Unique Component Analysis

Robin Tu[1], Alexander H. Foss [2], and Sihai Dave Zhao[1]

[1]Department of Statistics, University of Illinois at Urbana-Champaign, Champaign, IL

[2]Statistical Sciences, Sandia National Laboratories, Albuquerque, NM

November 16, 2020

1 Introduction

Dimension reduction for high-dimensional data visualization and exploration is an universal problem across many disciplines. For continuous data, methods include Principal Component Analysis (PCA [10]), Nonnegative Matrix Factorization [14], Projection Pursuit [9], Independent Component Analysis (ICA [7]), with newer nonlinear techniques such as t-SNE and UMAP gaining popularity in the field of bioinformatics [CITATION??].

However, data is universally plagued by biological and/or technical variability which reduce the effectiveness of dimension reduction methods to find meaningful latent structure/variables. To handle these sources of variation, vast problem specific (sometimes technology specific: methylation 450k vs 850k) pipelines have been established to remove experimental artifacts such as batch effects (batch normalization), inherent within sample variation, and correct for machine/data generating error [17]. These pipelines sometimes make assumptions about the data generating process, such as removing top principal components (PCs) due to mostly comprising of noise, possibly discarding relevant information. This is especially problematic in high-dimensions, with limited samples.

Contrastive methods are a recently developed class of procedures for simultaneously removing unwanted variation while performing dimension reduction. By leveraging a background dataset that represents unwanted or known variation, contrastive methods seek to model or explain variation unique to the target data of interest, variation which doesn't exist in the background. There are many other types of contrastive implementations including latent models [21] and autoencoders [2], but we focus on dimension reduction for continuous data types, specifically contrastive PCA [1].

Abid et al introduced the contrastive methodology with contrastive PCA (cPCA) [1], which utilized a secondary background data set of known and unwanted variability to uncovered new unique directions of maximal variation in a target data. Let A denoted the $p \times p$ sample covariance matrix constucted from target data $X_{n_x \times p}$ and B denote the $p \times p$ sample background covariance constructed from background data $Y_{n_y \times p}$. For some contrastive parameter λ, cPCA seeks the eigenvectors v satisfying

$$\max_{v \in \mathbb{R}^p} v^T (A - \lambda B) v$$

resulting in eigenvectors which maximize variability in the target data, while simultaneously minimizing variability in the background data. The tuning parameter λ, which measures how much to penalize the background data covariance, greatly impacts the resulting eigenvectors. $\lambda = 0$ reduces cPCA to normal PCA since no weight is applied to the background data whereas a $\lambda \to \infty$ is equivalent to PCA after projecting the target data onto the nullspace of the background data. The authors of cPCA suggested that λ is chosen using spectral clustering via a parameter sweep of logarithmically spaced candidate values.

cPCA has already inspired several variants. Fujiwara et al. [11] applied cPCA to identify variables most influential in maximally contrasting clusters by using the entire data as the target data, and the data of interest as the background. The authors suggest choosing λ from a similar candidate set which maximize the separation between the target cluster and all other data while still having enough variance in the target cluster K. Salloum and Kuo [20] introduced cPCA++, where they seek to maximize the ratio, rather than the difference, of the variance explained in the target to the variance explained in the background.Boileau et al. [4] described Sparse cPCA (scPCA) by adding l_1 and l_2 penalties to sparsify the estimated eigenvectors of the contrastive covariance matrix.

However, there are two main disadvantages to these existing methods. First, they cannot accommodate multiple background datasets. Multiple backgrounds would allow researchers to target the exact variability of interest by removing unwanted variation from multiple sources and/or conditions. For example... Naively applying existing methods by stacking the multiple background data matrices together and forming a pooled covariance matrix will not work well. The pooled covariance is a sample-weighted average of the individual background covariance matrices and may not be representative of any of the individual datasets.

The second disadvantage of existing contrastive methods is that they require one (e.g., cPCA and ccPCA) or several (e.g., scPCA) tuning parameters, user input variables that influence an algorithm's behavior. Since existing contrastive methods heavily depend on the tuning parameter, finding the optimal tuning parameter(s) which balances the dual mandate of maximizing variability in the target and minimizing variability variability in the background may be tedious and frustrating. Although cPCA and ccPCA have criteria-based tuning parameter solutions, the right tuning parameter may not be among the default candidates. An apparent exception is cPCA++, but in practice it requires a tuning parameter when its background sample covariance matrix is not positive definite, which is common many applications.

We propose Unique Component Analysis (UCA) to address both of these issues. UCA works by constraining the objective function in cPCA to give directions that are orthonormal and explain small amounts of variation in each background dataset. Specifically, let A be the $p \times p$ target covariance matrix and B_j, $j \in 1, ..., m$ be the m $p \times p$ background covariance matrices. We seek to:

$$\max_{v\in\mathbb{R}^p} v^T A v$$
$$\text{subject to} \;\; v^T v = I, \;\; v^T B_1 v \leq 1, \cdots, v^T B_m v \leq 1$$

These constraints are naturally found in generalized eigenvalue problems and can be solved using weak duality of the Lagrangian. UCA is tuning-free, is easily extensible to multiple background data sets, and is scalable to high-dimensions. We hope to provide a way for researchers to explore higher dimensional data and uncover variation unique to the data of interest, variation not shown in previous studies, control groups, and other known sources of variation, simultaneously. UCA seeks to find the unique directions of variation which both maximize the varability in the target dataset and also account for little variation in each of the background dataset(s). We show that UCA achieves similar results as cPCA and cPCA++ with a single background dataset and that it can outperform them when using multiple backgrounds.

Another way to conduct contrastive dimension reduction is with "Residual PCA". We note that residual PCA (rPCA) is not a formal technique but is used as an ad-hoc method of removing noise from known variability [15]. Again, let A be the $p \times p$ target covariance matrix and B be the $p \times p$ background covariance matrix. For some integer k which denotes the low dimensional representation of B, rPCA performs PCA after projecting A onto the "residual" space of B. More specifically, if W denotes the eigenvectors of B and $W_{(k+1):p}$ the top $(k+1) : p$ eigenvectors of B,then rPCA seeks to:

$$\max_{v\in\mathbb{R}^p} v^T \left(W_{(k+1):p} W_{(k+1):p}^T A \right) v$$

Note that cPCA with contrastive parameter $\lambda \to \infty$ is different from rPCA because rPCA requires user to choose the number of background components to be kept, whereas cPCA with $\lambda \to \infty$ maps reduces the target on the nullspace of the background– they would be equivalent for $k = p$. We present rPCA to demonstrate that some of the same underlying directions can be found in a more intuitive way. It also can easily be extensible to multiple backgrounds by projecting onto addition background residual spaces. Since the residual space of the background can be done easily with singular-value decomposition, it is computationally simple and straight forward. However, one still needs to choose the background data dimension, k. To solve this tuning parameter problem, we use the elbow method of finding the best linear single knot spline at k such that the mean-square error is minimized in the scree plot.

