Big Supervised Manifold Learning by Low-Ranl Optimal Linear Discriminant Analysis (LOL)

Joshua T. Vogelstein, Brett Mensh,
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 (SL)—the art and science of estimating statistical relationships using labeled training data—is a crucial tool in scientific discovery. SL has been enabled a wide variety of basic and applied findings, ranging from discovering biomarkers in omics data [**?**], to object recognition from images [**?**], and includes celebrated methods such as support vector machines, random forests, and deep networks [**?**]. A special case of SL is classification; a classifier predicts the 'class' of a novel observation via partitioning the space of observations into different classes. (for example, predicting male vs. female from MRI scans). In this big data age, the ambient (or observed) dimensionality of the observations is quickly ballooning, and with it, the ambient dimensionality of the discriminant boundary. While historical data may have consisted of only a few dimensions (e.g., height and weight), modern scientific datasets often consist of hundreds, thousands, or even millions of dimensions (e.g. genetics, neuroscience, omics). Regardless of the dimensionality, when scientists or analysts obtain new datasets, they must make a number of decisions, by either implicitly or explicitly weighting considerations across several levels of analysis, including both statistical and computational considerations.

As suggested above, the goal of such problems is to estimate a discriminant boundary that partitions the space as optimally and efficiently as possible. If we knew the "true data generating mechanism", then writing down the optimal discriminant boundary would be relatively trivial. However, in all real data science problems, at least some parts of the true data generating mechanism remain unknown, because the "Truth" is likely infinitely complex. Because we do not know the truth, we seek tools that will estimate boundaries as accurately and efficiencly as possible given the training data available to us. To do so, one must make a series of decisions. These decisions have a variety of ramifications; the art of supervised learning is making choices that balance those ramifications appropriately for the problem at hand. The choices can reasonably be organized into four levels of analysis: (i) model, (ii) algorithm, (iii) implementation, and (iv) platform, each have different implications, both quantitative and qualitative. The quantitative impact can be further subdivided into statistical and computational implications, with statistical implications largely determined by the top two levels, and computational implications governed by the bottom two. Below, we list in italics the *choices* one must make, and in bold the **design considerations** associated with each. These considerations directly determine the design goals of our machine learning system, as will be elaborated upon below.

First, one must choose a *discriminant model* (often called feasible region or action space in other contexts, and often this choice is implicit in the algorithm choice). This modeling assumptions determine geometric constraints on the shape of the estimated boundary. For example, a linear boundary is a potential discriminant model, and the particular feasiable optimal boundary for a given problem is then specified by (a potentially multidimensional) parameter: the slope and intercept. **Model bias** is the minimum distance between the optimal discriminant boundary in the discriminant model and true boundary (determined by the data generating mechanism). As mentioned above, in real data problems, we always expect some model bias unless the discriminant model is "non-parametric", meaning that it can account for any discriminant boundary.

Second, one must choose an *algorithm* to estimate the boundary, such as Fishers Linear Discriminant (`FLD`), linear Support Vector Machines (`SVM`), or logistic regression. Each such algorithm has different convergence properties as sample size increases. Importantly, for different data generating mechanisms, different algorithms may converge on different solutions. The additional distance between the expected estimated boundary and the optimal boundary is called the **estimator bias**, which we also want to minimize by choosing an algorithm that will select an estimator as close as possible to the truth. Different algorithms will also differ in their **estimator variance**, which is a function both of the size of the discriminant model, and the idiosyncrasies of the algorithm. Of note, in certain circumstances, the two bias terms and the variance together determine expected mean square error, so one also desires to minimize the variance of the estimated boundary.

In real data problems, there are two additional quantitative considerations impacted by the algorithm choice: **hyper-parameter searches** and **robustness**. As mentioned above, we desire to minimize both the bias and the variance; alas, decreasing one often comes at the cost increasing the other. For any given data generating mechanism and dataset, the optimal trade-off is unknown. Fortunately, many algorithms are equiped with a hyper-parameter, that enables the practictioner to "tune" the bias vs. variance to obtain

an “optimal” fit in terms of expected error on other data. The ease and efficiency with which those hyper-parameters can be tuned can substantially impact performance. The robustness of an algorithm quantifies its ability to estimate nearly optimal boundaries when there are outliers. This is another consideration that plays a crucial role in real data.

Third, one must choose an *implementation*, which can have dramatic computational implications. First, implementations should be **numerically stable**, over the range of the number of samples and dimensions under consideration. Second, one desires an implementation that minimizes computational complexity, both in terms of **space** and **time**.

Finally, one must choose a *platform* upon which the implementation operates, including both software dependencies and hardware choice. In the big data age, both **scale-up** and **scale-out** are crucial design considerations. Scale-up, generically speaking, refers to the ability of utilizing all available computational resources, rather than allowing FLOPS to sit idly. Scale-out refers to the ability to incorporate additional resources as required. Together, these terms determine the scalability and resource efficiency of a system, which can have dramatic consequences in real world big data problems.

All of the above quantitative considerations can be evaluated **theoretically**, but typically only under (overly)restrictive assumptions. Even when the assumptions are met, theory often merely provides bounds, without concrete numbers in any particular situations. **Simulated experiments** can therefore buttress the theoretical results. By knowing the truth, one can compare the performance of different approaches to provide insight into when the theories are working as expected, including in settings that extend beyond the theoretical constraints. Moreover, **real data performance**, provide further insight into different methodologies in real data settings.

The design or choice of a machine learning system for solving a given real world problem reflects each of the above design considerations. Regardless of the problem, we want a model that exhibits low bias, an algorithm with low estimator bias and variance, with easy hyper-parameter searches that is robust to outliers, an implementation that is numerically stable with small space and time requirements, deployed on a platform that scales both up and out. And we want all the properties demonstrated via theory, buttressed by simulations, and corroborated by real data experiments. Obviously, it is quite a challenge to optimize along all of the above dimensions for any particular kind of exploitation task. We therefore focus on the simplest supervised learning task we can think of: a high-dimensional two-class classification problem.

The first century of statistics focused largely on low-dimensional problems, establishing a beautiful framework to extend to this setting [], but relatively little in terms of algorithms, implementations, and systems that scale sufficiently.

To our knowledge, there does not yet exist a machine learning system that performs well along each of the above design considerations for even this problem.

, where each class is sampled from a Gaussian distribution and they only differ in their means. Such a situation is a big data problem when either the dimensionality p or sample size n is large; we focus on the setting where p is large, and in fact, when p is much larger than n, that is, $p > n$. In this setting, much of classical statistics that relies on asymptotic results with n increasing and p fixed, is not appropriate. Moreover, methods developed on the basis of the increasing sample size assumption suffer along many of the above considerations; in particular, they exhibit unacceptably large variance and numerical instability.

To address the above statistical considerations in such high-dimensional settings, a number of strategies have been developed recently in the literature. A very natural fix to this is to first “pre-process” the data via a truncated Principal Components Analysis (tPCA), and then apply FLD(an approach called Fisherfaces []. While first running tPCA greatly reduces the variance relative to directly running FLD, and therefore makes the algorithm numerically stable, it adds estimator bias because the dimension reduction performed by PCA does not consider the class labels, only the predictor variables, and the discriminant boundary is determined by the relationship between the class labels and predictor variables. Other “manifold learning techniques”, such as isomap [**?**], local linear embedding [**?**], Laplacian eigenmaps [**?**], Diffusion maps [**?**] and others are similarly “unsupervised”, and therefore, do not explicitly search for a discriminant boundary, and therefore may add significant bias. Adding supervision to these techniques so far has been difficult, methods proposed thus far embed the data in an unsupervised fashion first, again, potentially adding bias [**?**], and require substantial computations. Sparse methods (e.g., []) can use both the class

labels and predictor variables, but they come with substantial costs. In particular, if certain very restrictive assumptions of the data are not satisfied, such as the restrictive isometry property, sparse methods add both significant estimator bias and variance. Moreover, they require much greater computational time than Fisherfaces.

To address the above computational considerations, a number of strategies have been developed recently in the literature. Most such approaches have focused on the large n setting, including sub-sampling [], random sketching [**?**], and distributed learning [**? ? ? ?**]. To date, we are unaware of any machine learning library designed specifically to address "wide data" (where $p > n$), rather they all focus on "tall data" (where $n > p$). The computational considerations and implementation details for wide and tall data are sufficiently different to demand different implementations for optimized efficiency.

Thus, a gap remains in the literature for a statistical tool that, under relatively general assumptions, achieves both small bias and variance, can be made robust, and has easily tunable hyper-parameters. And complementarily, a software package that deploys the tool that is numerically stable, requires minimal space and time, is highly scalable for large p problems, and consumes minimal resources. Importantly, the method should perform well both on simulated and real data experiments. And ideally, the software is intuitive, free and open source, and easy to use.

