

Learning Policies for Efficiently Identifying Objects of Many Classes

Ramana Isukapalli
Lucent Technologies, Bell Labs Innovations
Whippany, NJ 07981, USA
risukapalli@lucent.com

Ahmed Elgammal
Rutgers University
New Brunswick, NJ 08854, USA
elgammal@cs.rutgers.edu

Russell Greiner
University of Alberta, Edmonton, CA T6G 2E8
greiner@cs.ualberta.ca

Abstract

Viola and Jones (VJ) cascade classification methods have proven to be very successful in detecting objects belonging to a single class — e.g., faces. This paper addresses the more challenging “many class detection” problem: detecting and identifying objects that belong to any of a set of classes. We use a set of learned weights (corresponding to the parameters of a set of binary linear separators) to identify these objects. We show that objects within many real-world classes tend to form clusters in this induced “classifier space”. As the result of a sequence of classifiers can suggest a possible label for each object, we formulate this task as a Markov Decision Process. Our system first uses a “decision tree classifier” (i.e., a policy produced using dynamic programming) to specify when to apply which classifier to produce a possible class label for each sub-image W of a test image. This corresponds to a leaf of the decision tree. It then uses a cascade of classifiers, specific to this leaf to confirm that W is an instance of the proposed class. We present empirical evidence to verify that our ideas work effectively: showing that our system is essentially as accurate as running a set of cascade classifiers (one for each class of objects), but is much faster than that approach.

1. Introduction

The pioneering work of Viola and Jones [9] has led to a successful face detection method based on “cascade classifiers”, where each classifier is a binary classifier that is learned by applying Adaboost [3] to a database of training images of faces and non-faces. The VJ learning algorithm takes as input several thousands of images correctly labeled as faces versus non-faces and produces a cascade of boosted classifiers. Every classifier consists of several “linear separators”, each built using features on a rectangular subregion (which VJ call “rectangle features”) of the training image. The learning algorithm selects the rectangle features that can best separate faces in training data from non-faces (like

the region across the mouth and nose, see the human face in Figure 1(a)) from these thousands of possible candidates, and uses them to build classifiers.

Let $C_j \approx \{c_j^i\}_{i=1..k}$ be a classifier based on k linear separators. Here, C_j classifies any sub-image W of a test image as a face if $V(C_j, W) = \sum_{i=1}^k \alpha_j^i \cdot c_j^i(W) \geq \frac{1}{2} \sum_{i=1}^k \alpha_j^i$ for some set of weights $\alpha_j = \{\alpha_j^i\} \in \mathbb{R}^k$ learned during training, where $c_j^i(W)$ is the boolean classification result of c_i on W as a face or non-face (see [9] for details). We refer to this $V(C_j, W)$ as the *SCO*-value (“sum of classifier output values”) of C_j on W .

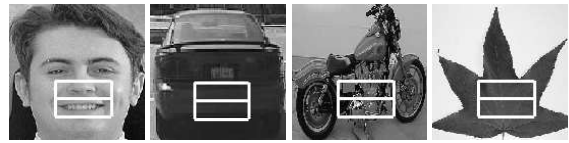
This approach can be used to detect objects that belong many other single classes (like cars, motorbikes, etc.) as well as just faces. “Many Class Detection”, which tries to detect and identify (i.e., assign a class label to) objects that belong to any of a set of different classes, is more challenging. One possible approach to solve this problem is to build one “single class Viola-Jones” (SC-VJ) cascade for each of the M class during training and run them *all* M during performance to identify objects of multiple classes. (We refer to this as the “M-SC-VJ” approach.) However, this does not scale up well; it requires running one cascade for each class of objects and is therefore M times more expensive. Moreover, it can be ambiguous if more than one classifier cascade labels an instance as positive. Another approach is to build one cascade of classifiers and use it to detect objects of multiple classes. That is, let $T = T^+ \cup T^-$ be a training set images of positive examples (T^+) and negative examples (T^-), such that $T^+ = \cup_{i=1}^M T_i$ where T_i has images of class i and T^- does not have any images of any of the M classes. We can run the VJ algorithm on this set to produce a classifier cascade, such that each classifier can detect objects of *any* of the M classes (with a certain false positive rate). This approach has two problems: (1) since this classifier return a single bit at performance time it just labels any detected object as a positive instance, but cannot identify this object

as belonging to a specific one of the classes. (2) A single classifier, built using objects of different classes as positive examples, can have a high false positive rate. This is not surprising: Many of these individual classes will naturally correspond to disjoint clusters (see below), and this classifier corresponds to their *union*. Any algorithm that attempts to form a convex hull around such disjoint clusters is likely to include many extraneous instances.