2 Results

2.1 Subgroup discovery in protein expression data

We make a direct comparison between UCA and cPCA by using the mouse protein expression dataset. This dataset measured expression of 76 mice proteins of 1080 control or trisomic (Down Syndrome) mice samples (each mice measurement considered an independent sample/mouse) receiving a combination of shock therapy (Context Shock vs. Shock

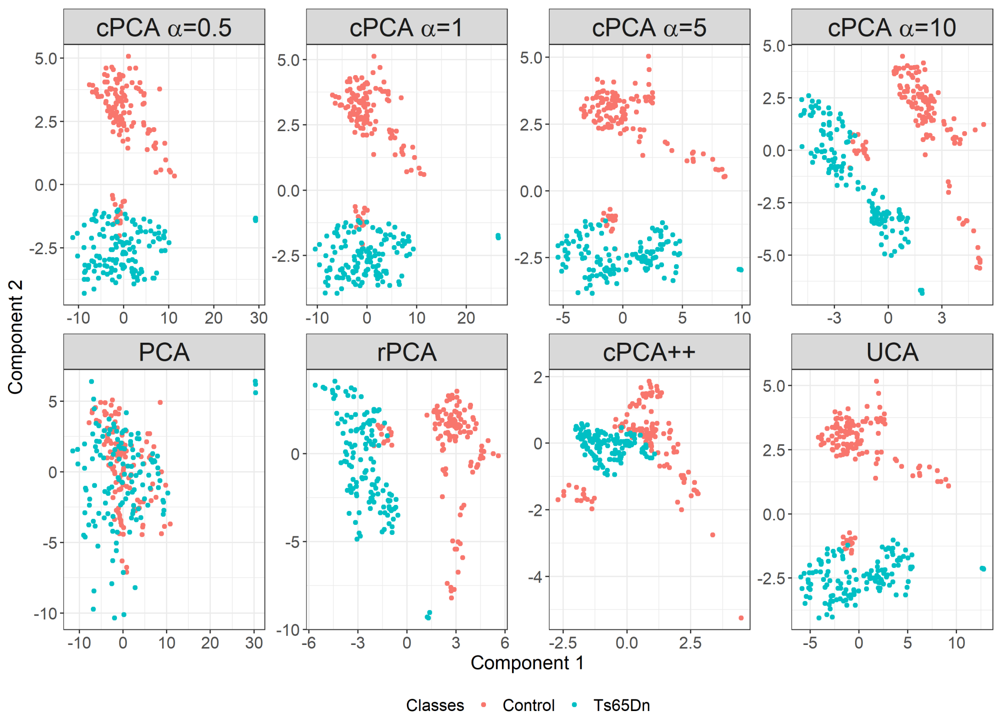

Figure 1: Mouse Protein Expression: Separability of Saline injected mice which were not shocked. confirming [1] results where a background of Saline injected mice who were shocked was used. This example is unseparable using PCA alone but is easily separable by cPCA for all values of α, cPCA++, rPCA, and UCA.

Context) and drugs (Saline vs. Memantine) [3, 13, 1]. Since saline treated mice not exposed to shock therapy (Shock Context) are similar to those that are saline treated and exposed (Context Shock) to shock therapy, we can investigate the variation unique to unshocked mice by removing variation resulting from shock therapy by contrasting shocked control mice protein expression measurements– an analysis that was carried out in the original cPCA paper [1]. We compare the directions found by UCA to those found by cPCA ($\lambda = 0.5, 1, 5, 10$), cPCA++, PCA, and rPCA on Saline injected mice which have not received shock therapy (Shock Context) and use the Down Syndrome information to evaluate these methods.

The results in Figure 1 show that PCA has difficulty distinguishing between mice with Down Syndrome and mice without. In this example, the resulting separation in the first two contrastive principal components does not really depend on the chosen λ. At all values of λ, cPCA achieves good separation, therefore it is unsurprising UCA has similar results. These results align with the results found in [1]. cPCA++, which also does not require specifying a contrastive tuning parameter, does better than PCA at separating control and Down Syndrome mice, but falls short of achieving similar performance to cPCA for any value of the contrastive tuning parameter. Interestingly, the simple and intuitive rPCA method, with k chosen by single knot spline with smallest MSE, achieved similar results to cPCA and UCA and outperformed cPCA++.

2.2 Subgroup discovery in protein expression data with multiple backgrounds

A more challenging problem for cPCA and it's variants is separating Down Syndrome and Control mice when the target involves Saline injected mice which received shock therapy. In figure 2 we show the results of PCA, cPCA++, rPCA, and UCA with a variety of backgrounds (shown in parenthesis): memantine injected mice with Down Syndrome that were shocked (Mem-C/S-Ts65Dn), memantine injected mice with Down Syndrome that were not shocked (Mem-S/C-Ts65Dn), saline injected mice with Down Syndrome that were not shocked (Saline-S/C-Ts65Dn), all of the aforementioned backgrounds stacked before forming their joint covariance matrix (Stack All), and aforementioned backgrounds treated as separate backgrounds simultaneously (Split All).

PCA and the single background condition UCA, cPCA++ and rPCA (k chosen with single knot spline with lowest MSE) with stacked background cannot distinguish between Down Syndrome and control mice. UCA with stacked background performs better at splitting the two groups compared with the other methods. However, UCA with split background yields almost complete separability compared with the stack background result, demonstrating the additional leverage when we treat each background separately. We note that this problem isn't easily separable by using cPCA++, rPCA, and cPCA with a variety of contrastive parameter and the results are generally worse than figure 2 (see figure 6 in the Appendix).

2.3 Faces

We illustrate the advantages of the multiple background capabilities of UCA using the Karolinska Directed Emotional Faces (KDEF) [6], which captured images of 7 emotions, 5 different perspectives, from 70 amateur actors in either a practice or final session. We first construct a target data set by combining Surprise, Angry, and Disgust emotions from the final session and attempt to uncover variation unique to Disgust. With PCA, eigenfaces represent faces that appear mostly angry. UCA with only Surprise practice faces as the background is dominated by surprise faces. UCA with only Angry practice faces as the background contains a mixture of surprise (2) and disgust (1,4,5) faces. When UCA has both Surprise and Angry in the background, we see that features of digust are highlighted in the first two eigenfaces However, when using the joint covariance of Surprise and Angry as the background, features of disgust are pushed until the 3rd eigenface. The 2nd eigenface here closely resembles that from the 3rd eigenface of Split, and the third eigenface closely resembles the 2nd eigenface from split. Using cPCA++ with the stacked background, I'm not sure what we get. it looks pretty bad.

3 4

3 Discussion

In many data analytics settings we are interested in removing uninteresting variation that contaminate the data of interest We propose an update to the original cPCA framework, UCA, by extending contrastive abilities to multiple background data. To do this, we first solved the tuning parameter problem which plague current methods. UCA does not require user input in selecting the best contrastive parameter, rather it works by maximizing the target data variability subject to keeping the variability in each background small using weak duality of Lagrangians. Furthermore, we reformulate the contrastive problem to avoid creating several $p \times p$ covariance matrices, enabling us to perform contrastive analysis in high-dimensions.

We show that UCA performs adequately as compared to cPCA, cPCA++, and rPCA in the easily separable case within the mouse data. We then presented a less separable case with the mouse data, in which cPCA, cPCA++, and rPCA fail to do an adequate job in separating Down Syndrome and control mice. We then showed how leveraging multiple backgrounds allows us to better separate Down Syndrome and control mice than each individual background dataset and simply stacking the individual datasets together. Lastly, we demonstrate how UCA can be used at highlighting emotions in eigenfaces and show that we can construct an example where stacking individual background data can cause cPCA to fail.

