Feature Mining for Image Classification

Piotr Dollár[1] pdollar@cs.ucsd.edu
Zhuowen Tu[2] zhuowen.tu@loni.ucla.edu
Hai Tao[3] tao@soe.ucsc.edu
Serge Belongie[1] sjb@cs.ucsd.edu

[1]Computer Science & Engineering, University of California, San Diego
[2]Lab of Neuro Imaging, University of California, Los Angeles
[3]Computer Engineering, University of California, Santa Cruz

Abstract

The efficiency and robustness of a vision system is often largely determined by the quality of the image features available to it. In data mining, one typically works with immense volumes of raw data, which demands effective algorithms to explore the data space. In analogy to data mining, the space of meaningful features for image analysis is also quite vast. Recently, the challenges associated with these problem areas have become more tractable through progress made in machine learning and concerted research effort in manual feature design by domain experts. In this paper, we propose a feature mining paradigm for image classification and examine several feature mining strategies. We also derive a principled approach for dealing with features with varying computational demands. Our goal is to alleviate the burden of manual feature design, which is a key problem in computer vision and machine learning. We include an in-depth empirical study on three typical data sets and offer theoretical explanations for the performance of various feature mining strategies. As a final confirmation of our ideas, we show results of a system, that utilizing feature mining strategies matches or outperforms the best reported results on pedestrian classification (where considerable effort has been devoted to expert feature design).

1. Introduction

Feature design is a key problem in computer vision and machine learning as it can largely determine the performance of a vision system. Informative features capture the essence of an image pattern and reliable feature extraction facilitates a wide range of tasks such as detection, matching, recognition, tracking, and more generally any learning task in the image domain. Feature extraction is essentially a dimensionality reduction problem with the goal of finding meaningful projections of the original data vectors. A good feature should be (1) *informative*, (2) *invariant* to noise or a given set of transformations, and (3) *fast* to compute. Also, in certain settings (4) *sparsity* of the feature response, either across images or within a single image, is desired.

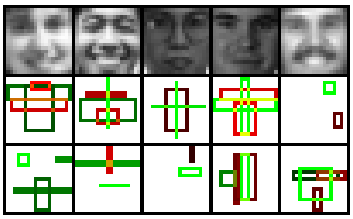


Figure 1. Example faces from the [4] database, and useful features for face detection discovered by feature mining (generalized Haar features described in Section 4). Note how many features have interesting, often symmetric patterns that to some degree resemble the structure of faces. Although these seem intuitive, they would be challenging to design.

Given an image, for example of a face, there are a plethora of ways to extract features, e.g. mean, variance, edges, gradients, filter responses, color features, geometric features, *etc*. and each can be computed at every position in the image with different sized windows, or pooled locally or globally over the entire image. The field continues to see significant advances in feature design; some recent work in feature design includes efforts in interest point detection and description, including the SIFT detector/descriptor [14] and improved versions of the Harris corner detector [15]. Interesting work in feature design also continues in specific domains, *e.g*. pedestrian detection [5] and tracking [3].

Though shown to be useful in various low-level, mid-level, and high-level vision tasks, existing features [5, 14, 15] are often good only in specific domains. One still needs to spend a considerable amount of time adapting and combining these features to specific problems. Such 'expert design' can require significant domain knowledge and insight to the problem. Still, most algorithms using these features remain far from perfect.

Another trend is to learn features automatically from training samples. Examples include work in dimensionality reduction, such as PCA or ICA, and approaches based on sparsity [17]. [13] advocates the use of a convolutional neural network, where feature extraction is implicitly performed by early layers of the network. In [11], the authors propose to automatically discover a sequence of image operations that results in useful features for classification. These methods, while promising, often tend to have restricted forms for learned features (typically linear) and

have not proven to be universally applicable.

We use the term feature mining to refer to the task of organizing and exploring large, possibly infinite, spaces of heterogeneous features. The aim of feature mining is to automatically discover meaningful features and to alleviate the burden of manual feature design. In this work, we introduce the feature mining paradigm and the concept of the data driven feature space. We also derive a principled approach for dealing with features with varying computational demands. We show experimental results on a number of typical data-sets [4, 5, 16], giving insight into the structure of the data driven feature space and various strategies for exploring it. We draw heavily from learning theory, and show both theoretical and empirical results of working with very large numbers of heterogenous features. Based on this study, we summarize some general principles for mining features efficiently and effectively. As a final confirmation of our ideas, we show results of a system, that utilizing feature mining strategies matches or outperforms the best reported results on pedestrian classification (where considerable effort has been devoted to expert feature design) [5, 16].

A related area of study is feature selection, where the goal is to pick a 'good' subset of features from some larger set of candidate features [2, 8, 10]. Feature selection methods can be divided into three types [2]: (1) wrapper methods that judge the quality of a subset of features by the performance of a trained classifier, (2) filter methods which assign a score to each feature, and (3) embedded methods where feature selection is a natural part of the learning method. Note that in this terminology the popular AdaBoost algorithm [7] can be used as an embedded method for feature selection [22]. In feature mining the goal is not to pick a subset of features from a larger set but rather to explore and model the entire space of features. Feature mining and feature selection are in this sense compatible.

Finally, [9] used the term feature mining in reference to search for evolving physical phenomenon in scientific data. Our framework bears similarity to this in name only, in fact [9] is a specialized technique for data mining.