Our approach begins by using the VJ learning algorithm to build N classifiers that each attempt to distinguish the positives (here the union of M different classes, $T^+ = \cup_{i=1}^M T_i$) from the negatives T^- . We define its “classifier space” as the N -dimensional space formed by using the SCO -value of each of N classifiers as a dimension. That is, the N classifiers collectively map each input image to a point in the N -dimensional classifier space. We anticipate that the SCO -values of objects in a single class should be similar, and that objects from different classes should have different SCO -values. Our results show that this holds — in that many individual classes will form “clusters” in the classifier space; see Figure 1(b). If we can place a new image within a cluster, we can then assign that image the class label ℓ associated with that cluster; see Section 3.1.

Figure 1(b) shows clusters of four classes of objects (cars, leaves, motorbikes and faces), plotted using the SCO -values of 2 of the classifiers, on training images of these four classes of objects. Of course, the SCO -values of every pair of classifiers will not necessarily form clusters. It is possible that only one subset of classes form clusters wrt one subset of classifiers, while some other subset of classes may form clusters wrt another subset of classifiers. Further, there may be no unique set of classifiers that yields these clusters. We therefore use a dynamic process, that sequentially decides which classifier to apply next when dealing with an input image, based on the values observed from the classifiers previously executed on this instance. The challenge is to *learn* the dynamic sequence of classifiers that can effectively distinguish the clusters of different classes.

We formulate this task as a Markov Decision Process (MDP), and use dynamic programming to find an optimal policy — *i.e.*, sequence of classifiers to use to partition the training images into clusters. At run time, we *dynamically* select which classifier to apply, based on the responses of previous classifiers. Hence, the detector is in the form of a decision tree (see Figure 2) that is built using the SCO -values of the learned classifiers as features. We apply this tree to each sub-image W of a given test image. If all the classifiers on the path from the root to the leaf label this W positively, we tentatively assign to W the class label ℓ of the corresponding leaf. We then apply a cascade specific to this leaf, to W , to confirm that W is an instance of class ℓ . If any of the classifiers in the decision tree or the class specific cascade label W negatively, we stop processing W



(a) Features of different classes on a rectangular region

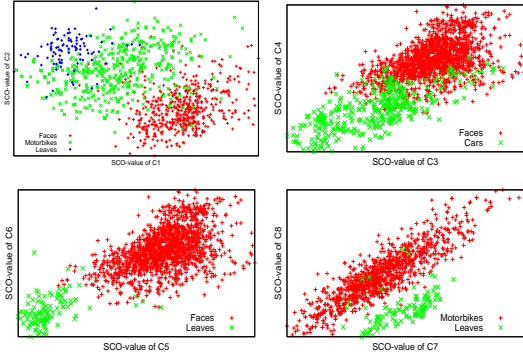


Figure 1. (b) Clusters of objects of the same class in classifier space

and proceed to the next sub-image.

To identify objects belonging to any of M classes, we can use the M-SC-VJ approach: compute M class specific cascades, each having around N classifiers, for a total of $M \times N$ classifiers. Using our detection method, however, by choosing classifiers carefully in the first stage, we can assign a tentative class label using $M_1 \leq M$ classifiers (recall clusters in Figure 1(b)), then run *one* cascade of length $N_1 \leq N$, for a total of $M_1 + N_1 \leq M + N$ classifiers; this is significantly faster than the M-SC-VJ approach.

Using four classes of objects, we show that our detection method has a detection rate comparable to a class specific Viola-Jones cascade. We also show that the performance time of our algorithm is much better than running M class specific cascades. After Section 2 provides the framework, Section 3 overviews how we address this “many class detection” task and Section 4 gives the results. Section 5 summarizes related work.

