Deep Canonical Correlation Analysis

Galen Andrew GALEN@CS.WASHINGTON.EDU
University of Washington

Raman Arora ARORA@TTIC.EDU
Toyota Technological Institute at Chicago

Jeff Bilmes BILMES@EE.WASHINGTON.EDU
University of Washington

Karen Livescu KLIVESCU@TTIC.EDU
Toyota Technological Institute at Chicago

Abstract

We introduce Deep Canonical Correlation Analysis (DCCA), a method to learn complex nonlinear transformations of two views of data such that the resulting representations are highly linearly correlated. Parameters of both transformations are jointly learned to maximize the (regularized) total correlation. It can be viewed as a nonlinear extension of the linear method *canonical correlation analysis* (CCA). It is an alternative to the nonparametric method *kernel canonical correlation analysis* (KCCA) for learning correlated nonlinear transformations. Unlike KCCA, DCCA does not require an inner product, and has the advantages of a parametric method: training time scales well with data size and the training data need not be referenced when computing the representations of unseen instances. In experiments on two real-world datasets, we find that DCCA learns representations with significantly higher correlation than those learned by CCA and KCCA. We also introduce a novel non-saturating sigmoid function based on the cube root that may be useful more generally in feedforward neural networks.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

1. Introduction

Canonical correlation analysis (CCA) (Hotelling, 1936; Anderson, 1984) is a standard statistical technique for finding linear projections of two random vectors that are maximally correlated. Kernel canonical correlation analysis (KCCA) (Akaho, 2001; Melzer et al., 2001; Bach & Jordan, 2002; Hardoon et al., 2004) is an extension of CCA in which maximally correlated *nonlinear* projections, restricted to reproducing kernel Hilbert spaces with corresponding kernels, are found. Both CCA and KCCA are techniques for learning representations of two data views, such that each view's representation is simultaneously the most predictive of, and the most predictable by, the other.

CCA and KCCA have been used for unsupervised data analysis when multiple views are available (Hardoon et al., 2007; Vinokourov et al., 2003; Dhillon et al., 2011); learning features for multiple modalities that are then fused for prediction (Sargin et al., 2007); learning features for a single view when another view is available for representation learning but not at prediction time (Blaschko & Lampert, 2008; Chaudhuri et al., 2009; Arora & Livescu, 2012); and reducing sample complexity of prediction problems using unlabeled data (Kakade & Foster, 2007). The applications range broadly across a number of fields, including medicine, meteorology (Anderson, 1984), chemometrics (Montanarella et al., 1995), biology and neurology (Vert & Kanehisa, 2002; Hardoon et al., 2007), natural language processing (Vinokourov et al., 2003; Haghighi et al., 2008; Dhillon et al., 2011), speech processing (Choukri & Chollet, 1986; Rudzicz, 2010; Arora & Livescu, 2013), computer vision (Kim et al., 2007), and multimodal signal processing (Sargin et al., 2007; Slaney & Covell,

2000). An appealing property of CCA for prediction tasks is that, if there is noise in either view that is uncorrelated with the other view, the learned representations should not contain the noise in the uncorrelated dimensions.

While kernel CCA allows learning of nonlinear representations, it has the drawback that the representation is limited by the fixed kernel. Also, as it is a nonparametric method, the time required to train KCCA or compute the representations of new datapoints scales poorly with the size of the training set. In this paper, we consider learning flexible nonlinear representations via deep networks. Deep networks do not suffer from the aforementioned drawbacks of nonparametric models, and given the empirical success of deep models on a wide variety of tasks, we may expect to be able to learn more highly correlated representations. Deep networks have been used widely to learn representations, for example using deep Boltzmann machines (Salakhutdinov & Hinton, 2009), deep autoencoders (Hinton & Salakhutdinov, 2006), and deep nonlinear feedforward networks (Hinton et al., 2006). These have been very successful for learning representations of a single data view. In this work we introduce *deep CCA* (DCCA), which simultaneously learns two deep nonlinear mappings of two views that are maximally correlated. This can be loosely thought of as learning a kernel for KCCA, but the mapping function is not restricted to live in a reproducing kernel Hilbert space.

The most closely related work is that of Ngiam *et al.* on multimodal autoencoders (Ngiam et al., 2011) and of Srivastava and Salakhutdinov on multimodal restricted Boltzmann machines (Srivastava & Salakhutdinov, 2012). In these approaches, there is a single network being learned with one or more layers connected to both views (modalities); in the absence of one of the views, it can be predicted from the other view using the learned network. The key difference is that in our approach we learn two separate deep encodings, with the objective that the learned encodings are as correlated as possible. These different objectives may have advantages in different settings. In the current work, we are interested specifically in the correlation objective, that is in extending CCA with learned nonlinear mappings. Our approach is therefore directly applicable in all of the settings where CCA and KCCA are used, and we compare its ability relative to CCA and KCCA to generalize the correlation objective to new data, showing that DCCA achieves much better results.

In the following sections, we review CCA and KCCA, introduce deep CCA, and describe experiments on two data sets comparing the three methods. In principle we could evaluate the learned representations on any task in which CCA or KCCA have been used. However, in this paper we focus on the most direct measure of performance, namely correlation between the learned representations on unseen test data.

