Integrating object detectors

Dave Grosvenor
Media Technologies Laboratory
HP Laboratories Bristol
HPL-2007-66
April 30, 2007*

ace detector, object detector, integration, expected computational cost, cascade of classifiers, decision tree

This paper describes a method for integrating object detectors that reduces the expected computational cost of evaluating all the detectors whilst obtaining the same logical behaviour as running the detectors independently. The method combines the decision trees of the different object detectors into a single composite decision structure controlling the evaluation of the classifiers from all of the original object detectors. The method intersperses the classifiers from the different object detectors allowing the evaluation of any object detector to be dependent on classifier results from other object detectors. The method exploits these extra results to rearrange the order in which classifiers from a particular object detector are evaluated. These rearrangements preserve the logical behaviour of the object detector whilst changing the expected computational cost of evaluating the decision structure.

All the object detectors are rare event detectors and so this method improves performance by exploiting the common need to reject the majority of patches being searched. Simplistically improved performance is obtained by rejecting non-object patches as soon as possible. The main contribution of this paper is the language used to formally express the problem of integrating the decision structures of different object detectors to obtain improved performance. Expressions for the expected computational cost of the integration are given, and the data structures defined for the single composite data structure. However additional work is needed to quantify the improved performance.

The paper was initially written as a HP invention disclosure which resulted in a patent application. The patent application is more complex and difficult to read because it generalised the method to work with a set of more general decision structures and adapted the method to be used with chaining or binning by allowing rearrangements of decision structures that did not preserve logical behaviour. More information on these generalisations is given on the web page: http://w3.hpl.hp.com/people/dag/integrating_object_detectors/overview.htm

* Internal Accession Date Only
Approved for External Publication

Integrating object detectors

Dave Grosvenor
11th April 2007

Abstract

This paper describes a method for integrating object detectors that reduces the expected computational cost of evaluating all the detectors whilst obtaining the same logical behaviour as running the detectors independently. The method combines the decision trees of the different object detectors into a single composite decision structure controlling the evaluation of the classifiers from all of the original object detectors. The method intersperses the classifiers from the different object detectors allowing the evaluation of any object detector to be dependent on classifier results from other object detectors. The method exploits these extra results to rearrange the order in which classifiers from a particular object detector are evaluated. These rearrangements preserve the logical behaviour of the object detector whilst changing the expected computational cost of evaluating the decision structure.

All the object detectors are rare event detectors and so this method improves performance by exploiting the common need to reject the majority of patches being searched. Simplistically improved performance is obtained by rejecting non-object patches as soon as possible.

The main contribution of this paper is the language used to formally express the problem of integrating the decision structures of different object detectors to obtain improved performance. Expressions for the expected computational cost of the integration are given, and the data structures defined for the single composite data structure. However additional work is needed to quantify the improved performance.

The paper was initially written as a HP invention disclosure which resulted in a patent application. The patent application is more complex and difficult to read because it generalised the method to work with a set of more general decision structures and adapted the method to be used with chaining or binning by allowing rearrangements of decision structures that did not preserve logical behaviour. More information on these generalisations is given on the web page:
http://w3.hpl.hp.com/people/dag/integrating_object_detectors/overview.htm

Introduction

Current object detectors work by brute force searching the whole image over all scales and orientations. In this search, each object detector repeatedly applies a cascade of weak classifiers to all possible patches. This invention is a method of integrating the cascades of different object detectors into another data structure that is evaluated during the brute force search.

The invention reduces the expected (or average) computational cost of a patch. This invention exploits the dependencies between the evaluations of the classifier cascades inside each object detector. An extra training stage is introduced which takes as input:

1. A set of classifier cascades for the input object detectors.
2. Statistical information about the inter-dependence between evaluations of an objects classifier given previous results of other classifiers.

It generates an N-object decision tree that is evaluated instead of the individual cascades from the object detectors. The evaluation of the N-object decision tree produces a result that is logically equivalent to evaluating each cascade in sequence. The N-object decision tree pre-computes both an interleaving of the classifier evaluations from the different object detectors, and adapts the order of classifiers by using the results from all the object detectors. This method predicts the expected cost of evaluating an N-object decision tree. This cost function is used to minimise the expected computation cost of evaluating every object detector.

Since the occurrence of an object is a rare event this cost is dominated by the cost of rejecting the regions that are not objects. This invention uses the computational steps of the other object detectors to re-order another object-

detector's cascade. The re-ordering ensures that the region is rejected sooner. At any stage of the decision tree the knowledge gained from previously evaluated classifiers is used to optimise the chance of rejecting a region as a candidate object. This is done by choosing the next classifier from any of the object detectors and using the classifiers from any one detector in a different order than given.

Since all the object detectors are rare event detectors this invention essentially exploits the common need to reject the majority of patches and merges the non-object detection functionality of early parts of a detectors cascade.

Problems Solved

The current learning algorithm used to determine the classifier cascades for an object-detector does not optimise a computational cost function during its search. For a single detector the ordering of classifiers in increasing complexity order probably produces results close to the optimum.

However when multiple objects detectors are integrated the computational cost rises linearly with the number of detectors. This invention provides a method that reduces the computational cost by exploiting the common need for each object detector to reject non objects quickly in the initial stage of the cascade.

So this invention provides improved performance of brute-force object detectors structured as a cascade or decision tree of weaker classifiers.

This invention enables a different approach to object detection and classification. It allows the use of more specific object detectors (such as child detector, men, women, spectacle wearer) that share the need to reject many of the same non-objects. This allows the Viola-Jones training to be performed on classes of objects with less variability within the class, enabling better individual detectors to be obtained and then using the invention to reduce the computational burden of integrating these more specific object detectors.

The development of a multi-pose or multi-view object-detector poses similar problems to that addressed by this invention. Different object detectors for a particular view are integrated into a single hybrid detector.
Aspects of this invention could be applied to this problem. Furthermore if the objects being integrated together are multi-view objects this will introduce predictable dependencies between the classifier computations

Advantages

This invention reduces the expected computational cost of evaluating a set of object detectors.

This invention does not alter the logical behaviour of any of the set of object detectors that are being evaluated.

This invention provides a means of merging the computationally significant parts of a set of object detectors, namely the early classifiers that cause the non-objects to be rejected early.

This invention re-uses the cascades obtained by individual training that produced a detector with particular accuracy characteristics (detection and false positive rates). Aspects of this invention will use the prior optimisation of cost for a single cascade for an initial search for an N-object decision tree with minimal expected computational cost.

