Canonical Correlation Clarified by Singular Value Decomposition

William H. Press

May 28, 2011

The Setup

You have a number n of data points, each one of which is a paired measurement of an x value in a p_1 dimensional space and a y value in a p_2 dimensional space. Thus, you are given two data matrices, $\mathbf{x}$ of size $(n \times p_1)$ and $\mathbf{y}$ of size $(n \times p_2)$. Without loss of generality we will suppose that you have subtracted the mean (centroid) of all the data, so that

$$\begin{aligned} \sum_{i=1}^{n} x_{ij} = 0, \quad j = 1, \ldots, p_1 \\ \sum_{i=1}^{n} y_{ij} = 0, \quad j = 1, \ldots, p_2 \end{aligned} \tag{1}$$

Canonical Correlation

You want to find a linear combination of the x coordinates that correlates well over the data with an (in general, different) linear combination of the y coordinates. In fact, you want to find the *best* such matched pair of linear combinations on the x and y sides, that is, the one yielding the largest coefficient of correlation.

But why stop there? Once you have the best pair, you can ask for the second-best pair, that is, the linear combination of x coordinates in the subspace orthogonal to the first combination (on the x side) that best correlates with a linear combination of y coordinates in the subspace orthogonal to the first combination (on the y side). And so on for further pairs or linear combinations in remaining orthogonal subspaces.

We can proceed like this until we have exhaused the orthogonal subspaces on either the x or the y side, whichever comes first. So the number of pairs (termed the number of canonical coordinates) is $d = \min[\text{rank}(\mathbf{x}), \text{rank}(\mathbf{y})]$, where rank means column rank.

We denote the matrices of coefficients of the linear combinations by $\mathbf{a}$ (on the x side), of size $p_1 \times d$, and $\mathbf{b}$ (on the y side), of size $p_2 \times d$. If matrices $\mathbf{u}$ and $\mathbf{v}$ denote the linear combinations evaluated for each data point, we have

$$\mathbf{u} = \mathbf{x}\mathbf{a}, \qquad \mathbf{v} = \mathbf{y}\mathbf{b} \tag{2}$$

The n rows of the matrices $\mathbf{u}$ and $\mathbf{v}$, are called the canonical coordinates of the data points. Both $\mathbf{u}$ and $\mathbf{v}$ have d columns.

The first columns of $\mathbf{a}$ and $\mathbf{b}$, denoted $\mathbf{a}_{(1)}$ and $\mathbf{b}_{(1)}$ have, by definition, the single best linear correlation, so we can write, symbolically,

$$(\mathbf{a}_{(1)}, \mathbf{b}_{(1)}) = \underset{\mathbf{a}_{(1)},\mathbf{b}_{(1)}}{\operatorname{argmax}} \operatorname{corr}[\mathbf{x}\mathbf{a}_{(1)}, \mathbf{y}\mathbf{b}_{(1)}] = \underset{\mathbf{a}_{(1)},\mathbf{b}_{(1)}}{\operatorname{argmax}} \left[\frac{\mathbf{u}_{(1)}^T \mathbf{v}_{(1)}}{\sqrt{\mathbf{u}_{(1)}^T \mathbf{u}_{(1)}} \sqrt{\mathbf{v}_{(1)}^T \mathbf{v}_{(1)}}} \right] \tag{3}$$

where $\mathbf{u}$ and $\mathbf{v}$ are considered functions of $\mathbf{a}$ and $\mathbf{b}$, respectively.

Now consider the correlation between $\mathbf{u}_{(1)}$ and some other column of $\mathbf{v}$, $\mathbf{v}_{(j)}$ with $j \neq 1$. This must be zero, because if $\mathbf{u}_{(1)}$ is correlated with two vectors, then it is even more strongly correlated with an appropriately weighted linear combination of the two. (This is left as an exercise for the reader.)

In like manner, we can see that all the columns of $\mathbf{u}$ and $\mathbf{v}$ are cross-orthogonal. That is,

$$\mathbf{u}^T \mathbf{v} = \mathbf{D} \tag{4}$$

where $\mathbf{D}$ is a diagonal matrix in d dimensions. The matrices $\mathbf{u}$ and $\mathbf{v}$ are orthogonal by their construction in successive orthogonal subspaces. We can choose to scale the columns of $\mathbf{a}$ and $\mathbf{b}$ to make

$$\mathbf{u}^T \mathbf{u} = \mathbf{1}_d, \qquad \mathbf{v}^T \mathbf{v} = \mathbf{1}_d \tag{5}$$

where $\mathbf{1}_d$ is the identity matrix in d dimensions.

