad category of problems in the machine learning literature. Different works seek to model various types of costs that arise when building a classifier. The cost can refer to feature acquisition [22, 16, 28], where some feature dimensions or component algorithms may be more expensive to compute or acquire than others. There may be different labeling costs associated with the user, depending on the type of annotations they are asked to provide [25]. In this paper, we are concerned with the costs associated with misclassifying different datapoints. Traditionally this problem has been approached in two different ways: class (CCS) and example (ECS) dependent cost-sensitive learning. In CCS, costs are defined using a cost matrix and all misclassifications for a given class are considered equal *e.g*. [9, 10]. In ECS, different costs are associated with misclassifying each individual datapoint *e.g*. [30, 4]. As a result, both standard classification and CCS can be seen as special cases of ECS.

Different approaches have been proposed to solve the example dependent cost-sensitive learning problem, such as reweighting the training examples based on their cost [10, 2]. Abe *et al*. [2] reduce the problem to standard classification using a method inspired by example reweighting and boosting. While rescaling the training points based on their costs has been shown to be effective for CCS [32], it is not trivial to apply it to ECS. Jan *et al*. [15] use multi-criterion optimization to maximize task scores and minimize the error rate. Tu and Lin [24] simplified the ECS problem to a form of one-sided regression, achieving the best results when compared to several other SVM based methods.

Decision tree based algorithms benefit from very fast training and test times, being easy to implement, producing probabilistic outputs, and naturally extending to the multiclass case. Decision trees can be adapted for many different types of machine learning problems such as multivariate regression [8], structured outputs [17], and Hough voting [12]. There are three main ways in which cost information can be incorporated into decision trees during training. The first option is to alter how the data is sampled. For the binary ECS case, Zadrozny *et al*. [31] propose cost-proportionate rejection sampling with aggregation. A variant of their method is illustrated on an ensemble of trees where each tree samples with replacement from the training data, and samples are drawn proportionately to their cost. The next option is to alter the class distribution at each node so it is cost aware. For CCS, Breiman *et al*. [6] alter the node posterior by weighting it by the cost vector for each class (the cost vector is the sum across each column of the cost matrix for the class of interest). In contrast, Ting [23] weights each datapoint individually in proportion to the cost. A drawback of both methods is that they will create the same trees for different cost matrices if summing the cost matrix columns happens to produce the same totals. This perhaps explains the similar performance for multiclass classification when compared to standard classification in [23]. The last option, the one being employed in this paper, is to create a novel impurity measure that is designed specifically for the example cost-sensitive case. We directly exploit the cost differences between the examples at a node, addressing some of the example- and cost-insensitivity limitations of the previous methods.

While ECS problems are quite common, a lack of ground truth has made it difficult to assess the performance of different algorithms. For CCS, a small number of real datasets exist, with cost matrices for problems such as intrusion detection [1] and bacteria classification [15]. This lack of data meant that experimental validation was typically performed by artificially generating cost matrices for standard machine learning datasets based on class frequency in the training set *e.g*. [32, 2, 24]. In CCS it is straightforward for human experts to define cost matrices based on misclassification costs, *e.g*. in medical applications with set costs for false negatives versus false positives for a particular diagnosis. For ECS, this additional per-datapoint information requires more annotation effort from the expert. However, increasingly there are classification problems in which these example dependent costs are available naturally [13, 18]. In another recent example, Everts *et al*. [11] showed that ECS can be used to choose the best descriptor for a local image patch. In this paper, we use these datasets to show how effective use of this cost-sensitive information at training time improves decision-tree based performance for three different algorithm selection tasks.

3. Cost-Sensitive Discriminative Classifier

We propose a novel multi-class example dependent cost-sensitive classification algorithm, which takes into account the *full* label vector information when building the classifier. Our classifier is based on the bagged ensemble Random Forests method of Breiman [5]. For a recent overview of decision tree ensembles, see [8].

3.1. Random Forests

To review, a Random Forest is an ensemble of decision trees [6], where each tree is trained independently on a random subset of the data. Trees are grown recursively from the root node where at each node, P, a set of random splitting decisions is proposed that attempt to separate the datapoints landing at the node into its left (L) and right (R) child nodes. Decision trees greedily minimize a loss function at each node to partition the data. Our goal is to minimize the loss function in (1). The information gain E_{inf} at the node serves as the quality measure of each potential split [19]. So among the proposals, the one that maximizes

$$E_{\text{inf}} = I(P) - \left(\frac{N_L}{N} I(L) + \frac{N_R}{N} I(R)\right) \tag{2}$$