2. Framework

A *Markov Decision Process* can be described as a 4-tuple $\langle \mathcal{S}, A, M, R \rangle$ where $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ is a finite set of states, $A = \{a_1, a_2, \dots, a_m\}$ is a finite set of actions, $M: \mathcal{S} \times A \times \mathcal{S} \rightarrow [0, 1]$ is the state transition probability function ($M_{s,s'}^a = P(s' | s, a)$ is the probability that taking action a in state s leads to state s') and $R: \mathcal{S} \times A \rightarrow \mathbb{R}$ is the reward an agent gets for taking an action $a \in A$ in state $s \in \mathcal{S}$. A *policy* $\pi: \mathcal{S} \rightarrow A$ is a mapping from states to actions. [7] presents a good description of MDPs and the different ways to solve them. In this section, we explain how we formulate many class detection as an MDP.

States: We identify each node in a DTC with a state

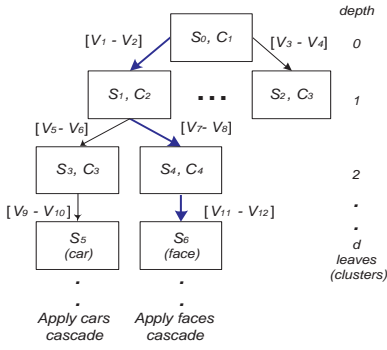


Figure 2. Decision Tree Classifier (DTC)

$s = \langle \vec{P}, C_1 : [V_{min,1}, V_{max,1}] \dots C_k : [V_{min,k}, V_{max,k}] \rangle$ that specifies the range of *SCO*-values of the classifiers $\langle C_1, \dots, C_k \rangle$ already applied to reach this node, and a posterior probability distribution over the class labels, $\vec{P} = \langle P_\ell \rangle$, where $P_\ell = P(class(W) = \ell | \langle V(C_1, W), \dots, V(C_k, W) \rangle)$ is the probability that ℓ will be the label of an instance W that reached this node s , based on the evidence, $\{V(C_i, W) \in [V_{min,i}, V_{max,i}]\}_{i=1..k}$ here. We are seeking a *policy* $\pi : S \mapsto A$ that specifies which classifier to apply in each state s , with the aim of reaching a leaf whose instances all belong to the same class. Figure 2 shows a simple DTC. The *SCO*-value of the classifier determines which branch to take.

Actions: The set of actions correspond to the classifiers that can detect objects of many classes; *i.e.*, $A = \{C_1, C_2, \dots, C_N\}$.

Reward: We assign a high reward to states that group objects of the same class together. We use the reward function

$$R(s, C_i) = \begin{cases} \max_\ell \{P(class(s) = \ell)\} & \text{if } depth = d \\ -\alpha \cdot FN(C_i, s) & \text{otherwise} \end{cases} \quad (1)$$

We assign the probability of the most likely class ℓ if $depth = d$, otherwise we penalize by $\alpha \times FN(C_i, s)$ where α is a constant (we set it to 0.1), and $FN(C_i, s)$ is the fraction of false negatives of C_i on the training images that would arrive in state s .

Transition Model: The transition model $M_{s,s'}^a$ is the probability that taking action (*i.e.*, applying some classifier) a in state s leads to s' . Let $s = \langle \vec{P}, C_1 : [V_{min,1}, V_{max,1}] \dots C_k : [V_{min,k-1}, V_{max,k-1}] \rangle$, and $s' = \langle \vec{P}', C_1 : [V_{min,1}, V_{max,1}] \dots C_k : [V_{min,k}, V_{max,k}] \rangle$. Consider attempting to classify a test subimage W . To reach s , we must have applied the sequence of classifiers $\langle C_1, C_2, \dots, C_{k-1} \rangle$ and found $V(C_i, W) \in [V_{min,i}, V_{max,i}]$ where “[$V_{min,i}, V_{max,i}$]” labels the associated arc for $1 \leq i \leq (k-1)$. Note $P(W \in s' | W \in s, V(a, W) = o) = 0$ for all actions $a \neq C_k$, since we can reach state s' from s only by applying C_k . During training, in any state s_1 , after applying

any classifier C , we partition all the images of s_1 that C classifies as positives into two equal halves based on $V(C, s_1)$. This results in two other states, s'_1 and s''_1 (see Section 3 and also [11] for details). As both s'_1 and s''_1 are equally likely, we have the transition probability $P(W \in s'_1 | W \in s_1, C_k) \approx P(W \in s''_1 | W \in s_1, C_k) = 0.5$

3. Many Class Detection and Identification

This section briefly describes how to learn a DTC and explains how to use it to detect objects of multiple classes. Please see [11] for details.

