Never Use PCA for Subsequent Classification, Low-Rank Optimal Linear is Better

Joshua T. Vogelstein, Minh Tang, Da Zheng, Randal Burns, Mauro Maggioni

Abstract

In the 21st century high-dimensional observations abound. In classical statistics, it is well known that as the dimensionality of a dataset increase, the number of samples required to estimate properties thereof increases even faster. A number of techniques have been developed to stave off this "curse of dimensionality", including dimensionality reduction and regularization. Mitigating this curse is especially demanding in supervised learning problems, such as classification. The challenge in finding a low-dimensional discriminant boundary in the face of high-dimensional observations is that direct searches are computationally infeasible. Therefore, previous art have taken one of two approaches. First, one can ignore the discriminant problem, and try to reduce the dimensionality of the data—for example, using principal components analysis or manifold learning techniques—in hopes that one has not discarded the discriminant information. Second, one directly solve the problem after making a number of overly restrictive assumptions of the data, such as independence and sparsity. Here, we demonstrate via a simple geometric argument that one can algebraically compute a low-dimensional embedding that is approximately optimal with high probability in many settings. The intuition is that we can project the data both in the directions of maximal variance (ignoring the classification task) and the linear direction of maximal discrimination (ignoring the variance). The result is an extremely computationally efficient supervised manifold learning method that we call "Linear Optimal Low-Rank" (LOL). LOL outperforms state-of-the-art algorithms in a wide variety of settings, including both those for which theoretical arguments predict it should, and in much more general settings for which we are unable as yet to obtain solid theoretical footing. This includes both simulated settings, where our intuition enables us to characterize a number of simple variants designed for more complex settings, as well as several benchmark problems. In particular, LOL outperforms several reference algorithms in terms of both efficiency and accuracy on said benchmarks. We therefore believe that LOL could be useful in myriad applications, and many extensions are readily available.

Supervised learning—the art and science of estimating statistical relationships using labeled training data—is a crucial tool in scientific discovery. Supervised learning has been enabled a wide variety of basic and applied findings, ranging from discovering biomarkers in omics data [**?**] to object recognition from images [**?**]. A special case is classification; a classifier predicts the "class" of a novel observation via partitioning the space of observations (for example, predicting male or female from MRI scans). One of the most foundational and important approaches to classification is called "Fisher's Linear Discrimnant Analysis" (LDA) [**?**]. One of the most compelling aspects of LDA is that it is based on very simple geometric reasoning. In particular, it is guaranteed to find the optimal linear classifier under the assumption that the data are Gaussian. Moreover, it can be applied no matter how many classes are present. Finally, its solution is analytic, therefore obviating costly numerical optimizations and enabling scalability. This solid theoretical foundation, easy interpretation, computational efficiency, and widespread utility perhaps contribute to LDA's longevity, as it remains one of *the* reference classifiers nearly a century later.

Modern scientific datasets, however, present challenges for classification that were not addressed in Fisher's era. Specifically, the dimensionality of datasets is quickly ballooning. Currently, across many scientific disciplines, the raw data might consist of millions of features or dimensions; for example, an entire genome or connectome. While the dimensionality of these data have increased precipitously, the sample sizes have not witnessed a concommitant increase. This "large p, small n" problem is disatrous for many classical statistical approaches because they were designed with "small p, large n" in mind. For example, LDA estimates a hyperplane in $p-1$ dimensions when the data are p dimensional. But there are an infinite number of $p-1$ dimensional hyperplanes that fit the data exactly when $p < n$. To see this, imagine fitting a line to a single point, or a plane to two points. In each case, one can choose any rotation, and still fit the data. Doing so, however, requires the algorithm to make a choice that is independent of the existing data, and therefore unlikely to be correct. This problem is typically called "over-fitting".

Many statistical and machine learning approaches have been developed to combat this over-fitting issue when the dimensionality exceeds the sample size. Perhaps the most prominent strategy is to use principal components analysis (PCA) [**?**] to "pre-process" the data, that is, to reduce the dimensionality of the data prior to a subsequent classification using the much lower dimensional data. While hugely successful, PCA is a wholly *unsupervised* dimensionality reduction technique, meaning that PCA does not utilize the class labels while learning the low dimensional representation. A closely related approach for supervised learning is called "reduced-rank LDA" (). As we will show below, while PCA does not optimally utilize the class labels in its low-dimensional representation, actually *discards* this information.

A different strategy that focuses on utilizing the lable information is called "sparsity". Sparse methods find a small subset of the original features to use for subsequent inference [**?**]. There are many such approaches (often called "feature selection" or "feature screening"); the advantage of these approaches is that the result is sometimes more easily interpretable. The disadvantage, however, is that exactly solving the problem is computationally intractable, requiring time that is exponential in the number of features. That said, varous approximations enable efficient algorithms that have provable guarantees under certain assumptions. LASSO is a particularly popular algorithm that has these properties. Unfortunately, LASSO still cannot run on millions of dimensions, and it has a hyper-parameter that requires careful tuning, and even then often produces spurious answers even when the unrealistically strict assumptions are met [**?**]. A more recent approach is called "regularized optimal affine discriminant" (ROAD). ROAD finds the optimal sparse dimensionaly reduction under certain gaussian assumptions. ROAD, however, can only be applied in two-class settings, and requires solving a computationally costly numerical optimization problem, and thus does not scale to large dimensionality.

Thus, scientists and data scientists desire a method that has a simple geometric intuition even in high-dimensions and can be applied when there are any number of classes, but also has theoretical gurantees, without making addition assumptions about the data, and can scale to arbitrarily high-dimensional data in a computationally efficient fashion.

To address these concerns, we describe a novel solution, called Low-Rank Optimal Linear Discriminant Analysis (LOL). To understand LOL, consider the following simplified scenario. Assume there exists two classes of data, both of which are sampled according to a high-dimensional Gaussian distribution, and both have the same covariance matrix. Assuming that there are the same number of observations from

each class (without loss of generality) implies that the only difference between the two classes is the mean. Figure **??** shows three different examples of this setting, each with 100 total samples in 1,000 dimensions (only the first two dimensions are shown). In panel **(A)**, each dimension is independent of the others, but the second dimension is different: the means in the second dimension are further away from one another than they are in all the others, and the variance in that dimension is smaller. This setting, which we called "stacked cigars", is easy of all of the approaches described above. In particular, PCA finds the second dimension, because it is the longest and therefore contains the most variance.

We demonstrate LOL outperforms—or at least performs no worse than—several reference benchmarks along every statistical consideration mentioned above. This includes proving the improvement over several standards, along both statistical considerations (bias and variance) for the simple Gaussian setting mentioned above. Moreover, we provide a distributed implementation that improves on previous computational capabilities in terms of: stability, time, space, and scalability under more general settings. Moreover, numerical experiments quantitatively demonstrate the improvement of LOL over reference methods on a wide range of simulated experiments that satisfy our theoretical assumptions. By virtue of LOL being built upon geometric intuition, it is easy to extend it along various dimensions. Additional numerical experiments demonstrate that practice matches our intuition, in particular, generalizations of LOL can easily be constructed to match different geometric assumptions of the data. Indeed, even when data are sampled from distributions outside our theory and intuition, we see that LOL achieves smaller error than its competitors. In fact, we can extend our intuition outside of classification problems to other problems, including wide regression and hypothesis testing. A trivial modification of LOL leads to regression and testing procedures that also outperform the natural competitors. Computationally, LOL is always numerically stable, and requires no more computational space, and smaller computational time, than both Fisherfaces and sparse methods. Moreover, we provide a highly scalable implementation that efficiently runs on terabyte datasets with hundreds of thousands or millions of features. Finally, we test LOL against a set of standard methods on four different benchmark real datasets. For any of the four different settings, there was not a single method that outperformed LOL on both accuracy and time.

In addition to the above quantitative properties, LOL is intuitive, only has a single hyper-parameter which is exceedingly easy to tune, and we provide free and open source implementations, in MATLAB, R, and FlashX, as well as pre-configured environments that will enable anybody to run LOL on commodity machines, including laptops, workstations, and cloud instances. LOL is not likely to necessarily be the best algorithm on any given new high-dimensional challenge problem. Rather, the arguments and methodology developed herein provide a framework for developing big supervised manifold learning algorithms to tackle the data science problems of the next century.

Results

An Illustrative Real Data Example of Supervised Linear Manifold Learning

A general strategy for high-dimensional classification proceeds as schematized in Figure **??**: (A) obtain/select n training samples of the data, (B) learn a low dimensional projection, (C) project m testing samples onto the lower dimensional space, (D) classify the embedded testing samples using a classifier. We consider four different linear dimensionality reduction methods—LASSO, LDA, PCA, and LOL—each of which we compose with a classifier to form high-dimensional classifiers.[1]

To demonstrate the utility of LOL, we first consider one of the most popular benchmark datasets ever, the MNIST dataset [**?**]. This dataset consists of $n = 60,000$ examples of images of the digits 0 through 9. Each such image is represented by a 28×28 matrix, which means that the observed (or ambient) dimensionality of the data is $p = 28^2 = 784$. Because we are motivated by the $n \ll p$ scenario, we subsample the data to select $n = 300$ examples of the numbers 3, 7, and 8 (100 of each). We then apply all four approaches to this subsample of the MNIST dataset, learning a projection, and embedding $m = 500$ testing samples, and classifying the resulting embedded data.

---

[1]Although LASSO is not a 2-step method (where embedding is learned first, and then a classifier is applied), adaptive lasso [**?**] and its variants improve on lasso's theoretical and empirical properties, so we consider such an approach here.
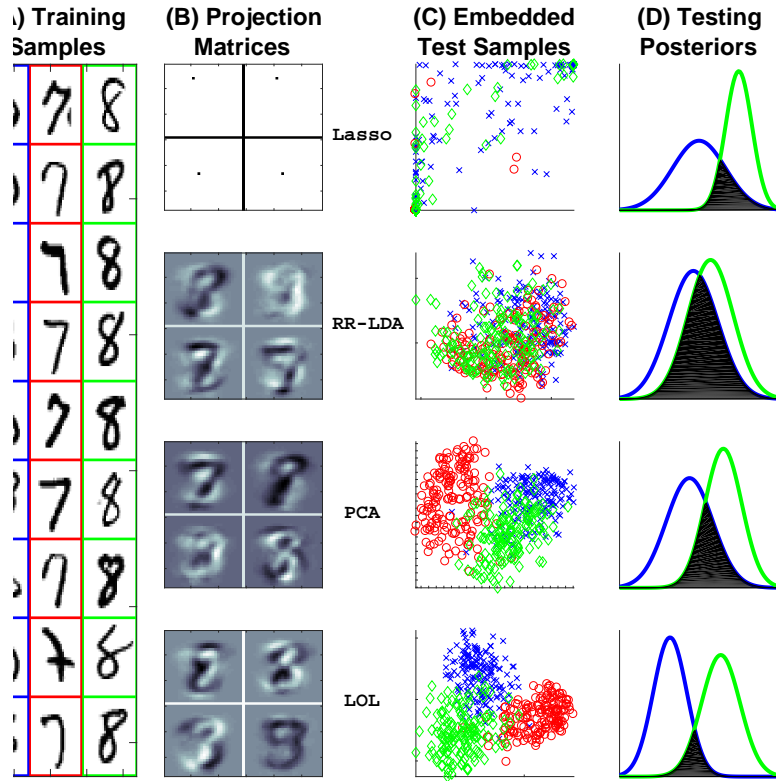
Figure 1: Illustrating four different classifiers—LASSO (top), LDA (second), PCA (third), and LOL (bottom)—for embedding images of the digits 3, 7, and 8 (from MNIST), each of which is $28 \times 28 = 784$ dimensional. **(A)** A subset of training samples (boundary colors are only for visualization purposes). **(B)** The first four projection matrices learned by the different approaches on 100 training samples from each class. Note that LASSO is sparse and supervised, PCA which is dense and unsupervised, and both LDA and LOL are dense and supervised. **(C)** Embedding 500 test samples into the top two dimensions using each approach. Digits color coded as in (A). **(D)** The estimated posterior distributions for 3 and 8 of test samples after projecting data into two dimensions via each method and using Lasso or LDA to classify. The filled area is the estimated error rate; the goal of any classification algorithm is to minimize that area. Clearly, LOL exhibits the best separation after embedding, which results in the best classification performance.

4

LASSO, by virtue of being a sparse method, finds the pixels that most discriminate the 3 classes. The resulting embeddings mostly live along the boundaries, because these images are close to binary, and therefore, images either have or do not have a particular pixel. Indeed, although the images themselves are nearly sparse (over 80% of the pixels in the dataset have intensity ≤ 0.05), a low-dimensional discriminant boundary does not seem to be sparse. LDA does poorly, essentially failing to discriminate these digits at all. This is because the low dimensional embedding implicit in LDA ignores the difference of the means, only determining the dimensions that maximize variance. In this setting, it seems the directions that maximally separate the classes are not well aligned with the directions of maximal variance. PCA does reasonably well, as it implictly includes the discriminant directions by not subtracting them out. LOL is our newly proposed supervised linear manifold learning method. The projection matrices it learns look qualitatively much like those of LDA and PCA. This is not surprising, as they are all linear combinations of the training examples. The resulting embeddings however, look quite different. The three different classes are very clearly separated by even the first two dimensions. The result of these embeddings yields classifiers whose performance is obvious from looking at the embeddings: LOL achieves significantly smaller error than the other approaches. This numerical experiment justifies the use of supervised linear manifold learning, we next build a geometric intuition for these methods in simple simulated examples, to better illustrate when we can expect LOL to outperform other methods, and perhaps more importantly, when we expect LOL to fail.

Linear Gaussian Intuition

The above real data example suggests the geometric intuition for when LOL outperforms its sparse and un-supervised counterparts. To further investigate, both theoretically and numerically, we consider the simplest setting that illustrates the relevant geometry. In particular, we consider a two-class classification problem, where both classes are distributed according to a multivariate normal distribution, where the two classes have the same covariance matrix. so that the only difference between the classes is their means (we call this the Linear Discriminant Analysis (LDA) model; see Methods for details).

Consider what the optimal projection would be in this scenario. The optimal low-dimensional projection is analytically available as the dot product of the difference of means and the inverse covariance matrix, $A_* = \delta^\top \Sigma^{-1}$ [?] (see Methods for derivation). PCA, the dominant unsupervised manifold learning method, utilizes only the covariance structure of the data, and ignores the difference between the means (because PCA for classification operates on the class-conditionally centered data, see Methods for details). In particular, PCA would project the data on the top d eigenvectors of the covariance matrix. **The key insight of our work is the following: we can combine the difference of the means and the covariance matrix in a simple fashion, rather than just the covariance matrix, to find a low dimensional projection.** Naïvely, this should typically improve performance, because in this stylized scenario, both are important. The F2M literature has a similar insight, but a different construction that requires the dimensionality to be smaller than the sample size [? ? ? ? ?]. We implement our idea by simply concatenating the difference of the means with the top d eigenvectors of the covariance matrix. This is equivalent to first projecting onto the difference of the means vector, and then projecting the residuals onto the first d principle components. Thus, it requires almost no additional computational time or complexity over that of PCA, rather, merely estimates the difference of the means. In this sense, LOL can be thought of as a very simple "supervised PCA".

