Learning complex object-class models in natural conditions

by

Dan Levi

A dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy in
Computer Science and Applied Mathematics

Faculty of Mathematics and Computer Science
The Weizmann Institute for Science

2009
Summary

Current approaches to object recognition use object-class models acquired during training for identifying and interpreting objects in novel images. Many natural object classes are highly variable in appearance and only class models with adequate complexity can fully capture this variability. In order to reliably learn such a complex model without over-fitting it to the training set, the training set should be sufficiently large. In most current approaches, the object-class model is learned from a fixed set of labeled training examples, which are fully available before training starts. As a result, current models are trained from a limited set of examples and are therefore bounded in their complexity and in their classification performance. This thesis introduces new approaches for learning complex object class models under less constrained and more natural conditions. The natural conditions for learning include two major aspects: online-incremental learning and unsupervised learning. In Online-incremental learning, the model is incrementally improved as examples are continuously provided to the learning system in an online manner. In unsupervised learning, the training examples are not labeled. Under such natural conditions it is possible to train from larger unconstrained datasets and as a result reliably learn more complex class models. The complexity of the model is expressed by its number of learned parameters, which in our model can be expressed by the number of learned sub-elements. In the object-class model we introduce, each object part is represented by a part-specific learned component of the model thus increasing the learned model complexity. Learning under such natural conditions poses a conceptual and computational challenge which requires introducing novel learning approaches. In this work we develop methods which can effectively learn complex object classes in an on-line manner and methods which can learn such classes without supervision. We test our learned classifiers experimentally on a wide range of object classes and various benchmark datasets and compare them to current state-of-the-art classifiers. The experimental results demonstrate that our new approach leads to improved classifier performance using less supervision.
Contents

1 Introduction 5

2 Learning to classify by ongoing feature selection 11
   2.1 Previous approaches ............................................. 14
   2.2 The on line feature selection algorithm ......................... 15
      2.2.1 Algorithm overview ......................................... 15
      2.2.2 Measuring feature merit value .............................. 16
      2.2.3 Algorithm description ...................................... 18
      2.2.4 Complexity analysis ........................................ 20
      2.2.5 Key properties .............................................. 20
   2.3 Experimental Results ........................................... 21
      2.3.1 Training and testing the classifier .......................... 21
      2.3.2 Classification results and analysis ......................... 22
      2.3.3 Comparison of online and comprehensive feature se-
          lection methods ............................................. 25
   2.4 Conclusions ..................................................... 30

3 Learning model complexity in an online environment 32
   3.1 Algorithmic Framework .......................................... 36
   3.2 Kernel Support Vector Machine Implementation ................ 40
   3.3 Experimental Results ........................................... 41
   3.4 Conclusions ..................................................... 44

4 Learning to detect object parts with multiple appearances 45
   4.1 The CNOR model: part detection using multiple-appearances 46
   4.2 Learned part models and object classification .................. 48
   4.3 Fine-Alignment of patches of corresponding object parts ... 49
   4.4 Implementation within supervised bag-of-words classification 50
   4.5 Conclusions ..................................................... 52
5 Unsupervised classification and part localization by consistency amplification 59
5.1 A new approach to the unsupervised class learning task . . . 60
5.2 The consistency amplification method . . . . . . . . . . . . . . 64
  5.2.1 The geometry phase . . . . . . . . . . . . . . . . . . . . 65
  5.2.2 The appearance phase . . . . . . . . . . . . . . . . . . . 68
  5.2.3 Applying the learned model to classify and localize
    objects and parts . . . . . . . . . . . . . . . . . . . . . . . 70
5.3 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
5.4 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
5.5 A Combined model for detecting, localizing, interpreting and
    recognizing faces . . . . . . . . . . . . . . . . . . . . . . . . . 76
  5.5.1 UCA-based face detection and recognition . . . . . . . . . 78
  5.5.2 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
  5.5.3 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83

6 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87

7 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 90
  7.1 Learning object part detectors with multiple appearance and
      multiple appearance similarity measures . . . . . . . . . . . . 91
  7.2 Minimal detectable parts . . . . . . . . . . . . . . . . . . . . . . 91
  7.3 Unsupervised learning of a visual categories with no restric-
      tions on the dataset . . . . . . . . . . . . . . . . . . . . . . . . 92
  7.4 Learning a model of simple classifier and exceptions with in-
      creasing complexity . . . . . . . . . . . . . . . . . . . . . . . . 93
  7.5 Unsupervised, bottom-up learning of hierarchical models for
      classification . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
  7.6 Incremental unsupervised learning of object classifiers in an
      online setting . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94

Appendix I: CNOR EM update rules . . . . . . . . . . . . . . . . . . . . 95
Chapter 1

Introduction
Although object recognition and classification schemes have made significant advances over the recent past, the performance is still far from comparable to human vision. Current classification methods have reached an impressive level of performance, and can be used to detect a large variety of object categories in complex scenes. At the same time, the performance level of current schemes is still significantly below human-level recognition, and the performance gap seems difficult to close. The performance gap also appears to widen when existing schemes are applied to multi-class settings, involving a large number of classes. The current performance of state-of-the-art natural-object classifiers (e.g. horses vs. background) is an equal error rate of 1-3% [40] under controlled settings in which viewpoint and object size are roughly constant. An average precision of 30%-80% [46] is obtained in more general settings in which the object size and its viewpoint significantly vary. In comparison, human performance on such tasks is usually perfect. In this thesis, I argue that the 'performance gap' will not be closed merely by incremental improvements and tuning of current models, but it will require significant conceptual changes and additions to existing methods. The goal of the thesis is to develop new methods for dealing effectively with complex object categories, primarily by the unsupervised, online incremental learning of more complex object models with learned sub elements.

A current dominant approach to object recognition relies on two major aspects: local appearance estimation, in which salient features are detected in the image and geometric configuration of local appearances, in which the object is detected using global configurations of the features. The local appearance estimation requires a method for comparing image features, while the geometric configuration requires an extensively trained object model. In order to adequately represent all the relevant variability within a visual class the class model will have to be substantially more detailed and complex than models used in current classification and recognition schemes in both these aspects. In a more complex model more fixed elements of the model become flexible and learned from examples. A major obstacle to learning complex models is the requirement for large datasets. It is known from statistical learning theory that there is a close connection between the complexity of a model class and the number of training samples needed to learn it. Complex models will therefore require efficient methods for extracting and then using information from a large number of training examples. The reported state-of-the-art results are often obtained with datasets containing less than 1,000 positive class examples. In contrast, handwritten character recognition systems [45] successfully approach human performance by using much larger training sets. In addition, new approaches for multi-class
object recognition report improved performance using millions of examples [66]. Human performance in classification also relies on extensive training. Even though a new category can sometimes be learned from a few examples only, the entire visual experience including objects with a similar appearance can be used in forming the new representation [3]. In order to learn more complex models from substantially larger sets of examples new learning approaches are required. Incremental learning is efficient in the online learning setting in which examples are continued to be presented and the training set is therefore unbounded in principle. Such learning from large datasets becomes practical by unsupervised learning or partially-supervised learning in which the need for hand labeling the examples is relaxed. An online and unsupervised approach is therefore more natural to use since it is less constrained, just as human learning is, compared with the traditional approach. This thesis introduces new approaches for online-incremental learning and for unsupervised learning of object classes. In addition we introduce new learning methods for previously fixed elements of the classification model, thus raising the model’s complexity. In particular, in our model each local appearance is replaced by a learned object part classifier. Notice that during the online learning process at each time point only a fixed training set has already been given. Although our final goal is a complex model for capturing the large class variability, at each point the learned model should be as simple as possible to avoid over-fitting. The difference between our approach and the current one is summarized in figure 1.1.

The final outcome of this new proposed approach will be eventually a complex classifier learned from a large number of unsupervised examples reaching human performance. We have made important and significant contributions within this new approach which are summarized in the remaining of this chapter.

Most existing classification methods can be described within the probabilistic framework as fitting a model to training data which computes $Prob_{\theta}(F,Y,C)$ where $F$ represents the observed visual features in each image, $C$ the target class, $Y$ the hidden variables of the model and $\theta$ the model parameters. In the test stage the model is applied to a new image produce its belief for $C$ by plugging in the features $F$ measured on the given image. The model for computing $Prob_{\theta}$ may range from a simple bag-of-features model [63] to complex probabilistic/geometric models [67]. The first contribution of our work was in the continuous online learning of $F$ and $\theta$. If the model is acquired incrementally over time, it becomes necessary to adapt the model complexity to the amount of available data. It then becomes natural to distinguish between the model parameters defining its complexity, $\alpha$, and the
Figure 1.1: Proposed approach for learning object classifiers

model parameters defining a specific model within a fixed complexity class $\theta$. The probability can then be written as: $\text{Prob}_{\theta}(F, Y, C)$. We contributed in three aspects. First, we proposed a new method for online learning of the visual features $F$. Second, we introduced a new framework allowing for an online learning of the model complexity $\alpha$. Third, we introduced a new method for learning complex multi-appearance part detectors $D$ instead of simple features $F$ and used these improved features for learning the model $\text{Prob}_{\theta}(F, Y, C)$ in an unsupervised manner.

In our model for ongoing feature selection (Chapter 2) the best features for classification $F$ are collected from image examples presented on the fly. The problem of selecting optimal features in an online learning environment is highly non-trivial. The online learning algorithm is only allowed to save a limited number of seen examples, and to perform a constant time operation each time an example is presented, in order for the system to be efficient and scalable. The online method we developed has the ability to adapt effectively to drifting distributions. In fixed environments, using limited computational resources, the online method continuously improves, and with
sufficient training will outperform off-line selection methods. The method can also be employed for efficiently exploring sets of features that are much larger than the ones used today by classification systems.

Since in online learning the training set expands over time, it is natural to allow the learned model to become more complex during the course of learning instead of confining the model to a fixed family of a bounded complexity. In Chapter 3 we present the new concept of adapting the model complexity during online learning. We developed a new algorithmic framework which enables online learning algorithms to adjust the complexity of the learned model to the amount of the training data as more examples become available. We show that in various settings the optimal model is more complex for larger training datasets. Using an implementation of our algorithmic framework with an online Kernel Support Vector Machine (K-SVM) we show that our algorithm discovers the optimal complexity of the model at each time step during the online learning process, and eventually outperforms a similar model working with a fixed model complexity.

Our contribution to classification model complexity is in introducing an algorithm for learning object part detectors described in Chapter 4. Our algorithm learns efficiently the representation of complex object part models with multiple representative appearances. A problem with current methods for detecting object parts is in their use of a fixed object part representation. In more accurate model of a higher complexity level each object part is detected by a learned part specific classifier. We introduce a new object part classifier which combines a set of representative part appearances in order to detect the object part in its full appearance range. Both representatives and combination parameters are selected in a single optimization using Structural EM [20] such that correct part detection is optimized. This method was subsequently used to learn an object classifier in an unsupervised manner as described next.

A major problem with supervised learning from large datasets is the difficulty in human labeling all the examples. In collaboration with Michael Dinerstenin and Leonid Karlinsky [29] we developed an algorithm for unsupervised classification described in Chapter 5. Given a set of unlabelled images, some of which contain an object of an unknown category, with unknown location and unknown size relative to the background, the method automatically identifies the images that contain the objects, localizes them and their parts, and reliably learns their appearance and geometry for subsequent classification. Our main contribution is a new approach which iteratively extracts new features and re-learns the induced classifier, improving class vs. non-class separation at each iteration. We have used this learn-
ing model to also introduce a new unified method for face detection and recognition (section 5.5).

To summarize the research contribution, this thesis presents a new natural learning approach for learning complex object classifiers, in which each individual part is detected by a learned part classifier. The first aspect of the natural approach is incremental learning which allows efficient learning in the online setting (Chapters 2,3). The second aspect of the natural approach is unsupervised learning (Chapter 5) which relaxes the requirement for human labeling which is not practical for extensive datasets, and consequently allows learning a more complex object model in which each part is detected by a learned classifier (Chapter 4).

The remaining of the thesis is organized as following. Chapters 2,3 discuss the methods for online learning, Chapter 4 the method for learning to recognize object parts and Chapter 5 the method for unsupervised learning. The thesis is concluded by a discussion (Chapter 6) and a suggestion for future work (Chapter 7).
Chapter 2

Learning to classify by ongoing feature selection
In this chapter we present an on-line feature selection method, which continuously modifies and improves the features used for classification based on the examples provided so far. The method is used for learning a new class, and to continuously improve classification performance as new data becomes available. In ongoing learning, examples are continuously presented to the system, and new features arise from these examples. The method continuously measures the value of the selected features using mutual information, and uses these values to efficiently update the set of selected features when new training information becomes available. The problem is challenging because at each stage the training process uses a small subset of the training data. Surprisingly, with sufficient training data the on-line process reaches the same performance as a scheme that has a complete access to the entire training data.

Current classification schemes typically use a set of examples for selecting appropriate classification features and training the classifier. The features used for a given class remain unchanged in subsequent applications of the classifier. This fixed learning is deficient for several reasons. First, properties of the class may change over time, requiring continual adjustment. Second, it is difficult to determine an optimal size for the training set: a small set will result in reduced performance, and a large set will require extensive training. Finally, it may be desirable to be able to perform classification as early as possible, and yet continue to improve when new data becomes available. A better approach is therefore to use an on-line scheme that adapts to changes in class appearance and continues to learn from new data as it becomes available.

The selection of useful features is a fundamental problem in learning a classification task. Irrelevant and redundant features not only complicate subsequent stages in the classification, but also degrade classification performance [22]. By selecting only relevant features, higher performance can be achieved with a smaller set of parameters in the model. Therefore, the goal is to select a set of features, which is optimal for classification and minimal in their redundancy.

The setting of the classical feature selection problem is the following. We are given initially a large set of $n$ potential features measured on a fixed training set. Our goal is to select from this set the optimal subset of $k$ features for the classification task. The problem can be summarized in the following way:

**Feature Selection** $(n, k)$: Given an initial set of $n$ features, find the subset with $k < n$ features that is maximally informative about the class.
For the use of information as a criterion for selecting effective features see [18],[70]. In the online learning scenario, examples and features extracted from them are continuously presented. This enables an ongoing improvement of the classifier focusing on the most recently seen examples. Such an improvement is possible if the learning algorithm is adaptive, in the sense that it improves the classifier each time a new example or a new feature is presented. Specifically, an adaptive feature selection algorithm is one that improves the subset of selected features in an online learning environment.

The online feature selection problem can be stated in the following way:

**Online Feature Selection** \((n, k, e)\): Given a time point in the online learning process following the presentation of \(e\) examples and \(n\) features, find the subset with \(k < n\) features that is maximally informative about the class, estimated on the \(e\) examples.

The main advantage of using an online learning scheme is the ability of the learning system to utilize new information while already functioning, and to adapt to a changing environment. This is important for several reasons. First, in cases in which the class appearance changes over time and cannot be foreseen in advance. For example, when learning to recognize a specific face, future appearance changes such as aging, glasses and facial hair, cannot be fully anticipated. Second, for recognition tasks with high variability, a large number of examples will be required. For example in handwritten character recognition hundreds of thousands of examples are used for training, and performance can still be improved if the system is trained on a new handwriting. Finally, for computational efficiency, an on-line selection method will also be of use when the set of features to consider is large, even in a non-online scheme. It then becomes possible to consider initially a limited set of candidate features, and consider new ones incrementally until an optimal subset is selected.

One approach to the online feature selection problem is simply to consider at each time step all the features and all the examples seen up to this time step, and apply a standard feature selection technique. However, this is highly inefficient, since the training set and feature set grow at each time step, and so does the running time of the feature selection method. We will develop therefore a method that considers only a fixed number of features and examples at each time step. As we shall see, such a method continues to improve with new examples and eventually outperforms methods that use a fixed number of examples.
2.1 Previous approaches

There are many different approaches to feature selection for classification [4]. Standard feature selection methods have been broadly divided into so-called wrapper and filter methods [31]. Wrapper methods directly evaluate the performance of a subset of features by measuring the performance of a model trained on this subset. A wrapper approach would require multiple model trainings at each update step, to compare different sets of features, and is therefore inappropriate due to online time constraints. Filter methods use some measure to estimate the importance of different features, independent of a specific classifier. A successful filter approach initiated with the MIFS algorithm [2] uses mutual information to measure the importance of features. Additional information-based methods were developed in [32], [70] and [71]. The mutual information between a set of features $F$ attaining discrete values belonging to $\mathcal{F}$ and a class variable $C$ with optional values $\mathcal{C}$ is defined as:

$$I(F; C) = \sum_{F \in \mathcal{F}} \sum_{C \in \mathcal{C}} P(F, C) \log \frac{P(F, C)}{P(F) \cdot P(C)}$$  \hspace{1cm} (2.1)$$

$I(F; C)$ measures the reduction in uncertainty about the identity of class $C$ given the observed features $F$, and high mutual information is necessary to obtain low classification error ([13], [52]). For these reasons, it is a natural measure to use for extracting useful classification features [18],[70].

We follow this line of work by using mutual information to measure feature quality, together with a novel strategy that is applicable in the online scenario. In [51] a filter type method was proposed for a simplified version of the online feature selection problem, in which all examples are available in advance, and the update step includes only new features. This problem formulation is inappropriate for our problem, since in learning online classification, new examples are continuously presented. Our algorithm, described in the following section, is the first to treat the on-line problem fully, and handles efficiently both feature updates and novel examples.