2. Background: CCA, KCCA, and deep representations

Let $(X_1, X_2) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$ denote random vectors with covariances $(\Sigma_{11}, \Sigma_{22})$ and cross-covariance Σ_{12}. CCA finds pairs of linear projections of the two views, $(w_1'X_1, w_2'X_2)$ that are maximally correlated:

$$(w_1^*, w_2^*) = \underset{w_1, w_2}{\operatorname{argmax}} \ \operatorname{corr}(w_1'X_1, w_2'X_2) \tag{1}$$

$$= \underset{w_1, w_2}{\operatorname{argmax}} \ \frac{w_1'\Sigma_{12}w_2}{\sqrt{w_1'\Sigma_{11}w_1 w_2'\Sigma_{22}w_2}}. \tag{2}$$

Since the objective is invariant to scaling of w_1 and w_2, the projections are constrained to have unit variance:

$$(w_1^*, w_2^*) = \underset{w_1'\Sigma_{11}w_1 = w_2'\Sigma_{22}w_2 = 1}{\operatorname{argmax}} \ w_1'\Sigma_{12}w_2 \tag{3}$$

When finding multiple pairs of vectors (w_1^i, w_2^i), subsequent projections are also constrained to be uncorrelated with previous ones, that is $w_1^i\Sigma_{11}w_1^j = w_2^i\Sigma_{22}w_2^j = 0$ for $i < j$. Assembling the top k projection vectors w_1^i into the columns of a matrix $A_1 \in \mathbb{R}^{n_1 \times k}$, and similarly placing w_2^i into $A_2 \in \mathbb{R}^{n_2 \times k}$, we obtain the following formulation to identify the top $k \leq \min(n_1, n_2)$ projections:

$$\begin{aligned} \text{maximize:} \quad & \operatorname{tr}(A_1'\Sigma_{12}A_2) \\ \text{subject to:} \quad & A_1'\Sigma_{11}A_1 = A_2'\Sigma_{22}A_2 = I. \end{aligned} \tag{4}$$

There are several ways to express the solution to this objective; we follow the one in (Mardia et al., 1979). Define $T \triangleq \Sigma_{11}^{-1/2}\Sigma_{12}\Sigma_{22}^{-1/2}$, and let U_k and V_k be the matrices of the first k left- and right- singular vectors of T. Then the optimal objective value is the sum of the top k singular values of T (the Ky Fan k-norm of T) and the optimum is attained at $(A_1^*, A_2^*) = (\Sigma_{11}^{-1/2}U_k, \Sigma_{22}^{-1/2}V_k)$. Note that this solution assumes that the covariance matrices Σ_{11} and Σ_{22} are nonsingular, which is satisfied in practice because they are estimated from data with regularization: given centered data matrices $\bar{H}_1 \in \mathbb{R}^{n_1 \times m}, \bar{H}_2 \in \mathbb{R}^{n_2 \times m}$, one can estimate, e.g.

$$\hat{\Sigma}_{11} = \frac{1}{m-1}\bar{H}_1\bar{H}_1' + r_1 I, \tag{5}$$

where $r_1 > 0$ is a regularization parameter. Estimating the covariance matrices with regularization also reduces the detection of spurious correlations in the training data, a.k.a. "overfitting" (De Bie & De Moor, 2003).

2.1. Kernel CCA

Kernel CCA finds pairs of nonlinear projections of the two views (Hardoon et al., 2004). The Reproducing Kernel Hilbert Spaces (RKHS) of functions on $\mathbb{R}^{n_1}, \mathbb{R}^{n_2}$ are denoted $\mathcal{H}_1, \mathcal{H}_2$ and the associated positive definite kernels are denoted κ_1, κ_2. The optimal projections are those functions $f_1^* \in \mathcal{H}_1, f_2^* \in \mathcal{H}_2$ that maximize the correlation between $f_1^*(X_1)$ and $f_2^*(X_2)$:

$$\begin{aligned}(f_1^*, f_2^*) &= \underset{f_1 \in \mathcal{H}_1, f_2 \in \mathcal{H}_2}{\operatorname{argmax}} \operatorname{corr}\left(f_1(X_1), f_2(X_2)\right) \\ &= \underset{f_1 \in \mathcal{H}_1, f_2 \in \mathcal{H}_2}{\operatorname{argmax}} \frac{\operatorname{cov}\left(f_1(X_1), f_2(X_2)\right)}{\sqrt{\operatorname{var}\left(f_1(X_1)\right) \operatorname{var}\left(f_2(X_2)\right)}},\end{aligned} \tag{6}$$

To solve the nonlinear KCCA problem, the "kernel trick" is used: Since the nonlinear maps $f_1 \in \mathcal{H}_1$, $f_2 \in \mathcal{H}_2$ are in RKHS, the solutions can be expressed as linear combinations of the kernels evaluated at the data: $f_1(x) = \alpha_1' \kappa_1(x, \cdot)$, where $\kappa_1(x, \cdot)$ is a vector whose i^{th} element is $\kappa_1(x, x_i)$ (resp. for $f_2(x)$). KCCA can then be written as finding vectors $\alpha_1, \alpha_2 \in \mathbb{R}^m$ that solve the optimization problem

$$\begin{aligned}(\alpha_1^*, \alpha_2^*) &= \underset{\alpha_1, \alpha_2}{\operatorname{argmax}} \frac{\alpha_1' K_1 K_2 \alpha_2}{\sqrt{(\alpha_1' K_1^2 \alpha_2)\ (\alpha_1' K_2^2 \alpha_2)}} \\ &= \underset{\alpha_1' K_1^2 \alpha_1 = \alpha_2' K_2^2 \alpha_2 = 1}{\operatorname{argmax}} \alpha_1' K_1 K_2 \alpha_2,\end{aligned} \tag{7}$$

where $K_1 \in \mathbb{R}^{m \times m}$ is the centered Gram matrix $K_1 = K - K\mathbf{1} - \mathbf{1}K + \mathbf{1}K\mathbf{1}$, $K_{ij} = \kappa_1(x_i, x_j)$ and $\mathbf{1} \in \mathbb{R}^{m \times m}$ is an all-1s matrix, and similarly for K_2. Subsequent vectors (α_1^j, α_2^j) are solutions of (7) with the constraints that $(f_1^j(X_1), f_2^j(X_2))$ are uncorrelated with the previous ones.

Proper regularization may be critical to the performance of KCCA, since the spaces $\mathcal{H}_1$, $\mathcal{H}_2$ could have high complexity. Since $\alpha_1' f_1(\cdot)$ plays the role of w_1 in KCCA, the generalization of $w_1' w_1$ would be $\alpha_1' K_1 \alpha$. Therefore the correct generalization of (5) is to use $K_1^2 + r_1 K_1$ in place of K_1^2 in the constraints of (7), for regularization parameter $r_1 > 0$ (resp. for K_2^2).

