1 Intro

1.1 Notation

In the following notes, $\mathbf{X}_i \in \mathbb{R}^{n \times p_i}$ will refer to the observation matrix for the i-th 'view' of some assumed underlying phenomenon, where n is the number of observations and p_i is the ambient dimension of the i-th view. $\mathbf{\Phi}_i \in \mathbb{R}^{p_i \times k}$ will refer to some linear transform of $\mathbf{X}_i$ into a k-dimensional space. The $k \times k$ identity matrix is denoted $\mathbf{I}_k$.

1.2 Canonical Correlation Analysis

1.3 Generalized Eigenvalue Problems

2 Optimization

2.1 Optimization Problem Formulation

2.1.1 Classical Two-View CCA

This problem formulation is limited to two views, and it does not take advantage of the full power of the corresponding generalized eigenvalue problem formulation.

$$\begin{aligned} \textbf{minimize} \quad & \|\mathbf{X}_1\mathbf{\Phi}_1 - \mathbf{X}_2\mathbf{\Phi}_2\|_F^2 \\ \textbf{subject to} \quad & \mathbf{\Phi}_1^\top \mathbf{X}_1^\top \mathbf{X}_1 \mathbf{\Phi}_1 = \mathbf{I}_k \\ & \mathbf{\Phi}_2^\top \mathbf{X}_2^\top \mathbf{X}_2 \mathbf{\Phi}_2 = \mathbf{I}_k, \end{aligned}$$

where $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ are the optimization variables.

2.1.2 Complete Multiview Graph

This problem formulation is capable of considering an arbitrary, finite number of views of the data, and it assumes the existence of non-spurious correlation between each pair of views.

$$\begin{aligned} \textbf{minimize} \quad & \sum_{i=1}^{m} \sum_{j=i+1}^{m} \|\mathbf{X}_i\mathbf{\Phi}_i - \mathbf{X}_j\mathbf{\Phi}_j\|_F^2 \\ \textbf{subject to} \quad & \mathbf{\Phi}_i^\top \mathbf{X}_i^\top \mathbf{X}_i \mathbf{\Phi}_i = \mathbf{I}_k \;\; i = 1, \ldots, m, \end{aligned}$$

where $\mathbf{\Phi}_i$ are the optimization variables.

2.1.3 Approximation of Complete Graph

This problem formulation simplifies the model in section 2.1.2 by replacing the exhaustive pairwise comparisons with m comparisons to a global auxiliary variable. The idea is that the different views are related to each other via $\mathbf{\Psi}$ without having to enumerate $O(m)$ penalties. However, doesn't provide much benefit in terms of the cost of computing the gradient for each $\mathbf{\Phi}_i$. Also, it is not obvious to me how to reformulate this as a GEP. I am not even sure that its possible. On the other hand, $\mathbf{\Psi}$ may have a meaningful and interesting interpretation as a summarizer.

$$\begin{aligned} \textbf{minimize} \quad & \sum_{i=1}^{m} \|\mathbf{X}_i\mathbf{\Phi}_i - \mathbf{\Psi}\|_F^2 \\ \textbf{subject to} \quad & \mathbf{\Phi}_i^\top \mathbf{X}_i^\top \mathbf{X}_i \mathbf{\Phi}_i = \mathbf{I}_k \;\; i = 1, \ldots, m, \end{aligned}$$

where $\mathbf{\Phi}_i$ and $\mathbf{\Psi}$ are the optimization variables.

2.1.4 Arbitrary Dependency Graph

This optimization problem is a good fit for a scenario in which the user has prior knowledge of which views should be correlated with each other. If this prior knowledge is close enough to correct, then using this formulation may reduce the risk of noise-dominated updates from high distances between uncorrelated views as well as discovery of spurious correlations.

$$\begin{aligned} \textbf{minimize} \quad & \sum_{(i,j)\in E} \|\mathbf{X}_i\mathbf{\Phi}_i - \mathbf{X}_j\mathbf{\Phi}_j\|_F^2 \\ \textbf{subject to} \quad & \mathbf{\Phi}_i^\top \mathbf{X}_i^\top \mathbf{X}_i \mathbf{\Phi}_i = \mathbf{I}_k \;\; i = 1, \ldots, m, \end{aligned}$$

where E is the set of edges in the dependency graph between the views, and $\mathbf{\Phi}_i$ are the optimization variables. Note that we recover the optimization problem from section 2.1.2 by setting $E = \{(i,j) : i \in \{1, \ldots, m-1\}, j \in \{i+1, \ldots, m\}\}$.