The remainder of the chapter is organized as follows. In the next section we describe in detail the proposed algorithm. Experimental evaluation of the scheme tested on different visual object classification tasks is presented in Section 2.3. These results and future directions for research are discussed in Section 2.4.
2.2 The on line feature selection algorithm

2.2.1 Algorithm overview

The proposed method is designed to select features for classification in an online training scheme. We assume a scheme in which new labeled examples and new features extracted from these examples are continuously provided to the method by a separate, task specific process (see section 2.3). The proposed algorithm proceeds in an iterative manner: it receives at each time step either a new example or a new feature or both, and adjusts the current set of selected features. When a new feature is provided, the algorithm makes a decision regarding whether to substitute an existing feature with the new one, or maintain the current set of features, according to a value computed for each feature relative to the current feature set. The value of each feature is evaluated by the amount of class information it contributes to the features in the selected set. The algorithm also keeps a fixed-size set of the most recent examples, used to evaluate newly provided features. In this way, the evaluation time of the features value, which depends only on the number of examples and the number of features in the selected set, is constant throughout the learning.

It is convenient to divide the design of such a scheme into two main components: search strategy and merit value. The search strategy handles the problem of searching the space of all subsets of features, which is of exponential size in the number of features \( n \). Most feature selection algorithms exploit greedy search strategies (e.g. sequential search [25]), which iteratively select the most informative feature given a set of already selected features ([2], [32], [31]). In an online environment, this strategy requires the re-use of the entire set of training examples and features at each update step, and it is therefore inapplicable in the online setting. Our algorithm uses therefore a different search strategy, termed substitution, in which each newly examined feature is added to an evaluated set of features, and then the least contributing feature in the resulting set is discarded. This strategy allows the algorithm to evaluate and compare only the features in the selected set of features, and not the entire set of features as greedy search strategies require, and it is a key to the algorithm’s ability to perform in an online manner. The merit value (MV) of a feature expresses the value of a feature for classification given an already selected set of features. Several merit values were suggested in [2], [71] and [32], and were incorporated in greedy search based methods. In order to evaluate the merit value of a feature, these methods require the entire set of training examples. We use
a small set of examples instead for evaluating a merit value similar to the one suggested in [71], and continue to update its evaluation as new examples are seen. In the next sections we present the details of our algorithm, starting with the details of our implementation for measuring the feature value online.

2.2.2 Measuring feature merit value

In this section we describe the computation of the classification value of each feature. In our method, for each feature belonging to the selected set, this value should express the value for classification lost by eliminating it, or, equivalently, the value for classification gained by adding it to an already selected set of features. We assume that each feature $f$ has discrete values on each example, and can be viewed as a discrete random variable. Similarly, we regard the class label $C$ to be a discrete random variable.

Given a feature $f$ and a set of selected features $S$, the desired merit value $MV(f, S)$ should express the additional class information gained by adding $f$ to $S$. This can be measured using mutual information by:

$$MV(f, S) = I(f, S; C) - I(S; C)$$ (2.2)

Where $I$ stands for mutual information. This merit value can be computed given the joint distribution $P(f, S, C)$, of all the variables in $S \cup \{f\} \cup \{C\}$. Unfortunately, estimating the joint distribution function of $|S|+2$ random variables is impractical in real world applications, due to sample size limitations, and therefore an approximation is often used. Several approximations were suggested in [71], [2] and [32]. We chose for our implementation the merit value suggested in [71], which was shown in empirical comparisons to produce highly effective classification features [18]. This value considers only second order relations between the feature variables and the class, and is expressed by:

$$MV(f, S) = \min_{g \in S} I(f, g; C) - I(g; C)$$ (2.3)

The term $I(f, g; C) - I(g; C)$ expresses the additional class information $f$ delivers with respect to $g$, where $g$ is an already selected feature. By taking the minimum over all features in $S$, we estimate the minimal additional class information delivered by $f$ with respect to $S$. There are two factors affecting this value: the feature’s individual contribution to the mutual information, and its redundancy with respect to the set. If a feature $g'$ similar to $f$ already exists in $S$, then the minimized expression for $g = g'$ will be low, regardless
of how informative \( f \) is, resulting in a low value for redundant features. In order to compute this merit value for each feature \( f \) we need to estimate the joint probability of just two features and the class, \( P(f = x, g = y, C = z) \). For each new feature \( f \), its values on the set of latest examples maintained by the algorithm are determined, and are used to estimate its distribution. In addition, each time a new example is seen, the values of \( f \), \( g \) and \( C \) are determined, and the probability estimates for all pairs of features in the selected set are updated. In a similar manner, we also estimate for each selected feature its joint probability with the class \( (P(f, C)) \) for a purpose explained below. This method of update allows us to utilize all the statistics seen for a feature in the selected set from its arrival time onwards. Clearly, the joint distribution of two features can be only estimated on the example set seen together by both features.

The online scheme poses the problem of different sample sets for different features. Consider the case in which \( f^{\text{old}} \) is an old feature and \( f^{\text{new}} \) has just been presented, and we wish to re-estimate the merit value of \( f^{\text{old}} \). The term \( I(f^{\text{old}}, f^{\text{new}}; C) - I(f^{\text{new}}; C) \) can only be estimated on the set of recent examples, although the old feature may have a much higher class information, estimated on previously seen examples. As a result, the new estimated score of \( f^{\text{old}} \) may become too low, leading to its elimination from the feature set. In practice, since we would like the full information gathered on \( f^{\text{old}} \) to be expressed in its score, we use an approximation instead of the discussed term. This approximation is motivated by the following inequality:

**Proposition:** Let \( X, Y, C \) be discrete random variables. If \( X \) and \( Y \) are independent given \( C \) (i.e. \( P(X,Y|C) = P(X|C) \cdot P(Y|C) \)) then the following inequality holds:

\[
I(X,Y;C) - I(X;C) \leq I(Y;C)
\]

**proof:**

From the conditional independence we compute \( I(X;Y|C) \) using equation 2.1 to obtain \( I(X;Y|C) = 0 \). We next evaluate the left hand side of the inequality:

\[
I(X,Y;C) - I(X;C) = (\text{mutual information chain rule})
\]
\[
= I(X;C) + I(Y;C|X) - I(X;C) =
\]
\[
= I(Y;C|X) = (\text{mutual information chain rule})
\]
\[
= I(Y;C,X) - I(Y;X) = (\text{mutual information chain rule})
\]
\[
= I(Y;C) + I(X;Y|C) - I(Y;X) = (\text{conditional independence of } X \text{ and } Y)
\]
\[
= I(Y;C) - I(Y;X) \leq (\text{non-negativity of mutual information})
\]
\( \leq I(Y; C) \).

We have proved that under the common assumption that the features are
independent given the class, \( I(f^{old}; C) \) is an upper bound for \( I(f^{old}, f^{new}; C) - I(f^{new}; C) \). When the bound is not tight, this causes the score for \( f^{old} \) to
be higher than the desired score (eq. 2.3), and therefore older features have
a priority over new ones. This property is justified due to the better estimation we have of the score of older features and it also proved useful in practice. The bound can be computed directly from the feature’s joint distribution with the class \( P(f^{old}, C) \), which can be reliably estimated from the entire set of examples seen for \( f^{old} \). The merit value used is therefore:

\[
MV(f, S) = \min_{g \in S(f)} \begin{cases} I(f, g; C) - I(g; C) & g < f; \\ I(f; C) & \text{else} \end{cases}
\] (2.4)

Where \( g < f \) means that \( g \) precedes \( f \). In the next section we describe how features are selected incrementally by the algorithm based on their estimated values.

### 2.2.3 Algorithm description

The algorithm maintains and updates three data structures: a set of examples (recent examples) containing the \( l \) most recently seen examples, a set of features (selected features), containing the most informative \( k \) features seen so far, and the estimates of the probability distributions (probability estimates) for the features in the selected set. The maintained probability estimates are: \( (P(f, g, C)) \) for each pair of features in the selected set, estimated on the common set of examples observed for both features, and \( (P(f, C)) \) for each feature in the selected set, estimated on the examples seen for that feature. The exact form of the features and the examples is task specific (see examples in Section 2.3); we only assume that we can obtain the discrete value of \( f \) on a given example \( e \). New examples are now observed, and from them some new features are being extracted. The algorithm operates whenever a new example or a new feature is provided. The new examples are used for improving the feature probability estimates, and the new features replace selected features if their measured merit value is higher. The following pseudo-code summarizes the algorithm:

1. **new_feature**\((f)\)

   1.1 add(feature_set, f)
(1.2) \( probability\_estimates \leftarrow estimate\_new\_feature \)
\[(probability\_estimates, f, recent\_examples)\]

(1.3) foreach \( f \in feature\_set \)

(1.4) \( MV(f) \leftarrow compute\_merit\_value \)
\[(f, feature\_set \setminus \{f\}, probability\_estimates)\]

(1.5) if \(|feature\_set| > k\)
\[remove(feature\_set, \text{arg } \min_f(MV))\]

(2) new\_example(\( e \))

(2.1) \( probability\_estimates \leftarrow update\_estimates \)
\[(probability\_estimates, e)\]

(2.2) enqueue(recent\_examples, \( e \))
(2.3) if \(|recent\_examples| > l\)
\[dequeue(recent\_examples)\]

The learning process can be presented either with a new example, in which case new\_example is applied, or with a new feature, in which case new\_feature is applied. When a new feature \( f \) is provided, it is added to the set of selected features (1.1), its value is determined on the set of recent examples and used to estimate its distribution (1.2). The value for classification (MV) of each feature (eq. 2.4) is computed according to its probability estimates (1.4). The feature with the lowest value is discarded from the feature set (1.5). When a new example is presented, the values of each feature are determined for that example, and the feature probability estimates are updated accordingly (2.1). In addition, the new example replaces the oldest one in the set of most recent examples (2.2 and 2.3).

Notice that by first adding the new feature to the selected set, the algorithm ensures that each seen feature is evaluated in the set at least once. During the ongoing presentation of features, the feature space is searched by occasionally substituting a feature in the set with the newly presented one. The properties of the merit value measure (eq. 2.4) ensure that the elimination of the selected feature results in the minimum possible loss in class information of the entire set. In the next section we analyze the time complexity of the algorithm, an important aspect in online learning schemes.
2.2.4 Complexity analysis

An important property of the algorithm is its ability to maintain a constant running time for each update step throughout the online learning process, as we show in this section. The **new_example** procedure performs an update of the joint probability estimate for each pair of selected features (2.1), and since there are \(k\) features in this set, this update is performed in \(\Theta(k^2)\) time. The **new_feature** procedure first computes the joint probabilities of the new feature with the selected features (1.2) estimated on the current examples set which is of size \(l\), and can therefore be accomplished in \(\Theta(l \cdot k)\) time. The procedure then computes the feature merit values (1.4), calculating the class mutual information of every pair of selected features from their joint distributions, and is therefore computed in \(\Theta(k^2)\). In total, the **new_feature** procedure time complexity is \(\Theta(k^2 + l \cdot k)\). The running time of both update procedures is therefore independent of the number of features or examples seen so far. In contrast, the running time of standard feature selection algorithms applied on a set of \(n\) features and \(e\) examples, depends on \(n\) and \(e\), and will therefore increase at each time step in an online environment. We compare our algorithm to the Max-Min algorithm \([71],[18]\), which uses a similar approach for feature selection and performs the task in time complexity \(\Theta(k \cdot n \cdot e)\). Consequently, the Max-Min algorithm is not applicable in an online environment, since each update requires the evaluation of the entire feature and example set which constantly grows in the online learning scheme.

The proposed algorithm offers a computational improvement of the Max-Min algorithm even in the standard feature selection case. Since \(k\) and \(l\) can be viewed as relatively small constants, the running time of the proposed algorithm for a fixed set of examples and features is \(\Theta(n + e)\), obtained by summing the running times of all update steps, which is an improvement to the Max-Min running time, \(\Theta(n \cdot e)\). In terms of memory complexity, the improvement is large, being constant for the proposed algorithm, and \(\Theta(n \cdot e)\) for the Max-Min algorithm. We conclude the algorithms presentation with a short review of its key properties.

2.2.5 Key properties

The algorithm maintains at each time point a set of recent examples of a constant size, and a set of the most informative features seen so far. Each time a new example is observed, the statistics of the features are updated, and the recent set of examples is updated. Each time a new feature arrives
it is evaluated on the set of recent examples, and competes together with the selected features to remain in the feature set. The competition is mediated by measuring each feature’s class information with respect to the other features in the set. Each update step is performed in a constant computation time throughout the online learning process. In the next section we examine the results obtained using this feature selection algorithm for visual classification tasks.

2.3 Experimental Results

In this section we describe the experiments testing the performance of the proposed method, and compare it with unrestricted methods that make full use of the entire sets of examples and features. The first part of the section briefly describes the experimental setup, the second part analyzes the proposed online method, and the last part compares the online method with an existing off-line feature selection method.

2.3.1 Training and testing the classifier

In our experiments we performed several different object detection tasks. The class domains tested were faces, cows, planes and cars. In all the domains the task was to decide whether or not the image contained a class instance. In order to accomplish the task, the classifier was presented with roughly aligned class images and non-class images. The images contained 50 × 80 pixels in the face detection task and 80 × 50 pixels in the remaining tasks (figure 2.1).

The detection method used was a fragment-based classification described in [71]. The basic features used by this method are image patches, or fragments, at different positions and sizes, similar to a number of recent schemes [14], [42]. Except for the feature selection stage, all other classification stages were kept the same in all the experiments, allowing us to compare the feature selection stage independent from the other classification stages. We next present a detailed description of our implementation.

In order to determine the results in each experiment, we averaged the results across 20 cross-validation iterations. At each iteration we randomly selected several hundreds of images, equally divided between class and non-class, for training. The remaining 500 images were used to test the classifier. In each experiment we used for training the fragments extracted from the class training images. The fragments varied in size and proportions, from 5 to 50 pixels in height and width, and their locations spread over the
Figure 2.1: Object detection tasks. (a) Examples of class images (b) Examples of non-class images. (c) Examples of extracted fragments.

entire source image with equal spacing. These image fragments form the initial pool of potential classification features, from which a subset of useful features is selected during learning. Figure 2.1(c) shows examples of automatically extracted features for the different tasks. The similarity between each fragment and each image was measured by the maximal normalized cross correlation between the fragment and the image, obtained across all possible image locations.

Using the training set, a detection threshold was set for each feature, by maximizing its mutual information with the class on the training set. The threshold was then used to quantize each feature measurement to a binary feature. In the next stage we applied the different feature selection methods to the set of binary feature vectors created by previous stages. In the online learning scheme we sequentially presented the training examples in a random order. The features were either presented in a random order during the example presentation or presented immediately after their source image example was presented. To measure the classification performance of a selected feature set at any point in the process, we used the selected features and the entire training set of images to construct a Naive-Bayes classification function [50], and tested its performance on the test images. In this manner we tested the performance of feature sets using the same classifier by different learning procedures.

2.3.2 Classification results and analysis

In this section we analyze the properties and performance of the on-line algorithm in the face and cow detection tasks. The algorithm was tested for both tasks with $l = 50$, and was gradually presented with all 400 training
examples and the entire set of features (about 8,000) arranged in a random order. The examples were presented in 200 epochs, each epoch consisting of one class example, one non-class example and a \( \frac{1}{200} \) fraction of the fragments. Figure 2.2 summarizes the results of this experiment for both detection tasks.

The classification performance of the online feature selection method was measured on the test images throughout the training period. Figure 2.2(b) shows the classification error of the resulting classifier as measured at selected time points during the training for both detection tasks. The decrease in classification error illustrates that the online feature selection method utilizes the new information as it becomes available, demonstrating the feasibility of constantly improving the set of selected features.

The algorithm can also be viewed as attempting to maximize an objective function: at each feature substitution, the algorithm tries to maximize the class information delivered by the remaining set. This objective function can be measured by the average merit value (eq. 2.4) across the features in the selected set. The value of this objective function during the algorithms operation, shown in figure 2.2(c), generally increases in time, and the increase in information value is correlated with the improved classification performance.

Another relevant characteristic of the algorithm is the rate of feature substitutions over time (figure 2.2(d)). The feature substitution rate is high during the initial iterations, indicating a fast change in the selected feature set, and gradually decreases to a low constant level, indicating that the set remains adaptive throughout the learning process.

One question that emerges from the scheme is the role of the \( l \) parameter, the number of maintained recent examples. Large \( l \) increases memory requirement and computation time which is linearly dependent on \( l \) (Section 2.2.4), but allows a better initial estimate for each new feature. We compared the performance of the algorithm with different values of \( l \), summarized in figure 2.2(e). As expected, large \( l \) improve classification performance, suggesting a trade-off between performance, length of training, and computational cost.

In the next part we compare the online method with a similar off-line scheme that uses in a comprehensive manner the entire set of examples and features, which we refer to as a ‘comprehensive’ selection method.
Figure 2.2: Analysis of the online feature selection method. The online algorithm was run with $l = 50$ (number of maintained examples, except in (e)) and $k = 100$ (number of selected features); results are averaged over 20 cross-validation iterations. Left: cow detection task, right: face detection task. (a) Examples of selected features after the last time step, shown in red outline over their source image. (b) Classification error of the classifier constructed with the currently selected features as a function of the percent of presented examples and features during the online process. The classification error is measured at the equal error point in the ROC curve (i.e. equal false alarm and miss rates). 100% of the examples = 400; 100% of the features = 10,530 cow features, 6,292 face features. (c) Average merit value of the selected features as a function of the percent of presented examples and features during the online process. (d) Feature substitution rate measured in the preceding 100 feature presentations, as a function of the percent of presented examples and features during the online process. (e) Classification error at the end of the training process, as a function of $l$, the number of maintained examples.
2.3.3 Comparison of online and comprehensive feature selection methods

In the following experiments we compared the on-line algorithm to a similar but comprehensive classification scheme. We used for comparison the feature selection method described in [71], also called the Max-Min method. This method uses a similar merit value to select features, allowing us to compare directly the online scheme with a similar, but off-line and comprehensive selection scheme. The Max-Min method is a greedy iterative procedure. It maintains a growing set of selected features $S$ which is initially empty. In each iteration the merit of all features not selected yet is computed according to equation 2.3 and the feature with the highest merit value is added to $S$. The stopping criteria is either reaching a minimal merit value or a size limit on $S$. We performed two different comparisons, which we subsequently present. The first compares the methods on the fixed cows and faces data-sets, while in the second comparison we further stress the advantage of using the on-line method by testing the methods in a changing environment requiring an ability to adapt.