The Big Canonical Trick

The big trick is that you don't actually have to do any of the maximizations seemingly implied by equation (3). In the simplest case, where the x and y spaces have the same dimensionality and where there are no column degeneracies in the data (so $p_1 = p_2 = d$), then essentially any construction of $\mathbf{a}$ and $\mathbf{b}$ matrices that makes the orthogonality equations (4) and (5) be satisfied will produce the desired canonical coordinates. We can see this by counting. The $d \times d$ matrices $\mathbf{a}$ and $\mathbf{b}$ have $2d^2$ total degrees of freedom. Each of the conditions in equation (5) imposes $d(d+1)/2$ constraints, taking into account that $\mathbf{u}^T \mathbf{u}$ and $\mathbf{v}^T \mathbf{v}$ are symmetric matrices. Equation (4) imposes $d^2 - d$ constraints, because the elements of $\mathbf{D}$ are not constrained. Now,

$$\tfrac{1}{2}d(d+1) + \tfrac{1}{2}d(d+1) + (d^2 - d) = 2d^2 \tag{6}$$

Thus, total constraints equals total degrees of freedom, and there can be only countable, isolated solutions. In fact (proof not given here), all the solutions are simply column permutations of one another.

If $p_1 \neq p_2$, or if the column ranks are not maximal, then there are "too many columns" on either the x or y sides, and the constraints alone don't uniquely specify the solution. We will see below how the largest correlations in this case are selected by singular value decomposition (SVD).

Solution by QR Decomposition

Most statistical packages, for example the MATLAB Statistical Toolbox [1], calculate the canonical coordinates by QR decomposition, essentially the ancient but beautiful algorithm of Golub [2, 3]. First decompose

$$\mathbf{x} = \mathbf{q}_1\,\mathbf{r}_1, \qquad \mathbf{y} = \mathbf{q}_2\,\mathbf{r}_2, \tag{7}$$

where the $\mathbf{q}$'s are column orthogonal, $\mathbf{q}_1^T\,\mathbf{q}_1 = \mathbf{1}_{p_1}$, $\mathbf{q}_2^T\,\mathbf{q}_2 = \mathbf{1}_{p_2}$, and the $\mathbf{r}$'s are upper triangular ($p_1 \times p_1$ and $p_2 \times p_2$, respectively). Next, form $\mathbf{q}_1^T\,\mathbf{q}_2$ and take its singular value decomposition,

$$\mathbf{q}_1^T\,\mathbf{q}_2 = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{8}$$

where $\mathbf{U}$ and $\mathbf{V}$ are column orthogonal and $\mathbf{S}$ is diagonal. Now try the ansatz,

$$\mathbf{a} \equiv \mathbf{r}_1^{-1}\mathbf{U}, \qquad \mathbf{b} \equiv \mathbf{r}_2^{-1}\mathbf{V} \tag{9}$$

Triangular backsubstitution is used to compute these, so the indicated inverse matrix is not actually required.

Let's now check equations (4) and (5):

$$\begin{aligned}\mathbf{D} = \mathbf{u}^T\,\mathbf{v} &= \mathbf{a}^T\,\mathbf{x}^T\,\mathbf{y}\,\mathbf{b}\\ &= (\mathbf{U}^T\mathbf{r}_1^{-1T})(\mathbf{r}_1^T\mathbf{q}_1^T)(\mathbf{q}_2\mathbf{r}_2)(\mathbf{r}_2^{-1}\mathbf{V})\\ &= \mathbf{U}^T(\mathbf{U}\mathbf{S}\mathbf{V}^T)\mathbf{V}\\ &= \mathbf{S}\end{aligned} \tag{10}$$

Here we've used just associativity and the orthogonality relations. Since $\mathbf{S}$ is diagonal, so is $\mathbf{D}$. Similarly,

$$\begin{aligned}\mathbf{u}^T\,\mathbf{u} &= \mathbf{a}^T\,\mathbf{x}^T\,\mathbf{x}\,\mathbf{a}\\ &= (\mathbf{U}^T\mathbf{r}_1^{-1T})(\mathbf{r}_1^T\mathbf{q}_1^T)(\mathbf{q}_1\mathbf{r}_1)(\mathbf{r}_1^{-1}\mathbf{U})\\ &= \mathbf{1}\end{aligned} \tag{11}$$

and an analogous calculation for $\mathbf{v}^T\,\mathbf{v}$. So it all works!