Figure **??** shows three different examples of data sampled from the LDA model to geometrically illustrate this intuition. In each, we sample $n = 100$ training samples in $p = 1000$ dimensional space, so $n \ll p$. Figure **??**A shows an example we call "stacked cigars". In this example the covariance matrix is diagonal, so all ambient dimensions are independent of one another. Moreover, the difference between the means and direction of maximum variance are both large along the same dimensions (they are highly correlated with one another). This is an idealized setting for PCA, because PCA finds the direction of maximal variance, which happens to correspond to the direction of maximal separation. However, in the face of high-dimensional data, PCA does not weight the discriminant directions sufficiently, and therefore performs only moderately well.[2] Because all dimensions are independent, this is a good scenario for sparse methods. Indeed, ROAD, a sparse classifier

---

[2]When having to estimate the eigenvector from the data, PCA performs even worse. This is because when $n \ll p$, PCA is an inconsistent estimator with large variance [? ?].
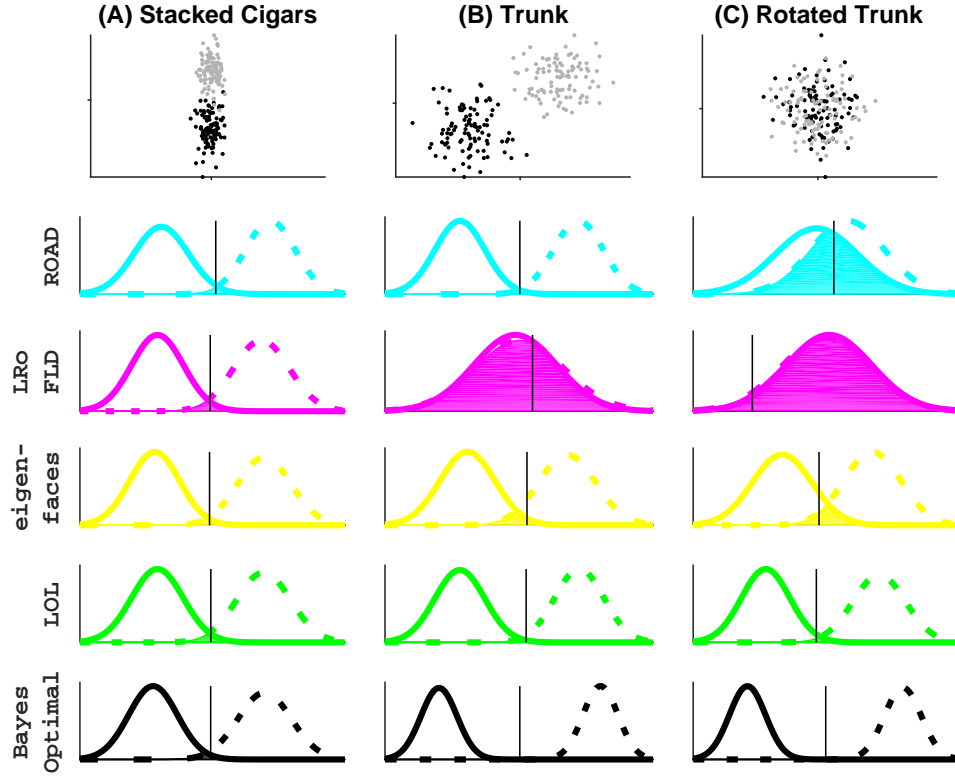
Figure 2: LoL achieves near optimal performance for a wide variety of distributions. Each point is sampled from a multivariate Gaussian; the three columns correspond to different simulation parameters (see Methods for details). In each of 3 simulations, we sample $n = 100$ points in $p = 1000$ dimensions. And for each approach, we embed into the top 20 dimensions. Note that we use the sample estimates, rather than the true population values of the parameters. The five columns show (in decreasing order): **Row 1**: A scatter plot of the first two dimensions of the sampled points, with class 0 and 1 as black and gray dots, respectively. **Row 2** FLD ○ PCA. **Row 3** ROAD, a sparse method designed specifically for this model [**?**]. **Row 4** LoL, our newly proposed method. **Row 5** the Bayes optimal classifier, which is what all classifiers strive to achieve. **(A)** The mean difference vector is aligned with the direction of maximal variance, maxing it ideal for both PCA to discover the discriminant direction and a sparse solution. In this setting, the results are similar for all methods, and essentially optimal. **(B)** The mean difference vector is orthogonal to the direction of maximal variance, making PCA fail, but sparse methods and LoL can still recover the correct dimensions, achieving nearly optimal performance. **(C)** Same as B, but the data are rotated, in this case, only LoL performs well. Note that LoL is closest to Bayes optimal in all three settings.

designed for precisely this scenario, does an excellent job finding the most useful ambient dimensions. LoL does the best of all three approaches, by using both the difference of the means and the covariance. This is obvious upon comparing each approach to the Bayes optimal solution for this setting, which is analytically available due to its simplicity.

Figure **??**B shows an example which is a worst case scenario for using PCA to find the optimal projection for classification. In particular, the variance is getting larger for subsequent dimensions, $\sigma_1 < \sigma_2 < \cdots < \sigma_p$, while the magnitudes of the difference between the means are decreasing with dimension, $\delta_1 > \delta_2 < \cdots > \delta_p$. Thus, for any truncation level, PCA finds exactly the *wrong* directions. ROAD is not hampered by this problem, it is also able to find the directions of maximal discrimination, rather than those of maximal variance. Again, LoL, by using both the means and the covariance, does extremely well.

Figure **??**C is exactly the same as B, except the data have been randomly rotated in all 1000 dimensions. This means that none of the original coordinates have much information, rather, linear combinations of them do. This is evidenced by observing the scatter plot, which shows that the first two dimensions fail to disambiguate the two classes. PCA, being rotationally invariant, fails in this scenario as it did in B. Now,

there is no small number of ambient dimensions that separate the data well, so ROAD also fails. LOL is unperturbed by this rotation; in particular, it is able to "unrotate" the data, to find dimensions that optimally separate the two classes.

Statistical Theory

The above numerical experiments provide the intuition to guide our theoretical developments.

Theorem 1. LOL *better than (or as good as)* PCA, *under the* LDA *model when the parameters are provided.*

In words, it is better to incorporate the mean difference vector into the projection matrix. The *degree* of improvement is a function of the embedding dimension d, the ambient dimensionality p, and the parameters (see Methods for details and proof), but the *existence* of an improvement, or at least no worse performance, is independent of those factors. Note that better here includes both bias and variance. In the population version of these two methods, there is no variance, so only the bias is important. Moreover, we know the conditions under which LOL is strictly better than PCA, they are formally stated in Assumption **??** (in Methods). Informally, LOL is better than PCA whenever the angle between the mean vector and the top d principal components is large enough. Under reasonable assumptions, in the appendix, we show that this happens 84% of the time.

Numerical Experiments Extending Our Theoretical Results

In the previous section, we proved that LOL is often better than PCA, and never worse under the LDA model, regardless of observed dimensionality, embedded dimensionality, and parameters, assuming the parameters are given. In this section, we numerically investigate what happens when the parameters must be estimated from training data, and in particular, from $n = 100$ training samples, when the observed dimensionality is $p = 100$. These results strengthen our theoretical results by providing exact magnitudes of performance, rather than merely better than guarantees, in addition to operating in a finite sample setting, more akin to real data scenarios.

LDA **Model** We begin by investigating three scenarios that satisfy the LDA model assumptions required by our proofs. First, consider the rotated trunk example from Figure **??**C as well as a "Toeplitz" example, as depicted in Figures **??**A and B, respectively. In both cases, we compare LOL to FLDoPCA and ROAD. In both scenarios, for all dimensions, LOL achieves a lower error rate than either of its competitors.

Multiple Classes LOL can trivially be extended to > 2 class situations. Naïvely it may seem like we would need to keep all pairwise differences between means. However, given k classes, the set of all k^2 differences is only rank $k - 1$. In other words, we can equivalently compute the distance of each mean to a single mean (in practice, for minimizing variance reasons, we use the class which has the maximum number of samples (breaking ties randomly). Figure **??**C shows a 3-class generalization of the rotated Trunk example from Figure **??**A. While LOL uses the additional class naturally, many previously proposed high-dimensional classifiers, such as ROAD, natively only work for 2-classes. Thus, we only compare LOL to FLDoPCA here. As before, LOL significantly outperforms FLDoPCA for all embedding dimensions.

Generalizations of LOL

The above simulations were all under the LDA model, for which we have theoretical guarantees that LOL should outperform FLDoPCA. In this section, we explore simulation settings that extend beyond the geometric constraints we required for our proofs, to develop an even deeper understanding of LOL.

Fat Tails Figure **??**D shows a sparse example with "fat tails" to mirror real data settings better. More specifically, each class is the sum of multiple Gaussians, with the same mean, but different covariances (see Methods for details). The qualitative results are consistent with those of A and B, even though the setting is no longer exactly the setting under which we have theoretical confirmation. More specifically, LOL outperforms FLDoPCA and ROAD for all dimensions.

QDA Sometimes, it makes more sense to model each class as having a unique covariance matrix, rather than a shared covariance matrix. Assuming everything is Gaussian, the optimal classifier in this scenario is called Quadratic Discriminant Analysis (QDA) [**?**]. Intuitively then, we can modify Lᴏʟ to compute the eigenvectors separately for each class, and concatenate them (sorting them according to their singular values). Moreover, rather than classifying the projected data with Lᴅᴀ, we can then classify the projected data with QDA. Indeed, simulating data according to such a model (Figure **??**E), Lᴏʟ performs slightly better than chance, regardless of the number of dimensions we use to project, whereas QOQ (for Quadratic Optimal QDA) performs significantly better regardless of how many dimensions it keeps. This demonstrates a straightforward generalization of Lᴏʟ, available to us because of the simplicity and intuitiveness of Lᴏʟ.

Outliers Outliers persist in many real data sets. Finding outliers, especially in high-dimensional data, is both tedious and difficult. Therefore, it is often advantageous to have estimators that are robust to certain kinds of outliers [**? ? ?**]. Pᴄᴀ and eigenvector computation are particularly sensitive to outliers [**?**]. Because Lᴏʟ is so simple and modular, we can replace typical eigenvector computation with a robust variant thereof, such as the geometric median subspace embedding [**?**]. Figure **??**F shows an example where we generated $n/2$ training samples according to the simple Lᴅᴀ model, but then added another $n/2$ training samples from a noise model. Lʀʟ (Linear Robust Low-Rank), performs better than Lᴏʟ regardless of the number of dimensions we keep. This simulation setting further demonstrates the flexibility of the Lᴏʟ framework to be extensible to other, more complicated scenarios.

XOR XOR is perhaps the simplest nonlinear problem, the problem that led to the demise of the perceptron, prior to its resurgence after the development of multi-layer perceptrons [**?**]. Thus, in our opinion, it is warranted to check whether any new classification method can perform well in this scenario. The classical (two-dimensional) XOR problem is quite simple: the output of a classifier is zero if both inputs are the same (00 or 11), and the output is one if the inputs differ (01 or 10). Figure **??**G shows a high dimensional and stochastic variant of XOR (see Methods for details). This simulation was designed such that standard classifiers, such as support vector machines and random forests, achieve chance levels (not shown). Lᴏʟ, performs moderately better than chance, and QOQ performs significantly better than chance, regardless of the chosen dimensionality. This demonstrates that our classifiers developed herein, though quite simple and intuition, can perform well even in settings where the data are badly modeled by our underlying assumptions. This mirrors previous findings where the so-called "idiots's Bayes" classifier outperforms more sophisticated classifiers [**?**]. In fact, we think of our work as finding intermediate points between idiot's Bayes (or naïve Bayes) and Fʟᴅ, by enabling degrees of regularization by changing the dimensionality used.

Computational Efficiency

In many applications, the main quantifiable consideration in whether to use a particular method, other than accuracy, is computational efficiency. Because implementing Lᴏʟ requires only highly optimized linear algebraic routines—including computing moments and singular value decomposition—rather than the costly iterative programming techniques currently required for sparse or dictionary learning type problems. Moreover, because Lᴏʟ only requires computing means and covariances, we can leverage recent advances in parallel matrix operations [**?**] on fast solid-state drives (SSDs). We implement LOL with the R interface of FlashX [**? ? ?**]. Given the FlashX programming framework, we can run Lᴏʟ on essentially arbitrarily large data. For these examples, we simulated data satisfying the Lᴅᴀ model; specifically, we used a spherically symmetric covariance matrix, with means just $\pm$ the one vector (see Methods for simulation details).

Figure **??**A demonstrates both the in memory and semi-external memory [**?**] computing models of our implementation. For the in memory implementation (light green line), we see that the run time increases linearly with the number of dimensions, requiring about 11 minutes to run Lᴏʟ on a $p = 32{,}000{,}000$ dimensional problem with $n = 2000$ samples. This linear increase is the optimal scale up according to Ahmdel's Law [**?**]. This example already requires 477 gigabyte (GB) to store the data, but <1 GB to perform the computations. For larger data, we developed a semi-external memory implementation, which stores the data matrix on solid state drives, and the low-rank estimates in RAM [**?**]. This strategy allows us to run Lᴏʟ as long as the data can fit on disk, and the low-rank approximation can fit in RAM. The semi-external memory implementation (dark green line) achieves the same performance as the in memory implementa-

257 tion whenever the data are small enough for in memory, and continues scaling optimally (linearly) as the
258 dimensionality further increases to $p = 128,000,000$ dimensions, a multi-terabyte scenario.

259 Another option for making Lᴏʟ scale up better is to utilize randomized algorithms. In particular, random
260 projections—for which the data are multiplied by a lower-dimensional random matrix—have been shown
261 to provide excellent approximation eigenvectors [**?**]. Moreover, very sparse random projections, in which
262 the elements of the matrix are mostly zero, with ± 1 randomly distributed, have been shown to be effective,
263 and have significant computational benefits [**?**]. We therefore further modified FlashX to incorporate
264 random projections and very sparse random projections, and let Lᴀʟ denote Linear Approximate Low-rank.
265 Figure **??**A shows an order of magnitude improvement in both the in-memory and semi-external memory
266 implementations of Lᴀʟ.

267 Figure **??**B shows the error for Lᴏʟ and Fʟᴅᴏ Pᴄᴀ in this scenario, simply to demonstrate that even in this
268 extremely high dimensional setting, Lᴏʟ still outperforms FʟᴅᴏPᴄᴀ. As expected, Lᴀʟ performs as well as
269 Lᴏʟfor the low dimensional settings.

Computational Theory

271 We reify the above computational experiments with theoretical statements. Computing the mean for each
272 class requires $\mathcal{O}(n_j p)$ floating point operations (flops), where n_j is the number of samples per class. Sub-
273 tracting the means requires $\mathcal{O}((J-1)p)$ flops, where J is the total number of classes. Computing the first d
274 singular triples (left and right singular vectors, and singular value) requires $\mathcal{O}(npd)$ flops. The sparse random
275 projection however only requires $\mathcal{O}(npd/c)$, where $c1$ is the sparsity of the sparse random projection matrix.
276 Since, $d \ll n, p$, this can substantially reduce computational complexity. Regardless, Lᴏʟ computation is
277 clearly bottlenecked by the Pᴄᴀ computation or random projection for single threaded operations.