Comparison of on-line and comprehensive feature selection methods on a fixed data-set

The following experiment was carried out on the the two data-sets already presented including cow and face images, as well as non-class examples, with the task of binary classification. For each cross-validation iteration, the training examples and the features were presented in an identical random order to both algorithms. The presentation order was arranged in epochs as in the previous experiment. The Max-Min method was applied every 4 epochs to the entire set of features and examples seen until that epoch. The limitation of any comprehensive method is that its training set is fixed to a certain size, compared with the online method which continues the learning process. We measured the classification performance of the Max-Min method at each example presentation, and compared it with the final performance of the online method using an $l = 50$ example memory size (figure 2.3(a)). We marked for example the point in figure 2.3(a) representing 100 examples seen by the offline method. For this point our algorithm shows a 2.8% lower classification error (on cow detection), even though it uses a buffer of 50 examples only. The results show that given a large enough training period the online method outperforms the comprehensive method trained with fixed sets of examples and features. The improved performance
is illustrated by the ROC curves in figure 2.3(b), obtained in the same experiment, at the epoch in which the number of examples used to train the Max-Min was 50. This is a relevant comparison, because the online method can also maintain only 50 examples.

When training both methods on the same sets of examples and features, the comprehensive method has a clear advantage at estimating each feature’s probabilities. The reason for this is that in the comprehensive method each feature is sampled on the entire example set, while in the online method the sample size per feature is much smaller (as low as $l$ for each new feature). To separate the difference in sample sizes from other factors, we performed another experiment, allowing the number of recent examples saved by the online method ($l$) to be 400, the size of the entire set of examples. In this case the selected features at the end of the online training process were evaluated on the entire training set, however, at each time point during the process, only the examples seen so far were considered. The results (figure 2.3(c)) show that once the sampling factor is reduced, the online algorithm performs as well as the comprehensive method, even though it uses a smaller memory for features ($k$ instead of $n$), and updates them in a random order. In these settings, our algorithm addresses the simplified OFS problem addressed in [51] (Section 2.1), in which all the examples are available for evaluating each feature. The results illustrate that our method provides an efficient solution to this problem, since it performs under these settings as well as the compared comprehensive method.

**Classification in a changing environment**

In the following experiment we compared the proposed and the Max-Min algorithms in a changing environment on two tasks: car detection and plane detection. The changing environment was simulated in the following way: the class examples consisted of two subclasses of images presented one after the other. The plane images contained military and civilian planes, and the car images contained sedans and SUVs (Sport Utility Vehicles). In both cases the two subclasses are related and share common visual features, but they also differ in others. The online method was first trained and tested on one subclass and was then trained and tested on the second, simulating a shift in the class appearance.

The online method was trained as following. First it was presented with a randomly ordered subset of examples belonging to one subclass and non-class examples. After each class image presentation, the fragments extracted from that image were presented to the algorithm as classification features.
Task: Cow Detection

![Classification error of the offline method as a function of the number of presented examples (dotted curve), and the classification error of the online method at the end of the training process (solid line). The marked point represents 100 observed examples.](image)

![ROC curves for online method (solid blue curve) trained with $l = 50$ (number of maintained examples) and for the offline method (dotted red curve) trained with the first 50 training examples, and the features provided so far.](image)

![ROC curves for the offline method (solid blue curve) trained on the entire training set and for the online method (dotted red curve) with $l = 400$, allowing it to remember all the examples seen at each time step.](image)

Figure 2.3: **Performance of online and offline feature selection methods.** The online algorithm was run with $l = 50$, and the offline method was iteratively applied at each presentation of a feature or an example. Both methods selected $k = 100$ features. (a) Classification error of the offline method as a function of the number of presented examples (dotted curve), and the classification error of the online method at the end of the training process (solid line). The marked point represents 100 observed examples. During the example presentations features were also presented according to the training order described in Section 2.3. The classification error is measured at the equal error point in the ROC curve. (b) ROC curves for online method (solid blue curve) trained with $l = 50$ (number of maintained examples) and for the offline method (dotted red curve) trained with the first 50 training examples, and the features provided so far. This is a detailed comparison of the methods for the point representing 50 examples in graph (a). (c) ROC curves for the offline method (solid blue curve) trained on the entire training set and for the online method (dotted red curve) with $l = 400$, allowing it to remember all the examples seen at each time step.
Following the first training period, the method was similarly trained with class images belonging to the second subclass. In total 300 training images were used in the car detection task and 200 training images in the plane detection task. The test set contained 500 novel images. From each class example about 500 fragments were extracted. The off-line method is capable of selecting the best features from a fixed training set. We trained the method once with class images belonging to one subclass, once with second one, and once with the combined training set. For each training set the off-line method used the entire set of fragments extracted from that training set to select the classification features. The selected features were evaluated by incorporating them in a Naive-Bayes classifier as in the previous experiment. The results were averaged across 20 cross-validation iterations and are summarized in figure 2.4 and table 2.1.

The learning curves for the on-line method in figure 2.4 (solid line) illustrate the method’s adaptation to the change in class appearance in both tasks. Despite the observed adaptation, the classification performance on the first subclass (gray bar) remains mostly unharmed, demonstrating that the new learned ability does not replace the old one but is accumulative. In order to test the asymptotic behavior of the algorithm we also trained it on 800 examples in the plane detection task. As shown in the bottom right part of figure 2.4, increasing the number of training examples contributes an additional 12% decrease in the classification error. These results show that for classification tasks with large inner class variability, the online method utilizes the ongoing example presentation to continually improve performance. Notice that the order in which environments were presented had little affect on the error of the online method (first 2 rows in table 2.1).

In comparison to the off-line method our scheme has three advantages. First, the results show that if the off-line was trained in the first environment (dashed line in figure 2.4) it has a poor ability to perform in the changed environment, in contrast with the online scheme. The results demonstrate the benefit of using an adaptive feature selection method instead of a comprehensive one, which is fixed after training and cannot adapt to a changing environment. Second, the online method outperforms the off-line trained with both subclasses (dash-dot line in figure 2.4). By maintaining the most recent examples only (Section 2.2), the online method is shown to specialize in the current environment. Finally, when tested separately on each data-set, the performance of the online method does not fall from that of the off-line method trained specifically on that data-set (2 bottom rows in table 2.1). These results are surprising considering the approximations made by the online algorithm, and its limited capacity for examples. They sug-
Task:

(a) Car detection

(b) Plane detection

Figure 2.4: A comparison of the on-line and off-line selection algorithms in a changing environment. (a) Examples of features selected by the online method (b) The equal error rate of the classifiers averaged over 20 cross-validation iterations. The online classifier was first trained in one environment (left half), and then in a modified environment (right half). \( l = 60 \), cars: \( k = 25 \), planes: \( k = 50 \). The off-line method was trained with the same examples in a comprehensive manner. In the left half of the graph for each class methods were tested on the first subclass and in the right half on the second. The gray bar at the right side of each graph marks the error of the online method on the first data-set, after being trained on the second one. The results show the advantage of the on-line scheme in adapting to a shifting population.
Table 2.1: Changing environment experiment. Equal error rates (and std) for the off-line and the on-line methods. The training sets for the online method are listed in their presentation order. The online method was tested at the end of each sub-class training epoch. The lowest error rates are emphasized.

### Car detection task

<table>
<thead>
<tr>
<th>Method</th>
<th>Train set</th>
<th>Test: sedan</th>
<th>Test: S.U.V</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-line</td>
<td>Sedan→SUV</td>
<td>0.7 ± 0.15%</td>
<td>1.9 ± 0.20%</td>
</tr>
<tr>
<td></td>
<td>SUV→sedan</td>
<td>0.8 ± 0.13%</td>
<td>1.9 ± 0.21%</td>
</tr>
<tr>
<td>Off-line</td>
<td>Sedan, SUV</td>
<td>1.2 ± 0.21%</td>
<td>2.4 ± 0.21%</td>
</tr>
<tr>
<td></td>
<td>Sedan</td>
<td>1.3 ± 0.26%</td>
<td>5.8 ± 0.67%</td>
</tr>
<tr>
<td></td>
<td>SUV</td>
<td>2.7 ± 0.32%</td>
<td>2.3 ± 0.29%</td>
</tr>
</tbody>
</table>

### Plane detection task

<table>
<thead>
<tr>
<th>Method</th>
<th>Train set</th>
<th>Test: military</th>
<th>Test: civil</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-line</td>
<td>military→civil</td>
<td>18.4 ± 0.4%</td>
<td>12.6 ± 0.4%</td>
</tr>
<tr>
<td></td>
<td>civil→military</td>
<td>18.6 ± 0.4%</td>
<td>11.4 ± 0.5%</td>
</tr>
<tr>
<td>Off-line</td>
<td>military, civil</td>
<td>18.9 ± 0.5%</td>
<td>12.9 ± 0.5%</td>
</tr>
<tr>
<td></td>
<td>military</td>
<td>18.6 ± 0.5%</td>
<td>18.3 ± 0.5%</td>
</tr>
<tr>
<td></td>
<td>civil</td>
<td>20.5 ± 0.5%</td>
<td>11.9 ± 0.5%</td>
</tr>
</tbody>
</table>

gest that the more efficient on-line algorithm can be a viable alternative to comprehensive methods even when the training set is fixed.

### 2.4 Conclusions

We presented a novel method for learning classification features in an online environment. The method enables classification systems to continuously improve the features they use for classification, and consequently continuously improve the overall classification performance. The method overcomes limitations of current methods which first use a set of training examples to select classification features, and then use the features for all future applications of the classifier. The online method has the ability to adapt effectively to drifting distributions. In fixed environments, using limited computational resources, the online method continuously improves, and with sufficient training will outperform off-line selection methods. The method can also be employed for efficiently exploring sets of features that are much larger than the ones used today by classification systems.
Several extensions to our method may be considered. The feature evaluation method can be improved by considering higher order dependencies between features, as suggested in [70], in addition to second order dependencies, considered by our method. It may also be of use to consider feedback from the classifier’s performance affecting the feature selection during the online process, enabling it to focus on specific features for difficult examples. Such a feedback may also be used to automatically determine the required number of features \(k\). For modelling biological vision, it will be of interest to enable the algorithm to operate in the limiting case \(l = 1\), where the system makes continual adjustments without explicitly storing multiple examples.

The proposed method provides a solution for a key component in the online learning scheme, the feature selection stage. Future work should examine the on-line learning of other aspects of the learning process, such as on-line setting of the optimal detection thresholds for the selected features, and training the parameters of the classification function. This will lead in the future to complete online learning systems, which no longer depend on a fixed training set.
Chapter 3

Learning model complexity in an online environment
In this chapter we introduce the concept and method for adaptively tuning the model complexity in an online manner as more examples become available. Challenging classification problems in the visual domain (such as recognizing handwriting, faces and human-body images) often require a large number of training examples, which may become available over a long training period. This motivates the development of scalable and adaptive systems which are able to continue learning at any stage and which can efficiently learn from large amounts of data, in an on-line manner. Previous approaches to on-line learning in visual classification have used a fixed parametric model, and focused on continuously improving the model parameters as more data becomes available. Here we propose a new framework which enables online learning algorithms to adjust the complexity of the learned model to the amount of the training data as more examples become available. Since in online learning the training set expands over time, it is natural to allow the learned model to become more complex during the course of learning instead of confining the model to a fixed family of a bounded complexity. Formally, we use a set of parametric classifiers $y = h_\theta^\alpha(x)$ where $y$ is the class and $x$ the observed data. The parameter $\alpha$ controls the complexity of the model family. For a fixed $\alpha$, the training examples are used for the optimal setting of $\theta$. When the amount of data becomes sufficiently large, the value of $\alpha$ is increased, and a more complex model family is used. For evaluation of the proposed approach, we implement an online Support Vector Machine with increasing complexity, and evaluate in a task of handwritten character recognition on the MNIST database.

The current performance of object classifiers is continuously improving, but is still far from human performance. Typical state-of-the-art natural-object classifiers [e.g. horses vs. background] achieve an equal error rate of 1-3% under controlled settings [40], and an average precision of 30%-80% in uncontrolled settings [46] in which the object size and its viewpoint significantly vary. This performance is still far from human performance which is usually perfect in such tasks. These state-of-the-art results are obtained with datasets containing less than 1,000 positive class examples. In contrast, handwritten character recognition systems [45] successfully approach human performance by using much larger training sets. In addition, new approaches for multi-class object recognition report improved performance using millions of examples [66]. Human performance in classification also relies on extensive training. Even though a new category can sometimes be learned from only a few examples, the entire visual experience including objects with a similar appearance can be used in forming the new representation [3]. These studies suggest that improved classification can be
obtained by training on larger datasets enabled in an online environment in which the training set is unbounded and in which large amounts of data can be efficiently processed. A larger dataset does not only lead to a more accurate model parameter estimation but also enables learning a more complex model which may lead to a more substantial performance improvement. While current online learning algorithms focus on learning the parameters of fixed complexity models we propose an online framework in which the model complexity is learned as more examples become available.

In the online setting the learning system receives one example at each time step and outputs a model based on all the examples received so far. Scalability and efficiency considerations usually do not allow the learning scheme to store the entire set or to go over it at each example presentation. Among online learning algorithms neural networks such as the Perceptron [56] are the most well-known. In a recent line of work a number of useful machine learning algorithms were adapted to online requirements: feature selection [35], similarity map learning [54], and SVM [5]. We follow this line of work, by proposing an online solution for model complexity learning. A main reason for the growing interest in online learning algorithms is their advantage in terms of memory and time complexity, often required to handle the large amounts of data which have become available mainly from the internet. From this point of view the online schemes are viewed as providing fast approximations to results obtained by batch methods. It is important to note, however, that online learning has two additional advantages: scalability - the system continues to learn in its working environment, and adaptability - the system adapts to changes in its environment. To meet these two requirements, it may not be enough to train the model in an online manner, but it is may also be required to update the model itself: either to choose an entirely new model family for adaptation to be optimal, or to increase the model complexity for improved scalability.

The goal of learning a classifier is to minimize its expected error on general examples (generalization error). This is usually achieved in two stages: in the first stage, a family \( H \) of possible classifiers is chosen and in the learning stage a specific classifier \( h \in H \) is selected. In the learning stage a natural goal is to minimize the empirical error on the training examples. If the set \( H \) of allowed models is too broad, it may result in over-fitting the classifier to the training data, and therefore fail to minimize the expected error of the classifier [10]. A method for avoiding over-fitting is to limit the complexity of the model used for classification. For example, fitting a high-degree polynomial to a small set of points is prone to over-fitting; the degree should determined by the number of data points, and can increase as more
Figure 3.1: A hierarchy of complexity classes. The family of models $H$ includes a series of nested complexity classes ($H^1 \subset H^2 \subset H^3 \subset \ldots \subset H$), with each complexity class providing additional optional models with respect to its predecessor. Given a machine learning algorithm which finds in each $H^\alpha$ the best model $h^\alpha$ describing the training data, the complexity selection task is to determine the $\alpha$ for which $h^\alpha$ best fits the unseen data.

data becomes available. To deal with this issue in the general case, consider the case in which $H$ includes a series of nested complexity classes $H^\alpha$ ($H^1 \subset H^2 \subset H^3 \subset \ldots \subset H$), with each complexity class providing additional optional models with respect to its predecessor, as illustrated in figure 3. The learning problem is then to first find the appropriate family $H^\alpha$, and then learn the optimal model within this class by optimizing $\theta$, which denotes the parameters defining the classifier within its complexity class. For example, the first stage can determine the degree of a polynomial used to approximate the data, and the second stage determines the optimal coefficients of the approximating polynomial. The model complexity selection is to determine the optimal value of $\alpha$. A common practice for selecting $\alpha$ is cross-validation.
The goal is to obtain an estimate of the generalization error by learning the model parameters on a training set, while validating the learned classifier on a separate validation set. Robust estimates are obtained by averaging the results of several such iterations with different training/validation sets. An average generalization error is obtained for each value of $\alpha$, and the most successful $\alpha$ is chosen. There also exist other approaches relying on methods for approximating the generalization error of a given classifier such as the Minimum Description Length (MDL) [55] and the Structural Risk Minimization (SRM) [69]). It is important to note that trying to learn $\alpha$ as an additional parameter together with the set $\theta$ does not produce the same results as treating them separately. Consider again the example of fitting a target real valued function given by $N$ noisy sample points with $H$ being the class of polynomials and $\alpha$ the polynomial degree. If both $\alpha$ (the degree) and $\theta$ (the coefficients) are used together to minimize the error on the training example, it will always be possible to achieve a perfect fit to the data with a polynomial of degree $\alpha = N - 1$, which will fail in approximating new sample points. In the current approach, the optimal value for $\alpha$ is determined online, and changes adaptively as more data becomes available. In the online learning setting standard cross-validation cannot be used, since it requires saving all past examples and since it is computationally expensive. We next present a framework allowing the selection of model complexity within the online learning setting. Our solution can be also applicable in cases in which the data set is fixed but too large for performing cross-validation on the entire dataset. In the next section we define the algorithmic framework of our method. In section 3.2 we present an actual implementation which uses an online SVM solver as the learning system and in the section 3.3 we present experimental results of this implementation on the MNIST handwritten digit recognition database.