If $p_1 \neq p_2$, or if there are degeneracies, then only the first d columns of $\mathbf{U}$ and $\mathbf{V}$ are kept in equation (9), corresponding to the d largest singular values in $\mathbf{S}$. This insures that the largest d canonical correlations are kept (though we are not giving a proof here).

This is a beautiful algorithm, and it is basically the fastest way to compute a canonical correlation. We have three small, essentially pedagogical, reservations, however. (1) The algorithm uses both QR and SVD uses decompositions. Many users have a reasonable intuitive understanding of SVD, because of its use in principal component analysis, but few have such an understanding of QR. (2) In fact, for large data sets, the algorithm requires a pretty fancy QR routine, one implemented for non-square matrices, and with pivoting that has to be tracked and undone at the end. (*Numerical Recipes* [5], for example, lacks this.) (3)

It is not entirely transparent how to answer ancillary questions, for example, "How much total variance in x is accounted for by each x canonical coordinate, and how much of *that* variance is accounted for by the corresponding y data values?" For these reasons, it can be useful to have an algorithm for canonical correlation that uses SVD only.

Solution Using SVD Only

SVD is slower than QR by a modest factor, typically less than 2 in the regime of interest, $n \gg \max(p_1, p_2)$. In fact, SVD and QR are related decompositions: most algorithms for SVD have an internal QR step. The goal here is not speed, but clarity and perhaps flexibility; but the speed penalty is not large.

The main purpose of the QR decompositions of $\mathbf{x}$ and $\mathbf{y}$ in the algorithm above is to construct an orthogonal basis for each, $\mathbf{q}_1$ and $\mathbf{q}_2$, respectively. A secondary desideratum is that the transformation into this basis be easily invertable, which follows from $\mathbf{r}_1$ and $\mathbf{r}_2$ being triangular. But SVD also constructs orthogonal bases; and its transformations are also easily invertable, using the orthogonality properties of its output matrices.

Hussmann, in a note immediately following this one [4], shows that, in fact, any orthogonalization of the data matrices $\mathbf{x}$ and $\mathbf{y}$ will produce the same canonical correlations.

In some problems one is interested in both the canonical correlations between $\mathbf{x}$ and $\mathbf{y}$, and also their individual principal components (principal component analysis, or PCA). In that case, the calculation of the canonical correlation is essentially free, because SVD must already be done on each data set to get its principal components.

The algorithm is, first, SVD both $\mathbf{x}$ and $\mathbf{y}$,

$$\mathbf{x} = \mathbf{u}_1 \mathbf{s}_1 \mathbf{v}_1^T, \qquad \mathbf{y} = \mathbf{u}_2 \mathbf{s}_2 \mathbf{v}_2^T \tag{12}$$

Next, form $\mathbf{u}_1^T \mathbf{u}_2$ and SVD it,

$$\mathbf{u}_1^T \mathbf{u}_2 = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{13}$$

Now define

$$\mathbf{a} \equiv \mathbf{v}_1 \mathbf{s}_1^{-1} \mathbf{U}, \qquad \mathbf{b} \equiv \mathbf{v}_2 \mathbf{s}_2^{-1} \mathbf{V} \tag{14}$$

Check equations (4) and (5) as above:

$$\begin{aligned} \mathbf{D} = \mathbf{u}^T \mathbf{v} &= \mathbf{a}^T \mathbf{x}^T \mathbf{y} \mathbf{b} \\ &= (\mathbf{U}^T \mathbf{s}_1^{-1} \mathbf{v}_1^T)(\mathbf{v}_1 \mathbf{s}_1 \mathbf{u}_1^T)(\mathbf{u}_2 \mathbf{s}_2 \mathbf{v}_2^T)(\mathbf{v}_2 \mathbf{s}_2^{-1} \mathbf{V}) \\ &= \mathbf{U}^T (\mathbf{U}\mathbf{S}\mathbf{V}^T) \mathbf{V} \\ &= \mathbf{S} \end{aligned} \tag{15}$$

