Journal of Machine Learning Research 17 (2016) 1-39 Submitted 1/15; Revised 11/15; Published 10/16

Jointly Informative Feature Selection Made Tractable by Gaussian Modeling

Leonidas Lefakis* LEONIDAS.LEFAKIS@ZALANDO.DE
Zalando Research
Zalando SE
Berlin, Germany

François Fleuret FRANCOIS.FLEURET@IDIAP.CH
Computer Vision and Learning group
Idiap Research Institute
Martigny, Switzerland

Editor: Amos Storkey

Abstract

We address the problem of selecting groups of jointly informative, continuous, features in the context of classification and propose several novel criteria for performing this selection. The proposed class of methods is based on combining a Gaussian modeling of the feature responses with derived bounds on and approximations to their mutual information with the class label. Furthermore, specific algorithmic implementations of these criteria are presented which reduce the computational complexity of the proposed feature selection algorithms by up to two-orders of magnitude. Consequently we show that feature selection based on the joint mutual information of features and class label is in fact tractable; this runs contrary to prior works that largely depend on marginal quantities. An empirical evaluation using several types of classifiers on multiple data sets show that this class of methods outperforms state-of-the-art baselines, both in terms of speed and classification accuracy.

Keywords: feature selection, mutual information, entropy, mixture of Gaussians

1. Introduction

Given a collection of data in $\mathbb{R}^D$ it is often advantageous to reduce its dimensionality by either extracting (Hinton and Salakhutdinov, 2006) or selecting (Guyon and Elisseeff, 2003) a subset of $d \ll D$ features, which carry "as much information as possible". The motivation behind this dimensionality reduction can be either to control over-fitting by reducing the capacity of the classifier space or to improve the computational overhead by reducing the optimization domain. It can also be used as a tool to facilitate the understanding or the graphical representation of high-dimensional data.

In the present work we focus on the selection, rather than extraction, of features. In general feature selection methods can be divided into two large families: Techniques from the first group, *filters*, are predictor agnostic as they do not optimize the selection of features

∗. Part of this research was conducted while at the Computer Vision and Learning group of the Idiap Research Institute

for a specific prediction method. They are usually based on classical statistics or information theoretic tools; the novel methods we propose in this article belong to this category. Techniques from the second group, *wrappers*, choose features to optimize the performance of a certain predictor. They usually require the retraining of the predictor at each step of a greedy search. Consequently they are typically computationally more expensive than filters. Furthermore, as the selected features are tailored to a specific predictor, they often do not work well with other families of predictors.

In the following we present a filter approach to feature selection in the context of classification based on the maximization of the mutual information between the selected features and the class to predict. The use of mutual information as a criterion for feature selection has been extensively studied in the literature, and can be motivated in the context of classification by Fano's inequality

$$H(Y|\hat{Y}) \leq 1 + P(e)\log\left(|\mathcal{Y}|\text{-}1\right)$$

where $Y, \hat{Y}$ are the true and predicted labels respectively, while e is the event that $Y \neq \hat{Y}$. Combining the above with the data processing inequality (specifically the chain $Y \rightarrow X \rightarrow \hat{Y}$) results in the following lower bound on the probability of an incorrect classification P_E by

$$P(e) \geq \frac{H(Y) - I(X;Y) - 1}{(|\mathcal{Y}|\text{-}1)}$$

where $H(Y)$ is the entropy of the class prior and $I(X;Y)$ is the mutual information between the output (Y) and input (X) features.

An important issue that arises in this context, is that of the joint "informativeness" of the selected features. Though wrappers by their very nature tend to select features which are jointly informative, this issue is only partially addressed for filters due to the resulting computational complexity and need for very large training sets. Most existing methods (Peng et al., 2005; Fleuret, 2004; Hall and Smith, 1999; Liu and Yu, 2003) typically compromise by relying on the mutual information between individual features or very small groups of features (pairs, triplets) and the class. We argue however, that rather than compromising on the joint behavior of the selected features, it is preferable to accept a compromise on the density model, which will allow to analyze this joint behavior in an efficient manner.

In a classification task with continuous features context, if we aim at taking into account the joint behavior of features, a Gaussian model is a natural choice. This unfortunately leads to a technical difficulty: If such a model is used for the conditional distributions of the features given the class then the non-conditioned distribution is a mixture of Gaussians and its entropy has no simple analytical form. There is extensive prior work on the problem of approximating the entropy of a mixture of Gaussians (Hershey and Olsen, 2007), but most of the existing approximations are too computationally intensive to be used during an iterative optimization process which requires the estimation of the mutual information of a very large number of subsets of features with the class to predict.

We propose here an alternative approach wherein we derive both bounds on and approximations to the Mutual Information – and maximize these quantities *in lieu* of the true mutual information. We also propose specific algorithmic implementations that rely

on updating the inverses of covariance matrices iteratively instead of computing them from scratch drastically reducing the computational cost during the optimization of the selected feature set.

2. Related Works

When selecting d features from a pool of D candidates in the context of a classification task, it does not suffice to select features independently informative with respect to the class. When such greedy strategies are employed the risk of acquiring redundant, or even identical, features increases. Thus, it is also important that these features exhibit low redundancy between them: *joint* informativeness is at the core of feature selection.

As mentioned in the introduction, wrappers, due to their very nature, address this issue by creating subsets of features that perform well when combined with a specific predictor. Examples of such methods include iteratively training a SVM , removing at each iteration the features with the smallest weights (Guyon et al., 2002) or employing Adaboost in connection with decision stumps to perform feature selection (Das, 2001). Other wrapper methods impose sparsity on the resulting predictor thus implicitly performing feature selection, for instance by using a Laplacian prior to perform sparse logistic regression (Cawley et al., 2006), casting a l_0 regularized SVM as a mixed integer programming problem (Tan et al., 2010), or imposing sparsity via a l_1-norm regularizer (Argyriou et al., 2008) or l_1-clipped norm (Xu et al., 2014). Such wrappers, that train predictors only once, tend to be much faster, however like other wrappers they tend not to generalize well across predictors.A wrapper approach that shares similarities with the algorithm proposed here, forward regression (Das and Kempe, 2011) iteratively augments a subset of features to build a linear regressor which is near-optimal in a least-squared error sense.

In the context of filters, the simplest methods are those that calculate statistics on individual features and then rank these features based on these values, keeping the d features of highest rank. Examples of such statistics are Fisher score, mutual information between the feature and the class (information gain), χ^2 etc. Though quick to compute, such approaches typically result in large feature redundancy and sub-optimal performance.

A slightly more computationally complex approach, the RelieFF algorithm (Robnik-Šikonja and Kononenko, 2003), looks at individual features assigning a score by randomly selecting samples and calculating for that feature and for each sample the difference in distance between the random sample and the nearest sample of the same class, dubbed "nearest hit", and the random sample and the nearest sample of a different class, dubbed "nearest miss". Despite looking at features in isolation, it has been shown to perform well in practice.

In the work on mRMR feature selection (Peng et al., 2005) the authors attempt to address the issue of redundancy by selecting feature of maximum relevance and minimum redundancy, that is features with high mutual information with the class and low mutual pairwise information with the remaining selected features, thus selecting features that are not pairwise redundant. Quadratic programming feature selection (Rodriguez-Lujan et al., 2010) casts the feature selection as an optimization task and can be used in conjunction with a number of similarity measure. When combined with mutual information it resembles

mRMR though it provides a ranking of features as opposed to the greedy selection process of mRMR.

Another approach (Vasconcelos, 2003) based on mutual information attempts to diversify the conditional distributions $p_{X|Y}$ by greedily choosing features that maximize the Kullback-Leibler divergence between the conditionals and the prior p_X. However only the marginal distributions pertaining to individual features are used and as such no joint informativeness of features is exploited.

The *FCBF* algorithm (Liu and Yu, 2003) uses symmetrical uncertainty $\frac{I(X;Y)}{H(X)+X(Y)}$ as a quantitative criterion and adds features to a pool based on a novel concept of predominant correlation, namely that the feature is more highly correlated with the class than any of the features already in the pool. *CFS* (Hall and Smith, 1999) similarly combines symmetric uncertainty with Pearson's correlation to add features exhibiting low correlation with the features already in the pool.

Redundancy may also be addressed via the concept of a Markov Blanket (Margaritis, 2009). The Markov blanket of a variable X is defined as the set of variables S such that X is independent of the remaining variables $D \setminus (S \cup X)$ given the values of the variables in S. Based on this concept, the authors in (Fleuret, 2004) select features that have high mutual information with the class when conditioned on one of the features already in the pool. The resulting algorithm is suitable only for binary data. Here however we explicitly address the problem of feature selection in a continuous domain.

Closely related, at least conceptually, to the work presented here is prior work (Torkkola, 2003) which similarly attempts to find features that are jointly informative by resorting to a Gaussian modeling. In that work however the aim is feature extraction and the mutual information is used as an objective to guide a gradient ascent algorithm.

Another promising line of work is that of (Song et al., 2012) which avoids density estimation necessary in mutual information based approaches by considering the Hilbert-Schmidt Independence Criterion. The authors show how to obtain unbiased estimates of the HSIC quantity. Furthermore the method can be kernelized thus allowing for the discovery of dependencies in high-(possibly infinite) dimensional feature space. The Hilbert-Schmidt Independence Criterion has also been used in conjunction with l_1-norm regularization (Yamada et al., 2014).

Finally, we note a family of feature selection algorithms, which have become very popular in recent years, based on spectral clustering. In such approaches, features can be selected based on their influence on the affinity graph Laplacian (Jiang and Ren, 2011), or by analyzing the spectrum of the Laplacian matrix (Wolf and Shashua, 2005).

3. Feature Selection Criteria

We propose two novel criteria which characterize the informativeness of a set of features in a classification context using their mutual information with the class under a Gaussian model of the features given the class. While this approach is conceptually straight-forward, it requires the evaluation of the entropy of a mixture of Gaussians for which no closed-form expression is available.

Our first approach, dubbed "Gaussian compromise" and described in § 3.2.1, uses the entropy of a single Gaussian of same expectation and variance as the mixture to obtain an upper bound on, and subsequently an approximation to, the true entropy.

Table 1: Notation

$F = \{X_1, X_2, \ldots, X_D\}$ the set of candidate features
X_j a single feature
Y the class label
S a subset of F
S_{n-1} the set of features selected up until iteration n
Σ_S the covariance matrix of the features in S
$\Sigma_{j,S}$ the covariance vector of feature X_j and the features in S
$\sigma^2_{i,j}$ the covariance of features X_j and X_i
σ^2_i the variance of feature i
Σ^y_S the variance of the features in S conditioned on $Y = y$
$\sigma^2_{j\|S}$ the variance of feature X_j conditioned on the value of the features in S
f_y the density of a normal approximation of the class conditional distributions
f^* the Gaussian approximation of the joint law $\sum_y p_y f_y$
p_y the prior on the class variable Y

Our second approach, described in § 3.2.2 is based on a decomposition of the mutual information, in the binary class case, as a sum of Kullback-Leibler divergence terms, which can be efficiently approximated. The n-class case is addressed by averaging the obtained quantity over the one-vs-all sub-problems.

3.1 Mutual Information and the Gaussian Model

Given a continuous variable X and a finite variable Y, their mutual information is defined as

$$I(X;Y) = H(X) - H(X|Y) \tag{1}$$

$$= H(X) - \sum_y H(X|Y=y)P(Y=y). \tag{2}$$

Using a Gaussian density model for continuous variables is a natural strategy, due in part to the simplicity of its parametrization, and to its ability to capture the joint behavior of its components. Moreover, the entropy of a n-dimensional multivariate Gaussian $X \sim \mathcal{N}(\mu, \Sigma)$ has a simple and direct expression, namely

$$H(X) = \frac{1}{2}\log(|\Sigma|) + \frac{n}{2}\left(\log 2\pi + 1\right).$$

3.2 Bounds on the Mutual Information and the Entropy

Estimating the mutual information as defined in equation (2) requires the estimation of the entropy of both the conditional distributions $X|Y = y$ for all y, and that of X itself. If we model the former with Gaussian distributions, the latter is a mixture of Gaussian distributions, the entropy of which has no simple analytic form.

We propose to mitigate this problem by deriving upper bounds and approximations with tractable forms. Let $f_y, y = 1, \ldots, C$ denote Gaussian densities on $\mathbb{R}^D$, p_y a discrete distribution on $\{1, \ldots, C\}$, and f^* the Gaussian approximation of the joint law

$$f = \sum_y f_y p_y,$$

that is the Gaussian density of same expectation and covariance matrix as the mixture. Let Y be a random variable of distribution p_y and X a continuous random variable with conditional distributions $\mu_{X|Y=y} = f_y$.

3.2.1 Gaussian Compromise Criterion

As mentioned, we propose here to use an approximation of $H(f)$ based on the entropy of $H(f^*)$. While modeling f as a Gaussian is not consistent with the Gaussian models of the conditioned densities, estimating the mutual information with it still has all the important properties one desires for continuous feature selection:

- It captures the information content of individual features, since adding a non informative feature would change by the same amount all the terms of equation (2).
- It accounts for redundancy, since linearly dependent features would induce a small determinant of the covariance matrix and by extension small mutual information. This can be seen if we consider that the determinant of a matrix which has rows (or columns) which are linearly dependent is 0.
- It normalizes with respect to any affine transformation of the features, since such a transformation changes by the same amount all the densities in equation (2). This can be seen if we consider that translation has no effect on the covariance matrix and that a linear transformation A gives

$$H(AX) = \frac{1}{2}\log(|A\Sigma A^T|) + \frac{n}{2}(\log 2\pi + 1) = H(X) + \log(|A|).$$

However, this first-order approximation suffers from a core weakness, namely that the entropy of f^* can become arbitrarily larger than the entropy of $\sum_y p_y f_y$ (see figure 1). This leads to degenerated cases where families of features look "infinitely informative". This effect can be mitigated by considering the following upper bound on the true entropy $H(\sum_y p_y f_y)$.