2. Features and Supervised Learning

We focus on the role of features in the context of classification. The general goal in supervised learning is to learn a function from inputs to desired outputs that generalizes well to unseen data. One can attempt to learn such a function directly from the data in the representation in which it is given; however, this often makes the problem intractable either because the chosen classifier does not have the representational power to encode the function or because the amount of training data needed is prohibitive (see Figure 2). In many real problems such as detection, tracking and recognition, considerable thought and effort have been given to both designing meaningful features and choosing the classifiers.

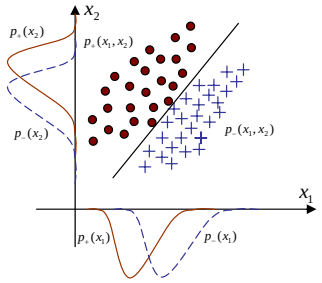


Figure 2. Toy example – the marginal distributions on x_1 and x_2 give little information about class membership, but classification becomes a simple thresholding given the feature $x_1 - x_2$. Here designing a useful feature or learning a classifier directly on the original data is simple. However, in many real problems such as detection, tracking and recognition, considerable thought and effort have been given to both designing meaningful features and choosing the classifiers. We use the term *feature mining* to refer to the task of organizing and exploring large spaces of heterogeneous features with the ultimate aim of discovering meaningful features.

For the remainder of this paper we will use Discrete AdaBoost [7] as our classifier. Using feature mining we generate many potential candidate features, AdaBoost then combines a subset of the mined features into the final classifier. Arguably, feature mining would be even more significant for techniques that for computational or theoretical reasons do not deal well with large feature sets, *e.g.* support vector machines [21] or neural networks.

2.1. Discrete AdaBoost

We begin with a brief review of AdaBoost [7]. Given N labeled training examples (x_i, y_i) with $y_i \in \{-1, 1\}$ and $x_i \in \mathcal{X}$, and an initial distribution $D_1(i)$ over the examples, AdaBoost combines a number of weak classifiers h_t to learn a strong classifier $H(x) = sign(f(x))$. Here $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$. The training error $\epsilon_{train} = \sum_i D_1(i)\mathbf{1}(y_i \neq H(x_i))$ is bounded by

$$\epsilon_t = \sum_i D_t(i)\mathbf{1}(y_i \neq h_t(x_i)) \quad (1)$$

$$\epsilon_{train} \leq \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)}, \quad (2)$$

where T is the number of weak classifiers, ϵ_t is the error of each weak classifier on the distribution D_t it was trained on, and $\mathbf{1}$ is an indicator function. See Figure 3 for details.

The VC dimension [21] of a classifier H, $VC(H)$, can be used to derive a loose upper bound on the expected test error of H. Roughly speaking, $\epsilon_{test}(H) \preceq \epsilon_{train}(H) + VC(H)$. Schapire *et al.* [18] further showed a bound on AdaBoost test error by analysis of the margin, which can very roughly be linked to VC dimension in the following manner: $VC(H) \approx O(\sqrt{d/m})$ where d is the VC dimen-

Given: N labeled training examples (x_i, y_i) with $y_i \in \{-1, 1\}$ and $x_i \in \mathcal{X}$, and an initial distribution $D_1(i)$ over the examples.
For $t = 1, ..., T$:

- Train a weak classifier $h_t : \mathcal{X} \rightarrow \{-1, 1\}$ using distribution D_t.
- Calculate the error of $h_t : \epsilon_t = \sum_{i=1}^{N} D_t(i)\mathbf{1}(y_i \neq h_t(x_i))$.
- Set $\alpha_t = -\frac{1}{2}\log(\epsilon_t/(1-\epsilon_t))$.
- Set $D_{t+1}(i) = D_t(i)\exp(-\alpha_t y_i h_t(x_i))/Z_t$, where $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$ is a normalization factor.

Output the the strong classifier $H(x) = sign(f(x))$, where $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.

Figure 3. **Discrete AdaBoost**

sion of each weak classifier and m is the number of training samples. Note that the test error does not depend on the number of weak classifiers. Thus, general tactics for training AdaBoost are to: (1) increase the number of training samples, (2) reduce training error and (3) reduce the complexity of the weak classifiers.

As is typical [22], we compute each weak classifier from a single feature. Here we use decision stumps (thresholded features). Extending the feature mining paradigm to real valued weak classifiers remains for future work.

2.2. Computational Complexity of a Feature

We begin with a simple extension to AdaBoost that allows us to deal with features of heterogenous computational complexity in a principled manner. Suppose that the average amount of computation needed to evaluate a feature is known. Given two features with similar error it is natural to favor the faster one. Below we derive a modified update rule for AdaBoost that takes into account features' computational complexity. As far as we know, no such approach has been proposed in the literature, possibly because existing systems use features of homogeneous type and computational complexity. As we will show, in the context of feature mining this rule proves crucial.

Recall from Equation (2) that the upper bound on AdaBoost training error is $\epsilon_{train} \leq \prod_{t=1}^{T} Z_t$, where $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$. In every stage t of learning, AdaBoost selects the weak feature that minimizes Z_t (*i.e.* has the lowest error ϵ_t). Now suppose each feature f_t takes c_t 'units' of time to compute, and we wish to train a classifier that uses a total of T units of time. In the standard setting $c_t = 1$ for all features, and AdaBoost selects a total of T weak classifiers.