$$\begin{aligned} \mathbf{u}^T \mathbf{u} &= \mathbf{a}^T \mathbf{x}^T \mathbf{x} \mathbf{a} \\ &= (\mathbf{U}^T \mathbf{s}_1^{-1} \mathbf{v}_1^T)(\mathbf{v}_1 \mathbf{s}_1 \mathbf{u}_1^T)(\mathbf{u}_1 \mathbf{s}_1 \mathbf{v}_1^T)(\mathbf{v}_1 \mathbf{s}_1^{-1} \mathbf{U}) \\ &= \mathbf{1} \end{aligned} \tag{16}$$

and correspondingly for $\mathbf{v}^T\mathbf{v}$. As in the previous algorithm, we keep only the first d columns of $\mathbf{U}$ and $\mathbf{V}$. There is no permutation or pivoting.

Allocations of Variance

The total variance of $\mathbf{x}$ is

$$\|\mathbf{x}\| \equiv \sum_{i,j} x_{ij}^2 = \mathrm{tr}(\mathbf{x}^T\mathbf{x}) = \mathrm{tr}(\mathbf{v}_1\mathbf{s}_1\mathbf{u}_1^T\mathbf{u}_1\mathbf{s}_1\mathbf{v}_1^T) = \mathrm{tr}(\mathbf{s}_1^2) = \sum_i s_{1i}^2 \tag{17}$$

(The second from last equality is obtained by cyclically permuting the factor $\mathbf{v}_1$.) Equation (17) shows the well-known partition of total variance into a descending sum of squares of singular values, each corresponding to a principal component of $\mathbf{x}$.

Canonical coordinates, which depend on $\mathbf{x}$ and $\mathbf{y}$ together, are not, of course, the same as principal coordinates, which depend only on $\mathbf{x}$ and $\mathbf{y}$ independently. However, since the canonical coordinates are orthogonal, equation (5), they also partition the variance. To see how, let $\mathbf{\Delta}$ be a diagonal matrix of size p_1 with only ones and zeros on the diagonal. Then the matrix

$$\mathbf{x}' \equiv (\mathbf{u}\mathbf{\Delta})\mathbf{a}^{-1} \tag{18}$$

(compare equation (2)) defines a modified data matrix in which certain canonical coordinates of each data point – those having zero diagonal element in $\mathbf{\Delta}$ – are set to zero. Now compute the total variance of $\mathbf{x}'$,

$$\begin{aligned}\|\mathbf{x}'\| &= \mathrm{tr}(\mathbf{x}'^T\mathbf{x}') = \mathrm{tr}(\mathbf{a}^{-1T}\mathbf{\Delta}\mathbf{u}^T\mathbf{u}\mathbf{\Delta}\mathbf{a}^{-1})\\ &= \mathrm{tr}[(\mathbf{v}_1\mathbf{s}_1\mathbf{U})\mathbf{\Delta}\mathbf{u}^T\mathbf{u}\mathbf{\Delta}(\mathbf{U}^T\mathbf{s}_1\mathbf{v}_1^T)]\\ &= \mathrm{tr}[\mathbf{\Delta}(\mathbf{U}^T\mathbf{s}_1)(\mathbf{s}_1\mathbf{U})]\\ &= \sum_i \Delta_i \sum_k (s_{1\,kk}\,U_{ki})^2\end{aligned} \tag{19}$$

The last line can be seen to partition the variance into pieces associated with each canonical coordinate, indexed by i,

$$C_i \equiv \sum_k (s_{1\,kk}\,U_{ki})^2 \tag{20}$$

In the nondegenerate case $p_1 = p_2 = d$, the sum of the C_i's will equal the total variance, equation (17), as can be seen formally by setting $\mathbf{\Delta} = \mathbf{1}$ in equation (19). If $\mathrm{rank}(\mathbf{y}) < \mathrm{rank}(\mathbf{x})$, then the total variance in the canonical coordinates will be less the total variance, because the canonical coordinates won't span the full range of $\mathbf{x}$.

Since the diagonal elements of $\mathbf{D}$, denoted D_{ii}, are coefficients of correlation, their squares are fractions of variance in x explained by y. The total variance in x thus explained by y through canonical coordinate i is thus $C_i D_{ii}^2$, with $i = 1, \ldots, d$. The total variance of x explained by y through *all* canonical coordinates is $\sum_i C_i D_{ii}^2$, whose value lies between 0 and the total variance (equation (17)).

Analogous equations of course hold for partitioning y's variance and explaining it from x.