is chosen, where N, N_L and N_R are the numbers of examples that have landed at the parent, left, and right child nodes respectively. To compute the information gain of (2), we need to calculate the impurity $I(\cdot)$ at each node. The goal of the impurity measure is to determine how much disagreement there is among the datapoint labels at that node. For classification, a node has minimum impurity when all the data points at the node belong to the same class, and maximum when they are all equally different. Several different types of impurity measure for classification have been proposed, such as Gini I_{gini} [6], entropy I_{ent} [20] and misclassification rate I_{mcl}. These are calculated as

$$I_{\text{gini}} = 1 - \sum_{c=1}^{C} p_c^2, \tag{3}$$

$$I_{\text{ent}} = -\sum_{c=1}^{C} p_c log_2(p_c), \tag{4}$$

$$I_{\text{mcl}} = 1 - \max(\mathbf{p}), \tag{5}$$

where $\mathbf{p}$ is a C dimensional vector, with each entry p_c being the (normalized) empirical frequency of class c at the node. As illustrated in Section 4, we use I_{gini} as the representative cost-oblivious impurity measure when growing a forest of classification decision trees. For our experiments, we will use the popular Classification-based Random Forest (CLRF) as our first baseline. For univariate regression I_{reg} [6], one aims to minimize the variance of all the continuous response values that land at a node, so

$$I_{\text{reg}} = \sum_{n \in \mathcal{N}^*} (y^n - \mu_y)^2. \tag{6}$$

$\mathcal{N}^*$ is the subset of datapoints $\mathcal{N}$ that landed at the node, and μ_y is the mean label value in $\mathcal{N}^*$. Regression Forests can be used to select specialists, but we found performance to be inferior to that of classification, possibly because regression needs more training data.

3.2. Cost-Sensitive Impurity Measure

Standard classification impurity measures cannot utilize the task scores at training time. Instead, they rely on the empirical frequency of examples that landed at that node. We could ignore the task scores (see the CLRF baseline) and set the class label for a given example $\mathbf{x}^n$ to the specialist that produces the highest task score, $c = \arg\max_c y_c^n$. However, by doing this we are throwing away valuable information that may improve classification accuracy.

Gini-impurity Cost-Sensitive Random Forest (GCSRF): The simplest way to use the task scores would be to adapt the C dimensional class posterior $\mathbf{p}$ at each node. Instead of counting the number of examples from each class that lands at a node ($\mathcal{N}^*$), we could use their task scores directly to weight the normalized frequency for each class. This altering of the class posterior at the node has been explored for class dependent cost-sensitive learning [23, 6]. For this class dependent version, each element of the modified node posterior is computed as

$$p_c = \sum_{n \in \mathcal{N}^*} y_c^n / \sum_{n \in \mathcal{N}^*} \sum_{k=1}^{C} y_k^n. \tag{7}$$

We can now use these new posteriors in any of the standard classification impurity measures, and use Gini for this baseline, for better comparison to the Gini-based CLRF.

PairWise Cost-Sensitive Random Forest (PWCSRF): In practice, we are interested not in the absolute task scores for each datapoint, but in the relative difference for each specialist's score *for that example*. We only wish to split examples when there is a significant difference between the scores of each of the specialists. With this aim, we define an impurity measure based on the *pairwise difference* between the task scores in the label vector, *i.e.* how much better is one specialist than another,

$$I_{\text{cs}} = \frac{1}{C^2 - C} \sum_{i=1}^{C} \sum_{j=1}^{C} (f_{i \to j} - f_{i \to j}^2) \quad \forall i \neq j. \tag{8}$$

The pairwise specialist empirical frequency $f_{i \to j}$ is computed between every pair of classes for every datapoint in $\mathcal{N}^*$, resulting in $C^2 - C$ comparisons of

$$f_{i \to j} = \frac{\sum_{n \in \mathcal{N}^*} (d_{i \to j}^n)^2}{\sum_{n \in \mathcal{N}^*} (d_{i \to j}^n + d_{j \to i}^n)}. \tag{9}$$

The difference vector $\mathbf{d}_{i \to j}$, is a vector of size $|\mathcal{N}^*|$, where each element $d_{i \to j}^n$ is the truncated positive difference between each element of the label vector, so

$$d_{i \to j}^n = \begin{cases} y_i^n - y_j^n & \text{if } y_i^n > y_j^n \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

At test time, for inference, we then use (7) to represent the posterior probability for each class at a node.

We will refer to the standard classification Random Forest with I_{gini} impurity measure as CLRF, the example dependent cost-sensitive forest with Gini impurity using the cost aware posterior of (7) as GCSRF, and our forest with pairwise cost-sensitive impurity measure as PWCSRF.

