

---

# Feature selection applied to image beauty estimation

---

Aldebaro Klautau \*

## Abstract

We investigate feature extraction and selection in the framework of automatically classifying forestry images according to their “beauty”. We experimentally evaluate three recent advances in feature selection: a wrapper method proposed by A. Ng that is more robust when the number of irrelevant features is large compared to the number of relevant ones, filter methods based on boosting and support vector machines (SVM). As baseline, we use 65 features chosen according to human expertise, which lead to 18.9% of error when the classifier is SVM. A data-driven method proposed by K. Tieu and P. Viola is used to obtain 46,875 alternative features. Feature selection methods are tested using these data-driven features. The SVM-based filter method achieved 18.9% of misclassification rate using 10,000 features, and a wrapper method with SVM as classifier led to 21.4% using only 4 features. We also discuss why, in practice, it is more convenient to use a filter method to pre-select features for wrappers than using Ng’s wrapper method with all features.

## 1 Introduction

Many applications require methods to represent data in a reduced number of dimensions given a set of measurements. This reduction can be achieved in essentially two different ways [Web99]. The first approach tries to identify those variables that contribute most to the classification task and keep only them, which is called *feature selection*. The second approach finds a transformation from the measures to a lower-dimensional space, such as in principal component analysis (PCA). This is called *feature extraction* and can be seen as part of the process of converting the data to a set of sensible features. For example, PCA can be applied to the wavelet coefficients obtained from raw image pixels [CB02], and the whole process (wavelet transform followed by PCA) seen as the feature extraction stage. First we discuss feature extraction.

In pattern recognition problems, the feature extraction stage is very important and often difficult to design. A common approach is to use all knowledge about the domain, trying to specify sensible features. For example, a set of 65 features (e.g.

---

\*Ph.D. student at UCSD, <http://speech.ucsd.edu/aldebaro>. Project for CSE254, Prof. Charles Elkan, Spring, 2002.

color content, entropy, fractal dimension, etc.) was extracted in order to classify images according to their beauty in [Zea99]. The choice of these features was mainly motivated by knowledge of image processing experts, given they were successfully used in other applications. Hence, we call them *knowledge-based* features. Such approach has drawbacks. For example, it is not clear that features useful for object recognition and other applications are the best to provide information about “beauty”.

Instead of pre-establishing a set of features, one can alternatively adopt a *data-driven* method that may extract a too large set, which can then be reduced in a second stage through feature selection. Following such approach, the image retrieval system proposed in [TV00] uses a cascade of filters to convert each image into a relatively large (46,875-dimensional) feature vector. A small subset of these features is selected through a boosting-based method. The idea is that few of these features respond in a highly selective way to a group of images or specific characteristic as, e.g., the light/dark/light region associated to the eyes and nose of a human face. We discuss now the boosting-based and other feature selection techniques.

As in [TV00], [Das01] used the AdaBoost algorithm [SS99] for selecting features. Both works use boosting as a filter method. In [SN99], boosting was integrated to wrapper methods. Wrappers are of interest because they can lead to improved accuracy when compared to filter methods. In [Ng98], an analysis of feature selection wrappers was presented and the *ordered-fs* wrapper algorithm was presented, which had improved asymptotic behavior when the number of irrelevant features is large compared to the relevant ones.

Our motivation to investigate filter and wrapper in this work can be summarized as follows. In [Das01], the boosting-based were compared to alternative wrapper algorithms and it was concluded they perform equivalently. However, there were no comparisons with other filter methods (e.g. [Hal99]). Therefore, we investigate how the boosting-based method of [TV00] compares with filter methods based on support vector machines (SVM) and information-gain [Hal99]. In terms of wrapper methods, the simulations in [Ng98] used only synthetic data. The ordered-fs algorithm was used with real-life (microarray) data in [XJK01], but no comparisons of ordered-fs with other methods were presented. Our experimental framework seems appropriate to evaluate some conclusions presented in [Ng98]. These comparisons will also allow to infer about what happens when we use a data-driven feature set, instead of a knowledge-based set. In other words, we evaluate how successfully we can mine relevant information in the domain of “beauty” estimation.

The paper is organized as follows. In Section 2 we briefly describe the dataset we will be working with. In Section 3 we discuss feature extraction methods and the image processing methods used in this work. Section 4 discusses the feature selection methods. The experimental results are presented in Section 5, followed by conclusions in Section 6.

## 2 USFS dataset

Beauty is a difficult concept to be learned by a machine. In this paper we deal with the scenic beauty of forestry images. The United States Forest Service (USFS) has a long-term interest on automatically obtaining a ranking of forest images according to their scenic beauty [Zea99]. The USFS dataset [Kea97] contains images labeled by people and was organized to facilitate experiments in this area. The ideal system should have performance similar to a person. However, there are several difficulties for achieving this goal. We are mainly concerned here with feature extraction and



Figure 1: Image that has highest score (107.13) in dataset (most “beautiful”).