Here, we describe a novel framework, called Low-Rank Optimal Linear Discriminant Analysis (LOL), that satisfies all of the above desiderata. We developed LOL by considering the above described simple scenario, and realizing that the optimal dimension reduction for such a setting would have to combine both the means and the covariances, because together those two parameters determine the optimal decision boundary. While other methods have previous had a similar insight (in particular the "First Two Moment" (F2M) methods []), F2M methods only operate when $n > p$, and we are interested in the setting where $p > n$, and possibly $p \gg n$. Therefore, we decided to combine the means and covariances differently, essentially by first projecting the data onto the means, and then effectively computing tPCA on the "residuals". Running FLD after projecting the data onto the subspace spanned jointly by the means and the top principal components comprises the LOL method. In addition to MATLAB and R, we also implemented LOL using FlashX, a recently developed big data analytics library. We were therefore able to take advantage of matrix parallelizations optimized for machines with solid state drives.

We demonstrate LOL outperforms—or at least performs no worse than—Fisherfaces along every statistical and computational consideration mentioned above. This includes proving the improvement over Fisherfaces along both statistical considerations (bias and variance) for the simple Gaussian setting mentioned above, as well as the computational considerations: stability, time, space, and scalability under more general settings. Moreover, numerical experiments quantitatively demonstrate the improvement of LOL over both Fisherfaces and sparse 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 mean and covariance 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

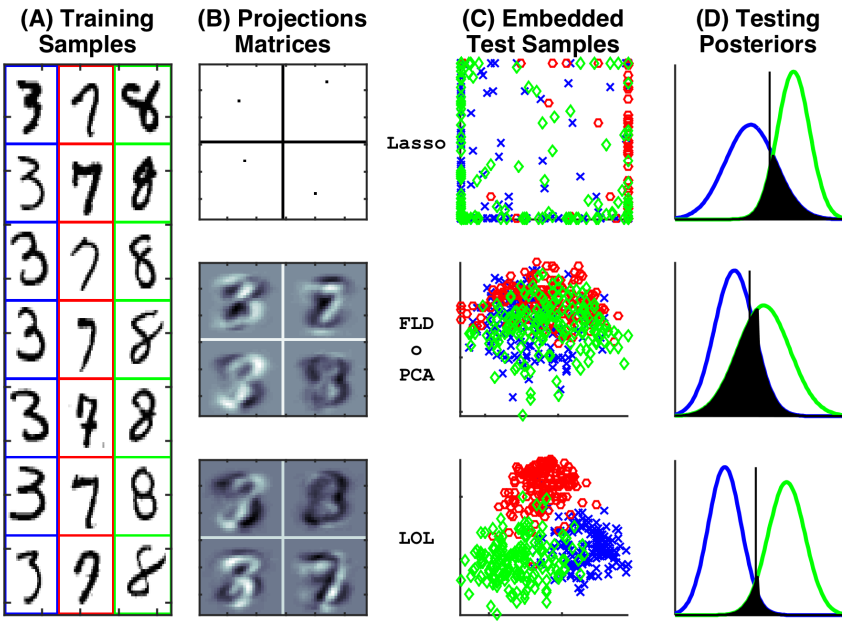

Figure 1: Illustrating three different classifiers—LASSO (top), FLDoPCA (middle), and LOL (bottom)—for embedding images of the digits 3, 7, and 8 (from MNIST), each of which is 28 $\times$ 28 = 784 dimensional. **(A)** Exemplars, boundary colors are only for visualization purposes. **(B)** The first four projection matrices learned by the three different approaches on 300 training samples. Note that LASSO is sparse and supervised, PCA is dense and unsupervised, and LOL is dense and supervised. **(C)** Embedding 500 test samples into the top 2 dimensions using each approach. Digits color coded as in (A). **(D)** The estimated posterior distribution of test samples after 5-dimensional projection learned via each method. We show only 3 vs. 8 for simplicity. The vertical line shows the classification threshold. 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.

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

Pseudocode of any method that embeds wide data as part of classification proceeds as schematized in Figure 1: (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 some classifier. We consider three different linear dimensionality reduction methods—LASSO, 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 many thousands of examples of images of the digits 0 through 9. Each such image is represented by a 28$\times$28 matrix, which means that the observed (or ambient) dimensionality of the data is $p = 784$. 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. We then apply all three approaches to this subsample of the MNIST dataset, learning a projection, and embedding $m = 500$ testing samples, and

[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.

classifying the resulting embedded data.

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. PCA, on the other hand, finds the linear combinations of training samples that maximize the variance. This unsupervised linear manifold learning method results in projection matrices that indeed look like linear combinations of the three different digits. The goal here, however, is separating classes, not maximizing variability. The resulting embeddings are not particularly well separated, suggesting the the directions of discriminability are not the same as the directions of maximum variance. LOL is our newly proposed supervised linear manifold learning method (see below for details). The projection matrices it learns look qualitatively much like those of PCA. This is not surprising, as both are 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 two approaches. This numerical experiment justifies the use of supervised linear manifold learning, we next investigate the performance of these methods in simpler simulated examples, to better illustrate when we can expect LOL to outperform other methods, and perhaps more importantly, when we expect this "vanilla" variant of LOL to fail.

Linear Gaussian Intuition

The above real data example suggests the geometric intuition for when LOL outperforms its sparse and unsupervised 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, the class priors are equal, and the joint distribution is centered, so that the only difference between the classes is their means (we call this the Linear Discriminant Analysis (LDA) model; see Methods for details).

To motivate LOL, and the following simulations, lets 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, $\boldsymbol{A}_* = \boldsymbol{\delta}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}$ [**?**] (see Methods for details). PCA, the dominant unsupervised manifold learning method, utilizes only the covariance structure of the data, and ignores the difference between the means. 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. Formally, we implement our idea by simply concatenating the difference of the means with the top d eigenvectors of the covariance. 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 2 shows three different examples of data sampled from the LDA model to geometrically illustration this intuition. In each, we sample $n = 100$ training samples in $p = 1000$ dimensional space, so $n \ll p$. Figure 2(A) shows an example we call "stacked cigars". In this example and the next, 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

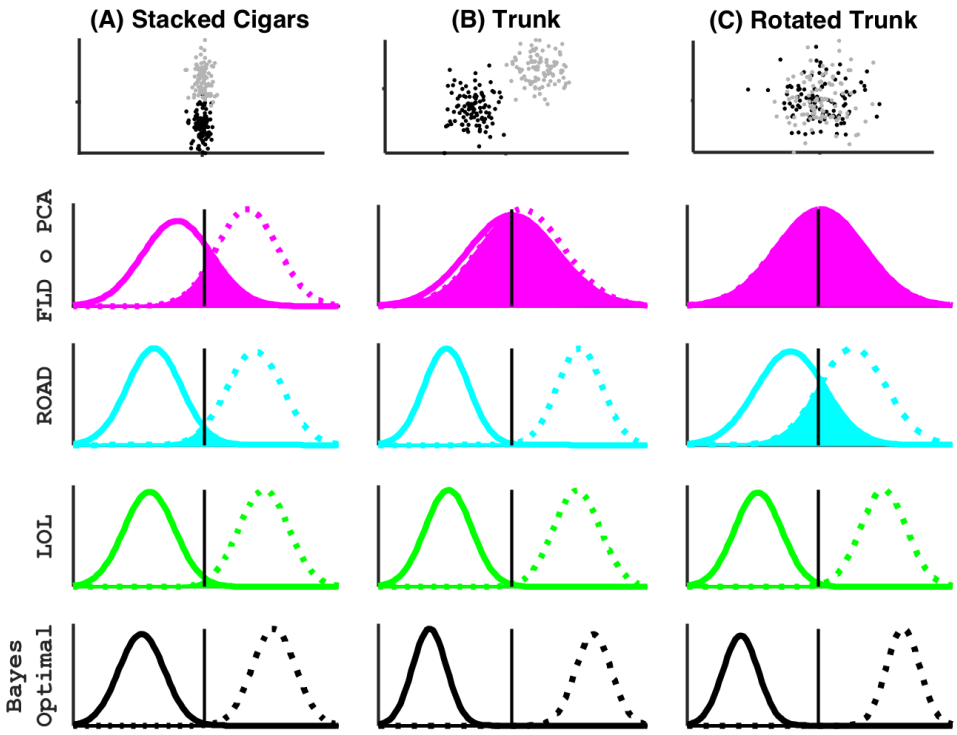

Figure 2: LOL achieves near optimal performance for a wide variety of distributions. Each point is sampled from a multivariate Gaussian; the three columns correspond to different simulation parameters (see Methods for details). In each of 3 simulations, we sample $n = 100$ points in $p = 1000$ dimensions. 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. In this setting, the results are similar. **(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. **(B)** The mean difference vector is orthogonal to the direction of maximal variance, making PCA fail, but sparse methods can still recover the correct dimensions. **(C)** Same as (B), but the data are rotated. **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. Note that LOL is closest to Bayes optimal in all three settings.

moderately well.[2] Because all dimensions are independent, this is a good scenario for sparse methods. Indeed, ROAD, a sparse classifier 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.