We have by definition

$$I(X;Y) = H\left(\sum_y f_y p_y\right) - \sum_y H(f_y)p_y.$$

Since f^* is a Gaussian density, it has the highest entropy for a given variance, and thus $H(f^*) \geq H(\sum_y f_y p_y)$, hence

$$I(X;Y) \leq H(f^*) - \sum_y H(f_y)p_y. \tag{3}$$

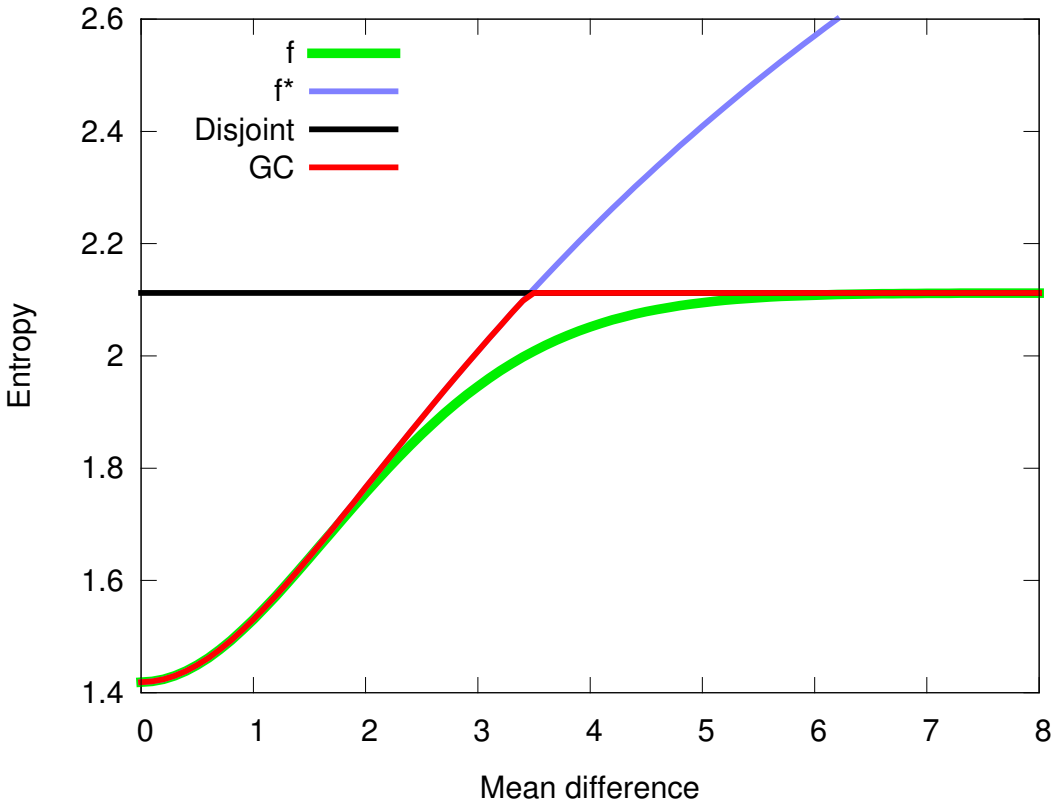

Figure 1: This graph shows several estimates of the entropy of a mixture of two 1D Gaussian densities of variance 1, as a function of the difference between their means. The green curve is the true value of the entropy, estimated numerically. The blue curve is the entropy of a single Gaussian fitted on the mixture. The black line stands for the limit entropy when the two components are 'far apart'. Finally, the red curve is the upper bound described in § 3.2.1.

The mutual information between X and Y is upper bounded by the entropy of Y as Y is discrete, hence

$$I(X;Y) \leq H(Y) = -\sum_y p_y \log p_y,$$

from which we get

$$I(X;Y) \leq \sum_y (H(f_y) - \log p_y) p_y - \sum_y H(f_y) p_y. \tag{4}$$

Taking the min of inequalities (3) and (4), we obtain the following upper bound

$$I(X;Y) \leq \min \left(H(f^*), \sum_y (H(f_y) - \log p_y) p_y \right) - \sum_y H(f_y) p_y. \tag{5}$$

Figure 1 illustrates the behavior of this bound in the case of two 1D Gaussians. An upper bound on $H(X)$ follows directly.

From the concavity of the entropy function we obtain the following lower bound

$$H(X) \geq \sum_y p_y H(f_y). \tag{6}$$

Combining eq. (5),(6) gives

$$\min \left(H(f^*), \sum_y \left(H(f_y) - log(p_y) \right) p_y \right) \geq H(f) \geq \sum_y p_y H(f_y).$$

Note that the difference between upper and lower bound is itself bounded

$$-\sum_y p_y log(p_y) \geq \min \left(H(f^*), \sum_y (H(f_y) - log(p_y)) p_y \right) - \sum_y p_y H(f_y).$$

The proposed GC criterion is based on an approximation to $H(X)$, namely

$$\tilde{H}(f) = \sum_y p_y \min \left(H(f^*), H(f_y) - log(p_y) \right).$$

We note that this approximation is also upper bounded by

$$\min \left(H(f^*), \sum_y (H(f_y) - log(p_y)) p_y \right) \geq \tilde{H}(f).$$

Furthermore, since $\forall y$

$$H(f_y) - log(p_y) > H(f_y),$$

it follows that if $\forall y$

$$H(f^*) \geq H(f_y) \tag{7}$$

then

$$\tilde{H}(f) \geq \sum_y p_y H(f_y),$$

meaning the approximation $\tilde{H}(f)$ also lies between the two bounds and by extension

$$-\sum_y p_y log(p_y) \geq \left|\tilde{H}(f) - H(f)\right|.$$

For (7) to hold it suffices that

$$\lambda_i^* \geq \lambda_i^y, \forall i, y \tag{8}$$

where λ_i^*, λ_i^y are the i_{th} (sorted by magnitude) eigenvalues of Σ^* and Σ^y respectively, in which case

$$\prod_i \lambda_i^* \geq \prod_i \lambda_i^y.$$

That is to say a sufficient condition is that the variance of f^* when projected along any of the eigenvectors of the covariance matrix Σ^* is at least as large as the variance of f^y when projected along the corresponding eigenvector of Σ^y, $\forall y$. Alternatively, for (7) to hold it suffices that $\forall x, y$ (and in particular $\forall x$ which are eigenvectors of Σ^y)

$$x^T \Sigma^* x \geq x^T \Sigma^y x.$$

Given that

$$\Sigma^* = \sum_y p_y \Sigma^y + \sum_y p_y (\mu_y - \mu^*)(\mu_y - \mu^*)^T \tag{9}$$

this translates to

$$\sum_y p_y x^T \Sigma^y x + \sum_y p_y x^T (\mu_y - \mu^*)(\mu_y - \mu^*)^T x^T \geq x^T \Sigma^y x.$$

That is to say that it suffices that the variance of f_y along any direction can be accounted for either by the variances of the mixture components along this direction or by the variance of the mixture means in this direction. Note that in this case (8) also holds.

Based on the above, we use the following approximation to the mutual information to perform feature selection

$$\tilde{I}(X;Y) = \sum_y \min(H(f^*), H(f_y) - \log p_y) p_y - \sum_y H(f_y) p_y. \tag{10}$$

In figure 2 we show a comparison of the GC-approximation and the true mutual information. To compare the two we draw samples from a mixture of five Gaussians and use these samples to estimate the mutual information. Specifically, we create this mixture by sampling the expectations uniformly in $[-5, 5]$, sampling the standard deviations uniformly in $[0.001, 2.001]$ and the priors in $[0, 1]$ (which are then normalized). We observe that by taking the minimum over two sub-optimal estimators (the prior and the fitted Gaussian f^*) we obtain a very good estimator of the true mutual information.

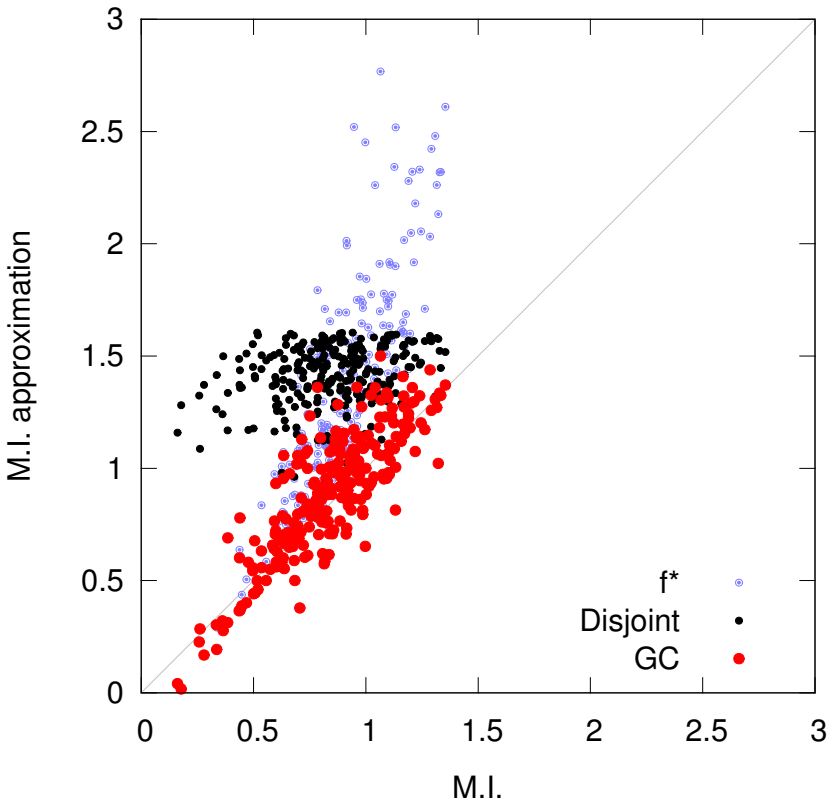

Figure 2: Comparison of the mutual information of a mixture of 5 Gaussians as estimated using the GC-approximation, using the fitted Gaussian f^*, and using the prior distribution (marked as *disjoint*), with the true mutual information calculated numerically.

3.2.2 KL-based Bound

We derive here a more general bound in the case of a distribution f which is a mixture of two distributions $f = p_1 f_1 + p_2 f_2$. In this case we have:

$$H(f) = -\int_{-\infty}^{\infty} p_1 f_1(u) \log\left(p_1 f_1(u) + p_2 f_2(u)\right) + p_2 f_2(u) \log\left(p_1 f_1(u) + p_2 f_2(u)\right) du.$$

Working with the first term in the above integral we have:

$$\begin{aligned}
-\int_{-\infty}^{\infty} p_1 f_1(u) \log\left(p_1 f_1(u) + p_2 f_2(u)\right) du & \\
&= -\int_{-\infty}^{\infty} p_1 f_1(u) \log\left(1 + \frac{p_2 f_2(u)}{p_1 f_1(u)}\right) du \\
&\quad -\int_{-\infty}^{\infty} p_1 f_1(u) \log\left(p_1 f_1(u)\right) du \\
&= c - \int_{-\infty}^{\infty} p_1 f_1(u) \log\left(1 + \frac{p_2 f_2(u)}{p_1 f_1(u)}\right) du + p_1 H(f_1(u)) \\
&\leq c - \int_{-\infty}^{\infty} p_1 f_1(u) \log\left(\frac{p_2 f_2(u)}{p_1 f_1(u)}\right) du + p_1 H(f_1(u)) \\
&= p_1 D_{KL}(f_1(u) \parallel f_2(u)) + p_1 H(f_1(u)) + c',
\end{aligned}$$

where c, c' are constants related to the mixture coefficients and the inequality comes from the fact that $\log(1+x) \geq \log(x)$. Similarly for the second term we have:

$$-\int_{-\infty}^{\infty} p_2 f_2(u) \log\left(p_1 f_1(u) + p_2 f_2(u)\right) du \;\leq\; p_2 D_{KL}(f_2(u) \parallel f_1(u)) + p_2 H(f_2(u)) + c''.$$

Based on this we have:

$$H(f) \;\leq\; p_2 D_{KL}(f_2(u) \parallel f_1(u)) + p_2 H(f_2(u)) + p_1 D_{KL}(f_1(u) \parallel f_2(u)) + p_1 H(f_1(u)) + c''',$$

and by extension we have the following bound on the mutual information:

$$I(X;Y) \leq p_2 D_{KL}(f_2(u) \parallel f_1(u)) + p_1 D_{KL}(f_1(u) \parallel f_2(u)) + c'''.$$

In the case where $f_1 = N(\mu_1, \Sigma_1)$ and $f_2 = N(\mu_2, \Sigma_2)$ are both multivariate Gaussian distributions of dimensionality D, we have that:

$$D_{KL}(f_1 \parallel f_2) = \frac{1}{2}\left(\mathrm{tr}\left({\Sigma_2}^{-1}\Sigma_1\right) - \ln\frac{|\Sigma_1|}{|\Sigma_2|} - D\right) + \frac{1}{2}(\mu_2 - \mu_1)^T {\Sigma_2}^{-1}(\mu_2 - \mu_1).$$

In the case of a binary classification problem, we can directly work with the above quantity for our mixture of two Gaussians. In the case where $|Y| > 2$, we consider the resulting $|Y|$ one-against-all binary classification problems and attempt to maximize the average of the upper bounds of the $|Y|$ mutual information values.

For each class y we consider the following mixture model:

$$f = p_y f_y + (1 - p_y) f_{Y \setminus y}$$

where the $f_{Y \setminus y}$ is the conditional distribution of $X|Y \neq y$. We then calculate the upper bound of the mutual information for all the possible mixtures f, one for each y.

Table 2: Greedy Forward Subset Selection

$S_0 \leftarrow \varnothing$
for $n = 1 \dots N$ **do**
  $s^* = 0$
  for $X_j \in F \setminus S_{n-1}$ **do**
    $S' \leftarrow S_{n-1} \cup X_j$
    $s \leftarrow \tilde{I}(S'; Y)$
    if $s > s^*$ **then**
      $s^* \leftarrow s$
      $S^* \leftarrow S'$
    end if
  end for
  $S_i \leftarrow S^*$
end for
return S_N

3.3 Greedy Forward Selection

The derived bounds and approximations provide measures for assessing the optimal set of features S_N^* of size N. However as there are $\frac{F!}{N!(F-N)!}$ possible sets S_N of size N finding the optimal one by checking all the candidate sets is computationally intractable. Due to this intractability we employ a greedy optimization process in order to find a good approximation $\tilde{S}_N$.

