Stochastic Optimization for PCA and PLS

Raman Arora, Andrew Cotter, Karen Livescu and Nathan Srebro
Toyota Technological Institute at Chicago
6045 S. Kenwood Ave.
Chicago, Illinois 60637
Email: {arora,cotter,klivescu,nati}@ttic.edu

***Abstract*—We study PCA, PLS, and CCA as stochastic optimization problems, of optimizing a population objective based on a sample. We suggest several stochastic approximation (SA) methods for PCA and PLS, and investigate their empirical performance.**

I. Introduction

In this paper we consider the Principal Component Analysis (PCA), Partial Least Squares (PLS), and Canonical Correlation Analysis (CCA) problems as stochastic optimization problems, of optimizing an objective functional of an unknown distribution based on i.i.d. draws from the distribution.

In PCA, we consider an unknown source distribution $\mathcal{D}$ over $\mathbb{R}^d$ and would like to find the k-dimensional subspace maximizing the (uncentered) variance of $\mathcal{D}$ inside the subspace. For PLS and CCA we consider covariances and correlations of a joint distribution over a pair of vectors; this will be formalized in Section IV. In all cases, our true objective refers to $\mathcal{D}$ itself, and we can never actually calculate it, instead estimating the objective based on i.i.d. samples. We focus on the "data-laden" regime, where we have access to effectively as many samples as we would like, and the bottleneck is the required runtime to process these samples. That is, we focus on the runtime required to achieve a good objective value.

The straightforward approach is "Sample Average Approximation" (SAA), where we collect a sample of data points, and then optimize an empirical version of the objective *on the sample* using standard deterministic techniques (in this case linear algebra). In the case of uncentered PCA, this amounts to computing the empirical second-moment matrix of the sample, and then seeking the best rank-k approximation to it, e.g. by computing the leading components of its eigendecomposition. The success of this approach is measured not by how well we approximate the *empirical* second-moment matrix, but rather how well the subspace we obtain captures the unknown source distribution (i.e. the *population* second-moment matrix).

The alternative, which we advocate here, is a "Stochastic Approximation" (SA) approach. A SA algorithm is an iterative algorithm, where in each iteration a single sampled point is used to perform an update, as in Stochastic Gradient Descent (SGD, the classic stochastic approximation algorithm). In the context of PCA this means iteratively using vectors sampled from $\mathcal{D}$ to update the subspace being considered.

Stochastic approximation has been shown to be computationally preferable to statistical average approximation (i.e. to "batch" methods) both theoretically and empirically for learning [1, 2] and more broadly for stochastic optimization [3]. Accordingly, SA approaches, mostly variants of SGD, are often the methods of choice for many learning problems, especially when very large data sets are available [4, 5, 6].

Our goal is to formalize PCA and PLS as stochastic optimization problems, study the runtime needed to achieve an accuracy goal *on the population objective*, and investigate different SA approaches. We also consider CCA as a stochastic optimization problem, and discuss why it is *not* amenable to the same type of stochastic optimization approaches as we use for PCA and PLS. In particular, we present three SA approaches to PCA—a stochastic power method related to the popular generalized Hebbian algorithm [7], a novel truncated incremental SVD approach, and an adaptation of an online method by Warmuth and Kuzmin [8]. We then adapt the methods to PLS and study their empirical behavior for both PCA and PLS, as well as compare them to the SAA (empirical optimization) approaches to these problems.

II. Stochastic Optimization for PCA

We consider PCA as a problem of finding the maximal (uncentered) variance k-dimensional subspace with respect to a *distribution* over $x \in R^d$. Parameterizing the subspace through a matrix $U \in \mathbb{R}^{d\times k}$ with columns spanning it, we consider the following specification of PCA:

$$\begin{aligned} &\underset{U}{\text{maximize}} : \mathbb{E}_x \left[\text{tr}(U^T x x^T U)\right] \\ &\text{subject to} : U^T U \preceq I, \end{aligned} \tag{1}$$

where the expectation is taken with respect to the distribution of interest. Note that at the optimum, all eigenvalues of U would be equal to one, and the columns of U would be orthonormal, but in order to obtain a convex constraint without changing the optimum, we consider all U with eigenvalues *at most* one to be feasible.

The situation we consider here, which we argue is the typical situation in practice, is where we do not have direct knowledge of the distribution over $x \in \mathbb{R}^d$ and so cannot exactly calculate the (population) objective specified in Problem 1, let alone optimize it. Instead, we only have access to i.i.d. samples from this distribution—these can be thought of as "training samples". The regime we consider in this paper is where we have an essentially unlimited supply of training examples, and would like to obtain an ϵ-suboptimal solution in the least possible runtime.

One possible approach to optimizing Problem 1 is through Sample Average Approximation (SAA): we collect T i.i.d. samples $x_1, \ldots, x_T$ and solve the empirical version of Problem 1, where the expectation is replaced with its empirical (sample average) approximation $\frac{1}{T}\sum_i \text{tr}(U^T x_i x_i^T U)$. This can be done by calculating the sample second-moment matrix $\hat{\Sigma} = \frac{1}{T}\sum_{t=1}^{T} x_t x_t^T$, and taking U to have columns equal to the top k eigenvectors of $\hat{\Sigma}$.

Let us consider the computational cost of the SAA approach: The eigendecomposition of $\hat{\Sigma}$ can be computed in time $O(d^3)$ using the algebraic QR algorithm, or via iterative methods [9, 10]. But the more significant cost is calculating $\hat{\Sigma}$ itself, which requires $O(Td^2)$ time and $O(d^2)$ memory. Now, even if the empirical problem is solved exactly, recall that we are interested in the population objective of Problem 1, and so using a finite sample T we obtain only an approximate solution. The runtime of the SAA approach is then $O(T_\epsilon d^2)$ where T_ϵ is the number of samples required to ensure the empirical minimizer is ϵ-suboptimal.

Recently, Tropp has developed a number of generalizations of familiar probabilistic inequalities to the matrix setting. His matrix Hoeffding bound [11, Theorem 1.3] can be used to show that with probability $1-\delta$:

$$T_\epsilon = O\left(\frac{M^2 k^2}{\epsilon^2}\log\left(\frac{d}{\delta}\right)\right), \tag{2}$$

where $\left\|xx^T - \Sigma\right\|_2 \leq M$ (this is the spectral norm) with probability 1. Although we believe that this bound could be improved (particularly its k-dependence), it provides a natural baseline for evaluating the rate of convergence of other approaches for solving Problem 1.