The key to updating AdaBoost's greedy selection rule in this setting is to introduce the notion of a partial feature (which, like an imaginary number, is for mathematical convenience only). For every feature f_t with an error bound of Z_t and complexity c_t, define the partial feature f'_t as having an error bound of $Z'_t = Z_t^{1/c_t}$ and complexity $c'_t = 1$. Selecting c_t copies of f'_t reduces the upper bound by $\prod_{t=1}^{c_t} Z'_t = Z_t$, *i.e.* selecting c_t copies of f'_t is exactly the same as selecting one copy of f_t, both in terms of computational cost and effect on upper bound. In other words, the reduction in the upper bound of the error per unit time from feature f_t can be characterized as Z_t^{1/c_t}.

This leads to the the following update rule for AdaBoost: **Select the feature f_t with computation cost c_t and error ϵ_t which minimizes**

$$Z'_t = Z_t^{1/c_t} = (2\sqrt{\epsilon_t(1-\epsilon_t)})^{1/c_t}. \quad (3)$$

This rule is intuitive. Two features, f_1^a and f_1^b, which each reduce the upper bound by Z_1 and have cost $c_1 = 1$, are for all intents identical to a single feature f_2 with $Z_2 = Z_1^2$ and cost $c_2 = 2$. Note however, that after selecting f_1^a there is no guarantee that f_1^b exists, so the choice of f_1^a may be suboptimal – this is of course the nature of a greedy algorithm. Nevertheless, as we show in Section 4, greedily minimizing Z'_t in every stage of the AdaBoost procedure is very effective.

3. Feature mining

In this section we elaborate on the concept of feature mining, in which our goal is to minimize the human effort needed to explore and organize the vast space of possible features for image classification.

3.1. Parameterized Feature Space $\mathcal{P}$

Let $\mathcal{X}$ be our data space. We begin with the concept of a *parameterized feature space*. The parameterized feature space $\mathcal{P}$ for a given class of features is simply a human designed space of features, where each feature $f \in \mathcal{P}$ is a function $f : \mathcal{X} \rightarrow \{-1, 1\}$. For simplicity, we assume real valued features are transformed to binary ones by thresholding. For example, given $\mathcal{X} = \mathbb{R}^n$, a possible parameterized feature space is $\mathcal{P}_{poly}$ where each feature is a polynomial computed over $\mathbf{x} \in \mathbb{R}^n$, *e.g.* $f_1(\mathbf{x}) = 3x_1^2 + x_2$ or $f_2(\mathbf{x}) = x_3^{1.73}x_{11}$. Note that $\mathcal{P}$ may be infinite, as is the case for the example above. We make no assumptions about the parametrization of f, for example the representation may have variable length, and there may be multiple parameterizations for the same feature or set of features.

Given a parameterized feature space $\mathcal{P}$, we need some way of sampling or searching the space for meaningful features. However, a priori, we have no notion of where to look in the space, *e.g.* all the useful features may be concentrated in a small region of the parameter feature space. We also lack a measure of distance between features – in the case of a fixed length representation standard vector norms could be used but there is little reason to believe this would yield meaningful measures of distance. The problem becomes even more challenging given multiple heterogenous feature types $\mathcal{P}_1, ..., \mathcal{P}_k$, *e.g.* in the case of image filter responses, gradient histograms, edges, *etc*.

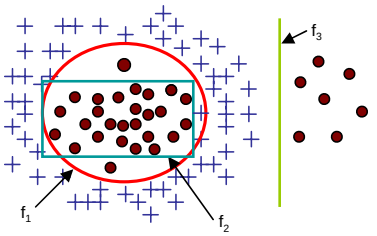


Figure 4. Toy data with 3 features shown. Features f_1 and f_2, with the circle and square decision boundaries respectively, are of fundamentally different types yet they agree on most of the data points shown. Using feature set $\{f_1, f_3\}$ should lead to a lower overall classification error than using $\{f_1, f_2\}$ even though individually f_2 has lower error than f_3.

Due to these challenges working directly with $\mathcal{P}$ can be challenging. Typically, through careful and systematic design an appropriate set of features is chosen from $\mathcal{P}$ for use within classification.

3.2. Data Driven Feature Space $\mathcal{F}$

Here we introduce the concept of a *data driven feature space*. The key idea is to represent each feature independently of its parametrization. Instead, each feature is characterized by its response to the data, which captures all the relevant information about the feature. There is a countless number of possible parameterized feature spaces, however, given the data there is a unique data driven feature space $\mathcal{F}$.

As before, let (x_i, y_i) with $y_i \in \{-1, 1\}$ and $x_i \in \mathcal{X}$ be the N labeled training examples. We characterize a feature f by its response to the data:

$$\Omega(f) = \big(f(x_1), ..., f(x_N)\big) \in \{-1, 1\}^N = \mathcal{F}. \tag{4}$$

Thus $\Omega : \mathcal{P} \rightarrow \mathcal{F}$. In an abuse of notation we will use $f \in \mathcal{F}$ instead of $\Omega(f) \in \mathcal{F}$. In the toy example shown in Figure 4, features f_1 and f_2, with the circle and square decision boundaries respectively, are of fundamentally different types yet they agree on most of the data points shown. That is $\Omega(f_1) \approx \Omega(f_2)$.

A natural measure of the informativeness of a single feature is $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$ (2), where ϵ_t (1) is its weighted error on the training data. If the computation time c_t of a feature is given, the informativeness of a feature is given according to (3) as $Z'_t = Z_t^{1/c_t}$.