In particular, we use greedy forward selection (see table 2) to iteratively build a sequence of sets S_n with $n = 1, \dots, N$ where each set S_n is built by adding one feature $X_{j(n)}$ to the previous set S_{n-1}, *i.e.* $S_n = S_{n-1} \cup X_{j(n)}$. At a given iteration n, the greedy forward selection algorithm calculates for every candidate feature $X_j \in F \setminus S_{n-1}$, the mutual information between the set $S'_n = S_{n-1} \cup \{X_j\}$ and the label Y. It then creates the set S_n by adding that feature which leads to the largest value of the optimization criterion.

3.4 Complexity of the Gaussian Compromise Method

Though forward selection leads to a computationally tractable feature selection algorithm, it remains nonetheless very expensive. In the case of the Gaussian compromise approach, at each iteration n and for each feature X_j not in S_{n-1}, forward selection requires the estimation of an approximation of $I(S_{n-1} \cup \{X_j\}; Y)$, which in turn requires the estimation of $|Y| + 1$ determinants of the size $n \times n$ covariance matrices. A naive approach would be to calculate these determinants from scratch, incurring a cubic cost of $O(n^3)$ for the calculation of each determinant and a $O(|Y||F \setminus S_{n-1}|n^3)$ cost per iteration with a $O(|Y|n^2)$ memory requirement for storing the covariance matrices and their inverses.

As shown in previous work (Lefakis and Fleuret, 2014) however it is possible to derive both an $O(|Y||F \setminus S_{n-1}|n^2)$ algorithm with $O(|Y|n^2)$ memory requirements and an $O(|Y||F \setminus S_{n-1}|n)$ algorithm with $O(|Y|n^2 + |Y||F \setminus S_{n-1}|n)$ memory requirements. In the following we expand upon those methods, in particular the $O(|Y||F \setminus S_{n-1}|n)$ one, and

present an approach that allows for a $O(|Y||F \setminus S_{n-1}|n)$ with $O(|Y|n^2 + |Y||F \setminus S_{n-1}|)$ memory requirements.

In order to speed up computations, we first note that for three random variables X, Y, Z

$$I(X, Z; Y) = I(Z; Y) + I(X; Y \mid Z)$$

which in the context of our forward selection algorithm, $\forall X_j \in F \setminus S_{n-1}$, and with $S'_n = S_{n-1} \cup \{X_j\}$, translates to

$$I(S'_n; Y) = I(S_{n-1}; Y) + I(X_j; Y \mid S_{n-1}).$$

The first term in the above expression is common for all candidate features X_j, meaning that finding the feature X_j that maximizes $I(S'_n; Y)$ is equivalent to finding the feature that maximizes $I(X_j; Y \mid S_{n-1})$. If $\sigma^2_{j|S_{n-1}}$ denotes the variance of feature j conditioned on the features in S_{n-1} and $\sigma^2_{j|y,S_{n-1}}$ the variance conditioned on the features in S_{n-1} and the class $Y = y$, we have

$$\begin{aligned} \underset{X_j \in F \setminus S_{n-1}}{\operatorname{argmax}} \; I(X_j; Y \mid S_{n-1}) &= \underset{X_j \in F \setminus S_{n-1}}{\operatorname{argmax}} \left(H(X_j \mid S_{n-1}) - H(X_j \mid Y, S_{n-1}) \right) && (11) \\ &= \underset{X_j \in F \setminus S_{n-1}}{\operatorname{argmax}} \left(\log \sigma^2_{j|S_{n-1}} - \sum_y P(Y = y) \log \sigma^2_{j|y,S_{n-1}} \right) && (12) \end{aligned}$$

where here and in the following, we slightly abuse notation and use S_n to denote both the set and its contents, which of the two is meant will in any case be clear from context. To derive equation 12 we have exploited the fact that the conditional variance $\sigma_{j|S_{n-1}}$ is independent of the specific values of the features in S_{n-1} and thus the integrations of the entropies over the conditioned values are straightforward. That is

$$\begin{aligned} H(X_j \mid S_{n-1}) &= \int_{\mathbb{R}^{|S_{n-1}|}} H(X_j \mid S_{n-1} = s) \mu_{S_{n-1}}(s) ds \\ &= \frac{1}{2} \log \sigma^2_{j|S_{n-1}} + \frac{1}{2} \left(\log 2\pi + 1 \right). \end{aligned}$$

Under the Gaussian assumption, we have

$$\sigma^2_{j|S_{n-1}} = \sigma^2_j - \Sigma^T_{j,S_{n-1}} \Sigma^{-1}_{S_{n-1}} \Sigma_{j,S_{n-1}}.$$

From the above we can derive an $O(|Y||F \setminus S_{n-1}|n^2)$ with $O(|Y|n^2)$ memory requirements by noting that computing $\Sigma^T_{j,S_{n-1}} \Sigma^{-1}_{S_{n-1}} \Sigma_{j,S_{n-1}}$ incurs a cost of $O(n^2)$ and that this must be done for every candidate feature j and every class y.

3.4.1 Efficient Computation of $\sigma^2_{j|S_{n-1}}$

As stated, we can further speed-up the proposed algorithm by a factor of n by considering more carefully the calculation of $\sigma^2_{j|S_{n-1}}$. If $S_{n-1} = S_{n-2} \cup X_i$, we have

$$\Sigma^T_{j,S_{n-1}} \Sigma^{-1}_{S_{n-1}} \Sigma_{j,S_{n-1}} = \left[\Sigma^T_{j,S_{n-2}} \sigma^2_{ji} \right] \Sigma^{-1}_{S_{n-1}} \begin{bmatrix} \Sigma_{j,S_{n-2}} \cdot \\ \sigma^2_{ij} \end{bmatrix} \quad (13)$$

We note that $\Sigma_{S_{n-1}}$ differs from $\Sigma_{S_{n-2}}$ by the addition of a row and a column

$$\Sigma_{S_{n-1}} = \begin{bmatrix} \Sigma_{S_{n-2}} & \Sigma_{i,S_{n-2}} \\ \Sigma^T_{i,S_{n-2}} & \sigma^2_i. \end{bmatrix}$$

Thus $\Sigma_{S_{n-1}}$ is the result of a rank-two update to the augmented matrix

$$\begin{bmatrix} \Sigma_{S_{n-2}} & 0_{n-2} \\ 0^T_{n-2} & \sigma^2_i \end{bmatrix},$$

specifically a one rank-one update corresponding to changing the final row and a rank-one update corresponding to changing the final column. By applying the Sherman-Morrison formula twice to update $\Sigma^{-1}_{S_{n-2}}$ to $\Sigma^{-1}_{S_{n-1}}$, we can obtain [1] an update formula of the form

$$\Sigma^{-1}_{S_{n-1}} = \begin{bmatrix} \Sigma^{-1}_{S_{n-2}} & -\frac{1}{\beta\sigma^2_i}u \\ -\frac{1}{\beta\sigma^2_i}u^T & \frac{1}{\beta\sigma^2_i} \end{bmatrix} + \frac{1}{\beta\sigma^2_i}\begin{bmatrix} u \\ 0 \end{bmatrix}\begin{bmatrix} u^T & 0 \end{bmatrix} \tag{14}$$

where

$$u = \Sigma^{-1}_{S_{n-2}}\Sigma_{i,S_{n-2}}$$

and

$$\beta = 1 - \frac{1}{\sigma^2_i}\Sigma^T_{i,S_{n-2}}\Sigma^{-1}_{S_{n-2}}\Sigma_{i,S_{n-2}}.$$

From equation (13) and (14) we have

$$\begin{aligned}
\sigma^2_{j|S_{n-1}} &= \sigma^2_j - \Sigma^T_{j,S_{n-1}}\left(\begin{bmatrix} \Sigma^{-1}_{S_{n-2}} & -\frac{1}{\beta\sigma^2_i}u \\ -\frac{1}{\beta\sigma^2_i}u^T & \frac{1}{\beta\sigma^2_i} \end{bmatrix} + \frac{1}{\beta\sigma^2_i}\begin{bmatrix} u \\ 0 \end{bmatrix}\begin{bmatrix} u^T & 0 \end{bmatrix}\right)\Sigma_{j,S_{n-1}} \\
&= \sigma^2_j - \Sigma^T_{j,S_{n-2}}\Sigma^{-1}_{S_{n-2}}\Sigma_{j,S_{n-2}} + \frac{\sigma^2_{ji}}{\beta\sigma^2_i}u^T\Sigma_{j,S_{n-2}} \\
&\quad -\Sigma^T_{j,S_{n-1}}\begin{bmatrix} -\frac{1}{\beta\sigma^2_i}u \\ \frac{1}{\beta\sigma^2_i} \end{bmatrix}\sigma^2_{ji} \\
&\quad -\frac{1}{\beta\sigma^2_i}\left(\Sigma^T_{j,S_{n-1}}\begin{bmatrix} u \\ 0 \end{bmatrix}\right)\left(\begin{bmatrix} u^T & 0 \end{bmatrix}\Sigma_{jS_{n-1}}\right)
\end{aligned} \tag{15}$$

In eq (15) the main computational cost is incurred by the calculation of $\Sigma^T_{j,S_{n-2}}\Sigma^{-1}_{S_{n-2}}\Sigma_{j,S_{n-2}}$ which costs $O(n^2)$. However this quantity has been already calculated $\forall j$ during the previous iteration of the algorithm since this involves calculating

$$\sigma^2_{j|S_{n-2}} = \sigma^2_j - \Sigma^T_{j,S_{n-2}}\Sigma^{-1}_{S_{n-2}}\Sigma_{j,S_{n-2}}.$$

Thus we only need carry this result over from the previous iteration incurring an additional memory load of $O(|Y||F \setminus S_{n-1}|)$.

The remaining terms in equation (15) can be calculated in $O(n)$ given β and u. As β and u depend only on the feature i selected in the previous iteration, remaining constant throughout iteration n, they can be pre-computed once at the beginning of each iteration. Thus the cost of calculating $\sigma^2_{j|S_{n-2}}$ can be reduced to $O(n)$ and the overall computational cost per iteration to $O(|Y||F \setminus S_{n-1}|n)$.

1. The proof follows from simple verification.

3.5 Complexity of the KL-based Algorithms

In the case of the KL-based algorithms, similarly with the Gaussian compromise approach, a naive implementation would incur a cost of $O(|Y||F \setminus S_{n-1}|n^3)$. In previous work(Lefakis and Fleuret, 2014) an algorithm was sketched which had a $O(|Y|n^2)$ memory footprint. Here we expand on this analysis and furthermore present an alternative algorithm with a $O(|Y||F \setminus S_{n-1}|n)$ complexity, which however has an increased memory footprint (specifically $O(|Y||F \setminus S_{n-1}|n^2)$).

Working with the value:

$$\begin{aligned} &P(Y=y)D_{KL}\left(p(S|Y=y) \parallel p(S|Y \neq y)\right) + P(Y=y)H(S \mid Y=y) \\ +&P(Y \neq y)D_{KL}\left(p(S|Y \neq y) \parallel p(S|Y=y)\right) + P(Y \neq y)H(S \mid Y \neq y) + c''' \end{aligned}$$

we note that the entropy values $H(S \mid Y=y)$ can be computed efficiently as in the previous subsection. What remains is to efficiently compute the Kullback-Leibler divergences for each of the $|Y|$ binary classification problems.

As both distributions are assumed to be Gaussians, $D_{KL}\left(p(S|Y \neq y) \parallel p(S|Y=y)\right)$ is equal to

$$\frac{1}{2}\left(\operatorname{tr}\left({\Sigma^{y}_{S'_n}}^{-1}\Sigma^{Y\setminus y}_{S'_n}\right) - \log\frac{|\Sigma^{Y\setminus y}_{S'_n}|}{|\Sigma^{y}_{S'_n}|} + \left(\mu^{Y\setminus y}_{S'_n} - \mu^{y}_{S'_n}\right)^T {\Sigma^{y}_{S'_n}}^{-1}\left(\mu^{Y\setminus y}_{S'_n} - \mu^{y}_{S'_n}\right) - |S|\right).$$

In the following we show how each of these terms can be computed in time $O(n)$.

3.5.1 The Term $\log\frac{|\Sigma^{Y\setminus y}_{S'_n}|}{|\Sigma^{y}_{S'_n}|}$

From the chain rule,

$$H(S_{n-1} \cup X_j) = H(S_{n-1}) + H(X_j \mid S_{n-1})$$

we have

$$\begin{aligned} \log|\Sigma_{S'_n}| + \frac{n}{2}(1+\log 2\pi) &= \log|\Sigma_{S_{n-1}}| + \frac{n-1}{2}(1+\log 2\pi) + \log\sigma^2_{j|S_{n-1}} + \frac{1}{2}(1+\log 2\pi) \\ \log|\Sigma_{S'_n}| &= \log|\Sigma_{S_{n-1}}| + \log\sigma^2_{j|S_{n-1}} \\ |\Sigma_{S'_n}| &= \sigma^2_{j|S_{n-1}}|\Sigma_{S_{n-1}}|. \end{aligned}$$

As shown in the previous section, the term $\sigma^2_{j|S_{n-1}}$ can be computed in $O(n)$ time. The term $|\Sigma_{S_{n-1}}|$ is independent of j and can be efficiently pre-computed from $|\Sigma_{S_{n-2}}|$ prior to iteration n using the matrix determinant lemma. By extension, the cost of calculating $\log\frac{|\Sigma^{Y\setminus y}_{S'_n}|}{|\Sigma^{y}_{S'_n}|}$ is itself $O(n)$.

