Sharing Visual Features for Multiclass and Multiview Object Detection

A. Torralba, K. Murphy and W. Freeman
IEEE TPAMI 2007

presented by J. Silva, Duke University
Detecting multiple classes/views of objects in clutter

Note: All figures adapted from the original paper.
Detecting multiple classes/views of objects in clutter

- Traditional approaches apply a battery of classifiers at multiple locations and scales
  - This is slow, requires computation of many features and thus large amounts of training data
  - *Runtime* computational complexity and *training time* sample complexity scale linearly with the number of classes

- This paper presents a multitask learning procedure based on boosted decision stumps
  - Common features are shared across classes
  - Detectors for each are trained jointly and share training examples
  - Computational and sample complexity scales *sublinearly* with the number of classes
  - Features are generic, edge–like rather than object–specific
Outline

- Review of boosting
- Multiclass boosting with feature sharing
- Results
  - Multiclass object detection
  - Multiview object detection
  - Feature sharing applied to face detection and recognition
- Conclusion
**Binary boosting**

- \( \mathbf{v} \) is a feature vector and \( z \in \{-1, 1\} \) is a class label
- Boosting sequentially fits additive models of the form
  \[
  H(\mathbf{v}) = \sum_{m=1}^{M} h_m(\mathbf{v})
  \]
- \( M \) is the number of boosting rounds
- \( H(\mathbf{v}) = \log \frac{P(z=1|\mathbf{v})}{P(z=-1|\mathbf{v})} \)
- Hence, \( P(z = 1|\mathbf{v}) = \sigma(H(\mathbf{v})) \) with \( \sigma(x) = 1/(1 + e^{-x}) \)
- \( h_m(\mathbf{v}) \) are weak learners and \( H(\mathbf{v}) \) is a strong learner
Binary boosting

- Boosting optimizes the following cost function one term at a time

\[
J = E \left[ e^{-zH(v)} \right]
\]

- \( J \) can be optimized in many ways

- This paper chooses gentleBoost (weighted squared error)

\[
\arg \min_{h_m} J(H + h_m) \approx \arg \min_{h_m} E \left[ e^{-zH(v)}(z - h_m)^2 \right]
\]

\[
J_{\text{WSE}} = \sum_{i=1}^{N} w_i (z_i - h_m(v_i))^2
\]

- \( N \) is the number of training examples

- Details of the optimization depend on \( h_m \)
Binary boosting: Example

- Regression stumps $h_m(v) = a\delta(v^f > \theta) + b\delta(v^f \leq \theta)$
  - $\delta(\cdot)$ is the indicator function
  - $v^f$ is the $f$–th component of vector $v$

- Optimize over parameters $a$, $b$, $\theta$ and weights $w_i$
  - Search over all $f$ and, for each one, try all $\theta$ induced by sorting the $N$ examples
  - Compute

\[
a = \frac{\sum_i w_i z_i \delta(v^f_i > \theta)}{\sum_i w_i \delta(v^f_i > \theta)} \quad b = \frac{\sum_i w_i z_i \delta(v^f_i \leq \theta)}{\sum_i w_i \delta(v^f_i \leq \theta)}
\]

\[w_i := w_i e^{-z_i h_m(v_i)}\]
The paper proposes the JointBoost algorithm

Modify the cost function as in Adaboost.MH

\[ J = \sum_{c} E \left[ e^{-z^c H(v, c)} \right] \]

- \( z^c \in \{-1, 1\} \) is the membership label for class \( c \)
- \( H(v, c) = \sum_{m=1}^{M} h_m(v, c) \) and \( H(v, c) = \log \frac{P(z^c=1|v)}{P(z^c=-1|v)} \)

This paper is different from Adaboost.MH in the structure of the \( h_m \)

Key idea

- Choose best subset of classes \( S(n) \) that will share a feature \( f \)
- Use greedy \( O(C^2) \) approach to avoid \( O(2^C) \) exhaustive search
Multiclass boosting with feature sharing