Figure 2(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 parameters, does extremely well.

Figure 2(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 two dimensions clearly 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. LOLis unperturbed by this rotation; in particular, it is able to "unrotate" the data, to find dimensions that optimally separate the two classes.

Theoretical Confirmation

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

Theorem 1. *Under the* LDA *model,* LOL *is better than* PCA*.*

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).

How many dimensions to keep?

In the above numerical and theoretical investigations, we fixed d, the number of dimensions to embed into. Much unsupervised manifold learning theory typically focuses on finding the "true" intrinsic dimensionality of the data. The analogous question for supervised manifold learning would be to find the true intrinsic dimensionality of the discriminant boundary. However, in real data problems, typically, their is no perfect low dimensional representation because of noise.

Thus, in all the following simulations, the true ambient dimensionality of the data is equal to the dimensionality of the optimal discriminant boundary (given infinite data). In other words, there does not exist a discriminant space that is lower dimensional than the ambient space, so we cannot find the "intrinsic dimension" of the data or the discriminant boundary. Rather, we face a trade-off: keeping more dimensions reduces bias, but increases variance. The optimal bias/variance trade-off depends on the distribution of the data, as well as the sample size [**?**].

Consider again the rotated trunk example as well as a "Toeplitz" example, as depicted in Figures 3(A) and (B), respectively. In both cases, the data are sampled from the LDA model, and in both cases, the optimal dimensionality under finite samples depends on the particular approach, but is never the true dimensionality. Moreover, LOL dominates the other approaches, regardless of the number of dimensions used. Figure 3(C) shows a sparse example with "fat tails" to mirror real data settings better. 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.

[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 [**? ?**]

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 find the class which has the maximum number of samples (breaking ties randomly), and subtract its mean from all other class means. Figure 3(D) shows a 3-class generalization of (A). While LOL uses the additional class naturally, many previously proposed high-dimensional FLD variants, such as ROAD, natively only work for 2-classes.

Generalizations of LOL

The simple geometric intuition which led to the development of LOL suggests that we can easily generalize LOL to be more appropriate for more complicated settings. We consider three additional scenarios:

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 LOL 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 LDA, we can then classify the projected data with QDA. Indeed, simulating data according to such a model (Figure 3(E)), LOL performs slightly better than chance, regardless of the number of dimensions we use to project, whereas QOQ (which denotes we estimate eigenvectors separately and then use QDA on the projected data) performs significantly better regardless of how many dimensions it keeps.

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 [**? ? ?**]. PCA and eigenvector computation are particularly sensitive to outliers [**?**]. Because LOL is so simple and modular, we can replace typical eigenvector computation with a robust variant thereof, such as the geometric median subspace embedding [**?**]. Figure 3(F) shows an example where we generated $n/2$ training samples according to the simple LDA model, but then added another $n/2$ training samples from a noise model. LRL (our robust variant of LOL that simply replaces the fragile eigenvector computation with a robust version), performs better than LOL regardless of the number of dimensions we keep.

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 3(G) shows a high dimensional and stochastic variant of XOR. This simulation was designed such that standard classifiers, such as support vector machines and random forests, achieve chance levels (not shown). LOL, 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 FLD, 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 LOL 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. To quantify the computational efficiency of LOL and its variants, Figure 4 shows the wall time it takes to run each method on the stacked cigars problem, varying the ambient dimensionality, embedded dimensionality,

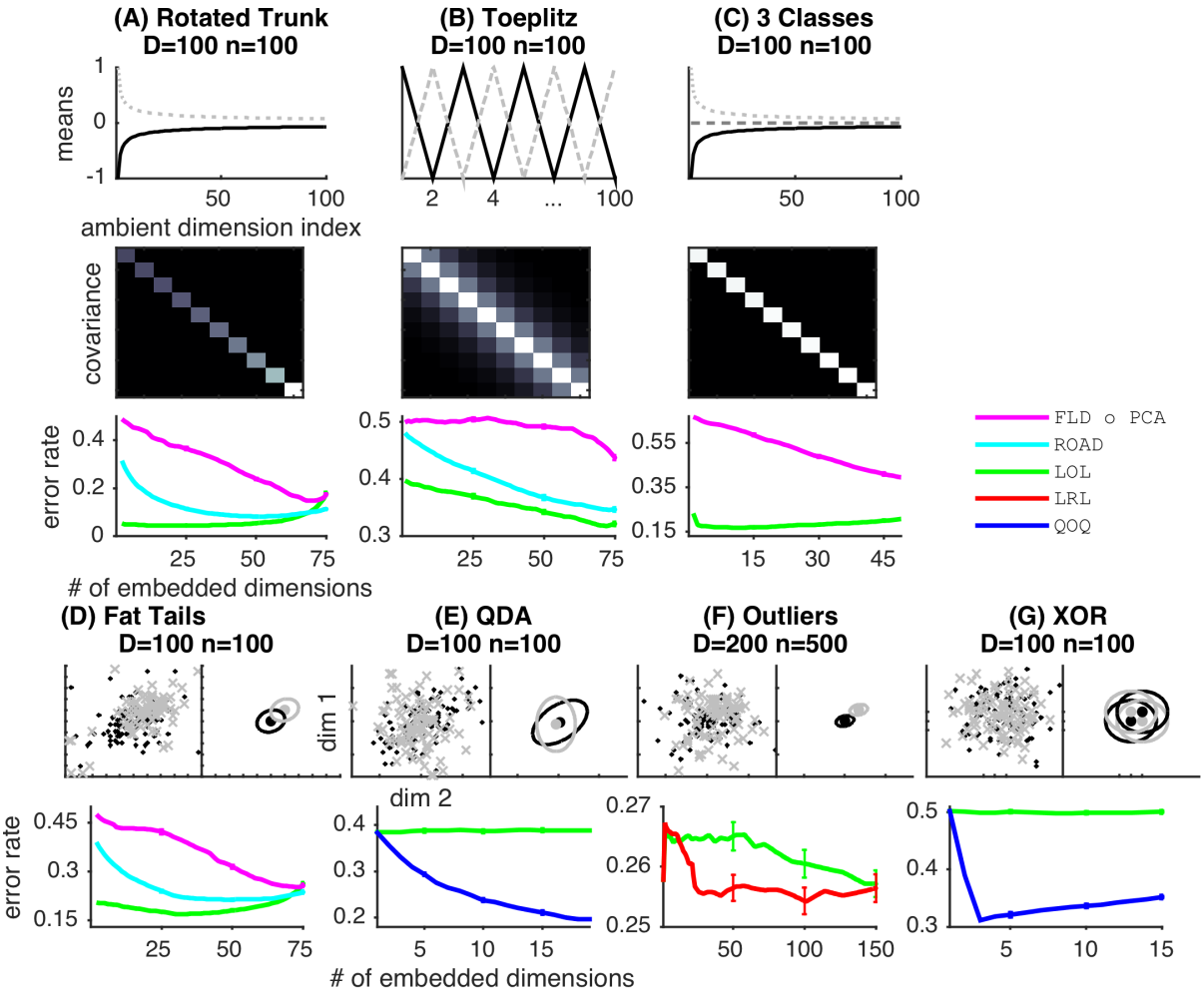

Figure 3: Seven simulations demonstrating that even when the true discriminant boundary is high-dimensional, LOL can find a low-dimensional projection that wins the bias-variance trade-off against competing methods. For the first four, the top panels depict the means (top), the shared covariance matrix (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 2(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 Classes: LOL naturally adapts to multiple classes. **(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. **(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). **(F)** XOR: a high-dimensional stochastic generalization of XOR, demonstrating the LOL and QOQ work even in scenarios that are quite distinct from the original motivating problems. In all 7 cases, LOL, or the appropriate generalization thereof, outperforms unsupervised, sparse, or other 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.

and sample size. Note that for completeness, we include two additional variants of LOL: LAL and LFL. LFL (short for Linear Fast Low-rank) replaces the standard SVD algorithm with a randomized variant, which can be much faster in certain situations [**?**]. LAL (short for Linear Approximate Low-rank) goes even one

step further, replacing SVD with random projections [**?**]. This variant of LOL is the fastest, its runtime is least sensitive to (p, d, n), and its accuracy is often commensurate (or better) than other variants of LOL. The runtime of all the variants of LOL are quite similar to FLD $\circ$ PCA. Given, given LOL's improved accuracy, and nearly identical simplicity, it seems there is very little reason to not use LOL instead of FLD $\circ$ PCA.