Figure 2: Image that has lowest score (-125.04) in dataset.

selection [JZ97, JDM00, Hal99].

The USFS dataset contains 637 forestry images. The images were split into training and test sets, as will be discussed in details in Section 5. All the images were assigned a score according to their beauty. The maximum score, corresponding to the most “beautiful” image, is 107.13, while the minimum is -125.04. These two images are shown in Figures 1 and 2, respectively.

The images were collected in 6 different months. The season in which the picture was taken influences the beauty score as shown in Table 1. Each of the four seasons has approximately the same number of images. The images collected during the Winter (January) have the lowest score in average, while the ones collected during Summer (July and September) tend to present high scores. Fig. 3 shows an histogram indicating this asymmetry.

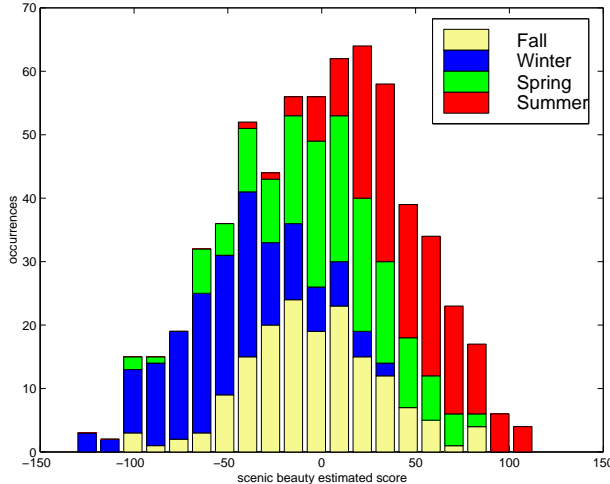


Figure 3: Stacked histogram of beauty scores per season.

Table 1: Occurrences of scores per month and associated statistics.

		Winter	Spring	Summer		Fall	
		Jan.	Apr.	Jul.	Sep.	Oct.	Nov.
Train	Occurrences (total = 478)	120	120	58	56	64	60
Train	Average score	-48.6	2.7	46.7	42.1	3.0	-8.4
Train	Minimum score	-125.0	-92.6	7.5	-37.2	-106.7	-97.8
Train	Maximum score	36.2	77.6	103.1	107.1	85.6	78.0
Test	Occurrences (total = 159)	40	40	20	20	20	19
Test	Average score	-52.7	6.5	47.9	40.0	0.7	24.1
Test	Minimum score	-124.3	-103.4	-21.1	-13.3	-73.9	-73.5
Test	Maximum score	16.5	77.5	99.3	103.4	53.5	37.6
Train+test	Occurrences (total = 637)	160	160	78	76	84	79
Train+test	Average score	-49.6	3.66	47.0	41.6	2.4	-12.2
Train+test	Minimum score	-125.0	-103.4	-21.1	-37.2	-106.7	-97.8
Train+test	Maximum score	36.2	77.6	103.1	107.1	85.6	78.0

### 3 Feature extraction

In this section we discuss feature extraction and its application to classification of forestry images according to their scenic beauty. We start by introducing some notation.

In supervised classification problems, one is given a training set  $\mathcal{T} = \{(x_1, y_1), \dots, (x_N, y_N)\}$  containing  $N$  *examples*. Each example  $(x, y)$  consists of an instance  $x \in \mathcal{X}$  and a label  $y \in \mathcal{Y} = \{1, \dots, K\}$ , where  $\mathcal{X}$  is the instance space and  $K \geq 2$  is the number of classes. Each instance  $x$  is a  $L$ -dimensional vector representing  $L$  features, and  $x_i \in \mathcal{X}_i$  its  $i$ -th feature. A *classifier* is a mapping  $F: \mathcal{X} \rightarrow \{1, \dots, K\}$  from instances to labels.

We assume that the instance  $x$  is obtained from some measure (“raw” data) (image pixels in our case) through a mapping that incorporates both the feature extraction and feature selection processes. The feature selection stage simply chooses a subset with  $S$  features from all  $2^L$  possible subsets.

Given that the knowledge-based features [Zea99] were available, we investigated two

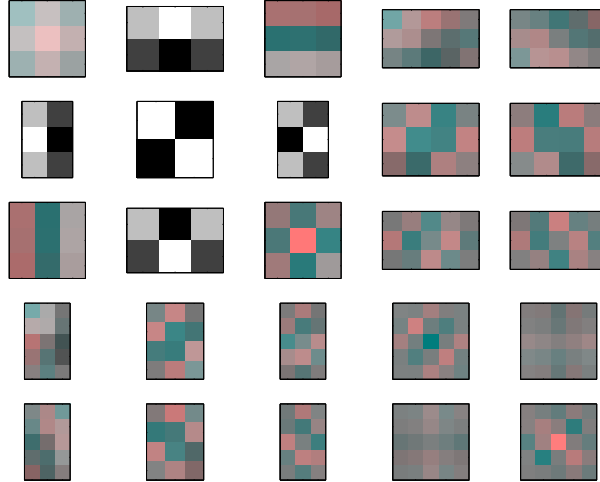


Figure 4: Filters used in the feature extraction process. For future reference, the filters are numbered according to a raster scanning (the first row shows filters 1 to 5 and so on).

other alternatives: Tieu & Viola’s [TV00] features (which we call here data-driven features) and raw pixels.

