1 Intro

2 Optimization Problems

2.1 Objective Functions

2.1.1 Classical Two-View CCA

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

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

2.1.2 Complete Multiview Graph

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

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

2.1.3 (Mean Field Variational?) Approximation of Fully Connected Graph

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

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

2.1.4 Arbitrary Dependency Graph

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

where E is the set of edges in the dependency graph between the views, and $\mathbf{\Phi}_i$ are the optimization variables.

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.

2.2.2 Alternatives

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. 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.

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 or regularizer, 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.

3 Algorithms

3.1 Stochastic AppGrad

Algorithm 1 Stochastic m-View AppGrad

```
1: Input: batch size $n$, step sizes $\eta_{1t}, \dots, \eta_{mt}$, max iter $T$, num components $k$
2: Output: Canonical bases $(\mathbf{\Phi}_1, \dots, \mathbf{\Phi}_m)$
3: Initialization: $t = 0$, $\tilde{\mathbf{\Phi}}_1^{(0)} \in \mathbb{R}^{p_1 \times k}, \dots, \tilde{\mathbf{\Phi}}_m^{(0)} \in \mathbb{R}^{p_m \times k}$ each with entries sampled invidually from $\mathcal{N}(0,1)$, $\mathbf{\Phi}_j^{(0)} = \mathbf{0}$ for $j = 1, \dots, m$
4: while $t < T$ and not converged do
5:   Receive minibatch $\mathbf{X}_1^{(t)} \in \mathbb{R}^{n \times p_1}, \dots, \mathbf{X}_m^{(t)} \in \mathbb{R}^{n \times p_m}$
6:   for $i = 1, \dots, m$ do
7:     $\tilde{\mathbf{\Phi}}_j^{(t+1)} \leftarrow \tilde{\mathbf{\Phi}}_i^{(t)} - \eta_{it} \left(\mathbf{X}_i^{(t)}\right)^\top \left((m-1)\mathbf{X}_i^{(t)}\tilde{\mathbf{\Phi}}_i^{(t)} - \sum_{j \neq i} \mathbf{X}_j^{(t)}\mathbf{\Phi}_j^{(t)}\right)$
8:     SVD: $\left(\tilde{\mathbf{\Phi}}_i^{(t+1)}\right)^\top \left(\frac{1}{n}\mathbf{X}_i{}^\top \mathbf{X}_i\right) \tilde{\mathbf{\Phi}}_i^{(t+1)} = \mathbf{U}_i^\top \mathbf{D}_i \mathbf{U}_i$
9:     $\mathbf{\Phi}_i^{(t+1)} \leftarrow \tilde{\mathbf{\Phi}}_i^{(t+1)} \mathbf{U}_i^\top \mathbf{D}_i^{-\frac{1}{2}} \mathbf{U}_i$
10:  end for
11:  $t \leftarrow t + 1$
12: 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.

3.1.1 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.

3.2 GenELinK and CCALin

Algorithm 2 GenELinK

```
1:
```

3.2.1 Stochastic GenELinK and CCALin

For transforming the CCA algorithm with GenELinK subroutine into a stochastic minibatch 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.

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:

- 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