The optimization is in principle simple: The objective is maximized by the top eigenvectors of the matrix

$$(K_1 + r_1 I)^{-1}\ K_2\ (K_2 + r_2 I)^{-1}\ K_1. \tag{8}$$

The regularization coefficients r_1 and r_2, as well as any parameters of the kernel in KCCA, can be tuned using held-out data. Often a further regularization is done by first projecting the data onto an intermediate-dimensionality space, between the target and original dimensionality (Ek et al., 2008; Arora & Livescu, 2012). In practice solving KCCA may not be straightforward, as the kernel matrices become very large for real-world data sets of interest, and iterative SVD algorithms for the initial dimensionality reduction can be used (Arora & Livescu, 2012).

2.2. Deep learning

"Deep" networks, having more than two layers, are capable of representing nonlinear functions involving multiply nested high-level abstractions of the kind that may be necessary to accurately model complex real-world data. There has been a resurgence of interest in such models following the advent of various successful unsupervised methods for initializing the parameters ("pretraining") in such a way that a useful solution can be found (Hinton et al., 2006; Hinton & Salakhutdinov, 2006). Contrastive divergence (Bengio & Delalleau, 2009) has had great success as a pretraining technique, as have many variants of autoencoder networks, including the denoising autoencoder (Vincent et al., 2008) used in the present work. The growing availability of both data and compute resources also contributes to the resurgence, because empirically the performance of deep networks seems to scale very well with data size and complexity.

While deep networks are more commonly used for learning classification labels or mapping to another vector space with supervision, here we use them to learn nonlinear transformations of two datasets to a space in which the data is highly correlated, just as KCCA does. The same properties that may account for deep networks' success in other tasks—high model complexity, the ability to concisely represent a hierarchy of features for modeling real-world data distributions—could be particularly useful in a setting where the output space is significantly more complex than a single label.

3. Deep Canonical Correlation Analysis

Deep CCA computes representations of the two views by passing them through multiple stacked layers of nonlinear transformation (see Figure 1). Assume for simplicity that each intermediate layer in the network for the first view has c_1 units, and the final (output) layer has o units. Let $x_1 \in \mathbb{R}^{n_1}$ be an instance of the first view. The outputs of the first layer for the instance x_1 are $h_1 = s(W_1^1 x_1 + b_1^1) \in \mathbb{R}^{c_1}$, where $W_1^1 \in \mathbb{R}^{c_1 \times n_1}$ is a matrix of weights, $b_1^1 \in \mathbb{R}^{c_1}$ is a vector of biases, and $s : \mathbb{R} \mapsto \mathbb{R}$ is a nonlinear function applied componentwise. The outputs h_1 may then be used to compute the outputs of the next layer as $h_2 = s(W_2^1 h_1 + b_2^1) \in \mathbb{R}^{c_1}$, and so on until the final representation $f_1(x_1) = s(W_d^1 h_{d-1} + b_d^1) \in \mathbb{R}^o$ is computed, for a network with d layers. Given an instance x_2 of the second view, the representation $f_2(x_2)$ is computed the

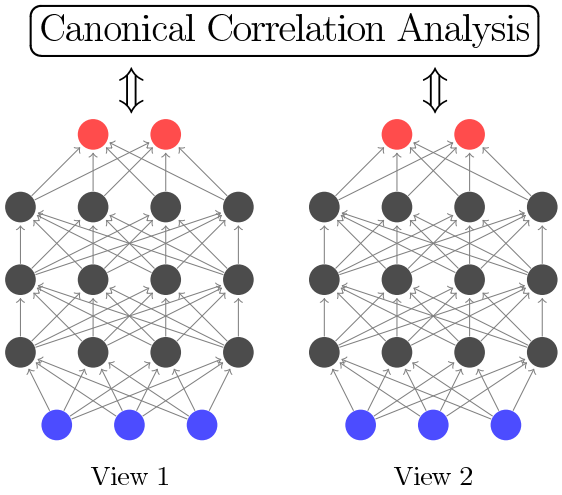

Figure 1. A schematic of deep CCA, consisting of two deep networks learned so that the output layers (topmost layer of each network) are maximally correlated. Blue nodes correspond to input features ($n_1 = n_2 = 3$), grey nodes are hidden units ($c_1 = c_2 = 4$), and the output layer is red ($o = 2$). Both networks have $d = 4$ layers.

same way, with different parameters W_l^2 and b_l^2 (and potentially different architectural parameters c_2 and d). The goal is to jointly learn parameters for both views W_l^v and b_l^v such that $\mathrm{corr}(f_1(X_1), f_2(X_2))$ is as high as possible. If θ_1 is the vector of all parameters W_l^1 and b_l^1 of the first view for $l = 1, \ldots, d$, and similarly for θ_2, then

$$(\theta_1^*, \theta_2^*) = \underset{(\theta_1, \theta_2)}{\mathrm{argmax}} \, \mathrm{corr}(f_1(X_1; \theta_1), f_2(X_2; \theta_2)). \quad (9)$$

To find (θ_1^*, θ_2^*), we follow the gradient of the correlation objective as estimated on the training data. Let $H_1 \in \mathbb{R}^{o \times m}, H_2 \in \mathbb{R}^{o \times m}$ be matrices whose columns are the top-level representations produced by the deep models on the two views, for a training set of size m. Let $\bar{H}_1 = H_1 - \frac{1}{m} H_1 \mathbf{1}$ be the centered data matrix (resp. $\bar{H}_2$), and define $\hat{\Sigma}_{12} = \frac{1}{m-1} \bar{H}_1 \bar{H}_2'$, and $\hat{\Sigma}_{11} = \frac{1}{m-1} \bar{H}_1 \bar{H}_1' + r_1 I$ for regularization constant r_1 (resp. $\hat{\Sigma}_{22}$). Assume that $r_1 > 0$ so that $\hat{\Sigma}_{11}$ is positive definite.