We first describe the processing to obtain the raw pixel representation. Raw pixels were also used in similar applications [EPPP01, CB02]. However, in most of the cases where pixels were found to be useful, it was assumed the object of interest (e.g. face) was located in the center of the image. It should be noted that our forestry images do not have such characteristic. Nevertheless, we used pixels as part of our exploratory research in this domain.

In the original USFS dataset, each image is stored in a high-resolution uncompressed file (PPM format,  $1536 \times 1024$  pixels, 3 bytes per pixel, 4MB file). In order to save storage space and reduce computational cost when processing the images, we performed the following steps: (a) eliminated dark border (see Figs. 1 and 2), (b) performed histogram equalization (using Matlab’s `histeq`) given that it improved results in [EPPP01], (c) saved images as high-quality compressed (JPEG format, using Matlab’s `imwrite` with  $Q = 95$ ) files, (d) downsampled by 10, such that the total number of features is  $L = 42,336$  ( $147 \times 96$  pixels, 3 bytes per pixel. Chosen to be close to the 46,875 features used in [TV00]).

The data-driven feature extraction method used is the one presented in [TV00]. This algorithm adopts three filtering stages. Each stage uses one among 25 empirically pre-selected filters, which are shown in Fig. 4. Each RGB (red, green and blue) color channel is filtered individually. On each stage, the output image is rectified (taking the absolute value of each pixel) and downsampled by 2. After the last stage, a single-valued feature is obtained by summing the output image pixels for the given color component (averaging was used in this work). Therefore, the total number of features is three times the different filtering schemes:  $3 \times 253=46,875$ . A given sequence of filters  $x$ ,  $y$  and  $z$  is represented as  $(x, y, z)$ . A given feature is represented as  $(x, y, z, c)$ , where  $c$  indicates the color channel.

Our third set of features is composed by the 65 features used in [Zea99], which were inspired by knowledge. In summary, these features are the number of long and short lines (obtained with an edge detection algorithm), the compression ratio and, for

each of the RGB color channels, 10-bins histogram, sharpness, standard deviation, entropy and fractal dimension. Histograms (10-bins) were also computed for the colors brown and yellow.

## 4 Feature selection

Feature selection methods can often be classified into two groups: *filter* and *wrapper* [Hal99, BL97, KJ97]. The latter consists of methods that select features using the algorithm  $\mathcal{L}$  that will be applied to learn classifier  $F$ . The filter methods evaluate the worth of features by using heuristics based on characteristics of the data and is independent of  $\mathcal{L}$ .

### 4.1 Filter methods

One of the simplest but yet popular filter method for feature selection is the information gain (also known as mutual information) [Hal99]. A distribution  $p(y)$  on the labels is estimated using the training set  $\mathcal{T}$ , and the associated random variable  $Y$  has entropy given by  $H(Y) = -\sum_{y \in \mathcal{Y}} p(y) \log_2 p(y)$ . After observing feature  $x_i$ , with  $i \in \{1, \dots, L\}$ , the training set can be partitioned according to  $x_i$ . Distributions  $p(x_i)$  and  $p(y|x_i)$  are estimated using  $\mathcal{T}$ . The entropy of  $Y$  conditioned on observing the random variable  $X_i$  is  $H(Y|X_i) = -\sum_{x_i \in \mathcal{X}} p(x_i) \sum_{y \in \mathcal{Y}} p(y|x_i) \log_2 p(y|x_i)$ . The information gain  $I(X_i; Y)$  is given by  $I(X_i; Y) = H(Y) - H(Y|X_i)$ . The method consists on calculating  $I(X_i; Y)$ , for  $i = 1, \dots, L$  and then selecting the  $S$  features with highest information gain. This is a very simple criterion and we use it as a baseline for comparison.

A method that does not select features independently is the boosting-based method proposed in [TV00]. Similar boosting-based feature selection methods were also investigated by Das [Das01]. The algorithm works as follows. The number of features  $S$  is specified in advance and, for each round  $i = 1, \dots, S$  of boosting (we used AdaBoostM1 [Wek] in our experiments), a feature  $f_i$  is chosen  $f_i = x_j$ , where  $x_j$ ,  $j \in \{1, \dots, L\}$  is the feature associated to the decision stump that minimized a weighted (the weights of hard-to-classify examples are increased by boosting) misclassification rate over  $\mathcal{T}$ . For round  $i$ , the previously selected features  $f_1, \dots, f_{i-1}$  are not considered.