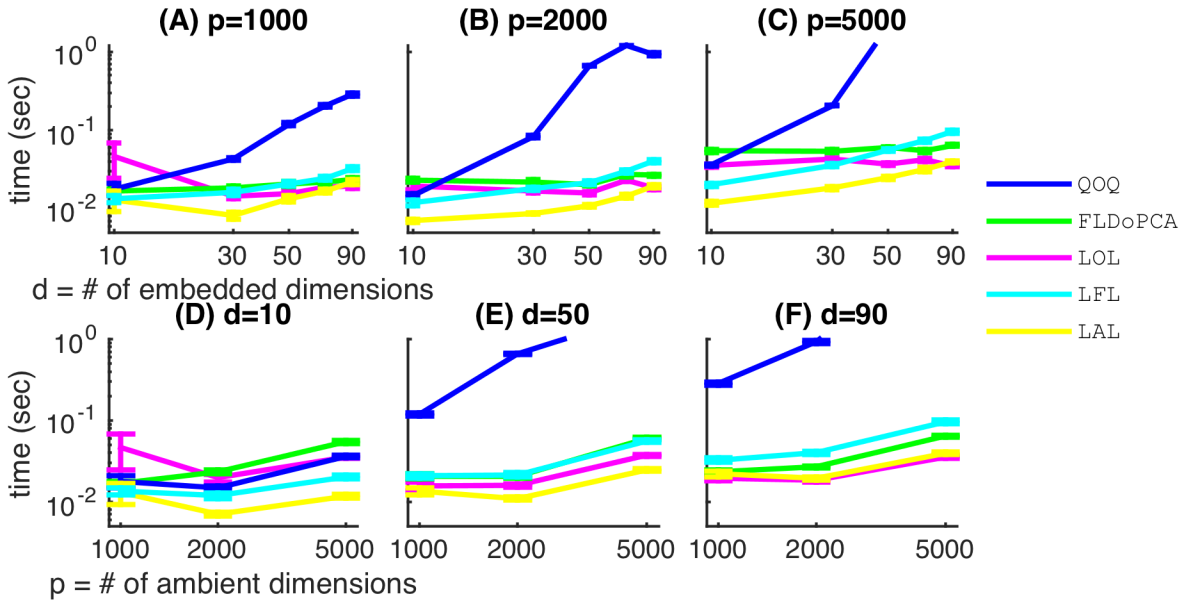

Figure 4: Computational efficiency of various low-dimensional projection methods. In all cases, $n = 100$, and we used the "stacked cigars" simulation parameters. We compare PCA with the projection steps from LOL, QOQ, LRL, LFL, and LAL, for different values of (p, d). The addition of the mean difference vector is essentially negligible. Moreover, for small d, the LFL is advantageous. LAL is always fastest, and its performance is often comparable to other methods (not shown).

Benchmark Real Data Applications

To more comprehensively understand the relative advantages and disadvantages of LOL with respect to other high-dimensional classification approaches, in addition to evaluating its performance in theory, and in a variety of numerical simulations, it is important to evaluate it also on benchmark datasets. For these purposes, we have selected four commonly used high-dimensional datasets (see Methods for details). For each, we compare LOL to (i) support vector machines (SVM), (ii) ROAD, (iii) lasso, (iv) and random forest (RF). Because in practice all these approaches have "hyperparameters" to tune, we consider several possible values for SVM, lasso, and LOL (but not RF, as its runtime was too high). Figure 5 shows the results for all four datasets.

Qualitatively, the results are similar across datasets: LOL achieves high accuracy and computational efficiency as compared to the other methodologies. Considering Figure 5(A) and (B), two popular sparse settings, we find that LOL can find very low dimensional projections with very good accuracy. For the prostate data, with a sufficiently non-sparse solution for ROAD, it slightly outperforms LOL, but at substantial computational cost, in particular, ROAD takes about 100 times longer to run on this dataset. Figure 5(C) and (D) are 10-class problems, so ROAD is no longer possible. Here, SVM can again slightly outperform LOL, but again, requiring 100 fold additional computational time. In all cases, the beloved random forest classifier performs subpar.

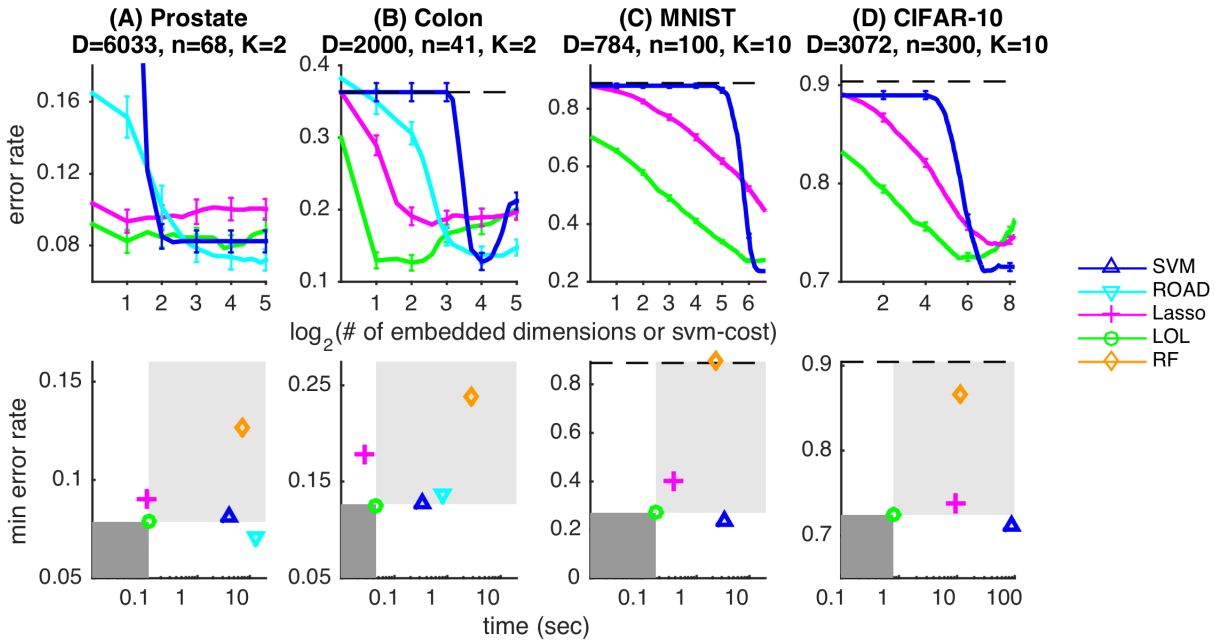

Figure 5: For four standard datasets, we benchmark LOL (green circles) versus standard classification methods, including support vector machines (blue up triangles), ROAD (cyan down triangles), LASSO (magenta pluses), and random forest (orange diamonds). Top panels show error rate as a function of $\log_2$ 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. **(C)** MNIST: 10 image categories here, 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 none of the problems is there an algorithm ever performing better and faster than LOL; rather, most algorithms typically perform worse and slower (though some are more accurate and much more computationally expensive. This suggests that regardless of how one subjectively weights computational efficiency versus accuracy, LOL is the best default algorithm in a variety of real data settings.

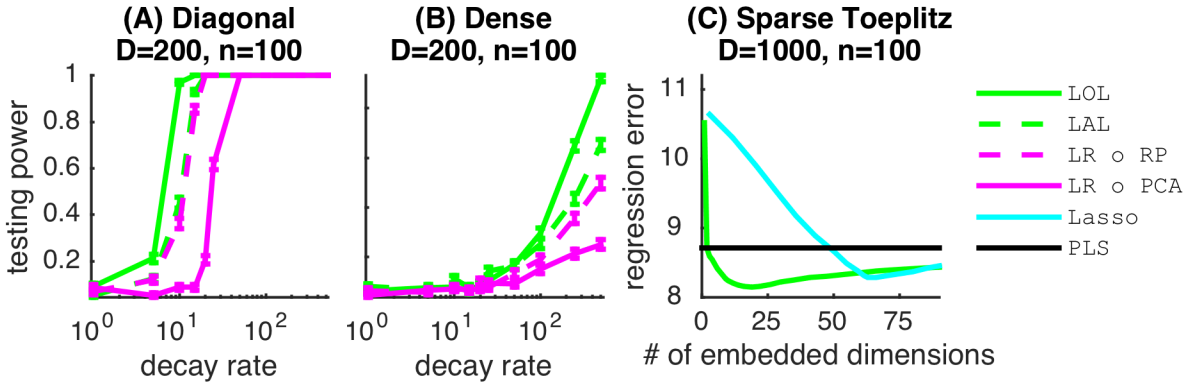

Figure 6: The intuition of including the mean difference vector is equally useful for other supervised manifold learning problems, including testing and regression. (A) and (B) show two different high-dimensional testing settings, as described in Methods. Power is plotted against the decay rate of the 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) shows a high-dimensional regression settings, as described in Methods. Log_{10} mean squared error is plotted against the number of embedded dimensions. Regression LOL composed with linear regression outperforms LASSO (cyan), the classic sparse regression method, as well as partial least squares (PLS; black). In the legend, 'A' denote either 'linear regression' (in (C)), or 'Hotelling' (in (A) and (B)). These three simulation settings therefore demonstrate the generality of this technique.