An alternative is a Stochastic Approximation (SA) approach, where samples x_i are considered sequentially, and only a simple computation, perhaps linear in the size of U (i.e. in dk), is performed at each iteration in order to update U. Even if such an approach might require more samples to obtain an ϵ-suboptimal solution, the hope is that the shorter runtime-per-sample will result in lower total runtime being required to obtain a solution of the same quality.

In the next section, we consider three stochastic approximation approaches to PCA. In Section VII, we will empirically compare both the number of samples and the overall runtime required to obtain a good solution to Problem 1, both with the SA approaches and with the SAA approach discussed above. As we will see, although the number of samples required by the SA approaches is higher, the computational cost is substantially lower.

III. PCA Algorithms

In this Section we present three stochastic approximation approaches to Problem 1. In Section III-A we present the stochastic power method as a stochastic gradient descent approach, and discuss how it could be implemented more efficiently while still maintaining the same sequence of updates. In Section III-B we derive a novel truncated incremental approach, which relies on recent work on incremental SVD. Finally, in Section III-C we consider an efficient implementation of Warmuth and Kuzmin's online PCA algorithm [8] as another alternative.

A. Stochastic Power Method and Variants

For convex optimization problems, stochastic gradient descent is a simple and often highly efficient optimization technique. The PCA objective function is convex, as is the constraint (as in Equation 1), but as the goal is *maximization* of this objective, the formulation of Equation 1 is *not* convex as an optimization problem. However, gradient descent is still a viable algorithm. If Σ were known exactly, then the gradient of the PCA objective function $\text{tr}(U^T\Sigma U)$ with respect to U would be $2\Sigma U$, leading one to consider updates of the form $U^{(t+1)} = \mathcal{P}_{\text{orth}}\left(U^{(t)} + \eta_t \Sigma U^{(t)}\right)$, where $\mathcal{P}_{\text{orth}}(U)$ performs a projection with respect to the spectral norm of UU^T onto the set of $d \times d$ matrices with k eigenvalues equal to 1 and the rest 0 (calling this a "projection" is a slight abuse of terminology, since it is UU^T which is projected, not U itself).

If one analytically performs an exact line search in order to determine the best positive η, then one finds that the optimal step size is essentially infinite, yielding the update $U^{(t+1)} = \mathcal{P}_{\text{orth}}\left(\Sigma U^{(t)}\right)$—this is the power method. As an alternative to performing an exact line search along the direction of the overall gradient, one could instead notice that $\mathbb{E}\left[xx^T\right] = \Sigma$, and take a step in the stochastic gradient direction:

$$U^{(t+1)} = \mathcal{P}_{\text{orth}}\left(U^{(t)} + \eta_t x_t x_t^T U^{(t)}\right). \tag{3}$$

Notice that, whereas calculating ΣU would require $O(kd^2)$ operations, finding $xx^T U$ requires only $O(kd)$. The renormalization step represented by $\mathcal{P}_{\text{orth}}$ (which can be performed in $O(k^2 d)$ operations using, e.g., the Gram-Schmidt procedure) does not affect the correctness of the solution—even if we do not perform it at all, each iterate will span the same subspace as it would have otherwise. To see this, suppose that we *do* renormalize $U^{(t)}$ after each iteration. We may then write $U^{(t)} = Q^{(t)}R^{(t)}$ with $Q^{(t)}$ having orthonormal columns and $R^{(t)}$ being a nonsingular $k \times k$ matrix (this is not necessarily a QR factorization, although it may be, if one renormalizes using Gram-Schmidt). The matrix $Q^{(t)}$ is then the renormalized version of $U^{(t)}$. With this representation of renormalization, the 1-step SGD update of Equation 3 is:

$$\begin{aligned} U^{(t+1)} &= Q^{(t)} + \eta_t x_t x_t^T Q^{(t)}, \\ U^{(t+1)}R^{(t)} &= U^{(t)} + \eta_t x_t x_t^T U^{(t)}. \end{aligned}$$

From this equation, it is easy to prove by induction on t that if $V^{(t)}$ is the sequence of iterates which would result if renormalization was *not* performed, then $V^{(t)} = Q^{(t)}R^{(t)}R^{(t-1)}\cdots R^{(1)}$. Because $R^{(t)}R^{(t-1)}\cdots R^{(1)}$ is a product of nonsingular matrices, it is nonsingular, showing that $V^{(t)}$ and $Q^{(t)}$ span the same subspace.

As a result of this observation, renormalization may be performed for purely numerical reasons, and only very infrequently. Hence, the computational cost of renormalization may be ignored, showing that performing T iterations of SGD costs only $O(Tkd)$ operations and $O(kd)$ memory, both of

which are better by a factor of d/k than merely *calculating* the empirical second-moment matrix over T samples (costing $O(Td^2)$ operations and $O\left(d^2\right)$ memory). For small k and large d, this represents an enormous potential performance difference over non-stochastic linear algebra-based methods.

Although not presented as instances of SGD, there are a number of algorithms in the literature that perform precisely the above SGD update, differing only in how they renormalize. For example, Oja and Karhunen [12] perform Gram-Schmidt orthonormalization after every iteration, while the popular generalized Hebbian algorithm [7], which was later generalized to the kernel PCA setting by Kim et al. [13], performs a partial renormalization. Both of these algorithms converge with probability 1 (under certain conditions on the distribution of x and step sizes η_t). However, the *rate* of convergence is not known.

B. Incremental PCA

Another approach to PCA in the stochastic setting is inspired by an incremental algorithm for finding the singular value decomposition (SVD) [14], although the linear algebra used by our approach is more similar to the technique of Bunch et al. [15]. The problem of incremental SVD has been studied recently in many areas, with application to face recognition [16], scalable recommender systems [17], time-series segmentation [18], and data imputation [14].