3.1 Algorithmic Framework

We define the problem of online model complexity selection as follows. The online learning system $M$ is presented with sequential example pairs:

$$\langle x^n, y^n \rangle : x^n \in X, y^n \in Y, n \geq 1$$

where $x^n$ is the $n$’th received pattern, and $y^n$ its target value (either discrete or continuous), each pair drawn independently at random according to an unknown distribution $\mu(X, Y)$. At each presentation the learner performs a
computation resulting in an output classifier:

\[ h_\alpha^\theta : X \mapsto Y, h_\alpha^\theta \in H^\alpha \subset H \]

where the positive integer \( \alpha \in \mathbb{N} \) represents the complexity of its class \( H^\alpha \) and the parameters \( \theta \) represent the classifier parameter of the model in \( H^\alpha \). The goal of online learning, is after each example presentation and based on all seen examples, to produce the classifier \( h_\theta^\alpha \) with the minimal generalization error:

\[ \text{Err}(\alpha, \theta) = E_{X,Y} \ell(Y, h_\theta^\alpha(Y)), (X,Y) \sim \mu \]

where \( \ell(Y, \hat{Y}) \) is a loss function associated with the classifier making decision \( \hat{Y} \) when the true target is \( Y \).

The goal of the online model complexity selection is to select the complexity \( \alpha \) for which the minimal error classifier is obtained. Notice that we do not explicitly require the nested structure of complexity classes \( (H^\alpha \subset H^{\alpha+1}) \) as long as the classes are of increasing complexity.

The online model selection (OMS) algorithm which we next present is a general framework for the online model complexity selection problem defined above. The OMS algorithm can be intuitively described as following. We train two online learning systems each with a fixed complexity. The learning systems receive the examples and produce after each example a learned classifier, based on the history of seen examples, with one learning system producing a classifier of complexity \( \alpha \), \( h_\theta^\alpha \), and the second a classifier with complexity \( \alpha + 1 \), \( h_\theta^{\alpha+1} \). \( \hat{\theta} \) marks the parameters learned by each learning system within its fixed complexity class. The classifier \( h_\theta^\alpha \) represents the current classifier learned by our algorithm. When a new example is presented, both classifiers are validated on it. Based on the history of validation errors made by each of the classifiers their generalization errors are estimated. When the estimate for the generalization error of \( h_\theta^{\alpha+1} \) is significantly lower than that of \( h_\theta^\alpha \) there is a complexity increase operation, after which the complexity levels of the classifiers which both learning systems used is increased. We next give a formal description of the algorithm.

The algorithm maintains two fixed complexity online learning systems \( M_1, M_2 \). The procedure \( M \leftarrow \text{System\_Init}(\beta, M') \) creates a new online learning system \( M \) which operates within the fixed complexity class \( H^\beta \), and which can take advantage of the experience accumulated by the input system \( M' \). At each point \( \alpha \) is the complexity of \( M_1 \) and \( \alpha + 1 \) that of \( M_2 \). The output of the algorithm after each example presentation is represented by the latest classifier produced by \( M_1, h_\theta^\alpha \). The algorithm also maintains
estimates of the error of each classifier. \( \hat{\text{Err}}_{1,2} \) represents the average validation error of classifier produced by learning systems \( M_1 \) and \( M_2 \) respectively, and \( \hat{\text{Var}}_{1,2} \) the variance of the validation errors. Finally, \( N \) counts the number of examples received since the last complexity increase operation. The OMS algorithm receives the sequence of examples \( \langle x^n, y^n \rangle \) and operates as following:

1: \( \alpha \leftarrow 1 \)
2: \( M_1 \leftarrow \text{System}_{\text{Init}}(1, \phi) \)
3: \( M_2 \leftarrow \text{System}_{\text{Init}}(2, \phi) \)
4: \( T \leftarrow 0, \hat{\text{Err}}_1 \leftarrow 0, \hat{\text{Err}}_2 \leftarrow 0, \hat{\text{Var}}_1 \leftarrow 0, \hat{\text{Var}}_2 \leftarrow 0 \)
5: \( h^\alpha_{\theta} \leftarrow M_1(x^1, y^1) \)
6: \( h^{\alpha+1}_{\theta} \leftarrow M_2(x^1, y^1) \)
7: \textbf{for} \( n \geq 2 \) \textbf{do}
8: \( T \leftarrow T + 1 \)
9: \( \text{Loss}_1 = \ell(y^n, h^\alpha_{\theta}(x^n)), \text{Loss}_2 = \ell(y^n, h^{\alpha+1}_{\theta}(x^n)) \)
10: \( \hat{\text{Err}}_{1,2} = \frac{1}{N} \cdot (\hat{\text{Err}}_{1,2} \cdot (N - 1) + \text{Loss}_{1,2}) \)
11: \( \hat{\text{Var}}_{1,2} = \frac{1}{N} \cdot (\hat{\text{Var}}_{1,2} \cdot (N - 1) + (\text{Loss}_{1,2} - \hat{\text{Err}}_{1,2})^2) \)
12: \textbf{if} \( (\hat{\text{Err}}_2 \leq \hat{\text{Err}}_1) \land \) \text{t-test2}(\hat{\text{Err}}_1, \hat{\text{Err}}_2, \hat{\text{Var}}_1, \hat{\text{Var}}_2, N) \) \textbf{then}
13: \( \alpha \leftarrow \alpha + 1 \)
14: \( M_1 \leftarrow M_2 \)
15: \( M_2 \leftarrow \text{System}_{\text{Init}}(\alpha + 1, M_1) \)
16: \( T \leftarrow 0, \hat{\text{Err}}_{1,2} \leftarrow 0, \hat{\text{Var}}_{1,2} \leftarrow 0 \)
17: \textbf{end if}
18: \( h^\alpha_{\theta} \leftarrow M_1(x^1, y^1) \)
19: \( h^{\alpha+1}_{\theta} \leftarrow M_2(x^1, y^1) \)
20: \textbf{end for}

In lines 1,2 the algorithm variables including the two learning systems are initialized. In lines 3,4 the initial classifiers are learned from the first example. The loop 5-15 is performed for each new provided example. Validation on the new example of both classifiers is performed in line 7 computing the loss of the classifiers (\( \text{Loss}_{1,2} \)) on this example. In line 8 the error of each learning system is estimated by the average loss of the classifier over all examples received by its corresponding learning system. The variance of the loss is estimated in line 9 by a similar average, this time over the square
distances from the error estimates. Notice that \( \hat{\text{Err}} \) is not the average loss of the current classifier over all seen examples, since it is computed along the online process in which the classifier keeps on changing. A key point is that the average is over validation loss, that is, over examples that were not included in the training of the classifier, since the training is performed later in lines 16,17. In line 10 we check weather \( h_\theta^{\alpha+1} \) outperforms \( h_\theta^\alpha \) significantly. The significance is tested by running a two sample ”Students t-test” with unequal variance given by:

\[
t = \frac{\hat{\text{Err}}_1 - \hat{\text{Err}}_2}{\sqrt{\frac{(\hat{\text{Var}}_1)^2}{N} + \frac{(\hat{\text{Var}}_2)^2}{N}}}
\]

If the distributions are different with a significance of \( p = 0.1 \) we perform the complexity increase operation (lines 11-14). The use of statistical significance provides stability to the algorithm, preventing unwanted complexity changes, in particular when the number of samples is relatively small and therefore the estimation is noisy. The complexity is increased in line 11, the more complex learning system becomes the current one (line 12) and a new learning system of a higher complexity is initialized utilizing the state of the already trained learning system (line 13). Finally, in lines 16,17 the learning systems are trained on the last received example.

The computational complexity of the algorithm is of the same order as that of the used learning system, since the training and testing of the models (lines 7,16,17) are the most computationally expensive operations. The performance of the algorithm also depends on that of the learning system. The output classifier is the output of some learning system \( M_1 \), therefore the best generalization error obtainable by the learning systems with all possible complexities is a lower bound on the error of our output classifier. There is another crucial requirement for success within the proposed framework, and that is the quality of initialization of higher complexity models from lower complexity ones in line 13. In this initialization the information from the examples already seen by \( M_1 \) must be incorporated by the new learning system in order to take full advantage of the entire set of examples presented to the algorithm. We next present an implementation of our algorithm with an online SVM learning system.
3.2 Kernel Support Vector Machine Implementation

We implemented our algorithm using a kernel-based support vector machine (K-SVM) [61], with two different types of kernels: a Radial Basis Function (RBF) kernel and a Polynomial kernel. The reason for this choice is that these are powerful classifiers with an existing online implementation, the LASVM [5]. The training examples \( x \) are real-valued vectors binary labeled with \( y \in \{-1, 1\} \). Theoretically, the K-SVM classifier performs linear classification in a higher-dimensional space onto which the original vectors are projected. The classification result is obtained by computing:

\[
h^\theta_\alpha(x) = \text{sign}(\sum_{i=1}^{m} w_i \cdot K(v_i, x) + b)
\]

where the model parameters \( \theta \) are: \( \{v_i| i = 1 \ldots r\} \) the support vectors which consist of a subset of examples selected from the training examples, \( w_i \), a weight associated with each support vector and \( b \), a bias term. For the RBF model the kernel function \( K \) is:

\[
K(x, z) = \exp(-\gamma \cdot \| x - z \|)
\]

The Polynomial kernel is defined by:

\[
K(x, z) = (x \cdot z + 1)^d
\]

where \( x \cdot z \) is the dot product of the two vectors.

We represent complexity of the RBF kernel by the band width parameter \( \gamma \). The decision surface of the RBF classifier is obtained by summing gaussian weighted windows around each of the support vectors. For a large band width (small values of \( \gamma \)) the classifier is almost linear, while for a small enough bandwidth (large \( \gamma \)) a dataset can practically be memorized by the classifier, by placing peaked gaussian distributions around each example. Similarly, the degree of the Polynomial kernel \( d \) controls the complexity of the decision boundary obtained by the classifier. For a more detailed discussion on the complexity of these kernel classifiers we refer the reader to [61] pp. 216-218.

In the RBF kernel we use cross-validation on a small initial set to determine the initial complexity level \( \gamma_{INIT} \). We then use a factor of 2 when raising the complexity level:

\[
\gamma(\alpha) = \gamma_{INIT} \cdot 2^{\alpha - 1}
\]
In the Polynomial kernel we simply set \( d = \alpha \). The loss function we use is:

\[
\ell(y, h_\alpha^\theta(x)) = \frac{1}{2} | y - h_\alpha^\theta(x) |
\]

The time and space complexity of the LASVM solver for each new example is linear in the number of support vectors \( m \) and in the feature space dimensionality [5]. In practice it can be made very efficient and was successfully trained on 8 million high dimensional examples [41]. We next present result obtained using this implementation of our algorithm on the MNIST handwritten character recognition set.

### 3.3 Experimental Results

The handwritten character MNIST database consists of 60,000 examples of 10 digits and 10,000 test examples, each example represented by a 28x28 gray level image. We used the raw representation with a 784 dimension vector representing each example. We tested our algorithm on the binary task of separating the digit “3” from the rest of the digits. As a preliminary test we ran the LASVM with several training sets of sizes varying between 1,000 and 60,000 and with different kernel parameter settings. The training examples and their presentation order were chosen at random. The performance of each classifier is represented by its error rate on the test set, shown in figure 3.3. We can see that for each fixed training set size there is an optimal value of the complexity parameter (marked by the black surrounding circle), for which test performance is better compared with a less complex model, as well as a more complex one (overfitting). We also see that for larger datasets the optimal complexity is higher and the optimal error is lower (marked by black arrows). For the RBF kernel with training set sizes 1000, 4000 and 60000, the optimal \( \gamma \) values are 0.01, 0.02 and 0.03 respectively while for polynomial kernel with training set sizes 1000 and 60000, the optimal \( d \) values are 2 and 3 respectively. We will use these values in order to asses our algorithm: assuming that the test set is representative, in the optimal case, our algorithm should choose these values for training sets of similar sizes.

We next present the result obtained with the LASVM implementation of our algorithm (OMS-LASVM). The 60,000 training examples were presented in a random order. We applied cross-validation on the first 100 examples in order to determine the RBF kernel initial bandwidth \( \gamma = 0.04 \). Figure 3.3 shows the error percent obtained on the 10,000 test examples at selected points during the online training. The vertical dotted lines represent the
Figure 3.2: Optimal complexity for different training set sizes. Classification error rate on the 10,000 MNIST test digits of the online SVM solver LASVM for different model complexities and training set sizes. Left: Polynomial kernel, the complexity parameter is the degree \((d)\). Right: RBF Kernel, the complexity parameter is bandwidth parameter \((\gamma)\). Training sizes: 1,000 examples (black curve), 60,000 examples (green curve) and 4,000 examples (red curve - RBF only) The optimal complexity (Marked by black circle) increases for larger datasets as the classification error decreases (black arrows).
Figure 3.3: **OMS-LASVM algorithm.** Performance (Classification error percent on the 10,000 MNIST test digits) as a function of the number of examples ($n$). X-axis: the log number of examples presented to the algorithm during the online learning scheme. Left: Polynomial kernel, Right: RBF kernel. The performance was tested at selected points during the learning (curve markers). Black dotted vertical lines represent the time point at which the complexity increase was performed. The complexity parameter values before and after the change are specified in the text boxes next to the vertical lines. Shown above the lines is the number of examples $n$ received before the change.

Figure 3.3 shows the comparison between the two LASVM models and our model at selected training points by measuring as before their error on the test set. The low complexity LASVM model outperforms the high complexity one when the number of available examples is small, while the opposite holds when more examples become available. We show that our model is able to maintain a comparable performance on both ends of the scale and at intermediate points. Notice that for particular sizes of the training set our method outperforms both fixed models (RBF, second marker representing 2,000 examples). The reason is that the optimal complexity for this dataset size is, discovered by our algorithm, is intermediate, and therefore outperforms both the high and the low complexity models. We also show that for the entire training set our algorithm in this current implementation performs almost as well as the optimal LASVM for that size (RBF: 0.49% instead of 0.42%, Polynomial 0.59% instead of 0.58%), despite the fact that the final classifier has not been *directly* trained on the entire dataset.
3.4 Conclusions

In this chapter we presented a new online learning concept in which the model complexity is increasing as more examples become available. The experiments show that although several approximations are made in order to meet online requirements, the performance remains close to that of an optimal classifier with optimal complexity fit to the test data. The future plans are to use the scheme for object classification tasks. Current object classifier rely on a limited training set and learn a fixed classifier. A classifier can be trained incrementally using a much larger training set and using the suggested scheme adapt its complexity to the training set to learn a more accurate object class representation and improve its performance.
Chapter 4

Learning to detect object parts with multiple appearances
In the training process of classifiers, having more examples enables learning a more complex model. This higher level of complexity can be obtained by substituting fixed parts of the classifier with flexible learned ones. Such a fixed element is the local appearance stage in which the method detects individual object parts in their correct positions in the image. In the current approach the classifier uses a fixed representation of each object part for detecting. Our approach learns this representation specifically for each object part optimizing correct part detection. Most current approaches represent each object part by a single appearance template. In [11] a method was shown for automatically identifying a number of different appearances of an object part and using them for classification. Once the set of different appearances is identified it is used as a fixed representation of the object part. It would be better to learn the representation from examples in such a way that object part detection is optimized. We introduce a part classifier that is trained to correctly detect the object part, just as the object classifier is trained to detect the entire object. The classifier uses a set of representative object part appearances to detect the part in its full range of possible appearance. We introduce a new probabilistic formulation of a model for combining and using the set of part appearance representatives for detecting the object part. This new formulation allows learning the model including the selection of representative appearances and their combination in one optimization using the structural EM algorithm [20].

The chapter is organized as follows. Section 4.1 defines the model and presents the learning algorithm. Section 4.2 describes how the model is used in supervised and unsupervised classification. Section 4.3 explains the fine-registration algorithm used for aligning object part appearance examples. The description of the part detector’s role for unsupervised classification is brought in Section 5.2.2. Section 4.4 presents an the application for supervised classification along with experimental results.

### 4.1 The CNOR model: part detection using multiple-appearances

The task is to detect a specific object part in a given image. The problem we are trying to solve is that different appearances of corresponding parts in different images may vary substantially. To overcome this we model an object part \( m \) as a combination of multiple appearances of the part. Each appearance is represented by an image patch \( Z \). In the training phase we are given a set of image patches of a specific object part \( A_m = \{Z_1^m, \ldots, Z_T^m\} \)
and negative patch examples $B_m$ consisting of either non-object patches or patches of other object parts. In the test phase we obtain an image and have to detect the object part in it if it is present. The task of our algorithm is to select a subset of appearance representatives $R_m \subseteq A_m$, and learn to optimally combine their detection evidence in order to reliably detect the object part. Both tasks are achieved simultaneously by training the CNOR model, depicted in Fig.5.3c. Let $P$ be an arbitrary image patch taken from an arbitrary location $L$ in a new image. The binary variable $O^P$ is set to $O^P = 1$ if $L$ and $P$ are the location and appearance of part $m$ respectively. The probability of $O^P$ is discriminatively modeled as:

$$
\Pr(O^P|V; \Theta) = \sum_Y \Pr(O|Y) \prod_{t=1}^T \Pr(Y_t|V_t; \theta_t, R_m) \quad (4.1)
$$

The $\Theta = \{R_m, \theta_1, \ldots, \theta_T\}$ are the learned parameters of the model. $V = \{V_t\}$, where $V_t$ is the output of a continuous SIFT similarity measure between $P$ and $Z_t \in A_m$. Note that we do not explicitly model $\Pr(V)$, which can be a complex distribution. $Y = \{Y_t\}$, where $Y_t$ is a latent binary variable representing the detection of appearance $Z_t$ with:

$$
\Pr(Y_t = 1|V_t; \theta_t, R_m) = \begin{cases} 
\frac{1}{1+e^{-\alpha_t(V_t-\tau_t)}} & Z_t \in R_m \\
0 & Z_t \in A_m \setminus R_m 
\end{cases} \quad (4.2)
$$

Here $\theta_t = \{\tau_t, \alpha_t\}$ are the parameters of the sigmoid in 4.2. $Y_t = 1$ becomes likely if patch $P$ exceeds a similarity threshold $\tau_t$ with the representative patch $Z_t \in R_m$, with $\alpha_t$ representing the uncertainty of $\tau_t$. If $Z_t \in A_m \setminus R_m$ (meaning $Z_t$ is not a chosen representative) then $V_t$ and $Y_t$ have no effect on $\Pr(O^P|V; \Theta)$. Finally, $\Pr(O^P|Y)$ is a deterministic "or" of $Y$:

$$
\Pr(O^P = 1|Y) = \begin{cases} 
1 & \exists_t Y_t = 1 \\
0 & \text{otherwise} 
\end{cases} \quad (4.3)
$$

The entire model can intuitively be described as follows: the part is detected ($O_p = 1$) whenever $P$ is "sufficiently" similar to at least one of the part $m$’s representative patches in $R_m$.

