Hierarchical Object Indexing and Sequential Learning

Xiaodong Fan
Dept. of Electrical and Computer Engineering
The Johns Hopkins University
Baltimore, MD 21218
xdfan@cis.jhu.edu

Donald Geman
Dept. of Applied Mathematics and Statistics
The Johns Hopkins University
Baltimore, MD 21218
geman@jhu.edu

Abstract

This work is about scene interpretation in the sense of detecting and localizing instances from multiple object classes. We concentrate on object indexing: generate an over-complete interpretation – a list with extra detections but none missed. Pruning such an index to a final interpretation involves a global, often intensive, contextual analysis. We propose a tree-structured hierarchy as a framework for indexing; each node represents a subset of interpretations. This unifies object representation, scene parsing, and sequential learning (modifying the hierarchy as new samples, poses and classes are encountered). Then we specialize to learning - designing and refining a binary classifier at each node of the hierarchy dedicated to the corresponding subset of interpretations. The whole procedure is illustrated by experiments in reading license plates.

1. Introduction

Our objective is to detect and localize instances from multiple object classes in cluttered grey level images. The literature is dominated by special cases, for example detecting instances from one object class such as faces, cars, pedestrians and characters. For multiple object classes, a common strategy is to collect evidence from a series of two-class decisions, for instance via "one-vs-all" [6, 10], pairwise comparisons [7] and error correcting output code [2]; see also [4] and [5]. Such methods, which execute a fixed number of binary classifiers for each image region, are often computationally inefficient, especially in rejecting background regions, and there is no obvious way to accommodate sequential learning.

Here, indexing in the sense of non-contextual detection primes global interpretation. First compile a list – an index – of candidate instances (object identities and poses) under a zero false negative constraint, but at the expense of numerous false positives. Then exploit contextual information, e.g., a priori constraints among poses, to resolve ambiguities and extract a final interpretation from the index.

This paper focuses on a unified framework for indexing – one data structure, a labeled tree, which accommodates how objects are represented by a nested decomposition of all possible instances; how individual binary classifiers, dedicated to corresponding sub-interpretations, are induced from data; and how scene parsing with these classifiers is organized in an efficient coarse-to-fine manner in order to detect object instances. Moreover, this structure can be updated indefinitely in order to refine existing classifiers as well as incorporate new poses or classes.

We start with a formulation of scene interpretation and object indexing in §2. Then, in §3, we present an abstract description of the hierarchical exploration of a space of hypotheses; this provides a unified framework for representation, learning and search. Related work and open issues are summarized in section §4. Designing the individual classifiers, as well as parameter learning and updating, is described in §5. Finally, experiments on reading license plates and conclusions are given in sections §6 and §7, respectively.

2. Scene Interpretation

The goal of multiple-class object detection is to compile a list of object instances (c, θ), where $c \in \mathcal{C}$ indicates the *class* or *category* of the object and $\theta \in \Theta$ represents its *pose* or *presentation* in the scene, an inclusive "nuisance parameter" which might convey more than merely the position, scale, etc. (e.g., the font of a character). Interpreting a scene will typically involve more information, more structure, than a simple list can provide, such as occlusion patterns and other relationships among object instances; for instance, subsets of instances may themselves form "parts" of more general semantic entities. Moreover, even if the final output is a mere list – as in recognizing license plates – a priori information about global geometry, such as constraints among the poses, will certainly impinge upon the

process of reducing an over-complete representation to a final interpretation.

Let $\mathcal{Z} = \mathcal{C} \times \Theta$ be the set of all possible object instances and let $Y \subset \mathcal{Z}$ denote the true list of instances. Indexing means identifying a subset $\mathcal{D} = \mathcal{D}(I) \subset \mathcal{Z}$ where I is the image. The fundamental constraint is that Y belongs to $\mathcal{D}$. In our stochastic framework (see below), $\mathcal{D}$ is in fact a *random subset* and we require $P(Y \subset \mathcal{D}) = 1$. Pruning the index $\mathcal{D}$ simply means reducing it to Y.