3.1. Learning DTC classifier

We use vJ to build N classifiers $\mathcal{C} = \{C_1, C_2 \dots C_N\}$ using the images of the training set $T = T^+ \cup T^-$.

Exploring sequences of classifiers: We explore every possible sequence of d classifiers on T^+ to find the sequence that yields the best clusters. That is, we first apply some classifier C_1 on each image $t \in T^+$. We remove all the images that C_1 labels as negatives. We sort the remaining images based on their *SCO*-values, $V(C_1, t)$, on these images, *i.e.*, $\langle t_1, \dots, t_{m/2}, t_{m/2+1}, \dots, t_m \rangle$, where $V(C_1, t_j) \geq V(C_1, t_k)$ when $j > k$. We split them into two equal halves $\langle T_1^L \rangle$ and $\langle T_1^R \rangle$ (denoting the left and right branches), such that $\langle T_1^L \rangle$ contains $\{t_1, \dots, t_{m/2}\}$ and $\langle T_1^R \rangle$ contains $\{t_{m/2+1}, \dots, t_m\}$. We then apply a classifier $C_2 \neq C_1$ on $\langle T_1^L \rangle$ resulting in $\langle T_1^L, T_2^L \rangle$ and $\langle T_1^L, T_2^R \rangle$ that each represents one half of the images of $\langle T_1^L \rangle$ that C_2 labeled as positives. Similarly, we apply any classifier $C_3 \neq C_1$ on $\langle T_1^R \rangle$ resulting in $\langle T_1^R, T_3^L \rangle$ and $\langle T_1^R, T_3^R \rangle$.

We repeat the process for d steps, applying a sequence of d classifiers, $\langle C_1, \dots, C_d \rangle$. For each sequence, the resulting 2^d leaves are clusters. Note that this is for one (random) sequence of d classifiers. When we consider the $P_d^N = N!/(N-d)!$ different sequences of d classifiers, it leads to a total of $P_d^N \times 2^d$ clusters. Note that many clusters can have the same class label.

Assigning utility to states: Each state s_d resulting after applying any random sequence of d classifiers, $\langle C_1 \dots C_d \rangle$, represents a cluster. We want to determine the best decision tree within this tableau, using those clusters that group images of only one class together. While exploring $N!/(N-d)!$ sequences of classifiers, let s_i be the state resulting after applying i classifiers, for $1 \leq i \leq d$. We use Equation 1 to compute $U(s_d)$, *i.e.*, we set

$$U(s_d) = \max_\ell \{P(class(s_d) = \ell)\} - \alpha \cdot \sum_{i=1}^d FN(C_i, s_i)$$

where $\alpha = 0.1$. Note that $FN(C_i, s_i) = \frac{m}{n}$ where n is the total number of images in s and m is the number that C_i misclassified as negatives. (see the example given later).

We use a dynamic programming approach to assign utilities. That is, we first compute $U(s_d)$, use those values to compute $U(s_{d-1})$ and so on. Using the $U(s_d) \dots U(s_{i+1})$ values, we set the utility $U(s_i)$ of any state s_i resulting after applying i classifiers. Let there be k possible classifiers that can be applied in s_i , i.e., none of these k classifiers were applied to reach s_i . We apply any classifier, C_j in s_i resulting in two states, $s_{i,jL}$ and $s_{i,jR}$. We compute $U(s_i) = \{P(C_j = L|s_i) \times U(s_{i,jL}) + P(C_j = R|s_i) \times U(s_{i,jR})\} = 0.5 \times \{U(s_{i,jL}) + U(s_{i,jR})\}$ ¹. Here L and R denote the left and right transitions to $s_{i,jL}$ and $s_{i,jR}$, respectively. We compute $U(s_i)$ after applying each of the k classifiers and set $U(s_i)$ to the maximum of those values. Similarly, we compute $U(s_{i-1}), \dots, U(s_0)$.

