EE 516: Mathematical Foundations of Machine Learning **Winter 2023**

Course Project: Dictionary Learning
Due: March 24, 2023, 11:59PM PT

Student Names: Adina Edwards, Andy Franck, Joe Rawson

1 Paper Descriptions

Our papers cover 3 works on the topic of Dictionary Learning.

1.1 Paper 1

Paper Title: Trainlets: Dictionary Learning in High Dimensions
Student Name: Adina Edwards

1.1.1 Problem Description & Formulation

The Trainlets algorithm describes an approach to efficiently determining a dictionary for handling highly dimensional data. Combining a basis dictionary of cropped wavelet atoms, with the novel Online Sparse Dictionary Learning (OSDL) learned dictionary, Trainlets can efficiently process two dimensional data of relatively large size. The authors discuss image restoration, image compression, and pursuing general dictionaries from large datasets such as wild images on the internet, however the door is open to other image processing applications.

By taking random atoms from the observed signal Y and using sparse regression to update the learned dictionary A as a whole matrix, the Trainlets algorithm can more efficiently approximate large dimensional data in both fewer iterations, and with less computational load per iteration relative to other algorithms that attempt sparse regression on highly dimensional data. This allows for more efficient and faster processing of larger images. Where wavelets and other image processing machine learning (ML) algorithms work well with smaller image patches around 12×12 pixels, Trainlets have shown good results on subimages, or image patches, of the same size and larger, up to 64×64 pixels. This means Trainlets can process larger images and/or larger sets of images more efficiency than similar ML algorithms.

Trainlets are based on the combination of two dictionaries: a sparse base dictionary Φ made of cropped Wavelets of the training data, and a learned dictionary A that is updated every iteration to incorporate a sparse representation of randomly chosen columns of the observed data Y. Iterations are performed until the cost function reaches a minimum, or enough training data has been processed to achieve the desired results (in the case of sufficiently high dimensional data).

The Trainlets algorithm builds on several existing algorithms that each try to solve an inconvenience when it comes to dictionary learning, however Trainlets takes a focus on developing double-sparse adaptable dictionaries for data with higher dimension than previous work in this area. This approach allows the dictionary to work equally well if not better on increasingly large sets of data, such as image patches of 64x64 pixels and possibly larger, compared to more commonly used 12x12 pixel image patches. The Trainlets algorithm is less computationally complex than other algorithms that also show promise on larger image patches, and therefore converges in less time on the same dataset.

Trainlets achieves this kind of performance by taking promising approaches from several algorithms and modifying them with an eye toward large dimensional data and low computational complexity, relative to its peer algorithms. Standard dictionary learning problems attempt to minimize the following equation over the dictionary D and the sparse vector approximation of the signal, X:

$$\arg\min_{D,X} \frac{1}{2} \|Y - DX\|_F^2 \text{ subject to } \|x_i\|_0 \leq p \ \forall i \qquad (1)$$

Trainlets take this several steps further by breaking the effective dictionary D into two dictionaries, the basis dictionary Φ and the learned dictionary A formed via OSDL. The former is a sparse basis dictionary based on continuous cropped Wavelets, while A is a sparse dictionary that is adapted each epoch of OSDL by incorporating information from a different random subset of the observed data Y.

1.1.2 Algorithm Description

Trainlets use two sparse dictionaries in order to handle all of the data involved with larger dimensions and the diversity within those signals. For this approach, the effective dictionary D is made of two components: the fixed base dictionary Φ and the sparse adaptable matrix A such that $D = \Phi A$. The dictionaries Φ and A may have a different number of columns, allowing for more or less redundancy in the effective dictionary. Using this dictionary approach and modifying (1) for a focus on sparsity, the problem can be rewritten as:

$$\arg\min_{A,X} \frac{1}{2} \|Y - \Phi AX\|_F^2 \text{ subject to } \begin{cases} \|x_i\|_0 \leq p \ \forall i \\ \|a_j\|_0 = k \ \forall j \end{cases} \qquad (2).$$

The fixed base dictionary Φ is a sparse representation of cropped wavelets of the training data. Traditional wavelets are subject to border effects, which hamper the sparsity of the resulting dictionary. Using a cropped version of these wavelets provides the benefits of wavelets without the edge effects. Cropped wavelets, mathematically described as $W_c = P^T W_s$, are formed by removing the zero padding from the Wavelet synthesis operator W_s. The Wavelet systhesis operator is the transpose of the Wavelet analysis matrix, $W_s = W_a^T$. The zero-padding required to calculate the wavelets is represented by the matrix P, and P^T undoes that zero padding. The resulting dictionary Φ from these cropped wavelets can be described as

$$\Phi^c = W_c \mathcal{W},$$

where $\mathcal{W}$ is a square diagonal matrix for normalizing the atoms (columns) in Φ^c.