Extensions to Other Supervised Learning Problems

The utility of incorporating the mean difference vector into supervised machine learning for 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. 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 6(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 linear regression is another supervised learning method that can utilize this idea. Linear regression, like classification and Hotelling's Test, requires inverting a singular 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 6(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 PLS, 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 wide data. In particular, we have proposed a supervised manifold learning procedure, the utilizes both the difference of the means, and the covariance matrices. This is in stark contrast to previous approaches, which only utilize the covariance matrices (or kernel variants thereof), or solve a difficult optimization theoretic problem. In addition to demonstrating the accuracy and numerical 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.

Qualitative Design Considerations

Finally, in addition to the above mentioned quantitative criteria, there are also many qualitative criteria that merit consideration at each level of analysis. For the problem specification level, the model choices also determine the **appropriateness**. Indeed, real decision criteria are often quite difficult to specify precisely, and so our problem specification is almost always related to but not exactly the problem we hope to achieve. For example, while we might want to parse a scene, we often tackle a simpler problem of checking whether an object of a particular kind is present because it is easier. For the algorithm level, the **interpretability** of the resulting estimator is often an important consideration. For example, when developing biomarkers, it is important that the doctors and insurance companies can understand the decision making process of the algorithm, otherwise they might not believe the results. For the implementation level, how **accessible** is the implementation is crucial. In particular as more people outside the ivory academic tower—such as citizen scientists and even academics in the developing world—are contributing to data science and scientific discovery in general, access to the implementation is increasingly important. Finally, at the platform level, the **ease of use** of the platform is increasingly important. For example, requiring tedious installations of many different software libraries, or specialized hardware, can make the platform difficult to employ.

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.

Such manifold learning methods, while exhibiting both strong theoretical [**?** **?** **?**] and empirical performance, are fully unsupervised. Thus, in classification problems, they discover a low-dimensional representation of the data, ignoring the labels. This can be highly problematic when the discriminant dimensions and the directions of maximal variance in the learned manifold are not aligned (see Figure 1 for an example). Supervised dimensionality reduction techniques, therefore, combine the best of both worlds, explicitly searching for low-dimensional discriminant boundaries. A set of methods from the statistics community is collectively referred to as "sufficient dimensionality reduction" (SIR) or "first two moments" (F2M) methods [**?** **?** **?** **?** **?**]. These methods are theoretically elegant, but typically require the sample size to be larger than the number of observed dimensions (although see [**?**] for some promising work). Other approaches formulate an optimization problem, such as projection pursuit [**?**], empirical risk minimization [**?**], or supervised dictionary learning [**?**]. These methods are limited because they are prone to fall into local minima, they require costly iterative algorithms, and lack any theoretical guarantees [**?**]. Thus, there remains a gap in the literature: a supervised learning method with theoretical convergence guarantees appropriate when the dimensionality is orders of magnitude larger than the sample size.

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 [**?**], multiscale methods [**?**], and more scalable implementations [**?**], are all of immediate interest.

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.

A Theoretical Background

I.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, $y \in \{0, 1, \dots C\}$. 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:

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

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

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

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 $L_* := L_{g^*}$.

I.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:

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

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 $(\boldsymbol{\mu}_0 + \boldsymbol{\mu}1)/2 = \mathbf{0}$, re-arranging a bit, we obtain

$$g_{\text{Lda}}(\boldsymbol{x}) := \operatorname*{argmin}_y \boldsymbol{x}^\intercal \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_y.$$

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

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

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.

I.C LDA Model

A statistical model is a family of distributions indexed by a parameter $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, $\mathcal{F}_{\boldsymbol{\theta}} = \{F_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$. 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 $\boldsymbol{\Theta}_{C-\text{Lda}} = (\triangle_C, \mathbb{R}^{p \times C}, \mathbb{R}_{\succ 0}^{p \times p})$, where $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_C)$, $\triangle_C$ 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 $\mathcal{F}_{\text{Lda}} = \{F_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \boldsymbol{\Theta}_{\text{Lda}}\}$, dropping the superscript C for brevity where appropriate. The following lemma is well known:

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

Proof. Under the LDA model, the Bayes optimal classifier is available by plugging the explicit distributions into Eq. (1). □

B Projection Based Classifiers

Let $\boldsymbol{A} \in \mathbb{R}^{d\times p}$ be an orthonormal matrix, that is, a matrix that projects p dimensional data into a d dimensional subspace, where $\boldsymbol{A}\boldsymbol{A}^\mathsf{T}$ is the $d \times d$ identity matrix, and $\boldsymbol{A}^\mathsf{T}\boldsymbol{A}$ 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 $\boldsymbol{x}$ onto a low-dimensional subspace, and the classifying via LDA in that subspace is equivalent to redefining the parameters in the low-dimensional subspace, $\boldsymbol{\Sigma}_A = \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\mathsf{T} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{\delta}_A = \boldsymbol{A}\boldsymbol{\delta} \in \mathbb{R}^d$, and then using g_{Lda}. When $C = 2$, $\pi_0 = \pi_1$, and $(\mu_0 + \mu_1)/2 = \mathbf{0}$, this amounts to:

$$g^d_A(x) := \mathbb{I}\{(\boldsymbol{A}\boldsymbol{x})^\mathsf{T}\boldsymbol{\Sigma}_A^{-1}\boldsymbol{\delta}_A > 0\}, \text{ where } \boldsymbol{A} \in \mathbb{R}^{d\times p}. \tag{2}$$

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

Formally, we seek to solve the following optimization problem:

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

where $\boldsymbol{I}_{u\times v}$ is the $u\times v$ identity matrix identity, that is, $\boldsymbol{I}(i,j) = 1$ for all $i = j \leq \min(u, v)$, and zero otherwise. Let $\mathcal{A}^d = \{\boldsymbol{A} : \boldsymbol{A} \in \mathbb{R}^{d\times p}, \boldsymbol{A}\boldsymbol{A}^\mathsf{T} = \boldsymbol{I}_{d\times d}\}$, and let $\mathcal{A}_* \subset \mathcal{A}$ be the set of $\boldsymbol{A}$ that minimize Eq. (3), and let $\boldsymbol{A}_* \in \mathcal{A}_*$ (where we dropped the superscript d for brevity). Let $L^*_{\boldsymbol{A}} = L_{\boldsymbol{A}_*}$ be the misclassification rate for any $\boldsymbol{A} \in \mathcal{A}_*$, that is, $L^*_{\boldsymbol{A}}$ is the Bayes optimal misclassification rate for the classifier that composes $\boldsymbol{A}$ with LDA.

In our opinion, Eq. (3) is the simplest supervised manifold learning problem there is: a two-class classification problem, where the data are multivariate Gaussians with shared covariances, the manifold is linear, and the classification is done via LDA. Nonetheless, solving Eq. (3) is difficult, because we do not know how to evaluate the integral analytically, and we do not know any algorithms that are guaranteed to find the global optimum in finite time. This has led to previous work using a surrogate function [**?**]. We proceed by studying a few natural choices for $\boldsymbol{A}$.