Prior solutions

There is prior work on improving the computational performance of the Viola-Jones Face detector in "Speeding up the Viola-Jones face detector" [2] .This is concerned with the performance of a single detector, but it indicates the difficulty of improving the performance. This work does not consider the re-ordering of the cascade, which is determined by the training phase. The speed-ups considered come from reducing the number of patches that are considered through manipulating the sampling grid or through pre-filtering. The sampling grid manipulation was successfully used to improve speed, whereas pre-filtering was less successful (other than on skin colour) because the cascade of weak classifiers provides its own pre-filtering.

"Joint induction of shape features and tree classifiers" Y. Amit, D. Geman, and K.Wilder, IEEE Trans. PAMI, 19:1300--1306, 1997

This is prior work on the use "decision trees" [8] referenced by Viola-Jones. The cascade is a degenerate form of binary decision tree. These "decision trees" are similar N-object decision trees used in this invention which combines the cascades from the different object detectors

"Robust Object Detection via Soft Cascade**"** Lubomir Bourdev, Jonathan Brandt, CVPR 2005

This is the most recent

US20050013479 A1 "Robust multi-view face detection methods and apparatuses"

This is related to the

"Robust multi-pose face detection in images", Rong Xia0, Ming-Jing Li, Hong-Jiang Zhang, IEEE transactions on circuits and systems for video technology [4]

Description

Introduction

Current object detectors work by brute force searching the whole image over all scales and orientations. In this search, each object detector repeatedly applies a cascade of weak classifiers to all possible patches. This invention is a method of integrating the cascades of different object detectors into another data structure that is evaluated during the brute force search.

The invention reduces the expected (or average) computational cost of a patch. This invention exploits the dependencies between the evaluations of the classifier cascades inside each object detector. An extra training stage is introduced which takes as input:

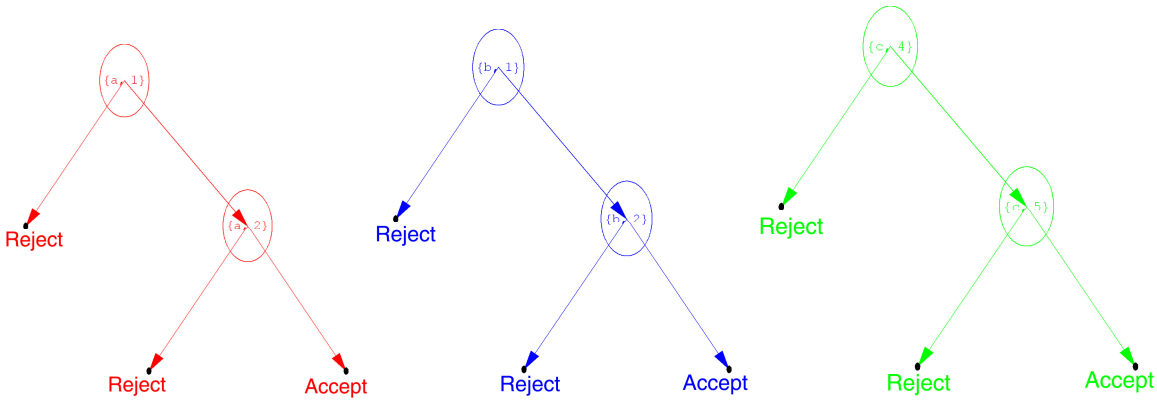
1. A set of classifier cascades for the input object detectors.
2. Statistical information about the inter-dependence between evaluations of an objects classifier given previous results of other classifiers.

It generates an N-object decision tree that is evaluated instead of the individual cascades from the object detectors. The evaluation of the N-object decision tree produces a result that is logically equivalent to evaluating each cascade in sequence. The N-object decision tree pre-computes both an interleaving of the classifier evaluations from the different object detectors, and adapts the order of classifiers by using the results from all the object detectors. This method predicts the expected cost of evaluating an N-object decision tree. This cost function is used to minimise the expected computation cost of evaluating every object detector.

Since the occurrence of an object is a rare event this cost is dominated by the cost of rejecting the regions that are not objects. This invention uses the computational steps of the other object detectors to re-order another object-detector's cascade. The re-ordering ensures that the region is rejected sooner. At any stage of the decision tree the knowledge gained from previously evaluated classifiers is used to optimise the chance of rejecting a region as a candidate object. This is done by choosing the next classifier from any of the object detectors and using the classifiers from any one detector in a different order than given.

Since all the object detectors are rare event detectors this invention essentially exploits the common need to reject the majority of patches and merges the non-object detection functionality of early parts of a detectors cascade.