The adaptable dictionary, A, in (2) is another sparse dictionary whose atoms are updated iteratively via Online Sparse Dictionary Learning (OSDL). While other adaptations improve the performance of Trainlets, it is the training of A via OSDL that is the most novel and useful adaptation to the overall algorithm. The OSDL algorithm iteratively uses stochastic gradient decent (SGD) with a step size, η^t, based upon a normalized gradient of the current iteration. These tactics were discussed in class. It also employs a momentum factor, $U_{\mathcal{S}}^{t+1}$, which is widely used in optimization problems, but was not covered in class. The momentum factor helps avoid convergence on local minima and instead encourages convergence on the global minimum.

Before OSDL can begin iterating through epochs, a handful of variables should be initialized. Note that the iteration term t is sometimes written as a superscript and others as a subscript in the original paper. That is avoided in this paper and instead will be written as a superscript. Algorithms upon which OSDL are based use a Gram Matrix of the effective dictionary D. Because the effective dictionary in Trainlets is a combination of the static dictionary Φ and the learned dictionary A, the initial Gram Matrix for OSDL is just $G_\Phi^0 = \Phi^T \Phi$. This matrix is later updated during each epoch by left and right multiplying it by the updated dictionary A^{t+1}. The momentum term U^0 is initialized to 0. The paper did not discuss how to initialize the learned matrix A. Algorithmic complexity calculations assume A is square, however this is

not a hard requirement. In fact, the shape of A will depend upon the desired redundancy of the effective dictionary D. This desired redundancy level is a tuning parameter left up to whomever wishes to implement Trainlets. For now, we can presume that A^0 is initialized to a square matrix of 0s. Further work needs to be done to determine if there is a better way to initialize A to reduce the overall number of epochs before the desired level of convergence is reached.

Once variables are initialized, iterations $t = 1, ..., T$ can begin. During each epoch, OSDL takes a random set of columns from the observed data Y called the mini-batch Y^t.Then sparse regression is done over that set using the base dictionary, Φ, the learned dictionary so far A^t and the Gram matrix G^t to determine a sparse representation of the mini-batch Y^t called X^t. To simplify and speed up calculations, several of the following calculations use the support of $A_{\mathcal{S}}$, otherwise known as the non-zero elements of A. The gradient is taken over $A_{\mathcal{S}}$, resulting in $\nabla f(A^t_{\mathcal{S}})$. The Frobenius norm of this gradient is then used along with every variable known so far to calculate the step size η^t according to the following equation:

$$\eta^t = \frac{\|\nabla f(A^t_{\mathcal{S}})\|_F}{\|\Phi \nabla f(A^t_{\mathcal{S}}) X^t_{\mathcal{S}}\|_F}.$$

This step size η^t is intended to decrease as t increases, which helps convergence. To further help convergence by attenuating oscillations, the momentum term $U^{t+1}_{\mathcal{S}}$ is then calculated my multiplying the current momentum $U^t_{\mathcal{S}}$ by $\gamma \in [0, 1]$ and adding to η^t multiplied by the gradient of $A^t_{\mathcal{S}}$, all of which were previously calculated. If γ is set to 1, the updated momentum term will include momentum from the prior epoch, otherwise it will be the standard stochastic gradient decent term studied in class. Again, the support of A and also of U are used instead of the full matrices to speed up calculations. Now, the learned dictionary can be updated all at once by subtracting the updated momentum $U^{t+1}_{\mathcal{S}}$ from the current $A^t_{\mathcal{S}}$ and performing hard thresholding. Hard thresholding here is column by column, but updates $A^{t+1}_{\mathcal{S}}$ in one step, without having to iterate over the entire matrix. This saves time compared to the similar Normalized Iterative Hard-Thresholding (NIHT) algorithm and is part of what makes OSDL faster in the majority of cases. Finally, the Gram matrix G can be updated by left multiplying it by the transpose of the current learned dictionary A and right multiplying it by the support of the same dictionary resulting in

$$G^{t+1}_{\Phi} = (A^{t+1})^T G^t_{\Phi} A^{t+1}_{\mathcal{S}}.$$

These epochs continue until either the max epochs T have been reached, or (2) converges within a pre-set error factor using the resulting A.

Trainlets should theoretically work as they are based on a deeper look at the standard Dictionary Learning cost function (1). Trainlets attempts to optimize select portions of standard Dictionary Learning Specifically to handle data of larger dimensions. While some taken relaxations void guarantees to converge (such as the use of the 0 pseudo-norm) other terms like momentum were added to help encourage convergence to a global minimum. A strong focus on keeping both the basis dictionary Φ and the learned dictionary A sparse and then frequently only using the support of A speeds up calculations significantly without changing the fundamental approach to standard Dictionary Learning.