2.2 Constraints

The classical formulation of generalized eigenvalue problems including CCA is to have quadratic equality constraints $\mathbf{X}^\top \mathbf{A} \mathbf{X} = \mathbf{I}$ on the parameters $\mathbf{X}$ for some quadratic form $\mathbf{A}$. The purpose of these constraints is to force the parameters to counteract/reflect the scale of the data so that the model is robust to each view having dramatically different scale.

2.2.1 Issues with Equality Constraints

For both the classical and projected gradient algorithms for estimating CCA parameters, to satisfy the equality constraints, we are forced to orthogonalize some basis either via Gram-Schmidt orthogonalization or matrix square-roots. For batch classical and projected gradient optimization techniques, this causes scaling issues for both computation and storage. For the stochastic projected gradient algorithm, the projections onto a non-convex feasible region (a quadratic surface) result in lack of convergence guarantees and very likely slower local convergence when this is attainable. Additionally, if we want to use the transformations as dimensionality reduction on unseen data, we might consider satisfaction of equality constraints to be overfitting.

If we want to improve convergence guarantees and reduce computation, but still account for and be robust to differences in scale between the two data views, it may be benificial to penalized the parameters for straying from the equality constraints rather than requiring exact constraint satisfaction on a non-convex (possibly not even locally convex) quadratic surface feasible region. In fact, when using the stochastic projected gradient algorithm, violating the equality constraints by a little bit is unavoidable because we use minibatches for projecting the parameters, but the feasible region is defined in terms of the full dataset.

2.2.2 Potential Proximal Alternatives

If we define some penality function $R(\mathbf{X}) = \left\|\mathbf{X}^\top \mathbf{A} \mathbf{X} - \mathbf{I}\right\|_F^2$ and use this penalty function as a proximal function, it may result in more efficient computation and better convergence properties for parameter estimation and better generalization ability for the dimensionality reduction defined in terms of the parameters. I am somewhat certain that this function, although non-convex, has no spurious critical points.

Using $R(\mathbf{X})$ as a proximal function is somewhat problematic because of the lack of closed-form solution. However, since we are relaxing the constraints to a soft penalty anyway, it may be acceptable to take a small, fixed number of gradient steps to approximate the proximal operator's solution at each round.

Also, I need to read further to see if I can apply this and if it will be helpful, but (I think) according to Parikh et al. (2014), in some cases, for a proximal operator on a matrix-valued parameter, there is some equivalent proximal operator of the parameter's singular values.

2.3 Algorithms

2.3.1 Why gradients?

Gradient-based updates allow us to easily derive stochastic and mini-batch algorithms and make incremental updates where we have complete control over the scale and influence of the incremental updates. These abilities facilitate a filter-like perspective of CCA and similar problems, and since we can control the scale/influence of each incremental update, we can control the rate of change of each parameter as a function of many quantities, including the update rates of other parameters. We can use tools from online learning and online optimization to approach these problems efficiently and with strong theoretical guarantees.

2.3.2 Stochastic m-View AppGrad

The algorithm in this section is derived from a similar algorithm for classical two-view CCA given in Ma et al. (2015).

Algorithm 1 Stochastic m-view, rank-k AppGrad

Input: batch size n, step sizes $\eta_{1t}, \ldots, \eta_{mt}$, max iter T, num components k
Output: Canonical bases $(\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_m)$
Initialization: $t = 0$, $\tilde{\mathbf{\Phi}}_1^{(0)} \in \mathbb{R}^{p_1 \times k}, \ldots, \tilde{\mathbf{\Phi}}_m^{(0)} \in \mathbb{R}^{p_m \times k}$ each with entries sampled invidually from $\mathcal{N}(0, 1)$, $\mathbf{\Phi}_j^{(0)} = \mathbf{0}$ for $j = 1, \ldots, m$
while $t < T$ **and** not converged **do**
  Receive minibatch $\mathbf{X}_1^{(t)} \in \mathbb{R}^{n \times p_1}, \ldots, \mathbf{X}_m^{(t)} \in \mathbb{R}^{n \times p_m}$
  for $i = 1, \ldots, m$ **do**
    $\tilde{\mathbf{\Phi}}_j^{(t+1)} \leftarrow \tilde{\mathbf{\Phi}}_i^{(t)} - \eta_{it} \left(\mathbf{X}_i^{(t)}\right)^\top \left((m-1)\mathbf{X}_i^{(t)} \tilde{\mathbf{\Phi}}_i^{(t)} - \sum_{j \neq i} \mathbf{X}_j^{(t)} \mathbf{\Phi}_j^{(t)}\right)$
    SVD: $\left(\tilde{\mathbf{\Phi}}_i^{(t+1)}\right)^\top \left(\frac{1}{n} \mathbf{X}_i{}^\top \mathbf{X}_i\right) \tilde{\mathbf{\Phi}}_i^{(t+1)} = \mathbf{U}_i^\top \mathbf{D}_i \mathbf{U}_i$
    $\mathbf{\Phi}_i^{(t+1)} \leftarrow \tilde{\mathbf{\Phi}}_i^{(t+1)} \mathbf{U}_i^\top \mathbf{D}_i^{-\frac{1}{2}} \mathbf{U}_i$
  end for
  $t \leftarrow t + 1$
end while