4 Method

In this section we introduce Unique Component Analysis (UCA) and show how we arrive at an optimal solution to the contrastive problem. Next, we will extend UCA to incorporate multiple background data sets. Finally, we address the high-dimensional limitation of cPCA and related methods in both single and multi-background scenarios using the Product SVD method to find our largest eigenvalue and corresponding eigenvector.

Let background data $X_{n_x \times p}$ and target data $Y_{n_y \times p}$ be two data sets of sample size n_x and n_y where $X, Y \in \mathbb{R}^p$ with respective $p \times p$ dimensional emperical background covariance matrix B and target covariance matrix A. The goal of cPCA is to find the directions of variability, $v \in \mathbb{R}^p$, which account for large variation in the target data Y and small variation in the background data X where $\|v\|_2 = 1$. Similar to PCA, cPCA measures variation along direction in target and background matrix with $v^T A v$ and $v^T B v$ respectively. For some tuning parameter $\lambda \in \mathbb{R}$, these directions can be isolated by identifying the eigenvector v such that:

$$\arg\max_{v \in \mathbb{R}^p} \left(v^T A v - \lambda v^T B v \right) = \arg\max_{v \in \mathbb{R}^p} v^T \left(A - \lambda B \right) v$$

For a given λ, the direction is simply the eigenvector of the weighted difference between covariance matrices where v maximize the variation in A while constraining variation in B. The tuning parameter λ represents how strongly background variation is minimized. When $\lambda = 0$, background variation isn't accounted for, thus the cPCA is reduced to PCA. But as λ increases, the relative background variation becomes more dominant, resulting in v_λ to focus on directions which minimize background variation rather than maximizing the target. For very large values of λ, v_λ finds the directions which map to the null space of the background data, which effectively is PCA of the target data after being projected onto the orthogonal complement of the background. Thus, the choice of λ can heavily influence the resulting eigenvector. Currently there are no rigorous ways in the choice of λ.

4.1 Unique Component Analysis: Single Background

We introduce the Unique Component Analysis (UCA) framework to identify the optimal contrastive parameter, λ, and the resulting directions of variation using the duality of lagrangians. By adding the additional constraints of $v^T v = 1$ and $v^T B v = 1$ we arrive at an objective method to choose λ. Specifically the primal optimization problem UCA solves is:

$$\max_{v \in \mathbb{R}^p} v^T A v \quad (1)$$
$$\text{subject to } \; v^T v = I, \;\; v^T B v \leq 1$$

We can solve 1 using weak duality of Lagrangians, $\mathcal{L}$:

$$\begin{aligned} \max_{\lambda \geq 0} \max_{v \in \mathbb{R}^P} \mathcal{L}(v, \lambda) &= \max_{\lambda \geq 0} \max_{v \in \mathbb{R}^P} \left(v^T A v - \lambda \left(v^T B v - 1 \right) \right) \\ &= \max_{\lambda \geq 0} \max_{v \in \mathbb{R}^P} \left(v^T (A - \lambda B) v + \lambda \right) \\ &= \max_{\lambda \geq 0} \left(\lambda_{\max} (A - \lambda B) + \lambda \right) \end{aligned} \quad (2)$$

where λ is the Lagrange multiplier associated with the constraint $v^T B v \leq 1$ and $\lambda_{max}(\cdot)$ is the largest eigenvalue associated with $A - \lambda B$. Although we have two constraints, we get $v^T v = I$ through Eigen decomposition. This is slightly different from cPCA++, where $v^T v$ is not necessarily orthonormal.

To solve for the largest Lagrange Multiplier, λ we follow the standard method of taking the derivative of the Lagrangian $\mathcal{L}$ with respect to the Lagrange Multiplier λ and set the derivative equal to zero:

$$\begin{aligned} \frac{\partial}{\partial \lambda} \max_{v \in \mathbb{R}^P} \mathcal{L}(v, \lambda) &= \frac{\partial}{\partial \lambda} \left(\lambda_{max} (A - \lambda B) + \lambda \right) \\ &= 1 - v_\lambda^T B v_\lambda = 0 \end{aligned} \quad (3)$$

where v_λ is the eigenvector associated with the largest eigenvalue for particular λ.

We use an iterative algorithm (L-BFGS-B) [5] to optimize between finding the Lagrange Multiplier λ, and finding the associated eigenvectors v_λ, stopping when equation 3 is satisfied.

With the optimal Lagrange Multiplier, eigenvector v_λ maximizes equation 1. Our constraints allows for a natural way to pick the contrastive parameter λ in cPCA. Similar to the constrastive parameter, the Lagrange Multiplier can't be less than zero, and allows us to use box constrained optimization methods.

4.2 Unique Component Analysis: Multiple Backgrounds

Similarly, for multiple backgrounds, again let the A be the target $p \times p$ covariance matrix and $B_1, \ldots, B_m$ be the m background $p \times p$ covariance matrices constructed from a $n_y \times p$ dimensional Y data matrix and corresponding $(n_{x_1} \times p), \ldots, (n_{x_m} \times p)$ dimensional $X_1, \ldots, X_m$ background data matrices.

One way to incorporate m background would be to use the framework in the single background setting and stack multiple background data sets before calculate the background

Algorithm 1: UCA Single Background

```
Input: centered target and background data $X$ and $Y$, initial $\lambda$, and the number of
  components $k$ ;
Compute the emperical covariance matrices:
  $$A = \frac{1}{n_y} Y^T Y, \quad B = \frac{1}{n_x} X^T X$$
  ;
Calculate $C_\lambda = A - \lambda B$ ;
Find $v_\lambda$ by performing eigen decomposition on $C_\lambda$;
while $1 - v_\lambda^T B v_\lambda \neq 0$ do
    Update $\lambda \leftarrow \lambda + \text{something}$ ;
    Calculate $C_\lambda = A - \lambda B$;
    Find $v_\lambda$ by performing eigenvalue decomposition on $C_\lambda$
end
compute $V_k \in \mathbb{R}^k$, spanned by the top $k$ eigenvectors of $C_\lambda$;
Return: $V_k$ ;
```

covariance matrix, creating a weighted average of the covariance matrices. This is sub-optimal since the background covariance matrix with the most samples may not accounts for the all the background noise, therefore potentially washing out background noise from the other smaller background data sets.