1.1.3 Theoretical Results

One thing that all dictionary learning takes advantage of is the assumption that natural signals can be represented by linear combinations of sparse atoms. This has been shown to hold well enough by several dictionary learning algorithms, and is used here as well. OSDL also assumes that a sparse effective dictionary D can be represented as the multiplication of a sparse base dictionary Φ and an adaptable sparse dictionary A. Further, it is assumed that the cropped Wavelets that make up Φ are orthogonal.
The theoretical results and experiments of Trainlets on real data are very promising. Compared to a handful of similar algorithms (such as Sparse K-SVD, the Online Learning Dictionary (OLD), Stochastic Normalized

Iterative Hard Thresholding (NIHT), and PCA) OSDL-based Trainlets usually result in higher signal-to-noise ratios with more sparse dictionaries, and often more quickly given the same constraints. The authors performed several tests, with as close to an A/B comparison as possible.

Part of the success of Trainlets is based on the use of cropped wavelets and their lack of discontinuities as this leads to sparser representation in the base matrix. It is possible to select Φ differently, or to adapt it during each epoch, though doing so would change the theoretical results. The authors left it as a future task to see how adapting Φ at each epoch would improve or hamper overall performance.

OSDL is not guaranteed to converge to a local minimum as the l_0 pseudo-norm makes these kinds of guarantees difficult and often means there are discontinuities. However, the momentum term, $U_{\mathcal{S}}^{t+1}$, in the calculation of $A_{\mathcal{S}}^{t+1}$ encourages convergence, and is furthered along by a decreasing step size, η^t. This tendency toward convergence is backed up by actual numerical results where the algorithm almost always converged. The relative simplicity of calculating the l_0 pseudo-norm is part of the benefit of this algorithm, as the computational cost can be spent on processing more columns of data instead of relatively complex norm calculations.

One interesting theoretical comparison that was shown to hold in experiments, is the performance of the related NIHT and OSDL algorithms. Both solve (2) by iterating through sparse regression and stochastic gradient decent (SGD) to determine the best sparse dictionary A. However, NIHT operates column by column through Y and updates A atom by atom, resulting in a double for loop per epoch. OSDL selects a random set of columns from Y per epoch, and updates A as a whole matrix. To encourage convergence to the global minimum, OSDL calculates a momentum value per epoch, which is used to update A, however this one additional calculation is less involved than iterating through each atom in A as it mostly depends on factors already calculated in the current epoch, and those from the previous epoch. The authors ran an experiment where the task was to train adaptive sparse dictionaries on the same dataset, but with varying image patch sizes and sparsity levels. The resulting graphs can be found on page 3187 of [1], where each marker represents one epoch for the given algorithm. Examining NIHT and OSDL, we can see that NIHT results in less error when the dimensionality of the image patch size is small and the sparsity is low, however it quickly looses ground to OSDL as sparsity and/or the patch size increase. It should also be noted that OSDL regularly completes epochs more quickly, meaning larger datasets can be processed more quickly holding other factors steady.

1.1.4 Relation to Course Material

Several topics covered in class were used for the development of and directly in this algorithm. Sparse regression is an obvious one, but there is also the use of SGD, $\mathcal{L}_0$ norms, and cost function optimization. A Kronecker product is used to describe cropped wavelets which make up the base dictionary, Φ. The pseudo-code for OSDL actively uses SGD with a guaranteed decreasing step size, the Frobenius norm, and hard thresholding. While not ultimately used, the authors toyed with the idea of holding one value steady while updating another, and then alternating back and forth through epochs.

Some empirical results use approximation error as a measure of when the algorithm has converged. Part of the comparison between the NIHT and OSDL algorithms used finding the mean step size selected throughout dozens of iterations over large datasets. Additional empirical results were found by comparing the performance of OSDL against some algorithms we read about and implemented in class such as K-SVD and PCA, respectively.

I might have been able to figure out some of the basic matrix multiplication and transpositions used, but I would have struggled to make it through the paper at all. This course gave me the tools to figure out enough to plod through the paper and put most of the pieces together. I was not familiar with various

types of norms or what they meant for convergence, nor was I familiar with the mathematical terminology for optimization.

Trainlets still contains some ideas not covered in class, such as wavelets, and momentum, however these concepts are likely outside of the scope of the course. I am still shaky on Kronecker products and how they allow 2-dimensional data to be processed in a single dimension, and how to form optimization problems for a specific task.