Whereas Brand [14] is interested in the SVD of a data matrix $X = [x_1, x_2, \ldots, x_T]$, we are interested in the eigen-decomposition of the matrix of second moments $\mathbb{E}[xx^T]$. Like Brand, we consider the setting where the examples are processed one at a time, resulting in a rank-one update to the unnormalized empirical second-moment matrix at each iteration. The update we propose can be expressed as:

$$C^{(t)} = \mathcal{P}_{\text{rank-}k}\left(C^{(t-1)} + x_t x_t^T\right). \tag{4}$$

The projection $\mathcal{P}_{\text{rank-}k}$ projects its argument onto the set of rank-k matrices with respect to the spectral norm. This can be accomplished by setting all but the k largest eigenvalues to 0.

The update of Equation 4 can be performed efficiently. Suppose that $C^{(t-1)}$ is a rank-ℓ approximation to the matrix $\sum_{s=1}^{t-1} x_s x_s^T$ with its eigendecomposition being $C^{(t-1)} = USU^T$, where $U \in \mathbb{R}^{d\times\ell}$ and $U^TU = I$. Given a new observation, x_t, define $\hat{x} = U^T x_t$ as the coefficients of the projection of x_t onto the columns of U, and $x_\perp = x_t - UU^T x_t$ as its orthogonal component. Then the updated matrix $\tilde{C}^{(t)} = C^{(t-1)} + x_t x_t^T$ of Equation 4 can be written as:

$$\begin{bmatrix} U & \dfrac{x_\perp}{\|x_\perp\|_2} \end{bmatrix} \underbrace{\begin{bmatrix} S + \hat{x}\,\hat{x}^T & \|x_\perp\|_2\,\hat{x} \\ \|x_\perp\|_2\,\hat{x}^T & \|x_\perp\|_2^2 \end{bmatrix}}_{Q \in \mathbb{R}^{(\ell+1)\times(\ell+1)}} \begin{bmatrix} U^T \\ x_\perp^T/\|x_\perp\|_2 \end{bmatrix}.$$

Notice that we have named the middle matrix in the above expression Q. The eigendecomposition of $Q = U'S'U'^T$ may be found in $O\left(k^3\right)$ time, and the eigendecomposition of $\tilde{C}^{(t)}$ is $\tilde{U}\tilde{S}\tilde{U}^T$, where:

$$\tilde{U} = \begin{bmatrix} U & \dfrac{x_\perp}{\|x_\perp\|_2} \end{bmatrix} U',\ \tilde{S} = S'.$$

If the rank of $\tilde{C}^{(t)}$ grows beyond k, then we truncate $\tilde{S}$ and $\tilde{U}$ to retain the top k eigenvalues and eigenvectors, resulting in the iterate $C^{(t)}$.

One advantage of the incremental approach is its low space complexity of $O(kd)$, compared to $O(d^2)$ for classical approaches to PCA. The dominant computation in our update is the matrix multiplication defining $\tilde{U}$, which has a cost of $O(k^2 d)$ operations, leading to an overall runtime of $O(Tk^2 d)$, which is worse by a factor of k than the SGD approach of Section III-A, but is better by a factor of d/k^2 than the cost of calculating the empirical second-moment matrix (under the assumption that $k \ll d$). The incremental approach has another advantage over SGD: it is *parameter-free*, whereas the use of SGD may require tuning of the step-size parameter η_t.

There do exist situations in which this algorithm fails to converge. For example, if the data are drawn from the distribution which samples $(3, 0)$ with probability $1/3$ and $(0, 2)$ with probability $2/3$, and we use the incremental algorithm to search for a 1-dimensional maximal subspace, then it will converge to $(1, 0)$ with probability $5/9$, despite the fact that the maximal eigenvector is $(0, 1)$. This example relies on the data being orthogonal (a situation which is unlikely to arise in practice), but does illustrate that the convergence properties of the incremental approach are not well-understood.

C. Online PCA

Warmuth and Kuzmin [8] introduced an *online* algorithm for solving the PCA problem. A full derivation of their algorithm is beyond the scope of this paper, but we will give a high-level overview of their algorithm, and describe a novel technique for improving its computational efficiency dramatically.

As we have seen, one may formulate PCA as the optimization problem of finding a $U \in \mathbb{R}^{d\times k}$ with orthonormal columns maximizing $\mathbb{E}_x\left[\text{tr}(U^T xx^T U)\right]$. Equivalently, one could seek not a matrix U of k column vectors spanning the maximal subspace, but instead a matrix U' of $d-k$ columns spanning the *minimal* subspace. From this matrix, the orthogonal complement (U) could easily be derived. We could go further, and seek instead a *projection matrix* M onto the $d-k$ dimensional minimal subspace (we may think of M as satisfying $M = U'U'^T = I - UU^T$). Because M is a rank $d-k$ projection matrix, it must have have exactly $d-k$ eigenvalues equal to 1, and k equal to 0—this will be a constraint which is imposed during optimization. Unfortunately, this is not a convex constraint, but if we *relax* it by taking the convex hull, then the result *is* a convex

optimization problem:

$$\begin{aligned} &\underset{M}{\text{minimize}} : \mathbb{E}_x \left[\text{tr}(Mxx^T)\right] \\ &\text{subject to} : M \succeq 0,\ \|M\|_2 \leq \frac{1}{d-k},\ \text{tr}(M) = 1. \end{aligned} \tag{5}$$

Here, $\|\cdot\|_2$ is the spectral norm, and we have scaled M by $1/(d-k)$. This is precisely the PCA formulation proposed by Warmuth and Kuzmin [8]. Their update rule (which is targeted towards the online setting, rather than the easier stochastic setting which we consider) is an instance of the Mirror Descent algorithm [19], with distance generating function $\Psi(M) = \text{tr} \ln M$ (the von Neumann entropy) and the Frobenius inner product and norm:

$$M^{(t+1)} = \mathcal{P}_{\text{RE}} \left(\exp \left(\ln M^{(t)} - \eta_t x_t x_t^T \right) \right). \tag{6}$$

These are *matrix* logarithms and exponentials. The projection $\mathcal{P}_{\text{RE}}$ onto the constraints is performed with respect to the quantum relative entropy, which turns out to be a straightforward operation: let k' be the largest number such that setting the largest $d-k'$ eigenvalues of M to $1/(d-k)$, and scaling the remaining k' eigenvalues so as to satisfy the normalization constraint $\text{tr}(M) = 1$, results in all eigenvalues of M being bounded above by $1/(d-k)$. Then perform this capping-and-scaling. Please see Warmuth and Kuzmin [8] for details.