To limit the number of representatives, the learning objective is to find the Minimum Description Length (MDL) [33] parameters $\Theta$, in other words, to find $\Theta$ that maximize a combined score of the model complexity (number of representatives) and model performance (data likelihood). We solve this learning problem using the Structural EM (SEM) algorithm optimizing the Bayesian Information Criterion (BIC) score [20]:

47
\[
BIC = \sum_{P \in E} \log(Pr(O^P|V^P; \Theta)) - \frac{\log T}{2} \cdot |R_m| \quad (4.4)
\]

The SEM algorithm iterates between two stages. The first stage is, given a set of representatives \( R_m \), to find the optimal values for the \( \{\theta_t\} \) parameters. This stage is solved using the EM algorithm. It is computed efficiently, since each iteration of the EM can be performed in linear time in our model using the method in [75]. The update equations for the CNOR model are brought in appendix 7.6. The second stage is, given the current assignment of \( \{\theta_t\} \), to estimate an improved \( R_m \). This is achieved by running several iterations of a greedy search over subsets of \( R_m \subseteq A_m \), where at every step of the search a current subset \( R_m \) is modified by either adding or removing one element.

Finally, The learned model is then used to assign to a given unseen patch \( P \) a probability of being a part example.

### 4.2 Learned part models and object classification

There are more than one possible ways to incorporate the suggested model within an object classifier learning and detection scheme. In particular, an important choice is how to obtain the positive training appearance patch examples \( A_m \) and the negative examples \( B_m \). We have explored the part learning scheme in two different overall classification schemes. The first, described in section 4.4, is within a supervised scheme for classification. For each object fragment a set of similar fragments is found in the class examples and are treated as part representatives \( A_m \). This approach relies on the appearance similarity for finding different appearances of an object part. This method however will not group together substantially different appearances of the same object part. To overcome this we use the position on the object with relation to a known object position to extract the set of part appearances \( A_m \) from a geometric point of view. This approach is used in an unsupervised classification scheme described in detail in Chapter 5. Consider a set of image examples \( H \) in which an object reference point \( \mu_1^n \) is known for each image \( I_n \in H \). For each object part we take the 40x40 image patch at position \( \mu_1^n + \rho_1^n \), where \( \rho_1^n \) is a fixed offset of the part. The accumulated set of image patches is the candidate set of part appearances, \( A_m = \{Z_1^m, \ldots, Z_T^m\} \), is an initial candidate set of part appearances. To compensate for local object deformations, the patches in are jointly aligned as follows. We iteratively align the most similar pair of patches that are not already brought into alignment (either directly, or through a path of
alignments between them). Section 4.3 further describes the fine alignment algorithm in more detail. Figure 5.4 shows an example of a learned detector and its performance. For example, in horse detection task the equal error rate using the generic feature set was 17% (iteration 1 of the algorithm) while using the same classifier but with the CNOR part detectors as features it reduced to only 2.7% (iteration 2 of the algorithm). The part detectors were learned successfully for several different object categories from the Caltech-101 object database. For further details on the unsupervised classification scheme and the role of the CNOR part detector in it please refer to Chapter 5. In the next section I describe the algorithm for fine registration of the candidate appearance patches.

4.3 Fine-Alignment of patches of corresponding object parts

In this section we describe an algorithm for fine-registration of semantically equivalent image patches. The problem we are trying to solve is the following. Given an algorithm that identifies a corresponding object anchor point (e.g. the tip of the nose) in several images of the object, one can obtain a rough estimate of the position of another corresponding point (e.g. middle of the chin) in each of the images by computing a fixed relative translation from the known corresponding point. Due to object deformations the estimate of the new corresponding points is rough. We cut an image fragment around each new corresponding point to obtain a set of roughly aligned initial fragments $A^{\text{init}}_m$ which capture more or less the same object part. The task of the fine-alignment algorithm is to jointly align these fragments in one coordinate system in order to obtain a set of fragments $A_m$ that reflect the same object part in a more accurate way (semantically equivalent fragments). The method described here uses the information from the fragment appearance for achieving this fine alignment, in addition to the initial rough guess obtained by the geometric information as described above. Two patches of the same object part may be entirely different in appearance, and therefore difficult to align directly. To overcome this difficulty, we exploit the fact that $A^{\text{init}}_m$ often contains intermediate examples and use the concatenation of the easier intermediate alignments to achieve the alignment between two different patches.

The algorithm receives as input a set of roughly aligned fragments $A^{\text{init}}_m = \{Z_1, \ldots, Z_T\}$, of the same height and width $(H, W)$ and the absolute image position of their centers $\{X_1, \ldots, X_T\}$. Around each fragment $Z_i$ a larger
context patch $\hat{Z}_i$ is extracted of double the size $(2H, 2W)$ and centered at the same position $X_i$. I next describe how relative alignment is achieved between two image patches $Z_i$, $Z_j$. We compute the SIFT descriptor of each fragment of size $(H, W)$ within $\hat{Z}_j$ and compute its $L_2$-distance with the SIFT descriptor of $Z_i$. We find the minimal distance $D'_{i,j}$ and mark the central position of the patch with the minimal distance by $R'_{i \rightarrow j}$ (this position is relative to the origin set at $X_i$). Similarly we compute $D'_{j,i}$ and $R'_{j \rightarrow i}$. We now set the relative alignment of the patches $R_{i \rightarrow j}$ and $R_{j \rightarrow i}$ as following. If $D'_{i,j} > D'_{j,i}$ we set $R_{i \rightarrow j} = R'_{i \rightarrow j}$ and $R_{j \rightarrow i} = -R'_{i \rightarrow j}$. Similarly in the opposite case of the condition we set $R_{i \rightarrow j} = -R'_{j \rightarrow i}$ and $R_{j \rightarrow i} = R'_{j \rightarrow i}$. We also set the confidence we have in the alignment as the minimal distance $D_{i,j} \equiv D'_{j,i} \equiv \min(D'_{i,j}, D'_{j,i})$. During the alignment process a position $X'_i$ is found for the object part in each image $i$, and the corresponding object part fragment $Z'_i$ is obtained by extracting the fragment of dimensions $(H, W)$ centered at $X'_i$ (we denote this operation by $\text{Frag}(X'_i)$). Suppose we already computed the position $X'_i$. Then the direct alignment operation sets the position of part $j$ by:

$$X'_j = \text{DirectAlign}(i, j) = X'_i + R_{i \rightarrow j}$$

We define a full undirected graph of pair-wise alignments:

$$G = (V = A_m^{\text{init}}, E = (A_m^{\text{init}} \times A_m^{\text{init}}), W = D)$$

assigning each edge $(Z_i, Z_j)$ a weight equal to the alignment confidence $D_{i,j}$. We then find the minimal spanning tree of $G$. The paths in this tree represent chains of alignment with minimal direct alignment errors ($D$), with the condition that there is a single path connecting every two fragments. The alignment is finally achieved as following. We set $X'_1 = X_1$, and perform a Breath First Search (BFS) on the minimal spanning tree starting from the first vertex $Z_1$. Each time node $Z_j$ is discovered in the BFS by node $Z_i$ we set $X'_j = \text{DirectAlign}(i, j)$. Finally for each fragment we set $Z'_i = \text{Frag}(X'_i)$. The output is the set of fine-aligned object part fragments $A_m = \{Z'_1, \ldots, Z'_T\}$.

### 4.4 Implementation within supervised bag-of-words classification

Another application of the model was to supervised bag-of-words classification. The difficulty compared to the previous implementation is that since
geometry is not explicitly modeled, we cannot extract different appearances of the same object part based on its geometric relation with other known part positions. In this case the multiple appearances do not represent entirely different part types (e.g. closed eye and open eye) but rather slight variations in the parts appearance (e.g. illumination variations).

In the fragment-based classification scheme [68] a set of informative fragments $F$ is extracted from the training class images, and used as binary classification features in a Naive-Bayes classification scheme. We improved the classification features using the CNOR model, by improving the reliability of the detection of each individual fragment. Each fragment was extended to a cluster of similar fragments, which were then combined using the CNOR model. The different clusters were then used for classification in a Naive-Bayes classification scheme. Let $m \in F$ be an informative fragment. An initial set of candidate fragments ($A_m$) is extracted, consisting of the most similar position in each of the class training images. Similarly, the negative examples ($B_m$) are created by taking the most similar fragments in the non-class images. Since the initial set $A_m$ may contain false detections of the original fragment $m$ (outliers), we discard the fragments with low similarity with the original fragment $m$, with the threshold determined by the similarity with fragments in ($B_m$). In this way we limit us to a set ($A_m$) which is close to the original fragment $m$. $A_m$ consists therefore of slight variations of the same part appearance, represented by the original fragment $m$. $A_m$ and $B_m$ are used as described above to train the CNOR model. This time the training is on class vs. non-class image examples, and the result is a part appearance detector which separates part examples from non-part examples more reliably. This method for extending each fragment may also be viewed as a multiple mode approach to fragment detection: the space of descriptors is broken into several modes each represented by a region around a mode peak - which in our case is the descriptor of a representative appearance.

The cow image dataset consists of 316 cow images and 450 non-class images, of which 100 class and 100 non-class exemplars were used for training. The similarity measure used was a combination of SIFT descriptor and $L_2$ norm. We set $|F| = 100$. Each fragment was extended to a cluster of appearances as explained above. The union of all the resulting clusters comprised of 531 fragments in total (averaging 5.31 fragments in each cluster of similar fragments). Figure 4.1 shows the clusters created for 20 representative fragments. Figure 4.2 shows the classification result ROC with the original 100 fragments versus using the proposed method. The equal error rate of the entire classifier is improved from 7.5% to 4.5%, and for low false alarm rate (2%) the improvement is as high as 11%. We also com-
pared the class information of each new cluster compared with the original fragments. Figure 4.3 shows the difference in information between the new part variable $O_m$ and the original fragment $m$. For each fragment we obtain on average a 25% increase in information. I also examined the learning of the parameters and structure. In the parameters case we can examine the difference between the globally learned thresholds (In this model’s case it is only global with respect to the scope of one cluster, but still local with respect to the entire classifier), and the standard thresholds earned on the same training set using maximal mutual information criterion. Figure 4.4 shows that in all cases the new thresholds are higher, as we would expect since we are using an OR model. Finally, an important question is what causes the improvement in classification. It may be the thresholds which are learned differently, it may be simply that we are using more fragments, or it may be the entire new structure of the graphical model. Figure 4.6 shows a comparison of performance obtained using the standard method, the proposed one and two more variants of the method. The first is the standard FBC method with allowed to use the 531 fragments found by the CNOR or only a part of them (number of features tested was 31 to 531 with steps of 50, best results obtained with 331 - Fig. 4.6 black curve). The second variant uses the same scheme as the first one, but instead of using the thresholds found by the FBC method, it uses the thresholds learned by the proposed method (Fig. 4.6 yellow curve). The increasing improvement of each variant plus the superior improvement of the full scheme suggest that each of the examined parts of the method (i.e. feature selection, threshold learning, and combination function) contribute to the final result.

4.5 Conclusions

I presented a new object part classifier which uses a set of representative object part appearances to detect the part in its full range of possible appearances. We introduced a novel probabilistic model, the continuous noisy-OR (CNOR), which is unique in its ability to perform both feature selection and parameter learning in one optimization. This optimization leads to a significant improvement in classifying and localizing individual object parts, leading to an overall improved classification performance. The CNOR model may also be applied to a wider range of machine learning problems in which the classification features are continuous, noisy, and not all are relevant to the target function in cases in which the target function is an OR-like combination of the features.
Figure 4.1: **Examples of learned clusters of cow fragments.** Clusters of similar fragments learned using the continuous Noisy OR. Clusters contain between 3 and 9 prototype fragments. We see that frequently the prototype fragments in each cluster are visually different but represent semantically equivalent parts of the cow.
Figure 4.2: Classification performance using clusters of fragments (cow classification). The graph shows the ROC curves in the cow classification task for the Naive-Bayes model with the most informative 100 fragments (dotted red curve), and for the 100 clusters of fragments learned using the continuous noisy-OR model, which uses a total amount of 531 fragments (solid blue curve).
Figure 4.3: **Clusters of fragments: information of individual parts.** The graph shows the class information given by each of the 100 part detectors in the cow classification task. The blue curve represents their information using clusters to detect the parts (and also defines their order in this graph), while the red curve represents the information obtained using the original single fragments. The clusters are clearly more informative, with an average gain of 25% in information over the original method.)
Figure 4.4: **Clusters of fragments: thresholds of individual fragments.** We show here how the different methods of learning the thresholds produce different thresholds. The graph shows the learned thresholds for the entire set of 531 cow fragments used in the cluster model, when learned within the CNOR model (blue curve), and when learned by maximizing the information of each fragment individually (red curve). The thresholds are much higher in the first case, providing a more reliable detection of each fragment.)
Figure 4.5: Clusters of fragments: learning curve in structure learning. The graph shows a representative run of the structure learning algorithm we presented for the CNOR model. The plot shows the BIC score, which encapsulates the log likelihood score and the structure complexity score, for each iteration in the structure learning algorithm.)
Figure 4.6: Clusters of fragments: full classifier performance analysis with different settings. This graph plots the ROC curves for the cluster method, and for intermediate methods, in order to identify which parts of the model are responsible for the improved performance. The basic method uses the original 100 fragments in the standard FBC model (dotted red), the extended method makes use of the entire set of 531 fragments, but combined in the same way (solid black), improved by taking the CNOR learned thresholds (dotted yellow), and further improved by using the full cluster model (solid blue). This experiment implies that the entire improvement is not solely due to the larger set of fragments of the improved detection thresholds.
Chapter 5

Unsupervised classification and part localization by consistency amplification
This chapter presents a novel method for unsupervised classification, including the discovery of a new category and precise object and part localization, developed in collaboration with Michael Dinerstein and Leonid Karlinsky. Given a set of unlabelled images, some of which contain an object of an unknown category, with unknown location and unknown size relative to the background, the method automatically identifies the images that contain the objects, localizes them and their parts, and reliably learns their appearance and geometry for subsequent classification. Current unsupervised methods construct classifiers based on a fixed set of initial features. Instead, we propose a new approach which iteratively extracts new features and re-learns the induced classifier, improving class vs. non-class separation at each iteration. We develop two main tools that allow this iterative combined search. The first is a novel star-like model capable of learning a geometric class representation in the unsupervised setting, developed by L. Karlinsky and M. Dinerstein. The second is learning of “part specific features” that are optimized for parts detection, and which optimally combine different part appearances discovered in the training examples. These novel aspects lead to precise part localization and to improvement in overall classification performance compared with previous methods. We applied our method to multiple object classes from Caltech-101, UIUC and a sub-classification problem from PASCAL. The obtained results are comparable to state-of-the-art supervised classification techniques and superior to state-of-the-art unsupervised approaches previously applied to the same image sets. The rest of the Chapter is organized as follows. Section 5.1 defines the unsupervised classification learning task and presents our new proposed approach contrasted with the previous one. Section 5.2 presents an overview followed by a detailed description of each of the method stages. Section 5.3 presents results obtained on various datasets, together with an analysis and comparison with previously reported results. Conclusions are discussed in Section 5.4.

5.1 A new approach to the unsupervised class learning task

The goal of the proposed approach is unsupervised classification, including discovery of a new category, learning a model of geometric arrangement of object parts and their appearance, and obtaining object and part localization, from a set of unlabeled images, which contains non-class images mixed with some unknown (usually small) percent of class images. The
class instances may be uncropped, unaligned and of small size relative to the background.

The problem of unsupervised object classification has gained considerable recent interest [21, 63, 16, 17, 15, 58, 6, 47, 36, 1, 38, 39], however, this task is still far from being completely solved. In this study we present a novel methodology to approach the problem. A common approach is to start from some limited, manageable set of initial features $F$, for example, a set of local descriptors extracted around image interest points or clusters extracted from such descriptors [21, 63, 38, 39, 34, 53, 40, 64, 15, 58, 6, 47, 36]. The set of features can be optimized by selecting a subset of the most useful features $F_1 \subset F$, or sometimes combinations of features in $F_1$ are used as new features [63, 47, 53]. However, there is no guarantee that the choice of initial features will in general be sufficient for complete separation. In contrast, we approach the problem as a combined iterative search for features and a classifier. We do not use the initial feature set to obtain the final class separation, but only for identifying a subset of sure class examples which can be reliably separated from the rest (Fig. 5.1a). This goal is achieved by unsupervised training of a classifier that combines both appearance and part-geometry information. The extracted class examples are
Figure 5.2: (a) Example results of unsupervised object and part localization on two datasets (UIUC cars, flamingo). The yellow star is the detected model center location (see text), color coded rectangles are examples of detected object parts (for each object several out of about 150 modeled parts are shown). (b) Schematic diagram of the UCA algorithm.

We develop two main tools that allow the iterative combined search. One is the incremental discovery of part specific features, which combine different part appearances discovered in the training examples. The other is a novel star-like class-geometry model of object parts, which differs from the similar past models [16, 17, 15, 40, 64, 7, 34] and which can be learned efficiently without supervision in very noisy conditions, developed by L. Karlinsky and M. Dinerstein. These two aspects are described briefly below, and explained in more detail in Sections 5.2.1 and 5.2.2.