A more natural way to incorporate m background data sets in a more generalized framework would be to append the background matrices like so:

$$\max_{v \in \mathbb{R}^p} v^T \left(A - \sum_{j=1}^{m} \alpha_j B_j \right) v$$

However, a parameter sweep for each α_j corresponding to background data B_j would be computationally expensive. We can easily extend our single background covariance method to address the m background covariance scenario by adding additional constraints to Lagrangian, enforcing $v^T B_j v \leq 1$ for each m background covariance matrix. Specifically, the primal problem of our multiple background UCA is:

$$\begin{aligned} &\max_{v \in \mathbb{R}^p} v^T A v \\ \text{subject to} \quad &v^T v = I, \quad v^T B_1 v \leq 1, \ldots, v^T B_m v \leq 1 \end{aligned} \tag{4}$$

Again, using weak duality of Lagrangians, our objective function now can be written as:

$$\max_{\lambda_1, \ldots, \lambda_m \geq 0} \max_{v \in \mathbb{R}^P} \mathcal{L}\left(v, \lambda_1, \ldots, \lambda_m\right) = \max_{\lambda_1, \ldots, \lambda_m \geq 0} \max_{v \in \mathbb{R}^P} \left(v^T \left(A - \sum_{j=1}^{m} \lambda_j B_j \right) v + \sum_{j=1}^{m} \lambda_j \right) \tag{5}$$

For background $j \in 1, ..., m$, we can solve for the jth Lagrange Multiplier in equation 5 similarly by taking the partial derivative and setting to zero:

$$\begin{aligned}
\frac{\partial}{\partial \lambda_j} \max_{v \in \mathbb{R}^P} \mathcal{L}(v, \lambda_1, \cdots, \lambda_m) &= \frac{\partial}{\partial \lambda_j} \max_{v \in \mathbb{R}^p} \left(v^T \left(A - \sum_{j=1}^{m} \lambda_j B_j \right) v + \sum_{j=1}^{m} \lambda_j \right) \\
&= \frac{\partial}{\partial \lambda_j} \max_{v \in \mathbb{R}^p} \left(v^T (A - \lambda_j B_j) v + \lambda_j \right) \\
&= 1 - v_{\lambda_j}^T B_j v_{\lambda_j} = 0 \qquad (6)
\end{aligned}$$

To solve this multiple background dataset problem we propose a coordinate descent algorithm that utilizes the single background dataset L-BFGS-B [5] algorithm above. Suppose there are m background data sets, we are on the k^{th} iteration. Let $A^*_{(j)}$ be the variation in A after removing Background variation that is not B_j:

$$A^*_{(j)} = A - \sum_{i=1}^{j-1} \lambda_i^{(k+1)} B_i - \sum_{i=j+1}^{m} \lambda_i^{(k)} B_i$$

to solve for the each of the m Lagrange Multipliers we use Algorithm 2.

The advantage to the Lagrangian method in the multiple background setting is that rather than constructing an aggregated covariance matrix by stacking data, which arbitrarily weighs covariance matrices by their sample size, our method objectively constructs weights which satisfy the constraint $v^T B_j v \leq 1$ for each m background covariance matrix. If multiple backgrounds contain the same information, constructing the covariance matrix by stacking data will change the covariance by a factor less than 1, e.g. if two backgrounds, the covariance changes by $\frac{2}{2n-1}$. If only one covariance matrix will constrain the Lagrangian, the corresponding Lagrange Multiplier will be non-zero and Lagrange Multipliers corresponding to all other redundant constraints on the Lagrangian are set to zero.

4.3 Extension to High Dimensional Data:

To extend our method to high-dimensional data, we avoid the traditional Eigen decomposition method of finding eigenvalue/eigenvectors of the covariance matrix. Just like how singular-value decomposition (SVD) of the data matrix can be used to find the top eigenvalue/eigenvectors of it's covariance, we introduce the Product SVD method to exploit the structure of how the contrastive covariance matrices are formed and employ SVD of a product of matrices to quickly find our top eigenvalue. Breaking the problem down in this way considerably speeds up our L-BFGS-B algorithm, and allows us to run UCA even in the high-dimensional setting.

For a single background, let the $p \times p$ covariance matrices A be the target covariance matrix and B be the background covariance matrix. A and B are formed from corresponding centered $n_y \times p$ target data matrix Y, and centered $n_x \times p$ background data matrix X. We can write the difference of the covariance matrices $A - \lambda B$ as a product of a $p \times (n_y + n_x)$ dimensional left matrix , L, and a $(n_y + n_x) \times p$ dimensional right matrix, R as seen in

Algorithm 2: Coordinate Descent Algorithm for UCA with Multiple Background

```
Inputs: $m$ Centered Background data $X_1, \cdots, X_m$, Centered Target Data $Y$, $k$ number of components;
Initialize $\lambda_1^{(0)}, \cdots, \lambda_m^{(0)}, \epsilon$;
Compute the emperical covariance matrices:
    $$A = \frac{1}{n_y} Y^T Y, \;\; B_1 = \frac{1}{n_{x_1}} X_1^T X_1, \;\; \cdots, B_m = \frac{1}{n_{x_m}} X_m^T X_m$$
while $l^{(k+1)} - l^{(k)} > \epsilon$ do
    for $j = 1 : m$ do
        Let $A$ in Algorithm 1 be $A^*_{(j)}$ and $B = B_j$. Solve for $\lambda_j^{(k+1)}$ via Algorithm 1, corresponding to the largest eigenvalue of
            $$C_{\lambda_j} = A^*_{(j)} - \lambda_j^{(k+1)} B_j$$
    After iterating through $m$ Lagrange Multipliers, calculate value of Lagrangian
        $$l^{(k+1)} = \lambda_{max} \left( A - \sum_{j=1}^{m} \lambda_j^{(k+1)} B_j \right) + \sum_{j=1}^{m} \lambda_j^{(k+1)}$$
Compute
    $$C_{\lambda_1, \cdots, \lambda_m} = A - \sum_{j=1}^{m} \lambda_j B_j$$
Find $V_k \in \mathbb{R}^k$, spanned by the top $k$ eigenvectors of $C_{\lambda_1, \cdots, \lambda_k}$;
Return: $V_k$
```

equation 7:

$$\begin{aligned} C_\lambda &= A - \lambda B \\ &= \frac{1}{n_y} Y^T Y - \frac{\lambda}{n_x} X^T X \\ &= \underbrace{\left[\frac{1}{\sqrt{n_y}} Y^T, -\frac{\lambda}{\sqrt{n_x}} X^T \right]}_{L} \underbrace{\begin{bmatrix} \frac{1}{\sqrt{n_y}} Y \\ \frac{1}{\sqrt{n_x}} X \end{bmatrix}}_{R} \end{aligned} \tag{7}$$

With this formulation, we can follow the steps in Golub et al. [12], and find the singular values and vectors of C_λ using a sequence of SVD and QR decompositions operating on the left and right matrices. Since singular values and eigenvalues coincide in square matrices, we use the singular vectors, U, to find the largest eigenvalues by sorting the diagonal of $ULRU^T$. We describe the Product SVD Method in algorithm 3, which can directly replace the eigen decomposition in the single background UCA (Algorithm 1).

```
Algorithm 3: Product SVD Method to calculate the largest Eigenvalue of $C_\lambda$
    Input: Centered background matrix $X$, centered target matrix $Y$, and $\lambda$;
  1 Construct $L = \left[\frac{1}{\sqrt{n_y}}Y^T, -\frac{\lambda}{\sqrt{n_x}}X^T\right]$, $R = \begin{bmatrix}\frac{1}{\sqrt{n_y}}Y \\ \frac{1}{\sqrt{n_x}}X\end{bmatrix}$;
  2 SVD the right matrix, $R = U_R S_R V_R^T$ ;
  3 QR decompose $LU_R$ into $Q_{LU_R} R_{LU_R}$ ;
  4 SVD the product of $R_{LU_R}$ (step 3) and $S_R$ (step 2), $R_{LU_R} S_R = EDF^T$ ;
  5 The singular vectors of $C_\lambda$ is the product of $Q$ (step 3) and $E$ (step 4),
     $U_{C_\lambda} = Q_{LU_R} E$ ;
  6 Find the largest eigenvalues of $C_\lambda$ by sorting the diagonal of $D_{C_\lambda}$, where
     $D_{C_\lambda} = U_{C_\lambda} C_\lambda U_{C_\lambda}^T$ ;
    Output: $\lambda_{\max}(C_\lambda)$
```

Similarly, for multiple backgrounds, again let the A be the target $p \times p$ covariance matrix and $B_1, \cdots, B_m$ be the m background $p \times p$ covariance matrices constructed from a $n_y \times p$ dimensional Y data matrix and corresponding $(n_{x_1} \times p), \cdots, (n_{x_m} \times p)$ dimensional $X_1, \cdots X_m$ background data matrices.