Besides no missed detections, we want efficient indexing. This will be achieved by searching $\mathcal{Z}$ in a coarse-to-fine manner based on a tree hierarchy introduced in following section §3. In fact, as will be seen, the tree-structured framework is entirely general and could accommodate other scenarios and search problems.

3. The Tree Hierarchy

Consider searching among a large set of patterns or hypotheses $\mathcal{Z}$ in order to identify a distinguished subset. Suppose $\mathcal{Z}$ is far too large to check individually for every pattern, in other words to perform brute-force template-matching. Instead, organize the search coarse-to-fine based on a recursive partitioning of $\mathcal{Z}$; see Figure 1. Hence, starting from $\mathcal{Z}$ itself (the root node), there is one partition of $\mathcal{Z}$ for each level in a tree hierarchy and these levels are nested – the partition of level $l+1$ is a refinement of the one at level l. The subset or "cell" of $\mathcal{Z}$ at internal node t is denoted $\mathcal{Z}_t$. The final level may not correspond to the individual patterns but rather to some desired level of pattern resolution. (In the case of object instances, the cells at the leaves are pure in class but may contain geometric poses within some range.)

There is a binary classifier $X_t \in \{-1, 1\}$ at every node t which represents a hypothesis test for $H_t : Y \cap \mathcal{Z}_t \neq \emptyset$ against an appropriate alternative hypothesis H_t^{alt} contained in $Y \cap \mathcal{Z}_t = \emptyset$ (see §5). One important requirement is that X_t can be continually refined as new training data arrives. In our setup, each X_t is a likelihood ratio test, actually linear in certain features; the coefficients of the features are defined directly in terms of observed feature statistics under H_t and H_t^{alt}, thereby allowing for sequential learning.

Another requirement is that $P(X_t = 1|H_t) = 1$. Naturally, this can only be enforced by allowing $P(X_t = -1|H_t^{alt}) \gg 0$, especially for coarse cells. We then define

$$\mathcal{D} = \bigcup_{t \in \partial T} \{\mathcal{Z}_t | X_s = 1 \ \forall s \in \mathcal{A}_t\}$$

where $\mathcal{A}_t$ is the set of ancestors of t, including t itself. The elements of $\mathcal{D}$ are precisely the hypotheses in $z \in \mathcal{Z}$ which survive every test X_t which "covers" z in the sense that $z \in \mathcal{Z}_t$. Evidently $Y \subset \mathcal{D}$ with probability one.

There are many possible online, computational strategies for actually determining the set $\mathcal{D}$ from I. One of them is breadth-first coarse-to-fine search. Start with the root test; if it is negative stop because $\mathcal{D} = \emptyset$; if it is positive, execute the test at each child. In general, proceed recursively: execute the tests at the children of node t *if and only if* $X_s = 1$ for every $s \in \mathcal{A}_t$ (see Figure 1). For each surviving leaf node t whose test is positive, add $\mathcal{Z}_t$ to $\mathcal{D}$.

This strategy is especially efficient when the event $Y \cap \mathcal{Z}_t = \emptyset$ has high probability for all leaves t. This is certainly the case for object indexing since the explanation "background" statistically dominates any given hypothesis about certain objects at certain poses. Hence the search terminates quickly along most paths with high probability. But some paths survive due to false positive error – the price we pay for no missed detections.

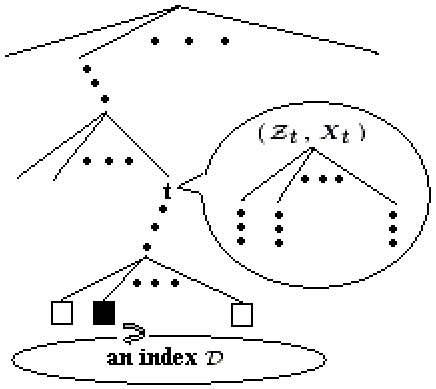

Figure 1. The tree hierarchy.