3.5.2 Calculating ${\Sigma^y_{S'_n}}^{-1}$

Setting, here and in the rest of this section, $\Sigma_S = \Sigma^y_S$ for ease of exposition[2] , we have similar to section 3.4 that

$$\Sigma^{-1}_{S'_n} = \begin{bmatrix} \Sigma^{-1}_{S_{n-1}} & -\frac{1}{\beta\sigma_j^2}u \\ -\frac{1}{\beta\sigma_j^2}u^T & \frac{1}{\beta\sigma_j^2} \end{bmatrix} + \frac{1}{\beta\sigma_j^2}\begin{bmatrix} u \\ 0 \end{bmatrix}\begin{bmatrix} u^T & 0 \end{bmatrix} \tag{16}$$

where

$$u = \Sigma^{-1}_{S_{n-1}}\Sigma_{j,S_{n-1}}$$

and

$$\beta = 1 - \frac{1}{\sigma_j^2}\Sigma^T_{j,S_{n-1}}\Sigma^{-1}_{S_{n-1}}\Sigma_{j,S_{n-1}}.$$

Here u and β cannot be pre-computed as they are different $\forall j$. They can either be calculated from scratch in $O(n^2)$ or, if we are willing to incur a memory overhead of $O(n)$, with $O(n)$ complexity from the product $\Sigma^{-1}_{S_{n-2}}\Sigma_{j,S_{n-2}}$ which has been computed during the previous iteration (as in section 3.4.1).

3.5.3 The Term $\left(\mu^{Y\setminus y}_{S'_n} - \mu^y_{S'_n}\right)^T {\Sigma^y_{S'_n}}^{-1}\left(\mu^{Y\setminus y}_{S'_n} - \mu^y_{S'_n}\right)$

Having computed u and β as defined above, we can efficiently calculate the product

$$M = \left(\mu^{Y\setminus y}_{S'_n} - \mu^y_{S'_n}\right)^T \Sigma^{-1}_{S'_n}\left(\mu^{Y\setminus y}_{S'_n} - \mu^y_{S'_n}\right)$$

given that $\mu^y_{S'_n} = \begin{bmatrix} \mu^y_{S_{n-1}} \\ \mu^y_j \end{bmatrix}$ by decomposing it as follows

$$\begin{aligned} M = &\left(\mu^{Y\setminus y}_{S_{n-1}} - \mu^y_{S_{n-1}}\right)^T \Sigma^{-1}_{S_{n-1}}\left(\mu^{Y\setminus y}_{S_{n-1}} - \mu^y_{S_{n-1}}\right) \\ &-\frac{\left(\mu^{Y\setminus y}_j - \mu^y_j\right)}{\beta\sigma_j^2}u^T\left(\mu^{Y\setminus y}_{S_{n-1}} - \mu^y_{S_{n-1}}\right) \\ &+\left(\mu^{Y\setminus y}_{S_{n-1}} - \mu^y_{S_{n-1}}\right)^T\begin{bmatrix} -\frac{1}{\beta\sigma_j^2}u \\ \frac{1}{\beta\sigma_j^2} \end{bmatrix}\left(\mu^{Y\setminus y}_j - \mu^y_j\right) \\ &+\frac{1}{\beta\sigma_j^2}\left(\left(\mu^{Y\setminus y}_{S_{n-1}} - \mu^y_{S_{n-1}}\right)^T\begin{bmatrix} u \\ 0 \end{bmatrix}\right)\left(\begin{bmatrix} u^T & 0 \end{bmatrix}\left(\mu^{Y\setminus y}_{S_{n-1}} - \mu^y_{S_{n-1}}\right)\right) \end{aligned}$$

Of these four terms, the final three involving the vectors u and β can be calculated in $O(n)$ as they only involve inner products. The first term requires $O(n^2)$, however as the term is independent of j it can be calculated once at the beginning of each iteration. Consequently the complexity of calculating $\left(\mu^{Y\setminus y}_{S'_n} - \mu^y_{S'_n}\right)^T {\Sigma_{S'_n}}^{-1}\left(\mu^{Y\setminus y}_{S'_n} - \mu^y_{S'_n}\right)$ given u and β is $O(n)$.

2. That is when a superscript is missing, y is implied.

3.5.4 The Term tr $\left(\Sigma^{y}_{S'_n}{}^{-1}\Sigma^{Y\setminus y}_{S'_n}\right)$

The term tr $\left(\Sigma^{y}_{S'_n}{}^{-1}\Sigma^{Y\setminus y}_{S'_n}\right)$ involves calculating, and summing, the main diagonal elements of the matrix product $\left(\Sigma^{y}_{S'_n}{}^{-1}\Sigma^{Y\setminus y}_{S'_n}\right)$. We have that

$$\Sigma^{Y\setminus y}_{S'_n} = \begin{bmatrix} \Sigma^{Y\setminus y}_{S_{n-1}} & \Sigma^{Y\setminus y}_{j,S_{n-1}} \\ \Sigma^{Y\setminus y}_{j,S_{n-1}}{}^{T} & \sigma_j^{Y\setminus y 2} \end{bmatrix}. \tag{17}$$

From equations (16) and (17) we see that the product can be decomposed into two parts. For the first

$$\begin{bmatrix} \Sigma^{-1}_{S_{n-1}} & -\frac{1}{\beta\sigma_j^2}u \\ -\frac{1}{\beta\sigma_j^2}u^T & \frac{1}{\beta\sigma_j^2} \end{bmatrix} \begin{bmatrix} \Sigma^{Y\setminus y}_{S_{n-1}} & \Sigma^{Y\setminus y}_{j,S_{n-1}} \\ \Sigma^{Y\setminus y}_{j,S_{n-1}}{}^{T} & \sigma_j^{Y\setminus y 2} \end{bmatrix}$$

it is straightforward to show that the main diagonal elements can be calculated in $O(n)$ provided we have pre-computed the main diagonal elements of $\Sigma^{-1}_{S_{n-1}}\Sigma^{Y\setminus y}_{S_{n-1}}$. As this product does not depend on j this can be done prior to the beginning of the iteration. For the second we have

$$\frac{1}{\beta\sigma_j^2}\begin{bmatrix} \Sigma^{-1}_{S_{n-1}}\Sigma_{j,S_{n-1}} \\ 0 \end{bmatrix}\begin{bmatrix} \Sigma^{T}_{j,S_{n-1}}\Sigma^{-1}_{S_{n-1}} & 0 \end{bmatrix}\begin{bmatrix} \Sigma^{Y\setminus y}_{S_{n-1}} & \Sigma^{Y\setminus y}_{j,S_{n-1}} \\ \Sigma^{Y\setminus y}_{j,S_{n-1}}{}^{T} & \sigma_j^{Y\setminus y 2} \end{bmatrix}$$

which, given that $\mathrm{tr}(w^T wA) = \mathrm{tr}(wAw^T)$, is equal to the product of $\frac{1}{\beta\sigma_j^2}$ and the trace of

$$\Sigma^{T}_{j,S_{n-1}}\Sigma^{-1}_{S_{n-1}}\Sigma^{Y\setminus y}_{S_{n-1}}\Sigma^{-1}_{S_{n-1}}\Sigma_{j,S_{n-1}}.$$

As we have already calculated the vector $\Sigma^{-1}_{S_{n-1}}\Sigma_{j,S_{n-1}}$ when calculating $\Sigma^{-1}_{S'_n}$ we concentrate here on calculating the vector $\Sigma^{T}_{j,S_{n-1}}\Sigma^{-1}_{S_{n-1}}\Sigma^{Y\setminus y}_{S_{n-1}}$.

As shown, the matrix $\Sigma^{-1}_{S_{n-1}}$ can be written in the form

$$\Sigma^{-1}_{S_{n-1}} = \begin{bmatrix} \Sigma^{-1}_{S_{n-2}} & \gamma v \\ \gamma v^T & -\gamma \end{bmatrix} - \gamma\begin{bmatrix} v \\ 0 \end{bmatrix}\begin{bmatrix} v^T & 0 \end{bmatrix}$$

where

$$v = \Sigma^{-1}_{S_{n-2}}\Sigma_{i,S_{n-2}}$$

and

$$\gamma = -\frac{1}{\beta\sigma_j^2}.$$

Thus the vector $\Sigma^{T}_{j,S_{n-1}}\Sigma^{-1}_{S_{n-1}}\Sigma^{Y\setminus y}_{S_{n-1}}$ can be decomposed into

$$\Sigma^{T}_{j,S_{n-1}}\begin{bmatrix} \Sigma^{-1}_{S_{n-2}} & \gamma v \\ \gamma v^T & -\gamma \end{bmatrix}\Sigma^{Y\setminus y}_{S_{n-1}} - \gamma\Sigma^{T}_{j,S_{n-1}}\begin{bmatrix} v \\ 0 \end{bmatrix}\begin{bmatrix} v^T & 0 \end{bmatrix}\Sigma^{Y\setminus y}_{S_{n-1}}.$$

As v does not depend on j the product $\left[\begin{array}{cc} v^T & 0 \end{array}\right] \Sigma^{Y\setminus y}_{S_{n-1}}$ can be computed prior to the iteration and by extension the product $\gamma \Sigma^T_{j,S_{n-1}} \left[\begin{array}{c} v \\ 0 \end{array}\right] \left[\begin{array}{cc} v^T & 0 \end{array}\right] \Sigma^{Y\setminus y}_{S_{n-1}}$ can be computed in $O(n)$. This leaves the final term

$$\Sigma^T_{j,S_{n-1}} \left[\begin{array}{cc} \Sigma^{-1}_{S_{n-2}} & \gamma v \\ \gamma v^T & -\gamma \end{array}\right] \Sigma^{Y\setminus y}_{S_{n-1}}.$$

We note that the vector $\Sigma^T_{j,S_{n-1}} \left[\begin{array}{cc} \Sigma^{-1}_{S_{n-2}} & \gamma v \\ \gamma v^T & -\gamma \end{array}\right]$ has the form

$$\left[\begin{array}{c} \Sigma_{j,S_{n-2}} \Sigma^{-1}_{S_{n-2}} + \gamma \sigma^2_{ji} v^T \\ \gamma \Sigma^T_{j,S_{n-2}} v - \gamma \sigma^2_{ji} \end{array}\right]^T.$$

Thus we have

$$\left[\begin{array}{cc} \Sigma_{j,S_{n-2}} \Sigma^{-1}_{S_{n-2}} \Sigma^{Y\setminus y}_{S_{n-2}} & 0 \end{array}\right] + \gamma \sigma^2_{ji} \left[\begin{array}{cc} v^T & 0 \end{array}\right] \Sigma^{Y\setminus y}_{S_{n-1}} + \left(\gamma \Sigma^T_{j,S_{n-2}} v - \gamma \sigma^2_{ji}\right) \Sigma^{Y\setminus y}_{i,S_{n-1}}.$$

During the previous iteration we have already computed the vector $\Sigma_{j,S_{n-2}} \Sigma^{-1}_{S_{n-2}} \Sigma^{Y\setminus y}_{S_{n-2}}$ and thus if we use $O(n)$ memory to store it between iterations, we can also compute this final term in $O(n)$.

4. Using the Eigen-decomposition to Bound Computations

The fast implementation of the GC-approximation method presented in 3.4 requires at iteration n the calculation, for each feature $X_j \in F \setminus S_{n-1}$, of the conditioned variance

$$\sigma^2_{j|S_{n-1}} = \sigma^2_j - \Sigma^T_{jS_{n-1}} \Sigma^{-1}_{S_{n-1}} \Sigma_{jS_{n-1}}.$$

As shown, each such computation can be done in $O(n)$. The main computational cost comes from the calculation of $\Sigma^T_{jS_{n-1}} \Sigma^{-1}_{S_{n-1}} \Sigma_{jS_{n-1}}$. Thus it would be advantageous to acquire a "cheap" (independent of n) bound which will allow us to skip the calculation of this quantity for certain non-promising features.

We note that $\Sigma^{-1}_{S_{n-1}}$ is positive definite and symmetric and thus can be decomposed as

$$\Sigma^{-1}_{S_{n-1}} = U \Lambda U^T,$$

where U is orthonormal and Λ is a diagonal matrix with positive elements. Thus

$$\Sigma^T_{jS_{n-1}} \Sigma^{-1}_{S_{n-1}} \Sigma_{jS_{n-1}} = \Sigma^T_{jS_{n-1}} U \Lambda U^T \Sigma_{jS_{n-1}}.$$

As U is orthonormal we have

$$\|\Sigma_{jS_{n-1}}\|_2 = \|\Sigma^T_{jS_{n-1}} U\|_2 = \|U^T \Sigma_{jS_{n-1}}\|_2.$$

Symbolizing the eigenvalues as $\lambda_1, \lambda_2, \ldots, \lambda_{n-1}$ and the elements of the vector $\Sigma^T_{jS_{n-1}} U$ as $x_1, x_2, \ldots, x_{n-1}$, we have that

$$\|\Sigma^T_{jS_{n-1}}\|^2_2 \min_i \lambda_i \leq \Sigma^T_{jS_{n-1}} \Sigma^{-1}_{S_{n-1}} \Sigma_{jS_{n-1}} \leq \|\Sigma^T_{jS_{n-1}}\|^2_2 \max_i \lambda_i. \tag{18}$$

Equation 18 gives us a bound, computable in $O(|Y|)$, which we can use to avoid unnecessary computations. Specifically, during iteration n of the algorithm, after having already calculated the scores of a subset of features, we have a candidate for best feature which has a score of s^*; for each subsequent candidate feature X_j we can compute the following upper bound on the feature's score

$$ub_1(X_j) = \log\left(\sigma_j^2 - \|\Sigma^T_{jS_{n-1}}\|_2^2 \max_i \lambda_i\right) - \sum_{y=0}^{|Y|-1}\left(\log\left(\sigma_j^{y^2} - \|\Sigma^{y^T}_{jS_{n-1}}\|_2^2 \min_i \lambda_i^y\right)\right), \quad (19)$$

where λ_i, λ_i^y are the eigenvalues of $\Sigma^{-1}_{S_{n-1}}$ and $\Sigma^{y^{-1}}_{S_{n-1}}$ respectively. Then if $ub(s) \leq s^*$, we can avoid the $O(|Y|n)$ computations required to estimate the feature's score s.