Measuring how informative a set of features is not as straightforward. Referring back to Figure 4, we can see that using feature set $\{f_1, f_3\}$ should lead to a lower overall classification error than using $\{f_1, f_2\}$ even though individually f_2 is more informative than f_3. Intuitively, this occurs because f_1 and f_2 have a very similar response to the data. We formalize this below.

3.3. A Metric for $\mathcal{F}$

We derive the following metric in the context of discrete AdaBoost. The concepts, as well as the metric, should be generally applicable although in specific contexts more appropriate metrics may be devised. Again, for notational simplicity we use f in place of $\Omega(f)$.

We begin with some notation. Given two features f_1 and f_2, let w_{00} be the fraction of data where both f_1, f_2 are incorrect, given distribution D_t:

$$w_{00} = \sum_{i=1}^{N} D_t(i)\mathbf{1}(y_i \neq f_1(x_i))\mathbf{1}(y_i \neq f_2(x_i)). \tag{5}$$

Likewise let w_{01} be the fraction of data where f_1 is incorrect and f_2 correct, and similarly for w_{10} and w_{11}. Note that $w_{00} + w_{01} + w_{10} + w_{11} = 1$.

Recent work [20] formalized the concept of complementary features. This was in the context of a lower bound on the error of any weak classifier f_{t+1} chosen at step $t+1$ given the error ϵ_t of a classifier f_t chosen as step t. We reproduce the proof here for clarity, changing notation:

$$\begin{aligned}
\epsilon_{t+1} &= \sum_{i=1}^{N} D_{t+1}(i)\mathbf{1}(y_i \neq f_{t+1}(x_i)) \\
&= \sum_{i=1}^{N} \frac{D_t(i)\exp\left(-\alpha_t y_i f_t(x_i)\right)}{Z_t}\mathbf{1}(y_i \neq f_{t+1}(x_i)) \\
&= \sum_{i=1}^{N} \frac{D_t(i)e^{-\alpha_t}}{Z_t}\mathbf{1}(y_i \neq f_{t+1}(x_i))\mathbf{1}(y_i = f_t(x_i)) \\
&\quad + \sum_{i=1}^{N} \frac{D_t(i)e^{\alpha_t}}{Z_t}\mathbf{1}(y_i \neq f_{t+1}(x_i))\mathbf{1}(y_i \neq f_t(x_i)) \\
&= \frac{1}{2(1-\epsilon_t)}w_{10} + \frac{1}{2\epsilon_t}w_{00}
\end{aligned} \tag{6}$$

Since AdaBoost selected the best feature at time t, the error of f_{t+1} on distribution D_t must be at least ϵ_t; in other words $w_{10} + w_{00} \geq \epsilon_t$. Combining with (6) gives the bound:

$$\epsilon_{t+1} \geq \frac{\epsilon_t^2 + (1-2\epsilon_t)w_{00}}{2\epsilon_t(1-\epsilon_t)}. \tag{7}$$

The smaller w_{00}, the better feature f_1 given f_2, and vice versa, so w_{00} reflects complementarity of two features. In this work we describe how to transform this notion into a measure of distance between features in the data driven space. We need to make sure the distance between a feature and itself is zero. We achieve this by:

$$d(f_1, f_2) = \frac{1 - w_{00} - w_{11}}{1 - w_{11}} = \frac{w_{01} + w_{10}}{w_{01} + w_{10} + w_{00}}. \tag{8}$$

If $\Omega(f_1) = \Omega(f_2)$, we have $w_{00} + w_{11} = 1$ and so $d(f_1, f_2) = 0$. d ranges from 0 to 1, and it measures the ratio of samples with exactly one error to samples with at least one error. Note that distance between features in $\mathcal{F}$ is unrelated to any notion of distance in $\mathcal{P}$; also $d(f_1, f_2)$ depends entirely on the data. d is defined everywhere unless $w_{11} = 1$, *i.e.* two features are identical and correct on

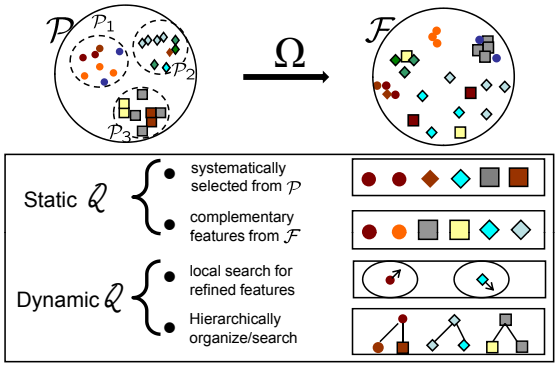


Figure 5. Illustration of the space $\mathcal{P}$ and the corresponding data driven space $\mathcal{F}$. Our task if to explore and organize the feature space to find an optimal set of features. At bottom the features are shown organized into various static and dynamic representations $\mathcal{Q}$.

all training points (although the existence of such features makes a given data set trivial).

The standard L_1 distance expressed in this notation is:

$$L_1(f_1, f_2) = \sum_{i=1}^{N} D_t(i)|f_1(x_i) - f_2(x_i)| \\ = 2(w_{01} + w_{10}) = 2(1 - w_{00} - w_{11}). \quad (9)$$

d differs from L_1 by a normalization factor of $2(1 - w_{11})$. The effect of this is that two features f_1 and f_2 with high accuracy can still have a distance near 1 so long as they rarely make mistakes in the same places. This property makes d well suited for our needs.

