Approximate CCA

Karl Stratos

1 CCA Projections

Let $X \in \mathbb{R}^{n_1}$ and $Y \in \mathbb{R}^{n_2}$ be two random variables. Define covariance matrices $\mathbf{C}_{XY} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{C}_{XX} \in \mathbb{R}^{n_1 \times n_1}$, and $\mathbf{C}_{YY} \in \mathbb{R}^{n_2 \times n_2}$ where $[\mathbf{C}_{XY}]_{i,j} = \mathrm{Cov}(X_i, Y_j)$, $[\mathbf{C}_{XX}]_{i,j} = \mathrm{Cov}(X_i, X_j)$, and $[\mathbf{C}_{YY}]_{i,j} = \mathrm{Cov}(Y_i, Y_j)$. We assume that $\mathbf{C}_{XX}$ and $\mathbf{C}_{YY}$ are invertible. Given a dimension $m \leq \min(n_1, n_2)$, the goal is to find projections $\mathbf{A}_m \in \mathbb{R}^{n_1 \times m}$ and $\mathbf{B}_m \in \mathbb{R}^{n_2 \times m}$ such that the correlation between $\mathbf{A}_m^\top X \in \mathbb{R}^m$ and $\mathbf{B}_m^\top Y \in \mathbb{R}^m$ is maximized.

CCA finds such projections in three steps. First, it "whitens" $\mathbf{C}_{XY}$ to obtain $\Omega \in \mathbb{R}^{n_1 \times n_2}$:

$$\Omega = \mathbf{C}_{XX}^{-1/2} \mathbf{C}_{XY} \mathbf{C}_{YY}^{-1/2}$$

Then it performs an SVD to find the top m left and right singular vectors of Ω. Finally, it de-whitens the singular vectors to recover the projection matrices. The algorithm is as follows:

CCA-PROJECTIONS
Input: covariance matrices $\mathbf{C}_{XY} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{C}_{XX} \in \mathbb{R}^{n_1 \times n_1}$, $\mathbf{C}_{YY} \in \mathbb{R}^{n_2 \times n_2}$, CCA dimension m.
Output: CCA projections $\mathbf{A}_m \in \mathbb{R}^{n_1 \times m}$ and $\mathbf{B}_m \in \mathbb{R}^{n_2 \times m}$

1. (Whitening) $\Omega \leftarrow \mathbf{C}_{XX}^{-1/2} \mathbf{C}_{XY} \mathbf{C}_{YY}^{-1/2} \in \mathbb{R}^{n_1 \times n_2}$
2. (SVD) Perform an SVD to decompose $\Omega = U \Lambda V^\top$. Let $U_m \in \mathbb{R}^{n_1 \times m}$ and $V_m \in \mathbb{R}^{n_2 \times m}$ be the first m columns of U and V, respectively.
3. (De-Whitening) Return $\mathbf{A}_m \leftarrow \mathbf{C}_{XX}^{-1/2} U_m$ and $\mathbf{B}_m \leftarrow \mathbf{C}_{YY}^{-1/2} V_m$

In practice, we will have N samples $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}) \ldots (\mathbf{x}^{(N)}, \mathbf{y}^{(N)}) \in X \times Y$ and derive empirical estimates $\hat{\mathbf{C}}_{XY} \in \mathbb{R}^{n_1 \times n_2}$, $\hat{\mathbf{C}}_{XX} \in \mathbb{R}^{n_1 \times n_1}$, and $\hat{\mathbf{C}}_{YY} \in \mathbb{R}^{n_2 \times n_2}$ of the covariance matrices where

$$[\hat{\mathbf{C}}_{XY}]_{i,j} = \frac{1}{N} \sum_{k=1}^{N} (\mathbf{x}_i^{(k)} - \mu_i^X)(\mathbf{y}_j^{(k)} - \mu_j^Y)$$
$$[\hat{\mathbf{C}}_{XX}]_{i,j} = \frac{1}{N} \sum_{k=1}^{N} (\mathbf{x}_i^{(k)} - \mu_i^X)(\mathbf{x}_j^{(k)} - \mu_j^X)$$
$$[\hat{\mathbf{C}}_{YY}]_{i,j} = \frac{1}{N} \sum_{k=1}^{N} (\mathbf{y}_i^{(k)} - \mu_i^Y)(\mathbf{y}_j^{(k)} - \mu_j^Y)$$

with the sample means $\mu^X = \frac{1}{N} \sum_k \mathbf{x}^{(k)}$ and $\mu^Y = \frac{1}{N} \sum_k \mathbf{y}^{(k)}$.