1.2 Paper 2

Paper Title: Task Driven Dictionary Learning
Student Name: Andy Franck

1.2.1 Problem Description & Formulation

The paper discusses the use of a learned dictionary of vector elements to decompose data into a linear combination of those elements. This been shown to be effective in various signal processing and classification tasks like handwritten digit recognition, nonlinear image mapping, art authentication, and compressed sensing.

The goal is to learn a dictionary matrix D that is well-suited for a specific task, such as image classification or regression. To achieve this, we need to solve an optimization problem that involves minimizing a cost function depending on both the model parameters W and the dictionary D. This is a task-driven approach to dictionary learning, where the dictionary is tuned to the specific job at hand.

Given a training set (x_i, y_i) of input-output pairs, the optimization problem is as follows: find a dictionary D and function f that maps the inputs to outputs, such that empirical risk is minimized [2].

The function and dictionary are learned simultaneously, using an iterative algorithm alternating between updating D and f. These updates are done using stochastic gradient descent, and they involve sub-problems that can be solved using convex optimization methods. The output dictionary and function can then be used to make predictions on new input data.

This approach is essential to develop solutions for classification and regression problems in a large-scale setting. As demonstrated in the experiment done on nonlinear image mapping, this task-based method is effective on millions of training pairs. And, it is considerably more robust to training data where only a few labeled samples are available.

This paper introduces a method for training dictionaries for specific tasks, rather than those created for general data reconstruction. The method described in this paper can be extended and modified to fit different tasks as demonstrated in sections 3.2 and 5 [2].

The sparse models demonstrated in this paper are also more flexible than decompositions based on principal component analysis, which requires its elements be orthogonal. This allows for more flexibility in the model.

Overall, this paper provides a general framework for more specific task-driven dictionary problems. It provides multiple example extensions to the basic framework in 3.2[2]. It also provides experimental results in section 5[2] that demonstrate the effectiveness of the supervised methods compared to unsupervised strategies.

1.2.2 Algorithm Description

This is an iterative algorithm using stochastic gradient descent. It is designed to minimise gradients with the form of:

$$\begin{cases} \nabla_W f(D,W) = \mathbb{E}_{y,x}[\nabla_W \ell_s(y,W,\alpha^*)] \\ \nabla_D f(D,W) = \mathbb{E}_{y,x}[-D\beta^*(\alpha^*)^T + (x - D\alpha^*)(\beta^*)^T] \end{cases} \tag{1}$$

Where D is the dictionary, W are the model parameters, y is the labels, β is a vector in $\mathbb{R}p$ that depends on y, x, W, with

$$\beta^*_\Lambda = (D^T_\Lambda D_\Lambda + \lambda_2 I)^{-1} \nabla_{\alpha_\Lambda} \ell_s(y,W,\alpha^*) \tag{2}$$

and α^* (short for $\alpha^*(x,D)$) being defined as:

$$\alpha^*(x,D) \triangleq \arg\min_{\alpha \in \mathbb{R}p} \frac{1}{2}||x - D\alpha||_2^2 + \frac{\lambda_2}{2}||\alpha||_2^2 \tag{3}$$

Note that we have shown that 4 is smooth and computed the gradients. It leverages a first-order stochastic gradient descent algorithm that sequentially draws samples (y_t, x_t) from the unknown probability distribution $p(y,x)$.

The algorithm utilizes similar tools to that of unsupervised data-driven dictionary learning. In this task-driven case, however, the goal is to find model parameters W at the same time as the dictionary model D. This is done by solving

$$\arg\min_{D \in \mathcal{D}, W \in \mathcal{W}} f(D,W) + \frac{v}{2}||W||_F^2, \tag{4}$$

where the task cost $f(D,W)$ is defined as

$$f(D,W) \triangleq \mathbb{E}_{y,x}[\ell_s(y,W,\alpha^*(x,D))] \tag{5}$$

The loss function takes a very similar form to that of data-driven dictionary learning:

$$\ell_u(x,D) \triangleq \min_{\alpha \in \mathbb{R}^p} \frac{1}{2}||x - D\alpha||_2^2 + \lambda_1||\alpha||_1 + \frac{\lambda_2}{2}||\alpha||_2^2 \tag{6}$$

Also note that if the second regularization parameter is set to zero, this function becomes Lasso, which we have covered and solved in class.

The paper leverages multiple tools in the algorithm. The authors note that, if gradients have the form as in 1, then the stochastic descent algorithm utilized will converge to stationary points under some assumptions not listed. The authors also make a quick aside about randomising the order of which the first approximation vectors are ordered. Similar to a homework in class, the authors state that the vectors are obtained by cycling over a randomly permuted training set, which they say is very common in similar machine learning settings [2].