For example, we input three object detectors



and can generate the N-object decision tree below

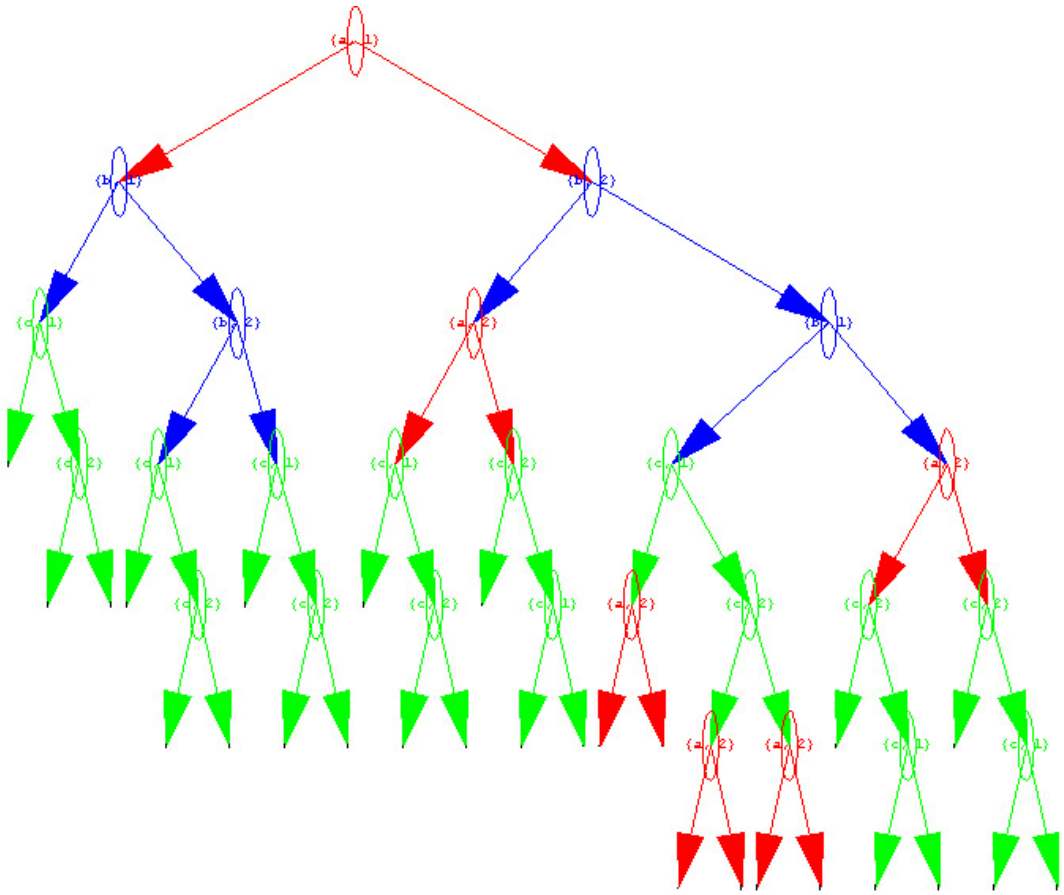


Figure 1 N-object decision tree

The evaluation of the N object decision tree computes the result of every object detector and only uses the same classifiers from the cascades input. Every N-object decision tree is the result of interleaving and re-ordering the classifier order of the original input cascades.

- Re-ordering a single cascade maintains the logical behaviour (object classification) of the cascade, but alters the expected computational cost of that object detector.
- The use of the decision tree allows the results from every classifier evaluation to be used to re-order an objects cascade to reduce the expected computational cost, given the results from evaluating the earlier classifiers.
- The interleaving of classifier evaluations from different object detectors allows prior information to be built up from any object.
- When evaluating a cascade the only knowledge prior to evaluating a classifier is that the earlier classifiers in the cascade have been accepted. For an N-object decision tree there are results from other objects classifiers. Furthermore the results from other objects can include rejection of that object's hypothesis.

In exceptional circumstances, the extra knowledge obtained from the overall set of classifiers evaluated makes a classifier in a cascade redundant. In some cases this means the object-detector immediately rejects the patch. In others, it means removing a classifier from the remaining cascade.

This invention allows the use of more specific object detectors (such as child detector, men, women, spectacle wearer) that share the need to reject many of the same non-objects. This allows the Viola-Jones training to be performed on classes of objects with less variability within the class, enabling better individual detectors to be obtained and then using the invention to reduce the computational burden of integrating these more specific object detectors.

The development of a multi-pose or multi-view object-detector poses similar problems to that addressed by this invention. Different object detectors for a particular view are integrated into a single hybrid detector.
Aspects of this invention could be applied to this problem. Furthermore if the objects being integrated together are multi-view objects this will introduce predictable dependencies between the classifier computations. However the geometric relationship between the views ought to be exploited more directly.

Computational cost

Current object detectors work by brute force searching the whole image over all scales and orientations. The computational cost of evaluating a single object detector is significant. The computational cost of evaluating N object detectors is proportional to the number of objects. Thus currently it is only practical to search for highly salient objects, such as faces, and to finely tune the granularity of the search space so that fewer regions or patches are tested for the presence of an object.

This invention is a means of reducing the expected cost of evaluating all the object detectors by exploiting the inter-dependence between the weak classifiers (feature detectors) in the different object detectors. Most object detectors are rare-event detectors and share a common need to quickly reject most regions that are non-objects. This common need introduces dependences between the classifiers that is exploited by this invention.

The computational cost of evaluating N object detectors:

1. Preparation cost for each detector type (e.g. preparing the integral image or image pyramid)
2. Search over all regions within the image over varying scales and orientation
3. Perform the classifier computation of every object-detector on each region (e.g. executing the cascade classifier or decision tree)

If all the object detectors are the same type then the preparation cost will be independent of the number of detectors.

There is some initial setup computation for the different scales and orientation for each scale and orientation. The brute-force search strategy can be modified for each detector depending upon the detector response (work varying sampling grid of the search is reported in [2]). In general, this manipulation of the sampling grid will be specific to particular detector as well as the particular scale and orientation of the patches.

The classifier computation tests a region or patch. The classifier computation is dependent on the number of objects. There are other computational costs of memory access for regions of the image at different scales and orientations.

This invention takes a simple statistical model of the expected computational cost of executing a set of object detectors and ignores much of the practical issues that also affect the computational costs.

Realistic object detectors have 25-30 classifiers in their cascades. This involves far greater complexity than has been simulated with brute force searching (see the Mathematica notebook attached to the invention disclosure). However such complex systems can be reduced to a simpler cascade or 4-5 classifiers by combining the later cascades into a single classifier with a large cost. This step (whilst not optimum) is justified because:

- The expected cost is dominated by the earlier classifiers that reject most of the patches.
- The end of the cascades is usually arranged to test the more complex features that are more specific to a particular object detector and so are likely to be independent of the other object detectors anyway.

Even so the brute force searching of all possible N-object decision trees needs to be replaced by some search procedure based upon transformations generating logically equivalent N-object decision trees with reduced cost.

This invention provides a general way of combining the evaluations of the classifiers from different object detectors. The primary embodiment applies the same patch to each object detector. In this case the N-object decision tree would occur in the innermost loop searching patches from the whole image, over all possible scales and rotations.

But this need not be the case since any patches that are not independent could be combined using this invention.

In other embodiments of this invention the object detectors integrated into an N-object decision tree could work on different patches giving greater inter-dependence between the detectors. This would complicate the brute search procedure in which this invention replaces the call to each object detectors cascade.

In the extreme case every call of the object detector made during the search over all patches of the image, over all scales and orientations could be combined together.

But patches from different spatial regions of the image should be independent if they are far enough apart and so there should be no benefit from combining them using this invention. In fact, in the HP face detector the inter-dependency of patches close together is already exploited -- the granularity of the search space is optimised to examine as few patches as possible (see[2]). This optimisation is highly specific to the set of objects integrated. So it is very likely that close patches (in scale, translation, orientation) are inter-dependent both for the same object, and other detectors. Differences in the scale of means an object detector with a larger patch could be related to several object detectors with a smaller patch. For example, we might combine a face detector with an ear, nose, mouth or eye detector. This might mean each object detector uses a different sized patch and sometimes a set of possible translations from each other.