Minibatch Management: It should be noted that the minibatches are managed by fixed-length queues to deal with different sampling rates for different views, so the pseudocode for Algorithm 1 is not entirely faithful to my implementation. I would like to eventually give more rigorous treatment of this later, possibly using some Markov chain theory.
Parameter Updates: It should be noted that I am not making naïve gradient updates. This is overlooked in the pseudocode for clarity and relevance reasons. I am actually applying AdaGrad scaling with optional dual averaging and shrinkage-and-thresholding to the singular values of the parameter matrix and gradient matrix (Duchi et al., 2011).

2.3.3 GenELinK and CCALin

Algorithms 2 and 3 in this section are taken directly from Ge et al. (2016). Algorithms 4 and 5 are inspired by Ge et al. (2016).

2.3.4 GenELinK and CCALin Filters

For transforming the CCA algorithm with GenELinK subroutine into an adaptive filtering algorithm, we need to ensure that two criteria are satisfied:

- There is an updated set of canonical bases available for filtering unseen datapoints after each minibatch is processed.
- State from previous parameter updates are maintained across minibatches.

Algorithm 2 GenELinK

Input: T, k, symmetric matrix $\mathbf{A}$, PSD matrix $\mathbf{B}$, subroutine $\mathrm{GS}_{\mathbf{B}}(\cdot)$ that performs Gram-Schmidt process, with inner product $\langle\cdot,\cdot\rangle_{\mathbf{B}}$
Output: top k eigen-space $\mathbf{W} \in \mathbb{R}^{d\times k}$
$\tilde{\mathbf{W}}^{(0)} \leftarrow$ random $d \times k$ matrix with each entry i.i.d. from $\mathcal{N}(0,1)$
$\mathbf{W}^{(0)} \leftarrow \mathrm{GS}_{\mathbf{B}}\left(\tilde{\mathbf{W}}^{(0)}\right)$
for $t = 0, \ldots, T-1$ **do**
  $\mathbf{\Gamma}^{(t)} \leftarrow \left(\left(\mathbf{W}^{(t)}\right)^\top \mathbf{B}\mathbf{W}^{(t)}\right)^{-1}\left(\left(\mathbf{W}^{(t)}\right)^\top \mathbf{A}\mathbf{W}^{(t)}\right)$
  $\tilde{\mathbf{W}}^{(t+1)} \leftarrow \arg\min_{\mathbf{W}} \mathrm{tr}\left(\frac{1}{2}\mathbf{W}^\top\mathbf{B}\mathbf{W} - \mathbf{W}^\top\mathbf{A}\mathbf{W}\right)$
  $\left\{\text{Use an optimization subroutine with initialization } \mathbf{W}^{(t)}\mathbf{\Gamma}^{(t)}\right\}$
  $\mathbf{W}^{(t+1)} \leftarrow \mathrm{GS}_{\mathbf{B}}\left(\tilde{\mathbf{W}}^{(t+1)}\right)$
end for
return $\mathbf{W}^{(T)}$