More generally, this algorithm is utilizing already known methods for data-driven dictionary learning, then demonstrating how they can be modified to create more efficient methods based on the specific task at hand.

In section 5[2], the authors choose some application methods to test their theories. These experiments include handwritten digit recognition, nonlinear image mapping, digital art authentication, and compressed sensing. Specifically in regards to digit recognition, the supervised methods of this paper performed considerably better than unsupervised.

In a second experiment on handwritten digit recognition, the authors test training data where only a small percentage of the samples were labeled. After initializing the dictionaries using the unsupervised methods in section 2, the authors report performance similar to that of other, more refined unsupervised learning methods[2].

1.2.3 Theoretical Results

A few assumptions are given for the theoretical results to hold. Firstly, the data (y, x), admits a probability density p. Secondly, given a finite set of labels Y, when it is a subset of a finite-dimensional vector space, the loss function is twice continuously differentiable and $p(y, x)$ is continuous. For all y in Y, $p(y, .)$ is then continuous and $\ell_s(y, .)$ is twice continuously differentiable[2]. These last two assumptions will allow us to use several loss functions depending on the task.

After showing that the loss function 9 is differentiable, the authors apply two propositions to demonstrate that even though the coefficients of the learned dictionary α^* are obtained by solving a non-differentiable optimization problem, the function f is differentiable on WxD and computing its gradient is possible/doable.

Because stochastic gradient descent methods are designed to minimize functions with gradients of the form 1, the results guarantee that given a function with gradients of that proposition, the algorithm will converge to a stationary point of optimization problems, given assumptions noted in another paper referenced by the authors. Note that the stationary points are not necessarily global optimizers[2].

In theory, the setting in which stochastic gradient descent is used is not guaranteed to converge. However, it has been demonstrated in the authors experiments to behave well. Because the problem can be non-convex, an assumed convergence proof assumes the cost function is three times differentiable[2].

Often in practice, the authors set the second regularization parameter λ_2 to zero, which provides satisfactory results and ensures the algorithm will converge. However, some modifications must be made depending on the experiment. For example, in the compressed sensing experiment, a small positive λ_2 value was necessary.

1.2.4 Relation to Course Material

In the algorithm, the authors choose to implement the LARS algorithm (42), which was originally developed to solve the Lasso formulation that we have studied in class (when $\lambda_2 = 0$). To attain good results, it is essential that the method is well initialized, so the authors utilize the SPAMS toolbox [3] to initialize the dictionary. Then, they optimize (with respect to W) the convex cost function

$$\arg\min_{W \in \mathcal{W}} f(D, W) + \frac{v}{2}||W||_F^2, \tag{7}$$

Where W is our convex set, v is a regularization parameter, and f is

$$f(W) \triangleq \ell_s(y, W, \alpha^*(x, D)) \tag{8}$$

We have studied multiple solutions to convex cost functions in class, which are applied here to get the pair (D, W) used to initialize the algorithm.

More generally, the application of dictionary learning involves a finite training set, and the minimization of a cost function. Said cost function also contains a loss function

$$\ell_u(x, D) \triangleq \min_{\alpha \in \mathbb{R}^p} \frac{1}{2}||x - D\alpha||_2^2 + \lambda_1||\alpha||_1 + \frac{\lambda_2}{2}||\alpha||_2^2 \tag{9}$$

similar to those covered in class. In this case, there is not one but two regularization parameters. However, we have covered similar loss functions in class, albeit with only one regularization parameter (Lasso).

In the application of handwritten digit recognition, Lasso is preferred. Given the demo in class, I found this application fairly easy to understand. Similarly, in the compressed sensing application, I saw a familiar term in PCA. Since I have spend considerable time reviewing PCA, I was able to follow an application in an area of which I have zero experience.

The idea of a dictionary is also something we have covered in class. Given a vector x, we can call it a "sparse approximation" over a dictionary D when a linear combination of the columns of D are "close" to x[2]. We have not dealt with approximate solutions to linear combinations, however the idea that a vector can be comprised of elements of other vectors from a dataset is not a new concept.

Certainly, I do not believe I could understand many of the concepts in this paper prior to taking this course. I understood iterative algorithms and gradient descent, but had no prior knowledge of solutions to regression equations such as Lasso. Considerable sections of the paper reference topics in linear transformations, vector spaces, convexity, and matrix decompositions. Perhaps with research I could have furthered my understanding of gradient descent to grasp the concept of stochastic gradient descent. However, with no knowledge on solutions to equations such as 9, even without regularization parameters, I would have needed considerable study to understand the underlying mathematical concepts behind such a solution.