We can construct $C_{\lambda_1,\cdots,\lambda_m} = A - \sum_{j=1}^m \lambda_j B_j$ analogously by appending the additional data sets to the left and right matrices:

$$\begin{aligned} C_{\lambda_1,\cdots,\lambda_m} &= A - \sum_{j=1}^m \lambda_j B_j \\ &= \frac{1}{n_y} Y^T Y - \sum_{j=1}^m \frac{\lambda_j}{n_{x_j}} X_j^T X_j \\ &= \underbrace{\left[\frac{1}{\sqrt{n_y}} Y^T, -\frac{\lambda_1}{\sqrt{n_{x_1}}} X_1^T, \cdots, -\frac{\lambda_m}{\sqrt{n_{x_m}}} X_m^T\right]}_{L} \underbrace{\begin{bmatrix} \frac{1}{\sqrt{n_y}} Y \\ \frac{1}{\sqrt{n_{x_1}}} X_1 \\ \vdots \\ \frac{1}{\sqrt{n_{x_m}}} X_m \end{bmatrix}}_{R} \end{aligned} \tag{8}$$

Using the Coordinate Descent (algorithm 2) to solve for each λ_j prescribed above, we can substitute the Eigen decomposition of the covariance matrix, C_{λ_j} with the Product SVD method (algorithm 3) using L and R defined in equation 8.

The Product SVD method is advantageous in high-dimensions because it never explicitly operates on the entire $p \times p$ covariance matrix. Not only is our method more memory efficient, scaling with $n \times p$ bytes rather than p^2 bytes, but our method is also computationally more efficient at finding the largest eigenvalue/eigenvector as operating SVD and QR decomposition on either the left or right matrices is faster than directly operating Eigen decomposition on the covariance matrix. In the single background scenario, λ only appears in L. Thus since, R doesn't change, we can pre-compute the SVD of R, and only update L when solving for $\lambda_{\max}$. Similarly, in the multi-background scenario, rather than modifying

A^* in the original Coordinate Descent algorithm, this method allows us to only modify the $j+1$ element of the left data matrix, L. The SVD of the right matrix, R, only needs to be computed once in our coordinate descent. Furthermore, at each step within the coordinate descent only step 3 and 4, the SVD and QR, are computed, and are done on matrices with dimensions much smaller than $p \times p$.

To demonstrate the speed improvements of the Product SVD method compared to the current fastest implementations of Eigen decomposition in high-dimensions, we conduct a simulation study with 25 sample $100 \times p$ target and background data matrices generated from a standard Normal distribution with p varying from 1.000 to 10,000 in steps of 1,000. To ensure a fair comparison, we leverage C++ in both implementations using a custom RcppArmadillo [8] function for the Product SVD method and the RSpectra package (0.16-0) [18] for the Eigen decomposition method; RSpectra being a package designed for large-scale Eigen decompositions based off the C++ Spectra library. We use the microbenchmark package [16] to ensure accurate timings. Our benchmark does not take into account the additional cost of forming the $p \times p$ covariance matrices, which would only exacerbate the difference between the two methods in real world applications.

Figure 6 show box plots of time (in seconds) versus the dimension, p, colored by method, summarizes the results of the simulation study. As dimension p increases, the computational time of our Product SVD method increases much slower than the current Eigen decomposition implementations. It should be mentioned that for $p < 1000$ the Product SVD method is slower due to overhead of additional computation on small matrices. In general, for low-dimensional settings where $n \geq p$, the Product SVD will be negligibly slower because of the additional QR, SVD, matrix products, and sort computations.

All computations in this paper were done with R 3.6.3 [19] on an AMD Ryzen 1700X 3.7 gHz processor and 64GB 3000 mhz DDR4 RAM, utilizing the Intel MKL libraries.

4.4 Code Availability

We have released a R implementation of UCA on GitHub. We have implemented both Product SVD on the data matrix and Eigen decomposition on the contrastive covariance matrix and allow the background to take either a single background or a list of backgrounds. The GitHub repository also includes R markdown and datasets that reproduce most of the figures in this paper and in the Supplementary.

4.5 Data availability

Datasets that have been used to evaluate UCA in this paper are publicly available from the authors of the original studies. You can download the mouse protein expression dataset from the UCI Machine Learning Repository [13] and the Karolinska Directed Emotional Faces (KDEF) from kdef website [6].

References

[1] Abubakar Abid, Martin J. Zhang, Vivek K. Bagaria, and James Zou. Exploring patterns enriched in a dataset with contrastive principal component analysis. *Nature Communications*, 9:2134, 05 2018.

[2] Abubakar Abid and James Zou. Contrastive variational autoencoder enhances salient features, 2019.

[3] Md. Mahiuddin Ahmed, A. Ranjitha Dhanasekaran, Aaron Block, Suhong Tong, Alberto C. S. Costa, Melissa Stasko, and Katheleen J. Gardiner. Protein dynamics associated with failed and rescued learning in the ts65dn mouse model of down syndrome. *PLOS ONE*, 10(3):1–25, 03 2015.

[4] Philippe Boileau, Nima S Hejazi, and Sandrine Dudoit. Exploring high-dimensional biological data with sparse contrastive principal component analysis. *Bioinformatics*, 36(11):3422–3430, 03 2020.

[5] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[6] Manuel G. Calvo and Daniel Lundqvist. Facial expressions of emotion (kdef): Identification under different display-duration conditions. *Behavior Research Methods*, 40(1):109–115, Feb 2008.

[7] Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287 – 314, 1994. Higher Order Statistics.

[8] Dirk Eddelbuettel and Conrad Sanderson. Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063, March 2014.

[9] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23(9):881–890, 1974.

[10] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[11] T. Fujiwara, O. Kwon, and K. Ma. Supporting analysis of dimensionality reduction results with contrastive learning. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):45–55, 2020.

[12] Gene Golub, Knut Solna, and Paul Van Dooren. Computing the svd of a general matrix product/quotient. *SIAM Journal on Matrix Analysis and Applications*, 22(1):1–19, 2000.

[13] Clara Higuera, Katheleen J. Gardiner, and Krzysztof J. Cios. Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLOS ONE*, 10(6):1–28, 06 2015.

[14] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, Oct 1999.

[15] Michael Lenz, Franz-Josef Müller, Martin Zenke, and Andreas Schuppert. Principal components analysis and the reported low intrinsic dimensionality of gene expression microarray data. *Scientific Reports*, 6(1):25696, Jun 2016.

[16] Olaf Mersmann. *microbenchmark: Accurate Timing Functions*, 2019. R package version 1.4-7.

[17] Lan Huong Nguyen and Susan Holmes. Ten quick tips for effective dimensionality reduction. *PLOS Computational Biology*, 15(6):1–19, 06 2019.