Building DTC: We collect the $\langle s_i, C_i^* \rangle$ pairs and also the corresponding utilities for various states. Here s_i denotes the state resulting after applying i classifiers, C_i^* denotes the classifier that, when applied to s_i , transitions it to another state s_{i+1}^* , with the maximum utility among the states resulting after applying one additional classifier to this sequence. Note that the various $\langle s_i, C_i^* \rangle$ pairs tell us precisely the i classifiers applied so far, their individual *SCO*-values and the best classifier C_i^* to apply in s_i . This corresponds precisely to the DTC.

Example: We illustrate the learning algorithm with the help of a simple example. Let $T^+ = \cup_{i=1}^M T_i$ be the set of positive examples such that $T_f = \{t_1 \dots t_{10}\}, T_l = \{t_{11} \dots t_{20}\}, T_m = \{t_{21} \dots t_{30}\}, T_c = \{t_{31} \dots t_{40}\}$ be images of faces, leaves, motorbikes and cars, respectively. Let C_1, C_2 and C_3 be three classifiers that can identify objects of all these classes. Further, let $d = 2$, i.e., we build a DTC to depth 2. We explore every possible sequence of two classifiers, on T^+ to build DTC. Let us consider one sequence $\langle C_1, C_2 \rangle$. Let s_0 be the initial state with all the 40 images of T^+ ; $P(\text{class}(s_0) = \ell) = \frac{1}{4}$ for each of the four classes, ℓ . As no classifier is applied to reach the initial state s_0 , we set $s_0 = \langle (0.25, 0.25, 0.25, 0.25); [] \rangle$. We first apply C_1 to each image in T^+ , and assume it classifies $\{t_1, \dots, t_{38}\}$ correctly, but misclassifies t_{39} and t_{40} as negatives; here $FN(C_1, s_0) = \frac{2}{40} = 0.05$. We use *SCO*-value $V(C_1, t)$ to sort the remaining 38 images $\{t\}$. Assume that $V(C_1, t_i) \leq V(C_1, t_j)$ for $i < j$. We split them into two halves $\langle T_1^L \rangle = \{t_1 \dots t_{19}\}$ with a *SCO*-value range of [100, 130] and $\langle T_1^R \rangle = \{t_{20} \dots t_{38}\}$ with a *SCO*-value range of [131, 200]. Notice $\langle T_1^L \rangle$ has 10 images of faces and 9 leaves and no images of any other class, which means $\bar{P}_1 = \langle 10/17, 9/17, 0, 0 \rangle$ denoting the probability of class label for each of faces, leaves, motorbikes and cars, respectively. Hence the state representation for $\langle T_1^L \rangle$ is $s_{1L} = \langle \bar{P}_1, [C_1 : 100, 130] \rangle$. As $\langle T_1^R \rangle$

¹We use a slightly different approach, we set $U(s_i)$ as the maximum of $U(s_{i,jL})$ and $U(s_{i,jR})$, see [11] for updated results

contains 1 leaf, 10 motorbikes and 8 cars, $\langle T_1^R \rangle$ will be $s_{1R} = \langle (0, 1/19, 10/19, 8/19), [C_1 : 131, 200] \rangle$.

We then apply C_2 from the sequence $\langle C_1, C_2 \rangle$ to $\langle T_1^L \rangle$, and assume it classifies all the images, except t_{19} , positively, i.e., $FN(C_2, s_{1L}) = \frac{1}{19} = 0.11$. Now assume that $V(C_2, \cdot)$ is also monotonic in $\langle t_1, \dots, t_{18} \rangle$ (i.e., $V(C_2, t_i) \leq V(C_2, t_j)$ for $1 \leq i < j \leq 18$). Once split into two halves, $\langle T_1^L, T_2^L \rangle = \{t_1, \dots, t_9\}$ and $\langle T_1^L, T_2^R \rangle = \{t_{10}, \dots, t_{18}\}$ with *SCO*-value ranges of [10, 60] and [61, 90], we have the state representation for $\langle T_1^L, T_2^L \rangle$ is $s_{1L,2L} = \langle (1.0, 0, 0, 0), [C_1 : 100, 130][C_2 : 10, 60] \rangle$. States $s_{1R,2L}$ and $s_{1R,2R}$ are clusters (since $d = 2$ classifiers are applied), with class labels ‘‘face’’ and ‘‘leaf’’, respectively. We can compute their utilities using Equation 1: $U(s_{1R,2L}) = 1.0 - 0.1 \times (0.05 + 0.11) = 0.984$ and $U(s_{1R,2R}) = 0.89 - 0.1 \times (0.05 + 0.11) = 0.874$.