As discussed in section 2 for CCA, the total correlation of the top k components of H_1 and H_2 is the sum of the top k singular values of the matrix $T = \hat{\Sigma}_{11}^{-1/2} \hat{\Sigma}_{12} \hat{\Sigma}_{22}^{-1/2}$. If we take $k = o$, then this is exactly the matrix trace norm of T, or[1]

$$\mathrm{corr}(H_1, H_2) = \|T\|_{\mathrm{tr}} = \mathrm{tr}(T'T)^{1/2}. \quad (10)$$

The parameters W_l^v and b_l^v of DCCA are trained to optimize this quantity using gradient-based optimization. To compute the gradient of $\mathrm{corr}(H_1, H_2)$ with respect to all parameters W_l^v and b_l^v, we can compute its gradient with respect to H_1 and H_2 and then use backpropagation. If the singular value decomposition of T is $T = UDV'$, then

$$\frac{\partial \mathrm{corr}(H_1, H_2)}{\partial H_1} = \frac{1}{m-1} \left(2 \nabla_{11} \bar{H}_1 + \nabla_{12} \bar{H}_2 \right). \quad (11)$$

where

$$\nabla_{12} = \hat{\Sigma}_{11}^{-1/2} U V' \hat{\Sigma}_{22}^{-1/2} \quad (12)$$

and

$$\nabla_{11} = -\frac{1}{2} \hat{\Sigma}_{11}^{-1/2} U D U' \hat{\Sigma}_{11}^{-1/2}, \quad (13)$$

and $\partial \mathrm{corr}(H_1, H_2) / \partial H_2$ has a symmetric expression. The derivation of the gradient is not entirely straightforward (involving, for example, the gradient of the trace of the matrix square-root, which we could not find in standard references such as (Petersen & Pedersen, 2012)) and is given in the appendix. We also regularize (10) by adding to it a quadratic penalty with weight $\lambda_b > 0$ for all parameters.

Because the correlation objective is a function of the entire training set that does not decompose into a sum over data points, it is not clear how to use a stochastic optimization procedure that operates on data points one at a time. We experimented with a stochastic method based on mini-batches, but obtained much better results with full-batch optimization using the L-BFGS second-order optimization method (Nocedal & Wright, 2006) which has been found to be useful for deep learning in other contexts (Le et al., 2011).

As discussed in section 2.2 for deep models in general, the best results will in general not be obtained if parameter optimization is started from random initialization—some form of pretraining is necessary. In our experiments, we initialize the parameters of each layer with a denoising autoencoder (Vincent et al., 2008). Given centered input training data assembled into a matrix $X \in \mathbb{R}^{n \times m}$, a distorted matrix $\tilde{X}$ is created by adding i.i.d. zero-mean Gaussian noise with variance σ_a^2. For parameters $W \in \mathbb{R}^{c \times n}$ and $b \in \mathbb{R}^c$, the reconstructed data $\hat{X} = W' s(W\tilde{X} + b\bar{1}')$ is formed. Then we use L-BFGS to find a local minimum of the total squared error from the reconstruction to the original data, plus a quadratic penalty:

$$l_a(W, b) = ||\hat{X} - X||_F^2 + \lambda_a (||W||_F^2 + ||b||_2^2), \quad (14)$$

where $||\cdot||_F$ is the matrix Frobenius norm. The minimizing values W^* and b^* are used to initialize optimization of the DCCA objective, and to produce the representation for pretraining the next layer. σ_a^2 and λ_a are treated as hyperparameters, and optimized on a development set, as described in section 4.1.

[1] Here we abuse notation slightly, writing $\mathrm{corr}(H_1, H_2)$ as the empirical correlation of the data represented by the matrices H_1 and H_2.

3.1. Non-saturating nonlinearity

Any form of sigmoid nonlinearity could be used to determine the output of the nodes in a DCCA network, but in our experiments we obtained the best results using a novel non-saturating sigmoid function based on the cube root. If $g : \mathbb{R} \mapsto \mathbb{R}$ is the function $g(y) = y^3/3 + y$, then our function is $s(x) = g^{-1}(x)$. Like the more popular logistic (σ) and tanh nonlinearities, s has sigmoid shape and has unit slope at $x = 0$. Like tanh, it is an odd function. However, logistic and tanh approach their asymptotic value very quickly, at which point the derivative drops to essentially zero (i.e., they saturate). On the other hand, s is not bounded, and its derivative falls off much more gradually with x. We hypothesize that these properties make s better-suited for batch optimization with second-order methods which might otherwise get stuck on a plateau early during optimization. In figure 2 we plot s alongside tanh for comparison.

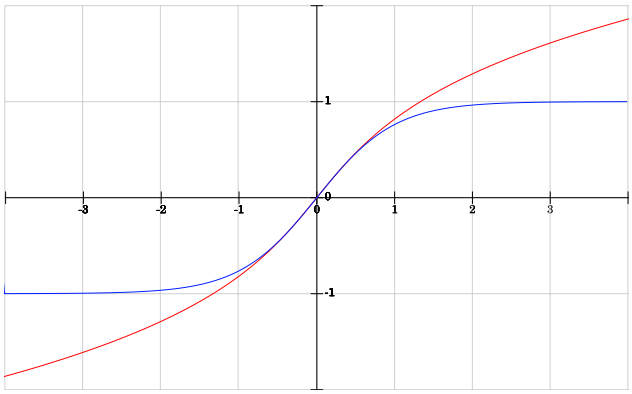

Figure 2. Comparison of our modified cube-root sigmoid function (red) with the more standard tanh (blue).