4. Insight Into Proposed Impurity Measure

Figure 1 illustrates the node-impurity binary classification curves for different classification impurity measures.

Again, during training, potential splits are accepted or rejected for a node in a tree on the basis of the node impurity. Also displayed are the impurity scores for two example sets of datapoints, $\mathcal{N}1$ and $\mathcal{N}2$. The label vectors along with impurity values for both sets in this toy example are presented in Table 1. Both sets contain four datapoints, the only difference being that in $\mathcal{N}2$ one of the datapoints has a very similar task score for the two specialists (red and green). For $\mathcal{N}1$, which contains disjoint labels, our newly proposed cost-sensitive measure I_{cs} simply produces the same impurity as I_{gini} and I_{csg} (Gini impurity using the cost aware posterior of (7)). However, in the second scenario, $\mathcal{N}2$, I_{gini} is very sensitive to tiny changes in the task scores, while I_{cs} does not punish these small differences. I_{cs} exploits the fact that the red specialist will give a high task score for the whole set. This is because for three observations, the red specialist scores best, and has a very similar score to the green specialist for the fourth observation. Unlike our I_{cs}, alternative impurity measures can overlook a good split because they are over-sensitive to negligible differences in the label vector (see Table 1). I_{csg} is sensitive to the absolute value of the task scores. It is unable to look at pairwise differences, producing a high impurity even when the difference between specialists is negligible.

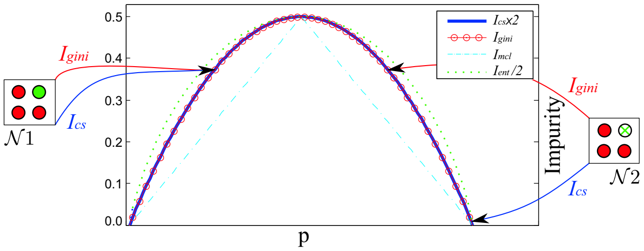


Figure 1. Comparison of node-impurity binary classification curves for different impurity measures (note that both I_{cs} and I_{ent} are scaled) for the binary classification problem. The horizontal axis corresponds to the probability of class 1 while the vertical represents impurity. Low impurity indicates a good grouping of the data. Outside the graph, solid colored discs represent datapoints best described by one of two specialists (red or green), while a disc with a colored cross indicates only a slight preference for one over the other. I_{cs} and I_{gini} return the same impurity for the standard binary classification task $\mathcal{N}1$, while in the non-disjoint case, $\mathcal{N}2$, I_{cs} recognizes that the red specialist achieves a relatively high task score, resulting in a much lower impurity.

	$\mathcal{N}1$					$\mathcal{N}2$				
n	$\mathbf{y}$		y	$d_{1\rightarrow 2}$	$d_{2\rightarrow 1}$	$\mathbf{y}$		y	$d_{1\rightarrow 2}$	$d_{2\rightarrow 1}$
1	1	0	1	1	0	1.0	0.00	1	1.0	0.00
2	1	0	1	1	0	1.0	0.00	1	1.0	0.00
3	1	0	1	1	0	1.0	0.00	1	1.0	0.00
4	0	1	2	0	1	0.5	0.51	2	0.0	0.01
I_{gini}	0.3750					0.3750				
$I_{\text{csg}} \times 2$	0.3750					0.2558				
$I_{\text{cs}} \times 2$	0.3750					0.0033				

Table 1. Impurity measure comparison for two different sets of observations $\mathcal{N}1$ and $\mathcal{N}2$, each containing four datapoints. Good splits should group data to yield low impurity. $\mathbf{y}$ represents the label vector, while y is the index of the specialist with the best task score. Previous methods quantize the label vector, throwing away important information [18, 13], making all data look like $\mathcal{N}1$.