Furthermore should $ub_1(X_j) > s^*$ we can still proceed with the computations in a greedy manner. That is instead of calculating the exact score

$$\log\left(\sigma^2_{j|S_{n-1}}\right) - \sum_{y=0}^{|Y|-1}\left(\log\left(\sigma^{y^2}_{j|S_{n-1}}\right)\right)$$

incurring $O(|Y|n)$ cost, we can compute the conditional variances one at a time and then reassess the upper bound. That is we first calculate $\sigma^2_{j|S_{n-1}}$ which costs us $O(n)$ and then re-estimate the upper bound as

$$ub_2(X_j) = \log\left(\sigma^2_{j|S_{n-1}}\right) - \sum_{y=0}^{|Y|-1}\left(\log\left(\sigma_j^{y^2} - \|\Sigma^{y^T}_{jS_{n-1}}\|_2^2 \min_i \lambda_i^y\right)\right)$$

re-checking whether $ub(s) \leq s^*$. We can then continue, if necessary, by calculating

$$ub3(X_j) = \log\left(\sigma^2_{j|S_{n-1}}\right) - \log\left(\sigma^{y_0^2}_{j|S_{n-1}}\right) - \sum_{y=1}^{|Y|-1}\left(\log\left(\sigma_j^{y^2} - \|\Sigma^{y^T}_{jS_{n-1}}\|_2^2 \min_i \lambda_i^y\right)\right)$$

and so forth. Thus we can avoid estimating a number of conditional variances, incurring a smaller cost. This process can continue greedily by computing $ub_{1\ldots|Y|+1}(X_j)$.

We note that the upper bound 19 involves $|Y|$ lower bounds

$$\log\left(\sigma_j^{y^2} - \|\Sigma^{y^T}_{jS_{n-1}}\|_2^2 \min_i \lambda_i^y\right)$$

on the conditional variances $\sigma^{y^2}_{j|S_{n-1}}$. As such the bound can become quite loose if the number of classes $|Y|$ is large. In order to empirically evaluate the usefulness of this bound in pruning computations, we considered 3 binary classification tasks; the binary task of the INRIA data set, and two tasks resulting from the CIFAR and STL data sets by a random partitioning of the classes (*i.e.* that is in each case 5 classes were randomly chosen to be labeled as positive and the rest as negative).

In figure 3 we can see for each of the three data sets, the number of features at each round for which we can skip a certain amount of computations. In the case $ub_1 \leq s^*$ we

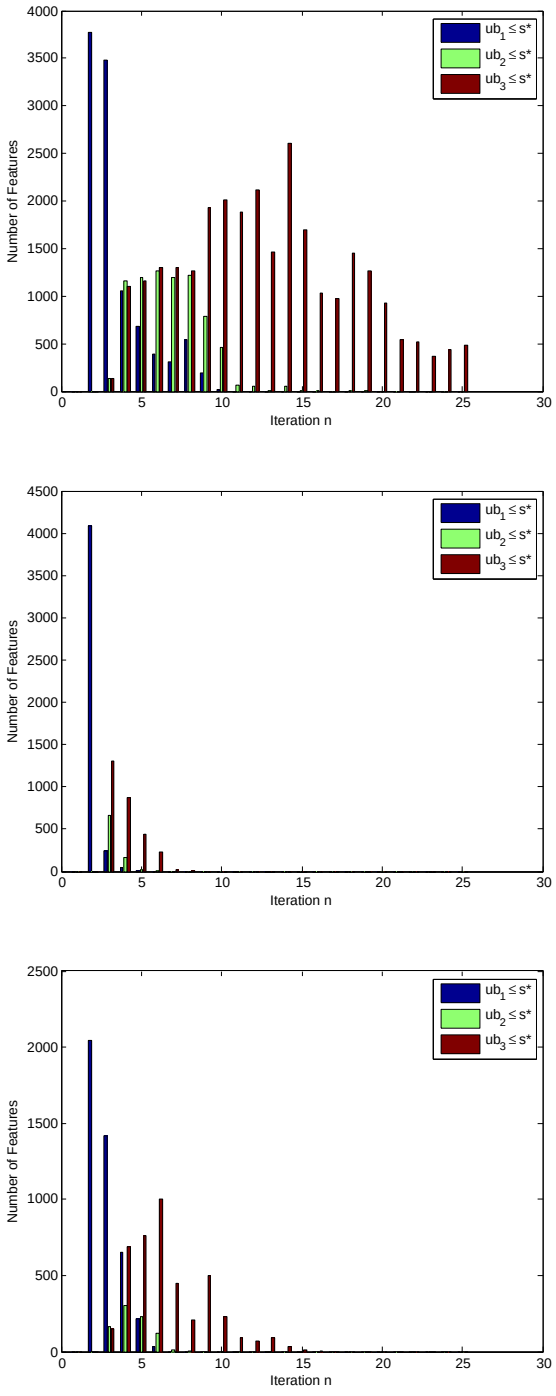

Figure 3: Number of features for which the upper bound ub_x allows us to skip computations for the top) INRIA, middle) STL, and bottom) CIFAR data sets.

can avoid all computations concerning the conditional variances. In the case $ub_2 \leq s^*$ only one conditional variance need be calculated, while in the case of $ub_3 \leq s^*$ two (out of a possible three). As can be seen, the bound can prove quite useful in pruning computations, as is the case in the INRIA data set. For the CIFAR data set, we see that the bound can still prove useful, especially early on. On the contrary in the case of the STL data set, the bound seems to provide little help in ways of avoiding computations.

Pruning computations using the above bounds requires access to the eigenvalues of the matrices $\Sigma^{y^{-1}}_{S_{n-1}}$ which are the reciprocals of the eigenvalues of the matrices $\Sigma^{y}_{S_{n-1}}$. As computing the eigen-decomposition of a matrix, from scratch, can be expensive we present in the following a novel algorithm for efficiently calculating these eigenvalues, and the corresponding eigenvectors, of $\Sigma^{y^{-1}}_{S_{n-1}}$ from the eigen-decomposition of $\Sigma^{y^{-1}}_{S_{n-2}}$.

4.1 Eigen-system Update

The matrices $\Sigma^{y^{-1}}_{S_{n-1}}$ results from the matrices $\Sigma^{y^{-1}}_{S_{n-2}}$ by the addition of a row and a column. We are faced thus with the problem of updating the eigen-system of a symmetric and positive definite matrix Σ when a vector is inserted as an extra row and column.

More specifically, we shall present a method for *efficiently* computing the eigen-system U^{n+1}, Λ^{n+1} of a $(n+1) \times (n+1)$ matrix Σ^{n+1} when the eigen-system U^n, Λ^n of the $n \times n$ matrix Σ^n is known and Σ^{n+1} is related to Σ^n as follows:

$$\Sigma^{n+1} = \begin{bmatrix} \Sigma^n & \boldsymbol{v} \\ \boldsymbol{v}^T & c \end{bmatrix},$$

where $\boldsymbol{v}$ and c are such that the matrix Σ^{n+1} is positive definite. Without loss of generality, we will consider in the following the special case $c = 1$.

If $\boldsymbol{u}^n_1, \ldots, \boldsymbol{u}^n_n$ and $\lambda^n_1, \ldots, \lambda^n_n$ are the eigenvectors and respective eigenvalues of Σ^n, then $\begin{bmatrix} \boldsymbol{u}^n_i \\ 0 \end{bmatrix}$ is obviously an eigenvector of

$$\Sigma' = \begin{bmatrix} \Sigma^n & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix},$$

with associated eigenvalue $\lambda'_i = \lambda^n_i$. Also, if $\forall i \in \{1, \ldots, n+1\}$, $\boldsymbol{e}_i$ is the i_{th} standard basis vector of $\mathbb{R}^{n+1}$, then $\Sigma' \boldsymbol{e}_{n+1} = \boldsymbol{e}_{n+1}$ and $\boldsymbol{e}_{n+1}$ is an eigenvector of Σ' with a corresponding eigenvalue of $\lambda'_{n+1} = 1$.

The matrix Σ^{n+1} can be expressed as the result of a rank-two update to Σ'

$$\Sigma^{n+1} = \Sigma' + \boldsymbol{e}_{n+1} \begin{bmatrix} \boldsymbol{v} \\ 0 \end{bmatrix}^T + \begin{bmatrix} \boldsymbol{v} \\ 0 \end{bmatrix} \boldsymbol{e}^T_{n+1}.$$

If U', Λ' denote the eigen-system of Σ', by multiplying

$$\Sigma^{n+1} \boldsymbol{u}^{n+1} = \lambda^{n+1} \boldsymbol{u}^{n+1},$$

on the left by U'^T, we have

$$U'^T \left(\Sigma' + \boldsymbol{e}_{n+1} \begin{bmatrix} \boldsymbol{v} \\ 0 \end{bmatrix}^T + \begin{bmatrix} \boldsymbol{v} \\ 0 \end{bmatrix} \boldsymbol{e}^T_{n+1} \right) \boldsymbol{u}^{n+1} = \lambda^{n+1} U'^T \boldsymbol{u}^{n+1}$$

Given that Σ' is positive definite and symmetric it follows that $U'U'^T = I$ and

$$U'^T \left(\Sigma' + \boldsymbol{e}_{n+1} \begin{bmatrix} \boldsymbol{v} \\ 0 \end{bmatrix}^T + \begin{bmatrix} \boldsymbol{v} \\ 0 \end{bmatrix} \boldsymbol{e}_{n+1}^T \right) U'U'^T \boldsymbol{u}^{n+1} = \lambda^{n+1} U'^T \boldsymbol{u}^{n+1}$$

and since $U'^T \Sigma' U' = \Lambda'$ we have

$$\left(\Lambda' + \boldsymbol{e}_{n+1}\boldsymbol{q}^T + \boldsymbol{q}\boldsymbol{e}_{n+1}^T \right) U'^T \boldsymbol{u}^{n+1} = \lambda^{n+1} U'^T \boldsymbol{u}^{n+1}$$

where $q = U'^T \begin{bmatrix} \boldsymbol{v} \\ 0 \end{bmatrix}$, note $\boldsymbol{e}_{n+1}^T U' = \boldsymbol{e}_{n+1}^T$.

Thus Σ^{n+1} and the matrix $\Sigma'' = \left(\Lambda' + \boldsymbol{e}_{n+1}\boldsymbol{q}^T + \boldsymbol{q}\boldsymbol{e}_{n+1}^T \right)$ share eigenvalues. Furthermore the eigenvectors U^{n+1} are related to the eigenvectors U'' of Σ'' by $U^{n+1} = U'U''$.

4.1.1 Computing the Eigenvalues and Eigenvectors of Σ''

The matrix Σ'' has non-zero elements only on its main diagonal and on its last row and column. By developing the determinant $|\Sigma'' - \lambda I|$ along the final row we have

$$|\Sigma'' - \lambda I| = \prod_j (\lambda_j^{'} - \lambda) + \sum_{i<n+1} \left(-q_i^2 \prod_{j \neq i, j<n+1} (\lambda_j^{'} - \lambda) \right),$$

where q_i is the i_{th} element of vector q. The determinant thus has the same roots as the function

$$f(\lambda) = \frac{|\Sigma'' - \lambda I|}{\prod_{j<n+1}(\lambda_j^{'} - \lambda)} \tag{20}$$

$$= \lambda'_{n+1} - \lambda + \sum_i \frac{-q_i^2}{(\lambda'_i - \lambda)}. \tag{21}$$

We have $\forall i$, $\lim_{\lambda \xrightarrow{>} \lambda'_i} f(\lambda) = +\infty$ and $\lim_{\lambda \xrightarrow{<} \lambda'_i} f(\lambda) = -\infty$. Furthermore we have

$$\frac{\partial f(\lambda)}{\partial \lambda} = -1 + \sum_i \frac{-q_i^2}{(\lambda'_i - \lambda)^2} \leq 0$$

meaning the function f is strictly decreasing between its poles. From this and from the positive definiteness of Σ'' we have that

$$0 < \lambda_1^{n+1} < \lambda'_1 < \lambda_1^{n+1} < \cdots < \lambda'_{n+1} < \lambda_{n+1}^{n+1}$$

i.e. the eigenvalues of Σ' and Σ'' are interlaced.

Though there is no analytical solution for finding the roots of $f(\lambda)$, given the above relationship they can be computed efficiently using a Householder method. We also note that $\mathrm{tr}(\Lambda') = \mathrm{tr}(\Sigma'')$ and consequently

$$\lambda_{n+1}^{n+1} = \sum_i \lambda'_i - \sum_{i<n+1} \lambda_i^{n+1}.$$

Once we have computed the eigenvalues $\lambda^{''}$, we can compute the eigenvectors U'' as follows: we first note that $\forall k$ the system of linear equations

$$\Sigma'' x = \lambda_k^{n+1} x$$

where

$$x = \left[\begin{array}{cccc} x_1 & x_2 & \dots & x_{n+1} \end{array} \right]^T$$

involves n equations of the form

$$\lambda'_i x_i + q_i x_{n+1} = \lambda_k^{n+1} x_i.$$

Given that $\lambda_k^{n+1} \neq \lambda'_i$ it follows that if $x_{n+1} = 0$ then $\forall i, x_i = 0$ and thus it must be that $x_{n+1} \neq 0$. As the system has one degree of freedom, we can set $x_{n+1} = 1$. We then have from the equations

$$\lambda'_i x_i + q_i x_{n+1} = \lambda_k^{n+1} x_i$$

that

$$x_i = \frac{q_i}{\lambda_k^{n+1} - \lambda'_i},$$

and by normalizing x we obtain the k_{th} column of U''. Finally we can obtain the eigen-decomposition of Σ^{n+1}

$$\Sigma^{n+1} = (U'U'')\Lambda''(U'U'')^T.$$

4.1.2 Computational Efficiency

In order to empirically evaluate the derived eigen-decomposition update algorithm, we compare a C++ implementation against the LAPACK library's eigen-decomposition implementation. In figure 4.1.2 we see such a comparison of the CPU time required to compute Σ^{n+1}, from scratch, using the LAPACK library and using the proposed update algorithm, as a function of n.

4.1.3 Numerical Precision

In order to test the numerical precision of the proposed update method, we consider two experimental setups. In the first setup, results on which are shown in figures 5,6, we iteratively augment a symmetric positive definite matrix by inserting a vector as an extra row and an extra column and at each iteration update its eigen-decomposition using the proposed method; that is to say at each iteration the eigen-decomposition is an updated version of previously updated decompositions. Figure 5 shows the maximum relative eigenvalue error as compared to the decomposition estimated by LAPACK which is assumed to be accurate. Similarly figure 6 shows the maximum angle between corresponding eigenvectors of the proposed method and the ones computed by the LAPACK library. As can be seen, after 2000 iterations the *maximum* relative eigenvalue error is of the order of magnitude of $\sim 1\%$, while the maximum angle between corresponding eigenvectors is less than 0.001 degrees.