II.A Bayes Optimal Projection

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

Proof. Let $\boldsymbol{B} = (\boldsymbol{\Sigma}^{-1}\boldsymbol{\delta})^\mathsf{T} = \boldsymbol{\delta}^\mathsf{T}(\boldsymbol{\Sigma}^{-1})^\mathsf{T} = \boldsymbol{\delta}^\mathsf{T}\boldsymbol{\Sigma}^{-1}$, so that $\boldsymbol{B}^\mathsf{T} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\delta}$, and plugging this in to Eq. (2), we obtain

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

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

II.B Principle Components Analysis (PCA) Projection

Principle Components Analysis (PCA) finds the directions of maximal variance in a dataset. PCA is closely related to eigendecompositions and singular value decompositions (SVD). In particular, the top principle component of a matrix $\boldsymbol{X} \in \mathbb{R}^{p \times n}$, whose columns are centered, is the eigenvector with the largest corresponding eigenvalue of the centered covariance matrix $\boldsymbol{X}\boldsymbol{X}^{\mathsf{T}}$. SVD enables one to estimate this eigenvector without ever forming the outer product matrix, because SVD factorizes a matrix $\boldsymbol{X}$ into $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal $p \times n$ matrices, and $\boldsymbol{S}$ is a diagonal matrix, whose diagonal values are decreasing, $s_1 \geq s_2 \geq \cdots > s_n$. Defining $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n]$, where each $\boldsymbol{u}_i \in \mathbb{R}^p$, then $\boldsymbol{u}_i$ is the i^{th} eigenvector, and s_i is the square root of the i^{th} eigenvalue of $\boldsymbol{X}\boldsymbol{X}^{\mathsf{T}}$. Let $\boldsymbol{A}_d^{\text{Pca}} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d]$ be the truncated PCA orthonormal 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:

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

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

Moreover, let $\boldsymbol{U}_d = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d] \in \mathbb{R}^{p \times d}$, and note that $\boldsymbol{U}_d^{\mathsf{T}}\boldsymbol{U}_d = \boldsymbol{I}_{d \times p}$ and $\boldsymbol{U}_d^{\mathsf{T}}\boldsymbol{U}_d = \boldsymbol{I}_{p \times d}$. Similarly, let $\boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^{\mathsf{T}} = \boldsymbol{\Sigma}$, and $\boldsymbol{U}\boldsymbol{S}^{-1}\boldsymbol{U}^{\mathsf{T}} = \boldsymbol{\Sigma}^{-1}$. Let $\boldsymbol{S}_d$ be the matrix whose diagonal entries are the eigenvalues, up to the d^{th} one, that is $\boldsymbol{S}_d(i,j) = s_i$ for $i = j \leq d$ and zero otherwise. Similarly, $\boldsymbol{\Sigma}_d = \boldsymbol{U}\boldsymbol{S}_d\boldsymbol{U}^{\mathsf{T}} = \boldsymbol{U}_d\boldsymbol{S}_d\boldsymbol{U}_d^{\mathsf{T}}$.

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

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

Proof. Plugging $\boldsymbol{U}_d$ into Eq. (2) for $\boldsymbol{A}$, and considering only the left side of the operand, we have

$$\begin{aligned}
(\boldsymbol{A}\boldsymbol{x})^{\mathsf{T}}\boldsymbol{\Sigma}_A^{-1}\boldsymbol{\delta}_A &= \boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{\Sigma}^{-1}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{\delta}, \\
&= \boldsymbol{x}^{\mathsf{T}}\boldsymbol{U}_d\boldsymbol{U}_d^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}_d\boldsymbol{U}_d^{\mathsf{T}}\boldsymbol{\delta}, \\
&= \boldsymbol{x}^{\mathsf{T}}\boldsymbol{U}_d\boldsymbol{U}_d^{\mathsf{T}}\boldsymbol{U}\boldsymbol{S}^{-1}\boldsymbol{U}\boldsymbol{U}_d\boldsymbol{U}_d^{\mathsf{T}}\boldsymbol{\delta}, \\
&= \boldsymbol{x}^{\mathsf{T}}\boldsymbol{U}_d\boldsymbol{I}_{d \times p}\boldsymbol{S}^{-1}\boldsymbol{I}_{p \times d}\boldsymbol{U}_d^{\mathsf{T}}\boldsymbol{\delta}, \\
&= \boldsymbol{x}^{\mathsf{T}}\boldsymbol{U}_d\boldsymbol{S}_d^{-1}\boldsymbol{U}_d^{\mathsf{T}}\boldsymbol{\delta}, \\
&= \boldsymbol{x}^{\mathsf{T}}\boldsymbol{\Sigma}_d^{-1}\boldsymbol{\delta}.
\end{aligned}$$

□

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

II.C Linear Optimal Low-Rank (LOL) Projection

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

to ensure that they are balanced appropriately, we normalize $\boldsymbol{\delta}$

each vector in $\boldsymbol{\delta}$ to have norm unity. Formally, let $\widetilde{\boldsymbol{\delta}}_j = \boldsymbol{\delta}_j / \|\boldsymbol{\delta}_j\|$, where $\boldsymbol{\delta}_j$ is the j^{th} difference of the mean vector (remember, the number of vectors is equal to $C - 1$, where C is the total number of classes),

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

C Theoretical Properties of LDA based Classifiers

III.A LDA is rotationally invariant

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

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

$$L^{F}_{g} = L^{\boldsymbol{W}\circ F}_{g}, \qquad F \in \mathcal{F}.$$

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

Now, we can state the main lemma of this subsection: LDA is rotationally invariant.

Lemma 4. *$L^{F}_{Lda} = L^{\boldsymbol{W}\circ F}_{Lda}$, for any $F \in \mathcal{F}$.*

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

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

□

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

III.B Rotation of Projection Based Linear Classifiers g_A

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

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

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

III.C Simplifying the Objective Function

Recalling Eq. (2), a projection based classifier is effectively thresholding the dot product of $\boldsymbol{x}$ with the linear projection operator $\boldsymbol{P}_A := \boldsymbol{A}^{\mathsf{T}}\boldsymbol{\Sigma}_A^{-1}\boldsymbol{\delta}_A \in \mathbb{R}^p$, and let $\boldsymbol{P}_* := \boldsymbol{P}_{\boldsymbol{A}_*}$. Unfortunately, the nonlinearity in in Eq. (3) makes analysis difficult. However, because of the linear nature of the classifier and projection matrix operator, an objective function that is simpler to evaluate is available. Define $\angle(\boldsymbol{P}, \boldsymbol{P}') = \frac{\boldsymbol{P}^{\mathsf{T}}\boldsymbol{P}'}{||\boldsymbol{P}||_2||\boldsymbol{P}'||_2} \in (0, 1)$, and consider

$$
\begin{aligned}
&\underset{\boldsymbol{A}}{\text{minimize}} && -\angle(\boldsymbol{P}_A, \boldsymbol{P}_*), \\
&\text{subject to} && \boldsymbol{A} \in \mathbb{R}^{p\times d}, \quad \boldsymbol{AA}^{\mathsf{T}} = \boldsymbol{I}_{d\times d}.
\end{aligned} \tag{4}
$$

Lemma 6. *The solution to Eq. (4) is also the solution to Eq. (3) for any given d.*

Proof. The minimum of Eq. (4) is clearly $\boldsymbol{A} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\delta}$, which is also the minimum of Eq. (3). □

Remark 1. *$\angle$ is merely the angle between two vectors, and is therefore scale invariant. In other words, $\angle(\boldsymbol{P}_A, \boldsymbol{P}) = \angle(\boldsymbol{P}_{cA}, \boldsymbol{P})$, for any $c > 0$.*

Given the above, we can evaluate various choices of $\boldsymbol{A}$ in terms of their induced projection operator $\boldsymbol{P}_A$ and the angle between said projection operators and the Bayes optimal projection operator.

Lemma 7. $\angle(\boldsymbol{P}_A, \boldsymbol{P}_*) < 1 \implies L_A > L_*$

Proof. If $\angle(\boldsymbol{P}_A, \boldsymbol{P}_*) < 1$, then there exists an $\boldsymbol{x}$ such that $\mathbb{I}\{\boldsymbol{x}^{\mathsf{T}}\boldsymbol{P}_A > 0\} \neq \mathbb{I}\{\boldsymbol{x}^{\mathsf{T}}\boldsymbol{P}_* > 0\}$, and therefore, $L_A > L_*$. □

Conjecture 1.

$$\angle(\boldsymbol{P}_A, \boldsymbol{P}_*) \leq \angle(\boldsymbol{P}_B, \boldsymbol{P}_*) \implies L_A \leq L_B.$$

III.C(1) When $d = 1$

Remark 2. *If $\boldsymbol{A} \in \mathcal{A}_*$ and $\boldsymbol{B} \notin \mathcal{A}_*$, then $\angle(\boldsymbol{A}^{\mathsf{T}}, \boldsymbol{A}_*^{\mathsf{T}}) = 1$ and $\angle(\boldsymbol{B}^{\mathsf{T}}, \boldsymbol{P}_*^{\mathsf{T}}) < 1$, and therefore $\angle(\boldsymbol{A}^{\mathsf{T}}, \boldsymbol{A}_*^{\mathsf{T}}) > \angle(\boldsymbol{B}^{\mathsf{T}}, \boldsymbol{A}_*^{\mathsf{T}})$.*