Of the stochastic algorithms that we consider, only Warmuth and Kuzmin's enjoys a satisfactory convergence guarantee. Using a certain constant step size $\eta_t = \eta$, assuming that $\|x_t\|_2 \leq 1$ uniformly, and for any M^* satisfying the constraints of Problem 5, Warmuth and Kuzmin [8, Equation 5.1] gives that after performing T iterations, where T satisfies:

$$T \geq O\left(\left(\frac{(d-k)\,\text{tr}\left(M^*\Sigma\right) + \epsilon}{\epsilon} \right) \frac{k \log \frac{d}{k}}{\epsilon} \right),$$

the iterates $M^{(t)}$ will satisfy:

$$\mathbb{E}\left[\frac{1}{T} \sum_{t=1}^{T} \text{tr}\left((d-k)\, M^{(t)} \Sigma \right) \right] - \text{tr}\left((d-k)\, M^* \Sigma \right) \leq \epsilon, \tag{7}$$

where M^* is the optimum of Problem 5, and both M^* and $M^{(t)}$ are scaled by $d-k$ to compensate for the fact that the objective of Problem 5 was scaled by $1/(d-k)$ relative to the original PCA objective.

Equation 7 is not directly comparable to the SAA bound of Equation 2, due to its dependence on M^*. In the "optimistic" setting in which the $d-k$ dimensional minimal subspace contains total variance on the order of ϵ, the first term of Equation III-C is of order 1, and this bound is superior to Equation 2 by a factor of k/ϵ. The bounds are comparable when this $d-k$ dimensional subspace contains total variance of order k, and in more "pessimistic" settings the bound on the SAA algorithm is superior.

The biggest drawback of Warmuth and Kuzmin's algorithm is computational cost. As written, performing a single iteration (Equation 6) requires two full-rank eigendecompositions: one for the matrix logarithm, and another for the exponential and projection. As was pointed out by Tsuda et al. [20], one may reduce this cost by maintaining an up-to-date eigendecomposition of M throughout a run of the algorithm. Each update to this eigendecomposition will take the form of either a rank-one update (when we add $\eta_t x_t x_t^T$) or a simple operation on the eigenvalues (the matrix logarithm and exponential, as well as the projection).

This optimization only partially addresses the algorithm's performance shortcomings, since M is still a full-rank $d \times d$ matrix. There is one more significant property of M which remains to be exploited: the projection step will set many of the eigenvalues of M to be exactly $1/(d-k)$. Let k' be the number of eigenvalues which are *less* than $1/(d-k)$, so that M has $d-k'$ repeated eigenvalues. Using the same technique as the incremental PCA algorithm of Section III-B, one may find the eigendecomposition of $M + \eta x x^T$ given an eigendecomposition of M in $O\left(k'^2 d\right)$ time. Furthermore, the eigenvectors corresponding to these repeated eigenvalues need not be stored, reducing the memory cost to $O\left(k'd\right)$. Of course, these results depend crucially on the value of k', which varies from iteration to iteration, and could potentially be as small as $k+1$ or as large as $d-1$. In practice, however, this change leads to significant savings.

One final issue with Warmuth and Kuzmin's algorithm is that it solves not the true PCA objective, but instead a *relaxation* of it. The eigenvalues of the state matrices $M^{(t)}$ are nonnegative and sum to 1, which suggests finding a minimal subspace of rank $d-k$ via sampling, and taking its orthogonal component to be our rank-k maximal subspace. Indeed, this is precisely the solution suggested by Warmuth and Kuzmin. In our experiments (Section VII), we use the less-"correct" but simpler (and empirically superior) approach of taking the eigenvectors corresponding to the minimal k eigenvalues of $M^{(T)}$. We use only the last iterate $M^{(T)}$, not the average over all iterates, as would be suggested by Equation 7.

IV. Stochastic optimization for PLS

Many machine learning problems benefit from multiple "views" of the data, possibly from different measurement modalities. In such multi-view learning problems, a common representation of the two views is provided by the shared semantic space. A common approach to extracting this space is through canonical correlation analysis (CCA), which finds pairs of maximally correlated projections of the data in the two views. CCA has been successfully applied to various tasks in speech [21, 22, 23, 24, 25, 26], natural language processing [27], and computer vision [28, 29]. A closely related technique is Partial Least Squares (PLS), which finds pairs of maximally *covarying* projections of the data in the two views. We begin by considering PLS, and return to CCA in Section VI.

In order to formulate PLS as a stochastic optimization problem, consider a joint distribution over pairs of vectors $x \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}^{d_y}$. A k-dimensional PLS solution can be parameterized by a pair of matrices $U \in \mathbb{R}^{d_x \times k}$ and

$V \in \mathbb{R}^{d_y \times k}$, where the corresponding columns of U and V represent corresponding covarying directions. The PLS problem can now be expressed as:

$$\begin{aligned} &\underset{U,V}{\text{maximize}} : \mathbb{E}_{x,y}\left[\text{tr}\left(U^T x y^T V\right)\right] \\ &\text{subject to} : U^T U = I,\ V^T V = I. \end{aligned} \tag{8}$$

The columns of U and V are singular vectors of Σ_{XY}. Like the PCA objective and most other learning problems, PLS is an optimization of an expectation subject to fixed constraints, and is therefore amenable to a stochastic approximation approach.

V. PLS Algorithms

We now show how the SA methods presented in Section III can be modified to solve the (uncentered) PLS Problem 8.

A. SGD for PLS

It is fairly straightforward to extend the SGD algorithm of Section III-A to PLS. The difference is that we now observe a pair of examples x_t, y_t at each iteration, with $\mathbb{E}[xy^T] = \Sigma_{XY}$, and seek to maximize $\text{tr}\left(U^T \Sigma_{XY} V\right)$, which has gradients $\Sigma_{XY} V$ with respect to U and $\Sigma_{XY}^T U$ with respect to V. This suggests the following stochastic gradient update:

$$\begin{aligned} U^{(t+1)} &= \mathcal{P}_{\text{orth}}\left(U^{(t)} + \eta_t x_t y_t^T V^{(t)}\right), \\ V^{(t+1)} &= \mathcal{P}_{\text{orth}}\left(V^{(t)} + \eta_t y_t x_t^T U^{(t)}\right), \end{aligned}$$

where, as in Section III-A, the projection $\mathcal{P}_{\text{orth}}$ orthonormalizes the columns of its argument, and needs to be performed only infrequently, for purely numerical reasons.