It can be shown that d satisfies all the conditions of a metric: (1) d is symmetric, (2) non-negative, (3) $d(f_1, f_2) = 0$ iff $f_1 = f_2$, and (4) d satisfies the triangle inequality. (1) and (2) are easily verified, (3) follows because $w_{00} + w_{11} = 1$ iff f_1 and f_2 are the same feature in the data driven feature space. We give a brief sketch of the proof of (4). Given three features f_1, f_2, f_3, define $w_{000}, ..., w_{111}$ in an analogous manner to $w_{00}, ..., w_{11}$. Rewriting $d(f_1, f_2) = 1 - \frac{w_{000}+w_{001}}{1-w_{110}-w_{111}}$, and similarly for $d(f_2, f_3)$ and $d(f_1, f_3)$, one can verify that $d(f_1, f_2)+d(f_2, f_3) \geq d(f_1, f_3)$ for any choice of $w_{000}, ..., w_{111}$.

We always compute d over the initial distribution D_1, (D_t for $t > 1$ is not available unless we are actually training AdaBoost); this works well in practice. Note also that any pairwise measure cannot with perfect accuracy predict the utility of sets of three or more features. Finally, note that d is distinct from the classic diversity measures of a set of classifiers [12], most notably in that it is a metric.

3.4. Exploring and Organizing the Feature Space

The input to a feature mining strategy is a data set, along with a number of possibly heterogeneous $\mathcal{P}_1, ..., \mathcal{P}_k$, collectively referred to as $\mathcal{P}$. Again, $\mathcal{F} = \{\Omega(f)|f \in \mathcal{P}\}$ is the corresponding data-driven feature space. $\mathcal{F}$ provides a measure of quality for each feature as well as a meaningful metric over features. Feature mining is the process of exploring and organizing $\mathcal{P}$ by exploiting the metric structure of $\mathcal{F}$ and whatever structure $\mathcal{P}$ may have.

For reasons already stated $\mathcal{P}$ is difficult to work with. For example, it is not feasible that AdaBoost, during every stage of training, examine every possible feature in $\mathcal{P}$. Therefore, we wish to organize the features into a representation $\mathcal{Q}$ that is more useful. $\mathcal{Q}$ can either be static, *i.e.* simply a fixed collection of features, or in addition have dynamic operations associated with it, *e.g.* a functions that generates a feature's neighbors. A static $\mathcal{Q}$ should contain a useful set of features for a given task, while a dynamic $\mathcal{Q}$ should in addition be efficiently searchable. Figure 5 shows an illustration.

In the literature, people often systematically choose a subset of features $\mathcal{Q}$ from $\mathcal{P}$ without a careful study of the feature space. Feature selection methods [2, 8, 11] can also be used to pick a subset of features if $\mathcal{P}$ is not too large, however, they cannot be used to build a dynamic $\mathcal{Q}$, nor do they exploit any structure $\mathcal{P}$ may have. Finally, in work in the vein of [19, 1], the authors manually construct a dynamic $\mathcal{Q}$ so that it can be searched efficiently during every stage of AdaBoost using evolutionary search strategies. These are examples of feature mining strategies; here we treat the concepts more generally.

We conclude with an outline of the desired properties static and dynamic $\mathcal{Q}$ should have, and some basic ideas of how to construct $\mathcal{Q}$. Details of the actual strategies we implemented are given in Section 4. Neither the guidelines below nor the implemented strategies are meant to be exhaustive, in fact we expect more sophisticated feature mining strategies and representations to be proposed.

Typically, given a new data set little is known about the feature space, including the quality and diversity of available features. Unless additional information about $\mathcal{P}$ is given, the process of exploring $\mathcal{P}$ must begin with random sampling, but as more of the space is explored the sampling can become more refined. For example, given a series of elements $f_1, ..., f_k \in \mathcal{P}$ and corresponding $\Omega(f_1), ..., \Omega(f_k) \in \mathcal{F}$, we could try to predict the next sample to draw from $\mathcal{P}$. Furthermore, if $\mathcal{P}$ is somewhat smooth, that is if a small (possibly discrete) perturbation in a feature f results in a small change in $\Omega(f)$, then steepest descent search can be used to refine existing features.

To create a static $\mathcal{Q}$, representative samples can be stored, in general these samples should be both informative (Equation 1) and diverse (Equation 8). One way to achieve this is to maximize a scoring function of the form:

$$J(\mathcal{Q}) = \sum_{f_i \in \mathcal{Q}} Z(f_i) + \beta \sum_{f_i \neq f_j} d(f_i, f_j), \quad (10)$$

where $d(f_i, f_j)$ is the metric defined before, and β balances the informativeness and complementarity of the features.

An alternative is to cluster the features using the metric d (again, in an entirely data driven way), and keep representative features with a bias toward informative features.

Instead of working with a static $\mathcal{Q}$, $\mathcal{Q}$ can be constructed so it is efficiently searchable. The idea is to do this once, so that during training of an algorithm like AdaBoost, which requires multiple features, each feature can be obtained efficiently. For example, using d we can create a hierarchical representation $\mathcal{Q}$ of the space, which can be searched efficiently by traversing the hierarchy. Although not guaranteed to find the optimal feature, such an approach can be orders of magnitude faster than a brute force search in every stage of training. Other, more complex methods may prove useful, *e.g.* learning a mapping $\Omega^{-1} : \mathcal{F} \rightarrow \mathcal{P}$, such that $\Omega^{-1}(\omega)$ gives a feature f where $\Omega(f) \approx \omega$, if such a feature exists. This is outside the scope of the present work.