278 Our parallel in memory implementation scales-up optimally (linearly) with the number of threads, requiring
279 $\mathcal{O}(npd/T)$ flops for Lᴏʟ, and $\mathcal{O}(npd/cT)$ flops for Lᴀʟ, where T is the number of threads. Our parallel
280 semi-external memory implementation scales-out optimally (linearly) as well. This is in agreement with the
281 linear fit of the lines in Figure **??**A.

Benchmark Real Data Applications

283 Although Lᴏʟ both statistically and computationally outperforms its natural competitors both in theory and
284 in a variety of simulation settings, real data often tell a different story. We have therefore selected four com-
285 monly used high-dimensional datasets to compare Lᴏʟ to several state-of-the-art algorithms (see Methods
286 for details). For each dataset, we compare Lᴏʟ to (i) support vector machines (SVM), (ii) Rᴏᴀᴅ, (iii) lasso,
287 (iv) and random forest (RF). Because in practice all these approaches have "hyperparameters" to tune,
288 we consider several possible values for SVM, lasso, and Lᴏʟ (but not RF, as its runtime was too high).
289 Figure **??** shows the results for all four datasets. For each, we use MATLAB implementations, partially
290 because Rᴏᴀᴅ provides MATLAB code, and also because of MATLAB's ubiquity in certain communities.
291 Because we also have R and FlashX implementations of Lᴏʟ, comparisons using other languages would
292 be straightforward.

293 Qualitatively, the results are similar across datasets: Lᴏʟ achieves high accuracy and computational ef-
294 ficiency as compared to the other methodologies. Considering Figure **??**A and B, two popular sparse
295 settings, we find that Lᴏʟ can find very low dimensional projections with very good accuracy. For the
296 prostate data, with a sufficiently non-sparse solution for Rᴏᴀᴅ, it slightly outperforms Lᴏʟ, but at substantial
297 computational cost; in particular, Rᴏᴀᴅ takes about 100 times longer to run on this dataset. Figure **??**C and
298 D are 10-class problems, so Rᴏᴀᴅ is no longer possible. Here, SVM can again slightly outperform Lᴏʟ, but
299 again, requiring 100 fold additional computational time. In all cases, the beloved random forest classifier
300 performs subpar. In all four scenarios, there is never an algorithm that achieves smaller error in less time,
301 as indicated by the dark gray box being empty in the lower panels of Figure **??**A-D.

9

Extensions to Other Supervised Learning Problems

The utility of incorporating the mean difference vector into supervised machine learning for big and wide data extends beyond merely classification. In particular, hypothesis testing can be considered as a special case of classification, with a particular loss function. Therefore we apply the same idea to a hypothesis testing scenario. The multivariate generalization of the t-test, called Hotelling's Test, suffers from the same problem as does the classification problem; namely, it requires inverting an estimate of the covariance matrix whose estimate would be low-rank and therefore singular, in the high-dimensional setting. To mitigate this issue in the hypothesis testing scenario, prior art applied similar tricks as they have done in the classification setting. One particularly nice and related example is that of Lopes et al. [**?**], who addresses this dilemma by using random projections to obtain a low-dimensional representation, following by applying Hotelling's Test in the lower dimensional subspace. Figure **??**A and B shows the power of their test alongside the power of the same approach, but using the LoL projection rather than random projections. The two different simulations include the simulated settings considered in their manuscript (see Methods for details). The results make it clear that the LoL test has higher power for essentially all scenarios. Moreover, it is not merely the replacing random projections with PCA (solid magenta line), nor simply incorporating the mean difference vector (dashed green line), but rather, it appears that LoL for testing uses both modifications to improve performance.

High-dimensional regression is another supervised learning method that can utilize the LoL idea. Linear regression, like classification and Hotelling's Test, requires inverting a matrix as well. By projecting the data only a lower dimensional subspace first, followed by linear regression on the low-dimensional data, we can mitigate the curse of high-dimensions. To choose the projection matrix, we partition the data into K partitions, based on the percentile of the target variable, we obtain a K class classification problem. Then, we can apply LoL to learn the embedding. Figure **??**C shows an example of this approach, contrasted with LASSO and partial least squares, in a sparse simulation setting (see Methods for details). LoL is able to find a better low-dimensional projection than LASSO, and performs significantly better than partial least squares, for essentially all choices of number of dimensions to embed into.

Discussion

We have introduced a very simple, yet new, methodology to improve performance on supervised learning problems with big and wide data. In particular, we have proposed a supervised manifold learning procedure that utilizes both the difference of the means, and the covariance matrices, and proved that it performs better than first applying PCA to the data under reasonable assumptions. This is in stark contrast to most previous approaches, which only utilize the covariance matrices (or kernel variants thereof), or try to solve a difficult optimization theoretic problem. In addition to demonstrating the statistical accuracy and computational efficiency of LoL on simulated and real classification problems, we also demonstrate how the same idea can also be used for other kinds of supervised learning problems, including regression and hypothesis testing. Theoretical guarantees suggest that this line of research is promising and can be extended to other more general settings and tasks.

Related Work One of the first publications to compose FLD with an unsupervised learning method was the celebrated Fisherfaces paper [**?**]. The authors showed via a sequence of numerical experiments the utility of embedding with PCA prior to classifying with FLD. We extend this work by adding a supervised component to the initial embedding. Moreover, we provide the geometric intuition for why and when this is advantageous, as well as show numerous examples demonstrating its superiority. We also prove that Fisherfaces can be implemented very efficiently (see Methods), while shedding light on why it is performing relatively well in general.

Next Steps The LoL idea, appending the mean difference vector to convert unsupervised manifold learning to supervised manifold learning, has many potential applications. We have presented the first few. Incorporating additional nonlinearities via kernel methods [**?**], ensemble methods such as random forests [**?**], and multiscale methods [**?**] are all of immediate interest. MATLAB, R, and FlashR code for the experiments performed in this manuscript is available from http://docs.neurodata.io/LOL/.
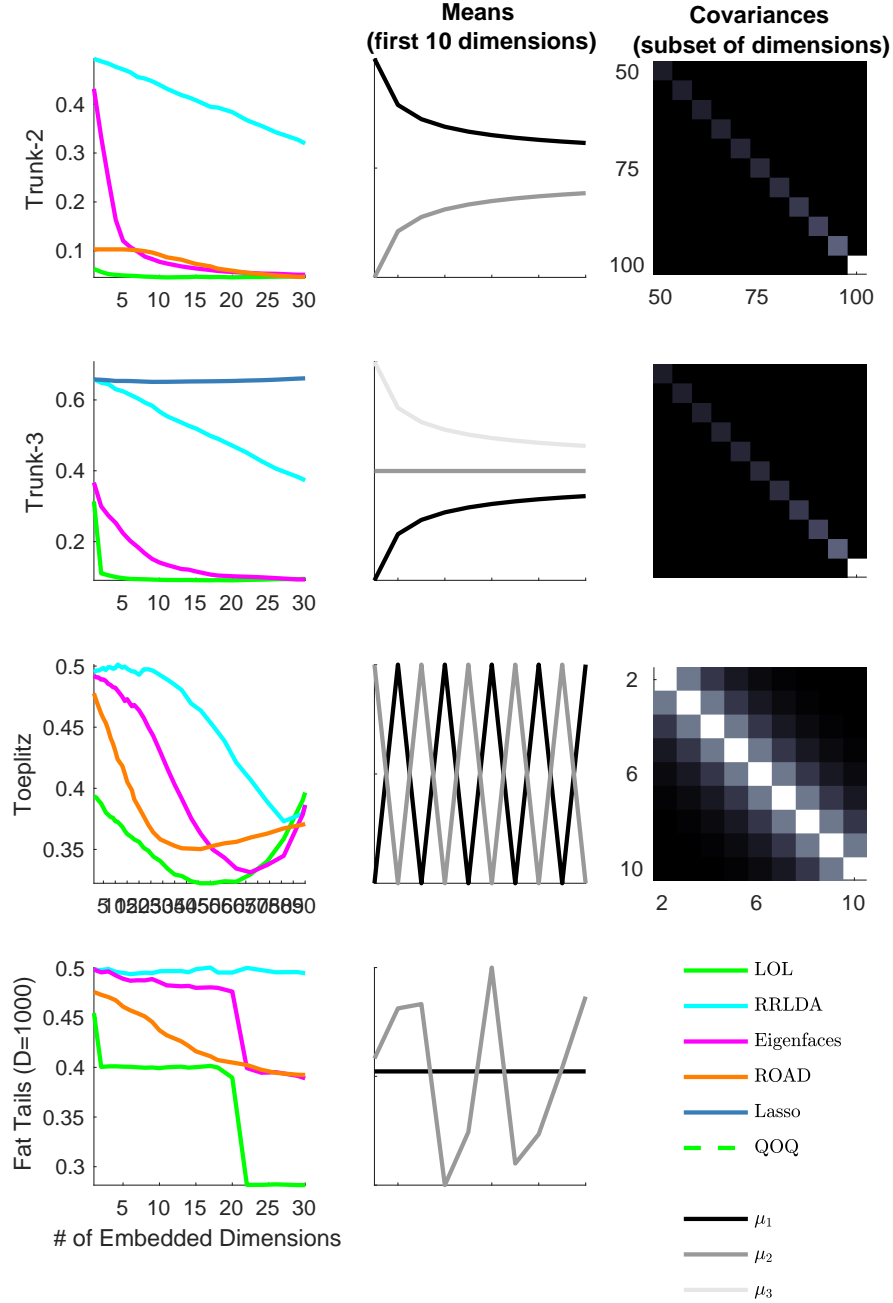
Figure 3: Seven simulations demonstrating `LOL` achieves superior finite sample performance over competitors both in settings for which we have asymptotic theoretical guarantees, and those for which we do not. For the first three, the top panels depict the means (top), the shared covariance matrix (middle). For the next three, the top panels depict a 2D scatter plot (left), mean and level set of one standard deviation of covariance matrix (right). For all seven simulations, the bottom panel shows misclassification rate as a function of the number of embedded dimensions, for several different classifiers. The simulations settings are as follows: **(A)** Rotated Trunk: same as Figure **??**C. **(B)** Toeplitz: another setting where mean difference is not well correlated with any eigenvector, and no ambient coordinate is particularly useful on its own. **(C)** 3 Class variant of the rotated Trunk example to demonstrate that `LOL` naturally adapts, and excels in, multi-class problems. **(D)** Fat Tails: a common phenomenon in real data that is more general than our theory supports. **(E)** QDA: `QOQ`, a variant of `LOL` when each class has a unique covariance, outperforms `LOL`, as expected, when the true discriminant boundary is a quadratic, rather than linear, function. **(F)** Outliers: adding high-dimensional outliers degrades performance of standard eigensolvers, but those can easily be replaced in `LOL` for a robust variants (called `LRL`). **(G)** XOR: a high-dimensional stochastic generalization of XOR, demonstrating that `QOQ` works even in scenarios that are quite distinct from the original motivating problems. In all 7 cases, `LOL`, or the appropriate generalization thereof, outperforms unsupervised or sparse methods. Moreover, the optimal embedding dimension is never the true discriminant dimension, but rather, a smaller number jointly determined by parameter settings and sample size.
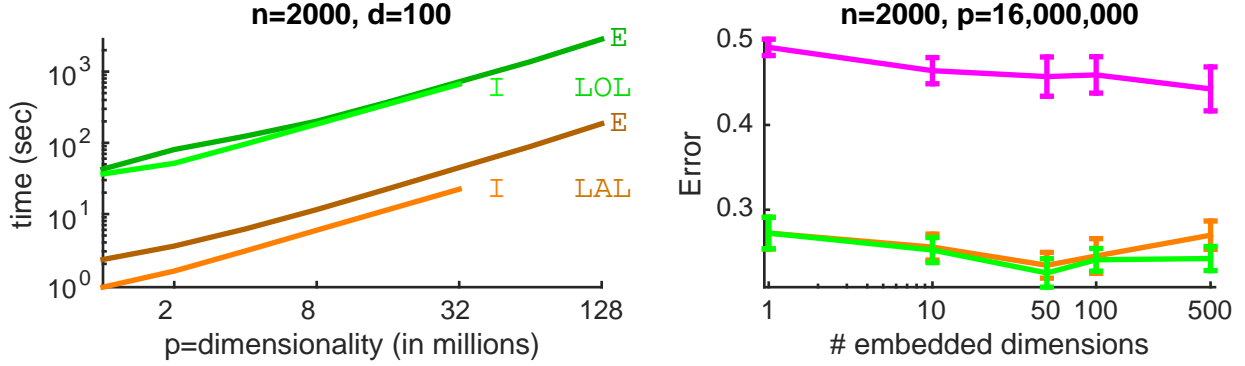
Figure 4: Computational efficiency of various low-dimensional projection methods. In all cases, $n = 2000$, and we used spherically symmetric simulation parameters (see Methods for details). We compare `PCA` with the projection step of `LOL` (light green for in memory, dark green for semi-external memory) and `LFL` (light orange for in-memory, dark orange for semi-external memory) for different observed dimensions (p). **(A)** `LOL` exhibits optimal (linear) scale up and scale out, requiring only 46 minutes to find the embedding on a 2TB dataset, and only 3 minutes using `LAL`(the sparse constant of sparse random projection $c = \dfrac{1}{\sqrt{p}}$). **(B)** Error for `LAL` is the same as `LOL` in this setting, and both are significantly better than `FLD∘PCA` for all choices of embedding dimension.
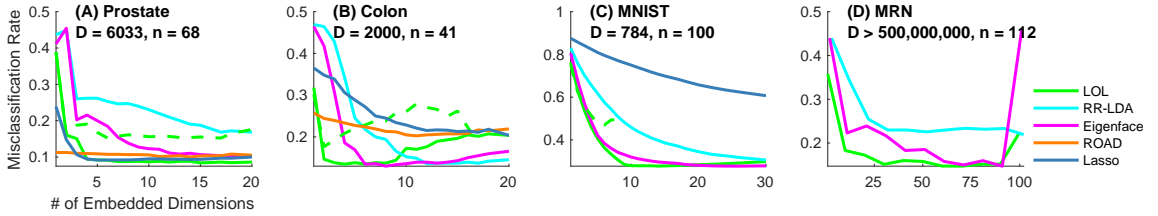
Figure 5: For four standard datasets, we benchmark `LOL` (green circles) versus standard classification methods, including support vector machines (blue up triangles), `ROAD` (cyan down triangles), `LASSO` (magenta pluses), and random forest (orange diamonds). Top panels show error rate as a function of $\log_2$ number of embedded dimensions (for `LOL`, `ROAD`, and `LASSO`) or cost (for SVM). Bottom panels show the minimum error rate achieved by each of the five algorithms versus time. The lower left dark gray (upper right light gray) rectangle is the area in which any algorithm is *better* (worse) than `LOL` in terms of both accuracy and efficiency. **(A)** Prostate: a standard sparse dataset. 1-dimensional `LOL` does very well, although keeping 2^5 ambient coordinates slightly improves performance, at a significant cost of compute time (two orders of magnitude), with minimal additional interpretability. **(B)** Colon: another standard sparse dataset. Here, 2-4 dimensions of `LOL` outperforms all other approaches considered regardless of how many dimensions they keep. **(C)** MNIST: 10 image categories, so `ROAD` is not possible. `LOL` does very well regardless of the number of dimensions kept. SVN marginally improves on `LOL` accuracy, at a significant cost in computation (two orders of magnitude). **(D)** CIFAR-10: a higher dimensional and newer 10 category image classification problem. Results are qualitatively similar to C. Note that, for all four of the problems, there is no algorithmperforming better and faster than `LOL`; rather, most algorithms typically perform worse and slower (though some are more accurate and much more computationally expensive).