[18] Yixuan Qiu and Jiali Mei. *RSpectra: Solvers for Large-Scale Eigenvalue and SVD Problems*, 2019. R package version 0.16-0.

[19] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[20] Ronald Salloum and C.-C. Jay Kuo. Efficient image splicing localization via contrastive feature extraction. *CoRR*, abs/1901.07172, 2019.

[21] Kristen A Severson, Soumya Ghosh, and Kenney Ng. Unsupervised learning with contrastive latent variable models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4862–4869, 2019.

5 Appendix

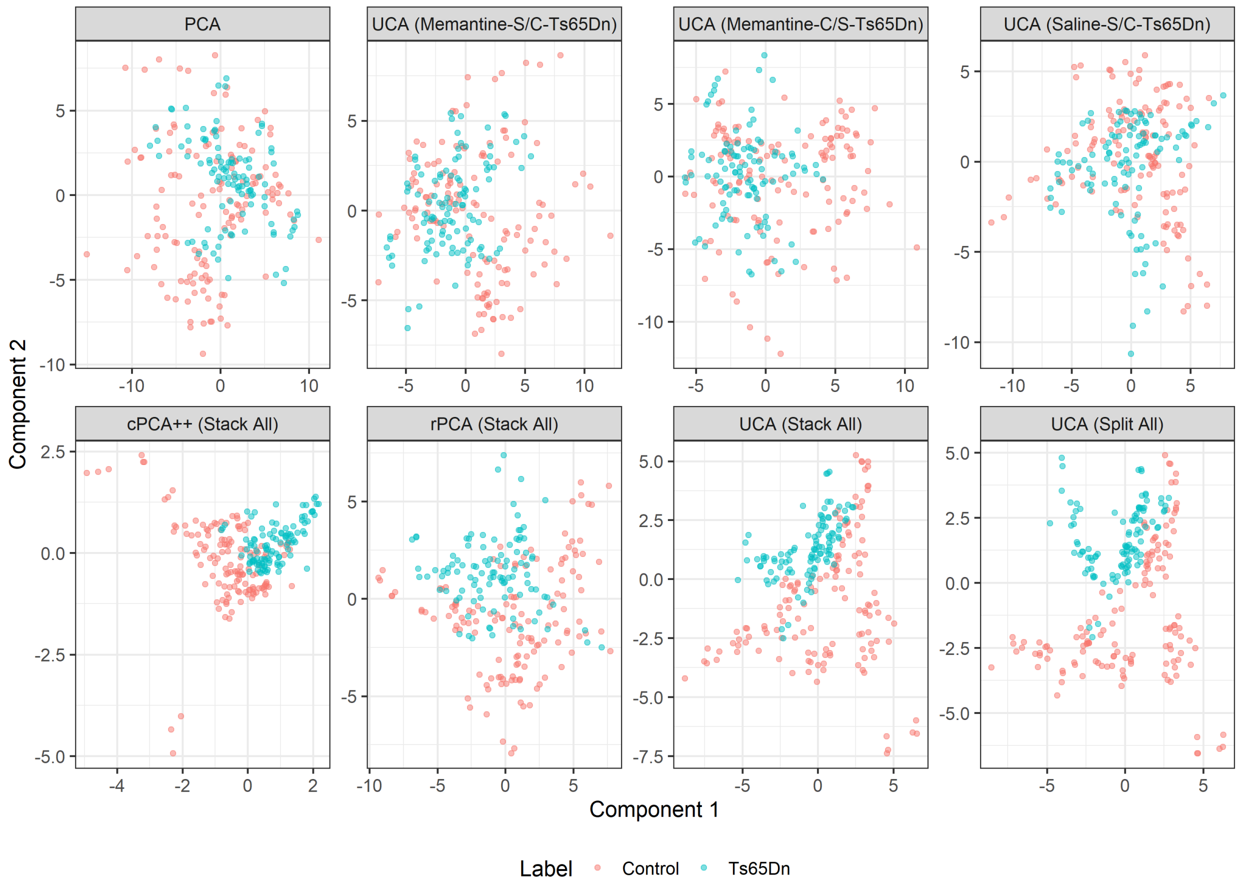

Figure 2: Mouse Protein Expression : Separability of saline injected mice which were shocked. Separating Down Syndrome and control mice is more difficult in this case when only using a single background dataset. UCA with every combination of background (in parenthesis) represents the best case seario for cPCA, and cannot separate between Down Syndrome and control mice. However, when combining information from all backgrounds via stacking, UCA improves the separation. Note that cPCA++ and rPCA both have difficulty even when leveraging the same stacked background. When treating each background separately, UCA improves on the stacking results.