4. Experiments

Here show experimental results on three data-sets [4, 5, 16]. The first is the MIT CBCL face data-set [4]. Although aligned frontal face detection has essentially been solved [22], this simple data-set is still useful for comparative studies. Recently, pedestrian detection has received much attention [5, 16, 23]; specifically there has been much interest in designing more effective features for the task. Thus, this challenging domain serves as a perfect test bed for feature mining. Here, we use the two data-sets from [5] and [16].

Recall that our goal was to put little effort into designing $\mathcal{P}$, yet $\mathcal{P}$ must be large and complex enough to represent diverse patterns. The fast to compute Haar wavelets [22] are widely used in the community. Each Haar wavelet is computed by summing the pixels of 2 to 4 weighted rectangles. Here we introduce generalized Haar wavelets, which are like the original Haars but with arbitrary configuration and number of rectangles. Even for a 50x50 image patch there are $\mathcal{O}(10^6)$ configurations for a single rectangle. With multiple rectangles per Haar, $\mathcal{P}$ becomes quite vast. These features are capable of representing some fairly interesting patterns, see Figure 1.

Following the methodology from [6], we compute multiple channels or views of an image, and compute Haar features over each channel. For the channels we use the original image, gradient magnitude, channels from convolution with a bank of Gabor filters, and the RGB color channels (given color information). We deem this a reasonable amount of design for $\mathcal{P}$, since implementing the above is trivial given a standard image processing toolbox.

We report all our results using receiver operating characteristic (ROC) curves, which plots true positives versus false positives as the detection threshold is varied. Due to inherent randomness in the experiments, we repeat each experiment 10 times, changing the training data used, and 'average' the resulting ROC curves (see [16] for a discussion). Confidence intervals are shown.

4.1. Feature Mining Strategies

We implemented a number of basic strategies. These are meant to confirm our theoretical results and more generally demonstrate the importance of feature mining.

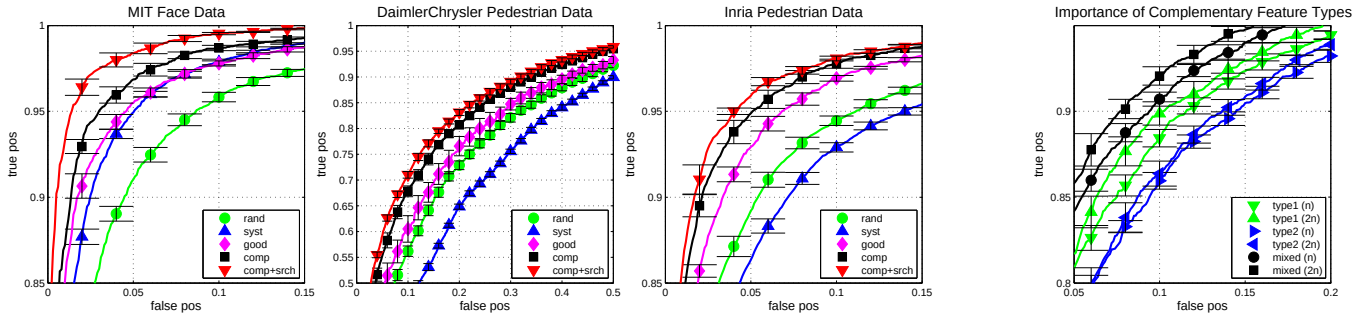
- SYST *Systematically* designed ftrs. for face detection [22].
- RAND *Randomly* sampled features from $\mathcal{P}$.
- GOOD The space is mined for *informative* features. Random features are sampled from $\mathcal{P}$, and the most informative (according to Z_t) are kept. Additionally, steepest descent search in $\mathcal{P}$ is used to refine the best features. Specifically given $f \in \mathcal{P}$, we can generate a set of nearby $f' \in \mathcal{P}$ by randomly perturbing the parameters of f, and keep the best feature from the set. We enforce that no two features f_1 and f_2 can have the same parametrization, however, no effort is made to ensure $d(f_1, f_2) > 0$. Given that the space is large, possibly infinite, the above process can be continued indefinitely. In the experiments below, we generate and refine 1000 random candidate features per final feature.
- COMP The space is mined for informative, *complementary* features. Here we exploit our metric on $\mathcal{F}$ and use a simple online clustering algorithm, where each cluster is a sphere of fixed size radius. Let $\mathcal{Q}$ denote the current set of selected features. Features are sampled randomly from $\mathcal{P}$, each new feature can (1) become a new cluster center, (2) replace an existing cluster center(s), or (3) be deemed as redundant. Given a new feature f, let F denote all $f' \in \mathcal{Q}$ such that $d(f, f') < r$. If (1) F is empty add f to $\mathcal{Q}$, otherwise if (2) the informativeness of f is greater than the informativeness of any $f' \in F$, add f to $\mathcal{Q}$ and remove all $f' \in F$, otherwise (3) discard f. The choice of the radius is based on the desired number of final clusters. Again we used steepest descent search to further optimize $\mathcal{Q}$, and again we generate and refine 1000 random candidate features per final feature.