The basic method

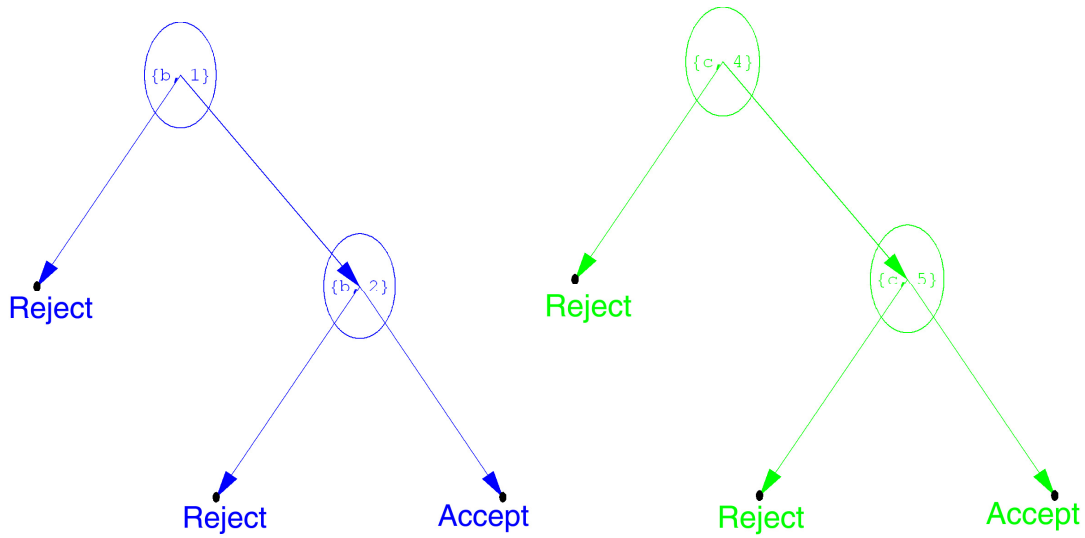
The primary method provided by this invention:

1. Input a set of N object-detectors each with their own cascades that have been trained and statistically characterised on the space of patches. Ideally each cascade should be individually in a computationally optimum order.
2. Find an N-object decision tree that both is an interleaving of the input cascades and which minimises the expected overall cost of evaluating all N object detectors.
    a. This minimisation requires the computation of the conditional probabilities of a classifier (from a particular object detector) accepting a patch given the results from evaluating earlier classifiers from any object detector.
    b. These conditional probabilities allow the computation of the expected cost of evaluating the N-object detectors on a patch (over all possible patches – determined over allow images, scales, rotations, etc..).
    c. The number of conditional probabilities required is considerable
    d. The space of logically equivalent N-object decision tree contains many decision trees that have identical costs because the computations are not inter-dependent

This method combines the input object detectors (cascades) and generates an N-object decision tree. This decision tree encodes

a. The "best" classifier to evaluate at each stage taking into account the results of evaluating previous classifiers,
b. Certain branches of the decision tree will re-order the classifiers for an object detector compared to others.
c. In special cases, it is possible to remove redundant classifiers from each cascade. An example of such removal occurs in the existing face detector where the first classifier in each cascade is always a variance test for the patch.

An example of combining two cascades into an N-object decision tree:



We represent rejection of the patch by a classifier using a left branch and use the right branch to represent acceptance by a classifier.

There are many possible ways to interleave the evaluation of the input cascades. Many of these will produce the same expected computational cost.

If we simply sequence the evaluation of the two cascades (i.e. evaluate the blue cascade followed by the green cascade) then we obtain the N-object decision tree below:

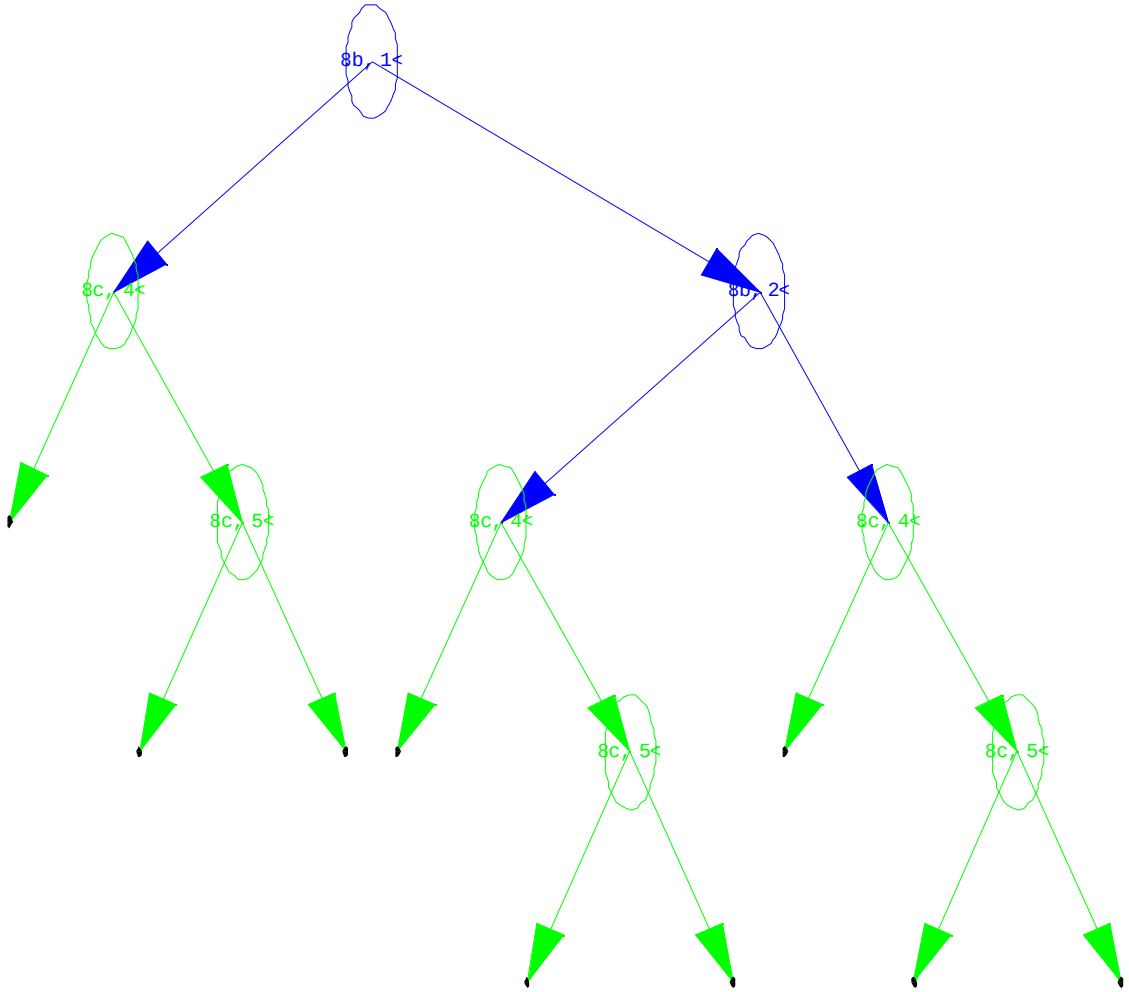


Figure 2 Sequence of two cascades

The N-object decision tree calculates the result of evaluating the N-object detectors.