- Overlapping subsets of classes are considered because objects may share features in a way that cannot be represented as a tree
Multiclass boosting with feature sharing

- The strong learner associated with subset $S(n)$ is

$$G^{S(n)}(v) = \sum_{m=1}^{M_n} h^m_n(v)$$

- $M_n$ is the number of stumps for subset $S(n)$

- For each class $c$, find all subsets $S(n)$ that contain $c$ and sum their respective strong learners
Multiclass boosting with feature sharing (toy problem)

- Three classes + background, 8 rounds of boosting
- Features are projections onto 60 lines at different angles, coming from the origin (in 2D)
- Feature sharing (top) vs. independent features (bottom)
Possible ways to share features

Fig. 4. (a) All possible ways to share features among three classifiers. The sets are shown in a lattice ordered by subset inclusion. The leaves correspond to single classes. (b) Decision boundaries learned by all the nodes in the sharing graph for the problem in Fig. 3.

\[ H(v, 1) = G^{1,2,3}(v) + G^{1,2}(v) + G^{1,3}(v) + G^1(v), \]
\[ H(v, 2) = G^{1,2,3}(v) + G^{1,2}(v) + G^{2,3}(v) + G^2(v), \]
\[ H(v, 3) = G^{1,2,3}(v) + G^{1,3}(v) + G^{2,3}(v) + G^3(v), \]

\( M_{123} = 3, \ M_{12} = 2, \ M_{23} = 1, \ M_{13} = 0, \ M_1 = 1, \ M_2 = 0, \) and \( M_3 = 1, \) so there are no pure boundaries for class 2 in this example (indicated by the blank \( G^2 \) square in Fig. 4b).
JointBoost algorithm (simple version)

1) Initialize the weights \( w_i^c = 1 \) and set \( H(v_i, c) = 0, i = 1..N, c = 1..C \).
2) Repeat for \( m = 1, 2, \ldots, M \)
   a) Repeat for \( n = 1, 2, \ldots, 2^C - 1 \)
      i) Fit shared stump:
         \[
         h_m^n(v_i, c) = \begin{cases} 
         a_S & \text{if } v_i^f > \theta \text{ and } c \in S(n) \\
         b_S & \text{if } v_i^f \leq \theta \text{ and } c \in S(n) \\
         k^c & \text{if } c \notin S(n) 
         \end{cases}
         \]
      ii) Evaluate error
         \[
         J_{wse}(n) = \sum_{c=1}^{C} \sum_{i=1}^{N} w_i^c (z_i^c - h_m^n(v_i, c))^2
         \]
   b) Find best subset: \( n^* = \arg\min_n J_{wse}(n) \).
   c) Update the class estimates
      \[
      H(v_i, c) := H(v_i, c) + h_m^{n^*}(v_i, c)
      \]
   d) Update the weights
      \[
      w_i^c := w_i^c e^{-z_i^c h_m^{n^*}(v_i,c)}
      \]
JointBoost algorithm (simple version)

- Fitting the stumps

\[
as_S(f, \theta) = \frac{\sum_{c \in S(n)} \sum_i w_i^c z_i^c \delta(v_i^f > \theta)}{\sum_{c \in S(n)} \sum_i w_i^c \delta(v_i^f > \theta)}
\]

\[
b_S(f, \theta) = \frac{\sum_{c \in S(n)} \sum_i w_i^c z_i^c \delta(v_i^f \leq \theta)}{\sum_{c \in S(n)} \sum_i w_i^c \delta(v_i^f \leq \theta)}
\]

\[
k^c = \frac{\sum_i w_i^c z_i^c}{\sum_i w_i^c}
\]

- Need to try all features \( f \) and all \( N \) values of \( \theta \)
JointBoost algorithm (efficient version)