4. Related Work and Open Issues

Our online indexing strategy is coarse-to-fine in "scope" (referring to the size of $\mathcal{Z}_t$). Hierarchical search in class/pose space also appears in [4]. Another design, a "cascade," can be found in work on face detection [9, 11].

Inducing decision trees from data [1] is different. It corresponds to recursively partitioning the *feature space* rather than the hypothesis space; moreover, each search terminates at a single leaf. Most importantly, the procedure here is *designed* rather than learned.

The proposal outlined in §§2,3 is not complete. Still, an experiment illustrating a preliminary version of the entire system, including contextual disambiguation, will be given in §6. Some issues to be explored here and elsewhere include:

- **Automatic Partitioning:** Starting with $\mathcal{Z}$, how is the hierarchy $\{\mathcal{Z}_t\}$ of nested partitions defined? In [3], there is only one object class and the "pose decomposition" is done manually, as in [4]. This will be addressed elsewhere based on hierarchical clustering.
- **Contextual Disambiguation:** How is $\mathcal{D}$ pruned to a final interpretation? The experiments here are based

on utilizing a Viterbi algorithm and prior knowledge about the geometric layout.

- **Bootstrapping:** What is the right alternative hypothesis H_t^{alt}? In previous work only nonspecific alternatives were considered. Presumably more discriminating classifiers can be obtained by taking advantage of the order of testing in order to anticipate the likely confusions, as in [9, 11]. This is addressed in §5.
- **Sequential Updating:** How is the hierarchy updated to accommodate previously unseen poses or classes? How are the tests then updated? The second question is addressed in §5, where X_t is given by a *likelihood ratio test* whose coefficients are defined directly in terms of observed frequencies under the both hypotheses, and hence can be refined indefinitely.

5. Learning

Throughout this section we fix a node t of the hierarchy and concentrate on constructing and updating X_t. We also assume the image I is represented by a fixed family of *binary* features $\{x_j\}$, usually local operators, but unspecified for the moment; an example is given in §6. Much of what follows extends to scalar-valued features with probabilities replaced by means. Finally, there is a training set $\mathcal{L}_t$ for node t (see below), divided into positive examples, $\mathcal{L}_t^+$, and negative examples, $\mathcal{L}_t^-$.

Alternative Hypothesis: We formulate X_t as a likelihood ratio test of H_t vs. H_t^{alt}. As in [8], the alternative is not $Y \cap \mathcal{Z}_t = \emptyset$, but rather $H_t^{alt} = B_{\bar{t}} \cap (Y \cap \mathcal{Z}_t = \emptyset)$. Here, $\bar{t}$ is the parent node of t and $B_{\bar{t}} = \{X_s = 1 \; \forall s \in \mathcal{A}_{\bar{t}}\}$, the event that every classifier "above" t is positive. In effect, we are "boosting" X_t by training against those object instances we anticipate to be especially confusing for the current task.

The sets $\mathcal{L}_t^+$ and $\mathcal{L}_t^-$ represent training samples belonging to $\mathcal{Z}_t$ and $B_{\bar{t}} \cap \mathcal{Z}_t^c$, respectively. Notice we are *not* using clutter - samples from the background which resemble objects of interest. This would likely bring further improvements.

Likelihood Ratio Test: Of course we cannot estimate the (conditional) *joint* distribution $\{x_j\}$ (given H_t and H_t^{alt}), at least not for any reasonably large set of features. No training set would be large enough. Instead, we simply use a linear test relative to a distinguished subset $\mathcal{J}_t$ of features. The one we use *would* be the LRT (at null type I error) under a naive Bayes model, i.e., *were* the features $\mathbf{x} = \{x_j, j \in \mathcal{J}_t\}$ to be independent under the two hypotheses.