In figure 7 we show the maximum relative eigenvalue when the eigen-decomposition is updated from the decomposition computed at the previous iteration using LAPACK.

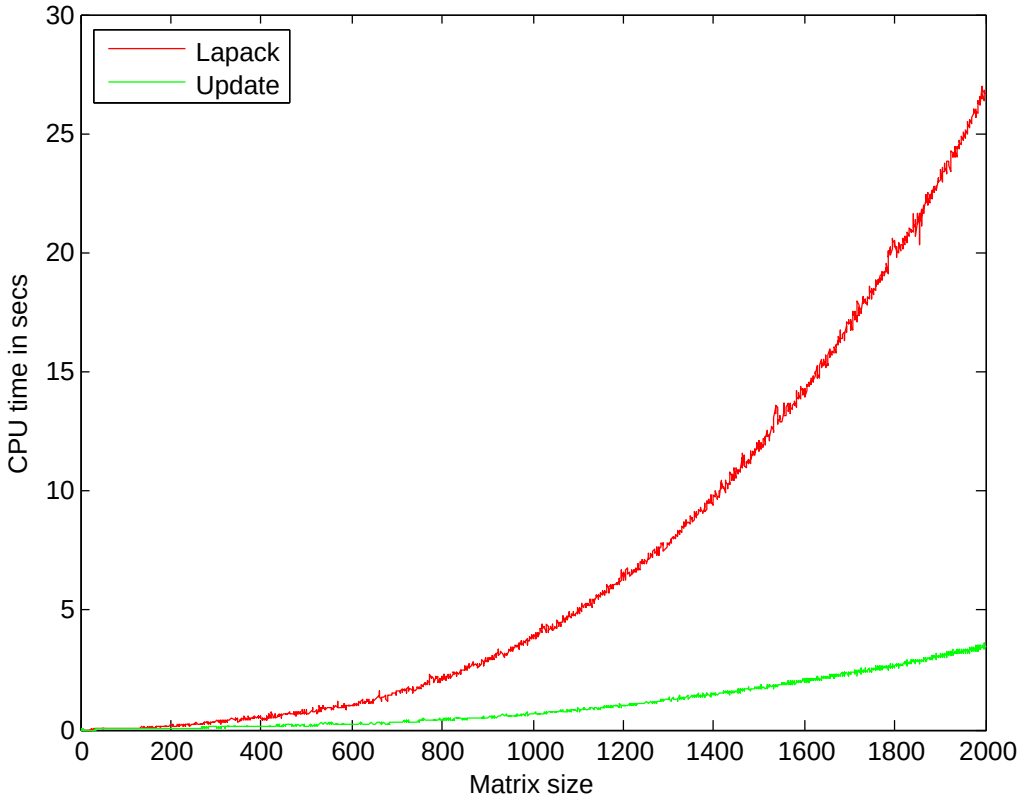

Figure 4: Comparison of computation time of the proposed update method and of computing the eigen-decomposition from scratch using the LAPACK library.

As can be seen the maximum relative error is of the order of magnitude of 10^{-10}. this value is related to the tolerance of the Newton method which was used to find the roots of equation 20 (which in these experiments we set to 10^{-10}). The maximum angle between corresponding eigenvectors was found to be 0, *i.e.* beneath double precision, in every case.

5. Experiments

In this section we present an empirical evaluation of the proposed algorithms. We first show on a synthetic controlled experiment that they behave as expected regarding groups of jointly informative features, and then provide results obtained on three popular real-world computer vision data sets.

5.1 Synthetic Examples

In order to show the importance of joint informativeness and the ability of the proposed algorithm to capture it we construct a simple synthetic experiment with a set of candidate features $F = \{X_1, X_2, X_3, X_4, X_5\}$ defined as follows:

$$\begin{aligned} X_1 &\sim N(0,1) + 10^{-1}Y \\ X_2 &\sim (2Y-1)X1 + N(0,1) \\ X_3 &\sim N(0,1) \\ X_4 &\sim N(0,1) + 10^{-1}Y \\ X_5 &\sim (2Y-1)X_4 + N(0,1) \end{aligned}$$

where Y is a classification label, $Y \sim B(0.5)$. Looking at the above marginals it can be seen that only X_1 and X_4 carry information regarding Y individually, X_2 and X_5 are very informative but only in conjunction with X_1 and X_4 respectively. X_3 is simply noise.

We generate $1,000$ synthetic data sets of $25,000$ data points each from the above distributions, and use the GC-approximation on the mutual information to select the features. In 49.2% of the experiments the algorithm ranks X_1 as the most informative feature. Even though X_4 would be the second most informative feature marginally, in every one of these experiments in the second iteration the algorithm chose the X_2 feature as it is jointly more informative when combined with X_1. Similarly, in 47.1% of the experiments the algorithm ranked X_4 first and in each of these experiments selected X_5 second. In every one of the runs X_3 was ranked as the least informative of the 5 features.

5.2 Data Sets

We report results on three standard computer vision data-sets which we used for our experiments:

CIFAR-10 contains images of size 32×32 of 10 distinct classes depicting vehicles and animals. The training data consists of $5,000$ images of each class. We pre-process the data as in (Coates and Ng, 2011) using the code provided by the authors. The original pool F of features consists of $2,048$ candidates.

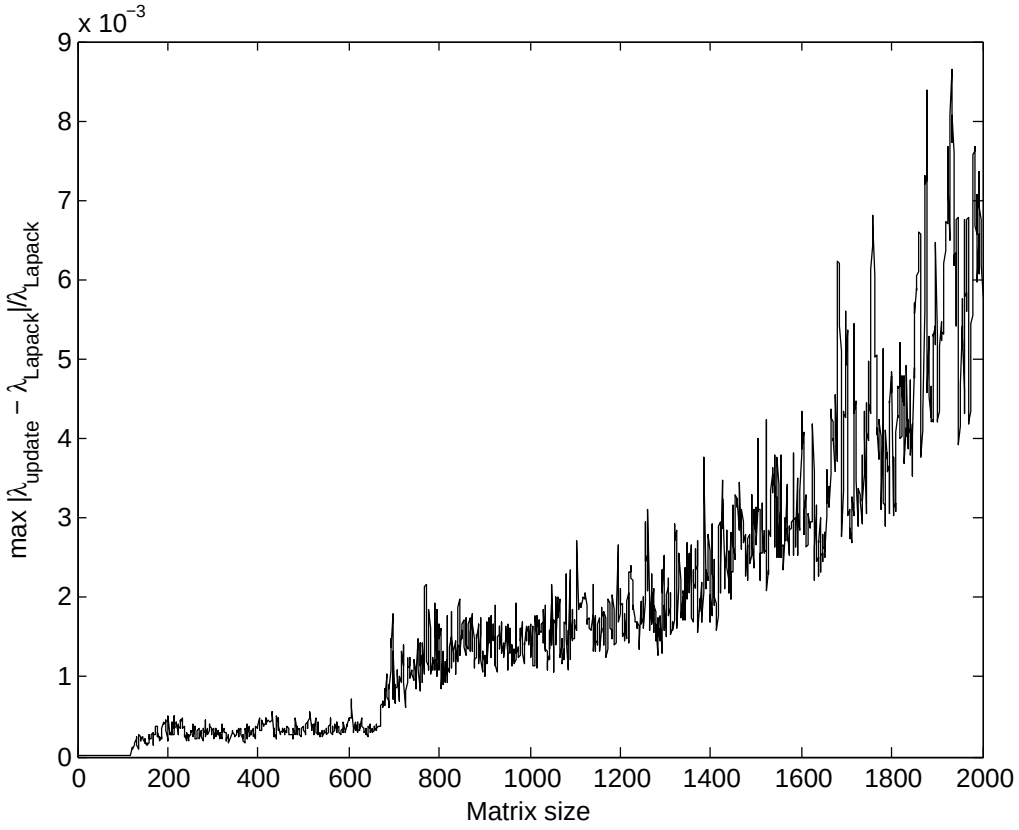

Figure 5: Numerical precision of the proposed update method when updating the decomposition iteratively. Maximum relative eigenvalue error.

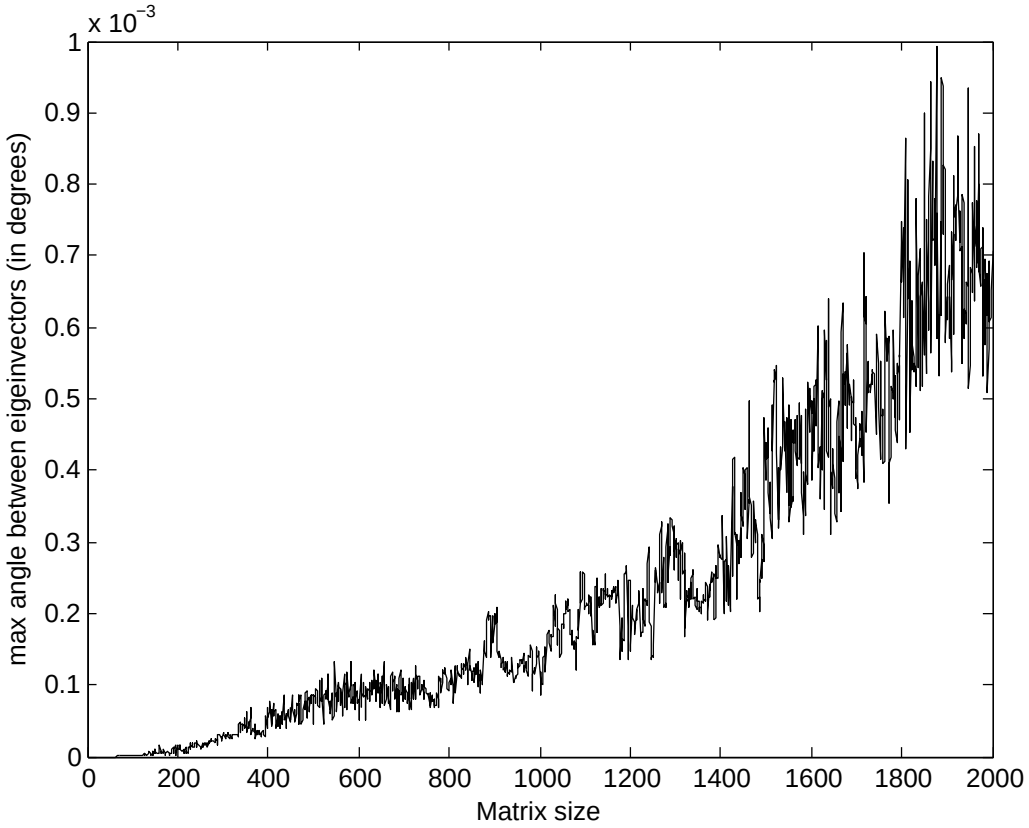

Figure 6: Numerical precision of the proposed update method when updating the decomposition iteratively. Maximum angle between corresponding eigenvectors.

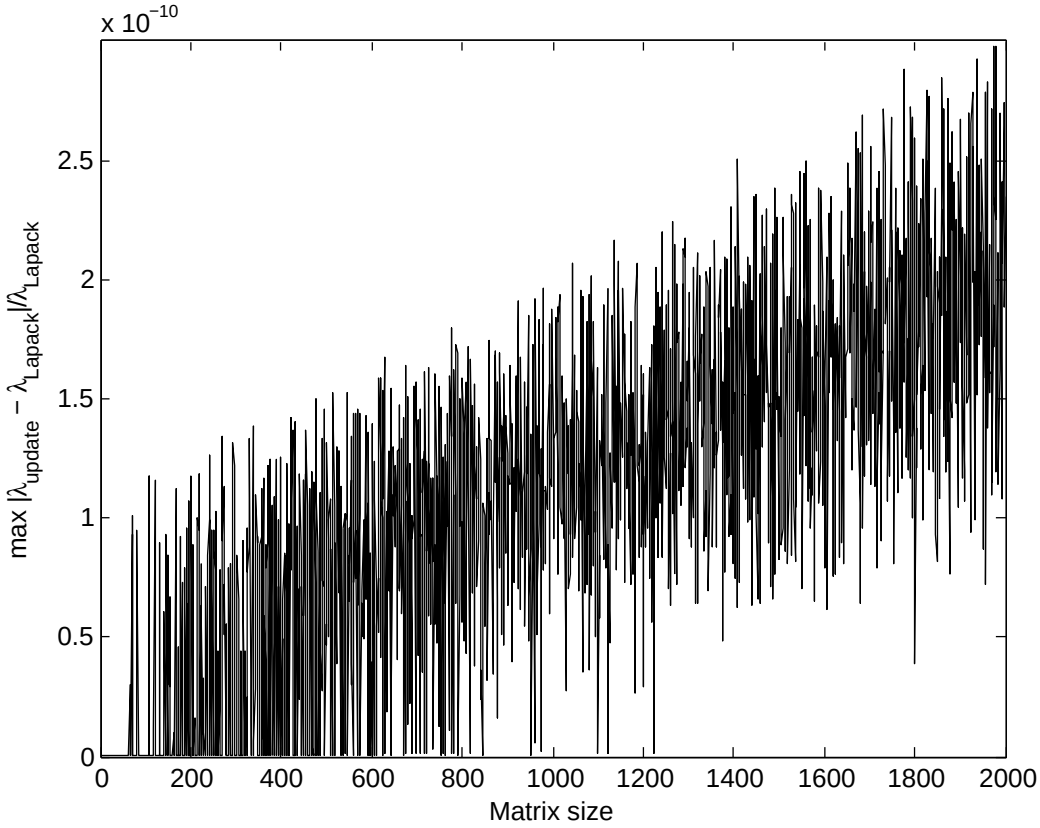

Figure 7: Maximum relative eigenvalue error when the eigen-decomposition is updated from the eigen-decomposition computed by LAPACK.

INRIA is a pedestrian detection data set. There are 12,180 training images of size 64×128 of pedestrians and background images. We use 3,780 HoG features that have been shown to perform well in practice (Dalal and Triggs, 2005).