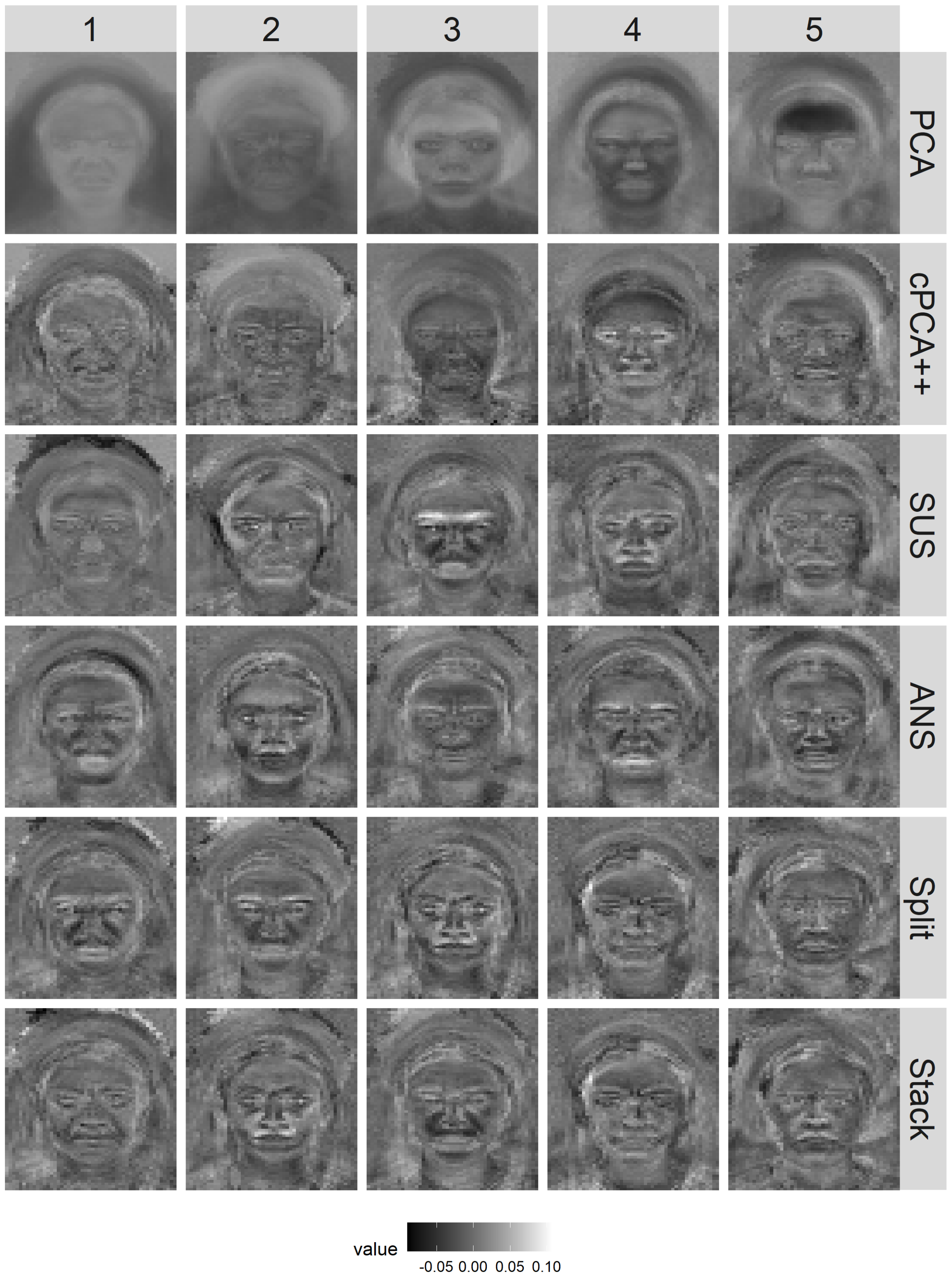

Figure 3: First Five Eigenfaces of Surprise, Angry, and Disgust emotions from the Female Actors from Karolinska Directed Emotional Faces (KDEF) [6]. Target data is sus/ans/dis

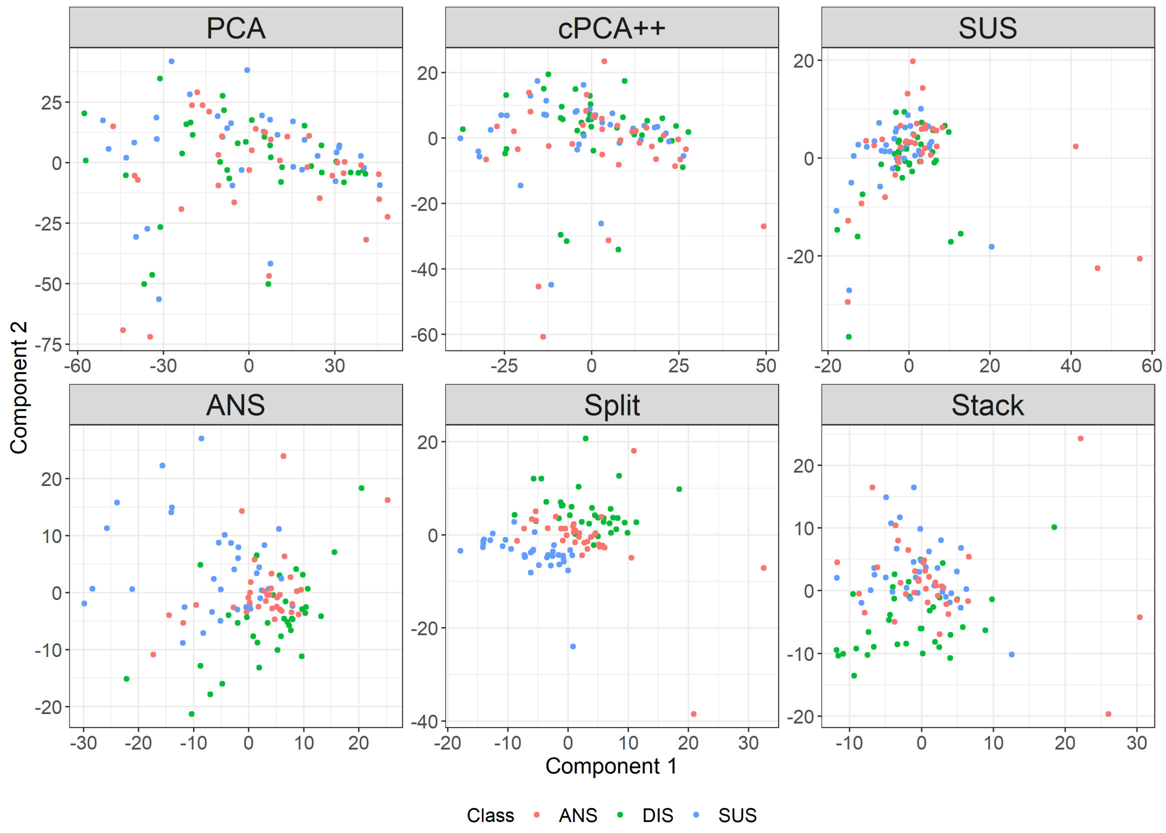

Figure 4: Eigenfaces projected onto the first two components found by PCA, cPCA++, UCA using Surprise (SUS) as the background, UCA using Angry (ANS) faces as the background, UCA using both Surprise and Angry as the background treated separately (Split), UCA using the joint covariance of Surprise and Split (Stack) as the background

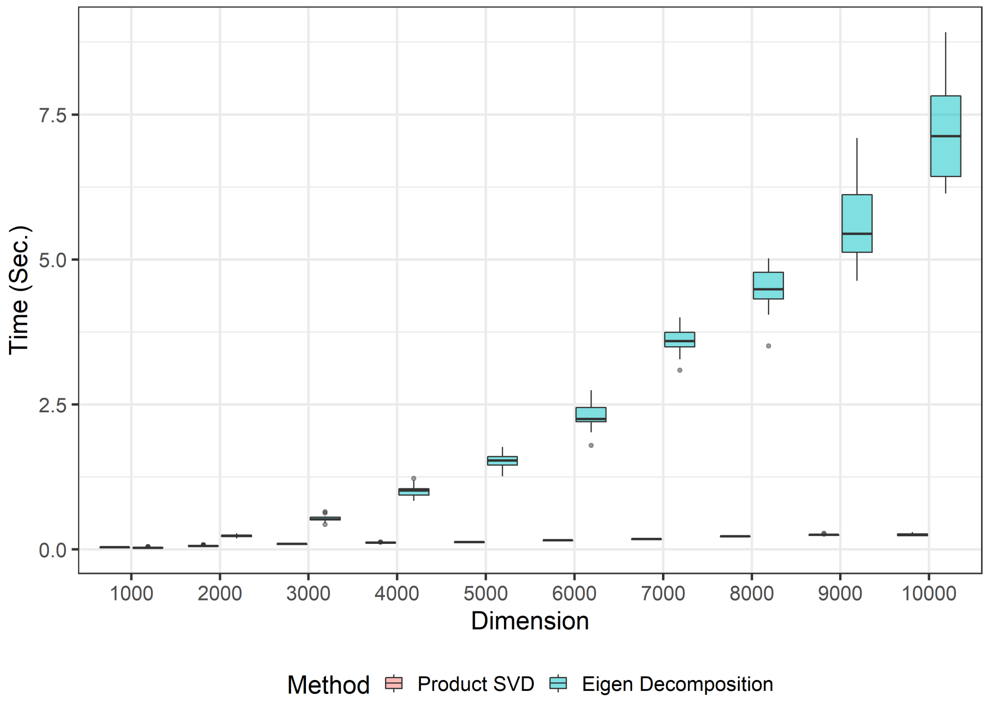

Figure 5: UCA and cPCA implementation using Product SVD vs. Eigen decomposition for high-dimensional data: 25 random $100 \times p$ target and background matrices are generated from a standard normal distribution and where p, the dimension varied from 1,000 to 10,000 in steps of 1,000. Box plots of time (in seconds) is plotted for both Eigen decomposition and the product SVD method. For small p, there is a negligible difference between the methods. However, as dimension p increases, the Product SVD is significantly faster.