Conjecture 2. *When $d = 1$:*

$$\angle(\boldsymbol{A}^{\mathsf{T}}, \boldsymbol{A}_*^{\mathsf{T}}) \leq \angle(\boldsymbol{B}^{\mathsf{T}}, \boldsymbol{A}_*^{\mathsf{T}}) \implies \angle(\boldsymbol{P}_A, \boldsymbol{P}_*) \leq \angle(\boldsymbol{P}_B, \boldsymbol{P}_*).$$

Proof. I believe so, but i don't see how to prove it. A little arithmetic shows that the left hand side means that:

$$\boldsymbol{A\Sigma}^{-1}\boldsymbol{\delta} > \boldsymbol{B\Sigma}^{-1}\boldsymbol{\delta}.$$

Similarly, the right hand side means that:

$$\boldsymbol{\delta}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{\Sigma}^{-1}\boldsymbol{A}^{\mathsf{T}}a > \boldsymbol{\delta}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{B}\boldsymbol{\Sigma}^{-1}\boldsymbol{B}^{\mathsf{T}}b,$$

where we substituted $a := \boldsymbol{A}\boldsymbol{\Sigma}^{-1}\boldsymbol{\delta}$ and $b := \boldsymbol{B}\boldsymbol{\Sigma}^{-1}\boldsymbol{\delta}$.

But I do not see how to go any further. □

III.D PCA versus LOL

We would like to prove that LOL is always better than PCA, when using one or the other to project the data onto a low dimensional space, followed by classifying with LDA, under a wide variety of of settings. Formally, we would like to prove $\mathbb{P}[L^d_{\text{Lol}} \leq L^d_{\text{Pca}}]$ is big. To do so, we ask a sequence of increasingly sophisticated questions that have the following form:

1. Under which parameter settings is LOL better than PCA?
2. How often do those parameter settings arise, under various statistical models of the parameters?

Recall that for the C-class classification problem, the parameter from which we sample the data is $\boldsymbol{\theta}_c = (\pi_c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$, where

- the class probabilities are non-negative and sum to unity: $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_C) \in \triangle_C := \sum_{c \in \mathcal{C}} \pi_c = 1$ and $\pi_c \geq 0 \forall c \in \mathcal{C}$,
- the class means are p-dimensional vectors: $\boldsymbol{\mu}_c \in \mathbb{R}^p$ is the class c mean vector, and
- the class covariances are positive definite $p \times p$ real matrices: $\boldsymbol{\Sigma}_c \in \mathbb{R}^{p \times p}_+$ is the class conditional covariance matrix (and $\mathbb{R}^{p \times p}_+$ is the set of positive definite real $p \times p$ matrices).

The fully unconstrained parameter space is therefore $\boldsymbol{\Theta} = \{\triangle_C \times (\mathbb{R}^p, \mathbb{R}^{p \times p}_+)^C\}$.

When using a projection based classifier, we also have the hyperparameter d which specifies the the dimensionality of the low-dimensional projection.

We thus try to answer the above two questions for increasingly relaxed constraints on the parameter space and d. To start, we assume that:

1. we have only two classes, $C = 2$;
2. each class has the same covariance matrix, $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1$.

For simplicity (but without loss of generality), we will also assume that the two classes have centered means and equal priors, that is, $(\boldsymbol{\mu}_0 + \boldsymbol{\mu}_1)/2 = 0$ and $\pi_0 = \pi_1 = 1/2$ (relaxing this assumption merely changes the threshold of the classifier). Let the parameter space defined by these constraints be denoted $\boldsymbol{\Theta}' = \{\mathbb{R}^{2p}, \mathbb{R}^{p \times p}_+\}$. In Section III.D, we will always assume $\boldsymbol{\theta} \in \boldsymbol{\Theta}'$, unless otherwise specified.

Given the above, we consider the following sequence of relaxations, first when $d = 1$, and then when $d < p$:

1. the covariance is a scaled identity matrix, that is, $\boldsymbol{\Sigma} = k\boldsymbol{I}$;
2. the covariance is a diagonal matrix, that is, $\boldsymbol{\Sigma} = \boldsymbol{S}$, where $\boldsymbol{S}_{ij} = \sigma_i$ when $i = j$ and is zero otherwise;
3. the covariance is an arbitrary positive definite matrix, that is $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}_{\succ}$.

After those, extensions to the multiclass and/or different covariance matrix setting might then also be explored.

III.D(1) $d = 1$

In this section, we will only consider $d = 1$, meaning that

- LOL is simply $\boldsymbol{\delta}$,
- PCA is simply $\boldsymbol{u}_1$, the eigenvector corresponding to the largest eigenvalue of $\boldsymbol{\Sigma}$.

III.D(1).1 Scaled Identity Covariance Matrix

Lemma 8. LOL *better than* PCA *almost always when* $\boldsymbol{\Sigma} = k\boldsymbol{I}$.

Proof. To prove this statement, we first show how to define $\boldsymbol{A}^1_{\text{Lol}}$ and $\boldsymbol{A}^1_{\text{Pca}}$ is this setting.

- $\boldsymbol{A}^1_{\text{Lol}}$ is simply $\boldsymbol{\delta}$, regardless of the covariance matrices and priors.
- When $\boldsymbol{\Sigma} = k\boldsymbol{I}$, $\boldsymbol{A}^1_{\text{Pca}}$ is a random vector, because the first principal component of a scaled identity matrix can equally be defined as any basis vector.

Thus, only when the first eigenvector of the covariance matrix is randomly assigned to $\boldsymbol{\delta}$ will PCA work as

well as LOL. □

Given the conditions under which LOL is better than PCA, we next ask how often that happens, under a reasonable model assumption.

Lemma 9. $\mathbb{P}[L^1_{Lol} \leq L^1_{Pca}] = 1$ *when* $\boldsymbol{\Sigma} = k\boldsymbol{I}$ *and* $\boldsymbol{\delta} \sim \mathcal{N}(\boldsymbol{\mu}_\delta, \boldsymbol{\Sigma}_\delta)$.

Proof. When $\boldsymbol{\delta} \sim \mathcal{N}(\boldsymbol{\mu}_\delta, \boldsymbol{\Sigma}_\delta)$, the probability that it exactly equals any vector $\boldsymbol{x} \in \mathbb{R}^d$ is real, this includes, of course, $\boldsymbol{u}_1$, the first eigenvector of $k\boldsymbol{I}$. Thus, $\mathbb{P}[\boldsymbol{\delta} \propto \boldsymbol{u}_1] = 0$, no matter how $\boldsymbol{u}_1$ is chosen (as long as it is not chosen using the knowledge of the value of $\boldsymbol{\delta}$). This implies that $\mathbb{P}[\angle(\boldsymbol{u}_1^\mathsf{T}, \boldsymbol{A}_*^\mathsf{T}) = 1] = 0$, and therefore, $\mathbb{P}[L_{\boldsymbol{u}_1} \geq L_*] = 1$. □

III.D(1).2 Diagonal covariance matrix

Now consider a simple generalization of the above scenario, namely, $\boldsymbol{\Sigma} = \boldsymbol{S}$ is diagonal. We want to know under what conditions is LOL better than PCA, and how often that happens under a reasonable model.

Let $\boldsymbol{s}$ be the singular values of $\boldsymbol{\Sigma}$. When $\boldsymbol{\Sigma}$ is diagonal, its diagonal elements are $s_1, \ldots, s_p$, and we call that matrix $\boldsymbol{S}$. Recalling that $s_i = \lambda_i^2$, where λ_i's are eigenvalues, and when a matrix is diagonal, $s_i = \lambda_i^2$, we have $\boldsymbol{A}_* = \boldsymbol{S}^{-1}\boldsymbol{\delta} = \boldsymbol{s}^{-1} \odot \boldsymbol{\delta} := \widetilde{\boldsymbol{\delta}} = (\delta_1/\lambda_1^2, \ldots, \delta_p/\lambda_p^2)$.