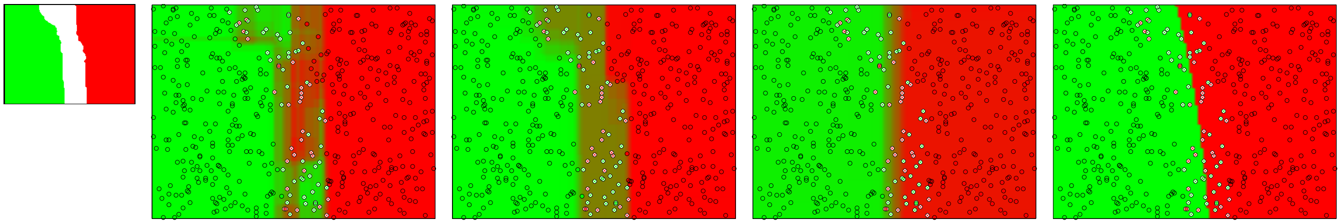
4.1. Synthetic Example

In the toy example of Figure 2, we generate datapoints at random from an underlying known distribution. The ground truth image in A) illustrates the generating distribution, while B) - E) show the results for different algorithms. Here we have two specialists, where datapoints in the green region are best described by the first specialist ($\mathbf{y}^n = (1, 0)$), in the red by the second specialist ($\mathbf{y}^n = (0, 1)$), and in the white region can be close to equally handled by either specialist, with some additive Gaussian noise ($\mathbf{y}^n = (0.5+\delta_1, 0.5+\delta_2)$). In Figures 2 B) - E), the colored discs represent training samples. For the white region, the color of the cross in the center represents the specialist that is marginally better (may require zooming in). At test time, we evaluate the probability of each location in the feature space and illustrate the posterior specialist suitability probability for each example.

For this illustrative example, we chose the following parameters: 500 training points, 10 trees, 60 random tests at each node, and a minimum node count of 3. This results in 26, 25 and 14 average number of nodes per tree for CLRF, GCSRF and PWCSRF respectively. We observe that CLRF tends to overfit the data and results in a noisy boundary. GCSRF maintains uncertainty in the ambiguous region at the expense of a more complex model. The SVM based OSSVR of [24] creates a non-linear separation down the middle of the ambiguous region. PWCSRF favors a simpler, yet loss-minimizing, decision boundary. Tests on real data follow.

5. Experiments

We validate our cost-sensitive learning algorithm on three challenging computer vision problems: motion model selection (with a small number of training examples) [13], local image descriptor matching (large number of classes) [11], and optical flow computation (low diversity among the individual specialists for large subsets of the data) [18]. For each experiment, we compare our Pairwise Cost-Sensitive Random Forest (PWCSRF) against two baselines: the Gini-based Cost-Sensitive Random Forest (GCSRF), and the established but naive Classification Random Forest (CLRF). We grow trees down to a maximum specified depth, unless the minimum sample count at a node is reached, and there is no pruning of the final trees. We use simple axis aligned feature tests at each node, though a va-



A) Ground Truth B) CLRF (Classification Baseline) C) GCSRF (Cost-sensitive Baseline) D) PWCSRF (ours) E) OSSVR

Figure 2. A) Ground truth distribution from which training data points are randomly sampled. Red and green regions indicate areas of the feature space where one of the two specialists is superior. If an example comes from the white region it is close to equally well represented by either specialist (white points with colored crosses). B) - D) Results for the different variants of the Random Forest. E) The cost-sensitive SVM of [24]. Note that suitability probabilities of C) - E) have different complexities and confidence for this toy example, but are not obviously good or bad.

riety of other tests are possible [8]. Unless otherwise stated, for bagging we randomly sample with replacement, ensuring an even number of examples from each specialist per tree. Our label vector contains continuous task score values, but for each observation with the CLRF, we set the class label to be the index of the maximum value of $\mathbf{y}^n$. Success is determined not in terms of classification score but task score. The classification score would only measure how often the best specialist was chosen, while the task score measures the real benefit of choosing specialists using a given model.

5.1. Tracking

Garcia Cifuentes *et al.* [13] posed motion-model-selection for tracking image features in video as a classification problem. Given an image sequence, and a set of motion models used to track features in that sequence, the goal is to choose the motion model which produces the most accurate tracking score for the whole sequence. For us, each motion model can be thought of as a specialist. The task score is the tracking accuracy for that motion model, with higher values indicating better accuracy.

We use the trajectory feature vector from [13], which is computed from each image sequence using the descriptor of Wang *et al.* [26]. This results in a 4911 dimensional feature vector. In total, there are 117 datapoints and six different motion models: Brownian, Constant Velocity, Right, Left, Forwards and Backwards. As in [13], tracking performance is measured in terms of track robustness, *i.e.* correct point locations when compared to manually clicked ground-truth, where early failures cost more.

For each of the forest based classifiers, we use the following parameters: 50 trees, $10,000$ random tests at each node, and a minimum sample count of 3. We train on all six classes jointly, performing leave one out testing on all 117 examples. This results in a task score of 0.5174 for our PWCSRF, 0.5172 for the CLRF and 0.4680 for GCSRF (all averaged over 20 runs). While best among forests, our score of 0.5174 is lower than the score of 0.5290 achieved in [13]. We use only simple axis aligned splits in contrast to their more complex SVMs. The forests are also hampered by the small amount of available training data.