Reference Code

An appendix lists reference code in C++, using the conventions of [5].

References

[1] MathWorks, Inc. (2009) “canoncorr.m” in MATLAB Statistical Toolbox, ver. 2009a.

[2] Golub GH (1969) Matrix decompositions and statistical calculations. Technical Report No. CS 124, Computer Science Department, Stanford. At http://infolab.stanford.edu/pub/cstr/reports/cs/tr/69/124/CS-TR-69-124.pdf .

[3] Bjorck A and Golub GH (1973) “Numerical Methods for Computing Angles between Linear Subspaces,” . Mathematics of Computation, **27**, 579–594. At http://www.jstor.org/stable/2005662 .

[4] Hussmann J (2011) “Why any orthonormal bases for the data matrices work for finding canonical correlations,” immediately following this note in the same PDF file.

[5] Press WH, Teukolsky SA, Vetterling WT, and Flannery BP (2007) *Numerical Recipes: The Art of Scientific Computing*, Third Edition (Cambridge University Press).

```
struct CanonCorr {
    // canonical correlation using SVD instead of QR
    // gives results identical to MATLAB canoncorr.m and
    // computes some additional quantities re variances
    Int n, p1, p2, d1, d2, d;
    Doub xtotvar, ytotvar;
    MatDoub a, b, u, v, x, y;
    VecDoub r, xccvar, yccvar, xexplvar, yexplvar;

    CanonCorr(MatDoub &xx, MatDoub &yy, Doub thresh=1.e-8) : x(xx), y(yy),
        n(xx.nrows()), p1(xx.ncols()), p2(yy.ncols()), d1(p1), d2(p2) {
        // thresh is the regularization threshold for degen cols in x or y
        Int i,j,k;
        Doub sum,sum1,sum2;
        VecDoub sinv1(p1), sinv2(p2);
        if (y.nrows() != n) throw("in CanonCorr unequal no. of rows");
        // demeanify the data
        for (j=0;j<p1;j++) {
            sum = 0.;
            for (i=0;i<n;i++) sum += x[i][j];
            sum /= n;
            for (i=0;i<n;i++) x[i][j] -= sum;
        }
        for (j=0;j<p2;j++) {
            sum = 0.;
            for (i=0;i<n;i++) sum += y[i][j];
            sum /= n;
            for (i=0;i<n;i++) y[i][j] -= sum;
        }
        // do SVDs and regularize the inverse sing vals
        SVD svd1(x);
        SVD svd2(y);
        for (i=0;i<p1;i++) {
            if (svd1.w[i] < thresh) { d1 = i; break; } // x has degen cols
            sinv1[i] = 1./svd1.w[i];
        }
        for (i=0;i<p2;i++) {
            if (svd2.w[i] < thresh) { d2 = i; break; } // y has degen cols
            sinv2[i] = 1./svd2.w[i];
        }
        // on output d1 == p1 if x is col nondegen, d2 == p2 if y is col nondegen
        // total variances (L2 norms of x and y, not divided by n)
        xtotvar = ytotvar = 0.;
        for (i=0;i<p1;i++) xtotvar += SQR(svd1.w[i]);
        for (i=0;i<p2;i++) ytotvar += SQR(svd2.w[i]);
        // form the cross-covariance matrix q
        MatDoub q(d1,d2);
        for (i=0;i<d1;i++) for (j=0;j<d2;j++) {
            sum = 0.;
            for (k=0;k<n;k++) sum += svd1.u[k][i]*svd2.u[k][j];
            q[i][j] = sum;
        }
        SVD svd(q);
        d = MIN(d1,d2);
        // fraction of variance in x and y due to each canonical coord
        xccvar.assign(d,0.);
        yccvar.assign(d,0.);
        for (i=0;i<d;i++) {
            for (k=0;k<d1;k++) xccvar[i] += SQR(svd1.w[k] * svd.u[k][i]);
            for (k=0;k<d2;k++) yccvar[i] += SQR(svd2.w[k] * svd.v[k][i]);
            xccvar[i] /= xtotvar;
            yccvar[i] /= ytotvar;
        }
        // compute the a and b matrices
        a.resize(p1,d);
        b.resize(p2,d);
        for (i=0;i<p1;i++) for (j=0;j<d;j++) {
            sum = 0.;
            for (k=0;k<d1;k++) sum += svd1.v[i][k]*sinv1[k]*svd.u[k][j];
            a[i][j] = sum;
        }
        for (i=0;i<p2;i++) for (j=0;j<d;j++) {
            sum = 0.;
            for (k=0;k<d2;k++) sum += svd2.v[i][k]*sinv2[k]*svd.v[k][j];
            b[i][j] = sum;
        }
        // project r into [0,1] to correct any roundoff
        r.resize(d);
        for (j=0;j<d;j++) r[j] = MAX(0.,MIN(1.,svd.w[j]));
        // fraction of total variance of x explained by y and vice versa for each cc
        xexplvar.resize(d);
        yexplvar.resize(d);
        for (i=0;i<d;i++) {
            xexplvar[i] = xccvar[i] * SQR(r[i]);
            yexplvar[i] = yccvar[i] * SQR(r[i]);
        }
        // normalize a and b to agree with MATLAB's (perhaps nutty) convention
        sum = sqrt(n-1.);
        for (j=0;j<d;j++) {
            for (k=0;k<p1;k++) a[k][j] *= sum;
            for (k=0;k<p2;k++) b[k][j] *= sum;
        }
        // form u and v matrices
        u.resize(n,d);
        v.resize(n,d);
        for (i=0;i<n;i++) {
            for (j=0;j<d;j++) {
                sum1 = sum2 = 0.;
                for (k=0;k<p1;k++) sum1 += x[i][k]*a[k][j];
                for (k=0;k<p2;k++) sum2 += y[i][k]*b[k][j];
                u[i][j] = sum1;
                v[i][j] = sum2;
            }
        }
    }
};
```