(A) Diagonal
D=200, n=100

(B) Dense
D=200, n=100

(C) Sparse Toeplitz
D=1000, n=100

LOL
LAL
H o RP
H o PCA
Lasso
PLS
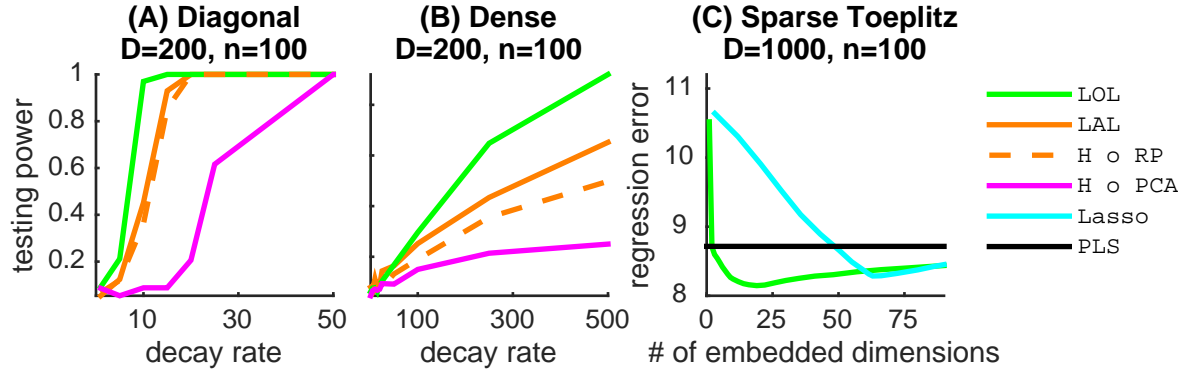
Figure 6: The intuition of including the mean difference vector is equally useful for other supervised manifold learning problems, including testing and regression. **(A)** and **(B)** show two different high-dimensional testing settings, as described in Methods. Power is plotted against the decay rate of the spectrum, which approximates the effective number of dimensions. LoL composed with Hotelling's test outperforms the random projections variants described in [**?**], as well as several other variants. **(C)** A sparse high-dimensional regression setting, as described in Methods, designed for sparse methods to perform well. Log_{10} mean squared error is plotted against the number of embedded dimensions. LoL composed with linear regression outperforms Lasso (cyan), the classic sparse regression method, as well as partial least squares (PLS; black). These three simulation settings therefore demonstrate the generality of this technique.

A Simulations

For most simulation settings, each class is Guassian: $f_{x|y} = \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$, $f_y = \mathcal{B}(\pi)$. We typically assume that both classes are equally like, $\pi = 0.5$, and the covariance matrices are the same, $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}$. Under such assumptions, we merely specify $\boldsymbol{\theta} = \{\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}\}$.

Stacked Cigars

- $\boldsymbol{\mu}_0 = \mathbf{0}$,
- $\boldsymbol{\mu}_1 = (a, b, a, \ldots, a)$,
- $\boldsymbol{\Sigma}$ is a diagonal matrix, with diagonal vector, $\boldsymbol{d} = (1, b, 1 \ldots, 1)$,

where $a = 0.15$ and $b = 4$.

Trunk

- $\boldsymbol{\mu}_0 = b/\sqrt{(1, 3, 5, \ldots, 2p)}$,
- $\boldsymbol{\mu}_1 = -\boldsymbol{\mu}_0$,
- $\boldsymbol{\Sigma}$ is a diagonal matrix, with diagonal vector, $\boldsymbol{d} = 100/\sqrt{(p, p-1, p-2, \ldots, 1)}$,

where $b = 4$.

Rotated Trunk

Same as Trunk, but the data are randomly rotated, that is, we sample $\boldsymbol{Q}$ uniformly from the set of p-dimensional rotation matrices, and then set:

- $\boldsymbol{\mu}_0 \leftarrow \boldsymbol{Q}\boldsymbol{\mu}_0$,
- $\boldsymbol{\mu}_1 \leftarrow \boldsymbol{Q}\boldsymbol{\mu}_1$,
- $\boldsymbol{\Sigma} \leftarrow \boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{Q}^\mathsf{T}$.

Toeplitz

- $\boldsymbol{\mu}_0 = b \times (1, -1, 1, -1, \ldots, 1)$,
- $\boldsymbol{\mu}_1 = -\boldsymbol{\mu}_0$,
- $\boldsymbol{\Sigma}$ is a Toeplitz matrix, where the top row is $\rho^{(0,1,2,\ldots,p-1)}$,

where b is a function of the Toeplitz matrix such that the noise stays constant as dimensionality increases, and $\rho = 0.5$.

3 Classes

Same as Trunk, but with a third mean equal to the zero vector, $\mu_2 = \mathbf{0}$.

Fat Tails

For this setting, each class is actually a mixture of two Gaussians with the same mean (the two classes have the same covariances):

- $\boldsymbol{\mu}_0 = \mathbf{0}$,
- $\boldsymbol{\mu}_1 = (0, \ldots, 0, 1, \ldots, 1)$, where the first $s = 10$ elements are zero,
- $\boldsymbol{\Sigma}_0$ is a matrix with one's on the diagonal, and 0.2 on the off diagonal,
- $\boldsymbol{\Sigma}_1 = 15 \times \boldsymbol{\Sigma}_0$,

and then we randomly rotated as in the rotated Trunk example.

³⁸⁷ **QDA**

³⁸⁸ A generalization of the Toeplitz setting, where the two classes have two different covariance matrices,
³⁸⁹ meaning that the optimal discriminant boundary is quadratic.

³⁹⁰   • $\boldsymbol{\mu}_0 = b \times (1, -1, 1, -1, \ldots, 1),$
³⁹¹   • $\boldsymbol{\mu}_1 = -\boldsymbol{Q} \times (\boldsymbol{\mu}_0 + 0.1),$
³⁹²   • $\boldsymbol{\Sigma}_0$ is the same Toeplitz matrix as described above, and
³⁹³   • $\boldsymbol{\Sigma}_1 = \boldsymbol{Q}\boldsymbol{\Sigma}_0\boldsymbol{Q}^{\mathsf{T}}.$

³⁹⁴ **Outliers**

³⁹⁵ In this dataset, we generate $n/2$ samples from an inlier model, and the remaining $n/2$ samples from an
³⁹⁶ outlier model. For the inlier model, we first generate a random $d \times p$ dimensional orthonormal matrix, V,
³⁹⁷ where $d = p/10$. Then, the first half of the inlier points are generated by $f_{x|0}$, the next half by $f_{x|1}$, and the
³⁹⁸ remaining points generated by $f_{x|\emptyset}$. For the outliers, we sampled their class randomly from a fair Bernoulli
³⁹⁹ distribution:

⁴⁰⁰   • $f_{x|0} = \mathcal{N}_d(0, \sigma^2) \times V^{\mathsf{T}},$
⁴⁰¹   • $f_{x|1} = \mathcal{N}_d(0, \sigma^2) \times V^{\mathsf{T}} + b,$
⁴⁰²   • $f_{x|\emptyset} = \mathcal{N}_p(\mathbf{0}, \sigma^2\boldsymbol{I}),$
⁴⁰³   • $f_\emptyset = \mathcal{B}(0.5).$

⁴⁰⁴ where we set $\sigma = 0.1$ and $b = 0.5$.

ALGORITHMS	p < n	2-Class LDA MODELS, p > n				GENERALIZED MODELS, p > n					OTHER TASKS, p > n		ALGORITHM PROPERTIES							ALGORITHMS	REF	NOTES
	LDA model	delta & cov aligned	& cov misaligned	delta & cov misaligned rotated	Fat Tails	>2 Class	QDA Model	Outlier Model	Non-linear	Testing	Regress	Fast	Simple	Theory	Open Source	Dense	2-step/ DL	Super-vised			
kNN	X	X	X	X	X	X		X	X		X		X	X	X	X		X	kNN	1	not considered
HCT	X	X	X									X	X	X			X	X	HCT	14	
LDA o Trace-Ratio	X	X	X	X								X	X	X		X	X	X	LDA o Trace-Ratio	4	
F2M / SDR	X			X		X							X	X	X	X		X	F2M / SDR	10	needs n>p
Deep Learning	X				X	X	X	X	X		X				X	X	X	X	Deep Learning	11	best when n>>p
Vowpal Wabbit	X				X	X		X			X	X			X	X	X	X	Vowpal Wabbit	5	best when n>>p
Ridge Regression	X											X	X	X	X	X		X	Ridge Regression	1	regression
Partial Least Squares											X	X						X	Partial Least Squares	1	regression
Naive Bayes	X	X	X			X						X	X	X	X	X		X	Naive Bayes	1,12	
HDDA	X	X	X	X		X								X	X			X	HDDA	6	
ROAD	X	X											X	X	X			X	ROAD	7,8,9	
LASSO	X	X				X					X	X	X	X	X		X	X	LASSO	3	
kSVM	X	X	X	X	X	X	X	X	X				X	X	X			X	kSVM	1	
Random Forest	X												X	X	X			X	Random Forest	1, 15	
LDA	X											X	X	X	X			X	LDA	1	
LDA o PCA	X	X				X					X	X	X	X	X	X	X	1/2	LDA o PCA	2	
LOL	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	LOL	*	
FIGURE		2A	2B, 3A	2C, 3B	3C	3D	3E	3F	3G	5A, 5D	5C, 5D	6									

Figure 7: Table of algorithms and their properties for high-dimensional data. Gray elements indicate that results are demonstrated in the Figure labeled in the bottom row. 'X' denotes relatively good performance for a given setting, or has the particular property.

B   Theoretical Background

II.A   The Classification Problem

Let $(\boldsymbol{X}, Y)$ be a pair of random variables, jointly sampled from $F := F_{\boldsymbol{X},Y} = F_{\boldsymbol{X}|Y}F_Y$. Let $\boldsymbol{X}$ be a multivariate vector-valued random variable, such that its realizations live in p dimensional Euclidean space, $\boldsymbol{x} \in \mathbb{R}^p$. Let Y be a categorical random variable, whose realizations are discrete, $y \in \{0, 1, \ldots C\}$. The goal of a classification problem is to find a function $g(\boldsymbol{x})$ such that its output tends to be the true class label y:

$$g^*(\boldsymbol{x}) := \underset{g \in \mathcal{G}}{\operatorname{argmax}}\, \mathbb{P}[g(\boldsymbol{x}) = y].$$

When the joint distribution of the data is known, then the Bayes optimal solution is:

$$g^*(\boldsymbol{x}) := \underset{y}{\operatorname{argmax}}\, f_{y|\boldsymbol{x}} = \underset{y}{\operatorname{argmax}}\, f_{\boldsymbol{x}|y}f_y = \underset{y}{\operatorname{argmax}}\{\log f_{\boldsymbol{x}|y} + \log f_y\} \tag{1}$$

Denote expected misclassification rate of classifier g for a given joint distribution F,

$$L_g^F := \mathbb{E}[g(\boldsymbol{x}) \neq y] := \int \mathbb{P}[g(\boldsymbol{x}) \neq y]f_{\boldsymbol{x},y}d\boldsymbol{x}dy,$$

where $\mathbb{E}$ is the expectation, which in this case, is with respect to F_{XY}. For brevity, we often simply write L_g, and we define $L_* := L_{g^*}$.

II.B   Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is an approach to classification that uses a linear function of the first two moments of the distribution of the data. More specifically, let $\mu_j = \mathbb{E}[F_{X|Y=j}]$ denote the class conditional mean, and let $\boldsymbol{\Sigma} = \mathbb{E}[F_X^2]$ denote the joint covariance matrix, and $\pi_j = \mathbb{P}[Y = j]$. Using this notation, we can define the LDA classifier:

$$g_{\text{Lda}}(\boldsymbol{x}) := \underset{y}{\operatorname{argmin}}\, \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_0)^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_0) + \mathbb{I}\{Y = y\}\log \pi_y,$$

where $\mathbb{I}\{\cdot\}$ is one when its argument is true, and zero otherwise. Let L_{Lda}^F be the misclassification rate of the above classifier for distribution F. Assuming equal class prior and centered means, $\pi_0 = \pi_1$ and $(\boldsymbol{\mu}_0 + \boldsymbol{\mu}1)/2 = \boldsymbol{0}$, re-arranging a bit, we obtain

$$g_{\text{Lda}}(\boldsymbol{x}) := \underset{y}{\operatorname{argmin}}\, \boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_y.$$

In words, the LDA classifier chooses the class for whom the projection of an input vector $\boldsymbol{x}$, onto $\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_y$, is maximized. When there are only two classes, this further simplies to

$$g_{2-\text{Lda}}(\boldsymbol{x}) := \mathbb{I}\{\boldsymbol{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\delta} > 0\},$$

where $\boldsymbol{\delta} = \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1$. Note that the equal class prior and centered means assumptions merely changes the threshold constant from 0 to something else.

II.C   LDA Model

A statistical model is a family of distributions indexed by a parameter $\boldsymbol{\theta} \in \Theta$, $\mathcal{F}_{\boldsymbol{\theta}} = \{F_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}$. Consider the special case of the above where $F_{\boldsymbol{X}|Y=y}$ is a multivariate Gaussian distribution, $\mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma})$, where each class has its own mean, but all classes have the same covariance. We refer to this model as the LDA model. Let $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, and let $\Theta_{C-\text{Lda}} = (\triangle_C, \mathbb{R}^{p \times C}, \mathbb{R}_{\succ 0}^{p \times p})$, where $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_C)$, $\triangle_C$ is the C dimensional simplex, that is $\triangle_C = \{\boldsymbol{x} : x_i \geq 0 \forall i, \sum_i x_i = 1\}$, and $\mathbb{R}_{\succ 0}^{p \times p}$ is the set of positive definite $p \times p$ matrices. Denote $\mathcal{F}_{\text{Lda}} = \{F_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta_{\text{Lda}}\}$, dropping the superscript C for brevity where appropriate. The following lemma is well known:

17

420 **Lemma 1.** $L_{Lda}^F = L_*^F$ *for any* $F \in \mathcal{F}_{Lda}$.

421 *Proof.* Under the LDA model, the Bayes optimal classifier is available by plugging the explicit distributions
422 into Eq. (**??**). $\qquad\square$

C   Projection Based Classifiers

Let $A \in \mathbb{R}^{d \times p}$ be an orthonormal matrix, that is, a matrix that projects p dimensional data into a d dimen-
sional subspace, where AA^T is the $d \times d$ identity matrix, and $A^\mathsf{T}A$ skis symmetric $p \times p$ matrix with rank d.
The question that motivated this work is: what is the best projection matrix that we can estimate, to use to
"pre-process" the data prior to applying LDA. Projecting the data x onto a low-dimensional subspace, and
the classifying via LDA in that subspace is equivalent to redefining the parameters in the low-dimensional
subspace, $\Sigma_A = A\Sigma A^\mathsf{T} \in \mathbb{R}^{d \times d}$ and $\delta_A = A\delta \in \mathbb{R}^d$, and then using g_{Lda}. When $C = 2$, $\pi_0 = \pi_1$, and
$(\mu_0 + \mu_1)/2 = 0$, this amounts to:

$$g_A^d(x) := \mathbb{I}\{(Ax)^\mathsf{T}\Sigma_A^{-1}\delta_A > 0\}, \text{ where } A \in \mathbb{R}^{d \times p}. \tag{2}$$

424 Let $L_A^d := \int \mathbb{P}[g_A(x) = y]f_{x,y}dxdy$. Our goal therefore is to be able to choose A for a given parameter
425 setting $\theta = (\pi, \delta, \Sigma)$, such that L_A is as small as possible (note that L_A will never be smaller than L_*).