The space complexity of the resulting algorithm is the sum of the sizes of U and V: $O\left(k\left(d_x + d_y\right)\right)$. Neglecting the cost of the projection, the computational cost is dominated by the matrix-vector multiplications of the update equation, costing $O\left(k\left(d_x + d_y\right)\right)$ per iteration, for a total of $O\left(Tk\left(d_x + d_y\right)\right)$. As before, these are superior to the $O\left(d_x d_y\right)$ space complexity and $O\left(T d_x d_y\right)$ computational cost of calculating $\hat{C}_{XY}$.

B. Incremental PLS

As in Section III-B, one may perform an incremental PLS update by maintaining a rank-k estimate $C_{XY}^{(t-1)}$ of the SVD of the empirical cross-second-moment matrix $\sum_{s=1}^{t-1} x_s y_s^T$, and performing a rank-one update and projection after observing each sample pair x_t, y_t, according to $C_{XY}^{(t)} = \mathcal{P}_{\text{rank-}k}\left(C_{XY}^{(t-1)} + x_t y_t^T\right)$, where $\mathcal{P}_{\text{rank-}k}$ projects onto the set of rank-k matrices with respect to the spectral norm.

The efficient implementation of this update follows broadly the same lines as that for PCA. Supposing that $C_{XY}^{(t-1)}$ has the rank-ℓ SVD USV^T, we define:

$$\begin{aligned} \hat{x} &= U^T x_t, \quad x_\perp = x_t - U U^T x_t, \\ \hat{y} &= V^T y_t, \quad y_\perp = y_t - V V^T y_t, \end{aligned}$$

The matrix $\tilde{C}_{XY}^{(t)} = C_{XY}^{(t-1)} + x_t y_t^T$ can then be written as:

$$\begin{bmatrix} U & \dfrac{x_\perp}{\|x_\perp\|} \end{bmatrix} \underbrace{\begin{bmatrix} S + \hat{x}\,\hat{y}^T & \|y_\perp\|\,\hat{x} \\ \|x_\perp\|\,\hat{y}^T & \|x_\perp\|\,\|y_\perp\| \end{bmatrix}}_{Q \in \mathbb{R}^{(\ell+1)\times(\ell+1)}} \begin{bmatrix} V^T \\ y_\perp^T / \|y_\perp\| \end{bmatrix}.$$

Finding the SVD of $\tilde{C}_{XY}^{(t)}$ requires an "inner" SVD on the smaller matrix Q, with a computational cost of $O(\ell^3)$. Given the SVD $Q = U' S' V'^T$, the SVD of $\tilde{C}_{XY}^{(t)}$ is $\tilde{U}\tilde{S}\tilde{V}^T$, where:

$$\tilde{U} = \begin{bmatrix} U & \dfrac{x_\perp}{\|x_\perp\|} \end{bmatrix} U',\ \tilde{S} = S',\ \tilde{V} = \begin{bmatrix} V & \dfrac{y_\perp}{\|y_\perp\|} \end{bmatrix} V'.$$

Once more, we ensure that the rank of $C_{XY}^{(t)}$ does not exceed k by taking only the top k singular values and vectors of $\tilde{C}_{XY}^{(t)}$.

This algorithm has the same space complexity as SGD: $O\left(k\left(d_x + d_y\right)\right)$. The computational cost is dominated by the matrix multiplications defining $\tilde{U}$ and $\tilde{V}$, costing $O\left(k^2\left(d_x + d_y\right)\right)$ operations, which is a factor of k worse than SGD, but still better, for sufficiently small k, than that of calculating the empirical cross-second-moment matrix.

C. Online PLS

Adapting the online algorithm of Warmuth and Kuzmin [8] to work on the PLS objective is less straightforward than it is for either SGD or the incremental algorithm, since it works in the matrix log domain, and therefore depends on symmetry. We may, however, symmetrize the PLS objective by making use of a self-adjoint dilation [11] of the matrix Σ_{XY}:

$$\Gamma = \begin{bmatrix} 0 & \Sigma_{XY} \\ \Sigma_{YX} & 0 \end{bmatrix} = \mathbb{E}_{x,y}\begin{bmatrix} 0 & xy^T \\ yx^T & 0 \end{bmatrix}.$$

Provided that Σ_{XY} has no repeated nonzero singular values, the nonzero eigenvalues of this matrix will be the singular values of Σ_{XY} and their negations, with the corresponding matrix of eigenvectors being (up to sign differences):

$$W = \frac{1}{\sqrt{2}}\begin{bmatrix} U & U \\ V & -V \end{bmatrix},$$

where U and V are the left and right singular vectors of Σ_{XY}, respectively. Hence, if we let $W_k \in \mathbb{R}^{(d_x+d_y)\times k}$ be a matrix with the top $k \leq d_x, d_y$ eigenvectors of Γ in its columns, then $\sqrt{2}$ times the first d_x rows of W_k will be the top k left singular vectors of Σ_{XY}, and likewise $\sqrt{2}$ times the last d_y rows of W_k will be the top k right singular vectors of Σ_{XY}. This suggests that we modify Equation 5 to be:

$$\begin{aligned} &\underset{M}{\text{minimize}} : \mathbb{E}_{x,y}\left[\text{tr}\left(M\begin{bmatrix} 0 & xy^T \\ yx^T & 0 \end{bmatrix}\right)\right] \\ &\text{subject to} : M \succeq 0,\ \|M\|_2 \leq \frac{1}{d-k},\ \text{tr}\left(M\right) = 1, \end{aligned}$$

which leads us to change the update of Equation 6 to:

$$M^{(t+1)} = \mathcal{P}_{\text{RE}}\left(\exp\left(\ln M^{(t)} - \eta_t Z_t\right)\right),$$

where:

$$Z_t = \begin{bmatrix} 0 & x_t y_t^T \\ y_t x_t^T & 0 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_t \\ y_t \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix}^T - \frac{1}{2} \begin{bmatrix} x_t \\ -y_t \end{bmatrix} \begin{bmatrix} x_t \\ -y_t \end{bmatrix}^T.$$

Notice that Z_t is a rank-2 matrix. The analysis of Warmuth and Kuzmin no longer applies to this modification of their algorithm, since Γ and Z_t have negative eigenvalues. However, the *interpretation* of the algorithm as exponentiated stochastic gradient descent is unchanged. The implementation of the algorithm is almost identical: the only difference is that a rank-2 update is performed at every iteration, instead of rank-1.