Additionally, during every stage t of AdaBoost training steepest descent search can be used to further optimize a given feature set $\mathcal{Q}$ based on the current distribution D_t. The steepest descent search is the same as for generating the set GOOD, except the distribution D_t is different. This gives rise to the strategies STRAT+SRCH, where STRAT=RAND, GOOD, *etc*. Additionally, we use STRAT+TC to denote that $Z'_t = Z_t^{1/c_t}$ (3) is used instead of Z_t.

4.2. Comparative Results

The first set of experiments is meant to compare systematically designed features and the various feature mining strategies define above. Results on the three data-sets are shown in Figure 6a[1]. The ordering of the performance of

[1]Since there are 8 strategies, 3 data-sets, and 10 repetition of each, for computational reasons we performed each of the 240 experiments using only 15 weak classifiers, and each mining strategy was limited to 1000 features. The study nevertheless gives insight.



(a) Comparative studies. See text for discussion.

(b) Complementary feature types.

Figure 6.

the feature mining strategies on each data-set is RAND $\prec$ GOOD $\prec$ COMP $\prec$ COMP+SRCH. This is as expected. One interesting thing was how near COMP was to COMP+SRCH. Coupled with the worse performance of GOOD, this serves as a verification that the metric over $\mathcal{F}$ works well.

Although not shown, GOOD+SRCH, often did not work as well since the candidate features for search were not diverse enough. RAND+SRCH, when used with a lot or random features and deep search can be seen as a brute force method for exploring $\mathcal{F}$ during each stage of boosting. In terms of performance RAND+SRCH with a large number of features did not greatly outperform COMP+SRCH.

The systematic features performed quite well on the face data-set, but not nearly as well on the pedestrian data-sets. This makes sense, since they were designed for face detection. In a separate experiment, not shown, it turns out that the performance of RAND on the face dataset becomes similar to SYST if the number of features allowed for RAND is 10 times that of SYST. This implies one can forgo the design stage for Haars given more computing time and the trivial feature mining strategy RAND.

4.3. General Observations

Although we have already discussed the importance of complementary features, complementary *classes* of features are necessary for complementary features to exist. We performed a simple experiment, with two different classes of features: (1) Haars computed over the original image and (2) Haars computed over the gradient magnitude image. It turns out that using just n mixed features (of both types) is better than either $2n$ features of type 1 or $2n$ features of type 2. ROC curves can be seen in Figure 6b.

It is interesting to observe the asymptotic performance of a trained AdaBoost classifier as the number of weak classifiers increases for the various feature mining strategies. The same question can be asked given the update rule in equation (3) that explicitly takes computational time into account. Results and discussion appear in Figure 7; overall the test error rate converges 8 times faster and lower final error is achieved.

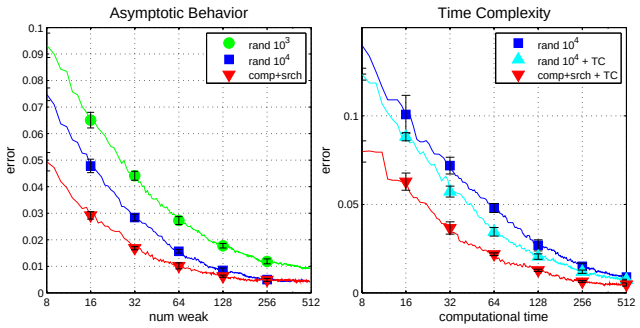


Figure 7. Test error as a function of (a) the number of weak classifiers and (b) computation time. In each case a log scale is used so the asymptotic performance can be seen. **(a)** Asymptotic performance of 3 different strategies: RAND with 10^3 and 10^4 features and COMP+SRCH. Note that the rate of convergence varies significantly between the strategies: by a factor of 2 between RAND 10^3 and RAND 10^4, and by another factor of 2 between RAND 10^4 and COMP+SRCH. Also, the test error for RAND 10^3 converges to a higher value. **(b)** The computational cost of a Haar feature with r rectangles is $c_t \approx 1+r$ (we let r range from 1 to 4). Since a feature with $r+1$ rectangles has representational power strictly greater than with r rectangles, without a constraint on time complexity AdaBoost tends to choose features with the maximum r. Using the update rule which takes time complexity into account, +TC, features with an average of $r = 1.5$ rectangles are chosen, for overall computational savings of an additional factor of 2 for any given level of error.

Finally, we again address the question of overfitting. Given that the space of features we're exploring is so large, by chance there will be features that spuriously fit the data well. Technically speaking, by enlarging the space we are increasing the VC dimension of the overall classifier, see Section 2.1. In an experiment not shown due to lack of space, we found that exploring the feature space more and more thoroughly we could continue to improve training error, but not test error. However, at no point did test error actually increase, meaning that too much exploration of the feature space was not helping but also not explicitly hurting. Using the update rule that favors faster features seemed to slightly improve test error, which probably occurs because the time complexity and VC dimension are correlated in this case. However, more experiments are needed.

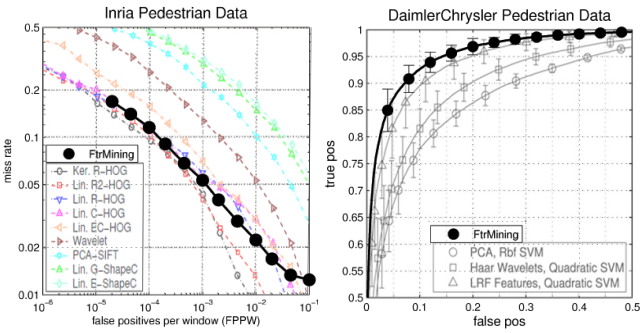


Figure 8. Results on 2 pedestrian detection data-sets. The thick black ROCs represents our result, the other ROCs are results obtained by the creators of each data-set. (a) Our results on the data from [5] essentially match results obtained using Histogram of Gradient features for low false positive rates. (b) Our results on the data from [16] beat the results reported.