Another property that our nonsaturating sigmoid function shares with logistic and tanh is that its derivative is a simple function of its value. For example, $\sigma'(x) = \sigma(x)(1 - \sigma(x))$, and $\tanh'(x) = 1 - \tanh^2(x)$. This property is convenient in implementations, because it means the input to a unit can be overwritten by its output. Also, as it turns out, it is more efficient to compute the derivatives as a function of the value in all of these cases (e.g., given $y = \tanh(x)$, $1 - y^2$ can be computed more efficiently than $1 - \tanh^2(x)$). In the case of s, we have $s'(x) = (s^2(x) + 1)^{-1}$ as is easily shown with implicit differentiation. If $y = s(x)$, then

$$x = y^3/3 + y, \quad \frac{dx}{dy} = y^2 + 1, \quad \text{and} \quad \frac{dy}{dx} = \frac{1}{y^2 + 1}.$$

To compute $s(x)$, we use Newton's method. To solve for $g(y) - x = 0$, iterate

$$\begin{aligned} y_{n+1} &= y_n - \frac{g(y_n) - x}{g'(y_n)} \\ &= y_n - \frac{y_n^3/3 + y_n - x}{y_n^2 + 1} = \frac{2y_n^3/3 + x}{y_n^2 + 1}. \end{aligned}$$

For positive x, initializing $y_0 = x$, the iteration decreases monotonically, so convergence is guaranteed. In the range of values in our experiments, it converges to machine precision in just a few iterations. When x is negative, we use the property that s is odd, so $s(x) = -s(-x)$. As a further optimization, we wrote a vectorized implementation.

4. Experiments

We perform experiments on two datasets to demonstrate that DCCA learns transformations that are not only dramatically more correlated than a linear CCA baseline, but also significantly more correlated than well-tuned KCCA representations. We refer to a DCCA model with an output size of o and d layers (including the output) as DCCA-o-d.

Because the total correlation of two transformed views grows with dimensionality, it is important to compare only equal-dimensionality representations. In addition, in order to compare the test correlation of the top k components of two representations of dimensionality $o_1, o_2 \geq k$, the components must be ordered by their correlation on the training data. In the case of CCA and KCCA, the dimensions are always ordered in this way; but in DCCA, there is no ordering to the output nodes. Therefore, we derive such an ordering by performing a final (linear) CCA on the output layers of the two views on the training data. This final CCA produces two projection matrices A_1, A_2, which are applied to the DCCA test output before computing test set correlation. Another way would be to compute a new DCCA representation at each target dimensionality; this is not done here for expediency but should, if anything, improve performance.

4.1. Hyperparameter optimization

Each of the DCCA models we tested has a fixed number of layers and output size, and the parameters W_l^v and b_l^v are trained as discussed in section 3. Several other values are treated as hyperparameters. Specifically, for each view, we have σ_a^2 and λ_a for autoencoder pretraining, c, the width of all hidden layers (a large integer parameter treated as a real value) and r, the CCA regularization hyperparameter. Finally there is a single hyperparameter λ_b, the fine-tuning regularization weight. These values were chosen to optimize total

correlation on a development set using a derivative-free optimization method.

4.2. MNIST handwritten digits

For our first experiments, we learn correlated representations of the left and right halves of handwritten digit images. We use the MNIST handwritten image dataset (LeCun & Cortes, 1998), which consists of 60,000 train images and 10,000 test images. We randomly selected 10% (6,000) images from the training set to use for hyperparameter tuning. Each image is a 28x28 matrix of pixels, each representing one of 256 grayscale values. The left and right 14 columns are separated to form the two views, making 392 features in each view. For KCCA, we use a radial basis function (RBF) kernel for both views: $k_1(x_i, x_j) = e^{-\|x_i - x_j\|^2/2\sigma_1^2}$ and similarly for k_2. The bandwidth parameters σ_1, σ_2 are tuned over the range $[0.25, 64]$. Regularization parameters r_1, r_2 for CCA and KCCA are tuned over the range $[10^{-8}, 10]$. The four parameters were jointly tuned to maximize correlation at $k = 50$ on the development set. We use a scalable KCCA algorithm based on incremental SVD (Arora & Livescu, 2012). The selected widths of the hidden layers for the DCCA-50-2 model were 2038 (left half-images) and 1608 (right half-images). Table 1 compares the total correlation on the development and test sets obtained for the 50 most correlated dimensions with linear CCA, KCCA, and DCCA.

	CCA	KCCA (RBF)	DCCA (50-2)
Dev	28.1	33.5	**39.4**
Test	28.0	33.0	**39.7**

Table 1. Correlation captured in the 50 most correlated dimensions on the split MNIST dataset.

4.3. Articulatory speech data

The second set of experiments uses speech data from the Wisconsin X-ray Microbeam Database (XRMB) (Westbury, 1994) of simultaneous acoustic and articulatory recordings. The articulatory data consist of horizontal and vertical displacements of eight pellets on the speaker's lips, tongue, and jaws, yielding a 16-dimensional vector at each time point. The baseline acoustic features consist of standard 13-dimensional mel-frequency cepstral coefficients (MFCCs) (Davis & Mermelstein, 1980) and their first and second derivatives computed every 10ms over a 25ms window. The articulatory measurements are downsampled to match the MFCC frame rate.

The input features X_1 and X_2 to CCA/KCCA/DCCA are the acoustic and articulatory features concatenated over a 7-frame window around each frame, giving acoustic vectors $X_1 \in \mathbb{R}^{273}$ and articulatory vectors $X_2 \in \mathbb{R}^{112}$. We discard frames that are missing any of the articulatory data (e.g., due to mistracked pellets), resulting in $m \approx 50,000$ frames for each speaker. For KCCA, besides an RBF kernel (described in the previous section) we also use a polynomial kernel of degree d, with $k_1(x_i, x_j) = (x_i^T x_j + c)^d$ and similarly for k_2.

We run five independent experiments, each using 60% of the utterances for learning projections, 20% for tuning hyperparameters (regularization parameters and kernel bandwidths), and 20% for final testing. For this set of experiments, kernel bandwidths for the RBF kernel were fixed at $\sigma_1 = 4 \times 10^6, \sigma_2 = 2 \times 10^4$ to match the variance in the un-normalized data. For the polynomial kernel we tuned the degree d over the set $\{2, 3\}$ and the offset parameter c over the range $[0.25, 2]$ to optimize development set correlation at $k = 110$. Hyperparameter optimization selected the number of hidden units per layer in the DCCA-50-2 model as 1641 and 1769 for the MFCC and XRMB views respectively. In the DCCA-112-3 model, 1811 and 1280 units per layer, respectively, were chosen. The widths for the DCCA-112-8 model were fixed at 781 and 552 as discussed in the last paragraph of this section.