After the course, although I am certainly not yet fluent, I find understanding references to these terms considerably easier to understand. I now understand benefits of convex functions and how they can be solved using methods such as gradient descent. During my first linear algebra class, I wondered about the applications of topics such as range and nullspace, and finally I see their place in training and understanding datasets. Although linear least squares solutions are not the main topic of this paper, I feel that our covering of the topic has helped me understand methods to solving more complicated functions.

I actually felt I had a fairly strong understanding of the paper with the knowledge from this class. There were certainly multiple definitions I did not understand, however. Multiple references were made for functions to be Lipschitz on a space. I have little knowledge of this term—although we covered it briefly in an optimization course last term—so I turned to google for some clarification.

Nearly all of the terms were things we had covered, albeit somewhat briefly. There is considerable discussion of subgradients and subdifferentials, which I do recall appearing on a homework problem in the past. Of course, I had to spend some time reviewing to understand how the terms were being used.

Similarly, there are references to the "probability distribution" $p(y, x)$[2], which is used in both proofs for theoretical results and the algorithm. I had no prior understanding as to what a probability distribution was. Perhaps it was covered in class, but I do not recall reviewing it. Since it is a fairly integral part of the algorithm, that was the subject I had most trouble understanding when reading the paper.

1.3 Paper 3

Paper Title: Complete Dictionary Learning via ℓ^4-Norm Maximization over the Orthogonal Group
Student Name: Joe Rawson

1.3.1 Problem Description & Formulation

The paper I'm covering is about learning a complete, sparse dictionary from sampled data using an ℓ^4-norm maximization technique starting from a random orthogonal matrix. The most obvious applications and areas of importance for this are in data compression, reconstruction and categorization of sampled signals.

Mathematically, given a dataset of $Y \in \mathbb{R}^{n\times p}$ samples, the objective is to find a non-singular matrix $D_0 \in \mathbb{R}^{n\times n}$ such that $Y = D_0 X$. The only known here is the Y samples, so this is formulated as an unsupervised learning problem. Finding the "ground truth" D_0 is NP-hard, so instead the authors approximate it with $D_\star$ and use a number of preconditions to estimate it. Since $D_\star$ is non-singular, we can find $X = D_\star^{-1} Y$.

The authors method solves for the $D_\star$ matrix iteratively using a few SVD's, so it should be much more efficient than other more complex algorithms. The novel contribution of this paper is supposed to be the efficiency by which it finds a complete estimate of $D_\star$.

1.3.2 Algorithm Description

The authors describe their algorithm as an iterative gradient ascent method to find

$$\underset{D\in O(n;\mathbb{R})}{\arg\max} ||D^T Y||_4^4,$$

where $O(n;\mathbb{R})$ is the group of orthogonal $n \times n$ matrices, $Y \in \mathbb{R}^{n\times p}$ where n is the dimension of the sampled signal and p is the number of signals. Interestingly they claim you can use an infinite step size to achieve a global maximum solution. Put simply, they iterate on calculating the gradient repeatedly then perform a single step using an inner product.

The intuition the authors use for this comes from the equation

$$\underset{X,D\in O(n;\mathbb{R}),Y=DX}{\arg\max} ||X||_4^4.$$

Using the ℓ^4-norm is what promotes sparsity since the 4^{th}-powers of the entries promotes "spikiness" or "sparsity" of the entries since large numbers will be driven larger and small numbers will be driven to zero. Since it is also a smooth differentiable function they expect it is "more amenable to optimization"[4] when taking the gradient.

They introduce a novel algorithm they call *matching, stretching, and projection* (MSP) where they scale an orthogonal matrix by the sample matrix, cube it then project it onto the orthogonal group in an iterative fashion. Doing this maximizes the function $||AY||_4^4$ which is approximately equivalent to maximizing the objective $||AD_0||_4^4$. The reasoning for this is $\forall A \in O(n;\mathbb{R}), \frac{1}{p}||AY||_4^4 = \frac{1}{p}||AD_0X_0||_4^4 = \frac{1}{p}\sum_{j=1}^{p}\frac{1}{p}||AD_0x_j||_4^4$ can be viewed as the mean of p independent and identically distributed (i.i.d.) random variables that will concentrate to their expectation $\mathbb{E}_{x_j}||AD_0x_j||_4^4$ and is largely characterized by $||AD_0||_4^4$ [4].

Although this paper is at times hard to grasp fully and cites a few other works, it is clear to see that they utilize concepts of orthogonality, optimization, and gradient step iteration like we covered in class.

Empirically the algorithm seems to perform very well against KSVD, SPAMS and Subgradient based implementations, (although SPAMS had a smaller error rate). In all cases, the MSP algorithm found it's solutions orders of magnitude faster than the other algorithms with a roughly constant small error rate of around 0.35% [4].