We also consider $\langle C_1, C_3 \rangle$. Here, this means applying C_3 to $\langle T_1^L \rangle$, resulting in two clusters, $s_{1L,3L}$ and $s_{1L,3R}$, with utilities of say $U(s_{1L,3L}) = 0.6$ and $U(s_{1L,3R}) = 0.7$. In s_{1L} , we can apply one of C_2 or C_3 . So, $U(s_{1L})$ is the maximum of the utilities that can be achieved by applying either of these classifiers. That is, $U(s_{1L}) = 0.5 \times (0.984 + 0.874) = 0.929$, if C_2 is applied in s_{1L} . We can similarly compute $U(s_{1L}) = 0.65$, if C_3 is applied. As C_2 results in the maximum utility, it is the best classifier to apply in s_{1L} , i.e., $\pi(s_{1L}) = C_2$. We similarly compute $U(s_{1R,2L}), U(s_{1R,2R}), U(s_{1R,3R})$ and $U(s_{1R,3R})$; and use those values to compute $\pi(s_{1R}) = C_3$.

We also compute the utilities for sequences starting with C_2 and determine policies for other states, like, $\pi(s_{2L}) = C_1, \pi(s_{2R}) = C_3$ and then for sequences starting with C_3 , $\pi(s_{3L}) = \pi(s_{3R}) = C_2$. Finally, we would then compute the optimal action at s_0 is $\pi(s_0) = C_1$. All these policies result in the DTC similar to the one shown in Figure 2.

3.2. Detection

Our ‘‘many class detection algorithm’’ (MCDA) examines each 24×24 pixel window of a test image I_t ; it then rescales by a factor 0.8 (i.e., resizes the current height and width of I_t by a factor of 0.8) and repeats. For each window W , MCDA first applies the classifier C_1 associated with the root of DTC (see Figure 2). This might label W as a negative instance; if so MCDA continues with the next window. Otherwise, MCDA computes $V(C_1, W)$ and uses this value to find the resulting state s' . It then applies C_2 associated with s' , on W . Again this could reject W , if not, $V(C_2, W)$ identifies the next state s'' and classifier C_3 to apply on W . This continues for d steps, until W reaches a cluster. If all the d classifiers label W as a positive instance, MCDA finds the class label ℓ associated with the cluster. It then runs a SC-VJ cascade $\langle C_1^\ell, C_2^\ell, \dots, C_P^\ell \rangle$ associated with this leaf node and declares W to be an object of class ℓ if it passes all these classifiers. Otherwise, it rejects W as a negative instance.

4. Experimental Results

4.1. Experimental setup

Data Used: We used four classes of objects, cars (rear view), leaves, motorbikes and faces in our experiments. The training set of faces T_F contained 1600 images of faces (taken from popular face databases) and for test images used the MIT-CMU database of faces, which has a total of 178 images with 532 faces. We used images from Caltech image database [1] for the other three classes. Our training sets for cars, leaves and motorbikes — T_C , T_L and T_M have 476, 156 and 776 images, while our test sets have 50, 50 and 67 images, respectively, with a total of 67 cars, 50 motorbikes and 67 leaves. Our training set for the negative examples T^- contains 2320 images, none of which has any pictures of faces or cars or leaves or motorbikes.

Building Classifiers: We used T_C , T_L , T_M , T_F and T^- to build 4 SC-VJ cascade classifiers² (one for each class), that involved 18, 17, 17 and 21 classifiers. We also built $N = 10$ classifiers that can detect objects of any of the four classes. Since we have four different classes, and with the application of each classifier (carefully, using DTC) we can distinguish between two classes, we set $d = 3$. That is, we built a DTC upto a depth of 3 using our learning algorithm.

Training time: Our system required about 3 hours to build each of the 4 class specific cascades and another 1 hour to build the classifiers³. It then required about 5 minutes to build DTC, so the total training time was about 18 hours.