All of the green sub-trees are identical showing that the evaluation of the green object detector is independent of the execution of the blue cascade.

Similarly we could evaluate the green cascade before the blue cascade.

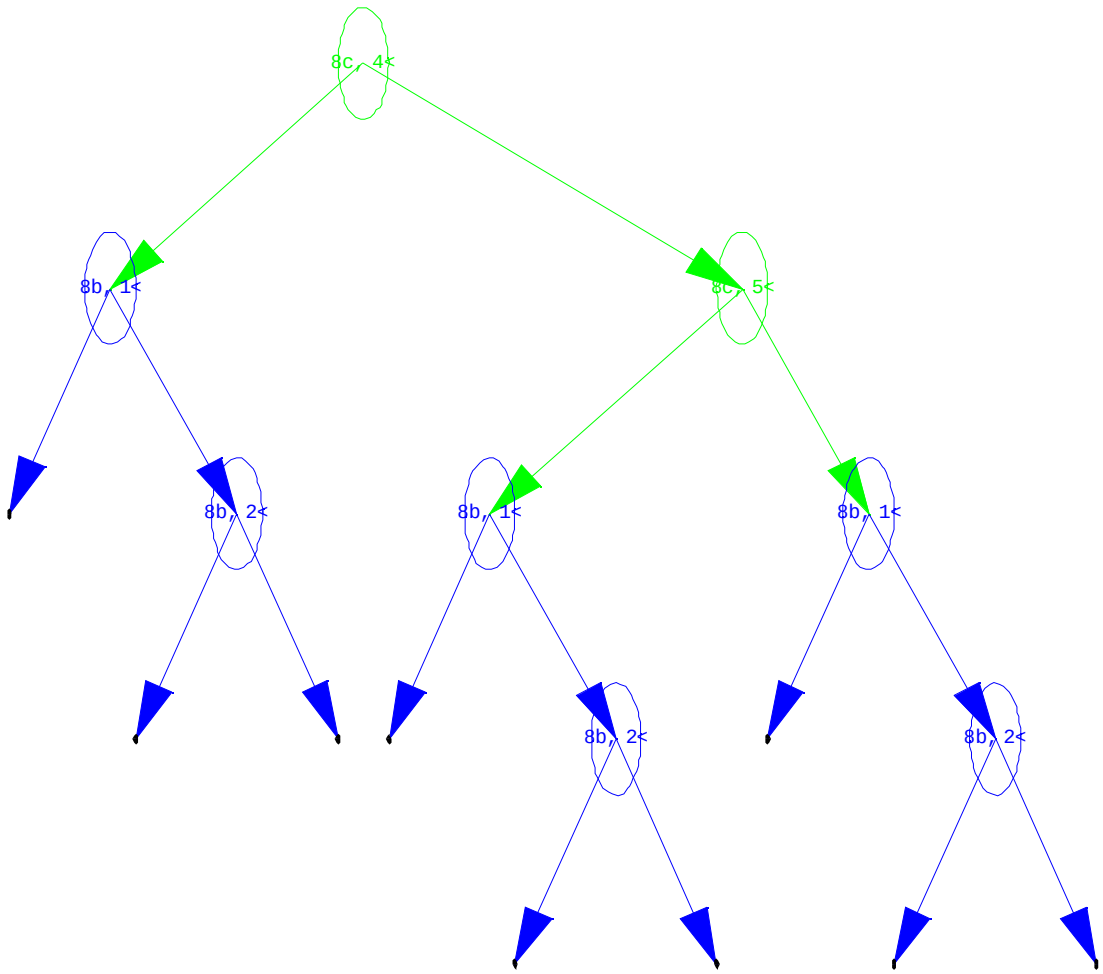


Figure 3 -- An alternative sequencing of two cascades

Again the evaluation of the following cascade (blue) is independent of the evaluation of the first cascade.
This N-object decision tree will always have the same cost as the previous decision tree obtained by pure sequencing the input cascades.

An aspect of this invention is that the computation of the input cascades is interleaved. Here we evaluate the first classifier from the blue cascade prior to always evaluating the first classifier from the green cascade.

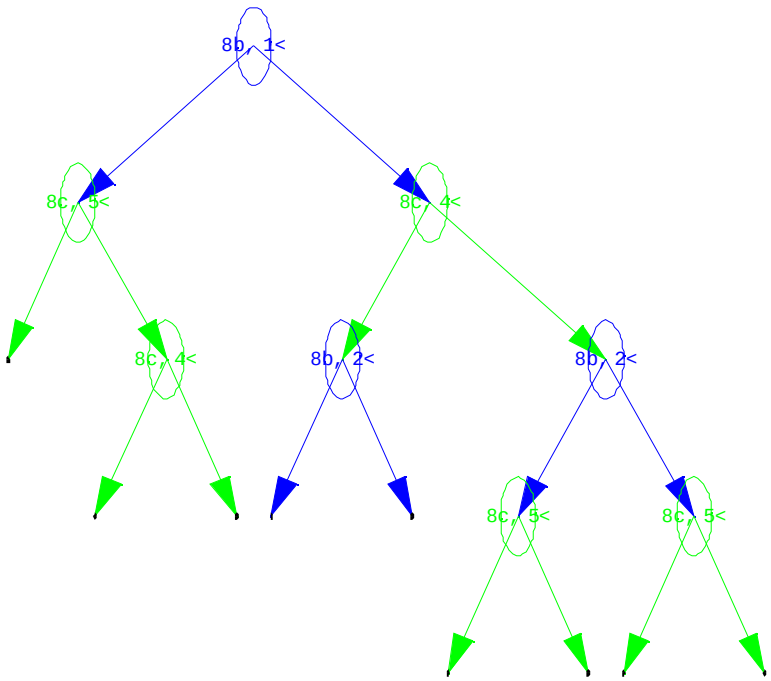


Figure 4 Overlapping independent evaluations

However since the evaluation order of the classifiers from each object detector is unchanged we obtain the same computational cost from this interleaving. In general, if we restrict an N-object decision tree to the labels of a single input cascade then we get a set of cascades that are permutations or re-orderings of the original cascade. But in the particular cases of the decision trees we have examined previously, the restriction to the labels of one cascade only ever produces a set with one cascade. This is because the evaluations of the different object detectors are independent of each other.

A further step is required to obtain some difference in the expected computational cost.

If we consider a single cascade of classifiers then we can re-order the classifiers to obtain logically equivalent cascades. However the expected computational cost of each permutation will be different. The expected cost is affected by both the cost of each classifier and the conditional probability of a patch being accepted given the results from the previous classifiers in the cascade.