**Feature learning:** most unsupervised approaches [21, 63, 15, 58, 6, 47, 36, 38, 39], including ours, start from some generic set of features $\mathcal{F}$. During learning, when a particular class is considered, the approaches select a subset $\mathcal{F}_1 \subset \mathcal{F}$ of so-called Class Specific Features (CSF), which coincide better
with the class compared with the background or other classes. In contrast, our method extracts and learns a new set of features, termed Part Specific Features (PSF). The PSF are optimized to have higher detection scores at specific locations on the class objects, and at the same time to have lower scores at incorrect locations on the same objects and in non-object detections. Different part specific features have been used successfully in a number of supervised approaches, such as k-fan [7] and semantic hierarchy [12], and were shown to be useful for both object and part localization. Constructing such features in an unsupervised manner is challenging; our method is the first unsupervised method that learns and uses such features, resulting in improved object and part detection and localization.

**Geometry learning:** Past supervised and unsupervised classification methods can be categorized by their modeling of object geometry. In bag-of-feature methods [63, 36], geometry is ignored. Methods, such as [63, 47, 21, 53], extend the bag-of-feature approach by using feature combinations. In [16, 17, 15, 38, 34, 40, 64, 7] object geometry is modeled by the spatial distribution of each feature in the object reference frame. A geometric part model is useful for classification, but it is challenging to construct such a model in an unsupervised setting. Most previous unsupervised methods therefore do not use a full geometric model [63, 58, 6, 47, 36, 38, 39]. Our method uses star-like geometry. It has several differences compared with similar past models. The method is not restricted by a small number of parts as in [16, 17], unlike [34, 40, 64, 7] it does not require any supervision, unlike [16, 15, 40, 38] it models distribution of feature locations on the background, unlike [15, 39] it does not rely on non-geometric pLSA [63] for internal supervision, and unlike [1, 58, 6], it is not based on prior image segmentation. These differences are explained in more detail in Sections 5.2 and 5.2.1.

In terms of class vs. background classification performance, our method outperforms the state-of-the-art unsupervised methods [63, 15, 6, 1, 38, 39] on 18 classes from the Caltech 101, Weizmann horses and UIUC cars datasets. Surprisingly, the method is also comparable in performance to existing state-of-the-art supervised (and weakly supervised) methods applied to the same datasets. We further demonstrate how our method can be used to separate different object views on the cars class from the PASCAL challenge 2007 dataset. As the method achieves precise object and part localization, it provides a basis for top-down segmentation.
5.2 The consistency amplification method

Our approach alternates between model learning and data partitioning. Given an image set $S$, an initial model (learned using initial features) is used to induce an initial partitioning by identifying highly likely class members. The initial partitioning is then used to improve both the appearance and geometrical aspects of the model, and the process is iterated. In this manner the process exploits intermediate classification results at a given stage to guide the next stage. Each stage leads to an improved consistency between the detected features and the model, which is why the process is termed Unsupervised Consistency Amplification (UCA). Each UCA iteration consists of two phases of learning: the feature learning Appearance-phase (A-phase) followed by the part model learning Geometry-phase (G-phase), explained in detail in sections 5.2.2 and 5.2.1 respectively. The approach and the order of the phases are summarized in Fig. 5.2b.

**Initial Appearance-phase:** In all our experiments, we use a generic codebook of SIFT descriptors of $40 \times 40$ patches for the initial (appearance) features. This codebook, denoted by $\mathcal{F}_0$, is computed by a standard technique [27] from all the images in given set $S$. The codebook descriptors are compared to the descriptors at all points of all the images in $S$ and storing the points of maximal similarity (either one or several, see below) in each image.

**Geometry-phase:** The detection of parts using the generic features is usually noisy, due to detections in non-class images, and at some incorrect locations in the class images. The goal of the geometric part model learning is to distinguish between the correct and incorrect detections, based on consistent geometric relations between features. This is accomplished by the G-phase of the algorithm, which is also used for the selection of the most useful features and the automatic assignment of each of their detections in every image in $S$ to either object or background model. In contrast with [16, 17, 15, 40, 38] that use uniform distribution of features on the background, we model the background by a distribution of the same family as the class object distribution, which allows to prevent the spurious geometric background consistency from being accounted for by the learned class model. In our experiments we found that modeling the background distribution is better than assuming uniformity with mean performance gain of $12 \pm 7\%$ EER in the first iteration of the UCA that uses initial generic appearance features. The learned background model is then discarded after the learning and is not used for classifying new images. Thus, the model used in the G-phase is a mixture of two stars, one for object and the other
for background. It is learned without supervision from all the images in $S$ using a novel graphical model formulation explained in detail in Section 5.2.1. After the geometric structure has been learned, a subset $H \subset S$ of images which contain class objects with high confidence is selected. In these images the object centers and parts are localized. Unlike [15, 39] that learn the geometric constraints using only a set of objects identified by the non-geometric pLSA [63], our method identifies and localizes objects, and learns their part geometry, jointly and explicitly from the entire data.

**Appearance-phase:** Each part-specific feature constructed in the A-phase represents an object part by extracting several typical appearance patches of the part, from different images. Part-patches can be extracted, because the locations of the parts in the images of the subset $H$ are already estimated from the previous G-phase. An optimal subset of these part patches is learned by a discriminative model described in section 5.2.2. The set of all part specific features extracted during the A-phase is denoted by $F$.

**Computing the output:** After the G-phase at each iteration, the learned model is applied to produce classification, as well as object and part localization results for either the given dataset or an unseen test set. This is done without introducing any supervision to the system. The way we apply the learned model to test images is described in detail in section 5.2.3.

### 5.2.1 The geometry phase

We first describe the G-phase model, and then explain how it is learned from the data. This model was solely developed by L. Karlinksy and M. Dinerstein and is brought here for the clarity of the presentation. The main goal of the G-phase is to identify the most likely locations of objects and their parts in all images of the given set $S$ and to estimate a subset $H \subset S$ of images which contain class objects with high confidence. The G-phase models the data by a generative probabilistic graphical model depicted in Fig. 5.3a. Let the image set $S$ have $N$ unlabelled images: $S = \{I_1, I_2, \ldots, I_N\}$ and the current feature set $F$ consist of $M$ features: $F = \{F_1, F_2, \ldots, F_M\}$. In the G-phase of the first UCA iteration these features are a codebook of generic SIFT descriptors $F_0$, and in the following UCA iterations these are the learned PSFs. During the G-phase each feature is associated with an object part or the background. Denote the detected location of feature $F_m$ in image $I_n$ by $X_{nm}^p$ (the G-phase uses a single (maximal) detected location per feature in each image, see extension below.) The G-phase model independently generates observed samples: $\text{Data} = \{(F_m, I_n, X_{nm}) \mid 1 \leq n \leq N, 1 \leq m \leq M\}$

The probability of observing a specific image $\Pr(I = I_n)$ is taken to be uni-
form. The overall observed data likelihood under the G-phase model can be written as:

$$
\Pr(Data) \propto \prod_{n=1}^{N} \prod_{m=1}^{M} \sum_{C_{nm}^{n}=1}^{2} \int \Pr(L_{nm}^{n} | I_{n}, C_{nm}^{n}) \Pr(L_{F} = X_{m}^{n} | F_{m}, C_{nm}^{n}, L_{nm}^{n}) dL_{nm}^{n}
$$

(5.1)

The meaning of the product inside the integral in eq. 5.1 is that each data sample \((F_{m}, I_{n}, X_{m}^{n})\) observed in image \(I_{n}\) for the feature \(F_{m}\) is independently generated as follows. First, the latent discrete binary “class” variable \(C_{nm}^{n}\) is drawn with probability \(\Pr(C_{nm}^{n} = k | I_{n}) = \alpha_{n}^{k}\), independent of the feature \(F_{m}\). \(C_{nm}^{n} = 1\) means that \(I_{n}\) contains a class object and \(F_{m}\) is generated from the class model. \(C_{nm}^{n} = 2\) means that \(F_{m}\) is generated from the background model, because either \(I_{n}\) does not contain an object or \(F_{m}\) was not detected consistently with the class model. After learning, the value \(\alpha_{n}^{k}\) is the likelihood of class \(k\) (either object or background) in image \(I_{n}\). Next, the latent location variable \(L_{nm}^{n}\) is drawn from a Gaussian distribution \(\Pr(L_{nm}^{n} | I_{n}, C_{nm}^{n} = k) = N(\mu_{k}^{n}, \Sigma_{k}^{n})\). \(L_{nm}^{n}\) represents the image position of the center of the star model (chosen by \(C_{nm}^{n}\)), which generates the feature \(F_{m}\) in image \(I_{n}\). Note that for every feature detected in image \(I_{n}\) that has chosen the class \(k\), there is a separate variable \(L_{nm}^{n}\), but all of these variables are generated from the same distribution specific to \(I_{n}\). Next, the observed feature variable \(F\) draws its value \(F_{m}\) from the distribution \(\Pr(F = F_{m} | C_{nm}^{n} = k) = \beta_{k}^{m}\) which depends on the chosen class \(k\), but is independent of the image \(I_{n}\). After learning, the value \(\beta_{k}^{m}\) is the likelihood of feature \(F_{m}\) to be consistent with the geometric model of class \(k\). Finally, the observed feature location variable \(L_{F}\) draws its value \(X_{m}^{n}\) from a linear Gaussian distribution \(\Pr(L_{F} = X_{m}^{n} | F_{m}, C_{nm}^{n} = k, L_{nm}^{n}) = N(L_{nm}^{n} + \rho_{k}^{m}, \Lambda_{k}^{m})\). This distribution models the uncertainty of the offset \(\rho_{k}^{m}\) of the feature \(F_{m}\) from the \(L_{nm}^{n}\) - center of the star model chosen by \(C_{nm}^{n}\). It is specific to the feature \(F_{m}\) and the chosen class \(k\) and is independent of the specific image \(I_{n}\).

To summarize, the parameters of the model are \(\alpha, \beta, \mu, \Sigma, \rho\) and \(\Lambda\), all of them are learned by soft EM as described further below. A schematic drawing illustrating the data generation process and the meaning of the main model parameters is shown in Fig. 5.3b. The model uses a star-like geometry, but an important difference between the current model and past star model formulations is worth noting. In contrast with [7, 64, 40], that have a single reference point or k-fan per image, in our model there exists a separate reference point (center) random variable for each part,
Figure 5.3: The probabilistic models used by UCA. Shaded ellipses are observed variables, unfilled are hidden (latent). (a) Graphical representation of the G-phase generative model. (b) Generating the object and background. The object model illustrated in red, the background in green. Each generates centers, denoted by ★ (star) and features denoted by ▲ (triangle). The ellipses denote the uncertainty in position. \( X_n^m \) denotes the detected location of feature \( F_m \). Every feature \( F_m \) detected on the object is generated using its own star center point \( L_n^m \), but all these \( L_n^m \) are generated from the same distribution \( N(\mu_{1n}^m, \Sigma_{1n}^m) \) specific to the image. As illustrated, the learned object distributions are tighter than the background distributions.

(c) Graphical representation of the “Continuous Noisy OR” discriminative model.

drawn, however, from the same distribution specific to the given image. This allows the features detected in the same image to be updated individually: features assigned to the class update the class star and features assigned to the background update the background star, both the assignments and the updates are soft. Although it may sound technical, it has fundamental importance, since, as we saw in our experiments, in different class images, different subsets of features are geometrically consistent with the object model. It is interesting to note that the transition from the standard star-model to our version is entirely analogous to the transition from Naive Bayes (NB) to pLSA. In the NB there is only a single class node generating the entire feature vector of an image, while in pLSA each feature has a separate topic node generated from an image specific distribution. The pLSA is more flexible than NB and was found useful for unsupervised classification [63, 36]. Similarly we found that the modified star is useful in modeling feature geometry in the unsupervised setting.
Learning: The model is learned from the data using the soft EM algorithm. The EM update equations are provided in the supplementary material. As mentioned above, the data samples fitted by our model are of the form \((F_m, I_n, X^n_m)\). In order to incorporate the features’ detection scores into the learning process, we weight each sample by its score. Namely, the sample \((F_m, I_n, X^n_m)\) is weighted by \(R^n_m\) - the similarity of \(F_m\) with \(I_n\) at location \(X^n_m\). The parameters of the model: \(\alpha, \beta, \mu, \rho\) and \(\Lambda\), are initialized at random and EM is run until convergence. In order to have the object model learned with respect to \(C^n_m = 1\), in all our experiments, we initialize \(|\Sigma^n_1| << |\Sigma^n_2|\) for every \(n\). During EM iterations, this initialization causes the object feature detections to tend to update the \(C^n_m = 1\) model, since they usually appear in more tight and repeatable configurations (i.e. fit a star with smaller center uncertainty \(\Sigma^n_1\)). At the same time, background feature detections, that are usually loosely scattered all over the image, will tend to coincide with the \(C^n_m = 2\) model. This method of the initialization of all the parameters (including \(\Sigma\)) was identical throughout all our experiments.

Identifying the set of high-likelihood images: After the EM converges, the value of \(\alpha^n_1\) (the image probabilities to belong to the class) shows a strong separation between a subset of class images and all the non-class images, see figure 5.6a. As a result, by the end of G-phase it becomes possible to identify a subset of high-likelihood class candidate images \(H\). In all our experiments, we marked an image \(I_n\) as high likelihood class candidate if \(\alpha^n_1 > \eta \cdot \max_n(\alpha^n_1)\), where \(\eta = 0.85\) was chosen empirically and used throughout all the experiments. Examples of objects automatically identified and localized by the G-phase of the first UCA iteration are shown in Fig. 5.6b. These are examples of the first G-phase output, obtained using the initial generic features. As can be seen, the localized object model centers appear at similar locations within the object in the different images. In the A-phase, these points are used to extract stacks of corresponding fragments, which are used to construct the part specific features - the CNOR part-detectors, as explained in the next section.

5.2.2 The appearance phase

In the G-phase we learned the position of each part relative to the object model center, and detected this center in the images belonging to \(H\). We localize each part in these images by assuming it is located at the learned relative position \(\rho^n_m\) from the center located at \(\mu^n_1\). In the A-phase we learn for each part a detector trained to distinguish image patches in correct part locations from patches in incorrect ones. The detector is trained using the
detected part locations as positive examples and all other locations on the
images of $H$ as negative examples. The constructed part detectors form
the new feature set $F$ for the G-phase of subsequent UCA iteration. We
next describe the novel probabilistic discriminative model used by the part
detector, the Continuous Noisy OR (CNOR), and how this model is trained.

For each part $m$, corresponding to $F_m$ above, we extract a set of appear-
ances in the following way. In each class candidate image $I_n \in H$, we take
the 40x40 image patch at position $\mu_n^m + \rho_n^m$, where $\mu_n^m$ is the location of the
learned object center in $I_n$ and $\rho_n^m$ is the learned offset of part $m$ from the
object center. The accumulated set of image patches is the candidate set of
part appearances: $A_m = \{Z_1^m, \ldots, Z_T^m\}$.

The initial candidate set part appearances is further improved using a
fine alignment algorithm. To compensate for local object deformations, the
patches in are jointly aligned as follows. We iteratively align the most similar	pair of patches that are not already brought into alignment (either directly,
or through a path of alignments between them). Section 4.3 describes in
detail the fine alignment algorithm. The result of the fine-alignment is a set of patches of the object part which replaces the \( A_m \) set described above.

The next step is to select a subset of appearance representatives \( R_m \subseteq A_m \), and learn to optimally combine their detection evidence in order to reliably detect the object part. Both tasks are achieved simultaneously by training the Continuous Noisy-OR (CNOR) model described in detail in 4.1.

To learn the CNOR model parameters, a training set of image patches \( E \) is constructed by taking all \( 40 \times 40 \) image patches (on a fixed step grid) from all the images in the current class candidate set \( H \). For each patch \( P \in E \) the observed data vector is constructed as: \( D^P = \langle V^P, O^P \rangle \) where \( V^P = \{ V^P_t \} \) is computed by measuring similarity between \( P \) and \( A_m \) patches and \( O^P = 1 \) iff \( P \in A_m \) (and \( O^P = 0 \) otherwise). Finally the training data for the CNOR model of part \( m \) is: \( D = \{ D^P | P \in E \} \). By treating the correct part appearances (\( A_m \)) as positive examples and all other appearances (either other parts of the object or background patches) as negative examples, the object part detector is trained for correct localization of the part. This unique choice of the training examples results in a detector optimized for detecting only the correct location of the object part, as opposed to other schemes, in which an "object-feature" may be found in multiple object locations.

After learning, the set of part \( m \) detections is obtained by identifying first few local maxima of the probability \( \Pr(O^P|V; \Theta) \) computed for all patches \( P \) in a given image. Selected representatives for an example part are shown in Fig. 5.4a. The resulting CNOR part detectors for all parts \( m \), are a significantly more reliable set of object features then the initial generic set of features, as demonstrated in Fig. 5.4b and in section 5.3, and it provides a general method for reliable part detection for both supervised and unsupervised classification.

5.2.3 Applying the learned model to classify and localize objects and parts

To compute the classification score for an unseen test image, and to localize the class objects in it, we use the learned \( \rho^m_1 \) (offsets from object star center) and \( \Lambda^m_1 \) (STDs for these offsets) parameters in a voting scheme similar to [34] as follows. For each part detector, a number (five in our experiments) of highest-scoring locations at each image are marked. To identify the object star’s center location, each detection \( X \) votes for a center location, by placing a Gaussian mask with STD \( \Lambda^m_1 \) around the expected location \( X - \rho^m_1 \). After all the detectors voted, the point with the maximal accumulated vote determines the location of the object star’s center in each
image (in case there are multiple objects, several local maxima that exceed a global threshold are taken). The accumulated vote value at the detected center point serves as the object detection score in the image. These scores are then used to create the ROC that tests the separation between the class and the non-class images in the results section 5.3. The object localization results of our method are evaluated in Fig. 5.4c. The parts are localized by "back-projection" as in [34]. Each part detector that voted into one of the selected object center locations (with one of its five detections) is declared as 'detected' and is marked in the image. The accuracy of our part localization is demonstrated in figures 5.2a and 5.5 and evaluated in Fig. 5.4d. Marking all the detected parts in the image can be used for a top-down segmentation of the detected object (see examples in supplementary material). The details of the top-down segmentation are outside the scope of the current discussion.