Results: We compared MCDA to the standard set of $M = 4$ SC-VJ cascades, with respect to accuracy, ROC curves and efficiency. Note that MCDA applies d classifiers (within DTC) to determine which class label to consider for each test sub-image, and then applies a cascade specific to that class. Each of the four SC-VJ detection algorithms has an easier task, as we explicitly identify which single class of objects it should seek for each image. This is why we do not expect the performance of MCDA to be better than SC-VJ, in terms of either efficiency or accuracy. However, our results indicate that MCDA does quite well in detecting objects as well as assigning class labels. In fact, our algorithm runs at least twice as fast as running M SC-VJ cascades to detect $M = 4$ classes of objects; see Section 4.4.

4.2. Accuracy and Execution Time

Figure 3 shows some test images in which MCDA could successfully detect cars, leaves, motorbikes and faces. Table 1 compares MCDA with the SC-VJ and M-SC-VJ algorithms in terms of accuracy and efficiency. The peak accuracy, as we vary the number of cascade classifiers at the

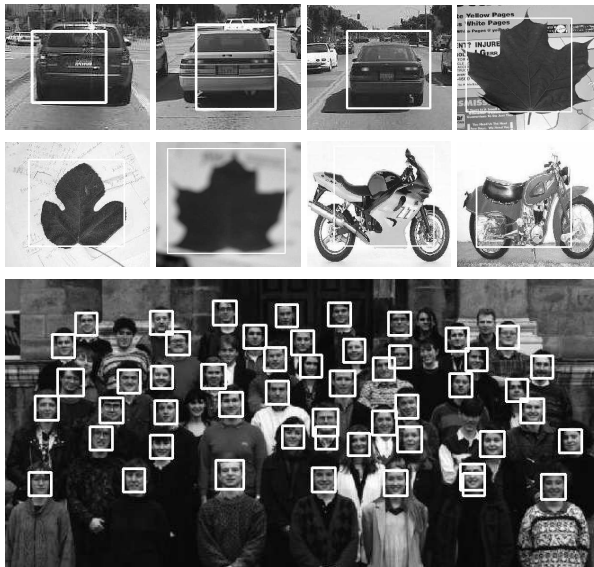


Figure 3. Performance on various test images

leaves,⁴ are given in Table 1. These values are statistically indistinguishable at $p < 0.05$. MCDA is slower than SC-VJ, by 63%, 83.7%, 67.7% and 22.26%. we attribute this to: (1) the time needed to run the extra $d = 3$ classifiers using DTC and (2) the overhead involved in assigning a class label to each sub-image of any test image. Note that this is much better than the obvious M-SC-VJ alternative.

4.3. ROC curves

Figure 4 compares the ROC curves of MCDA with that of the SC-VJ detection algorithm. In the graph, we plotted accuracy against the number of false positives per window processed. Figure 4 shows that the SC-VJ detection method has a slightly better performance than the many class detection method, while the overall detection for many class detection is comparable to SC-VJ.

4.4. Comparison to M-SC-VJ

On the test set of each class, we ran each of the four cascade classifiers (*i.e.*, M-SC-VJ). As expected, the execution time of this algorithm (Table 1) is linear in the number of classes, which means it will not scale up well. As MCDA does not need to run multiple cascades, it will scale up well.

5. Related Work

There has been a lot of recent interest in many class detection. Torralba *et al.* present a many class boosting procedure that shares features across different classes [8]. They train binary classifiers “jointly” (for several classes) and use the common features to detect objects of multiple classes.

²We use the Wu and Rehg [10] implementation of SC-VJ.

³All results presented here were run on a 1 GHz. Intel Pentium processor with 256 Mbytes of memory running Windows-2000.

⁴We define “peak accuracy” as the accuracy value with negligible rate of increase with increasing values false positives. We found it manually.