The order of the cascade for each object detector can be optimised for each detector independently of other detectors. Although currently this is not done formally, the cascade is arranged in increasing order of complexity and each classifier is selected to optimise target detection and false positive rates. However this arrangement of the cascade has been found to be computationally efficient. Most patches are rejected by the initial classifiers. The initial classifiers are very simple and reject around 50% of the patches whilst having low false negative rates. The later classifiers are more complex, but have less effect on the expected computational cost.

Given the evaluation of classifiers from other object detectors we change the evaluation order of the classifiers of another object detector. This allows the results of a classifier from another object detector to influence the evaluation of another object detector.

An example of such an N-object decision tree is given below:

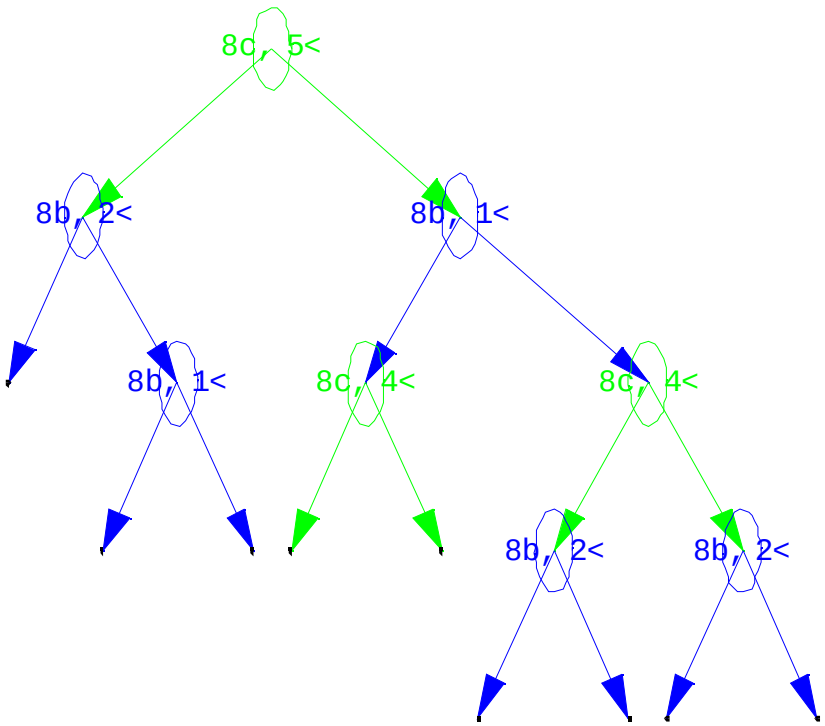


Figure 5 non-independent evaluation

Here we find that depending on the result of evaluating a green classifier a different blue classifier is evaluated next. This arrangement produces different expected computational costs.

However the evaluations of both sub-cascades are independent of each other after the (c,5) green classifier there are many other decision trees that would have the same expected computational cost.

The independence or near independence (when the expected computational cost is approximately the same) of evaluating object cascades is an important means of reducing the combinatorial explosion in the number of decision trees that need to be explored.

Performance of a single object detector

Current object detectors work by brute force searching the whole image over all scales and orientations. Since objects (such as faces) are rare events the computational cost is dominated by the cost of rejecting the regions that are not-objects. The HP face detector is a development of the Viola-Jones face detector [1]. This uses a set of weak binary classifiers constructed from single rectangular image features. A stronger face classifier is obtained by arranging these weak classifiers into a sequence or cascade of tests where a non-face is signalled by the failure of any one test, and a face is only signalled if all the tests succeed.

The cascade of tests can be re-ordered to produce an equivalent cascade that will produce the same logical results or classify the same faces. However the expected computational cost of executing a cascade can be altered by re-ordering.

Appendix 1 gives an expression for the expected cost of a single cascade of weak classifiers
This cost is affected by:

- The expected cost of evaluating a classifier.

- The conditional probabilities of accepting a patch given the prior acceptance of the earlier classifiers in the cascade.
- The order of the classifiers in the cascade matters because the order affects the prior knowledge applied to the conditional probabilities.

The set of weak classifiers is determined by the "AdaBoost" learning algorithm. The training parameters (target detection and false positive rates – see [6] [7]) at each stage only provide indirect control over execution speed of the final cascade. i.e. the training phase does not explicitly model the computational cost and the search does not explore the trade-off between computational cost and detector accuracy.

Generally the classifiers are arranged so that the early classifiers are simple features while later ones use more complex features so that non-faces are quickly discarded. However this need not be optimal and some re-ordering of the cascade could yield some improved performance. Although with a single detector the choice of the initial classifiers in the cascade are most crucial (and these almost certainly need to be the simplest classifiers).

Performance of multiple object detectors

A particular N-object decision tree represents a possible means of interleaving the evaluation of the classifiers from the set of object detectors input whilst achieving the same logical behaviour of evaluating every object detector independently.

In Appendix 2, we give both a definition of an N-object decision tree, and the properties that relate an N-object decision tree to the set of original object detectors. In addition, an expression is given for the expected cost of evaluating a patch using an N-object decision tree. This cost function is used to minimise the expected computational cost of evaluating a patch using a particular decision tree.

This cost is affected by:

- The expected cost of evaluating a classifier.
- The conditional probabilities of accepting a patch given the prior acceptance of the earlier classifiers in the cascade.
- The order of the classifiers from a particular object matters because the order affects the prior knowledge applied to the conditional probabilities.
- Similarly the order in which classifiers are taken from the different object detectors can matter because this will also affect the prior knowledge applied to the conditional probabilities. i.e. the interleaving of the different objects detectors affects the cost

For a single cascade it is possible to re-order the classifiers to obtain another cascade with the same logical behaviour (but different expected computational cost). Similarly it is possible to transform one decision tree to another decision tree with the same logical behaviour. Again whilst the logical behaviour is the same for both decision trees they have different expected computational costs.

It is also possible to show that many of these equivalent decision trees will have the same expected computational cost. This arises when the evaluation represented as a decision tree proceeds independently. Thus any search of the equivalent decision trees should only explore those that can yield different expected costs.

Conditional probabilities for classifiers accepting an event

This invention requires the determination or estimation of statistical information about the inter-dependence of the classifiers occurring in the set of input object detectors. This information is used to determine the cost of an N-object decision tree and so is used to minimise the expected cost when generating an optimum N-object decision tree (or event one with reduced cost).

Another alternative to this approach is to model the cost function for the N-object decision tree directly, such as is done by McCane[6] for a single cascade rather than an N-object decision tree.

In the Mathematica notebook attached to this invention disclosure there is a routine that generates the conditions that can occur in the evaluation of an N-object decision tree. The conditions that are the acceptance or rejection of a previously evaluated classifier in the decision tree generated from the set of input object detectors.