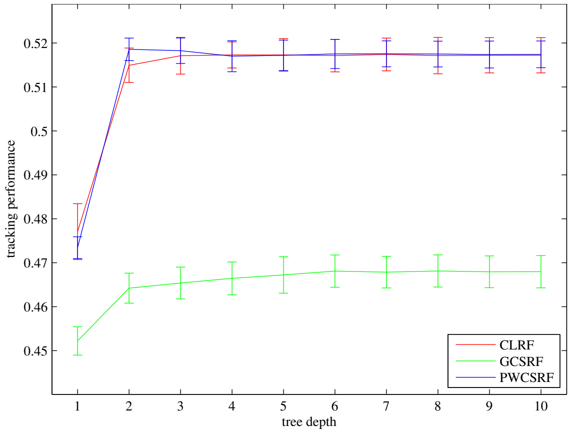


Figure 3. Comparison of the different forest based classifiers on the motion model estimation data from [13]. Higher scores are better. Our PWCSRF is comparable to CLRF, and both beat the GCSRF baseline.

5.2. Descriptors

Matching interest points across multiple images is a challenging problem. Variations in lighting, object properties, and viewpoint can make it difficult to find the correct correspondence for a local image patch in another image. Everts *et al.* [11] acknowledge that there is no one best descriptor for all scenarios, and choosing the best one is situation specific. In their work, they attempt to automatically assign the best descriptor for a local image patch using a classifier trained on multiple image patches with known ground truth correspondences.

We compare our PWCSRF to [11] using the same Aloi dataset from [14]. We use the same training and test split which results in 66K training and 66K test examples. For each example, there is a 73 dimensional feature vector which characterizes the local appearance of the patch. The

	CLRF	REGRF	GCSRF	PWCSRF	OSSVR [24]	CSSVM [11]
Precision	0.5675 ±0.0003	0.5770 ±0.0003	0.5849 ±0.0002	**0.5878 ±0.0004**	0.5922	0.5836
Percent Best	75.33	77.09	78.51	**79.02**	79.98	
Train Time (mins)	**0.99**	54.12	3.15	31.12	624.35	
Test Time (mins)	0.05	0.80	**0.03**	0.06	47.61	

Table 2. Descriptor selection results using data from [11]. The results are averaged over five runs. We can see that OSSVR [24] (with $C = 1000$) has superior performance compared to the forest based methods, but at an even larger increase in training time than required for PWCSRF, and a much longer testing time. Timing and other details of CSSVM performance are not available in [11], but the precision comes from their Figure 5.

specialists correspond to ten different image descriptors that could be used to describe the patch. The task scores are the average precision for each descriptor, with 0 being the worst and 1 the best. The single descriptor (sBest), which performs best overall on the test set, results in an average precision of 0.5185, with the worst possible score being 0.1702 and the best 0.7393. Everts *et al.* [11] use a cost-sensitive SVM with the one versus all formulation of [31], resulting in a score of 0.5836. As can be observed in Table 2, both the GCSRF and PWCSRF improve on this score, while the CLRF performs poorly. Figure 4 shows how the test results are affected by tree depth. We set the number of trees to 200, performed 400 random tests at each node, and had a minimum sample count of 2. In addition, we also compare ourselves to the multi-class cost-sensitive one-sided support vector regressor OSSVR of [24], which represents the state of the art in cost-sensitive support vector classification. Using a linear kernel OSSVR, we achieve an average precision of 0.5922 compared to the 0.5878 of our PWCSRF but with a large additional increase in training time (Table 2). We also perform a comparison to univariate regression Forests REGRF, where a separate Forest is trained for each specialist and at test time we choose the winning specialist for a datapoint as the one whos regression Forest predicts the best task score.

5.3. Optical Flow

Given an image pair and a set of optical flow algorithms, in an earlier work we attempted to determine the flow algorithm which would result in the lowest error for each pixel [18]. This was posed as a multi-class classification problem and a standard Random Forest was used to learn the pixel-to-algorithm mapping using a feature vector computed from the image and proposed optical flow fields. Here, our specialists correspond to one of C different optical flow algorithms, with the task score representing the end point error (EPE) for a given optical flow vector for each of the algorithms. The EPE for a given specialist, epe_c^n, is the Euclidean distance between the proposed flow vector and the ground truth vector, with the lowest error corresponding to 0 and the worst to ∞. We use a sigmoid function to map