STL-10 consists of images of size 96×96 belonging to 10 classes, each represented by 500 training images. As for CIFAR we pre-process the data as in (Coates and Ng, 2011), resulting in a pool F of 4,096 features.

5.3 Baselines

We compare the proposed feature selection methods against a number of baselines. The **Fisher**, **T-test**, χ^2, and **InfoGain** methods all compute statistics on individual features. In particular **InfoGain** calculates the mutual information of the individual features to the class, without taking into account their joint informativeness. As such, its comparison with our approaches is a very good indicator of the merit of joint informativeness and its effect on classification performance.

As noted in section 2, the **FCBF** (Liu and Yu, 2003) and **CFS** (Hall and Smith, 1999) baselines employ symmetric uncertainty criteria and check for pairwise redundancy of features. Similarly **MRMR** (Peng et al., 2005), uses mutual information to select features that have high relevance to the class while having low mutual information with the other selected features, thus checking for pairwise informativeness. Comparison with the proposed methods proves the importance of going beyond pairwise redundancy.

The **RelieFF** baseline (Robnik-Šikonja and Kononenko, 2003) looks at the nearest neighbors of random samples along the individual features. In order to compare with spectral clustering approaches we show results for (Wolf and Shashua, 2005), marked as **Spectral** in the tables, which we found to outperform (Zhao and Liu, 2007) in practice. Finally, we also show results for three wrapper methods, namely **SBMLR** (Cawley et al., 2006), which employs a logistic regression predictor, **CMTF** (Argyriou et al., 2008) which uses a sparsity inducing l_1-norm, and **GBFS** (Xu et al., 2014) which uses gradient boosted trees.

We compare these baselines against the four methods proposed here, namely maximizing the entropy (**GC.E**) or the mutual information (**GC.MI**) under the Gaussian compromise approximation, and maximizing the KL-based entropy (**GKL.E**) and mutual information (**GKL.MI**). In the case of the **GC** methods, when an iteration is reached where for all candidate features and for all classes the prior over the class variable is lower than the entropy of f^*, we halt the GC-approximation feature selection procedure and randomly select the remaining features.

5.4 Results

In tables 4, 5, 6, and 7, we show experimental results for the three data sets. In order to show the general applicability of the proposed methods, we combined the selected features with four different classifiers: AdaBoost with classification stumps, linear SVM, RBF-kernel SVM, and quadratic discriminant analysis (QDA). We show results for several numbers of selected features $\{10, 25, 50, 100\}$.

In each of these tables, for each data set and classifier we highlight the best three performing methods in bold, while underlining the best performing method. As can be seen

from these tables, **GC.MI** and **GKL.E** consistently rank amongst the top three methods, with **GC.MI** ranking in the top three 24 out of 48 times and first 8 times, and **GKL.E** being in the top three 27 out of 48 times and first 11 times. The only other comparable methods are **SBMLR** which ranks in the top three 17 out of 48 times and first 10 times and **GBFS** which ranks in the top three 20 out of 48 times and first 5 times. We note that both of these methods are wrapper methods.

Furthermore as can be seen in table 3, the running time of the proposed methods[3] is very competitive with respect to the more complex feature selection algorithms. The Gaussian Compromise algorithms are especially fast as they are two orders of magnitude faster than practically all other methods. We especially note that the **SBMLR** method which performs comparably in terms of accuracy is very slow when compared to **GC.MI**. Though the **MRMR** algorithm is faster than **GC.MI** on the INRIA data set, it performs considerably worse accuracy-wise on that data set; on the data sets where it does perform well (e.g. CIFAR) it is considerably slower than **GC.MI**. We also note that the **GBFS** method is quite slower than **GC.MI**.

The computation times provided were obtained with C++ implementations of the proposed methods. The **MRMR** algorithm is also implemented in C++, while the **Spectral** and **CMTF** baselines are implemented in MATLAB, as both these algorithms mainly use matrix algebra we believe these timings to be indicative. The remaining algorithms were implemented in Java. As noted in (Bouckaert et al., 2010) these implementations should be competitive in speed with C++ implementations.

	FCBF	MRMR	SBMLR	Spectral	GBFS	CFS	CMTF	RelieFF	**GC.MI**	**GKL.E**
CIFAR	621	56	1449	1379	95	4262	394	1652	**20**	486
STL	68	20	1002	367	41	409	208	2089	**5**	887
INRIA	247	**32**	88	1072	233	2516	459	2413	43	135

Table 3: (Approximate) Cost in CPU time of running the more sophisticated feature selection algorithms in order to select 100 features on the three data sets. We highlight in bold the fastest algorithm for each data set.

5.5 Finite Sample Analysis

The proposed methods all depend on the covariance matrices Σ, Σ^y. Of course, in practice, one rarely has access to the true covariance matrices of the underlying distributions but rather estimates based on using finite sets of samples. Given a $N \times D$ matrix P where each row is a sample from the underlying D-dimensional distribution, we symbolize

$$\hat{\Sigma}_N = \frac{1}{N} P^T P$$

the empirical estimation $\hat{\Sigma}_N$ of matrix Σ, computed from these N samples. The accuracy of this approximation is important to the success of the proposed methods. It can be shown

3. We note that for the **GKL.E/MI** methods we used an $O(n^2)$ (per feature per iteration) implementation, in practice and assuming access to adequate memory the method should be even faster.

	CIFAR				STL				INRIA			
AdaBoost	10	25	50	100	10	25	50	100	10	25	50	100
Fisher	29.23	36.96	42.07	49.06	31.86	35.78	39.72	41.81	86.90	89.83	90.38	91.45
FCBF	**37.77**	44.42	51.15	54.83	33.25	38.05	39.87	42.81	90.87	94.02	95.44	94.67
MRMR	**39.42**	**45.84**	49.76	54.85	32.24	39.61	40.61	43.00	81.53	88.48	93.48	94.91
χ^2	28.13	35.54	43.68	49.46	29.61	36.88	39.39	41.89	**92.81**	93.11	93.94	94.91
SBMLR	34.87	**45.08**	**52.17**	**56.70**	**34.22**	**41.26**	**44.65**	**47.15**	86.40	87.50	88.04	88.06
tTest	25.74	31.30	36.57	43.16	31.74	34.75	39.31	42.34	85.01	88.41	88.84	91.70
InfoGain	29.01	35.90	40.20	48.34	31.13	36.60	38.62	42.03	92.58	93.29	93.96	94.93
Spectral	19.90	25.13	33.18	40.44	19.06	26.30	33.52	38.51	92.78	93.69	93.92	94.83
RelieFF	28.13	34.64	40.85	47.70	33.91	37.46	42.79	45.22	91.79	**95.44**	95.83	96.43
CFS	33.50	38.96	44.58	54.22	30.75	38.40	41.85	44.39	89.69	92.60	**96.41**	**97.69**
CMTF	21.79	31.98	39.43	45.23	28.70	33.55	34.71	36.86	80.01	83.72	92.55	95.58
GBFS	32.02	40.20	48.87	54.34	30.96	38.56	42.30	45.57	**93.90**	**95.87**	**96.90**	**97.66**
GC.E	32.45	42.54	50.15	55.06	31.86	37.41	42.19	**46.99**	89.54	90.09	94.30	95.81
GC.MI	36.47	44.55	**51.44**	**55.39**	**36.50**	**40.79**	**43.82**	44.39	**95.04**	**95.87**	**96.68**	**97.30**
GKL.E	**37.51**	**46.41**	**52.11**	**56.41**	**34.76**	**39.71**	**43.49**	**46.46**	89.92	91.84	94.14	96.63
GKL.MI	33.71	40.04	47.17	51.12	33.00	38.80	42.13	43.58	92.18	93.09	95.21	96.15

Table 4: Test accuracy of an AdaBoost classifier trained on a different number of selected features $\{10, 25, 50, 100\}$ on the three data sets.

	CIFAR				STL				INRIA			
SVMLin	10	25	50	100	10	25	50	100	10	25	50	100
Fisher	25.19	33.53	39.47	48.12	26.09	30.79	34.63	38.02	92.55	**93.73**	94.03	94.68
FCBF	**33.65**	42.02	47.77	54.97	31.74	34.85	38.11	40.66	**94.14**	**96.03**	**96.03**	96.03
MRMR	**35.48**	**42.53**	46.02	52.64	**32.50**	39.06	43.69	49.36	79.85	84.18	91.73	93.91
χ^2	21.77	32.06	40.65	48.58	22.61	31.82	34.29	37.96	**92.94**	93.27	93.50	94.61
SBMLR	30.43	**42.60**	**51.41**	**56.81**	32.29	38.26	43.29	47.15	85.92	87.95	88.57	88.64
tTest	25.69	32.56	40.17	45.12	26.72	29.95	36.23	39.14	80.01	87.21	87.64	89.23
InfoGain	24.79	32.32	37.98	47.37	27.17	31.82	33.70	37.84	92.35	93.08	93.75	94.68
Spectral	17.19	23.14	32.78	42.60	18.91	26.55	32.65	38.24	**92.67**	93.57	93.64	94.44
RelieFF	24.56	30.60	38.17	46.51	29.16	32.40	38.05	42.94	90.99	**95.04**	**95.97**	96.36
CFS	31.49	36.46	42.17	51.70	28.63	34.45	38.54	41.88	88.64	91.68	96.11	**97.53**
CMTF	21.10	31.64	40.39	47.71	27.61	34.81	38.99	42.32	79.09	80.29	89.49	93.01
GBFS	28.37	38.18	45.89	52.36	30.78	**39.29**	**45.06**	**50.39**	76.79	92.55	95.38	**97.03**
GC.E	28.76	41.14	48.70	**55.16**	31.20	37.60	43.31	**49.75**	87.73	87.67	91.96	93.13
GC. MI	**34.02**	42.14	**49.16**	**55.07**	**32.50**	**39.75**	44.15	48.88	89.76	93.09	**95.71**	**96.45**
GKL.E	32.39	**43.26**	**50.12**	55.02	**33.44**	38.62	**44.27**	**50.54**	85.31	89.46	92.05	96.36
GKL. MI	28.67	34.65	43.30	48.69	32.16	**39.35**	**44.87**	47.96	85.66	90.99	92.14	95.16

Table 5: Test accuracy of a linear SVM trained on a different number of selected features $\{10, 25, 50, 100\}$ on the three data sets.

	CIFAR				STL				INRIA			
SVM-RBF	10	25	50	100	10	25	50	100	10	25	50	100
Fisher	29.11	39.22	46.05	54.68	34.71	40.13	43.87	45.77	92.44	93.55	93.38	92.97
FCBF	**40.48**	**51.15**	57.73	64.26	**38.86**	43.35	46.06	47.20	88.29	93.91	92.60	95.66
MRMR	**41.80**	**51.97**	57.31	62.14	**38.39**	**44.87**	47.02	48.92	80.07	79.99	88.89	90.48
χ^2	27.16	38.23	47.60	54.70	32.53	41.27	43.22	44.88	**92.78**	93.16	93.02	93.25
SBMLR	36.06	49.83	**60.32**	**64.97**	32.29	38.26	43.29	47.15	82.82	86.05	87.39	87.14
tTest	28.68	35.75	41.89	49.13	34.30	38.73	44.30	45.90	80.01	87.00	87.11	87.32
InfoGain	29.21	38.68	43.92	53.94	35.57	41.23	42.92	45.12	92.28	92.71	93.01	93.38
Spectral	22.89	30.92	40.41	49.75	24.80	32.91	40.11	43.70	92.67	93.09	92.85	93.29
RelieFF	29.49	37.08	45.39	53.96	38.22	42.36	47.27	**50.35**	90.62	**94.56**	**95.05**	95.20
CFS	35.50	43.74	50.98	61.01	35.32	42.72	**47.46**	49.82	88.34	91.31	**95.44**	**97.14**
CMTF	23.90	36.74	45.51	52.86	31.80	36.94	38.06	39.65	80.01	83.72	92.55	93.68
GBFS	34.98	45.07	54.70	61.27	33.65	**43.99**	**49.04**	**51.52**	**93.00**	**95.25**	**95.83**	**96.48**
GC.E	35.29	51.12	**60.34**	**65.76**	36.16	42.64	45.37	47.79	87.73	87.67	91.96	93.13
GC. MI	39.57	49.91	57.79	64.32	35.86	43.35	45.80	47.81	**94.26**	**94.17**	94.44	**95.76**
GKL.E	**39.84**	**52.80**	**60.94**	**65.64**	**39.67**	**46.31**	**50.06**	**52.89**	86.01	88.94	92.79	95.43
GKL. MI	34.49	43.09	51.48	56.54	35.95	41.65	45.27	45.86	91.03	91.91	93.36	93.98

Table 6: Test accuracy of a SVM with a RBF kernel when trained on a different number of selected features $\{10, 25, 50, 100\}$ on the three data sets.

	CIFAR				STL				INRIA			
QDA	10	25	50	100	10	25	50	100	10	25	50	100
Fisher	25.41	33.31	39.67	47.53	34.73	39.91	44.24	48.35	87.41	88.63	89.17	91.31
FCBF	**35.02**	43.97	52.32	**58.99**	**37.44**	41.89	45.70	48.89	**89.95**	**94.00**	**94.00**	94.00
MRMR	**36.19**	**44.54**	48.22	53.88	36.89	42.84	46.44	49.90	62.98	76.56	86.84	90.60
χ^2	21.81	31.85	39.39	47.75	32.71	40.45	43.64	47.25	87.85	88.20	89.30	91.75
SBMLR	31.71	43.46	**53.31**	**58.86**	36.89	**45.26**	**49.58**	**51.65**	76.30	80.36	81.00	81.49
tTest	26.34	33.39	39.16	45.33	33.92	40.09	45.05	47.63	76.16	82.50	82.85	85.23
InfoGain	22.38	31.61	37.65	46.47	34.17	40.94	44.51	47.81	87.99	88.08	89.49	91.77
Spectral	17.97	24.80	34.99	44.25	25.39	35.45	44.39	49.68	87.99	88.26	89.07	91.26
RelieFF	24.61	29.67	38.20	47.18	**37.57**	42.59	47.60	50.92	83.38	**92.14**	93.16	94.24
CFS	31.50	36.18	42.93	52.92	34.06	42.29	**48.45**	51.09	83.79	88.31	**94.00**	**96.66**
CMTF	20.61	31.98	41.04	48.60	33.76	**43.04**	47.51	51.02	61.04	72.31	89.23	92.67
GBFS	29.48	38.39	46.86	53.84	34.92	42.83	**47.89**	**51.26**	**91.19**	91.35	**94.15**	**96.02**
GC.E	31.01	**44.10**	**53.21**	58.41	33.76	**43.04**	47.51	51.02	85.06	86.31	91.22	94.10
GC. MI	33.68	43.02	51.84	**58.43**	35.31	42.24	47.21	49.88	**92.14**	**94.08**	**95.07**	**96.31**
GKL.E	**34.06**	**44.21**	**53.29**	57.98	**37.39**	**43.30**	47.39	**51.82**	79.30	86.21	90.83	95.74
GKL. MI	29.06	35.10	42.50	46.24	33.56	40.02	45.07	46.44	85.09	88.03	91.72	94.84

Table 7: Test accuracy using Quadratic Discriminant Analysis on a different number of selected features $\{10, 25, 50, 100\}$ on the three data sets.