It should be noted that several learning algorithms can perform feature selection as a by-product of learning, as e.g. C4.5 decision trees. In general, given that a learner  $\mathcal{L}$  outputs hypothesis using only a subset of the input features, it can be used as a filter for other learning methods. For example, the conventional implementation of boosted decision stumps [SS99] will output at most  $T$  distinct features after  $T$  rounds of boosting. However, excluding previously selected features from further consideration as in [TV00, Das01], makes easier to achieve the desired number of features.

Given the success of SVM in high-dimensional spaces, we decided to investigate its performance on ranking features. A linear SVM is equivalent to a perceptron. The idea is to use techniques similar to the feature selection in linear regression (see e.g. [HTF01], page 55). Here, we used a simple approach that assumes the magnitude of the perceptron weights should correlate well with the relevance of the associated features. The selected  $S$  features are the ones corresponding to the largest weights of the perceptron derived from SVM training.

## 4.2 Wrapper methods

The conventional wrapper method for feature selection uses the induction algorithm  $\mathcal{L}$  that will be adopted for designing classifier  $F$ , to guide the search for the best subset of features. Usually, cross-validation is adopted, and for each feature subset examined,  $\mathcal{L}$  is invoked  $V$  times in an  $V$ -fold cross-validation using the training set  $\mathcal{T}$ . Exploring all  $2^L$  possible subsets is usually unfeasible and a heuristic search is adopted. One simple search strategy, called greedy hill climbing [Hal99], considers local changes to the current feature subset. A local change often means the addition or deletion of a single feature from the subset. When the algorithm considers only additions to the feature subset it is called *forward selection*, and considering only deletions is known as *backward selection*. In both cases, once a change is accepted, it is never reconsidered. More sophisticated algorithms as *best first* search allows backtracking along the search path. We used forward selection in our experiments.

Now we present a brief summary of [Ng98]. The paper presents theoretical results that bring some light to fundamental questions as “Why performing feature selection?” and “How the performance of wrappers scale with the number of irrelevant features?”. Following the insights from the theoretical bounds, the paper proposes a wrapper algorithm for feature selection called ordered-fs.

Ng’s analysis assumes the wrapper uses held-out data  $\mathcal{H}$  consisting of a fraction  $\gamma$  of  $\mathcal{T}$  (usually a stratified fraction and  $\gamma = 0.33$ ) instead of cross-validation. Let  $\mathcal{T}' = \mathcal{T} - \mathcal{H}$  be the new training data. The author explains that in terms of asymptotic behavior, the results with held-out data should not be different from the ones with cross-validation or *leave-one-out* [Ng98]. In practice, cross-validation may be advantageous when compared to using held-out data in spite of an increase in computations. Another simplification is to assume the search explores all  $2^L$  possible feature subsets. Under these two assumptions, the *standard-wrap* wrapper will invoke  $\mathcal{L}$  for all  $2^L$  possible subsets, using  $\mathcal{T}'$  for training and  $\mathcal{H}$  for estimating the accuracy when using a subset. In other words, for  $s = 1, \dots, L$ , standard-wrap will pick the classifier  $F_s$  and feature subset  $\mathcal{A}_s$  that minimized the error  $\xi_{\mathcal{H}}(F_s)$  over the held-out data. The algorithm outputs the best classifier  $\hat{F} = \arg \min_{s=0, \dots, L} \xi_{\mathcal{H}}(F_s)$  (and its associated feature subset). Note that  $\mathcal{H}$  is used for both choosing the best  $F_s$  among the ones derived from all possible subsets with  $s$  features and for selecting  $S$ .

Let  $\xi_{\infty}(F_s)$  be the best generalization error achievable by any classifier using exactly  $s$  features. One can bound  $\xi_{\infty}(F_s)$  given estimates obtained with finite data  $\xi_{\mathcal{H}}(F_s)$  and, therefore, bound the generalization error  $\xi_{\infty}(\hat{F})$  of the classifier obtained with standard-wrap. Define

$$c \stackrel{\text{def}}{=} \frac{s_{VC}}{(1-\gamma)N} \left( \ln \frac{2(1-\gamma)N}{s_{VC}} + 1 \right),$$

where  $s_{VC}$  is the VC dimension when using  $s$  features (see e.g. [Ng98, HTF01] for definitions). We assume that there are enough training examples, such that  $2(1-\gamma)N > s_{VC}$ . Then, with probability at least  $1 - \delta$ , the generalization error  $\xi_{\infty}(\hat{F}_{\text{standard}})$  of the classifier obtained with standard-wrap is bounded by

$$\begin{aligned} \xi_{\infty}(\hat{F}_{\text{standard}}) &\leq \min_{0 \leq s \leq L} \left\{ \xi_{\infty}(F_s) + 4 \sqrt{c + \frac{1}{(1-\gamma)N} \ln \frac{18}{\delta}} \right\} \\ &+ 2 \sqrt{\frac{1}{2\gamma N} \left( (L+2) \ln 2 + \ln \frac{1}{\delta} \right)}. \end{aligned} \quad (1)$$