The Achilles heel of this algorithm seems to be the relationship between the number of dimensions n and the number of required samples p. This relationship is noted in the paper as $p = \Omega(n^2)$ and means that the relationship between p and n is exponentially asymptotic.

1.3.3 Theoretical Results

The theoretical assumptions made by the authors are:

1. $y = D_0 x, \forall y, x \in \mathbb{R}^n, D_0 \in \mathbb{R}^{n \times n}$, where D_0 is sparse, non-singular, complete and orthogonal, and that coefficients in x are i.i.d. and Bernoulli-Gaussian sparsely distributed.
2. That the ℓ^4-norm is optimally efficient at finding the objective. (There is discussion in the paper on the family of ℓ^{2k}-norm functions that support this.)
3. That $||AY||_4^4 \approx ||AD_0||_4^4, A \in \mathbb{R}^{n \times n}$ is a good approximation.

The paper mostly justifies these assumptions, with the possible exception of (1), which they state is a typical statistical model. This makes me question whether or not it's possible to bias the algorithm by violating the i.i.d. or binomial distribution assumptions and cause it give bad results.

Their main result is stated in Theorum 1[4]: Suppose a matrix $A_\star$ is a global maximizer of the optimization problem:

$$\arg\max_{A \in O(n;\mathbb{R})} ||AY||_4^4$$

then for any $\epsilon \in [0, 1]$ then there exists a signed permutation matrix $P \in SP(n)$ such that:

$$\frac{1}{n}||\hat{A}_\star^T - D_0 P||_F^2 \leq C\epsilon$$

In the above equation, $A_\star$ is $n \times p$. The p samples are assumed to be independent and identically distributed (i.i.d.) random variables (the authors make references to Bernoulli-Gaussian distributions), so the intuition in the above is that given enough samples the mean of the difference between the estimate $\hat{A}_\star$ and the actual (permuted) ground truth D_0 drops below some constant (small) error. The authors acknowledge that the ground truth can only be estimated within signed permutation ambiguities of the original matrix.

In Theorum 7 [4] the authors refine their formulation a bit (after several proofs, lemmas and remarks) and further qualify the above inequality by saying that the signed permutation P exists with a probability of $1 - \frac{1}{p}$ when $p = \Omega(\theta n^2 \ln \frac{n}{\epsilon^2})$ and $C > \frac{4}{3\theta(1-\theta)}$. The value $\theta \in (0, 1)$ is a measure of sparsity, lower is more sparse.

Looking closely at these conditions, what this says is there is an interplay between sparsity, the number of samples, and the dimensionality of the samples that will determine whether or not this algorithm will converge to minimum error. As noted earlier, the relationship between p and n is $p = \Omega(n^2)$, but the curve of the exponential can be adjusted by θ if the ratio of p/n is low.

To summarize, this should be a good algorithm provided you:

1. Have enough samples.
2. Your dimensionality isn't too large. (Sub-400 when $\theta = 0.5$)

Given these constraints, this algorithm should converge to a global minimum with high probability.

1.3.4 Relation to Course Material

There are many topics from class that are used in this paper: Orthogonal matrices and their properties, maximization/minimization techniques, gradient step iterations, projection onto a subspace, and ℓ^p-norms.

I feel like Random Processes would have helped me understand some of the probabilistic arguments they are making but I think I get the overall argument they make. Some of their formulations rely on the notion of expectations of an estimating function that I think could be part of that course. But aside from the statistical probability portions of this paper I think most everything else fits with topics of this course.

2 Comparison of Algorithms

It is hard to make a direct comparison between the three algorithms in the papers we read, as they target different applications with different goals within a similar application domain. While all of them can be used for image processing, the task driven algorithm, [5], shifted its approach based on the specific task of what kind of processing to do on an image. The ℓ^4-norm paper, [4], focused on finding a dictionary using an efficent algorithm. The third paper on Trainlets, [1], focused on efficiently handling larger image patches for dealing with larger image signals and image datasets.

Between the three papers [1], [4], and [5], the algorithm in the ℓ^4-norm paper [4] seems the simplest algorithm to implement. Compared to the other two papers, each iteration has fewer computational steps in its complexity, although it has an extensive formulation.

The Task Driven paper [5] was more of a patch-work of task specific implementations where algorithms were suited to different problems. As such, the paper wasn't very strong on theoretical guarantees of any particular solution. The ℓ^4-norm paper [4] has some strong theoretical guarantees with stated high probability, so long as you have enough samples and your dimensionality is low enough. The Trainlets paper [1] also had few theoretical guarantees, but was empirically very promising, aided greatly by the use of cropped wavelets, and a momentum term with a guaranteed decreasing step size.