426 Formally, we seek to solve the following optimization problem:

$$\begin{aligned}
\underset{A}{\text{minimize}} \quad & \mathbb{E}[\mathbb{I}\{x^\mathsf{T}A^\mathsf{T}\Sigma_A^{-1}\delta_A > 0\} \neq y] \\
\text{subject to} \quad & A \in \mathbb{R}^{p \times d}, \quad AA^\mathsf{T} = I_{d \times d},
\end{aligned} \tag{3}$$

427 where $I_{u \times v}$ is the $u \times v$ identity matrix identity, that is, $I(i, j) = 1$ for all $i = j \leq \min(u, v)$, and zero otherwise.
428 Let $\mathcal{A}^d = \{A : A \in \mathbb{R}^{d \times p}, AA^\mathsf{T} = I_{d \times d}\}$, and let $\mathcal{A}_* \subset \mathcal{A}$ be the set of A that minimize Eq. (**??**), and let
429 $A_* \in \mathcal{A}_*$ (where we dropped the superscript d for brevity). Let $L_A^* = L_{A_*}$ be the misclassification rate for
430 any $A \in \mathcal{A}_*$, that is, L_A^* is the Bayes optimal misclassification rate for the classifier that composes A with
431 LDA.

432 In our opinion, Eq. (**??**) is the simplest supervised manifold learning problem there is: a two-class classifi-
433 cation problem, where the data are multivariate Gaussians with shared covariances, the manifold is linear,
434 and the classification is done via LDA. Nonetheless, solving Eq. (**??**) is difficult, because we do not know
435 how to evaluate the integral analytically, and we do not know any algorithms that are guaranteed to find the
436 global optimum in finite time. This has led to previous work using a surrogate function [**?**]. We proceed by
437 studying a few natural choices for A.

III.A   Bayes Optimal Projection

439 **Lemma 2.** $\delta^\mathsf{T}\Sigma^{-1} \in \mathcal{A}_*$

Proof. Let $B = (\Sigma^{-1}\delta)^\mathsf{T} = \delta^\mathsf{T}(\Sigma^{-1})^\mathsf{T} = \delta^\mathsf{T}\Sigma^{-1}$, so that $B^\mathsf{T} = \Sigma^{-1}\delta$, and plugging this in to Eq. (**??**), we
obtain

$$\begin{aligned}
g_B(x) &= \mathbb{I}\{xB^\mathsf{T}\Sigma_B^{-1}\delta_B > 0\} \\
&= \mathbb{I}\{x^\mathsf{T}\Sigma^{-1}\delta \times (\Sigma_B^{-1}\delta_B) > 0\} && \text{plugging in } B \\
&= \mathbb{I}\{x^\mathsf{T}\Sigma^{-1}\delta k > 0\} && \text{because } \Sigma_B^{-1}\delta_B > 0.
\end{aligned}$$

440 In other words, letting B be the Bayes optimal projection recovers the Bayes classifier, as it should. Or,
441 more formally, for any $F \in \mathcal{F}_{\mathrm{Lda}}$, $L_{\delta^\mathsf{T}\Sigma^{-1}} = L_*$ $\qquad\square$

₄₄₂ **III.B   Principle Components Analysis (PCA) Projection**

₄₄₃ Principle Components Analysis (PCA) finds the directions of maximal variance in a dataset. PCA is closely
₄₄₄ related to eigendecompositions and singular value decompositions (SVD). In particular, the top principle
₄₄₅ component of a matrix $X \in \mathbb{R}^{p \times n}$, whose columns are centered, is the eigenvector with the largest corre-
₄₄₆ sponding eigenvalue of the centered covariance matrix XX^{T}. SVD enables one to estimate this eigenvector
₄₄₇ without ever forming the outer product matrix, because SVD factorizes a matrix X into USV^{T}, where U
₄₄₈ and V are orthonormal $p \times n$ matrices, and S is a diagonal matrix, whose diagonal values are decreasing,
₄₄₉ $s_1 \geq s_2 \geq \cdots > s_n$. Defining $U = [u_1, u_2, \ldots, u_n]$, where each $u_i \in \mathbb{R}^p$, then u_i is the ith eigenvector, and s_i
₄₅₀ is the square root of the ith eigenvalue of XX^{T}. Let $A_d^{\text{Pca}} = [u_1, \ldots, u_d]$ be the truncated PCA orthonormal
₄₅₁ matrix.

The PCA matrix is perhaps the most obvious choice of a orthonormal matrix for several reasons. First,
truncated PCA minimizes the squared error loss between the original data matrix and all possible rank d
representations:

$$\underset{A \in \mathbb{R}^{d \times p} : AA^{\mathsf{T}} = I_{d \times d}}{\operatorname{argmin}} \left\| X - A^T A \right\|_F^2 .$$

₄₅₂ Second, the ubiquity of PCA has led to a large number of highly optimized numerical libraries for computing
₄₅₃ PCA (for example, LAPACK [**?**]).

₄₅₄ Moreover, let $U_d = [u_1, \ldots, u_d] \in \mathbb{R}^{p \times d}$, and note that $U_d^{\mathsf{T}} U_d = I_{d \times p}$ and $U_d^{\mathsf{T}} U_d = I_{p \times d}$. Similarly, let
₄₅₅ $USU^{\mathsf{T}} = \Sigma$, and $US^{-1}U^{\mathsf{T}} = \Sigma^{-1}$. Let S_d be the matrix whose diagonal entries are the eigenvalues, up to
₄₅₆ the dth one, that is $S_d(i, j) = s_i$ for $i = j \leq d$ and zero otherwise. Similarly, $\Sigma_d = US_dU^{\mathsf{T}} = U_dS_dU_d^{\mathsf{T}}$.

₄₅₇ Let $g_{\text{Pca}}^d := g_{A_{\text{Pca}}^d}$, and let $L_{\text{Pca}}^d := L_{A_{\text{Pca}}^d}$. And let $g_{\text{Lda}}^d := \mathbb{I}\{x\Sigma_d^{-1}\delta > 0\}$ be the regularized LDA classifier, that
₄₅₈ is, the LDA classifier, but sets the bottom $p - d$ eigenvalues to zero.

₄₅₉ **Lemma 3.** $L_{Pca}^d = L_{Lda}^d$.

Proof. Plugging U_d into Eq. (**??**) for A, and considering only the left side of the operand, we have

$$
\begin{aligned}
(Ax)^{\mathsf{T}}\Sigma_A^{-1}\delta_A &= x^{\mathsf{T}}A^{\mathsf{T}}A\Sigma^{-1}A^{\mathsf{T}}A\delta, \\
&= x^{\mathsf{T}}U_dU_d^{\mathsf{T}}\Sigma^{-1}U_dU_d^{\mathsf{T}}\delta, \\
&= x^{\mathsf{T}}U_dU_d^{\mathsf{T}}US^{-1}UU_dU_d^{\mathsf{T}}\delta, \\
&= x^{\mathsf{T}}U_dI_{d \times p}S^{-1}I_{p \times d}U_d^{\mathsf{T}}\delta, \\
&= x^{\mathsf{T}}U_dS_d^{-1}U_d^{\mathsf{T}}\delta, \\
&= x^{\mathsf{T}}\Sigma_d^{-1}\delta.
\end{aligned}
$$

₄₆₀ $\qquad\qquad\square$

₄₆₁ The implication of this lemma is that if one desires to implement Fisherfaces, rather than first learning the
₄₆₂ eigenvectors and then learning LDA, one can instead directly implement regularized LDA by setting the
₄₆₃ bottom $p - d$ eigenvalues to zero.

₄₆₄ **III.C   Linear Optimal Low-Rank (LOL) Projection**

₄₆₅ The basic idea of LOL is to use both δ and the top d eigenvectors. Most naïvely, we could simply concatenate
₄₆₆ the two, $A_{\text{Lol}}^d = [\delta, A_{\text{Pca}}^{d-1}]$. Recall that eigenvectors are orthonormal. To maintain orthonormality, we could
₄₆₇ easily apply Gram-Schmidt, $A_{\text{Lol}}^d = \text{ORTH}([\delta, A_{\text{Pca}}^{d-1}])$. Both in practice and in theory (as will be shown
₄₆₈ below), this orthogonalization step does not matter much.

₄₆₉ to ensure that they are balanced appropriately, we normalize δ

₄₇₀ each vector in δ to have norm unity. Formally, let $\widetilde{\delta}_j = \delta_j / \|\delta_j\|$, where δ_j is the jth difference of the
₄₇₁ mean vector (remember, the number of vectors is equal to $C - 1$, where C is the total number of classes),

472   and let $A_{\text{Lol}}^d = [\widetilde{\delta}, A_{\text{Pca}}^{d-(C-1)}]$. The eigenvectors are all normalized and orthogonal to one another; to
473   impose orthogonality between $\widetilde{\delta}$ and the eigenvectors, we could use any number of numerically optimized
474   algorithms. However, in practice, orthogonalizing does not matter very much, so we do not bother. We
475   formally demonstrate this below.

D   Theoretical Properties of LDA based Classifiers

IV.A $\texttt{LDA}$ is rotationally invariant

478   For certain classification tasks, the ambient coordinates have intrinsic value, for example, when simple
479   interpretability is desired. However, in many other contexts, interpretability is less important [**?**]. When
480   the exploitation task at hand is invariant to rotations, then we have no reason to restrict our search space
481   to be sparse in the ambient coordinates, rather, for example, we can consider sparsity in the eigenvector
482   basis. Fisherfaces is one example of a rotationally invariant classifier, under certain model assumptions.
483   Let W be a rotation matrix, that is $W \in \mathcal{W} = \{W : W^\mathsf{T} = W^{-1} \text{ and } \det(W) = 1\}$. Moreover, let
484   $W \circ F$ denote the distribution F after transformation by an operator W. For example, if $F = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ then
485   $W \circ F = \mathcal{N}(W\boldsymbol{\mu}, W\boldsymbol{\Sigma}W^\mathsf{T})$.

Definition 1. *A rotationally invariant classifier has the following property:*

$$L_g^F = L_g^{W \circ F}, \qquad F \in \mathcal{F}.$$

486   *In words, the Bayes risk of using classifier g on distribution F is unchanged if F is first rotated, for any*
487   *$F \in \mathcal{F}$.*

488   Now, we can state the main lemma of this subsection: $\texttt{LDA}$ is rotationally invariant.

489   **Lemma 4.** $L_{Lda}^F = L_{Lda}^{W \circ F}$*, for any $F \in \mathcal{F}$.*

490   *Proof.* $\texttt{LDA}$ simply becomes thresholding $x^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\delta}$. Thus, we can demonstrate rotational invariance by
491   demonstrating that $x^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\delta}$ is rotationally invariant.

$$
\begin{aligned}
(Wx)^\mathsf{T}(W\boldsymbol{\Sigma}W^\mathsf{T})^{-1}W\boldsymbol{\delta} &= x^\mathsf{T}W^\mathsf{T}(WUSU^\mathsf{T}W^\mathsf{T})^{-1}W\boldsymbol{\delta} && \text{by substituting } USU^\mathsf{T} \text{ for } \boldsymbol{\Sigma} \\
&= x^\mathsf{T}W^\mathsf{T}(\widetilde{U}S\widetilde{U}^\mathsf{T})^{-1}W\boldsymbol{\delta} && \text{by letting } \widetilde{U} = WU \\
&= x^\mathsf{T}W^\mathsf{T}(\widetilde{U}S^{-1}\widetilde{U}^\mathsf{T})W\boldsymbol{\delta} && \text{by the laws of matrix inverse} \\
&= x^\mathsf{T}W^\mathsf{T}WUS^{-1}U^\mathsf{T}W^\mathsf{T}W\boldsymbol{\delta} && \text{by un-substituting } WU = \widetilde{U} \\
&= x^\mathsf{T}US^{-1}U^\mathsf{T}\boldsymbol{\delta} && \text{because } W^\mathsf{T}W = I \\
&= x^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\delta} && \text{by un-substituting } US^{-1}U^\mathsf{T} = \boldsymbol{\Sigma}
\end{aligned}
$$

492   $\qquad\qquad\square$

493   One implication of this lemma is that we can reparameterize without loss of generality. Specifically, defining
494   $W := U^\mathsf{T}$ yields a change of variables: $\boldsymbol{\Sigma} \mapsto S$ and $\boldsymbol{\delta} \mapsto U^\mathsf{T}\boldsymbol{\delta} := \boldsymbol{\delta}''$, where S is a diagonal covariance
495   matrix. Moreover, let $d = (\sigma_1, \ldots, \sigma_D)^\mathsf{T}$ be the vector of eignevalues, then $S^{-1}\boldsymbol{\delta}' = d^{-1} \odot \widetilde{\delta}$, where
496   $\odot$ is the Hadamard (entrywise) product. The $\texttt{LDA}$ classifier may therefore be encoded by a unit vector,
497   $\widetilde{d} := \frac{1}{m}d^{-1} \odot \boldsymbol{\delta}'$, and its magnitude, $m := \left\| d^{-1} \odot \widetilde{\delta} \right\|$. This will be useful later.

IV.B   Rotation of Projection Based Linear Classifiers g_A

499   By a similar argument as above, one can easily show that:

20

$$\begin{aligned}
(\boldsymbol{AWx})^\mathsf{T}(\boldsymbol{AW\Sigma W}^\mathsf{T}\boldsymbol{A}^\mathsf{T})^{-1}\boldsymbol{AW\delta} &= \boldsymbol{x}^\mathsf{T}(\boldsymbol{W}^\mathsf{T}\boldsymbol{A}^\mathsf{T})(\boldsymbol{AW})\boldsymbol{\Sigma}^{-1}(\boldsymbol{W}^\mathsf{T}\boldsymbol{A}^\mathsf{T})(\boldsymbol{AW})\boldsymbol{\delta}\\
&= \boldsymbol{x}^\mathsf{T}\boldsymbol{Y}^\mathsf{T}\boldsymbol{Y}\boldsymbol{\Sigma}^{-1}\boldsymbol{Y}^\mathsf{T}\boldsymbol{Y}\boldsymbol{\delta}\\
&= \boldsymbol{x}^\mathsf{T}\boldsymbol{Z}\boldsymbol{\Sigma}^{-1}\boldsymbol{Z}^\mathsf{T}\boldsymbol{\delta}\\
&= \boldsymbol{x}^\mathsf{T}(\boldsymbol{Z}\boldsymbol{\Sigma}\boldsymbol{Z}^\mathsf{T})^{-1}\boldsymbol{\delta} = \boldsymbol{x}^\mathsf{T}\widetilde{\boldsymbol{\Sigma}}_d^{-1}\boldsymbol{\delta},
\end{aligned}$$

where $\boldsymbol{Y} = \boldsymbol{AW} \in \mathbb{R}^{d\times p}$ so that $\boldsymbol{Z} = \boldsymbol{Y}^\mathsf{T}\boldsymbol{Y}$ is a symmetric $p \times p$ matrix of rank d. In other words, rotating and then projecting is equivalent to a change of basis. The implications of the above is:

Lemma 5. *g_A is rotationally invariant if and only if span(A)=span($\boldsymbol{\Sigma}_d$). In other words,* PCA *is the only rotationally invariant projection.*

IV.C   Chernoff information

We now introduce the notion of the Chernoff information, which serves as our surrogate measure for the Bayes error of any classification procedure given the *projected* data – in the context of this paper the projection is via LOL or PCA. Our discussion of the Chernoff information is under the context of decision rules for hypothesis testing, nevertheless, as evidenced by the fact that the Maximum A Posterior decision rule – equivalently the Bayes classifier – achieves the Chernoff information rate, this distinction between hypothesis testing and classification is mainly for ease of exposition.