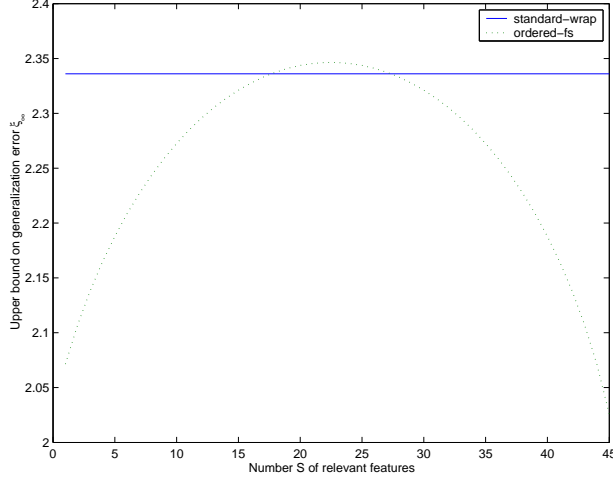


Figure 5: Comparison between standard-wrap and ordered-fs in terms of upper bound on  $\xi_\infty(\hat{F})$  for different number  $S$  of relevant features.

The ordered-fs algorithm consists of a simple modification of standard-wrap. It uses the held-out data  $\mathcal{H}$  only for choosing  $S$ , but selects the classifiers  $F_s$  as the ones that minimize the error  $\xi_{\mathcal{T}'}(F_s)$  over the training data  $\mathcal{T}'$ . With probability at least  $1 - \delta$ , the generalization error of the classifier obtained with ordered-wrap  $\xi_\infty(\hat{F}_{ordered})$  is bounded by

$$\begin{aligned} \xi_\infty(\hat{F}_{ordered}) \leq & \min_{0 \leq s \leq L} \left\{ \xi_\infty(F_s) + 4\sqrt{c + \frac{1}{(1-\gamma)N} \ln \frac{18 \binom{L}{s}}{\delta}} \right\} \\ & + 2\sqrt{\frac{1}{2\gamma N} \ln \frac{4(L+1)}{\delta}}. \end{aligned} \quad (2)$$

The second term of Eq. (1) is  $O\left(\sqrt{\frac{L}{\gamma N}}\right)$ , while the ordered-fs reduces it to  $O\left(\sqrt{\frac{\ln L}{\gamma N}}\right)$  at the expense of increasing the first term. As mentioned in [Ng98], when the number of relevant features is relatively large (say  $S \approx 0.5L$ ), the standard-wrap can do better than ordered-fs. This is illustrated in Fig. (5), which was obtained by setting  $N = 630$ ,  $\gamma = 0.33$ ,  $L = 45$ ,  $\delta = 0.1$  and  $\xi_\infty(F_s) = 0.2$ . It was assumed a linear SVM as classifier, such that  $s_{VC} = S + 1$  (valid for any linear discriminant function).

## 5 Experimental results

### 5.1 Results without feature selection

In [Zea99], the 637 images were divided into 3 subsets according to their score  $z$ : *low* ( $z < -49$ ), *high* ( $z > 45$ ) and *medium* (otherwise) scenic beauty, each with 110, 104 and 423 examples, respectively. This split was based on the standard deviation of  $z$ . Note that the most popular class corresponds to 66% of the data. When the

Table 2: Error rate (%) when using the 65 knowledge-based features [Zea99] and  $K = 3$  classes (our results are consistent with [Zea99]).

Classifier	Train	Test
Decision stump (attribute red-histogram, 4-th bin)	32.6	36.5
Naive Bayes	66.5	67.3
C4.5 decision tree	6.5	45.3
one-against-all SVM's (C=1, linear, 194 support vectors)	32.6	36.5
one-against-all SVM's (C=1, Gaussian (RBF) kernel, $\gamma = 1$ )	27.0	<b>35.8</b>
one-against-all SVM's (C=1, Gaussian (RBF) kernel, $\gamma = 10$ )	0.2	<b>35.8</b>

test set obeys this proportion, it is trivial to achieve around 34% of error rate by always choosing the most popular *medium* class.

The experiments in [Zea99] used 4-fold cross-validation. Typically, the first split (fold) led to slightly worst performance, and we chose this set for testing the classifiers in our experiments. This split corresponds to a training and testing sets with 478 and 159 instances, respectively. There are 29, 101 and 29 images of classes high, medium and low beauty, respectively. Therefore, always choosing the medium class leads to 36.5% of error rate.