5.3 Results

To test the performance of the UCA method, it was applied to the task of fully unsupervised classification and object and part localization on 18 different object classes. The list of the classes, the parameters of the datasets and ROC EERs obtained by the UCA are summarized in Table 5.1. The results show that our method obtains superior performance over the existing unsupervised methods in challenging conditions such as small objects relative to the background (e.g. UIUC cars, Caltech101 cars, flamingo), small percent of class images in the set (e.g. schooner, guitars), significant inter-class variability due to non-rigid deformations (e.g. bonsai, horses, crab, flamingo, starfish) and significant lack of alignment (e.g. UIUC cars, faces, PASCAL car views). Examples of the classes and object and part localizations obtained by UCA are shown in figures 5.2a and 5.5. Fig. 5.4c,d shows quantitative evaluation of automatic object and part localization by UCA compared to hand generated ground truth on several dataset. The background images for each dataset were chosen randomly out of Caltech backgrounds set containing 900 images. To challenge our method, we tested it on different class vs. non-class mixes, namely 10%, 20%, 30% and 50%. This is compatible with experimenting with Google data, since manual validation done by [15] showed that on average, above 25% of images returned by Google image search are good examples. For every dataset, increasing percent of class images above the percent reported in the table gives even better results. The UIUC cars dataset contained only the 170 non-cropped
Table 5.1: Summary of fully unsupervised classification results obtained by the UCA method. For all datasets, the EER STD for UCA was ≤ 2% (computed by cross-validation). For motorbikes, airplanes and cars-rear, the class images were randomly chosen from larger sets and remaining images were also used for testing the learned models obtaining 1.3%, 2.3% and 2.8% average EER respectively. The average EER of UCA on the Caltech-5 datasets was 2.65%. Results of other unsupervised methods reported for Caltech-5 were: average EER of 4.08% [39], 7.35% [15] and 11.38% [63] and average multiclass detection rate of 5.4% [6]. Results of leading supervised methods on Caltech-5 are comparable to our unsupervised result: average EER of 2.25% [40] and 1% [9] ([9] did not test on cars-rear class). The object size relative to the background for each dataset was approximated and non-aligned test images of the original set (and equal amount of random background images), the training images of the original set are cropped, so to make the task harder they were not used. The Caltech-5 datasets (from [16]) were tested in order to compare with past unsupervised approaches that were tested on the same data, namely [63, 39, 15, 1, 6]. To ensure that the chosen Caltech101 classes are sufficiently hard, 9 of the 11 tested Caltech101 classes are the ones with lowest reported performance by [6] (average of entries for these classes on [6]’s confusion matrix diagonal is 49%). Note that unlike [6], we do not use color information in our scheme. An important characteristic of the UCA is its ability to deal with a low percentage of class images in the dataset. Methods such as [15, 39] that apply the pLSA method of [63] as a pre-processing step to identify and localize class examples may fail on such datasets. This was validated by testing the pLSA method with eight topics (optimal number proposed by [15]) on the
schooner dataset that contains only 10% class images. From the $N$ examples with maximal score in the class topic, less then 40% were class examples ($N = 10, 20, \ldots, 100$).

In the PASCAL car views experiment, we tested the ability of UCA to separate related sub-classes. In particular, out of the PASCAL 2007 training images, images depicting frontal and side views of cars were extracted. The scale of the images was normalized by vertical size and large background areas around each car was taken to make the set un-cropped and un-aligned. The UCA was then applied to this set in order to separate the views. Furthermore, when applied on a set of about 700 images containing all the car views, UCA successfully learned the ”frontal cars” subclass with similar EER to the two view experiment. Applying pLSA to the same set has yielded high error (32% EER). The ability of our method to separate similar sub-classes and specifically different views of the same class can also be useful in supervised learning applications. If a given training set of images of the same class can be automatically separated into a meaningful set of (inherently similar) subclasses, then it can greatly facilitate the learning task, by allowing the modeling of each subclass separately.

5.4 Conclusions

The UCA method has a number of basic advantages compared with previous unsupervised classification methods. First, the overall classification results are higher than obtained previously, and remain high even when class examples are sparsely distributed within the dataset. Surprisingly, on the tested classes, results of the unsupervised method are as good as leading supervised methods. Second, the method obtains precise object localization, indicated by a repeatable reference point on each detected object. Third, precise locations of the parts participating in the model are also made available. Fourth, the method is capable of separating similar classes and sub-classes, such as different views of the same class. The main novel aspects of the UCA method are the following. The model is iteratively improved by exploiting intermediate classification results, consistently improving the performance. A novel geometric model is used, which can be efficiently learned from the entire dataset, and therefore improve the methods ability to capture geometric consistencies, even when consistent configurations are sparse. The model uses a part detection scheme, which is trained to detect object parts with diverse appearances in their correct position. The resulting detections are therefore more reliable, providing precise part localization and improved
Figure 5.5: More examples of unsupervised object and part localizations obtained by the UCA method. See explanation in fig. 5.2a

overall performance.
Figure 5.6: Improvements due to consistency amplification. (a) Example of class vs. background separation obtained by the first iteration for the schooners class. Yellow part are the class images and the rest are backgrounds (the ordering is only for illustration purposes). The horizontal line shows the adaptive threshold $\eta \cdot \text{Max}$ used to select the set $H$ of high likelihood class examples for the next UCA iteration. (b) Examples of the objects identified and localized. (c) Table of EER improvement with UCA iterations. Only the classes that ran for more then two iterations are shown. The iterations continue until the set $H$ stops growing. The average EER of the first iteration (that used the generic features and not PSFs) was 30% for these classes. This illustrates the low (relative to the PSFs) consistency between the generic features and the class objects.
5.5 A Combined model for detecting, localizing, interpreting and recognizing faces

In this section I describe a method that combines face detection, localization, part interpretation and recognition, and which is capable of learning from very limited data, in a semi-supervised or even fully unsupervised manner using the UCA method described in this Chapter. Current state-of-the-art techniques for face detection and recognition are subject to two major limitations: extensive training requirement, often demanding tens of thousands of images, and detecting faces without explicitly detecting relevant facial parts. Both of these limitations hinder the recognition task, since for a specific face the number of available examples is usually small, and because the strongest cues for identity lie in the specific appearance of facial parts. The proposed method alleviates both these limitations by effective learning from a small training set and by detecting the face through, and together with, its main parts. This is obtained by the novel unsupervised training method. We tested our method on face detection and localization tasks both in a set of ‘real life’ images collected from the web as well as in LFW and MIT-CMU databases. We also show promising results of our method when applied to a face recognition task. The focus of this work is on a method that combines face detection (‘what is it?’), localization (‘where is it?’), part interpretation (‘where are the facial parts?’) and recognition (‘who is it?’), and which is capable of learning from very limited data, in a semi-supervised or even fully unsupervised manner. The method was developed for general object detection and localization and in this work we extend and apply it to the task of detecting and localizing faces. In addition, we extend it to perform the recognition task.

The face detection and localization problem has a long history in the literature, but a significant dividing line was the influential work by Viola and Jones [26]. Based on a survey by [44], before [26] the state-of-the-art methods for face detection were divided into several approaches: knowledge-based [73], feature invariant [74], template matching [62], and appearance based [57, 65, 60]. Following [26] the focus of face detection approaches turned towards efficient real-time face detection. In [26], the face is represented by a cascade of linear classifiers each using simple low-level ‘Haar features’ and trained using boosting methods [19]. Currently, many state-of-the-art face detectors are variants of the approach in [26], with different improvements and extensions. For example, [37] extend the set of Haar features, [59] improve feature computation efficiency and introduce a coarse-to-fine
Technique, [43] improve the optimization of the cascade, [23] offer an alternative to the boosting technique used in [26] and provide efficient methods for multi-view detection, and [72] suggest an improved set of low-level features and improve the computational efficiency. Currently, the best results were reported by [72].

Despite all the developments listed above, there are still two main limitations shared by current state-of-the-art face detection techniques. These limitations are especially relevant if one wants to extend such techniques to face recognition. The methods based on [26] usually have an extensive training requirement. For instance [23] train on 75,000 face examples, and [72] have around 23,000 faces and 30,000 non-faces in their training set. While general face examples are abundant, this is usually not the case for face recognition tasks, where only a handful of images may be available for a specific individual. The second limitation is non-specificity to facial parts. The Haar-like features based methods detect faces without explicitly detecting facial parts and thus are capable of face detection and localization, but are not suited for recognizing and discriminating between different shapes of face parts such as eyes, nose, ears, mouth, chin and etc. The ability to detect specific facial parts plays a key role in face recognition task, as the distinction between two individuals is often based on fine differences in specific face parts appearance such as types of nose, eyes, hair, etc.

The proposed method addresses both of these limitations. It is capable of learning from a small number of examples, even in the challenging unsupervised setting, in which the object examples appear at unknown locations and only in an unknown subset of the training images. The subset of images containing class examples may even be as small as 10% of the training set. In addition, our method is part-based, meaning that the object is detected through first detecting its parts. The parts and their spatial configuration are learned automatically using the UCA method described earlier. Moreover, each part is represented by a (learned) set of its so-called semantic equivalents. For example, the nose part is represented by a set of image fragments representing the nose viewed with varying poses, illuminations, and other types of variations. These semantic equivalents are learned without supervision, and they are combined using the efficient representation of the CNOR model (4) robust detector for a corresponding part.

The method was tested on unconstrained frontal face detection and localization tasks in a challenging set of ‘real-life’ images, collected randomly from the web. The faces are usually small relative to the images, appear at random locations in the images and exhibit strong scale, lighting, pose and expression variations. The set also contains partially occluded faces (e.g.
by sunglasses or other objects) and multiple face instances. We also tested
detection performance on the LFW database [24] and detection and local-
ization performance on the MIT-CMU dataset [48]. Examples of the output
of our method on images taken from various datasets appear in figure 5.12.
Finally, we applied our method to the simultaneous object detection, local-
ization and recognition task on a dataset consisting of the set of 'real-life’
images mentioned above mixed with an additional set of about 130 images
containing a face of a specific person. The person images also exhibit large
scale, pose, lighting, location, expression and occlusion variations. Our sys-
tem showed good performance on this combined detection, localization and
recognition task despite being trained on a limited set of only 10 images of
the specific individual. Examples of combined person detection and recogni-
tion appear at the bottom row of figure 5.12. In addition, we tested the
contribution of various aspects of the proposed method by removing some of
them and evaluating the performance on the same detection and localization
task.

The next Section (5.5.1) describes the method and the learning algo-

rithms used, section 5.5.2 provides details of the experimental validation
and section 5.5.3 contains the summary and proposes possible future re-
search directions in the domain of faces.

5.5.1 UCA-based face detection and recognition

The model for face detection (for brevity detection = detection + localiza-
tion + part interpretation) and recognition is trained in three stages, each
stage training a separate UCA model. In the first stage, $M_1$ - a model for
detection is trained on an unlabeled set of mixed face and non-face images
in an entirely unsupervised manner. In the second stage an improved model
for detection $M_2$ is trained on an additional training set, in which the faces
are detected using $M_1$. In the third stage, a model for recognition $M_3$ is
trained using several examples of a specific face to be recognized detected
and localized, based on the application of $M_2$.

**Stage 1:** The first stage is fully unsupervised, its training set consisting of
mixed non-face and face images (faces appear at unknown locations). This
stage was performed in the experiments of [29] and here we use the model it
learned (denoted $M_1$) for the second stage. Fifty face examples from Caltech
faces dataset were present in the unsupervised training set of the first stage
(together with 450 non-face images). In general this stage may be used in
cases we train on a semi-supervised set (having only images that contain
the learned object), but when object examples vary in viewing direction (or

78
other viewing conditions), e.g. if we get a set of mixed frontal and profile faces. Although it is possible to try to capture all the viewing directions by a single model, it is clearly harder and will typically cost in loss of ability to detect facial parts. So in this case it is beneficial to first apply unsupervised learning to separate different views and only later to learn a separate model for each view. Experiments along these lines were performed in [29] for automatically separating different car views.

**Stage 2:** The second stage is learning an improved model using weak supervision. The goal of this stage is to improve the performance of the model learned during the unsupervised stage by providing it with more object (face) examples. This stage could also be performed by the system in autonomous (unsupervised) online manner by crawling on web images and detecting instances of the object using the $M_1$ model. In our case we gave the system 300 additional image examples randomly chosen from the LFW database and ran the $M_1$ model on them to detect and localize faces on these images. The output of $M_1$ were face bounding boxes detected in the training images. We then train the model $M_2$ using the UCA method by assuming the output of $M_1$ to be the output of the G-phase in one of UCA iterations (see section 5.2 for the definition of the G-phase). The model $M_2$ is then used to perform the face detection and localization in section 5.5.2. The second stage can be seen as additional iterations of the UCA method applied in the first stage. The main difference is that the standard UCA loops over the same image
set over and over again, while in \( M_2 \) training stage new images are provided either in a weakly supervised or in online unsupervised manner.

**Stage 3:** The third stage, training a recognition model for a specific face, is performed when we receive a (limited) set of examples of a face of a certain individual (that we wish to recognize in the future) and train a UCA model on these images. The training examples need not be cropped or aligned and can exhibit any scale variation, all we require them to be is ‘roughly frontal’ (as the example images in Figure 5.12). The training is organized along the lines of the second stage. The resulting model \( M_3 \) can be used by itself for simultaneous detection, localization and recognition of a specific person, but it is better used in conjunction with \( M_2 \), since \( M_3 \) had only limited training on the few examples provided for the specific person. We investigate both of these options in the section 5.5.2.

Figure 5.7 summarizes the proposed approach. Section 5.5.2 describes the experimental validation of our method.

### 5.5.2 Results

To test the proposed method, we applied it to frontal face detection, localization and recognition tasks in several databases, namely: people-containing images gathered from Google image search (denoted WEB database); Labeled Faces in the Wild (LFW) database [24]; MIT-CMU dataset [48]; and a set of unconstrained images of a person taken under different viewing conditions, and at different locations and times (denoted PERSON database). The MIT-CMU dataset combines all the images from tests A, B and C in the dataset description [48]. The statistics of all the datasets are summarized in Table 5.2.

<table>
<thead>
<tr>
<th>Name</th>
<th># images</th>
<th># frontal faces</th>
<th>image size (rows x columns + STD)</th>
<th>Faces size (min – max)</th>
<th>Has multiple people</th>
<th>Has scale variations</th>
<th>Faces are aligned</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEB</td>
<td>354</td>
<td>477</td>
<td>626±593±197±220</td>
<td>31x20–180x130</td>
<td>+</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>LFW</td>
<td>13233</td>
<td>13233</td>
<td>250±250±0±0</td>
<td>135±95</td>
<td>–</td>
<td>–</td>
<td>+</td>
</tr>
<tr>
<td>MIT-CMU</td>
<td>130</td>
<td>511</td>
<td>429±440±227±208</td>
<td>21x14–253x168</td>
<td>+</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>PERSON</td>
<td>162</td>
<td>162</td>
<td>480±640±10±0</td>
<td>36x28–209x160</td>
<td>–</td>
<td>+</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 5.2: Provides statistics of the datasets used in our experiments

In the first experiment, the model \( M_2 \) was applied to the face detection task in the LFW database. Since faces in the LFW are cropped, roughly aligned and equally scaled, there was no reason to test localization on that
Figure 5.8: Summary of the LFW face detection experiments. Although faces in LFW are of the same scale, a full method including scale search was applied both to face-containing and background images.

database, although in several dozens of images that we manually checked the localization was perfect. The LFW database is comprised only of images containing faces detected by Viola & Jones face detector, so in order to test face detection, maximal response of the UCA was computed for all images in the LFW and in additional 916 background images which were a union of the Caltech and Google background sets. These measurements were used to build the ROC curve for face detection depicted on Figure 5.8. The error bars of the ROC were computed by 10-fold cross validation.

In the second experiment we tested the combined detection and localization performance of our method. To this end, model $M_2$ was applied to the WEB and MIT-CMU datasets. The results are summarized in ROC curves in Figures 5.9a and 5.9b respectively. We also compared our performance on the WEB database with the performance of the OpenCV [49] implementation of Viola & Jones face detector [26] and shown a significant performance gain (see figure 5.9a). The face was considered correctly localized if the detected bounding box exceeded the standard Jaccard index overlap score of 0.5 with the manually marked ground truth bounding box:

$$Jaccard\ Index\ (BB_1, BB_2) = \frac{|BB_1 \cap BB_2|}{|BB_1 \cup BB_2|}$$  \hspace{1cm} (5.2)

In order to test the contribution of various components of our method to
the final performance, Figure 5.9a also contains ROC curves of performance after removing different components. Specifically, we tested our method without using semantic equivalents for part detection ('No semantics' in the figure), without using the CNOR model for combining the semantic equivalents ('No CNOR combination') and without using the learned geometry of parts ('No star geometry'). In all cases, removal of these components resulted in significant drop in performance. When semantic equivalents were not used for part detection, a single best image fragment was chosen to represent the part. When CNOR model was not used for combining semantic equivalents, they were combined using a simple summation. When part geometry was not used, the parts were split into four groups corresponding to the four quarters of the face. The face was then detected as a combined vote of the four quarter detections, while each of the quarters was detected by the bag-of-features method. Localizing faces directly by a bag-of-features of the whole face (without using this four quarters scheme) failed to produce reasonable results due to the limitation imposed by the overlap score (eq. 5.2) and the currently used non-maximal suppression scheme.