The test is then

$$X_t = sign\left(\sum_{j\in\mathcal{J}_t} \lambda_t(j)x_j - \tau_t\right) \tag{1}$$

where

$$\lambda_t(j) = \log \frac{p_t(j)(1-q_t(j))}{q_t(j)(1-p_t(j))} \tag{2}$$

and the frequencies $p_t(j) = P(x_j = 1|H_t)$ and $q_t(j) = P(x_j = 1|H_t^{alt})$ are estimated from $\mathcal{L}_t^+$ and $\mathcal{L}_t^-$ respectively. (We convert x_j to $1 - x_j$ for features with $p_t(j) < q_t(j)$, thus forcing $\lambda_t(j) > 0$.) Using equation (2) directly often over-estimates $\lambda_t(j)$ when $p_t(j)$ or $q_t(j)$ is very close to 0 or 1; hence we approximate (2) by $p_t(j) - q_t(j)$ instead. The coefficient $\lambda_t(j)$ reflects the discrimination power of feature x_j at node t in the hierarchy. The threshold τ_t (now necessarily positive) is chosen to enforce the null false negative constraint: $P(X_t = 1|H_t) = P(\sum_{j\in\mathcal{J}_t} \lambda_t(j)x_j > \tau_t|H_t) = 1$ where this probability is estimated from $\mathcal{L}_t^+$. Hence $\tau_t = \tau(\mathcal{J}_t)$.

The learning problem is then to induce $\mathcal{J}_t$ from $\mathcal{L}_t$. (Determining τ and $\{p_t(j), q_t(j)\}$ are relatively straightforward.) Of course we want discriminating features. More directly, we want to control the type II error $\delta(\mathcal{J}_t) = P(X_t = 1|H_t^{alt})$. By the Markov inequality:

$$\begin{aligned} \delta(\mathcal{J}_t) &= P(\textstyle\sum_{j\in\mathcal{J}_t} \lambda_t(j)x_j > \tau(\mathcal{J}_t)|H_t^{alt}) \\ &\leq \frac{\sum_{j\in\mathcal{J}_t} \lambda_t(j)q_t(j)}{\tau(\mathcal{J}_t)} \end{aligned}$$

We propose a greedy, recursive algorithm: add features x_j to $\mathcal{J}_t$ one at a time based on minimizing the bound for $\delta(\mathcal{J}_t)$ with $|\mathcal{J}_t|$ fixed. At each iteration we re-estimate τ_t and choose the feature which minimizes the ratio bound. The features are examined in order of individual discriminating power.

For sequential learning, we do not re-estimate $\mathcal{J}_t$ from scratch; for one thing, this would require keeping all the data. Instead, we update $\mathcal{J}_t$ by adding a few features selected in the same way as above but based on the refined probability estimates and new training samples. Moreover, *all* the coefficients $\lambda_t(j)$ are updated, which is computationally trivial.

6. Reading License Plates

The objective is to identify the characters on the license plate from the photograph of the rear of a car, as shown in Figure 2. We omit the process of detecting and extracting the plates from the photograph as this is not the bottleneck in applications. The plates displayed in Figure 2 demonstrate the challenges due to variations in stroke widths, variable illumination, background clutters and other effects.

Classes $\mathcal{C}$: There are 37 object classes – 26 letters, 10 digits and one special symbol, a vertical bar.

Poses Θ: The pose θ has three components: location $z(\theta)$, scale $\sigma(\theta)$ and tilt $\rho(\theta)$. Since the variation in tilt is usually

Figure 2. A typical photograph and samples of extracted plates.

relatively small we can assume $|\rho(\theta)| \leq 5^\circ$. Similarly, we assume: $1 \leq \sigma(\theta) \leq 1.4$ (relative to a reference size). (Bigger characters can easily be accommodated; see below.)

Hierarchy: The first levels of the hierarchy are purely pose-based. The top level partitions the position $z(\theta)$ in the plate lattice into non-overlapping 5×5 regions. (Were widely disparate scales encountered, the next level would partition σ and larger characters found by downsampling and searching again at the smallest range.) Since objects at widely different positions (or scales) have very little if anything in common, the tests X_t for the root and the first level nodes are virtual – always positive. Thus, each 5×5 region R is visited in order to determine if there is a character whose center lies in R. Therefore, the real recursive partitioning starts with $\mathcal{Z} = \mathcal{C} \times \{\theta \in \Theta : z(\theta) \in R\}$. Obviously the tests are translation-invariant, so any R can serve as a reference. We use an automated, clustering procedure to generate a tree hierarchy with 13 levels and 1332 leaf nodes (hence unbalanced).