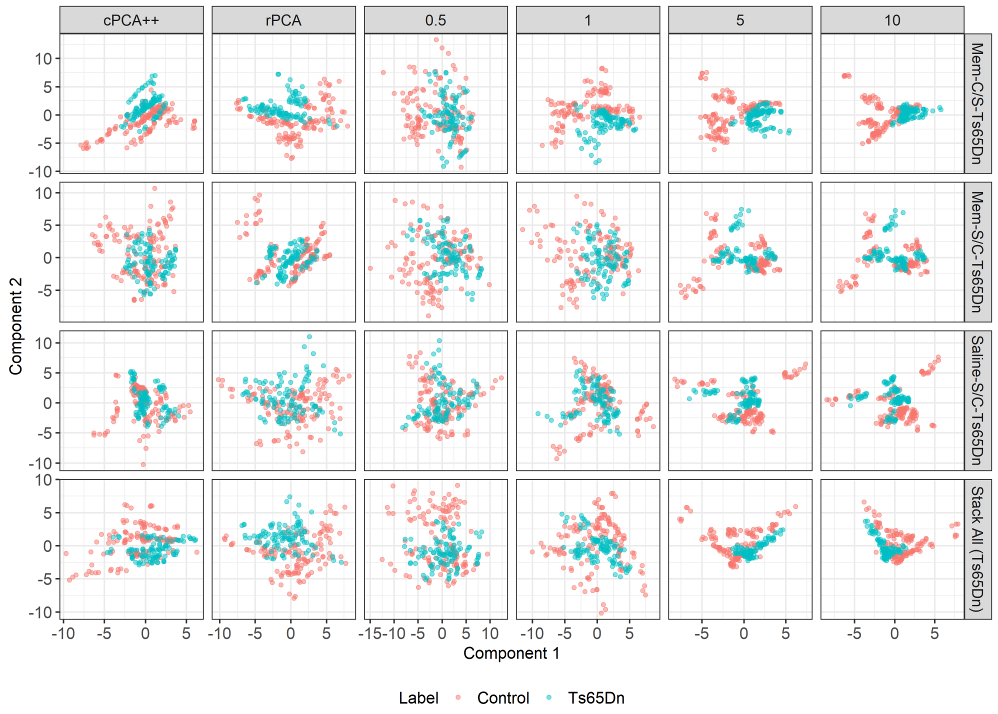

Figure 6: Mouse Protein Expression Dataset

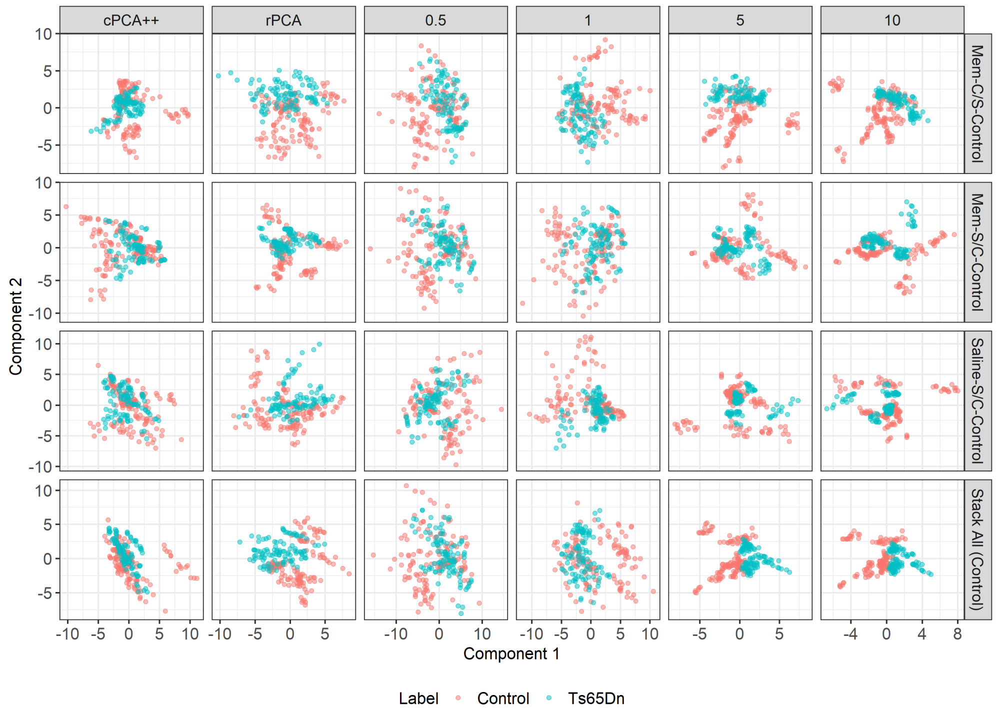

Figure 7: Mouse Protein Expression Dataset

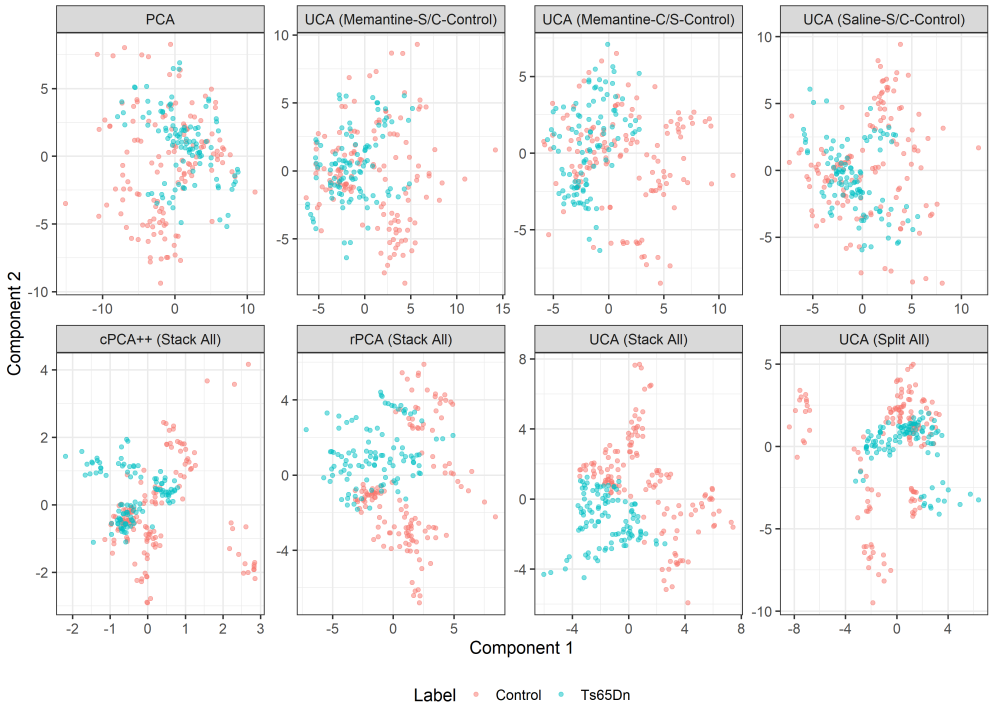

Figure 8: Mouse Protein Expression Dataset