2 Approximate CCA Projections

When dimensions n_1 and n_2 are large, the following aspects of the above approach can be challenging:

- Storing $\mathbf{C}_{XY} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{C}_{XX} \in \mathbb{R}^{n_1 \times n_1}$, and $\mathbf{C}_{YY} \in \mathbb{R}^{n_2 \times n_2}$
- Calculating $\Omega = \mathbf{C}_{XX}^{-1/2} \mathbf{C}_{XY} \mathbf{C}_{YY}^{-1/2}$
- Performing an SVD on Ω

The covariance matrices are not necessarily sparse, so they may be too large to store in memory. These large matrices also make whitening difficult, since we must compute the inverse square root of $\mathbf{C}_{XX}$ and $\mathbf{C}_{YY}$ and multiply by $\mathbf{C}_{XY}$. Finally, a conventional SVD may be too computationally expensive to perform on Ω since it has time and storage complexity superlinear in n_1 and n_2.

The algorithm below, an approximate version of **CCA-PROJECTIONS**, bypasses these difficulties. This algorithm only takes a (sparse) matrix, two count vectors, and a few parameter values as the input and returns the CCA projections as the output.

APPROXIMATE-CCA-PROJECTIONS
Input:

- Co-occurrence matrix $\mathbf{O}_{XY}$ of size $n_1 \times n_2$ (sparse)
- Count vectors $\mathbf{c}^X$ of length n_1 and $\mathbf{c}^Y$ of length n_2
- Number of samples N
- CCA dimension m
- Pseudocount κ (for smoothing), oversampling parameter l, number of power iterations q

Output: approximate CCA projections $\hat{\mathbf{A}}_m \in \mathbb{R}^{n_1 \times m}$ and $\hat{\mathbf{B}}_m \in \mathbb{R}^{n_2 \times m}$

1. (Whitening) Compute diagonal matrices $\hat{\mathbf{C}}_{XX}^{-1/2} \in \mathbb{R}^{n_1 \times n_1}$ and $\hat{\mathbf{C}}_{YY}^{-1/2} \in \mathbb{R}^{n_2 \times n_2}$ where

$$[\hat{\mathbf{C}}_{XX}^{-1/2}]_{i,i} = \frac{1}{\sqrt{\frac{\mathbf{c}_i^X + \kappa}{N} - \left(\frac{\mathbf{c}_i^X + \kappa}{N}\right)^2}} \qquad [\hat{\mathbf{C}}_{YY}^{-1/2}]_{i,i} = \frac{1}{\sqrt{\frac{\mathbf{c}_i^Y + \kappa}{N} - \left(\frac{\mathbf{c}_i^Y + \kappa}{N}\right)^2}}$$

   and compute

$$\hat{\Omega} = \hat{\mathbf{C}}_{XX}^{-1/2} \left(\frac{\mathbf{O}_{XY}}{N} - \left(\frac{\mathbf{c}^X}{N}\right)\left(\frac{\mathbf{c}^Y}{N}\right)^\top \right) \hat{\mathbf{C}}_{YY}^{-1/2}$$

2. (SVD) $[\hat{U}_m, \hat{V}_m] \leftarrow$ **APPROXIMATE-SVD**$(\hat{\Omega}, m, l, q)$
3. (De-Whitening) Return $\hat{\mathbf{A}}_m \leftarrow \hat{\mathbf{C}}_{XX}^{-1/2} \hat{U}_m$ and $\hat{\mathbf{B}}_m \leftarrow \hat{\mathbf{C}}_{YY}^{-1/2} \hat{V}_m$

The above algorithm makes use of a randomized approximate SVD, given below.