- Propagate the stumps from the leaves (single classes) via weighted combinations

\[ a_S(f, \theta) = \frac{\sum_{c \in S} a_c(f, \theta) w^c_+(f, \theta)}{\sum_{c \in S} w^c_+(f, \theta)} \]
\[ w^c_+(f, \theta) = \sum_{i} w^c_i \delta(v^f_i > \theta) \]

\[ b_S(f, \theta) = \frac{\sum_{c \in S} b_c(f, \theta) w^c_-(f, \theta)}{\sum_{c \in S} w^c_-(f, \theta)} \]
\[ w^c_-(f, \theta) = \sum_{i} w^c_i \delta(v^f_i \leq \theta) \]

- The weighted error is now

\[ J_{WSE}(n) = (1 - a_S^2) \sum_{c \in S(n)} w^c_+ + (1 - b_S^2) \sum_{c \in S(n)} w^c_- + \sum_{c \notin S(n)} \sum_{i=1}^{N} w^c_i (z^c_i - k^c)^2 \]
JointBoost algorithm (avoiding $2^C$ search via greedy forward selection)

- At each round, decide which classes will share features
- Start by computing all features for the leaves (single classes) as above
- Do
  - Select the class $c_1$ with best error reduction. This is (singleton) set 1.
  - Then, select the second class $c_2$ that most reduces the error *jointly* with $c_1$. The pair $\{c_1, c_2\}$ is set 2.
  - Iterate until all $C$ classes have been added
- Select the best set from 1, ... , $C$
- Complexity is $O(C^2)$
Fig. 6. (a) Comparison of number of stumps needed to achieve the same performance (area under ROC equal to 0.95) when using exact search, best-first, best pair, random sharing, and no sharing at each round. We use a toy data set with $C = 9$ classes plus a background class in $D = 2$ dimensions. (b) Complexity of the multiclass classifier as a function of the number of classes. The complexity of a classifier is evaluated here as the number of stumps needed for achieving a predefined level of performance (area under the ROC of 0.95).
Outline

- Review of boosting
- Multiclass boosting with feature sharing
- Results
  - Multiclass object detection
  - Multiview object detection
  - Feature sharing applied to face detection and recognition
- Conclusion
Results for multiclass object detection (LabelMe dataset)

Fig. 7. (a) Each feature is composed of a template (image patch on the left) and a binary spatial mask (on the right) indicating the region in which the response will be averaged. The patches vary in size from $4 \times 4$ pixels to $14 \times 14$. (b) Each feature is computed by applying normalize correlation with the template. From each image, we get positive ($\sigma^c = 1$) and negative (background, $\sigma^c = -1$ $\forall c$) training samples by sampling the set of responses from all the features in the dictionary at various points in the background and in the center of each target object.
Fig. 8. Examples of typical detections for computer screen, mouse, do-not-enter sign, mug, and chairs (results are the first five images processed from a typical run). For each row, only the output of one object class detector is shown. The results are obtained training 21 object classes using 50 training samples per class and 1,000 background samples. The classifier uses 500 features (rounds of boosting). Images are cropped so that the difficulty of detecting all the object classes is the same independent of their real size. Images have about $180 \times 180$ pixels. Detections are performed by scanning the image across locations and scales. Scale is explored by scaling the image with steps of 0.9.
Results for multiclass object detection (LabelMe dataset)