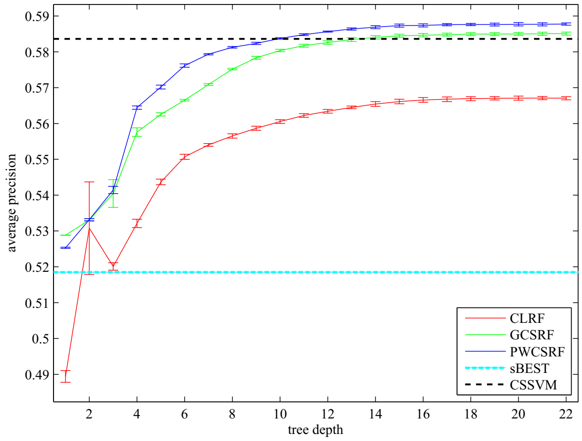
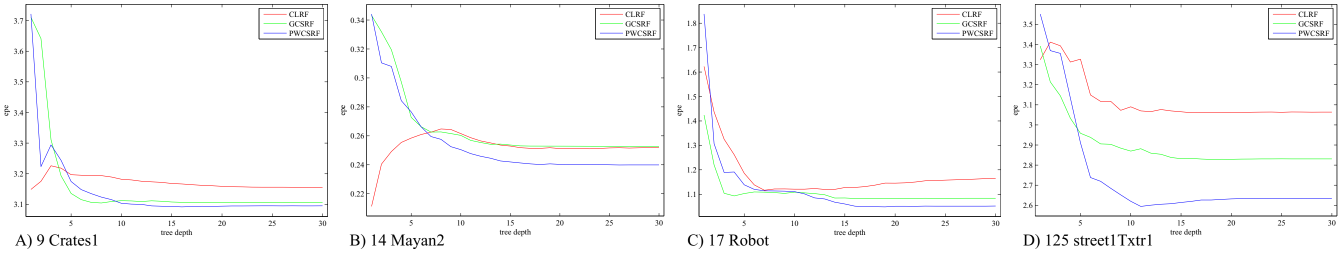


Figure 4. Comparison of different classification methods for image descriptor algorithm assignment [11]. Higher average precision values indicate better performance. Forest results are averaged over five runs, and are compared to CSSVM and "sBest", the single descriptor that performed best overall.

EPEs to a task score range of $[0, 1]$, where 1 represents the lowest possible error,

$$y_c^n = 1 - 2\left(\frac{1}{1+\exp(-\lambda epe_c^n)} - \frac{1}{2}\right). \qquad (11)$$

Table 3 presents results for leave one out optical flow experiments on 22 different sequences from [18] and 8 from [3]. We randomly sample $8,000$ datapoints with replacement from each of the 22 sequences from [18], ensuring an even distribution of wins for each specialist. The optical flow algorithms chosen as specialists were TV [29], FL [27], CN [21] and LD [7]. We used 50 trees with a minimum sample count of 10, 2000 random tests at each node, maximum possible depth of 20, and set $\lambda = 1.0$ for the sigmoid function, with results showing an average over three runs. We can see in Table 3 that our PWCSRF produces the best mean EPE, the largest number of wins, and has the best overall rank. Surprisingly, the const-insensitive baseline CLRF comes second in terms of the number of wins. In Figure 5, we display the effect of varying tree depth for a subset of the sequences. We observe that beyond a depth



A) 9 Crates1 B) 14 Mayan2 C) 17 Robot D) 125 street1Txtr1

Figure 5. Task score results as a function of tree depth for the different forest based classifiers. Lower values of End Point Error (EPE) indicate better scores, and while PWCSRF is not always best, it produces lower average EPE overall. A) - D) is a subset of the sequences from Table 3, and the rest are provided in the supplementary materials. Results are averaged over three runs.