With  $K = 3$  classes, the best results (average over 4-fold cross-validation) in [Zea99] were 33% and 33.4% of misclassification rate for SVM and independent component analysis (ICA), respectively. PCA and decision trees achieved 36.6% and 43%, respectively. Table 2 shows some results we obtained, which were consistent with the one shown in [Zea99].

Table 2 shows that the algorithms tested here had performance similar or worse than simply choosing the most popular class. Given the difficulty of this task, we decided to label the dataset using only two classes. This was performed using the median (equal to 1.07) of score  $z$  as a threshold. For testing, we use the same set with 159 instances. Some results with  $K = 2$  classes are shown in Table 3. We did not use cross-validation to tune parameters, and we simply present results for some configurations. Therefore, it is questionable to interpret the test error as an indication of generalization capability.

Given the clear superiority of SVM over the other learning algorithms, we decided to exclusively use SVM hereinafter. Using SVM as classifiers, Tables 4 and 5 show the results obtained with raw pixels and data-driven features, respectively. As expected, the pixels are outperformed by both the knowledge-based and data-driven features. It can be seen that when using pixels and data-driven features, the SVM's are badly overfitting the training data.

## 5.2 Results with feature selection using filters methods

Here we evaluate three filter methods for feature selection: information-gain, boosting and SVM. We used the data-driven features [TV00] for the experiments. We chose the following five values for the number  $S$  of features to be selected: 10, 65,  $10^2$ ,  $10^3$ ,  $10^4$ . Selecting  $10^3$  and  $10^4$  features with the boosting-based method was not possible due to its high computational cost. For each of these five values, all three filters methods were used to generate the training and test data for SVM classifiers. Several configurations of SVM were tested: polynomial and Gaussian kernels, various  $C$ , etc. For each  $S$ , we picked the best result on the test set to obtain both Figs. 6 and 7. Figs. 6 shows that using the SVM-based filter to choose  $10^3$  features led to the best result: misclassification rate of 18.9%. This is the same result obtained with knowledge-based features. It corresponds to a 2.5% absolute

Table 3: Error rate (%) when using the 65 knowledge-based features [Zea99] and  $K = 2$  classes.

Classifier	Train	Test
Decision stump (attribute blue-histogram, 1-st bin)	33.3	34.0
Naive Bayes	34.5	37.1
C4.5 decision tree (default parameters, size of the tree = 75)	7.9	34.0
10 boosted (AdaBoostM1) stumps	23.0	28.3
50 boosted (AdaBoostM1) stumps	14.2	25.2
100 boosted (AdaBoostM1) stumps	13.0	25.8
200 boosted (AdaBoostM1) stumps	10.0	25.2
1000 boosted (AdaBoostM1) stumps	0.2	25.2
Boosted Naive Bayes (22 iterations of AdaBoostM1)	24.3	26.4
SVM (C=1, linear, 309 support vectors)	21.8	25.2
SVM (C=10, linear, 261 support vectors)	18.6	23.3
SVM (C=100, linear, 234 support vectors)	17.6	<b>18.9</b>
SVM (C=1000, linear, 222 support vectors)	18.0	22.0
SVM (C=1, 2-nd order homogeneous polynomial, 245 support vectors)	13.8	19.5
SVM (C=10, 2-nd order homogeneous polynomial, 222 support vectors)	6.7	<b>18.9</b>
SVM (C=100, 2-nd order homogeneous polynomial, 192 support vectors)	1.5	26.4
SVM (C=10, 2-nd order non-homogeneous polynomial, 217 support vectors)	6.7	<b>18.9</b>
SVM (C=0.1, Gaussian (RBF) kernel, $\gamma = 1$ , 419 support vectors)	31.4	28.9
SVM (C=1, Gaussian (RBF) kernel, $\gamma = 1$ , 362 support vectors)	13.4	20.1
SVM (C=10, Gaussian (RBF) kernel, $\gamma = 1$ , 302 support vectors)	2.3	23.3
SVM (C=1, Gaussian (RBF) kernel, $\gamma = 0.5$ , 340 support vectors)	16.9	20.7
SVM (C=1, Gaussian (RBF) kernel, $\gamma = 2$ , 393 support vectors)	7.7	20.1
SVM (C=1, Gaussian (RBF) kernel, $\gamma = 10$ , 475 support vectors)	0.0	23.0

Table 4: Error rate (%) when using raw pixels as features and  $K = 2$  classes.

Classifier	Train	Test
SVM (C=1, linear)	0	28.3
SVM (C=1, 4-th order homogeneous polynomial)	0	27.7
SVM (C=1, 5-th order homogeneous polynomial)	0	<b>25.8</b>

Table 5: Error rate (%) when using Tieu & Viola’s [TV00] data-driven features and  $K = 2$  classes.