As we did in Section III-C, we find a rank-k maximal subspace by taking the eigenvectors corresponding to the k minimal eigenvalues of $M^{(T)}$. There is one additional wrinkle, however: although the top d_x and bottom d_y rows of the matrix containing the *optimal* k eigenvectors of Γ immediately give a SVD of Σ_{XY}, for *suboptimal* solutions it may not be the case that the resulting U and V will have orthogonal columns. In our experiments, we resolve this by the simple expedient of taking the orthonormalized $U\left(U^TU\right)^{-1/2}$ and $V\left(V^TV\right)^{-1/2}$ as our approximate singular vectors.

VI. Stochastic optimization for CCA

We now return to the question of formulating CCA as a stochastic optimization problem. We again consider a joint distribution over pairs of vectors $x \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}^{d_y}$. As in PLS, a k-dimensional CCA solution is parameterized by a pair of matrices $U \in \mathbb{R}^{d_x \times k}$ and $V \in \mathbb{R}^{d_y \times k}$, where the corresponding columns of U and V now represent corresponding *correlated* directions. The CCA problem can be expressed as:

$$\begin{aligned} \underset{U,V}{\text{maximize}} :& \mathbb{E}_{x,y}\left[\text{tr}\left(U^T x y^T V\right)\right] \\ \text{subject to} :& \mathbb{E}_x\left[U^T x x^T U\right] = I,\ \mathbb{E}_y\left[V^T y y^T V\right] = I. \end{aligned} \tag{9}$$

It is straightforward to show [28] that the CCA optimum can be obtained from a generalized eigenvalue problem, where the u_k are eigenvectors of $\Sigma_{XX}^{-1}\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}$ with corresponding $v_k = \Sigma_{YY}^{-1}\Sigma_{YX}u_k$, where Σ_{XY} is the cross-second-moment matrix between X and Y and Σ_{XX}, Σ_{YY} are the matrices of second moments of X and Y.

This suggests the immediate SAA approach where the matrices Σ_{XX}, Σ_{XY} and Σ_{YY} are replaced with their empirical (sample based) approximations, and the generalized eigenvalue problem is solved. This approach is applicable to both CCA and PLS, and is frequently applied to both.

A more careful look at the CCA problem 9 reveals that this is a difficult stochastic optimization problem, and that many stochastic approximation approaches might not be valid here. The difficulty stems from the fact that unlike the PCA and PLS problems, and many other stochastic optimization problems encountered in learning, the constraints also involve the unknown distribution. That is, not only can we not evaluate the objective value precisely, but we also cannot know which matrices are feasible. In order to appreciate this difficulty, consider the simple $k = 1$ case, in which we seek the two most correlated directions u and v. The normalization constraints can be removed and instead we can normalize u and v in the objective by dividing by the square roots of the second moments of the directions u and v, obtaining the following population objective:

$$\rho(u^T x, v^T y) = \frac{\mathbb{E}_{x,y}\left[u^T x y^T v\right]}{\sqrt{\mathbb{E}_x\left[u^T x x^T u\right]}\ \sqrt{\mathbb{E}_y\left[v^T y y^T v\right]}}. \tag{10}$$

This is now an unconstrained optimization problem, but the objective is no longer an expectation, but rather a ratio of expectations. This creates a difficult situation and departs significantly from the typical stochastic approximation scenario. For example, it is not at all clear how to obtain unbiased estimates of the gradient of Equation 10.

In this regard the PLS problem seems much more amenable to stochastic approximation approaches—it is, like the PCA objective and most other learning problems, an optimization of an expectation subject to fixed constraints. We thus focus on PLS, rather than CCA, in this initial study.

We note that CCA is equivalent to PLS after a normalization by the matrices Σ_{XX} and Σ_{YY}, i.e. to PLS on the transformed $\tilde{x} = \Sigma_{XX}^{-1/2}x$ and $\tilde{y} = \Sigma_{YY}^{-1/2}y$. Thus, in applications where it is feasible to normalize the data in this way (i.e., Σ_{XX} and Σ_{YY} are identity matrices, are known, or have a dominant diagonal that can be relatively easily estimated), CCA can be reduced to PLS by normalization.

VII. Experimental Results

In this section we compare the performance of the algorithms of Sections III and V in terms of progress made on the objective as a function of iteration number and CPU runtime.

For PCA experiments we use the MNIST dataset, consisting of $70,000$ binary images of handwritten digits. For PLS experiments, we use the Wisconsin X-Ray Micro-Beam (XRMB) data, consisting of acoustic and articulatory measurements of English speech [30]. We use roughly $225,000$ examples from four speakers (JW11, JW13, JW24, JW30).

All experiments include pre-normalization, consisting of mean-centering the feature vectors and then dividing each coordinate by its standard deviation times the square root of the length of the feature vector.

Our implementations are written in Matlab. In order to ensure a fair comparison with the parameter-free incremental algorithm, we deliberately made little effort to find optimal step sizes for SGD and Warmuth and Kuzmin's algorithms, choosing $\eta_t = 1/\sqrt{t}$ uniformly for all experiments. All algorithms were run for only one "pass" over the training data, resulting in some algorithms being far from convergence at termination, but ensuring that all training samples were drawn independently from the population.