The notebook function enumerates those conditions after one classifier event, two classifier events, until every classifier has been evaluated. A classifier is only ever evaluated once, and no further classifiers from one object detector are evaluated after any of its classifiers have been rejected.

Thus for the simple case that we have examined in this section (consisting of two object detectors each with a cascade of two classifiers).

The possible conditions after one classifier event are:

```
   8<     88b, 1<<
   8<     88b, 2<<
   8<     88c, 4<<
   8<     88c, 5<<
88b, 1<<     8<
88b, 2<<     8<
88c, 4<<     8<
88c, 5<<     8<
```

Figure 6 Conditions after one event

This consists of two columns:- the first column for the classifiers that have been accepted and the second column for those classifiers that have been rejected. So there are 8 conditions after one classifier event.

For each classifier we need to determine the probability of this classifier accepting a patch after each of the eight conditions above. For the {b,1} classifier we obtain a list of values for each of the conditions:

```
80., 0., 0.360771, 0.58584, 1., 0.487999, 0.141349, 0.816197<
```

For a “valid” N-object decision tree derived from a set of input cascades there are entries in this list there are never looked up (they have probabilities assigned to either 0 or 1) . In particular, another “b” classifier is never tested after any “b” classifier is rejected, but for clarity we assign it a probability of zero indicating it is rejected. Similarly, every classifier is only ever evaluated once, but again for clarity we assign the probability of the {b,1} classifier being accepted after it had previously been accepted as 1.0.

We have four classifiers ({b,1},{b,2},{c,4},{c,5}) so we obtain a table:

```
 80., 0., 0.360771, 0.58584, 1., 0.487999, 0.141349, 0.816197<
80., 0., 0.047161, 0.470576, 0.842053, 1., 0.0895926, 0.51444<
80.773998, 0.774143, 0., 0., 0.441773, 0.235777, 1., 0.574468<
80.165935, 0.880688, 0., 0., 0.549061, 0.903405, 0.930554, 1.<
```

Again this table contains entries that will never be called by an N-object tree derived from a set of input cascades, and these entries have probabilities that are assigned as 0 or 1.

The other values have been generated by a random number generator and do not have any significance, but they were used to simulate the algorithm on simple examples. For a single classifier the numbers should be about 0.5 for each stage of the cascade, but we have no idea of the conditional probabilities between classifiers from different detectors.

Similarly we generate the set of conditions from two classifier events.

```
      8<        88b, 1<, 8c, 4<<
      8<        88b, 1<, 8c, 5<<
      8<        88b, 2<, 8c, 4<<
      8<        88b, 2<, 8c, 5<<
   88b, 1<<        88b, 2<<
   88b, 1<<        88c, 4<<
   88b, 1<<        88c, 5<<
   88b, 2<<        88c, 4<<
   88b, 2<<        88c, 5<<
   88c, 4<<        88b, 1<<
   88c, 4<<        88b, 2<<
   88c, 4<<        88c, 5<<
   88c, 5<<        88b, 1<<
   88c, 5<<        88b, 2<<
88b, 1<, 8b, 2<<       8<
88b, 1<, 8c, 4<<       8<
88b, 1<, 8c, 5<<       8<
88b, 2<, 8c, 4<<       8<
88b, 2<, 8c, 5<<       8<
88c, 4<, 8c, 5<<       8<
```

Figure 7 Conditions from two classifier events

Any classifier can occur only once. There are no examples of an object detector rejecting a classifier more than once, but obviously there are examples of an object detector accepting several classifiers.

For each pairing of these conditions with a classifier a conditional probability needs to be determined (or in the simulator randomly assigned).

Similarly the conditions generated from three classifier events

```
      88b, 1<<        88b, 2<, 8c, 4<<
      88b, 1<<        88b, 2<, 8c, 5<<
      88c, 4<<        88b, 1<, 8c, 5<<
      88c, 4<<        88b, 2<, 8c, 5<<
   88b, 1<, 8b, 2<<       88c, 4<<
   88b, 1<, 8b, 2<<       88c, 5<<
   88b, 1<, 8c, 4<<       88b, 2<<
   88b, 1<, 8c, 4<<       88c, 5<<
   88b, 1<, 8c, 5<<       88b, 2<<
   88b, 2<, 8c, 4<<       88c, 5<<
   88c, 4<, 8c, 5<<       88b, 1<<
   88c, 4<, 8c, 5<<       88b, 2<<
88b, 1<, 8b, 2<, 8c, 4<<      8<
88b, 1<, 8b, 2<, 8c, 5<<      8<
88b, 1<, 8c, 4<, 8c, 5<<      8<
88b, 2<, 8c, 4<, 8c, 5<<      8<
```

Figure 8 - Conditions from three classifier events

Appendices

Appendix 1 – The expected cost of a cascade

The cost of computing a single weak classifier from the cascade of weak classifiers is given as C_i^s for the i^{th} element of the sequence of weak classifiers (s). For the Viola-Jones object detector this is does not vary with the region or patch.

An expression for the cost of classifier computation on a single region (r)

$$\text{cost}(s, r) = \text{cost}(s, 0, r)$$

where we define the cost recursively

$$\begin{aligned}
&\text{cost}(s, n, r) = \\
&\text{if(n >= length(s)) then } 0 \\
&\text{else if (reject(s, n, r)) then } C_n^s \\
&\text{else } C_n^s + \text{cost}(s, n+1, r)
\end{aligned}$$

We can determine a simple expression for the expected cost in terms of the cost of evaluating a weak classifier and a predicate (P) that is a product of conditional probabilities (Q).

$$\text{Exp}[\text{cost}(s, r)] = C_0^s + \sum_{i=1..length(s)-1} C_i^s P(s, i, r)$$

where

$$P(s, n, r) = \prod_{i=0..n-1} Q(s, i, r)$$

$$\begin{aligned}
Q(s,0,r) &= \Pr[accept(s,0,r)] \\
Q(s,1,r) &= \Pr[accept(s,1,r) \mid accept(s,0,r)] \\
Q(s,2,r) &= \Pr[accept(s,2,r) \mid accept(s,0,r) \wedge accept(s,1,r)] \\
Q(s,3,r) &= \Pr[accept(s,3,r) \mid accept(s,0,r) \wedge accept(s,1,r) \wedge accept(s,2,r)] \\
&\cdots \\
Q(s,n,r) &= \Pr[accept(s,n,r) \mid \bigwedge_{i=0..n-1} accept(s,i,r)]
\end{aligned}$$

With the exception of the first predicate Q is the conditional probability that a given patch is accepted by the nth classifier given that all previous classifiers accepted the patch.