APPROXIMATE-SVD
Input: matrix $M \in \mathbb{R}^{n_1 \times n_2}$, CCA dimension m, oversampling parameter l, number of power iterations q
Output: approximate top m left and right singular vectors of M: $\hat{U}_m \in \mathbb{R}^{n_1 \times m}$ and $\hat{V}_m \in \mathbb{R}^{n_2 \times m}$

- Find an orthonormal "basis" $Q \in \mathbb{R}^{n_1 \times (m+l)}$ of M such that $M \approx QQ^\top M$.
  - Make a random $\Theta \in \mathbb{R}^{n_2 \times (m+l)}$ with entries drawn from the standard normal distribution.
  - (Power Iteration) Let $Y = (MM^\top)^q M\Theta \in \mathbb{R}^{n_1 \times (m+l)}$.
  - Find the orthonormal basis Q of the range of Y by performing a QR-decomposition on $Y = QR$.
- Obtain a smaller matrix $B = Q^\top M \in \mathbb{R}^{(m+l) \times n_2}$.
- Perform an SVD to have the decomposition $B = \tilde{U}\Lambda\hat{V}^\top$, and let $\tilde{U}_m \in \mathbb{R}^{(m+l) \times m}$ and $\hat{V}_m \in \mathbb{R}^{n_2 \times m}$ be the first m columns of $\tilde{U}$ and $\hat{V}$, respectively.
- Return $\hat{U}_m = Q\tilde{U}_m \in \mathbb{R}^{n_1 \times m}$ and $\hat{V}_m$.

3 Justification of the Approximation

3.1 Approximate Whitening

We want to estimate $\Omega = \mathbf{C}_{XX}^{-1/2}\mathbf{C}_{XY}\mathbf{C}_{YY}^{-1/2}$ without explicitly computing $\mathbf{C}_{XY}$, $\mathbf{C}_{XX}^{-1/2}$, or $\mathbf{C}_{YY}^{-1/2}$. To do this, we make the following key assumptions on the data:

Assumption 1. The variables are binary: $X \in \{0,1\}^{n_1}$ and $Y \in \{0,1\}^{n_2}$. This is mild in that people typically use binary features to encode the data.

Assumption 2. Each dimension is independent: $\mathrm{Cov}(X_i, X_j) = 0$ and $\mathrm{Cov}(Y_i, Y_j) = 0$ for $i \neq j$. This is significantly stronger but reasonable if the features are more or less independent of each other.

Under assumption 2, $\mathbf{C}_{XX}$ and $\mathbf{C}_{YY}$ are *diagonal* where $[\mathbf{C}_{XX}]_{i,j} = \mathrm{Var}(X_i)$ and $[\mathbf{C}_{YY}]_{i,j} = \mathrm{Var}(Y_i)$ if $i = j$ and 0 otherwise. Therefore, their inverse square roots $\mathbf{C}_{XX}^{-1/2}$ and $\mathbf{C}_{YY}^{-1/2}$ also have a simple diagonal form. Now each entry of $\Omega = \mathbf{C}_{XX}^{-1/2}\mathbf{C}_{XY}\mathbf{C}_{YY}^{-1/2}$ can be computed analytically as

$$[\Omega]_{i,j} = \frac{\mathrm{Cov}(X_i, Y_j)}{\sqrt{\mathrm{Var}(X_i)}\sqrt{\mathrm{Var}(Y_j)}} \tag{1}$$

Note that this is impossible in general, i.e., we used the fact that $\mathbf{C}_{XX}^{-1/2}$ and $\mathbf{C}_{YY}^{-1/2}$ are diagonal. To estimate this Ω from samples $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}) \ldots (\mathbf{x}^{(N)}, \mathbf{y}^{(N)}) \in X \times Y$, compute a co-occurrence matrix $\mathbf{O}_{XY}$ of size n_1 by n_2, count vectors $\mathbf{c}^X$ of length n_1 for X and $\mathbf{c}^Y$ of length n_2:

$$[\mathbf{O}_{XY}]_{i,j} = \sum_{k=1}^{N} \mathbf{x}_i^{(k)}\mathbf{y}_j^{(k)} \qquad [\mathbf{c}^X]_i = \sum_{k=1}^{N} \mathbf{x}_i^{(k)} \qquad [\mathbf{c}^Y]_i = \sum_{k=1}^{N} \mathbf{y}_i^{(k)}$$