Classifier	Train	Test
SVM (C=0.1, linear)	0	<b>21.4</b>
SVM (C=10, linear)	0	<b>21.4</b>
SVM (C=100, linear)	0	<b>21.4</b>
SVM (C=1, 2-nd order homogeneous polynomial)	0	24.5

decrease on error when compared to classification without feature selection. When increasing the number of features to  $10^4$ , the SVM-based filter method combined with the SVM classifier, overfitted the training data.

Now we inspect the features that were selected by the three methods. Table 5.2 shows the first 10 selected features. The information-gain method selects features independently and they seem highly redundant. The method based on information-gain always selected the filter number 11 in Fig. 4 for the first filtering stage and the green color channel. This makes sense given that filter 11 should respond well to trees and the amount of green is correlated with the beauty score as indicated in Fig. 3. The SVM-based method is the only one that includes all three color channels among its top-10 features. As boosting performs a greedy search, it may be the case that 10 rounds were not enough to detect the eventual discrimination capability of the red color channel.

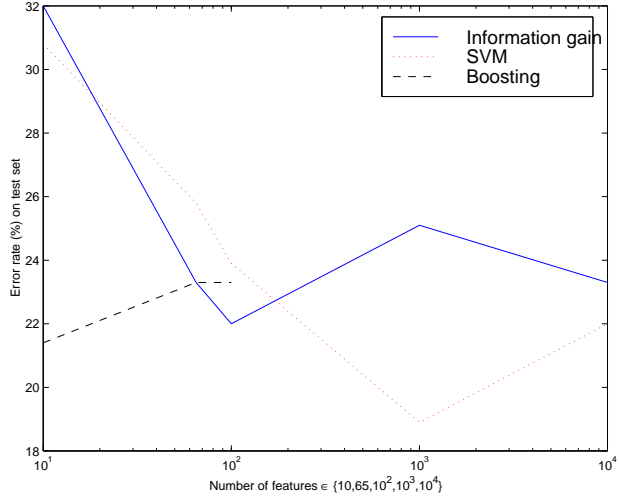


Figure 6: Error on test set for different feature selection filter methods with the data-driven features [TV00]. This is the result obtained by the best SVM configuration on the test set. Selecting 10<sup>3</sup> and 10<sup>4</sup> features with the boosting-based method was not possible due to its high computational cost.

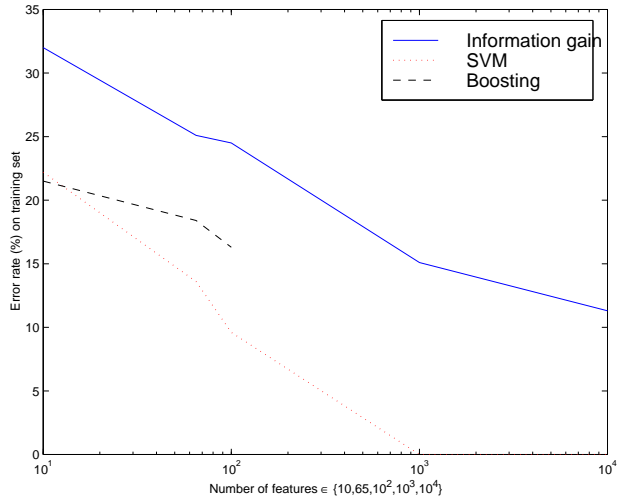


Figure 7: Error on training set for different feature selection filter methods with the data-driven features [TV00]. This is the result obtained by the best SVM configuration on the test set. Selecting 10<sup>3</sup> and 10<sup>4</sup> features with the boosting-based method was not possible due to its high computational cost.

Table 6: First 10 features selected by the filter methods, where  $(x,y,z,c)$  indicates the three filters numbered according to Fig. 4 and  $c$  is the RGB color channel (e.g. 2 corresponds to green).

"Relevance"	Information-gain	SVM	Boosting
1	(11,16,4,2)	(4,2,11,3)	(11,11,24,2)
2	(11,6,13,2)	(1,9,7,1)	(1,1,2,2)
3	(11,9,12,2)	(18,4,17,3)	(4,8,18,3)
4	(11,6,14,2)	(16,4,25,3)	(15,1,13,2)
5	(11,11,14,2)	(16,4,21,3)	(22,10,2,3)
6	(11,6,4,2)	(18,21,25,3)	(8,7,16,2)
7	(11,16,23,2)	(18,21,21,3)	(15,6,3,2)
8	(11,21,2,2)	(18,6,6,1)	(1,16,1,3)
9	(11,16,18,2)	(3,19,5,1)	(11,17,5,2)
10	(11,21,24,2)	(11,8,17,2)	(7,17,13,2)

Table 7: Results with wrapper methods. Each column represents an initial number  $S_0$  of features selected by the boosting-based filter method, and each entry  $a/b/c$  corresponds to error (%) on training data, error (%) on test data and number  $S$  of features selected by the wrapper, respectively.