Sequence	TV	FL	CN	LD	CLRF	GCSRF	PWCSRF
1 Venus	0.408	0.342	0.229	0.433	0.2800 ±0.005	0.2938 ±0.004	**0.2693 ±0.003**
2 Urban3	1.132	0.524	0.377	0.600	0.4827 ±0.017	0.5359 ±0.003	**0.4805 ±0.004**
3 Urban2	0.506	0.444	0.207	0.334	0.3124 ±0.009	**0.2804 ±0.009**	0.2812 ±0.005
4 RubberWhale	0.135	0.096	0.077	0.120	**0.0922 ±0.001**	0.1108 ±0.001	0.0933 ±0.001
5 Hydrangea	0.196	0.164	0.154	0.178	**0.1590 ±0.001**	0.1632 ±0.001	0.1607 ±0.001
6 Grove3	0.745	0.624	0.438	0.657	0.5726 ±0.002	**0.5655 ±0.005**	0.5773 ±0.006
7 Grove2	0.220	0.169	0.091	0.159	**0.1200 ±0.002**	0.1400 ±0.001	0.1281 ±0.002
8 Dimetrodon	0.211	0.144	0.115	0.117	0.1447 ±0.003	**0.1262 ±0.001**	0.1361 ±0.004
9 Crates1	3.464	3.730	3.150	3.104	3.1557 ±0.004	3.1058 ±0.002	**3.0952 ±0.013**
10 Crates2	4.615	12.572	10.409	2.513	2.9804 ±0.030	2.5558 ±0.036	**2.4481 ±0.013**
13 Mayan1	2.331	0.727	1.718	5.567	**3.7306 ±0.261**	4.8670 ±0.320	3.7523 ±0.105
14 Mayan2	0.442	0.344	0.211	0.350	0.2519 ±0.005	0.2528 ±0.003	**0.2399 ±0.001**
17 Robot	2.335	1.857	1.525	1.212	1.1651 ±0.026	1.0836 ±0.003	**1.0517 ±0.009**
18 Sponza1	1.006	1.013	1.102	0.917	0.9883 ±0.003	**0.9555 ±0.004**	0.9785 ±0.000
19 Sponza2	0.531	0.494	1.674	0.481	**1.2177 ±0.070**	1.5917 ±0.006	1.5364 ±0.024
22 Crates1Htxtr2	1.106	0.693	1.640	0.548	**0.6741 ±0.056**	0.8596 ±0.068	0.6974 ±0.024
24 Crates2Htxtr1	3.128	10.210	8.805	0.809	1.3330 ±0.049	0.8808 ±0.032	**0.7994 ±0.023**
26 Brickbox1t1	1.094	0.394	0.228	2.602	**0.3179 ±0.010**	0.3425 ±0.001	0.3185 ±0.008
29 Brickbox2t2	7.478	1.827	2.192	3.505	2.8284 ±0.087	1.5644 ±0.092	**1.4889 ±0.058**
30 GrassSky0	2.102	2.484	1.317	1.039	1.3662 ±0.015	1.2335 ±0.017	**1.2266 ±0.029**
39 GrassSky9	0.722	0.438	0.273	0.510	0.3273 ±0.005	**0.3095 ±0.004**	0.3154 ±0.001
49 TxtRMovement	3.166	0.241	0.132	0.356	0.4489 ±0.040	**0.2112 ±0.009**	0.2321 ±0.012
50 TxtLMovement	1.521	0.282	0.126	0.604	0.3596 ±0.010	0.3231 ±0.058	**0.2317 ±0.039**
51 blow1Txtr1	0.085	0.050	0.027	0.081	**0.0387 ±0.001**	0.0509 ±0.004	0.0421 ±0.002
88 blow19Txtr2	0.525	0.380	0.199	0.319	0.2896 ±0.001	0.2970 ±0.004	**0.2878 ±0.003**
89 drop1Txtr1	0.119	0.071	0.052	0.084	**0.0551 ±0.002**	0.0723 ±0.001	0.0570 ±0.001
106 drop9Txtr2	5.195	1.985	2.715	4.369	**2.8761 ±0.055**	3.0909 ±0.038	3.0783 ±0.022
107 roll1Txtr1	0.004	0.005	0.002	0.002	0.0029 ±0.000	**0.0028 ±0.000**	0.0033 ±0.000
124 roll9Txtr2	0.040	0.048	0.014	0.023	0.0249 ±0.001	0.0254 ±0.001	**0.0240 ±0.000**
125 street1Txtr1	3.647	3.585	4.097	2.664	3.0637 ±0.025	2.8310 ±0.060	**2.6332 ±0.013**
Mean EPE	1.6070	1.5312	1.4432	1.1419	0.9887	0.9574	**0.8888**
Rank					2.100	2.267	**1.633**
Wins					10	7	**13**
Num Nodes					9257	**758**	9336
Train Time (mins)					**12.08**	17.14	83.78
Test Time (mins)					0.03	**0.01**	0.02

Table 3. End Point Error (EPE) results for optical flow experiments for four different specialists: TV [29], FL [27], CN [21] and LD [7] are displayed in the center, and the different forest based algorithms that choose between them for each pixel are on the right.

of 20, the results do not improve substantially. For further illustrated results, please see our supplementary material.