A similar expression for the expected cost is given in [6].

Some observations:

1. It is better to choose an initial classifier in the cascade that has lower cost, but it is also important that a classifier rejects as many patches as soon as possible so that later stages are not evaluated.

2. Reordering the sequence of classifiers in the cascade will change the expected cost of the
3. The contribution to the overall cost made by the later stages of the cascade is insignificant. This is because the weight given to each cost is a product of probabilities, each of which is less than one and so later overall cost contributions converge to zero.
4. Making optimum choices for the initial classifiers of the cascade will achieve most of the benefits.
5. It is difficult to predict the probability of accept later stages accepting/rejecting a patch because the space of patches is greatly pruned by earlier classifiers. A simple model would replace the later probabilities with a uniform random choice (0.5).
6. The conditions used as prior knowledge is the fact that the patch has been accepted by earlier parts of the cascade. The "accept" decision made by a weak classifier in the cascade is a binary decision taken using a threshold. Other approaches use a weight to indicate the importance of the classifier and some normalised scalar value that was used in the threshold. Similar prior knowledge could be exploited here.
7. However if we consider the evaluation of a single cascade in the context of a set of other object detectors then there is a richer set of prior knowledge that can be optimised. This extra knowledge would be results from the classifiers that been evaluated by the other object detectors. This would give both a larger set of classifiers that had accepted the patch as well a set of classifiers that had rejected the patch.
8. The expression for the expected cost of the cascade can be adapted (by simple conjunction of the extra conditions) to give this extra prior knowledge from the other object detectors. This would give us a means of adapting a cascade to particular prior knowledge from the other object detectors, but would not allow us to optimise the whole system of object detectors (see next appendix) For this we need to derive an N-object decision tree from the input cascades.

Appendix 2 – The expected cost of an N-object decision tree

We examine the expected cost of a set of object detectors each built using a cascade of weak classifiers

We define the data structure for N-object decision tree recursively

$$\mathrm{NDT} = \mathrm{empty}() \mid \mathrm{makeNDT}(\mathrm{OBJECT\_ID} \times \mathrm{CLASSIFIER}, \mathrm{NDT}, \mathrm{NDT})$$

It is either empty or contains a classifier labelled with its object identifier, and two other N-object decision trees. The first decision tree is evaluated when the classifier accepts a patch, and the second decision tree is evaluated when the classifier rejects a patch.

When an N-object decision tree is derived from the cascades of the input object detectors it will possess a number of important properties making it different from an arbitrary decision tree.

- When the decision tree is restricted to a particular object detector the result is a set of cascades. These cascades will be re-orderings of the original input cascade for the object detector.
- At every leaf of the decision tree – the results of all the object detectors will have been obtained, and these results will be the same as those obtained by running each object detector independently.
- The only classifiers that are run are the classifiers from the input object detectors.

In the Mathematica notebook attached to this invention disclosure there is a definition of the restrict operator, and some examples of its use on an n-object decision tree.

The cost of evaluating a particular patch and decision tree is defined recursively by:

$$\text{cost}(\text{empty}(), patch) = 0$$

$$\text{cost}(\text{makeNDT}((id, classifier), accept, reject), patch) =$$
$$\text{Classifier Cost}(classifier, patch) +$$
(if (accept*(classifier,patch)*)
then
cost(*accept,patch*)
else
cost(*reject,patch*)
)

The expected cost of evaluating a patch can be derived as

$$\text{Exp}[\text{cost}(dt, patch)] = \text{ExpCostNDT}(dt, \{\}, \{\})$$

Where we define the expected cost recursively

$$\text{ExpCostNDT}(\text{empty}(), as, rs) = 0$$

$$\text{ExpCostNDT}(\text{makeNDT}((id, classifier), accept, reject), as, rs) =$$
$$\text{ExpClassifierCost}(classifier) +$$
$$(\text{let}(p = \Pr[\text{accept}(classifier, patch) \mid \text{makeCondition}(as, rs, patch)])\text{in}$$
$$p\text{ExpCostNDT}(accept, \text{Append}(as, (id, classifier), rs) +$$
$$(1 - p)\text{ExpCostNDT}(reject, as, \text{Append}(rs, (id, classifier)))$$

The condition for the probability of accepting a patch is formed from the conjunction of the classifiers that accept and reject the patch

$$\text{makeCondition}(as, rs, patch) = \text{AcceptCondition}(as, patch) \wedge \text{RejectCondition}(rs, patch)$$

where

$$\text{AcceptCondition}(\{\}, patch) = \text{true}$$

$$\text{AcceptCondition}(\text{Append}(as, (id, classifier)), patch) =$$
$$\text{accept}(classifier, patch) \wedge \text{AcceptCondition}(as, patch)$$

and

$$\text{RejectCondition}(\{\}, patch) = \text{true}$$

$$\text{RejectCondition}(\text{Append}(rs, (id, classifier)), patch) = \text{reject}(classifier, patch) \wedge \text{RejectCondition}(rs, patch)$$

References

1. "Robust real-time face detection", Paul Viola, Michael J. Jones, International Journal of Computer Vision 57(2), 137-154, 2004, Kluwer Academic Publishers. http://w3.hpl.hp.com/people/dag/Face_detection/papers/ICCV01-Viola-Jones.pdf
2. "Speeding up for Viola-Jones face detector", Darryl Greig, Huitao Luo, Casey Miller, http://lib.hpl.hp.com/techpubs/2005/HPL-2005-69.pdf
3. "Darryl Greig's active face detection bibliography", http://w3.hpl.hp.com/people/dag/Face_detection/papers/DarrylsActive Bibliography.doc
4. "Robust multi-pose face detection in images", Rong Xia0, Ming-Jing Li, Hong-Jiang Zhang, IEEE transactions on circuits and systems for video technology, http://w3.hpl.hp.com/people/dag/Face_detection/papers/robust_multipose_face_detection.pdf
5. "Face detection with a boosting cascade" An online tutorial, Jason Strutz, http://www.navyrain.net/facedetection.html
6. "Optimising cascade classifiers" , Brendan McCane, Kevin Novins, Michael Albert,, Submitted to Journal of Machine Learning Research, 2005. http://www.cs.otago.ac.nz/staffpriv/mccane/publications/mccane_jmlr_submission.pdf
7. **"**Robust Object Detection via Soft Cascade**"** Lubomir Bourdev, Jonathan Brandt, CVPR 2005 http://www.lubomir.org/Academic/SoftCascade.pdf
8. "Joint induction of shape features and tree classifiers" Y. Amit, D. Geman, and K.Wilder, IEEE Trans. PAMI, 19:1300--1306, 1997