Algorithm 3 CCALin

Input: T, k, data matrix $\mathbf{X} \in \mathbb{R}^{n\times d_1}$, $\mathbf{Y} \in \mathbb{R}^{n\times d_2}$, subroutine $\mathrm{GS}_{\mathbf{M}}(\cdot)$ that performs Gram-Schmidt process, with inner product $\langle\cdot,\cdot\rangle_{\mathbf{M}}$ where $\mathbf{M}$ is any PSD matrix
Output: top k canonical subspace $\mathbf{W}_x \in \mathbb{R}^{d_1\times k}$, $\mathbf{W}_y \in \mathbb{R}^{d_2\times k}$
$\mathbf{S}_{xx} \leftarrow \mathbf{X}^\top\mathbf{X}/n$, $\mathbf{S}_{yy} \leftarrow \mathbf{Y}^\top\mathbf{Y}/n$, $\mathbf{S}_{xy} \leftarrow \mathbf{X}^\top\mathbf{Y}/n$
$\mathbf{A} \leftarrow \begin{pmatrix} 0 & \mathbf{S}_{xy} \\ \mathbf{S}_{xy}^\top & 0 \end{pmatrix}$, $\mathbf{B} \leftarrow \begin{pmatrix} \mathbf{S}_{xx} & 0 \\ 0 & \mathbf{S}_{yy} \end{pmatrix}$
$\begin{pmatrix} \bar{\mathbf{W}}_x \in \mathbb{R}^{d_1\times k} \\ \bar{\mathbf{W}}_y \in \mathbb{R}^{d_2\times k} \end{pmatrix} \leftarrow \text{GenELinK}(\mathbf{A}, \mathbf{B}, T, k)$
$\mathbf{U} \leftarrow 2k \times k$ random Gaussian matrix
$\tilde{\mathbf{W}}_x \leftarrow \bar{\mathbf{W}}_x\mathbf{U}$, $\tilde{\mathbf{W}}_y \leftarrow \bar{\mathbf{W}}_y\mathbf{U}$
$\mathbf{W}_x \leftarrow \mathrm{GS}_{\mathbf{S}_{xx}}\left(\tilde{\mathbf{W}}_x\right)$, $\mathbf{W}_y \leftarrow \mathrm{GS}_{\mathbf{S}_{yy}}\left(\tilde{\mathbf{W}}_y\right)$

Algorithm 4 GenELinK Filter

Initialization: $\tilde{\mathbf{W}}^{(0)} \leftarrow$ random $d \times k$ matrix with each entry i.i.d. from $\mathcal{N}(0,1)$, $t \leftarrow 0$
while True **do**
  Receive symmetric matrix $\mathbf{A}^{(t)}$, PSD matrix $\mathbf{B}^{(t)}$, subroutine $\mathrm{GS}_{\mathbf{B}^{(t)}}(\cdot)$ that performs Gram-Schmidt process, with inner product $\langle\cdot,\cdot\rangle_{\mathbf{B}^{(t)}}$
  $\mathbf{W}^{(t)} \leftarrow \mathrm{GS}_{\mathbf{B}^{(t)}}\left(\tilde{\mathbf{W}}^{(t)}\right)$
  $\mathbf{\Gamma}^{(t)} \leftarrow \left(\left(\mathbf{W}^{(t)}\right)^\top \mathbf{B}_t\mathbf{W}^{(t)}\right)^{-1}\left(\left(\mathbf{W}^{(t)}\right)^\top \mathbf{A}_t\mathbf{W}^{(t)}\right)$
  $\tilde{\mathbf{W}}^{(t+1)} \leftarrow \arg\min_{\mathbf{W}} \mathrm{tr}\left(\frac{1}{2}\mathbf{W}^\top\mathbf{B}^{(t)}\mathbf{W} - \mathbf{W}^\top\mathbf{A}^{(t)}\mathbf{W}\right)$
  $\left\{\text{Use an optimization subroutine with initialization } \mathbf{W}^{(t)}\mathbf{\Gamma}^{(t)}\right\}$
  $\mathbf{W}^{(t+1)} \leftarrow \mathrm{GS}_{\mathbf{B}^{(t)}}\left(\tilde{\mathbf{W}}^{(t+1)}\right)$
  yield $\mathbf{W}^{(t+1)}$
  $t \leftarrow t+1$
end while