Additionally, figure 5.9a contains an ROC curve of HOG + SVM classifier implemented along the lines of [8]. HOG descriptors were computed for all the face bounding boxes detected by \( M_1 \) in all the images used to train \( M_2 \). For these HOG descriptors RBF kernel SVM was trained against HOG descriptors of maximal score bounding boxes detected by \( M_1 \) in background images (Caltech + Google backgrounds).

In the third experiment combined detection, localization and recognition were tested on the combination of the PERSON and the WEB datasets. The results are summarized in figures 5.10a and 5.10b. Ten images of a person were used to train the model \( M_3 \). The models \( M_2 \) and \( M_3 \) were used in a cascade-like sequence. First, \( M_2 \) was applied to detect candidate faces in the image and then \( M_3 \) was applied to search for the face of the specific person at locations and scales close to the ones detected by \( M_2 \). We also tested the combined detection, localization and recognition using a single model \( (M_3) \). When used by itself, it produces less good detection and localization results (see Fig. 5.10a) due to the limited training on only 10 images. In figure 5.10b we also compare our approach to several baseline approaches. Specifically, we tried to replace the \( M_3 \) in the \( M_2 \rightarrow M_3 \) sequence by Nearest Neighbor PCA, NCC or HOG, all trained on the same 10 images we used for training \( M_3 \). As can be seen in Figure 5.10b, the performance of these methods is significantly worse. This can be partially attributed to the fact that these methods try to detect the whole face using a single template, while our method applies part based detection which, as explained in the
Figure 5.9: (a) Summary of the WEB face detection and localization experiments. Additionally provides comparison with OpenCV implementation of [26], HOG+SVM and performance after removing different components of our method. (b) Summary of the MIT-CMU face detection and localization experiments, ROC curve.

introduction, is more appropriate for the recognition task. Examples of face interpretation, that is detection of various facial parts, are given in Figure 5.11.

5.5.3 Discussion

This section presented a method for combined faces detection, localization, part interpretation and recognition in unconstrained images, where faces may appear at any image location, scale and under a variety of difficult viewing conditions. The method shows promising performance in various experiments on different datasets, superior to several standard baseline methods and the standard implementation of the Viola & Jones face detector. The method requires only a few images for training and can be trained in a fully unsupervised manner. The underlying models employed by the method are general and not limited to face detection.

Even in cases of semi-supervised training, the ability to automatically separate different object views in a given training set (containing mixed views) can improve classification performance of the learned models. Moreover, this ability also facilitates learning part based models that are capable of detecting ‘meaningful’ object parts. Initial experiments performed in [29] for automatic separation of car views from the PASCAL 2007 dataset, show
that our method has this ability. An interesting future research direction would be learning view-point invariant part based models for face detection from a mixed set of examples of different face views. Additional interesting extension may be linking the models of different views in terms of their detectable parts (such that semantically equivalent parts from different view models are linked) in order to facilitate view-invariant part interpretation.

In its current version, the method does not make full use of the detailed part interpretation obtained during the detection process for the purpose of subsequent individual face identification. Since part detection obtained by our method is usually highly accurate, for each facial part it is possible (in future work) to build a universal dictionary of part appearances, such as semantic equivalent image fragments proposed in [11]. Then, one may learn an association between each of the part appearances and the conditions (lighting, pose, expression, etc.) under which the part is viewed. Given even a single image of a specific individual, her facial parts may be detected (by our model) and categorized according to the learned part appearance dictionaries. Subsequently, when confronted with a new image of the same individual taken under different conditions, the learned association between the conditions and the part appearances in the new image (known following the detection, localization and part interpretation performed by our model) could be used to recognize the individual. Some ideas along these lines were explored in [28] (but without having a method for reliable part interpretation) and extending it might be an interesting future research direction.
Figure 5.11: Examples of detections of several of the (about 50) modeled parts. Yellow star shows the detected object model center location and the colored boxes show the parts that were detected by the model towards individual recognition under highly varying viewing conditions.
Figure 5.12: Examples of face detection, localization and recognition obtained by our method. The databases used for experiments are explained in the Results section. Examples inside the red box are from the WEB database, examples in the green box are from the MIT-CMU database and blue box contains examples of combined detection, localization and recognition of a specific person in the PERSON database. The current algorithm does not use color information, all the color images were processed in grayscale by the algorithm.
Chapter 6

Discussion
The long term goal we have addressed is to learn complex object models for object recognition which will be capable of bridging the gap with human performance. We contributed to this goal by introducing new models for classification with new levels of complexity such as learned object part detectors, and by introducing novel natural approaches to learning such models including incremental learning methods in an online setting and unsupervised learning. Guiding us in our research was an understanding of the limitations of the current approach to object recognition which relies on a fixed pre-determined hand labeled training set. Below I summarize the contributions presented in this Thesis which address the different aspects of this problem.

With online feature selection [1], the classification system is no longer limited to a fixed set of features obtained in an early learning stage. To handle the immense variability in natural objects it is essential to be able to collect features from a large variety of examples. One reason is that some features can appear infrequently, but are still essential for obtaining truly high recognition levels. A second reason is that some aspects of the class may slowly change over time, requiring continuous adaptation to the class properties. The online algorithm we developed enabled for the first time the collection and selection of large amounts of useful features and showed the advantages of online learning in practical object recognition tasks in both terms of scalability and adaptability. This direction has been since further explored and developed by other research teams with direct reference to our work [16]. In order for a learning system to fully benefit from continuously presented examples in the online setting it we allow the learned model’s complexity to be increased as more examples become available [2]. Our work in [2] shows that this computationally demanding task of determining model complexity can be accomplished efficiently in an online setting and lead to improved performance. Utilizing large training sets for object recognition is made possible only by a relaxation in the requirement for supervision (hand labeling). In [3] we introduced a new approach for unsupervised learning of object recognition models. In [4] we presented a system which utilizes this approach for the tasks of face detection and recognition. [3, 4] present a unique new approach to the unsupervised problem with practical outcomes which may come as a surprise: in our limited experimental validations our unsupervised model performed as well as models trained with supervision on the same tasks. In all our works [1-4] we experimentally tested the learned classifier for a wide range of object classes and on various benchmark datasets and compared them with current state-of-the-art classifiers. The experimental results suggest that our new approach
leads to improved classifier performance using less supervision. These results also imply that pursuing this new approach may lead to closing the performance gap with human performance by learning more complex classifiers from larger datasets.
Chapter 7

Future Work
The future goal is to automatically learn object classifiers which reach human performance in recognition tasks from the vast amount of visual data available today and with minimal human supervision. In particular we would like to develop a system which learns online and unsupervised. The system continuously receives images and automatically learns from them visual object categories to which the system is able to associate new unseen instances of objects. The model for each object classifier is updated and improved incrementally from each new example until finally the classifier for each visual category is as accurate as possible and surpasses performance of current classifiers. There are several unsolved challenges towards achieving this goal, some of them we have started working on and some are planned to be addressed in the future research. I next give a list of these research directions.

7.1 Learning object part detectors with multiple appearance and multiple appearance similarity measures

A current limitation of our part detectors (Section 5) is their reliance on a single measure for similarity between image patches, which is in our implementation the normalized cross correlation between their SIFT descriptors. This is a proved good measure of similarity between the local shapes in the two patches. It has been shown that performance can be improved by synergy of several types of descriptors which capture other relevant information such as color distribution, specific textures and global shape. A natural extension is to allow our part detector to learn a combination of not only multiple part appearances but also of multiple appearance similarity measures such as the ones described above. To achieve this extension the model should be reformulated to encapsulate the relations between different measures. This extension will lead to a more accurate and general object part detector able to deal with more variability of objects while decreasing the false alarm rate for detecting object parts, and thus improving the overall classification results.

7.2 Minimal detectable parts

Another interesting direction is to learn so called minimal detectable parts. These are object parts which are individually reliably detectable by humans.
For example in the task of human body detection it is enough to reliably detect an eye of a person or the neck area in order to deduce that there is a person in the image. In addition, detecting several such parts individually would assist in estimating the body-pose, a task which is today very difficult. This later task also explains why we need the part to be minimal, since a larger part detected (e.g. upper body) would contribute less to pose estimation compared to detecting the hand.

We have initiated an approach for learning such minimal detectable parts with two new aspects. The first is identifying such parts and learning in an unsupervised manner a deformable classifier for the part building upon the unsupervised approach described in [3]. This requires a major adaptation of the system in order to achieve the "minimality" constraint. The second aspect is clutter reduction. For example a major problem in detecting the neck area is the large amount of clutter due to different patterns on the clothing which is irrelevant for the task. It would be helpful to separate between this clutter and the useful information. We have started developing a low level method for achieving this separation. In principle, if we knew a point on the flesh (e.g. neck) we could define the local edges around it as useful information and edges occluded by these edges with respect to the reference point as clutter.

7.3 Unsupervised learning of a visual categories with no restrictions on the dataset

A current restriction of the unsupervised learning method described in [3] is that it learns a class of objects which are roughly the same size and pose. In order to learn from unconstrained datasets such as the ones returned by Google search when typing "car" we need to relax this assumption and be able to group together instances of different poses and sizes. This grouping can be achieved both at the object level by identifying parts common to several views and doing a scale space search and at the object part level by using object part detectors invariant to affine transformations. This seems as a technical step but is in fact very difficult to achieve since it raises the already large amount of clutter from which the unsupervised method needs to handle, and would therefore impose a major improvement in the entire scheme. The result would have of high practical implications.
7.4 Learning a model of simple classifier and exceptions with increasing complexity

It has been shown that the performance of current classifier can be obtained by a relatively simple linear combination of carefully learned classification features [18]. Improving classification by increasing dataset size requires an increase in model complexity [2]. In the case of object classification a natural way to encode the increased classification is by introducing additional exceptions to the initially learned simple classifier. Such exceptions can be obtained from both positive and negative errors made by the classifier. An initial plan for this research is as following.

The basic classification model is a 'semantic star' such as the one described in [3]. It has a number of parts, and each part is composed of a number of features, each feature represented by a part detector such as the one described in Section 5, [3, 4]. I next elaborate on how we represent the 'model' and the 'exceptions': as an example, suppose we have a semantic start with 5 parts, each part contains 2 appearances. The total number of features, or components, is 10. We try during learning to find the best semantic star with 10 components, but we will still make in general some errors, both misses and FA. Each error is an exception, and we treat it by adding a special 'configuration'. Each configuration as an additional star model with a specific appearance feature at each object part node aimed at fixing the error, but carefully chosen such that it does not add new errors.

For a given k (number of components) the full model selection is therefore as follows. We construct the best semantic star with k components we can. With this model we will get some error. For each error, we find an optimal positive or negative configuration. When k is small, the number of exception will be larger. With a large k, we can eventually have no error on the data set. We select the k that gives the best performance in cross generalization to avoid over-fitting. As we get more data, we expect k to grow, and we can find the optimal k in the on-line incremental manner described in [2].

7.5 Unsupervised, bottom-up learning of hierarchical models for classification

We wish to follow a direction we have initiated in which visual features are grouped together in a bottom up way by a measure we termed "suspicious configurations" which measures the saliency of the combined feature. Important visual information can be obtained by grouping similar visual
structures emerging from low level features. Due to the low class specificity of such features and the lack of human "feature labeling", such features cannot be learned in a supervised manner. In addition, feature hierarchies have been shown to successfully capture visual structures [11]. Since we have no supervision, it is natural to perform the learning in a bottom up manner by grouping features that often appear together into more complex features. The initial idea for using such features is to replace the fragments level representation currently represented by fixed descriptors with such models. The final model is a hierarchical model of object categories combining supervised top down learning with unsupervised bottom up learning.

7.6 Incremental unsupervised learning of object classifiers in an online setting

A very important step towards the goal described in the beginning of this section would be to allow the unsupervised method described in [3] to be updated incrementally from examples in an online setting. Once this is achieved we would have a system which learns classifiers independently from arbitrary images in the internet. A relative simple extension of the current system would be to use the incremental principles from [1], which add the new features of a given example to the current classifier model. However, for such a system to continue improving we would eventually have to incorporate in it all the ideas described above. Specifically, such a system should increase its complexity as more examples become available as described in [2]. A natural way to encode the increased complexity is by introducing exceptions (7.4) and hierarchy (7.5). The demanding task would need to rely on a good detection of object parts in the bottom of the hierarchy (7.1, 7.2). Finally, to learn from arbitrary images size and viewpoint constrains should be eliminated (7.3).
Appendix I: CNOR EM update rules
Section 4 describes the probabilistic model we termed by "continuous noisy OR" (CNOR). In this appendix we derive the formulas of the EM learning of the model. The variables in the model are: \( V = \{v_1, \ldots, v_T\} \) (continuous, observed), \( Y = \{y_1, \ldots, y_T\} \) (binary, hidden, one for each element of \( V \)), and \( O^P \) (binary, observed in train, hidden in test). We assume here one element in \( V \) for each fragment in the selected set \( A_m \) of size 1, and ignore the learning of the representative set \( A_m \) which is achieved independently during the structure learning iterations of the SEM algorithm. We therefore assume a fixed structure and bring the equations for EM with a fixed structure. The maximization step of the EM is achieved numerically.

The probability decomposes as following:

\[
Pr(O^P|V; \Theta) = \sum_Y Pr(O|Y) \cdot \prod_{t=1}^T Pr(Y_t|V_t; \theta_t, R_m) \quad (7.1)
\]

The decomposition is also depicted by the graphical representation of the model in figure 5.3b. The definition of the model is completed by the following conditional probabilities:

\[
Pr(Y_t = 1|V_t; \theta_t, R_m) = \begin{cases} 
\frac{1}{1+e^{-\alpha_t(V_t - \tau_t)}} & Z_t \in R_m \\
0 & Z_t \in A_m \setminus R_m 
\end{cases} 
\quad (7.2)
\]

\[
Pr(O^P = 1|Y) = \begin{cases} 
1 & \exists t. Y_t = 1 \\
0 & \text{otherwise}
\end{cases} 
\quad (7.3)
\]

For convenience we define an indicator \( I \) which is 1 if the joint probability of \((Y, O)\) is positive:

\[
I(O, Y) = Pr(O^P, Y) > 0 = \begin{cases} 
0 & \forall t. Y_t = 0 \land O^P = 1 \\
1 & \text{otherwise}
\end{cases} 
\quad (7.4)
\]

The parameters to be learned by EM are \( \Theta = \{R_m, \theta_1, \ldots, \theta_T\} \) where for each fragment \( \theta_t = \{\tau_t, \alpha_t\} \) are the sigmoid function parameters defining the probability of \( Y_t \). Notice that if we wanted to express the joint probability \( Prob(V, O^P; \Theta) \) we simply have to multiply the left hand side of Eq. 7.1 by \( Prob(V) \). In the training stage this term does not affect the optimization of the parameters since it is constant with respect to the parameters. In the test stage we are actually interested in the conditional probability. We therefore only use conditional probability in the following derivations, and it can be easily shown the parameter learning is equivalent when using the joint probability. Assume we are given the set of training examples \( Data = \langle \vec{O}, \vec{V} \rangle = \langle (O^1, V^1, \ldots, O^N, V^N) \rangle \). The variable \( v^i_t \) denotes...
the observed similarity of training example \( i \) with selected representative \( t \).
The parameter learning goal is then to find the parameters maximizing the log-likelihood of the data \((LL(Data)):

\[
\Theta^* = \arg \max_{\Theta} \left( \sum_{i=1}^{N} \log \text{Prob}(O^i|V^i; \Theta) \right)
\]

(7.5)

The complete-data probability is defined by:

\[
\text{Prob}(O, Y|X; \Theta) = I(O, Y) \prod_{t=1}^{T} \left( \frac{1}{1 + e^{-\alpha(x_t-\tau_t)}} \right)^{y_t} \left( 1 - \frac{1}{1 + e^{-\alpha(x_t-\tau_t)}} \right)^{1-y_t}
\]

(7.6)

\[
= I(O, Y) \prod_{t=1}^{T} \left( \frac{1}{1 + e^{-\alpha(2y_t-1)(x_t-\tau_t)}} \right)
\]

(7.7)

The complete-data log likelihood is defined by:

\[
\log \text{Prob}(\vec{O}, Y|\vec{V}; \Theta) = \sum_{i=1}^{N} \log \text{Prob}(O^i, Y|V^i; \Theta)
\]

Given a current estimation of the parameters \( \Theta' \) the E-Step in the EM algorithm computes the expected value of the complete-data log likelihood given the current parameters estimation:

\[
Q(\Theta, \Theta') = E \left[ \log \text{Prob}(O, Y|V)|O, V; \Theta' \right] = \sum_{y=(0,1)^T} \sum_{i=1}^{N} \log \text{Prob}(O^i, Y|V^i; \Theta) \cdot \text{Prob}(y|O^i, V^i; \Theta')
\]

(7.9)

Denote by \( Z^i = \prod_{t=1}^{T} \text{Prob}(y = 0|v^i_t) \). Then \( Q \) can be expressed by:

\[
Q(\Theta, \Theta') = \sum_{t=1}^{T} \left( \sum_{\{i|O^i=0\}} \log \left( \frac{1}{1 + e^{\alpha(x_t-\tau_t)}} \right) + \frac{\text{Prob}(y = 0|v^i_t; \Theta') - Z^i}{1 - Z^i} \log \left( \frac{1}{1 + e^{-\alpha(x_t-\tau_t)}} \right) + \frac{\text{Prob}(y = 0|v^i_t; \Theta')}{Z^i} \right) \log \left( \frac{1}{1 + e^{\alpha(x_t-\tau_t)}} \right)
\]

97
In the M-step a local maximum of $Q$ with respect to the parameters $(\tau_t, \alpha_t)$ is found using numerical methods. From the equation above the maximizing parameters for each $t$: $(\tau_t, \alpha_t)$, can be made found independently.
Bibliography