Wrapper	10	65	100
standard-wrap	25.9 / <b>20.1</b> / 4	25.3 / 30.8 / 5	23.4 / 30.2 / 5
Ng's ordered-fs	20.7 / 21.4 / 8	16.7 / 24.5 / 55	15.7 / 21.4 / 76
cross-validation wrapper	20.7 / 22.6 / 6	20.1 / 22.6 / 11	15.7 / 23.9 / 13

### 5.3 Results with feature selection using wrapper methods

We used the data-driven features for the experiments with wrappers and the classifier was a linear SVM with  $C = 1$ . Due to a high computational cost, we did not consider all 46,875 features, but as in [XJK01], we used the best  $S_0$  features selected by the boosting-based filter method, with  $S_0 = 10, 65, 100$ . The obtained results are shown in Table 5.3. It can be seen that standard-wrap selects a smaller number of features (4 or 5) than ordered-fs in this problem. While standard-wrap achieves the best result, it overfits the held-out data when  $S_0$  is 65 and 100. A more popular alternative than using held-out data for wrappers design is to use cross-validation [Hal99]. Results using wrapper based on 10-fold cross-validation are also shown in Table 5.3.

## 6 Conclusions

We presented an experimental investigation about feature extraction and selection for automatically classifying forestry images according to their "beauty". This is a challenging problem, given that the concept of beauty is not well-defined. We verified that a data-driven approach was able to identify relevant features and perform as well as a knowledge-based one.

The SVM-based filter method allowed to achieve 18.9% of misclassification rate using  $10^3$  features, which is also the best result obtained with the 65 knowledge-based features. The wrapper method led to 21.4% of error using only 4 data-driven features.

The main problem of using wrapper methods is their computational cost. Because of that, several interesting simulations were avoided. For example, the wrapper methods should be allowed to explore all features, instead of using a subset chosen by filter methods.

The ordered-fs algorithm has a better asymptotic behavior than standard-wrap when the number of irrelevant features is large. However, due to the cost of running wrappers, if there are many irrelevant features it seems more convenient to use a filter method to pre-select some of them, as done here and in [XJK01]. In such scenarios, it is not clear whether or not the asymptotically better behavior of ordered-fs can be identified in practice.

Unfortunately, this domain did not allow to properly compare the filter methods. It would be interesting to evaluate the SVM-based and the boosting-based methods on other datasets. This is subject for a future work.

## Acknowledgments

Thanks go to Prof. Charles Elkan (UCSD) for his guidance through the development of this project, Prof. Joseph Picone (Mississippi State Univ.) for providing the USFS dataset, Prof. Mark Hall (Waikato Univ.) for lessons on using Weka for feature selection and Prof. Nuno Vasconcelos (UCSD) for exchanging ideas about feature extraction for image retrieval.

## References

- [BL97] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, pages 245–271, 1997.
- [CB02] V. Castelli and L. Bergman. *Image Databases - Search and Retrieval of Digital Imagery*. John Wiley and Sons, 2002.
- [Das01] S. Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *ICML*, 2001.
- [EPPP01] T. Evgeniou, M. Pontil, C. Papageorgiou, and T. Poggio. Image representations for object detection using kernel classifiers. 2001.
- [Hal99] M. Hall. *Correlation-based feature selection for machine learning*. PhD thesis, University of Waikato, 1999.
- [HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, 2001.
- [JDM00] A. Jain, R. Duin, and J. Mao. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 4–37, 2000.
- [JZ97] A. Jain and D. Zongker. Feature selection: evaluation, application and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 153–158, 1997.
- [Kea97] N. Kalidindi and et al. Scenic beauty estimate of forestry images. In *IEEE Southeastcon*, pages 337–339, 1997.
- [KJ97] R. Kohavi and G. John. Wrapper for feature subset selection. *Artificial Intelligence*, pages 273–324, 1997.
- [Ng98] A. Ng. On feature selection: Learning with exponentially many irrelevant features as training examples. In *International Conference on Machine Learning*, 1998.
- [SN99] M. Sebban and R. Nock. Contribution of boosting in wrapper models. In *PKDD*, pages 214–222, 1999.
- [SS99] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, pages 297–336, 1999.
- [TV00] K. Tieu and P. Viola. Boosting image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 228–235, 2000.
- [Web99] A. Webb. *Statistical Pattern Recognition*. Oxford University Press Inc., 1999.

- [Wek] <http://www.cs.waikato.ac.nz/ml/weka>.
- [XJK01] R. Xing, M. Jordan, and R. Karp. Feature selection for high-dimensional genomic microarray data. In *ICML*, 2001.
- [Zea99] X. Zhang and et al. Scenic beauty estimation using independent component analysis and support vector machines. In *IEEE Southeastcon*, 1999.