Table 2 compares total correlation captured in the top 50 dimensions on the test data for all five folds with CCA, KCCA with both kernels, and DCCA-50-2. The pattern of performance across the folds is similar for all four models, and DCCA consistently finds more correlation.

	CCA	KCCA (RBF)	KCCA (Poly)	DCCA (50-2)
Fold 1	16.8	29.2	32.3	**38.2**
Fold 2	15.8	25.3	29.1	**34.1**
Fold 3	16.9	30.8	34.0	**39.4**
Fold 4	16.6	28.6	32.4	**37.1**
Fold 5	16.2	26.2	29.9	**34.0**

Table 2. Correlation captured in the 50 most correlated dimensions on the articulatory dataset.

Figure 3 shows correlation obtained using linear CCA, KCCA with RBF kernel, KCCA with a polynomial kernel ($d = 2, c = 1$), and various topologies of Deep CCA, on the test set of one of the folds as a function of number of dimensions. The KCCA models tend to detect slightly more correlation in the first few components, after which the Deep CCA models outperform them by a large margin. We note that DCCA may particularly have an advantage when k is equal to the

number of output units o. We found that DCCA models with only two or three output units can indeed find more correlation than the top two or three components of KCCA (results not shown). This is also consistent with the observation that DCCA-50-2 has the highest performance of any model at $k = 50$.

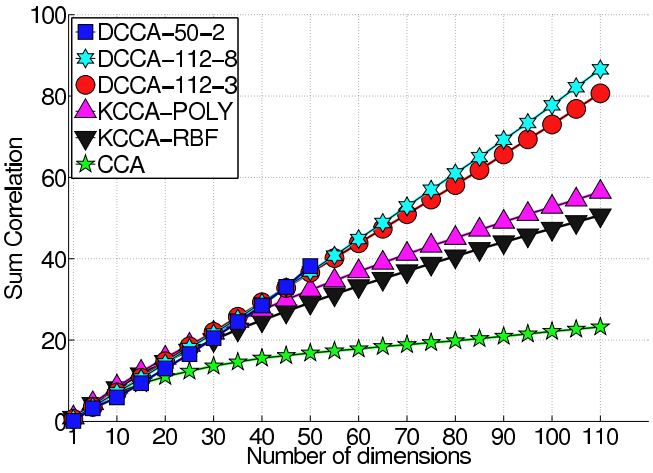

Figure 3. Correlation as a function of number of dimensions. Note that DCCA-50-2 is truncated at $k = o = 50$.

To determine the impact of model depth (number of layers) on performance, we conducted an experiment in which we increased the number of layers from three to eight, while reducing the number of hidden units in each layer in order to keep the total number of parameters approximately constant. The output width was fixed at 112, and all hyperparameters other than the number of hidden units were kept fixed at the values chosen for DCCA-112-3. Table 3 gives the total correlation on the first fold as a function of the number of layers. Note that the total correlation of both datasets increases monotonically with the depth of DCCA and even with eight layers we have not reached saturation.

layers (d)	3	4	5	6	7	8
Dev set	66.7	68.1	70.1	72.5	76.0	**79.1**
Test set	80.4	81.9	84.0	86.1	88.5	**88.6**

Table 3. Total correlation captured, on one of the folds, by DCCA-112-d, for d ranging from three to eight.

5. Discussion

We have shown that deep CCA can obtain improved representations with respect to the correlation objective measured on unseen data. DCCA provides a flexible nonlinear alternative to KCCA. Another appealing feature of DCCA is that, like CCA, it does not require an inner product. As a parametric model, representations of unseen datapoints can be computed without reference to the training set.

In many applications of CCA, such as classification and regression, maximizing the correlation is not the final goal and the correlated representations are used in the service of another task. A natural next step is therefore to test the representations produced by deep CCA in the context of prediction tasks and to compare against other nonlinear multi-view representation learning approaches that optimize other objectives, e.g., (Ngiam et al., 2011; Srivastava & Salakhutdinov, 2012).

6. Acknowledgments

This research was supported by NSF grant IIS-0905633 and by the Intel/UW ISTC. The opinions expressed in this work are those of the authors and do not necessarily reflect the views of the funders.

7. Appendix: Derivation of DCCA Gradient

To perform backpropagation, we must be able to compute the gradient of $f = \text{corr}(H_1, H_2)$ defined in Equation (10). Denote by ∇_{ij} the matrix of partial derivatives of f with respect to the entries of $\hat{\Sigma}_{ij}$. Let the singular value decomposition of $T = \hat{\Sigma}_{11}^{-1/2}\hat{\Sigma}_{12}\hat{\Sigma}_{22}^{-1/2}$ be given as $T = UDV'$. First we will show that

$$\nabla_{12} = \hat{\Sigma}_{11}^{-1/2} UV' \hat{\Sigma}_{22}^{-1/2} \tag{15}$$

and

$$\nabla_{11} = -\frac{1}{2}\hat{\Sigma}_{11}^{-1/2} UDU' \hat{\Sigma}_{11}^{-1/2} \tag{16}$$

(resp. ∇_{22}). To prove (15), we use the fact that for a matrix X, $\nabla \|X\|_{\text{tr}} = UV'$, where $X = UDV'$ is the singular value decomposition of X (Bach, 2008). Using the chain rule:

$$\begin{aligned}
(\nabla_{12})_{ab} &= \frac{\partial f}{\partial (\hat{\Sigma}_{12})_{ab}} \\
&= \sum_{cd} \frac{\partial f}{\partial T_{cd}} \cdot \frac{\partial T_{cd}}{\partial (\hat{\Sigma}_{12})_{ab}} \\
&= \sum_{cd} (UV')_{cd} \cdot (\hat{\Sigma}_{11}^{-1/2})_{ca} (\hat{\Sigma}_{22}^{-1/2})_{bd} \\
&= (\hat{\Sigma}_{11}^{-1/2} UV' \hat{\Sigma}_{22}^{-1/2})_{ab}
\end{aligned}$$