Class	TestData			#Windows	Peak Accuracy		Av.Detcn.Time(sec)		
	#Images	#Objects	Av.Image Size		SC-VJ	MCDA	SC-VJ	MCDA	M-SC-VJ
Cars	50	65	265× 360	10,114,613	87.69%	86.15%	0.495	0.806	1.787
Leaves	67	67	318× 436	20,607,663	97.01%	95.52%	0.454	0.834	2.006
Motorbikes	50	50	279× 297	8,680,218	97.0%	92.0%	0.574	0.963	1.912
Faces	169	532	403× 402	76,957,710	92.11%	92.0%	1.541	1.883	4.558

Table 1. Comparison of test results for SC-VJ cascade and MCDA algorithm

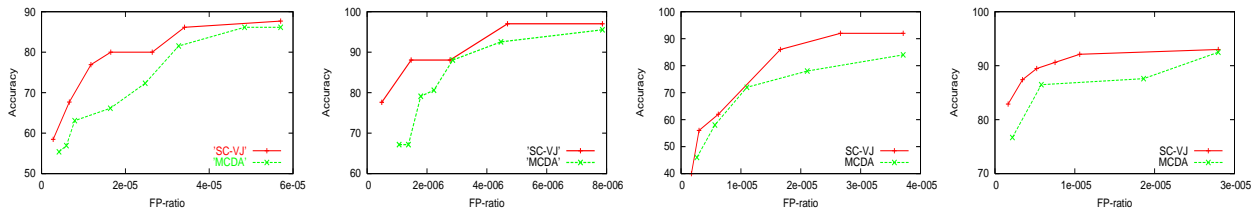


Figure 4. ROC curves for SC-VJ cascades and MCDA for (a) Cars (b) Leaves (c) Motorbikes (d) Faces.

We have implicit feature sharing because we use rectangular features of many classes to build classifiers. But our work is significantly different, feature sharing is not our main focus. Fan [2] presents an algorithm that learns a hierarchical partitioning of the hypothesis space. They test their algorithm to detect handwritten digits. Li *et al.* [6] use a generative probabilistic model to represent the shape and appearance of a constellation of features of an object. They learn the parameters of the model incrementally in a Bayesian manner. They test it on 101 different object categories. Our work is significantly different from the two approaches given above. We used similar techniques to identify the facial expression of a face during face detection, by formulating the problem as MDP and use dynamic programming to solve it [5]. We also addressed related issues in a feature-based face-recognition system by posing the task as MDP [4]. Our current work is similar to these two methods, but here, we find the best sequence of classifiers to assign class labels for each sub-image, by matching them to clusters.

6. Conclusions

This work provides a method of using binary classifiers to detect and identify objects of many classes. We show that images of the same class form clusters in the classifier space of classifiers, then present examples of clusters using four classes of objects. We motivate the need to select the classifiers dynamically and formulate the problem as an MDP. We use dynamic programming to find a good policy, represented as a decision tree classifier, DTC. Our detection algorithm (MCDA) uses DTC and tentatively assigns a class label ℓ to each subimage W that DTC labels as a positive instance. It then applies a class-specific cascade to confirm that W as an instance of ℓ . We present empirical results that demonstrate that our ideas work effectively: showing that our results are comparable to the SC-VJ algo-

rithm in terms of accuracy, efficiency and ROC-curves, but much faster than running M different SC-VJ classifiers.

References

- [1] R. Fergus, P. Perona and A. Zisserman. Object class recognition by unsupervised scale-invariant learning, *CVPR*, 2003
- [2] X. Fan. Efficient multiclass object detection by a hierarchy of classifiers, *CVPR* 2003
- [3] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, *Computational Learning Theory: Eurocolt*, 1995.
- [4] R. Isukapalli and R. Greiner. Use of Off-line Dynamic Programming for Efficient Image Interpretation, *IJCAI*, 2003
- [5] R. Isukapalli, A.Elgammal and R. Greiner. Learning to Identify Facial Expression During Detection Using Markov Decision Process, *Face and Gesture Recognition Conference*, 2006
- [6] F.F. Li, R.Fergus and P. Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories, *Proceedings of the Workshop on Generative Model Based Vision*, 2004
- [7] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, 1998.
- [8] A. Torralba, K. Murphy and W.T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection, *CVPR*, 2004.
- [9] P. Viola and M. Jones. Robust real-time face detection, *IJCV* 2004
- [10] J. Wu, J.M. Rehg and M.D. Mullin. Learning a rare event detection cascade by direct feature selection, *NIPS* 2003.
- [11] <http://www.cs.ualberta.ca/~greiner/Research/ManyClassesIdentifier>