Because X and Y are binary, $\mathbf{O}_{XY}$ will typically be a very sparse matrix that is easy to store. Then the emprical estimate of $\mathrm{Cov}(X_i, Y_j)$, $\mathrm{Var}(X_i)$, and $\mathrm{Var}(Y_j)$ will have the following form.

$$\begin{aligned}
\widehat{\mathrm{Cov}}(X_i, Y_j) &= \hat{\mathbf{E}}[X_i Y_j] - \hat{\mathbf{E}}[X_i]\hat{\mathbf{E}}[Y_j] \\
&= \frac{[\mathbf{O}_{XY}]_{i,j}}{N} - \left(\frac{\mathbf{c}_i^X}{N}\right)\left(\frac{\mathbf{c}_j^Y}{N}\right) \\
\widehat{\mathrm{Var}}(X_i) &= \hat{\mathbf{E}}[X_i^2] - \hat{\mathbf{E}}[X_i]^2 = \frac{\mathbf{c}_i^X}{N} - \left(\frac{\mathbf{c}_i^X}{N}\right)^2 \\
\widehat{\mathrm{Var}}(Y_i) &= \hat{\mathbf{E}}[Y_i^2] - \hat{\mathbf{E}}[Y_i]^2 = \frac{\mathbf{c}_i^Y}{N} - \left(\frac{\mathbf{c}_i^Y}{N}\right)^2
\end{aligned}$$

Thus we can compute each entry of matrix $\hat{\Omega}$, an empirical estimate of Ω in Eq. (1), as

$$[\hat{\Omega}]_{i,j} = \frac{\frac{[\mathbf{O}_{XY}]_{i,j}}{N} - \left(\frac{\mathbf{c}_i^X}{N}\right)\left(\frac{\mathbf{c}_j^Y}{N}\right)}{\sqrt{\frac{\mathbf{c}_i^X}{N} - \left(\frac{\mathbf{c}_i^X}{N}\right)^2}\sqrt{\frac{\mathbf{c}_i^Y}{N} - \left(\frac{\mathbf{c}_i^Y}{N}\right)^2}}$$

Step 1 of **APPROXIMATE-CCA-PROJECTIONS** computes this Ω with one difference. It adds a *pseudocount* κ to the count vectors $\mathbf{c}^X$ and $\mathbf{c}^Y$. This has an effect of smoothing the inverse variance estimates, and can be important in practice.

3.2 Approximate SVD

Algorithm **APPROXIMATE-SVD** performs an approximate SVD on a matrix $M \in \mathbb{R}^{n_1 \times n_2}$ as in Halko et al. (2011). The basic idea is that we can use a randomized approach to rapidly create a "thin" orthonormal matrix Q whose range approximates the range of M, reduce the size of M with Q, do an SVD on this smaller matrix, and recover the components in the original space via the orthogonal projection formed by QQ^T.

We must provide two additional parameters, l and q. l is the number of excessive dimensions on top of m in approximating Q, in order to handle the noise in approximation—this can be seen as an instance of oversampling. q is the number of iterations for the power iteration algorithm. The power iteration is used to amplify the decay in the singular spectrum of M, to make sure Q is able to capture most of the range of M with a smaller number of dimensions. Small values suffice in most cases, e.g., $l = 5$ and $q = 1$.

For more details of the algorithm, please refer to the referenced paper.

Acknowledgement. This approach was developed by many practitioners including Jenny Finkel and Dean Foster.