Let F_0 and F_1 be two absolutely continuous multivariate distribution in $\Omega \subset \mathbb{R}^d$ with density function f_0 and f_1, respectively. Suppose that $Y_1, Y_2, \ldots, Y_m$ are independent and identically distributed random variables, with Y_i distributed either F_0 or F_1. We are interested in testing the simple null hypothesis $\mathbb{H}_0\colon F = F_0$ against the simple alternative hypothesis $\mathbb{H}_1\colon F = F_1$. A test T is a sequence of mapping $T_m\colon \Omega^m \mapsto \{0,1\}$ such that given $Y_1 = y_1, Y_2 = y_2, \ldots, Y_m = y_m$, the test rejects $\mathbb{H}_0$ in favor of $\mathbb{H}_1$ if $T_m(y_1, y_2, \ldots, y_m) = 1$; similarly, the test rejects $\mathbb{H}_1$ in favor of $\mathbb{H}_0$ if $T_m(y_1, y_2, \ldots, y_m) = 0$. The Neyman-Pearson lemma states that, given $Y_1 = y_1, Y_2 = y_2, \ldots, Y_m = y_m$ and a threshold $\eta_m \in \mathbb{R}$, the likelihood ratio test which rejects $\mathbb{H}_0$ in favor of $\mathbb{H}_1$ whenever

$$\left(\sum_{i=1}^m \log f_0(y_i) - \sum_{i=1}^m \log f_1(y_i)\right) \le \eta_m$$

is the most powerful test at significance level $\alpha_m = \alpha(\eta_m)$, i.e., the likelihood ratio test minimizes the type-II error β_m subject to the contraint that the type-I error is at most α_m.

Assuming that $\pi \in (0,1)$ is a prior probability that $\mathbb{H}_0$ is true. Then, for a given $\alpha_m^* \in (0,1)$, let $\beta_m^* = \beta_m^*(\alpha_m^*)$ be the type-II error associated with the likelihood ratio test when the type-I error is at most α_m^*. The quantity $\inf_{\alpha_m^* \in (0,1)} \pi\alpha_m^* + (1-\pi)\beta_m^*$ is then the Bayes risk in deciding between $\mathbb{H}_0$ and $\mathbb{H}_1$ given the m independent random variables $Y_1, Y_2, \ldots, Y_m$. A classical result of Chernoff [**?**] states that the Bayes risk is intrinsically linked to a quantity known as the *Chernoff information*. More specifically, let $C(F_0, F_1)$ be the quantity

$$\begin{aligned}
C(F_0, F_1) &= -\log\left[\inf_{t\in(0,1)} \int_{\mathbb{R}^d} f_0^t(\boldsymbol{x})f_1^{1-t}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\right]\\
&= \sup_{t\in(0,1)}\left[-\log \int_{\mathbb{R}^d} f_0^t(\boldsymbol{x})f_1^{1-t}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\right]
\end{aligned} \tag{4}$$

Then we have

$$\lim_{m\to\infty} \frac{1}{m} \inf_{\alpha_m^* \in (0,1)} \log(\pi\alpha_m^* + (1-\pi)\beta_m^*) = -C(F_0, F_1). \tag{5}$$

Thus $C(F_0, F_1)$ is the *exponential* rate at which the Bayes error $\inf_{\alpha_m^* \in (0,1)} \pi\alpha_m^* + (1-\pi)\beta_m^*$ decreases as $m \to \infty$; we also note that the $C(F_0, F_1)$ is independent of π. We also define, for a given $t \in (0,1)$ the Chernoff divergence $C_t(F_0, F_1)$ between F_0 and F_1 by

$$C_t(F_0, F_1) = -\log \int_{\mathbb{R}^d} f_0^t(\boldsymbol{x})f_1^{1-t}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}.$$

The Chernoff divergence is an example of a f-divergence as defined in [**?**]. When $t = 1/2$, $C_t(F_0, F_1)$ is the Bhattacharyya distance between F_0 and F_1.

The result of Eq. (**??**) can be extended to $K + 1 \geq 2$ hypothesis, with the exponential rate being the minimum of the Chernoff information between any pair of hypothesis. More specifically, let $F_0, F_1, \ldots, F_K$ be distributions on $\mathbb{R}^d$ and let $Y_1, Y_2, \ldots, Y_m$ be independent and identically distributed random variables with distribution $F \in \{F_0, F_1, \ldots, F_K\}$. Our inference task is in determining the distribution of the Y_i among the $K + 1$ hypothesis $\mathbb{H}_0 \colon F = F_0, \ldots, \mathbb{H}_K \colon F = F_K$. Suppose also that hypothesis $\mathbb{H}_k$ has *a priori* probability π_k. For any decision rule δ, the risk of δ is $r(\delta) = \sum_k \pi_k \sum_{l \neq k} \alpha_{lk}(\delta)$ where $\alpha_{lk}(\delta)$ is the probability of accepting hypothesis $\mathbb{H}_l$ when hypothesis $\mathbb{H}_k$ is true. Then we have [**?**]

$$\inf_\delta \lim_{m \to \infty} \frac{r(\delta)}{m} = -\min_{k \neq l} C(F_k, F_l). \tag{6}$$

where the infimum is over all decision rule δ, i.e., for any δ, $r(\delta)$ decreases to 0 as $m \to \infty$ at a rate no faster than $\exp(-m \min_{k \neq l} C(F_k, F_l))$.

When the distributions F_0 and F_1 are multivariate normal, that is, $F_0 = \mathcal{N}(\mu_0, \Sigma_0)$ and $F_1 = \mathcal{N}(\mu_1, \Sigma_1)$; then, denoting by $\Sigma_t = t\Sigma_0 + (1 - t)\Sigma_1$, we have

$$C(F_0, F_1) = \sup_{t \in (0,1)} \left(\frac{t(1-t)}{2} (\mu_1 - \mu_2)^\top \Sigma_t^{-1} (\mu_1 - \mu_2) + \frac{1}{2} \log \frac{|\Sigma_t|}{|\Sigma_0|^t |\Sigma_1|^{1-t}} \right).$$

IV.D Projecting data and Chernoff information

We now discuss how the Chernoff information characterizes the effect a linear transformation A of the data has on classification accuracy. We start with the following simple result whose proof follows directly from Eq. (**??**).

Lemma 6. *Let $F_0 = \mathcal{N}(\mu_0, \Sigma)$ and $F_1 \sim \mathcal{N}(\mu_1, \Sigma)$ be two multivariate normals with equal covariance matrices. For any linear transformation A, let $F_0^{(A)}$ and $F_1^{(A)}$ denotes the distribution of AX when $X \sim F_0$ and $X \sim F_1$, respectively. We then have*

$$\begin{aligned} C(F_0^{(A)}, F_1^{(A)}) &= \frac{1}{8}(\mu_1 - \mu_0)^\top A^\top (A\Sigma A^\top)^{-1} A(\mu_1 - \mu_0) \\ &= \frac{1}{8}(\mu_1 - \mu_0)^\top \Sigma^{-1/2} \Sigma^{1/2} A^\top (A\Sigma A^\top)^{-1} A \Sigma^{1/2} \Sigma^{-1/2}(\mu_1 - \mu_0) \\ &= \frac{1}{8} \| P_{\Sigma^{1/2} A^\top} \Sigma^{-1/2}(\mu_1 - \mu_0) \|_F^2 \end{aligned} \tag{7}$$

where $P_Z = Z(Z^\top Z)^{-1} Z^\top$ denotes the matrix corresponding to the orthogonal projection onto the columns of Z.

Thus for a classification problem where $X|Y = 0$ and $X|Y = 1$ are distributed multivariate normals with mean μ_0 and μ_1 and the same covariance matrix Σ, Lemma **??** then states that for any two linear transformations A and B, the transformed data AX is to be preferred over the transformed data BX if

$$(\mu_1 - \mu_0)^\top A^\top (A\Sigma A^\top)^{-1} A(\mu_1 - \mu_0) > (\mu_1 - \mu_0)^\top B^\top (B\Sigma B^\top)^{-1} B(\mu_1 - \mu_0).$$

As an example, suppose Σ is diagonal with distinct eigenvalues where the diagonal entries of Σ are in non-increasing order. Denote by $\delta = \mu_1 - \mu_0$ and let $A = \delta^\top$ and $B = e_1^\top = (1, 0, 0, \ldots, 0)$ be the linear transformations for LOL and PCA of X into $\mathbb{R}$. We then have

$$C(F_0^{(A)}, F_1^{(A)}) = \frac{(\delta^\top \delta)^2}{\delta^\top \Sigma \delta} = \frac{(\sum_i \delta_i^2)^2}{\sum_i \delta_i^2 \lambda_i}; \quad C(F_0^{(B)}, F_1^{(B)}) = \frac{\delta_1^2}{\lambda_1}$$

where λ_1 is the largest eigenvalue of Σ. Suppose furthermore that $\delta_1 \leq \delta_2 \leq \cdots \leq \delta_p$ and $\lambda_1 > \lambda_2 > \cdots > \lambda_p$. Then $C(F_0^{(A)}, F_1^{(A)})$ can be lower-bounded as

$$C(F_0^{(A)}, F_1^{(A)}) = \frac{(\sum_i \delta_i^2)^2}{\sum_i \delta_i^2 \lambda_i} \geq \frac{(p\delta_1^2)^2}{p\delta_p^2 \lambda_1}$$

539   and hence $C(F_0^{(A)}, F_1^{(A)}) > C(F_0^{(B)}, F_1^{(B)})$ provided $p\delta_1^2 \geq \delta_p^2$.

When $A = \begin{bmatrix} \delta \mid e_1 \mid e_2 \cdots \mid e_{d-1} \end{bmatrix}^\top \in \mathbb{R}^{d \times p}$ and $B = \begin{bmatrix} e_1 \mid e_2 \mid \cdots \mid e_d \end{bmatrix}^\top \in \mathbb{R}^{d \times p}$ are the linear transformation for LOL and PCA of X into $\mathbb{R}^d$, we have

$$C(F_0^{(A)}, F_1^{(A)}) = \frac{(\sum_{i=d}^p \delta_i^2)^2}{\sum_{i=d}^p \delta_i^2 \lambda_i} + \sum_{i=1}^{d-1} \frac{\delta_i^2}{\lambda_i}; \quad C(F_0^{(B)}, F_1^{(B)}) = \sum_{i=1}^d \frac{\delta_i^2}{\lambda_i}.$$

540   This can be seen as follows. Let $\xi_{d-1} = \begin{bmatrix} e_1 \mid e_2 \mid \cdots \mid e_{d-1} \end{bmatrix} \in \mathbb{R}^{p \times (d-1)}$ and $\zeta_{d-1} = (\lambda_1 \delta_1, \lambda_2 \delta_2, \dots, \lambda_{d-1}\delta_{d-1})^\top \in$
541   $\mathbb{R}^{d-1}$. Then

$$A\Sigma A^\top = \begin{bmatrix} \delta \mid \xi_{d-1} \end{bmatrix}^\top \Sigma \begin{bmatrix} \delta \mid \xi_{d-1} \end{bmatrix} = \begin{bmatrix} \delta^\top \Sigma \delta & \delta^\top \Sigma \xi_{d-1} \\ \xi_{d-1}^\top \Sigma \delta & \xi_{d-1}^\top \Sigma \xi_{d-1} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^p \delta_i^2 \lambda_i & \zeta_{d-1}^\top \\ \zeta_{d-1} & \Sigma_{d-1} \end{bmatrix} \qquad (8)$$

542   where $\Sigma_{d-1} = \mathrm{diag}(\lambda_1, \lambda_2, \dots, \lambda_{d-1})$ is the submatrix of Σ corresponding to the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_{d-1}$.
543   Using a formula for the inverse of a partitioned matrix, we have

$$(A\Sigma A^\top)^{-1} = \begin{bmatrix} \sum_{i=1}^p \delta_i^2 \lambda_i & \zeta_{d-1}^\top \\ \zeta_{d-1} & \Sigma_{d-1} \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} \left(\sum_{i=1}^p \delta_i^2 \lambda_i - \zeta_{d-1}^\top \Sigma_{d-1}^{-1} \zeta_{d-1}\right)^{-1} & -\left(\sum_i \delta_i^2 \lambda_i - \zeta_{d-1}^\top \Sigma_{d-1}^{-1} \zeta_{d-1}\right)^{-1} \zeta_{d-1}^\top \Sigma_{d-1}^{-1} \\ -\Sigma_{d-1}^{-1} \zeta_{d-1} \left(\sum_i \delta_i^2 \lambda_i - \zeta_{d-1}^\top \Sigma_{d-1}^{-1} \zeta_{d-1}\right)^{-1} & \Sigma_{d-1}^{-1} + \Sigma_{d-1}^{-1} \zeta_{d-1} \left(\sum_i \delta_i^2 \lambda_i - \zeta_{d-1}^\top \Sigma_{d-1}^{-1} \zeta_{d-1}\right)^{-1} \zeta_{d-1}^\top \Sigma_{d-1}^{-1} \end{bmatrix}$$
$$(9)$$

544   Now, $\sum_{i=1}^p \delta_i^2 \lambda_i - \zeta_{d-1}^\top \Sigma_{d-1}^{-1} \zeta_{d-1} = \sum_{i=1}^p \delta_i^2 - \sum_{i=1}^{d-1} \delta_i^2 \lambda_i = \sum_{i=d}^p \delta_i^2 \lambda_i$. Therefore,

$$(A\Sigma A^\top)^{-1} = \begin{bmatrix} \left(\sum_{i=d}^p \delta_i^2 \lambda_i\right)^{-1} & -\frac{\zeta_{d-1}^\top \Sigma_{d-1}^{-1}}{\sum_{i=d}^p \delta_i^2 \lambda_i} \\ -\frac{\Sigma_{d-1}^{-1} \zeta_{d-1}}{\sum_{i=d}^p \delta_i^2 \lambda_i} & \Sigma_{d-1}^{-1} + \frac{\Sigma_{d-1}^{-1} \zeta_{d-1} \zeta_{d-1}^\top \Sigma_{d-1}^{-1}}{\sum_{i=d}^p \delta_i^2 \lambda_i} \end{bmatrix}. \qquad (10)$$