For (16) we use the identity $\nabla \operatorname{tr} X^{1/2} = \frac{1}{2} X^{-1/2}$. This is easily derived from Theorem 1 of (Lewis, 1996):

Theorem 1. *A matrix function f is called a* spectral function *if it depends only on the set of eigenvalues of its argument. That is, for any positive definite matrix*

X and any unitary matrix V, $f(X) = f(VXV')$. If f is a spectral function, and X is positive definite with eigendecomposition $X = UDU'$, then

$$\frac{\partial f(X)}{\partial X} = U \operatorname{diag} \frac{\partial f(D)}{\partial D} U' \tag{17}$$

Now we can proceed

$$\begin{aligned}(\nabla_{11})_{ab} &= \frac{\partial f}{\partial (\hat{\Sigma}_{11})_{ab}} \\ &= \sum_{cd} \frac{\partial f}{\partial (T'T)_{cd}} \frac{\partial (T'T)_{cd}}{\partial (\hat{\Sigma}_{11})_{ab}} \\ &= \sum_{cd} \left(\frac{1}{2} (T'T)^{-1/2} \right)_{cd} \frac{\partial (T'T)_{cd}}{\partial (\hat{\Sigma}_{11})_{ab}}\end{aligned} \tag{18}$$

Since $T'T = \hat{\Sigma}_{22}^{-1/2} \hat{\Sigma}_{21} \hat{\Sigma}_{11}^{-1} \hat{\Sigma}_{12} \hat{\Sigma}_{22}^{-1/2}$, and using Eq. 60 from (Petersen & Pedersen, 2012) for the derivative of an inverse,

$$\begin{aligned}\frac{\partial (T'T)_{cd}}{\partial (\hat{\Sigma}_{11})_{ab}} &= \sum_{ij} \frac{\partial (T'T)_{cd}}{\partial (\hat{\Sigma}_{11}^{-1})_{ij}} \frac{\partial (\hat{\Sigma}_{11}^{-1})_{ij}}{\partial (\hat{\Sigma}_{11})_{ab}} \\ &= -\sum_{ij} (\hat{\Sigma}_{22}^{-1/2} \hat{\Sigma}_{21})_{ci} (\hat{\Sigma}_{12} \hat{\Sigma}_{22}^{-1/2})_{jd} (\hat{\Sigma}_{11}^{-1})_{ia} (\hat{\Sigma}_{11}^{-1})_{bj} \\ &= -(\hat{\Sigma}_{22}^{-1/2} \hat{\Sigma}_{21} \hat{\Sigma}_{11}^{-1})_{ca} (\hat{\Sigma}_{11}^{-1} \hat{\Sigma}_{12} \hat{\Sigma}_{22}^{-1/2})_{bd} \\ &= -(T' \hat{\Sigma}_{11}^{-1/2})_{ca} (\hat{\Sigma}_{11}^{-1/2} T)_{bd}\end{aligned}$$

So continuing from (18),

$$\begin{aligned}(\nabla_{11})_{ab} &= -\frac{1}{2} \sum_{cd} (T' \hat{\Sigma}_{11}^{-1/2})_{ca} (T'T)_{cd}^{-1/2} (\hat{\Sigma}_{11}^{-1/2} T)_{bd} \\ &= -\frac{1}{2} \sum_{cd} (\hat{\Sigma}_{11}^{-1/2} T)_{ac} (T'T)_{cd}^{-1/2} (T' \hat{\Sigma}_{11}^{-1/2})_{db} \\ &= -\frac{1}{2} (\hat{\Sigma}_{11}^{-1/2} T (T'T)^{-1/2} T' \hat{\Sigma}_{11}^{-1/2})_{ab} \\ &= -\frac{1}{2} (\hat{\Sigma}_{11}^{-1/2} UDV' (VD^{-1}V') VDU' \hat{\Sigma}_{11}^{-1/2})_{ab} \\ &= -\frac{1}{2} (\hat{\Sigma}_{11}^{-1/2} UDU' \hat{\Sigma}_{11}^{-1/2})_{ab}\end{aligned}$$

Using ∇_{12} and ∇_{11}, we are ready to compute $\partial f / \partial H_1$. First (temporarily moving subscripts on H_1 and $\hat{\Sigma}_{11}$ to superscripts so subscripts can index into matrices)

$$\begin{aligned}\frac{\partial \hat{\Sigma}_{ab}^{11}}{\partial H_{ij}^1} &= \begin{cases} \frac{2}{m-1} (H_{ij}^1 - \frac{1}{m} \sum_k H_{ik}^1) & \text{if } a = i, b = i \\ \frac{1}{m-1} (H_{bj}^1 - \frac{1}{m} \sum_k H_{bk}^1) & \text{if } a = i, b \neq i \\ \frac{1}{m-1} (H_{aj}^1 - \frac{1}{m} \sum_k H_{ak}^1) & \text{if } a \neq i, b = i \\ 0 & \text{if } a \neq i, b \neq i \end{cases} \\ &= \frac{1}{m-1} (1_{\{a=i\}} \bar{H}_{bj}^1 + 1_{\{b=i\}} \bar{H}_{aj}^1).\end{aligned}$$

Also,

$$\begin{aligned}\frac{\partial \hat{\Sigma}_{ab}^{12}}{\partial H_{ij}^1} &= \frac{1}{m-1} 1_{\{a=i\}} \left(H_{bj}^2 - \frac{1}{m} \sum_k H_{bk}^2 \right) \\ &= \frac{1}{m-1} 1_{\{a=i\}} \bar{H}_{bj}^2.\end{aligned}$$