6. Conclusion

We have presented a novel impurity measure for tree based classifiers for example dependent cost-sensitive classification. Our classifier retains all the advantages of tree classifiers such as fast test time, ease of implementation, inherent multi-class classification, and probabilistic output. We have shown that posing tracking, descriptor selection, and optical flow estimation as cost-sensitive classification tasks usually results in better test time performance when compared to standard classification trees. In the case of optical flow estimation, our new impurity measure achieves a 10% and 7% improvement in flow accuracy over classification and an alternative ensemble of cost-sensitive trees respectively. Crucially, by exploiting all the task score data available at training time, we can build more representative classifiers that better generalize at test time.

These benefits come at the expense of moderately increased training times, though still better than support-vector-based methods. Other opportunities for future improvements exist. For example, the functions for mapping algorithm errors to task scores have not been optimized for the three domains tested so far. Also, for problems like flow or tracking, inference could be performed jointly instead of choosing an algorithm independently for each pixel. Finally, it would be interesting to apply this model to other classes of vision algorithms.

Acknowledgements Funding for this research was provided by the NUI Travelling Studentship in the Sciences and EPSRC grant EP/K015664/1.

References

[1] UCI KDD Archive. http://kdd.ics.uci.edu/. 2

[2] N. Abe, B. Zadrozny, and J. Langford. An iterative method for multi-class cost-sensitive learning. In *KDD*, 2004. 2

[3] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2011. 6

[4] U. Brefeld, P. Geibel, and F. Wysotzki. Support vector machines with example dependent costs. *ECML*, 2003. 2

[5] L. Breiman. Random forests. *Machine Learning*, 45(1), 2001. 2

[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. 1984. 1, 2, 3

[7] T. Brox and J. Malik. Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation. *PAMI*, 2010. 6, 7

[8] A. Criminisi and J. Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. 2013. 2, 4

[9] P. Domingos. Metacost: a general method for making classifiers cost-sensitive. In *KDD*, 1999. 1, 2

[10] C. Elkan. The foundations of cost-sensitive learning. In *Int. Joint Conference on Artificial Intelligence*, 2001. 2

[11] I. Everts, J. van Gemert, and T. Gevers. Per-patch descriptor selection using surface and scene properties. In *ECCV*, 2012. 2, 4, 5, 6

[12] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *PAMI*, 2011. 2

[13] C. García Cifuentes, M. Sturzel, F. Jurie, and G. J. Brostow. Motion models that only work sometimes. In *BMVC*, 2012. 2, 4, 5

[14] J. Geusebroek, G. Burghouts, and A. Smeulders. The amsterdam library of object images. *IJCV*, 2005. 5

[15] T.-K. Jan, D.-W. Wang, C.-H. Lin, and H.-T. Lin. A simple methodology for soft cost-sensitive classification. In *KDD*, 2012. 2

[16] S. Ji and L. Carin. Cost-sensitive feature acquisition and classification. *Pattern Recognition*, 2007. 2

[17] P. Kontschieder, S. R. Buló, H. Bischof, and M. Pelillo. Structured class-labels in random forests for semantic image labelling. In *ICCV*, 2011. 2

[18] O. Mac Aodha, A. Humayun, M. Pollefeys, and G. J. Brostow. Learning a confidence measure for optical flow. *PAMI*, 2012. 2, 4, 6

[19] S. Nowozin. Improved information gain estimates for decision tree induction. In *ICML*, 2012. 2

[20] J. Quinlan. *C4.5: Programs for Machine Learning*. 1993. 3

[21] D. Sun, S. Roth, and M. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010. 6, 7

[22] M. Tan. Cost-sensitive learning of classification knowledge and its applications in robotics. *Machine Learning*, 1993. 2

[23] K. M. Ting. An instance-weighting method to induce cost-sensitive trees. *Knowledge and Data Engineering, IEEE Transactions on*, 2002. 2, 3

[24] H. Tu and H. Lin. One-sided support vector regression for multiclass cost-sensitive classification. In *ICML*, 2010. 2, 4, 5, 6

[25] S. Vijayanarasimhan and K. Grauman. Cost-sensitive active visual category learning. *IJCV*, 2011. 2

[26] H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin. Action Recognition by Dense Trajectories. In *CVPR*, 2011. 5

[27] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 Optical Flow. In *BMVC*, 2009. 6, 7

[28] P. Yin, A. Criminisi, J. Winn, and I. A. Essa. Bilayer segmentation of webcam videos using tree-based classifiers. *PAMI*, 2011. 2

[29] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L1 Optical Flow. In *DAGM*, 2007. 6, 7

[30] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *KDD*, 2001. 1, 2

[31] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *ICDM*, 2003. 2, 5

[32] Z.-H. Zhou and X.-Y. Liu. On multi-class cost-sensitive learning. In *AAAI*, 2006. 2