In addition, $A(\mu_1 - \mu_0) = (\delta^\top \delta, \delta_1, \delta_2, \dots, \delta_{d-1})^\top = (\delta^\top \delta, \zeta_{d-1}^\top \Sigma_{d-1}^{-1})^\top \in \mathbb{R}^d$. Hence

$$(\mu_1 - \mu_0)^\top A^\top (A\Sigma A^\top)^{-1} A(\mu_1 - \mu_0) = \begin{bmatrix} \delta^\top \delta \mid \zeta_{d-1}^\top \Sigma_{d-1}^{-1} \end{bmatrix} \begin{bmatrix} \left(\sum_{i=d}^p \delta_i^2 \lambda_i\right)^{-1} & -\frac{\zeta_{d-1}^\top \Sigma_{d-1}^{-1}}{\sum_{i=d}^p \delta_i^2 \lambda_i} \\ -\frac{\Sigma_{d-1}^{-1} \zeta_{d-1}}{\sum_{i=d}^p \delta_i^2 \lambda_i} & \Sigma_{d-1}^{-1} + \frac{\Sigma_{d-1}^{-1} \zeta_{d-1} \zeta_{d-1}^\top \Sigma_{d-1}^{-1}}{\sum_{i=d}^p \delta_i^2 \lambda_i} \end{bmatrix} \begin{bmatrix} \delta^\top \delta \\ \Sigma_{d-1}^{-1} \zeta_{d-1} \end{bmatrix}$$

$$= \frac{(\delta^\top \delta)^2}{\sum_{i=d}^p \delta_i^2 \lambda_i} - 2\delta^\top \delta \frac{\zeta_{d-1}^\top \Sigma_{d-1}^{-2} \zeta_{d-1}}{\sum_{i=d}^p \delta_i^2 \lambda_i} + \left(\zeta_{d-1}^\top \Sigma_{d-1}^{-3} \zeta_{d-1} + \frac{(\zeta_{d-1}^\top \Sigma_{d-1}^{-2} \zeta_{d-1})^2}{\sum_{i=d}^p \delta_i^2 \lambda_i}\right)$$

$$= \frac{(\delta^\top \delta - \zeta_{d-1}^\top \Sigma_{d-1}^{-2} \zeta_{d-1})^2}{\sum_{i=d}^p \delta_i^2 \lambda_i} + \zeta_{d-1}^\top \Sigma_{d-1}^{-3} \zeta_{d-1}$$

$$= \frac{(\sum_{i=1}^p \delta_i^2 - \sum_{i=1}^{d-1} \delta_i^2)^2}{\sum_{i=d}^p \delta_i^2 \lambda_i} + \sum_{i=1}^{d-1} \frac{\delta_i^2}{\lambda_i} = \frac{(\sum_{i=d}^p \delta_i^2)^2}{\sum_{i=d}^p \delta_i^2 \lambda_i} + \sum_{i=1}^{d-1} \frac{\delta_i^2}{\lambda_i}.$$

The derivation of $C(F_0^{(B)}, F_1^{(B)})$ is straightforward and will be omitted. We therefore have

$$C(F_0^{(A)}, F_1^{(A)}) - C(F_0^{(B)}, F_1^{(B)}) = \frac{(\sum_{i=d}^p \delta_i^2)^2}{\sum_{i=d}^p \delta_i^2 \lambda_i} - \frac{\delta_d^2}{\lambda_d}.$$

From our assumption that $\lambda_d \geq \lambda_{d+1} \geq \cdots \geq \lambda_p$, we have $\sum_{i=d}^{p} \delta_i^2 \lambda_i \leq \lambda_d \sum_{i=d}^{p} \delta_i^2$ and hence

$$\frac{\left(\sum_{i=d}^{p} \delta_i^2\right)^2}{\sum_{i=d}^{p} \delta_i^2 \lambda_i} \geq \frac{\left(\sum_{i=d}^{p} \delta_i^2\right)^2}{\lambda_d \sum_{i=d}^{p} \delta_i^2} = \frac{1}{\lambda_d} \sum_{i=d}^{p} \delta_i^2 \geq \frac{\delta_d^2}{\lambda_d}$$

545   and hence $C(F_0^{(A)}, F_1^{(A)}) \geq C(F_0^{(B)}, F_1^{(B)})$ always, and the inequality is strict provided that $\sum_{d+1}^{p} \delta_d^2 > 0$.

Finally we consider the case where Σ is an arbitrary covariance matrix. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ be the eigenvalues of Σ and let $u_1, u_2, \ldots, u_p$ be the corresponding eigenvectors. For $d \leq p$, let $U_{d-1} = \begin{bmatrix} u_1 \mid u_2 \mid \cdots \mid u_{d-1} \end{bmatrix} \in \mathbb{R}^{p \times (d-1)}$ be the matrix whose columns are the eigenvectors $u_1, u_2, \ldots, u_{d-1}$ of Σ. Then the LOL projection matrix into $\mathbb{R}^d$ is given by $A = \begin{bmatrix} \delta \mid U_{d-1} \end{bmatrix}^\top$. We first have

$$A\Sigma A^\top = \begin{bmatrix} \delta \mid U_{d-1} \end{bmatrix}^\top \Sigma \begin{bmatrix} \delta \mid U_{d-1} \end{bmatrix} = \begin{bmatrix} \delta^\top \Sigma & \delta^\top \Sigma U_{d-1} \\ U_{d-1}^\top \Sigma \delta & U_{d-1}^\top \Sigma U_{d-1} \end{bmatrix} = \begin{bmatrix} \delta^\top \Sigma & \delta^\top \Sigma U_{d-1} \\ U_{d-1}^\top \Sigma \delta & \Lambda_{d-1} \end{bmatrix}$$

where $\Lambda_{d-1} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{d-1})$ is the $(d-1) \times (d-1)$ diagonal matrix formed by the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{d-1}$. Therefore, letting $\gamma = \delta^\top \Sigma \delta - \delta^\top \Sigma U_{d-1} \Lambda_{d-1}^{-1} U_{d-1}^\top \Sigma \delta$, we have

$$(A\Sigma A^\top)^{-1} = \begin{bmatrix} \delta^\top \Sigma \delta & \delta^\top \Sigma U_{d-1} \\ U_{d-1}^\top \Sigma \delta & U_{d-1}^\top \Sigma U_{d-1} \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} \gamma^{-1} & -\delta^\top \Sigma U_{d-1} \Lambda_{d-1}^{-1} \gamma^{-1} \\ -\Lambda_{d-1}^{-1} U_{d-1}^\top \Sigma \delta \gamma^{-1} & \left(\Lambda_{d-1} - \frac{U_{d-1}^\top \Sigma \delta \delta^\top \Sigma U_{d-1}}{\delta^\top \Sigma \delta}\right)^{-1} \end{bmatrix}.$$

The Sherman-Morrison-Woodbury formula then implies

$$\left(\Lambda_{d-1} - \frac{U_{d-1}^\top \Sigma \delta \delta^\top \Sigma U_{d-1}}{\delta^\top \Sigma \delta}\right)^{-1} = \Lambda_{d-1}^{-1} + \frac{\Lambda_{d-1}^{-1} U_{d-1}^\top \Sigma \delta \delta^\top \Sigma U_{d-1} \Lambda_{d-1}^{-1}/(\delta^\top \Sigma \delta)}{1 - \delta^\top \Sigma U_{d-1} \Lambda_{d-1}^{-1} U_{d-1}^\top \Sigma \delta/(\delta^\top \Sigma \delta)}$$

$$= \Lambda_{d-1}^{-1} + \frac{\Lambda_{d-1}^{-1} U_{d-1}^\top \Sigma \delta \delta^\top \Sigma U_{d-1} \Lambda_{d-1}^{-1}}{\delta^\top \Sigma \delta - \delta^\top \Sigma U_{d-1} \Lambda_{d-1}^{-1} U_{d-1}^\top \Sigma \delta}$$

$$= \Lambda_{d-1}^{-1} + \gamma^{-1} \Lambda_{d-1}^{-1} U_{d-1}^\top \Sigma \delta \delta^\top \Sigma U_{d-1} \Lambda_{d-1}^{-1}$$

We note that $\Sigma U_{d-1} = U_{d-1} \Lambda_{d-1}$ and $U_{d-1}^\top \Sigma = \Lambda_{d-1} U_{d-1}^\top$ and hence

$$\gamma = \delta^\top \Sigma \delta - \delta^\top \Sigma U_{d-1} \Lambda_{d-1}^{-1} U_{d-1}^\top \Sigma \delta = \delta^\top \Sigma \delta - \delta^\top U_{d-1} \Lambda_{d-1} \Lambda_{d-1}^{-1} \Lambda_{d-1} U_{d-1}^\top \delta$$

$$= \delta^\top \Sigma \delta - \delta^\top U_{d-1} \Lambda_{d-1} U_{d-1}^\top \delta = \delta^\top (\Sigma - \Sigma_{d-1}) \delta$$

where $\Sigma_{d-1} = U_{d-1} \Lambda_{d-1} U_{d-1}^\top$ is the best rank $d-1$ approximation to Σ with respect to any unitaritly invariant norm. In addition,

$$\Lambda_{d-1}^{-1} U_{d-1}^\top \Sigma \delta \delta^\top \Sigma U_{d-1} \Lambda_{d-1}^{-1} = \Lambda_{d-1}^{-1} \Lambda_{d-1} U_{d-1}^\top \delta \delta^\top U_{d-1} \Lambda_{d-1} \Lambda_{d-1}^{-1} = U_{d-1}^\top \delta \delta^\top U_{d-1}.$$

We thus have

$$(A\Sigma A^\top)^{-1} = \begin{bmatrix} \gamma^{-1} & -\delta^\top \Sigma U_{d-1} \Lambda_{d-1}^{-1} \gamma^{-1} \\ -\Lambda_{d-1}^{-1} U_{d-1}^\top \Sigma \delta \gamma^{-1} & \left(\Lambda_{d-1} - \frac{U_{d-1}^\top \Sigma \delta \delta^\top \Sigma U_{d-1}}{\delta^\top \Sigma \delta}\right)^{-1} \end{bmatrix} = \begin{bmatrix} \gamma^{-1} & -\gamma^{-1} \delta^\top U_{d-1} \\ -\gamma^{-1} U_{d-1}^\top \delta & \Lambda_{d-1}^{-1} + \gamma^{-1} U_{d-1}^\top \delta \delta^\top U_{d-1} \end{bmatrix}.$$

Therefore,

$$\delta^\top A^\top (A\Sigma A^\top)^{-1} A\delta = \delta^\top \begin{bmatrix} \delta \mid U_{d-1} \end{bmatrix} \begin{bmatrix} \gamma^{-1} & -\gamma^{-1} \delta^\top U_{d-1} \\ -\gamma^{-1} U_{d-1}^\top \delta & \Lambda_{d-1}^{-1} + \gamma^{-1} U_{d-1}^\top \delta \delta^\top U_{d-1} \end{bmatrix} \begin{bmatrix} \delta \mid U_{d-1} \end{bmatrix}^\top \delta$$

$$= \begin{bmatrix} \delta^\top \delta \mid \delta^\top U_{d-1} \end{bmatrix} \begin{bmatrix} \gamma^{-1} & -\gamma^{-1} \delta^\top U_{d-1} \\ -\gamma^{-1} U_{d-1}^\top \delta & \Lambda_{d-1}^{-1} + \gamma^{-1} U_{d-1}^\top \delta \delta^\top U_{d-1} \end{bmatrix} \begin{bmatrix} \delta^\top \delta \\ U_{d-1}^\top \delta \end{bmatrix}$$

$$= \gamma^{-1} (\delta^\top \delta)^2 - 2\gamma^{-1} \delta^\top \delta \delta^\top U_{d-1} U_{d-1}^\top \delta + \delta^\top U_{d-1} (\Lambda_{d-1}^{-1} + \gamma^{-1} U_{d-1}^\top \delta \delta^\top U_{d-1}) U_{d-1}^\top \delta$$

$$= \gamma^{-1} (\delta^\top \delta - \delta^\top U_{d-1} U_{d-1}^\top \delta)^2 + \delta^\top U_{d-1} \Lambda_{d-1}^{-1} U_{d-1}^\top \delta$$

$$= \gamma^{-1} (\delta^\top (I - U_{d-1} U_{d-1}^\top) \delta)^2 + \delta^\top \Sigma_{d-1}^\dagger \delta$$

546   where $\Sigma_{d-1}^{\dagger}$ is the Moore-Penrose pseudo-inverse of Σ_{d-1}. The PCA projection matrix into $\mathbb{R}^d$ is given by
547   $B = U_d^{\top}$ and hence

$$\delta^{\top} B^{\top} (B\Sigma B^{\top})^{-1} B\delta = \delta^{\top} U_d \Lambda_d^{-1} U_d^{\top} \delta = \delta^{\top} \Sigma_d^{\dagger} \delta. \tag{11}$$

We thus have

$$
\begin{aligned}
C(F_0^{(A)}, F_1^{(A)}) - C(F_0^{(B)}, F_1^{(B)}) &= \gamma^{-1}(\delta^{\top}(I - U_{d-1}U_{d-1}^{\top})\delta)^2 - \delta^{\top}(\Sigma_d^{\dagger} - \Sigma_{d-1}^{\dagger})\delta \\
&= \frac{(\delta^{\top}(I - U_{d-1}U_{d-1}^{\top})\delta)^2}{\delta^{\top}(\Sigma - \Sigma_{d-1})\delta} - \delta^{\top}(\Sigma_d^{\dagger} - \Sigma_{d-1}^{\dagger})\delta \\
&\geq \frac{(\delta^{\top}(I - U_{d-1}U_{d-1}^{\top})\delta)^2}{\lambda_d \delta^{\top}(I - U_{d-1}U_{d-1}^{\top})\delta} - \frac{1}{\lambda_d}\delta^{\top} u_d u_d^{\top} \delta \\
&= \frac{1}{\lambda_d}\delta^{\top}(I - U_{d-1}U_{d-1})\delta - \frac{1}{\lambda_d}\delta^{\top}(U_d U_d^{\top} - U_{d-1}U_{d-1}^{\top})\delta \geq 0
\end{aligned}
$$

548   where we recall that u_d is the d-th column of U_d. Thus $C(F_0^{(A)}, F_1^{(A)}) \geq C(F_0^{(B)}, F_1^{(B)})$ always, and the
549   inequality is strict whenever $\delta^{\top}(I - U_d U_d^{\top})\delta > 0$.