4 Appendix

4.1 Practical Implementation

In the first step of **APPROXIMATE-CCA-PROJECTIONS**, we naively store the $n_1 \times n_2$ matrix $\hat{\Omega}$ which is now *dense*. A practical trick to get around this problem is to store instead a *sparse* matrix $M \in \mathbb{R}^{n_1 \times n_2}$ and vectors $\mathbf{v} \in \mathbb{R}^{n_1}$ and $\mathbf{w} \in \mathbb{R}^{n_2}$ such that $\hat{\Omega} = M - \mathbf{v}\mathbf{w}^\top$. We maintain this expression of $\hat{\Omega}$ until after we have reduced the size of the matrix in **APPROXIMATE-SVD**. See the matlab code shown below.

```
% Approximate CCA
%
% Input:
%   countsXY = (sparse) cooccurrence counts for X and Y
%   countsX = counts for X
%   countsY = counts for Y
%   numData = number of samples
%   ccaDim = CCA dimension
%   extraDim = oversampling parameter
%   powerNum = number of power iterations
%   kappa = smoothing term for inverse variance
%
% Output:
%   A = projection matrix for X
%   B = projection matrix for Y
%   s = top m singular values of Omega
%________________________________________________
function [A B s] = approx_cca(countsXY, countsX, countsY, numData, ...
                              ccaDim, extraDim, powerNum, kappa)

    % whitening
    invsqrt_covX = get_invsqrt_cov(countsX, numData, kappa);
    invsqrt_covY = get_invsqrt_cov(countsY, numData, kappa);

    % maintain sparseness of Omega=XY-X*Y'
    XY = (1/numData) * invsqrt_covX * countsXY * invsqrt_covY; % sparse
    X = (1/numData) * invsqrt_covX * countsX;
    Y = (1/numData) * invsqrt_covY * countsY;

    % do an SVD on Omega
    [U,s,V] = approx_svd(XY, X, Y, ccaDim, extraDim, powerNum);

    % de-whitening
    A = invsqrt_covX * U;
    B = invsqrt_covY * V;
```

```
function [invsqrt_cov] = get_invsqrt_cov(counts, numData, kappa)

    inc_counts = counts + kappa; % smoothing with kappa
    mean = inc_counts/numData;
    var = mean - mean.^2;

    numFeat = length(counts);
    invsqrt_cov = sparse(1:numFeat, 1:numFeat, var.^(-.5));

function [U,s,V] = approx_svd(XY, X, Y, ccaDim, extraDim, powerNum)

    % find orth Q such that XY-X*Y'  is close to QQ'(XY-X*Y')
    numFeat2 = length(Y);
    Theta = randn(numFeat2, ccaDim + extraDim);
    reduced = XY*Theta - X*(Y'*Theta);
    for i=1:powerNum % power iteration
        t1 = XY*(XY'*reduced);
        t2 = XY*(Y*(X'*reduced));
        t3 = X*(Y'*(XY'*reduced));
        t4 = X*(Y'*(Y*(X'*reduced)));
        reduced = t1-t2-t3+t4;
    end
    [Q, ~] = qr(reduced,0);

    % obtain a smaller matrix and do a thin svd
    B = Q'*XY - (Q'*X)*Y';
    [U_,s,V] = svd(B,'econ');
    U_ = U_(:,1:ccaDim);
    s = diag(s(1:ccaDim,1:ccaDim));
    V= V(:,1:ccaDim);

    U = Q*U_; % bring U back to the original dimension
```

4.2 Projecting Data

We are ultimately interested in the reduced dimensional variables $\underline{X} = \mathbf{A}_m^\top X \in \mathbb{R}^m$ and $\underline{Y} = \mathbf{B}_m^\top Y \in \mathbb{R}^m$. Once we compute the CCA projections $\hat{\mathbf{A}}_m \in \mathbb{R}^{n_1 \times m}$ and $\hat{\mathbf{B}}_m \in \mathbb{R}^{n_2 \times m}$ from **APPROXIMATE-CCA-PROJECTIONS**, suppose we wish to compute the CCA representation $\underline{\mathbf{x}} \in \mathbb{R}^m$ of a sample $\mathbf{x} \in \{0,1\}^{n_1}$ in the first view. In a proper approach, we will first center $\mathbf{x}$ by subtracting the sample means $\mu^X \in \mathbb{R}^{n_1}$ before projection:

$$\underline{\mathbf{x}} = \hat{\mathbf{A}}_m^\top(\mathbf{x} - \mu^X) = \hat{\mathbf{A}}_m^\top \mathbf{x} - \hat{\mathbf{A}}_m^\top \mu^X$$

However, since $\hat{\mathbf{A}}_m^\top \mu^X$ is constant for all data points $\mathbf{x}$, it does not make much difference in practice if we just directly project without centering: $\underline{\mathbf{x}} = \hat{\mathbf{A}}_m^\top \mathbf{x}$. In that case, $\underline{\mathbf{x}} \in \mathbb{R}^m$ ($\mathbf{x}$ being a binary vector) is simply a linear combination of the rows of $\hat{\mathbf{A}}_m$.

References

Halko, N., Martinsson, P. G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. In *SIAM Rev.*