Algorithm 5 CCALin Filter

Initialization: $t \leftarrow 0$, GF $(\cdot, \cdot, \cdot)$ an instance of GenELinK Filter
while True **do**
  Receive $\mathbf{X}^{(t)} \in \mathbb{R}^{n \times d_1}$, $\mathbf{Y}^{(t)} \in \mathbb{R}^{n \times d_2}$
  $\mathbf{S}_{xx}^{(t)} \leftarrow \left(\mathbf{X}^{(t)}\right)^\top \mathbf{X}^{(t)}/n$, $\mathbf{S}_{yy}^{(t)} \leftarrow \left(\mathbf{Y}^{(t)}\right)^\top \mathbf{Y}^{(t)}/n$, $\mathbf{S}_{xy}^{(t)} \leftarrow \left(\mathbf{X}^{(t)}\right)^\top \mathbf{Y}^{(t)}/n$
  $\mathbf{A}^{(t)} \leftarrow \begin{pmatrix} 0 & \mathbf{S}_{xy}^{(t)} \\ \left(\mathbf{S}_{xy}^{(t)}\right)^\top & 0 \end{pmatrix}$, $\mathbf{B}^{(t)} \leftarrow \begin{pmatrix} \mathbf{S}_{xx}^{(t)} & 0 \\ 0 & \mathbf{S}_{yy}^{(t)} \end{pmatrix}$
  $\begin{pmatrix} \bar{\mathbf{W}}_x \in \mathbb{R}^{d_1 \times k} \\ \bar{\mathbf{W}}_y \in \mathbb{R}^{d_2 \times k} \end{pmatrix} \leftarrow \text{GF}\left(\mathbf{A}^{(t)}, \mathbf{B}^{(t)}, \text{GS}_{\mathbf{B}^{(t)}}\right)$
  $\mathbf{U}^{(t)} \leftarrow 2k \times k$ random Gaussian matrix
  $\tilde{\mathbf{W}}_x^{(t)} \leftarrow \bar{\mathbf{W}}_x^{(t)}\mathbf{U}^{(t)}$, $\tilde{\mathbf{W}}_y^{(t)} \leftarrow \bar{\mathbf{W}}_y^{(t)}\mathbf{U}^{(t)}$
  $\mathbf{W}_x^{(t)} \leftarrow \text{GS}_{\mathbf{S}_{xx}^{(t)}}\left(\tilde{\mathbf{W}}_x^{(t)}\right)$, $\mathbf{W}_y^{(t)} \leftarrow \text{GS}_{\mathbf{S}_{yy}^{(t)}}\left(\tilde{\mathbf{W}}_y^{(t)}\right)$
  yield $\mathbf{W}_x^{(t)}$, $\mathbf{W}_y^{(t)}$
  $t \leftarrow t + 1$
end while

To satisfy these criteria, it will likely be necessary to make a call to the GenELinK subroutine or something like it for each minibatch. There are at least two clear options aside from what is described in Algorithm 4.

- Perform the full GenELinK subroutine on each minibatch and initialize each round's GenELinK with the previous round's canonical bases.
- For each minibatch, perform only a single stochastic gradient step on the quadratic program inside the GenELinK subroutine before giving the output back to the CCA algorithm

2.3.5 m-View CCALin

To derive the m-view variation of CCALin, it should be sufficient to receive m datasets $\mathbf{X}_1, \ldots, \mathbf{X}_m$ (or minibatches for the filtering scenario) and change the generation of $\mathbf{A}$ and $\mathbf{B}$ to the following

$$\mathbf{A} \leftarrow \begin{pmatrix} 0 & \mathbf{S}_{x_1x_2} & \cdots & \mathbf{S}_{x_1x_m} \\ \left(\mathbf{S}_{x_1x_2}\right)^\top & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{S}_{x_{m-1}x_m} \\ \left(\mathbf{S}_{x_1x_m}\right)^\top & \cdots & \left(\mathbf{S}_{x_{m-1}x_m}\right)^\top & 0 \end{pmatrix}$$

$$\mathbf{B} \leftarrow \begin{pmatrix} \mathbf{S}_{x_1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{S}_{x_m} \end{pmatrix}$$

where $\mathbf{S}_{x_ix_j} \leftarrow \mathbf{X}_i^\top \mathbf{X}_j/n$ and $\mathbf{S}_{x_i} \leftarrow \mathbf{X}_i^\top \mathbf{X}_i/n$.

3 Probabilistic Models

Eventually, I would like to develop rigorous probabilistic models for each of the objective functions in section **??**. The place to start is probably Bach and Jordan (2005), which develops such a model for the optimization problem in section 2.1.1. It is important to note that the resulting graphical models are not identical to the dependency graphs discussed in section 2.1.

References

Bach, F. R. and Jordan, M. I. (2005). A probabilistic interpretation of canonical correlation analysis.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Ge, R., Jin, C., Kakade, S. M., Netrapalli, P., and Sidford, A. (2016). Efficient algorithms for large-scale generalized eigenvector computation and canonical correlation analysis. *arXiv preprint arXiv:1604.03930*.

Ma, Z., Lu, Y., and Foster, D. (2015). Finding linear structure in large datasets with scalable canonical correlation analysis. *arXiv preprint arXiv:1506.08170*.

Parikh, N., Boyd, S. P., et al. (2014). Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239.