Our experiments compare performance in terms of the objective function value. Because we cannot evaluate the true population objective for Problems 1 and 8, we instead approximate them by evaluating on a held-out testing sample

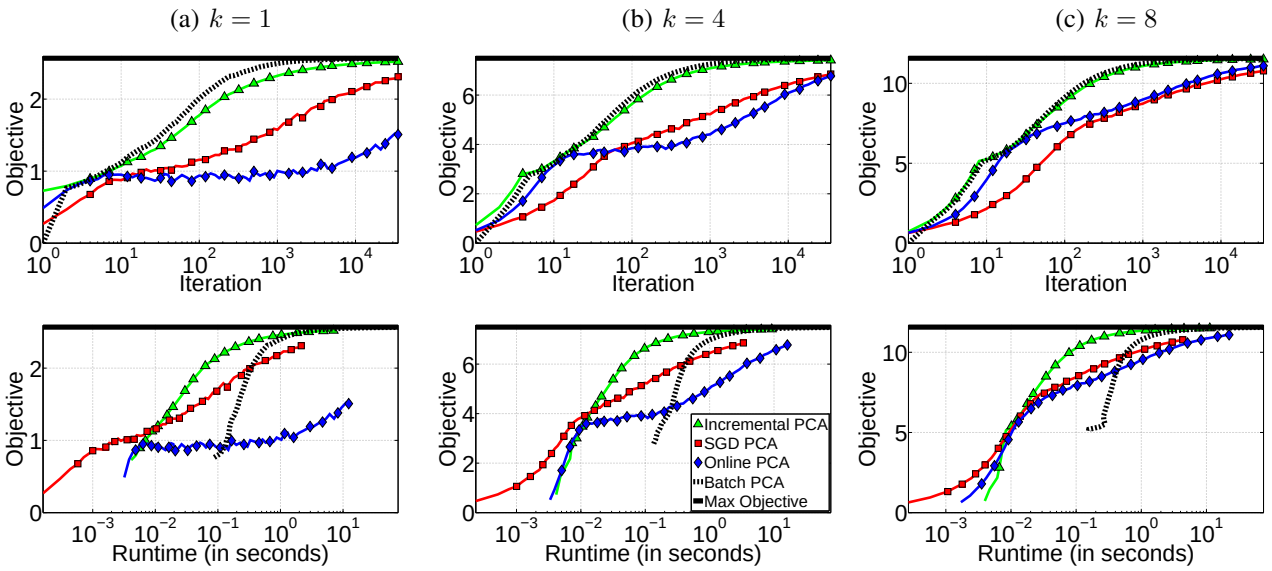

Fig. 1. Comparison of the incremental, SGD and online algorithms for stochastic PCA optimization on the MNIST dataset, in terms of the objective value as a function of iteration (top row) and as a function of CPU runtime (bottom row). The "Batch" curve is for the Sample Average Approximation (SAA) approach.

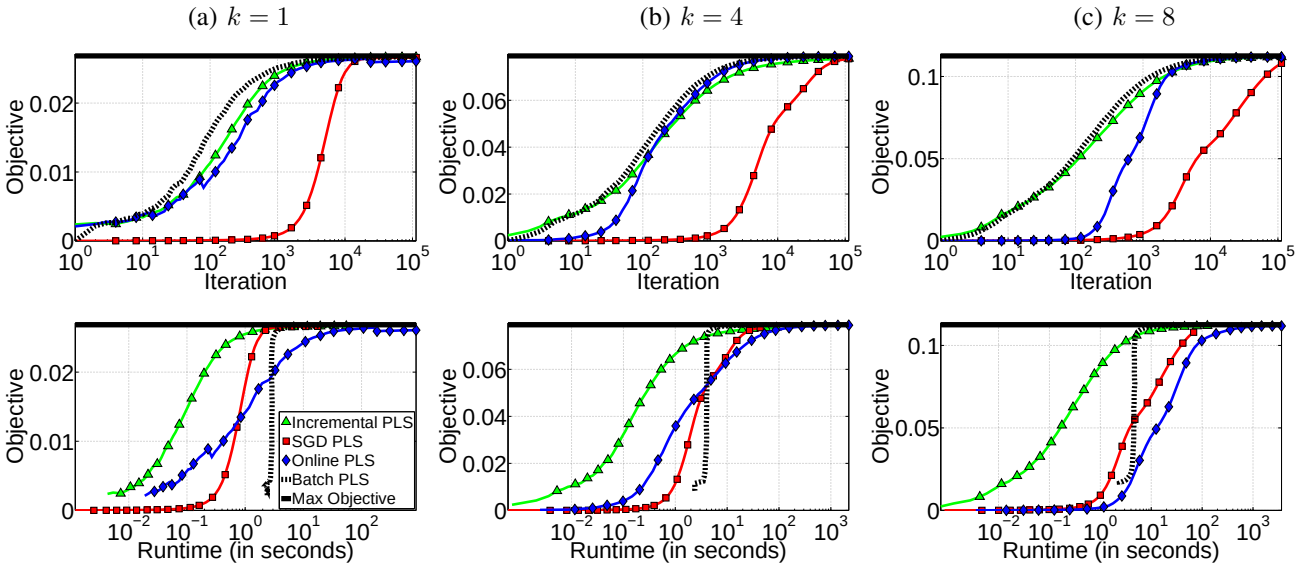

Fig. 2. Comparison of the incremental, SGD and online algorithms for stochastic PLS optimization on the XRMB dataset, in terms of the objective value as a function of iteration (top) and as a function of CPU runtime (bottom).

(half of the dataset, with the other half being used for training). All results are averaged over 50 random train/test splits.

Figures 1 and 2 show the PCA and PLS objectives, respectively, as a function of the number of samples processed (iterations) as well as CPU runtime, for ranks $k = 1$, 4 and 8. As expected, SGD is the fastest, but also makes the least progress, per iteration. The online algorithm makes better progress per iteration on PLS but performs poorly in terms of runtime. Amongst the stochastic algorithms, the incremental algorithm is consistently the best in terms of both runtime and progress-per-iteration, and generally attains an objective close to the optimum faster than the batch algorithm.