550   Next we consider the case when $A = \left[\delta | U_{d-1}\right]^{\top}$ and $B = U_d$ where U_d is an arbitrary $p \times d$ matrix with
551   $U_d^{\top} U_d = I$, i.e., U_d has d orthonornal columns, and U_{d-1} is the first $d-1$ columns of U_d. A similar derivation
552   to the above yields

$$C(F_0^{(A)}, F_1^{(A)}) = \frac{(\delta^{\top}\Sigma^{-1/2}(I - V_{d-1}V_{d-1}^{\top})\Sigma^{1/2}\delta)^2}{\delta^{\top}\Sigma^{1/2}(I - V_{d-1}V_{d-1}^{\top})\Sigma^{1/2}\delta} + \delta^{\top}\Sigma^{-1/2}V_{d-1}V_{d-1}^{\top}\Sigma^{-1/2}\delta \tag{12}$$

$$C(F_0^{(B)}, F_1^{(B)}) = \delta^{\top}\Sigma^{-1/2}V_d V_d^{\top}\Sigma^{-1/2}\delta \tag{13}$$

where $V_d V_{d-1}^{\top} = \Sigma^{1/2}U_d(U_d^{\top}\Sigma U_d)^{-1}U_d^{\top}\Sigma^{1/2}$ is the orthogonal projection onto the column space of $\Sigma^{1/2}U_d$.
Hence $C(F_0^{(A)}, F_1^{(A)}) > C(F_0^{(B)}, F_1^{(B)})$ if and only if

$$\frac{(\delta^{\top}\Sigma^{-1/2}(I - V_{d-1}V_{d-1}^{\top})\Sigma^{1/2}\delta)^2}{\delta^{\top}\Sigma^{1/2}(I - V_{d-1}V_{d-1}^{\top})\Sigma^{1/2}\delta} > \delta^{\top}\Sigma^{-1/2}(V_d V_d^{\top} - V_{d-1}V_{d-1}^{\top})\Sigma^{-1/2}\delta.$$

553   Let $C(F_0^{(A)}, F_1^{(A)})$ and $C(F_0^{(B)}, F_1^{(B)})$ be the Chernoff informations when $A = [\delta \mid U_{d-1}]^{\top}$ and $B = [\delta \mid$
554   $U_d]^{\top}$ where U_{d-1} and U_d contain the eignevectors of the population covariance matrices Σ; similarly, let
555   $\hat{C}(F_0^{(A)}, F_1^{(A)})$ and $\hat{C}(F_0^{(B)}, F_1^{(B)})$ be the Chernoff informations when $A = [\delta \mid \hat{U}_{d-1}]$ and $B = [\delta \mid \hat{U}_d]$
556   where $\hat{U}_{d-1}$ and $\hat{U}_d$ contain the eigenvectors of the *sample* covariance matrices $\hat{\Sigma}$. Suppose furthermore
557   that $C(F_0^{(A)}, F_1^{(A)}) > C(F_0^{(B)}, F_1^{(B)})$. Then for sufficiently large n, with high probability, $\hat{C}(F_0^{(A)}, F_1^{(A)}) >$
558   $\hat{C}(F_0^{(B)}, F_1^{(B)})$.

559   Finally we consider the case when $B = \tilde{U}_d^{\top}$ where $\tilde{U}_d$ is the $p \times d$ matrix whose columns are the d largest
560   eigenvectors of the *pooled* covariance matrix $\tilde{\Sigma} = \mathbb{E}[(X - \frac{\mu_0 + \mu_1}{2})(X - \frac{\mu_0 + \mu_1}{2})^{\top}]$. Assume, without loss of
561   generality, that $\mu_1 = -\mu_0 = \mu$. We then have

$$\tilde{\Sigma} = \mathbb{E}[XX^{\top}] = \pi\Sigma + \pi\mu_0\mu_0^{\top} + (1-\pi)\Sigma + (1-\pi)\mu_1\mu_1^{\top} = \Sigma + \mu\mu^{\top} = \Sigma + \tfrac{1}{4}\delta\delta^{\top}.$$

Therefore

$$(B\Sigma B^{\top})^{-1} = \left(\tilde{U}_d^{\top}\Sigma\tilde{U}_d\right)^{-1} = \left(\tilde{U}_d^{\top}(\tilde{\Sigma} - \tfrac{1}{4}\delta\delta^{\top})\tilde{U}_d\right)^{-1} = \left(\tilde{S}_d - \tfrac{1}{4}\tilde{U}_d^{\top}\delta\delta^{\top}\tilde{U}_d\right)^{-1} = \tilde{S}_d^{-1} + \frac{\tilde{S}_d^{-1}\tilde{U}_d^{\top}\delta\delta^{\top}\tilde{U}_d\tilde{S}_d^{-1}}{4 - \delta^{\top}\tilde{U}_d\tilde{S}_d^{-1}\tilde{U}_d^{\top}\delta}$$

562   where $\tilde{S}_d$ is the diagonal matrix containing the d largest eigenvalues of $\tilde{\Sigma}$. Hence

$$
\begin{aligned}
C(F_0^{(B)}, F_1^{(B)}) = \delta^\top B^\top (B\Sigma B^\top)^{-1} B\delta &= \delta^\top \tilde{U}_d \Big(\tilde{S}_d^{-1} + \frac{\tilde{S}_d^{-1} \tilde{U}_d^\top \delta \delta^\top \tilde{U}_d \tilde{S}_d^{-1}}{4 - \delta^\top \tilde{U}_d \tilde{S}_d^{-1} \tilde{U}_d^\top \delta} \Big) \tilde{U}_d^\top \delta \\
&= \delta^\top \tilde{U}_d \tilde{S}_d^{-1} \tilde{U}_d^\top \delta + \frac{(\delta^\top \tilde{U}_d \tilde{S}_d^{-1} \tilde{U}_d^\top \delta)^2}{4 - \delta^\top \tilde{U}_d \tilde{S}_d^{-1} \tilde{U}_d^\top \delta} \\
&= \delta^\top \tilde{\Sigma}_d^\dagger \delta + \frac{(\delta^\top \tilde{\Sigma}_d^\dagger \delta)^2}{4 - \delta^\top \tilde{\Sigma}_d^\dagger \delta} = \frac{4\delta^\top \tilde{\Sigma}_d^\dagger \delta}{4 - \delta^\top \tilde{\Sigma}_d^\dagger \delta}.
\end{aligned}
\tag{14}
$$

563   where $\tilde{\Sigma}_d = \tilde{U}_d \tilde{S}_d \tilde{U}_d^\top$ is the best rank d approximation to $\tilde{\Sigma} = \Sigma + \frac{1}{4}\delta\delta^\top$.

We recall that the LOL projection $A = [\delta \mid U_{d-1}]^\top$ yields

$$
C(F_0^{(A)}, F_1^{(A)}) = \frac{(\delta^\top (I - U_{d-1} U_{d-1}^\top)\delta)^2}{\delta^\top (\Sigma - \Sigma_{d-1})\delta} + \delta^\top \Sigma_{d-1}^\dagger \delta.
$$

564 To illustrate the difference between the LOL projection and that based on the eigenvectors of the *pooled*
565 covariance matrix, consider the following simple example. Let $\Sigma = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$ be a diagonal matrix
566 with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$. Also let $\delta = (0, 0, \ldots, 0, s)$. Suppose furthermore that $\lambda_p + s^2/4 < \lambda_d$. Then we
567 have $\tilde{\Sigma}_d = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d, 0, 0, \ldots, 0)$. Thus $\tilde{\Sigma}_d^\dagger = \mathrm{diag}(1/\lambda_1, 1/\lambda_2, \ldots, 1/\lambda_d, 0, 0, \ldots, 0)$ and $\delta^\dagger \tilde{\Sigma}_d^\dagger \delta = 0$.
568 Therefore, $C(F_0^{(B)}, F_1^{(B)}) = 0$.

On the other hand, we have

$$
C(F_0^{(A)}, F_1^{(A)}) = \frac{(\delta^\top (I - U_{d-1} U_{d-1}^\top)\delta)^2}{\delta^\top (\Sigma - \Sigma_{d-1})\delta} + \delta^\top \Sigma_{d-1}^\dagger \delta = \frac{s^4}{s^2 \lambda_p} + 0 = s^2/\lambda_p.
$$

We can generalize the previous example as follows. Let Σ be a $p \times p$ covariance matrix of the form

$$
\Sigma = \begin{bmatrix} \Sigma_d & 0 \\ 0 & \Sigma_d^\perp \end{bmatrix}
$$

where Σ_d is a $d \times d$ matrix. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ be the eigenvalues of Σ and suppose furthermore that the
eigenvalues of Σ_d are $\lambda_1, \lambda_2, \ldots, \lambda_d$. Let $\gamma = \lambda_d - \lambda_{d+1}$. We now assume that δ are generated randomly as
follows. The entries of δ are i.i.d. random variable sampled according to the following distribution. Given an
index i, with probability ϵ, $\delta_i = 0$ and with probability $1 - \epsilon$, δ_i is distributed according to a normal distribution
with mean $\tau > 0$ and variance σ^2. Then with probability at least ϵ^d, the covariance matrix for $\tilde{\Sigma}$ is of the form

$$
\tilde{\Sigma} = \begin{bmatrix} \Sigma_d & 0 \\ 0 & \Sigma_d^\perp + \frac{1}{4}(\tilde{\delta}\tilde{\delta}^\top) \end{bmatrix}
$$

where $\tilde{\delta} \in \mathbb{R}^{p-d}$ is formed by excluding the first d elements of δ. Now, if $\lambda_{d+1} + \frac{1}{4}\|\tilde{\delta}\|^2 < \lambda_d$, then the d largest
eigenvalues of $\tilde{\Sigma}$ is still $\lambda_1, \lambda_2, \ldots, \lambda_d$, and thus the eigenvectors corresponding to the d largest eigenvalues
of $\tilde{\Sigma}$ is the same as that for the d largest eigenvalues of Σ. That is to say,

$$
\lambda_{d+1} + \tfrac{1}{4}\|\tilde{\delta}\|^2 < \lambda_d \implies \tilde{\Sigma}_d^\dagger = \Sigma_d^\dagger \implies \delta^\top \tilde{\Sigma}_d^\dagger \delta = 0 \implies C(F_0^{(B)}, F_1^{(B)}) = 0.
$$

We now compute the probability that $\lambda_{d+1} + \frac{1}{4}\|\tilde{\delta}\|^2 < \lambda_d$. Suppose for the moment that $\epsilon > 0$ is fixed and
do not varies with p. We then have

$$
\frac{\sum_{i=d+1}^p \delta_i^2 - (p-d)(1-\epsilon)\tau^2}{\sqrt{(p-d)(2(1-\epsilon)(2\tau^2\sigma^2 + \sigma^4) + \epsilon(1-\epsilon)(\tau^4 + 2\tau^2\sigma^2 + \sigma^4))}} \xrightarrow{\mathrm{d}} N(0,1).
$$

Thus, as $p \to \infty$, the probability that $\lambda_{d+1} + \frac{1}{4}\|\tilde{\delta}\|^2 < \lambda_d$ converges to that of

$$
\Phi\Big(\frac{4(\lambda_d - \lambda_{d+1}) - (p-d)(1-\epsilon)\tau^2}{\sqrt{(p-d)(2(1-\epsilon)(2\tau^2\sigma^2 + \sigma^4) + \epsilon(1-\epsilon)(\tau^4 + 2\tau^2\sigma^2 + \sigma^4))}} \Big).
$$

This probability can be made arbitrarily close to 1 provided that $\lambda_d - \lambda_{d+1} \geq Cp(1-\epsilon)\tau^2$ for all sufficiently large p and for some constant $C > 1/4$. Since the probability that $\delta_1 = \delta_2 = \cdots = \delta_d$ is at least ϵ^d, we thus conclude that for sufficiently large p, with probability at least ϵ^d,

$$C(F_0^{(B)}, F_1^{(B)}) = 0 < C(F_0^{(A)}, F_1^{(A)}).$$

In the case where $\epsilon = \epsilon(p) \to 1$ as $p \to \infty$ such that $p(1-\epsilon) \to \theta$ for some constant K, then the probability that $\lambda_{d+1} + \frac{1}{4}\|\tilde{\delta}\|^2 < \lambda_d$ converges to the probability that

$$\frac{1}{4} \sum_{i=1}^K \sigma^2 \chi_1^2(\tau) \geq \lambda_d - \lambda_{d+1}$$

where K is Poisson distributed with mean parameter θ and $\chi_i^2(\tau)$ is the non-central chi-square distribution with one degree of freedom and non-centrality parameter τ. Thus if $\lambda_d - \lambda_{d+1} \geq C\theta\tau^2 \log p$ for sufficiently large p and for some constant C, then this probability can also be made arbitrarily close to 1.

IV.E   Finite Sample Performance

We now consider the finite sample performance of LOL and PCA-based classifiers in the high-dimensional setting with small or moderate sample sizes, e.g., when p is comparable to n or when $p \gg n$. Once again we assume that $X|Y = i \sim \mathcal{N}(\mu_i, \Sigma)$ for $i = 0, 1$. Furthermore, we also assume that Σ belongs to the class $\Theta(p, r, k, \tau, \lambda)$ as defined below.

Definition Let $\lambda > 0$, $\tau \geq 1$ and $k \leq p$ be given. Denote by $\Theta(p, r, k, \tau, \lambda, \sigma^2)$ the collection of matrices Σ such that

$$\Sigma = V\Lambda V^\top + \sigma^2 I$$

where V is a $p \times r$ matrix with orthonormal columns and Λ is a $r \times r$ diagonal matrix whose diagonal entries $\lambda_1, \lambda_2, \ldots, \lambda_r$ satisfy $\lambda \geq \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \geq \lambda/\tau$. In addition, assume also that $|\mathrm{supp}(V)| \leq k$ where $\mathrm{supp}(V)$ denote the non-zero rows of V, i.e., $\mathrm{supp}(V)$ is the subset of $\{1, 2, \ldots, p\}$ such that $V_j \neq 0$ if and only if $j \in \mathrm{supp}(V)$.

We note that in general $r \leq k \ll p$ and $\lambda/\tau \gg \sigma^2$. We then have the following result.

Theorem [?] Suppose there exists constants M_0 and M_1 such that $M_1 \log p \geq \log n \geq M_0 \log \lambda$. Then there exists a constant $c_0 = c_0(M_0, M_1)$ depending on M_0 and M_1 such that for all n and p for which

$$\frac{\tau k}{n} \log \frac{ep}{k} \leq c_0,$$

there exists an estimate $\hat{V}$ of V such that

$$\sup_{\Sigma \in \Theta(p, r, k, \tau, \lambda, \sigma^2)} \mathbb{E}\|\hat{V}\hat{V}^\top - VV^\top\|^2 \leq \frac{Ck(\sigma\lambda + \sigma^2)}{n\lambda^2} \log \frac{ep}{k}$$

where C is a universal constant not depending on p, r, k, τ, λ and σ^2.

We can therefore show that provided that M_0 and M_1 is large enough, and n and p satisfies the condition in the preceding theorem, then provided that the Chernoff information of the population version of LOL is larger than the Chernoff information of the population version of PCA, the expected Chernoff information for the sample version of LOL is also larger than the expected Chernoff information of the sample version of PCA. We emphasize that it is necessary that the LOL and the PCA version both projected into the top $d \leq r$ dimension of the sample covariance matrices.

27

E   The R implementation of LOL

Figure **??** shows the R implementation of LOL for binary classification using FlashMatrix [**?**]. The implementation takes a $D \times I$ matrix, where each column is a training instance and each instance has D features, and outputs a $D \times k$ projection matrix.

```
LOL <- function(m, labels, k) {
        counts <- fm.table(labels)
        num.labels <- length(counts$val)
        num.features <- dim(m)[1]
        nv <- k - (num.labels - 1)
        gr.sum <- fm.groupby(m, 1, fm.as.factor(labels, 2), fm.bo.add)
        gr.mean <- fm.mapply.row(gr.sum, counts$Freq, fm.bo.div, FALSE)
        diff <- fm.get.cols(gr.mean, 1) - fm.get.cols(gr.mean, 2)
        svd <- fm.svd(m, nv=0, nu=nv)
        fm.cbind(diff, svd$u)
}
```

Figure 8: The R implementation of LOL.