(see theorem 5.39 in (Vershynin, 2012)) that in the case of a matrix with (sub)-Gaussian generated rows, we have $\forall t \geq 0$ with probability at least $1 - 2e^{-ct^2}$

$$\|\hat{\Sigma}_N - \Sigma\| \leq \max(\delta, \delta^2),$$

where

$$\delta = \frac{C\sqrt{D} + t}{\sqrt{N}}$$

and c, C are constants related to the sub-Gaussian norm of the rows of P. Replacing t with $C^{'}t\sqrt{D}$ in the above (see Corollary 5.50 in Vershynin (2012)), we have, for sufficiently large $C^{'}$, $\forall \epsilon \in (0, 1)$, and $\forall t \geq 1$ with probability at least $1 - 2e^{-ct^2}$

$$If \;\; N \geq C(t/\epsilon)^2 D \quad then \;\; \|\hat{\Sigma}_N - \Sigma\| \leq \epsilon.$$

Thus $N = O(D)$ samples are needed to sufficiently approximate the covariance matrix by the finite sample covariance matrix when the underlying distribution is sub-Gaussian, compared to $O(D \log D)$ for an arbitrary distribution (Corollary 5.52 (Vershynin, 2012)).

Based in this, in order to select d features, the proposed methods theoretically require $O(d)$ samples. This however assumes that the feature selection methods depend solely on the covariance matrices Σ. This holds true only for the GC-approximation in section 3.2.1, the KL-based bound depends both on Σ and Σ^{-1}. Furthermore, as shown in section 3.4, the more efficient implementation of the GC-approximation also depends on Σ^{-1}.

Unfortunately, estimating the precision matrix by taking the inverse of the sample covariance matrix is known to be unstable (Cai et al., 2016). Though a number of methods have been proposed to address this issue (Cai et al., 2011), their complexity makes them unsuitable in the present setting. To investigate whether this instability affects the performance of the proposed methods, we present in the following an empirical analysis of the effect of sample size on prediction performance.

5.5.1 Empirical Evaluation

In order to assess the influence of sample set size on performance, we consider the accuracy on the test set of a linear SVM. Specifically we perform feature selection using a subset of the training data by selecting uniformly at random without replacement. Thus the relevant sample covariance matrices are estimated using these smaller sets. We then train a linear SVM using the selected subset of features but using the entire set of samples; this is done to avoid any influence of sample size on the training of the SVM and by extension on the final results.

In figure 8 we show the empirical results on the three data sets (STL, CIFAR, INRIA) in the case where feature selection is performed using the GC-approximation. As can be seen, in the case of the STL and CIFAR data sets the feature selection method proves to be very robust with regards to sample set size; performance degrades only slightly when the sample set size is very small (50 samples per class). On the contrary, in the case of the INRIA data set, we see that the method does not prove to be so robust and the performance suffers. We note that in the plot for the INRIA data set, the x-axis relates the number of samples in the positive class, the number of samples sampled from the negative class was chosen to preserve the class ratio ($\sim 6/1$).

Similarly in figure 9 we show results for the KL-bound case. Here we see that sample size is more influential. A possible explanation is that though the feature selection method arising from the GC-approximation involves the estimation of $|Y| + 1$ inverse covariance matrices, in the case of the KL-bound feature selection method involves $2|Y|$ inverses. Examining the plots in figure 9, we see that in the case of the STL and CIFAR data sets there is some degradation of performance for small sample set sizes, though the performance quickly reaches that of the full set as the subset size increases. In the case of the INRIA data set however we see that the method performs considerably worse when only a subset of the data set is used to perform feature selection.

6. Conclusion

The present work concentrates on developing tractable algorithms to exploit information theoretic criteria for feature selection. The proposed methods focus on feature selection in the context of classification and demonstrate that it is possible to choose features that are jointly informative by careful density modeling and algorithmic implementation. Thus the joint mutual information of variables can in fact be employed efficiently for feature selection as opposed to using only the mutual information of marginal or pairwise distributions as has typically been used in the literature.

The proposed methods rely on modeling the conditional joint distributions of the features given the class to predict and subsequently maximizing either upper bounds on the information theoretic measures or relevant approximations. To reduce the computational cost of a forward feature selection scheme incorporating these criteria, we have proposed efficient implementations for both approaches, so that they are competitive with other state-of-the-art methods in terms of speed. We have also presented a novel method for updating the eigen-decompositions of a specific family of matrices (and updates) which our GC-approximation feature selection algorithm exploits.

Empirical results show the methods to be competitive with current state-of-the-art with respect to prediction accuracy. Furthermore, an empirical analysis of the performance of these methods in connection with the number of samples in the data sets has shown them to be relatively robust in this respect.

Acknowledgments

This work was supported by the Hasler Foundation through the MASH-2 project.

References

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

Remco R. Bouckaert, Eibe Frank, Mark A. Hall, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. WEKA—Experiences with a Java open-source project. *Journal Machine Learning Research (JMLR)*, 11:2533–2541, 2010.

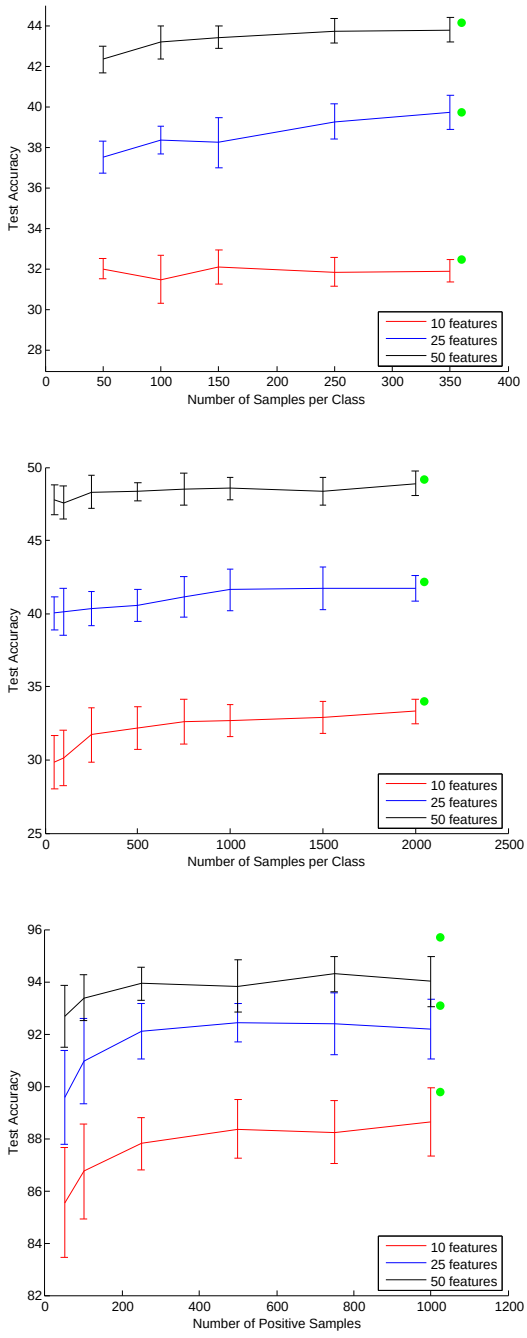

Figure 8: Effect of sample set size on performance when using the GC-approximation for the top) STL, middle) CIFAR, and bottom) INRIA data sets. The green circles mark the performance of the method when the entire data set is used.

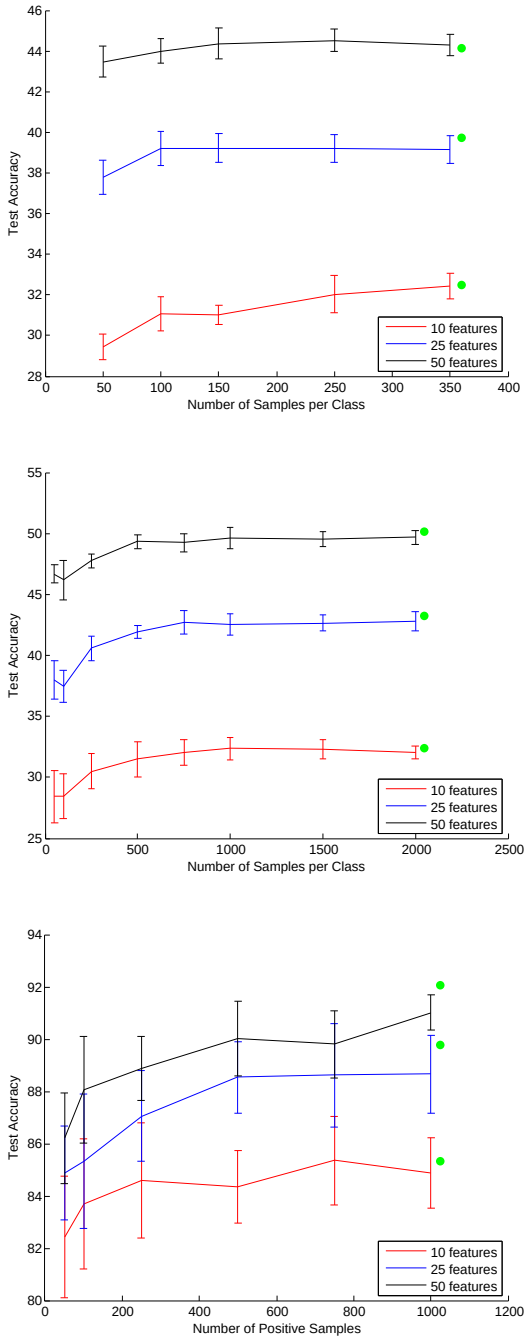

Figure 9: Effect of sample set size on performance when using the KL-bound for the top) STL, middle) CIFAR, and bottom) INRIA data sets.The green circles mark the performance of the method when the entire data set is used.

T. Cai, W. Liu, and H. Zhou. Estimating sparse precision matrix: optimal rates of convergence and adaptive estimation. *Annals of Statistics*, 44:455–488, 2016.

T.T. Cai, W. Liu, and X. Luo. A constrained l1 minimization approach to sparse precision matrix estimation. *Journal of the American Statististical Association*, 494:594–607, 2011.

Gavin C. Cawley, Nicola L. C. Talbot, and Mark Girolami. Sparse multinomial logistic regression via Bayesian l1 regularisation. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 209–216, 2006.

A. Coates and A. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 921–928, 2011.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the Conference On Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, 2005.

A. Das and D. Kempe. Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1057–1064, 2011.

S. Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 74–81, 2001.

F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research (JMLR)*, 5:1531–1555, 2004.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research (JMLR)*, 3:1157–1182, 2003. ISSN 1532-4435.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

Mark A. Hall and Lloyd A. Smith. Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. In *Proceedings of the International Florida Artificial Intelligence Research Society Conference*, pages 235–239, 1999.

J.R. Hershey and P.A. Olsen. Approximating the kullback leibler divergence between Gaussian mixture models. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 4, pages IV–317–IV–320, 2007.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Yi Jiang and Jiangtao Ren. Eigenvector sensitive feature selection for spectral clustering. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 114–129, 2011.

L. Lefakis and F. Fleuret. Jointly informative feature selection. In *Proceedings of the international conference on Artificial Intelligence and Statistics (AISTATS)*, 2014.

H. Liu and L. Yu. Feature selection for high-dimensional data: a fast correlation-based filter solution. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 856–863, 2003.

Dimitris Margaritis. Toward provably correct feature selection in arbitrary domains. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 1240–1248, 2009.

H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(8):1226–1238, 2005.

Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 53(1-2):23–69, 2003.

Irene Rodriguez-Lujan, Ramon Huerta, Charles Elkan, and Carlos Santa Cruz. Quadratic programming feature selection. *The Journal of Machine Learning Research (JMLR)*, 11: 1491–1516, 2010.

Le Song, Alex Smola, Arthur Gretton, Justin Bedo, and Karsten Borgwardt. Feature selection via dependence maximization. *Journal of Machine Learning Research*, 13(1): 1393–1434, 2012.

Mingkui Tan, Li Wang, and Ivor W. Tsang. Learning sparse SVM for feature selection on very high dimensional datasets. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1047–1054, 2010.

Kari Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research (JMLR)*, 3:1415–1438, 2003.

Nuno Vasconcelos. Feature selection by maximum marginal diversity: optimality and implications for visual recognition. In *Proceedings of the Conference On Computer Vision And Pattern Recognition (CVPR)*, 2003.

R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *Compressed Sensing*, pages 210–268, 2012.

Lior Wolf and Amnon Shashua. Feature selection for unsupervised and supervised inference: the emergence of sparsity in a weight-based approach. *Journal of Machine Learning Research (JMLR)*, 6:1855–1887, 2005.

Zhixiang Xu, Gao Huang, Kilian Q Weinberger, and Alice X Zheng. Gradient boosted feature selection. In *Proceedings of the international conference on Knowledge discovery and data mining (SIGKDD)*, pages 522–531. ACM, 2014.

Makoto Yamada, Wittawat Jitkrittum, Leonid Sigal, Eric P Xing, and Masashi Sugiyama. High-dimensional feature selection by feature-wise kernelized lasso. *Neural computation*, 26(1):185–207, 2014.

Zheng Zhao and Huan Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1151–1157, 2007.