4.4. Application To Pedestrian Detection

The data-set in [5] is known to be challenging, for example [23] showed that a cascaded classifier [22] with standard Haar wavelets does not achieve reasonable performance.

We trained a cascaded classifier with 20 levels and features mined from $\mathcal{P}$ as described above, using an identical training, bootstrapping and testing setup as [5]. We applied two feature mining strategies: (1) RAND+SRCH+TC with a small number of initial features and (2) RAND+TC with a very large number of candidates, giving similar results. In both cases using complementary channels and TC were essential. Our results are shown in Figure 8a overlayed on Figure 3b from [5]. For low false positives, our results essentially match the best reported results obtained using the Histogram of Gradient (HOG) features designed specifically for this data. Note that $\mathcal{P}$ did not contain any histogram features.

We also evaluated the same strategies on the pedestrian data-set described in [16]. The best reported results in [16] were obtained using SVM on features learned by a convolution net. Our approach improves the error, our results are shown in Figure 8b overlayed on Figure 5d from [16].

5. Conclusion

In this work we have aimed to lay out a general framework for feature mining, grounding it in theory and supporting it with experiments. Feature mining is meant to alleviate the effort and expertise necessary for feature design, and ultimately serve as a foundation for systems that can outperform those based on manually designed features.

Acknowledgements

This work was funded by the following grants and organizations: NSF Career Grant #0448615, Alfred P. Sloan Research Fellowship, NSF IGERT Grant DGE-0333451, and UCSD Division of Calit2. Z.T. was funded by the NIH through grant U54 RR021813 entitled CCB. We thank Microsoft Research Asia for organizing the CTCV workshop in which some initial ideas were stimulated. We would like to thank Boris Babenko for valuable input and Anna Shemorry for her support.

References

[1] Y. Abramson, F. Moutarde, B. Steux and B. Stanciulescu, "Combining AdaBoost with a Hill-Climbing Evolutionary Feature Search," SCIP workshop at *FLINS*, 2006.

[2] A. Blum and P. Langley, "Selection of Relevant Features and Examples in Machine Learning," *AI*, 1997.

[3] R.T. Collins and Y. Liu, "On-line selection of discriminative tracking features," *ICCV*, Nice, 2003.

[4] CBCL Face Database #1, *MIT Center For Biological and Computation Learning* `http://cbcl.mit.edu/`

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *CVPR*, 2005.

[6] P. Dollár, Z. Tu, and S. Belongie, "Supervised learning of edges and object boundaries," *CVPR*, 2006.

[7] Y. Freund and R. E. Schapire, "A Decision-theoretic Generalization of On-line Learning And Application to Boosting," *J. of Comp. and Sys. Sci.*, 55(1), 1997.

[8] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, 2003.

[9] M. Jiang, T.S. Choy, S. Mehta, M. Coatney, S. Barr, K. Hazzard, D. Richie, S. Parthasarathy, R. Machiraju, D. Thompsonn, J. Wilkins, and B. Gatlin, "Feature mining paradigms for scientific data," *Proc. of 3rd SIAM Intl' Conf. on Data Mining*, 2003.

[10] D. Koller and M. Sahami, "Toward Optimal Feature Selection," *ICML*, 1996.

[11] K. Krawiec and B. Bhanu, "Visual Learning by Evolutionary Feature Synthesis," *ICML*, 2003.

[12] L. I. Kuncheva, C. J. Whitaker, "Measures of Diversity in Classifier Ensembles," *Machine Learning*, 2003.

[13] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Gradient-Based Learning Applied to Document Recognition," *Proc IEEE*, 1998.

[14] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 60, 2, 2004.

[15] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," *ECCV*, 2002.

[16] S. Munder and D. Gavrila, "An Experimental Study on Pedestrian Classification," *PAMI*, 28(11), 2006.

[17] B. A. Olshausen, D. J. Field, "Emergence of simple-cell receptive field properties by Learning a Sparse Code for Natural Images," Nature, 381, 1996.

[18] R. Schapire, Y. Freund, P. Bartlett and W. Lee, "Boosting the margin: a new explanation for effectiveness of voting methods," *Annals of Statistics*, 26(5), 1998.

[19] A. Treptow and A. Zell, "Combining Adaboost Learning and Evolutionary Search to Select Features for Real-Time Object Detection," *Congress on Evolutionary Computation*, 2004.

[20] Z. Tu, X. Zhou, A. Barbu, L. Bogoni, and D. Comaniciu, "Probabilistic 3D polyp detection in ct images: the role of sample alignment," *CVPR*, June, 2006.

[21] V. Vapnik, "Statistical Learning Theory," Wiley-Interscience, New York, 1998.

[22] P. Viola & M. Jones, "Fast multi-view face detection," *CVPR*, 2001.

[23] Q. Zhu, M. Yeh, K. Cheng, and S. Avidan, "Fast Human Detection Using a Cascade of Histograms of Oriented Grad.," *CVPR*, 2006.