Training samples: Each class prototype (template) is resized and rotated to create training samples with various poses. All nodes other than the (position-restricted) root have both positive and negative training sets.

Initial learning: We use the same binary features as in [3] based on oriented intensity edges. "Spreading" is used to elevate the probabilities $p_t(j)$ and generate pose-invariance. Each X_t is constructed from $\mathcal{L}_t$ as in §5. As might be expected, the *conditional* type-II error decreases exponentially with the depth in the hierarchy, starting at 0.38 for the coarsest nontrivial test (low discrimination is the price for essentially null type I error) and going down to less than $.001$ at the leaves. These values were estimated from test data, averaging $P(B_t|Y \cap \mathcal{Z}_t = \emptyset)$ over all nodes t at each level.

Indexing results: There are roughly 400 false alarms per plate (see Figure 3(a)). Simple post-processing by clustering detections with the same class at nearby locations (which explain the same image data) and filtering very low-scoring detections (measured by proximity to the thresholds) reduces the number to around 40 per plate (see Figure 3(b)). The final result (see Figure 3(c)) is based on disambiguation by contextual analysis and will be presented in forthcoming papers.

Sequential learning: We did a preliminary experiment in refining all the tests X_t by first inducing them from the full training set subsampled in location by a factor of 2, and then updating the feature sets and coefficients as in §5. The average false positive rate of indexing increased somewhat, but the increased offline efficiency is well worth the price.

(a) initial detections (b) after post-processing (c) final interpretation

Figure 3. Selected detections, represented by the white dots.

7. Conclusion

We have filled in several components of a new framework for visual indexing which unifies representing objects, learning classifiers and scene parsing. In principle, this framework – a tree-structured hierarchy – can accommodate a great many classes and still maintain efficiency, both offline (by sequential learning) and online (by coarse-to-fine search). Upcoming papers will describe how the hierarchy is automatically generated from training samples and sequentially updated as new poses and classes are presented.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and regression trees*, Chapman & Hall, New York, 1993.

[2] T. G., Dietterich, and G. Bakiri, "Solving multiclass learning via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol 2, pp. 263-286, 1995.

[3] F. Fleuret and D. Geman, "Coarse-to-fine face detection," *Inter. Journal of Computer Vision*, vol. 41, pp. 85-107, 2001.

[4] D.M. Gavrila, "Multi-feature hierarchical template matching using distance transforms," *Proc. ICPR'98*, 1998.

[5] B. Leibe and B. Schiele, "Analyzing appearance and contour based methods for object categorization," *Proc. CVPR'03*, pp. II: 409-423, 2003.

[6] S. Mahamud and M. Hebert, "The optimal distance measure for object detection," *Proc. CVPR'03*, pp. I: 248-255, 2003.

[7] J.Platt, N. Cristianini and J. Shawe-Taylor, "Large margin DAGS for multiclass classification," *Advances in Neural Information Processing Systems, 12 ed*, MIT Press, 2000.

[8] H. Sahbi, "Support vector machines for hierarchical face detection," Ph.D. Thesis, University of Versailles, Dept. of Computer Science, 2003.

[9] D. Socolinsky, J. Neuheisel, C. Priebe, J. De Vinney and D. Marchette, "Fast face detection with a boosted cccd classifier," Technical report, Johns Hopkins University, 2002.

[10] X. Song, Y. Abu-Mostafa, J. Sill and H. Kasdan, "Image recognition in context: application to microscopic urinalysis", *Proc. NIPS'99*, 1999.

[11] P. Viola and M. J. Jones, "Robust real-time face detection," *Proc. ICCV'01*, pp. II: 747, 2001.