Why any orthonormal bases for the data matrices work for finding canonical correlations

Jeff Hussmann

May 26, 2011

Given $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times l}$, the problem is to find the pair $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^l$ that maximize

$$\frac{y^T B^T A x}{\|By\|_2 \|Ax\|_2}.$$

Let

$$A = U_A N_A$$

and

$$B = U_B N_B$$

be any decompositions of A and B such that $U_A \in \mathbb{R}^{m\times n}$ and $U_B \in \mathbb{R}^{m\times l}$ with $U_A^T U_A = I_n$ and $U_B^T U_B = I_l$ (i.e. U_A and U_B each have orthonormal columns), and $N_A \in \mathbb{R}^{n\times n}$ and $N_B \in \mathbb{R}^{l\times l}$ are nonsingular. (For example, each N could result from a QR decomposition and be upper-triangular, or could result from a reduced SVD decomposition and be the product of a diagonal matrix and an orthogonal matrix.)

Then the original objective becomes

$$\frac{y^T N_B^T U_B^T U_A N_A x}{\|U_B N_B y\|_2 \|U_A N_A x\|_2}.$$

Since the columns of U_A and of U_B are orthonormal, the 2-norms in the denominator are unaffected by the presence of the Us. Since N_A and N_B are nonsingular, we can set $s = N_A x$ and $t = N_B t$ and solve the equivalent problem of finding the pair $s \in \mathbb{R}^n$ and $t \in \mathbb{R}^l$ that maximize

$$\frac{t^T U_B^T U_A s}{\|t\|_2 \|s\|_2}.$$

This is problem **P2.5.5** in *Matrix Computations*, 3rd Ed. Let $C = U_B^T U_A$. If we view the SVD of C as a summary of the action of C on the 2-norm unit ball, it is intuitively clear that the answer consists of the first left and right singular vectors of C. To make this rigorous, note that the problem is equivalent to maximizing $t^T C s$ subject to $\|t\|_2 = \|s\|_2 = 1$. Form the Lagrangian

$$\Lambda(s, t, \lambda, \mu) = t^T C s - \frac{\lambda}{2}(\|s\|_2 - 1) - \frac{\mu}{2}(\|t\|_2 - 1).$$

Then $\nabla \Lambda = 0$ gives

$$C^T t - \lambda s = 0$$

and

$$Cs - \mu t = 0.$$

Substitute to give

$$CC^T t = \lambda\mu t$$

and

$$C^T C s = \lambda\mu s,$$

and the eigenvectors corresponding to the largest eigenvalue of CC^T and $C^T C$ are the first left and right singular vectors of C, respectively.

Of course, this only deals with the first canonical correlation, but the idea extends to the subsequent orthogonal correlations; see Theorem 8.6.1 in *Matrix Computations*, 3rd Ed.