VIII. Conclusions and Discussion

We believe that in data-driven applications, the stochastic optimization view is often the correct view of PCA, PLS, and CCA, rather than a more conventional empirical optimization or linear-algebra view. In this paper we have taken our first steps at formalizing these problems as problems of optimizing a population objective and studying it as such, and we hope this will be the seed of future work with this view in mind. We have not been able to provide any quantitative theoretical guarantee on the performance of the stochastic power method or the truncated incremental method, and thus cannot analytically compare them with the SAA approach. The difficulty of providing such guarantees is highlighted by

the lack of guarantees for the variants of the stochastic power method mentioned in Section III-A. Nevertheless, we hope to progress toward such a theoretical understanding. Beyond PCA and PLS, another outstanding challenge is obtaining a good stochastic optimization method for CCA—as discussed in Section VI, the stochastic constraints make this problem much more challenging, and we do not have a satisfying solution to the problem at this stage.

With that said, the stochastic optimization view allows us to see the advantages of a SA approach over the standard SAA (empirical optimization) approach, and we have demonstrated this advantage empirically for both PCA and PLS. In particular, for PCA:

- We have presented a novel truncated incremental approach (Section III-B)
- We have presented a novel method for obtaining the iterates of Warmuth and Kuzmin [8], which is orders of magnitude faster and more memory efficient, and is thus practical in large-scale applications (Section III-C).
- We have pointed out that a renormalization at every iteration of the stochastic power method is redundant and yields the same iterates, in terms of the subspace spanned (Section III-A), thus allowing for a faster implementation.

We have also extended these methods to PLS, describing, we believe for the first time, SA approaches for this problem. We also report on an empirical comparison of the three SA approaches, as well as SAA, for PCA and for PLS. For PCA, both the stochastic power method and the truncated incremental approach outperform the SAA approach, with each method preferable in a different regime.

REFERENCES

[1] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *NIPS'07*, 2007, pp. 161–168.

[2] S. Shalev-Shwartz and N. Srebro, "SVM optimization: Inverse dependence on training set size," in *ICML'08*, 2008, pp. 928–935.

[3] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1574–1609, January 2009.

[4] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM," in *ICML'07*, 2007, pp. 807–814.

[5] M. Collins, A. Globerson, T. Koo, X. Carreras, and P. L. Bartlett, "Exponentiated gradient algorithms for conditional random fields and max-margin markov networks," *J. Mach. Learn. Res.*, vol. 9, pp. 1775–1822, Jun. 2008.

[6] S. Shalev-Shwartz and A. Tewari, "Stochastic methods for l1 regularized loss minimization," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML'09. New York, NY, USA: ACM, 2009, pp. 929–936.

[7] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Networks*, vol. 12, pp. 459–473, 1989.

[8] M. K. Warmuth and D. Kuzmin, "Randomized PCA algorithms with regret bounds that are logarithmic in the dimension," in *NIPS'06*, 2006.

[9] M. Gu, "Single- and multiple-vector iterations," in *Templates for the solution of algebraic eigenvalue problems: a practical guide*, Bai, Zhaojun, Demmel, James, Dongarra, Jack, Ruhe, Axel, and van der Vorst, Henk, Eds. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2000, ch. 4.3.

[10] A. Ruhe, "Lanczos method," in *Templates for the solution of algebraic eigenvalue problems: a practical guide*, Bai, Zhaojun, Demmel, James, Dongarra, Jack, Ruhe, Axel, and van der Vorst, Henk, Eds. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2000, ch. 4.4.

[11] J. A. Tropp, "User-friendly tail bounds for sums of random matrices," *Foundations of Computational Math.*, Aug 2011.

[12] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *Journal of Mathematical Analysis and Applications*, vol. 106, pp. 69–84, 1985.

[13] K. I. Kim, M. O. Franz, and B. Schölkopf, "Iterative kernel principal component analysis for image modeling," *IEEE Trans. PAMI*, vol. 27, no. 9, pp. 1351–1366, 2005.

[14] M. Brand, "Incremental singular value decomposition of uncertain data with missing values," in *ECCV*, 2002.

[15] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, no. 1, pp. 31–48, 1978.

[16] T.-J. Chin, K. Schindler, and D. Suter, "Incremental kernel SVD for face recognition with image sets," in *Proc. 7th Intl. Conf. Automatic Face and Gesture Recog. (FG06)*, 2002.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental singular value decomposition algorithms for highly scalable recommender systems," in *Fifth International Conference on Computer and Information Science*, 2002.

[18] Y. N. Rao and J. C. Principe, "A fast on-line generalized eigendecomposition algorithm for time series segmentation," *IEEE Transactions on Signal Processing*, 2000.

[19] A. Beck and M. Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex optimization," *Operations Research Letters*, vol. 31, no. 3, pp. 167–175, 2003.

[20] K. Tsuda, G. Rätsch, and M. K. Warmuth, "Matrix exponentiated gradient updates for on-line learning and bregman projection," *J. Mach. Learn. Res.*, pp. 995–1018, Dec. 2005.

[21] K. Choukri, G. Chollet, and Y. Grenier, "Spectral transformations through canonical correlation analysis for speaker adptation in ASR," in *ICASSP*, 1986.

[22] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan, "Multi-view clustering via canonical correlation analysis," in *ICML'09*, 2009.

[23] K. Livescu and M. Stoehr, "Multi-view learning of acoustic features for speaker recognition," in *ASRU*, 2009.

[24] F. Rudzicz, "Adaptive kernel canonical correlation analysis for estimation of task dynamics from acoustics," in *ICASSP*, 2010.

[25] S. Bharadwaj, R. Arora, K. Livescu, and M. Hasegawa-Johnson, "Multiview acoustic feature learning using articulatory measurements," in *Intl. Workshop on Stat. Machine Learning for Speech Recognition (IWSML)*, 2012.

[26] R. Arora and K. Livescu, "Kernel CCA for multi-view learning of acoustic features using articulatory measurements," in *Symp. on Machine Learning in Speech and Language Processing (MLSLP)*, 2012.

[27] A. Haghighi, P. Liang, T. Berg-Kirkpatrick, and D. Klein, "Learning bilingual lexicons from monolingual corpora," in *ACL*, 2008.

[28] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural Computation*, vol. 16, no. 12, pp. 2639–2664, 2004.

[29] M. B. Blaschko and C. H. Lampert, "Correlational spectral clustering," in *CVPR*, 2008.

[30] J. R. Westbury, *X-ray microbeam speech production database user's handbook*, Waisman Center on Mental Retardation & Human Development, University of Wisconsin, Madison, WI, USA, June 1994.