Putting this together, we obtain

$$\begin{aligned}\frac{\partial f}{\partial H_{ij}^1} &= \sum_{ab} \nabla_{ab}^{11} \frac{\partial \hat{\Sigma}_{ab}^{11}}{\partial H_{ij}^1} + \sum_{ab} \nabla_{ab}^{12} \frac{\partial \hat{\Sigma}_{ab}^{12}}{\partial H_{ij}^1} \\ &= \frac{1}{m-1} \left(\sum_b \nabla_{ib}^{11} \bar{H}_{bj}^1 + \sum_a \nabla_{ai}^{11} \bar{H}_{aj}^1 + \sum_b \nabla_{ib}^{12} \bar{H}_{bj}^2 \right) \\ &= \frac{1}{m-1} \left((\nabla_{11} \bar{H}_1)_{ij} + (\nabla_{11}' \bar{H}_1)_{ij} + (\nabla_{12} \bar{H}_2)_{ij} \right).\end{aligned}$$

Using the fact that ∇_{11} is symmetric, this can be written more compactly as

$$\frac{\partial f}{\partial H_1} = \frac{1}{m-1} \left(2 \nabla_{11} \bar{H}_1 + \nabla_{12} \bar{H}_2 \right).$$

References

Akaho, S. A kernel method for canonical correlation analysis. In *Proc. Int'l Meeting on Psychometric Society*, 2001.

Anderson, T. W. *An Introduction to Multivariate Statistical Analysis (2nd edition)*. John Wiley and Sons, 1984.

Arora, R. and Livescu, K. Kernel CCA for multi-view learning of acoustic features using articulatory measurements. In *Symp. on Machine Learning in Speech and Language Processing*, 2012.

Arora, R. and Livescu, K. Multi-view CCA-based acoustic features for phonetic recognition across speakers and domains. In *Int. Conf. on Acoustics, Speech, and Signal Processing*, 2013.

Bach, F. R. Consistency of trace norm minimization. *J. Mach. Learn. Res.*, 9:1019–1048, June 2008.

Bach, F. R. and Jordan, M. I. Kernel independent component analysis. *J. Mach. Learn. Res.*, 3:1–48, 2002.

Bengio, Y. and Delalleau, O. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621, 2009.

Blaschko, M. B. and Lampert, C. H. Correlational spectral clustering. In *CVPR*, 2008.

Chaudhuri, K., Kakade, S. M., Livescu, K., and Sridharan, K. Multi-view clustering via canonical correlation analysis. In *ICML*, 2009.

Choukri, K. and Chollet, G. Adaptation of automatic speech recognizers to new speakers using canonical correlation analysis techniques. *Speech Comm.*, 1:95–107, 1986.

Davis, S. B. and Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. Acoustics, Speech, and Signal Proc.*, 28(4):357–366, 1980.

De Bie, T. and De Moor, B. On the regularization of canonical correlation analysis. In *Proc. Int'l Conf. on Independent Component Analysis and Blind Source Separation*, 2003.

Dhillon, P., Foster, D., and Ungar, L. Multi-view learning of word embeddings via CCA. In *NIPS*, 2011.

Ek, C. H., Torr, P. H., , and Lawrence, N. D. Ambiguity modelling in latent spaces. In *MLMI*, 2008.

Haghighi, A., Liang, P., Berg-Kirkpatrick, T., and Klein, D. Learning bilingual lexicons from monolingual corpora. In *ACL-HLT*, 2008.

Hardoon, D. R., Szedmák, S., and Shawe-Taylor, J. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004.

Hardoon, D. R., Mourao-Miranda, J., Brammer, M., and Shawe-Taylor, J. Unsupervised analysis of fMRI data using kernel canonical correlation. *NeuroImage*, 37(4): 1250–1259, 2007.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Hinton, G. E., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, 18 (7):1527–1554, 2006.

Hotelling, H. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

Kakade, S. M. and Foster, D. P. Multi-view regression via canonical correlation analysis. In *COLT*, 2007.

Kim, T. K., Wong, S. F., and Cipolla, R. Tensor canonical correlation analysis for action classification. In *CVPR*, 2007.

Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., and Ng, A. Y. On optimization methods for deep learning. In *ICML*, 2011.

LeCun, Y. and Cortes, C. The MNIST database of handwritten digits, 1998.

Lewis, A. S. Derivatives of spectral functions. *Mathematics of Operations Research*, 21(3):576–588, 1996.

Mardia, K. V., Kent, J. T., and Bibby, J. M. *Multivariate Analysis*. Academic Press, 1979.

Melzer, T., Reiter, M., and Bischof, H. Nonlinear feature extraction using generalized canonical correlation analysis. In *ICANN*, 2001.

Montanarella, L., Bassami, M., and Breas, O. Chemometric classification of some European wines using pyrolysis mass spectrometry. *Rapid Communications in Mass Spectrometry*, 9(15):1589–1593, 1995.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. Multimodal deep learning. In *ICML*, 2011.

Nocedal, J. and Wright, S. J. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

Petersen, K. B. and Pedersen, M. S. The matrix cookbook, Nov 2012. URL http://www2.imm.dtu.dk/pubdb/p.php?3274.

Rudzicz, F. Adaptive kernel canonical correlation analysis for estimation of task dynamics from acoustics. In *ICASSP*, 2010.

Salakhutdinov, R. and Hinton, G. E. Deep Boltzmann machines. In *AISTATS*, 2009.

Sargin, M. E., Yemez, Y., and Tekalp, A. M. Audiovisual synchronization and fusion using canonical correlation analysis. *IEEE. Trans. Multimedia*, 9(7):1396–1403, 2007.

Slaney, M. and Covell, M. FaceSync: A linear operator for measuring synchronization of video facial images and audio tracks. In *NIPS*, 2000.

Srivastava, N. and Salakhutdinov, R. Multimodal learning with deep Boltzmann machines. In *NIPS*, 2012.

Vert, J.-P. and Kanehisa, M. Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA. In *NIPS*, 2002.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *ICML*. ACM, 2008.

Vinokourov, A., Shawe-Taylor, J., and Cristianini, N. Inferring a semantic representation of text via cross-language correlation analysis. In *NIPS*, 2003.

Westbury, J. R. *X-ray microbeam speech production database user's handbook*. Waisman Center on Mental Retardation & Human Development, U. Wisconsin, Madison, WI, version 1.0 edition, June 1994.