Lemma 10. LOL *is better than* PCA *whenever* $1 - \delta_1 < s_1/\delta_1 \sum_{i=2}^p \delta_i^2/s_i$. *A consequence of this is that* LOL *is always better than* PCA *whenever* $\delta_1 > 1$. *This means that when the variables are independent*

Proof. Recalling that $\boldsymbol{u}_i = \boldsymbol{e}_i$, where $\boldsymbol{e}_i$ is a vector of all zeros except a one in the i^{th} element, and let $\boldsymbol{u}^i = \boldsymbol{u}_i$ for simplicity here, we can show that:

$$\angle(\boldsymbol{\delta}, \widetilde{\boldsymbol{\delta}}) = \sum_i \delta_i \widetilde{\delta}_i = \sum_i \delta_i s_i^{-1} \delta_i = \sum_i \delta_i^2/s_i$$

and

$$\angle(\boldsymbol{u}^1, \widetilde{\boldsymbol{\delta}}) = \sum_i u_i^1 \widetilde{\delta}_i = \sum_i u_i^1 s_i^{-1} \delta_i = \sum_i u_i^1 \delta_i/s_i = u_1^1 \delta_1/s_1 = \delta_1/s_1.$$

Thus, $\angle(\boldsymbol{\delta}, \widetilde{\boldsymbol{\delta}}) > \angle(\boldsymbol{u}_1, \widetilde{\boldsymbol{\delta}})$ any time that $\sum_i \delta_i^2/s_i > \delta_1/s_1$. Note that at a minimum, it must be that $\delta_1 < 1$.

Conjectures 1 and 2 demonstrate that if the angle is better, than Bayes error is also better, and this completes the proof. □

If Conjecture 1 and 2 are true, then the above also implies that LOL's Bayes error is also better than PCA's, in this setting. Numerical experiments suggest this is true, and moreover, that when randomly sampling $\boldsymbol{\delta}$ and $\boldsymbol{S}$, LOL always does better than PCA.

Conjecture 3. LOL *is almost always better than* PCA *whenever* $s_i \overset{iid}{\sim} \mathcal{U}(0,1)$ *and* $\delta_i \overset{iid}{\sim} \mathcal{N}(0,1)$ *for all* $i \in [p]$.

Proof. We start with the very simple scenario of $p = 1$ and condition on s to build intuition. In this setting we have:

$$\mathbb{P}[\delta^2/s > \delta/s] = \mathbb{P}[\delta^2/s - \delta/s > 0]. \tag{5}$$

Let $z = x + y$, where $x = \frac{\delta^2}{s}$ and $y = \frac{\delta}{-s}$. So, x has a generalized chi-squared distribution and y has a normal distribution. A bit of algebra reveals the explicit form of this distribution, however, it includes an integral that we do not know how to evaluate. Thus, instead of solving exactly for the above, we provide a bound.

Bennett's inequality states given $X_1, \ldots, X_n$, assume:

- each X_i is independent of all others

- (without loss of generality) $\mathbb{E}[X_i] = 0 \forall i$
- $|X_i| < a$ almost surely for all i
- $\sigma^2 = \frac{1}{n}\sum_i \mathbb{V}(X_i)$.

Then, for any $t \geq 0$

where $h(u) = (1+u)\log(1+u) - u$.

Recalling that $\mathbb{V}(\sum_i X_i) = \sum_i \mathbb{V}(X_i)$ for independent random variables, and that

1. $\mathbb{V}(\delta/s) = s^{-2}$
2. $\mathbb{V}(\delta^2/s) = h(s)$

where Φ is the cumulative distribution function for a standard normal random variable.

Note that re-introducing randomness into s

Coming soon. "almost always" and "better" mean with probability 1 and better in the sense of the angle between $\boldsymbol{A}^1_{\texttt{Lol}}$ and the Bayes vector is larger than that of $\boldsymbol{A}^1_{\texttt{Pca}}$ and the Bayes vector. I believe this also implies Bayes error is better, but we have not yet shown that. □

III.D(1).3 Arbitrary Covariance Matrix

III.D(2) $d \geq 1$

III.D(3) Scaled Identity Covariance Matrix

III.D(4) Diagonal Covariance Matrix

III.D(5) Arbitrary Covariance Matrix

D Asymptotic Theory

In real data problems, the true joint distribution is unknown. Instead, what is provided is a set of training data. We therefore assume the existence of n training samples, each of which has been sampled identically and independently from the same distribution, $(\boldsymbol{X}_i, Y_i) \overset{iid}{\sim} F_{\boldsymbol{X},Y}$, for $i = 1, 2, \ldots, n$. We can use these training samples to then estimate $f_{x|y}$ and f_y. Plugging these estimates in to Eq. (**??**), we obtain the Bayes plugin classifier:

$$\hat{g}_n^*(\boldsymbol{x}) := \underset{y}{\operatorname{argmax}}\, \hat{f}_{\boldsymbol{x}|y}\hat{f}_y. \tag{6}$$

Under suitable conditions, it is easy to show that this Bayes plugin classifiers performance is asymptotically optimal. Formally, we know that: $L_{\hat{g}_n^*} \to L_{g^*}$.

When the parameters, and we want to use a linear approach, we can implement a Bayes plug-in `LDA`, which we call Fisher's Discriminant Analysis (`FLD`) [**?**]. Under the two-class, equal prior, and centered means assumption, we have

$$\hat{g}_n^*(\boldsymbol{x}) := \mathbb{I}\{\boldsymbol{x}^\mathsf{T}\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\delta}} > 0\}, \tag{7}$$

and $L_{\hat{g}_n}$ is the misclassification rate for an estimated classifier, $\hat{g}_n$. Unfortunately, when $p \gg n$, the estimate of the covariance matrix $\boldsymbol{\Sigma}$ will be low-rank, and therefore, not invertible (because an infinite number of solutions all fit equally well). In such scenarios, we seek alternative methods, even in the `LDA` model.

We would like to prove:

Lemma 11. $L^d_{\hat{Lol}} \to L_*$ *for any* $\boldsymbol{\theta} \in \boldsymbol{\Theta}'$.

Lemma 12. $\mathbb{P}[L^d_{\hat{Pca}} \to L_*] > 0$ *for any* $\boldsymbol{\theta} \in \boldsymbol{\Theta}'$

```
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.

E Finite Sample Theory

It would be awesome to prove something like:

Theorem 2.

$$\mathbb{P}[\hat{L}^d_{Lol} - \hat{L}^2_{Pca} > 0] < f(\boldsymbol{\theta}, d, n),$$

which would state that LOL is better than PCA, again, under suitable assumptions.

Theorem 3.

$$\mathbb{P}[\boldsymbol{P}^\mathsf{T}_{Pca}\boldsymbol{P}_* - \boldsymbol{P}^\mathsf{T}_{Lol}\boldsymbol{P}_* > t\,\|\boldsymbol{P}_A\|\,\|\boldsymbol{P}_*\|] < f(t, p, d),$$

which would state that LOL is better than PCA, again, under suitable assumptions.

In terms of distributiosn of the above, it seems that perhaps we could start simple. Assume for the moment that $\boldsymbol{\delta}, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_p \overset{iid}{\sim} \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$, and let $\boldsymbol{\Lambda} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p)^\mathsf{T}$, and $\boldsymbol{\Sigma} = \boldsymbol{\Lambda}^\mathsf{T}\boldsymbol{\Lambda}$.

The reason the above is probabilistic is because it is under certain assumptiosn on the *distributions* of $bdel$, $\boldsymbol{\Sigma}$, and $\boldsymbol{A}$.

Perhaps even simpler is to start with specific assumptions about $\boldsymbol{\delta}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{A}$. Because LDA is rotationally invariant, I believe that we can assert, without loss of generality, that $\boldsymbol{\Sigma} = \boldsymbol{S}$, where $\boldsymbol{S}$ is a diagonal matrix with diagonal entries $\sigma_1, \ldots, \sigma_p$, where all $\sigma_j > 0$. Now, the optimal projection $\boldsymbol{\Sigma}^{-1}\boldsymbol{\delta}$ is just a simple dot product, $\boldsymbol{d}^\mathsf{T}\boldsymbol{\delta}$, where $\boldsymbol{d}$ =diag($\boldsymbol{S}$)$\in \mathbb{R}^p$.

For example, letting $\boldsymbol{A} = \boldsymbol{U}_d$, and letting $\boldsymbol{U}_i = e_i$ be the unit vector, with zeros everywhere except a one in the i^{th} position, we have

$$\boldsymbol{P}^\mathsf{T}_A\boldsymbol{P}_* = \boldsymbol{\delta}^\mathsf{T}\boldsymbol{U}^\mathsf{T}_d\boldsymbol{U}_d\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^\mathsf{T}_d\boldsymbol{U}_d\boldsymbol{\Sigma}^{-1}\boldsymbol{\delta}\boldsymbol{\delta}^\mathsf{T}\boldsymbol{\Sigma}_d\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_d\boldsymbol{\Sigma}^{-1}\boldsymbol{\delta} = \boldsymbol{\delta}^\mathsf{T}\boldsymbol{\Sigma}^{-2}\boldsymbol{\delta}.$$

So, we want to understand the probability that $\alpha_{\texttt{Pca}}$ is small under different parameter settings, $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

F The R implementation of LOL

Figure 8 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.