Fig. 9. ROC curves for 21 objects (red (lower curve) = isolated detectors, blue (top curve) = joint detectors). ROC is obtained by running the detector on entire images and sampling the detector output at the location of the target and on the background. For each graph, the horizontal axis is the false alarm ratio and the vertical axis is the ratio of correct detections. For each object, we show the ROC obtained with different training parameters. From left to right: 1) 70 features in total (on average $70/21 \approx 3.3$ features per object) and 20 training samples per object, 2) 15 features and 20 training samples, and 3) 15 features and two training samples. In the second and third cases, there are fewer features than classes, so training each class separately will inevitably result in some classifiers performing at chance (shown by diagonal ROC lines).
Fig. 10. (a) Evolution of classification performance of the test set as a function of number of boosting rounds (or features). Performance is measured as the average area below the ROC across all classes. Chance level is 0.5 and perfect detection for all objects corresponds to area = 1. Both joint and independent detectors are trained using up to 70 features (boosting rounds), 20 training samples per object, and 21 object classes. The dashed lines indicate the number of features needed when using joint or independent training for the same performance. (b) This graph shows how many objects share the same feature at each round of boosting during training. Note that a feature shared among 10 objects is, in fact, using $20 \times 10 = 200$ training samples.
Fig. 12. Matrix that relates features to classifiers, which shows which features are shared among the different object classes. The features are sorted from left to right from more generic (shared across many objects) to more specific. Each feature is defined by one filter, one spatial mask, and the parameters of the regression stump (not shown). These features were chosen from a pool of 2,000 features in the first 40 rounds of boosting.
Fig. 13. Clustering of objects according to the number of shared features. Objects that are close in the tree are objects that share more features and, therefore, share most of the computations when running the classifiers on images. This clustering is obtained by jointly training 21 objects, using 70 stumps, and 50 training samples per object.
Fig. 14. Specific versus generic features for object detection. (a) An object with very little intraclass variation. (b) Selected features by a single detector. When training an independent detector, the system learns template-like filters. (c) When trained jointly with 20 other classes, the system learns more generic, wavelet-like filters.

Fig. 15. Comparison of the efficiency of class-specific and shared features to represent many object classes (in this experiment, we used 29 object classes by adding to 21 previous classes frontal faces, parking meter, pot, paper cup, bookshelf, desk, laptop, and fire hydrant). (a) Total number of features needed to reach a given classification performance for all the objects (area under the ROC equal to 0.95). The results are averaged across 20 training sets and different combinations of objects. Error bars correspond to 80 percent interval. As we increase the number of objects to be represented, the number of features required to keep performance constant increases linearly for class-specific features and sublinearly for shared features. (b) Number of features allocated for each object class. When sharing features, the features become less informative for a single class, and we therefore need more features per class to achieve the same performance compared to using class-specific features.
Fig. 17. (a) Detection results on images from the PASCAL collection (cars test set 2 [8]). The classifier is trained on 12 views of cars from the LabelMe data set (50 positive examples for each view and 12,860 background samples) and uses 300 shared features. The detection results are organized according to the confidence of the detector (from high precision/low recall to low precision/high recall). The first row's images are randomly selected among the most confident detections. Each row represents a different point in the precision-recall curve. (b) Precision-recall curves comparing our algorithm with algorithms evaluated during the PASCAL challenge.
Results for multiview object detection (Pascal dataset)

Fig. 18. Detection performance as a function of number of training examples per class. (a) Twelve objects of different categories. (b) Twelve views of the same object class. Sharing features improves the generalization when few training samples are available, especially when the classes have many features in common (case b). The boosting procedure (both with class-specific and shared features) is run for as many rounds as necessary to achieve maximal performance on the test set.
Results for face detection and recognition (MacBrain Face Stimulus dataset)

Fig. 19. Example of the emotions used.
Results for face detection and recognition (MacBrain Face Stimulus dataset)

Fig. 20. Sharing matrix for face detection and emotion classification. This matrix shows the features selected using 30 rounds of boosting. The (face) generic features are used to distinguish faces from nonfaces (detection task), while the intraclass specific features perform both detection (distinguish faces from the background) and recognition (distinguish among face categories). Here, the degree of sharing is larger than the sharing obtained in the multiclass and multiview experiments.
Results for face detection and recognition (MacBrain Face Stimulus dataset)

Fig. 21. This figure evaluates the performances of the joint classifier by splitting both tasks, detection and recognition. (a) ROC for face detection and (b) confusion matrix for emotion classification with 30 shared features and 15 emotion categories. The numbers correspond to percentages.
The idea of sharing features between classes has also been exploited elsewhere

- Bernstein and Amit use EM applied to a mixture of Bernoulli products to discover features that are shared between classes
- LeCun et al. propose Convolutional Neural Networks
- A related idea is error correcting output codes (ECOC) by Dietterich and Bakiri