The iterations of the MSP algorithm in the ℓ^4-norm over the orthogonal group paper [4] seem to have the lowest computational complexity compared to either the Trainlets [1] or Task Driven [5] algorithms. (Though in the case of the task driven paper it would depend on the task we decided to implement.) Trainlets [1] work well for wide matrices, but has a higher computational cost than the ℓ^4-norm algorithm [4]. As the sparsity decreases and/or the image patch size increases, the amount of time it takes to complete an epoch of the ℓ^4-norm sky rockets exponentially as can be seen in Figure Eight on page 25 of [4].

Performance of two of these algorithms, the MSP [4] and the Trainlets [1] algorithms, heavily depended upon the size of the image signals and the level of sparsity in the resulting dictionary for representing those images. While it is difficult to make a direct comparison between the two algorithms given the number of and (both our and the authors') lack of understanding around the effect of individual tuning parameters, it is known that both require significantly larger datasets to learn from as the dimensionality of the image signal increases. This is in part because more information about an image is obtained if the image patch size is small. Holding the image size fixed, as the patch size increases, less information can be obtained from a single image so additional images are needed to obtain similar levels of information.

3 Algorithm Implementation & Testing

We decided to implement the MSP algorithm using the ℓ_4-norm over the orthogonal group as described in [4]. Relative to the other two papers, the algorithm in this one seemed more straight forward to implement. The Trainlets algorithm, while interesting, was designed to really shine with very high dimension input data. Though Trainlets are designed to be more efficient, we were not familiar with wavelets in general, and didn't want to struggle with long iteration times and a hard deadline. The Task Driven algorithm was too generic and we ultimately decided that we understood the algorithm for the ℓ_4-norm paper better.

Unless otherwise noted, our experiments were run with an AMD Ryzen 9 5950X at 5GHz with DDR4-3200 RAM on the Arch Linux distribution.

3.1 Results on Synthetic Data

The authors did two different experiments that caught our eye. In one, they held θ steady at 0.5 while varying n and p. This experiment showed that as n and p increase, the error increases and that n is the dominant term. This coincides with $p = \Omega(\theta n^2 \ln \frac{n}{\epsilon^2})$ as discussed in the theoretical results section for this paper 1.3.3.

In their second experiment they instead held n and p steady and varied θ. Their results indiciated that a smaller θ would allow for larger dimensions n without sacrificing the error rate.

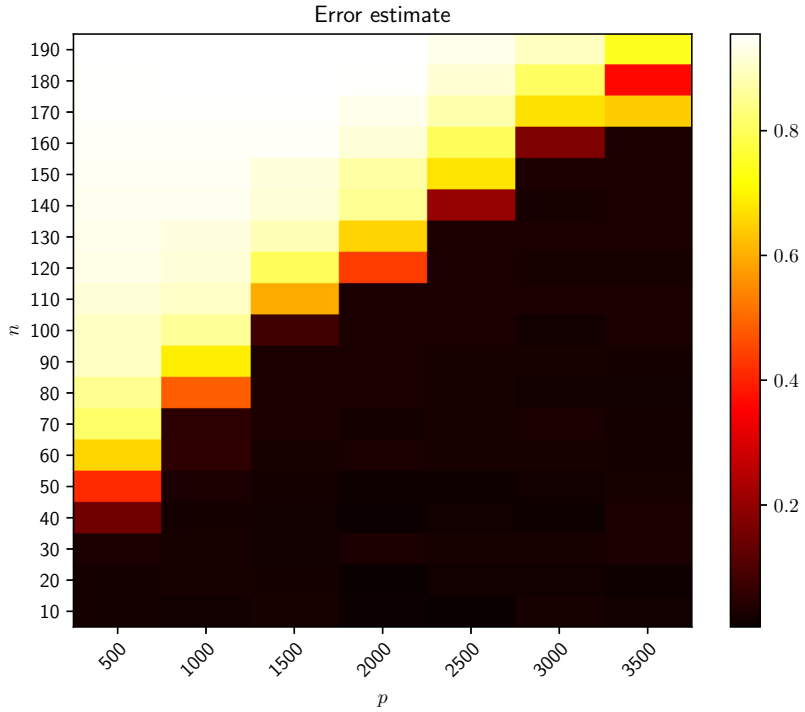

Figure 1: Error rate as the dimensions n and samples p increase and $\theta = 0.1$.

We were curious to see if lowering θ would allow for a larger n without affecting the error, so we chose

a smaller θ at 0.1 and did our own sweep of n and p. We swept n from 10 to 190 and swept p from 500 to 3900. For each combination of n and p, we performed 100 iterations of the algorithm. As can be seen in the figure 1 above, the error rate is low when n is sufficienently small relative to p. This result agrees with the theoretical lower bound for p 1.3.3.

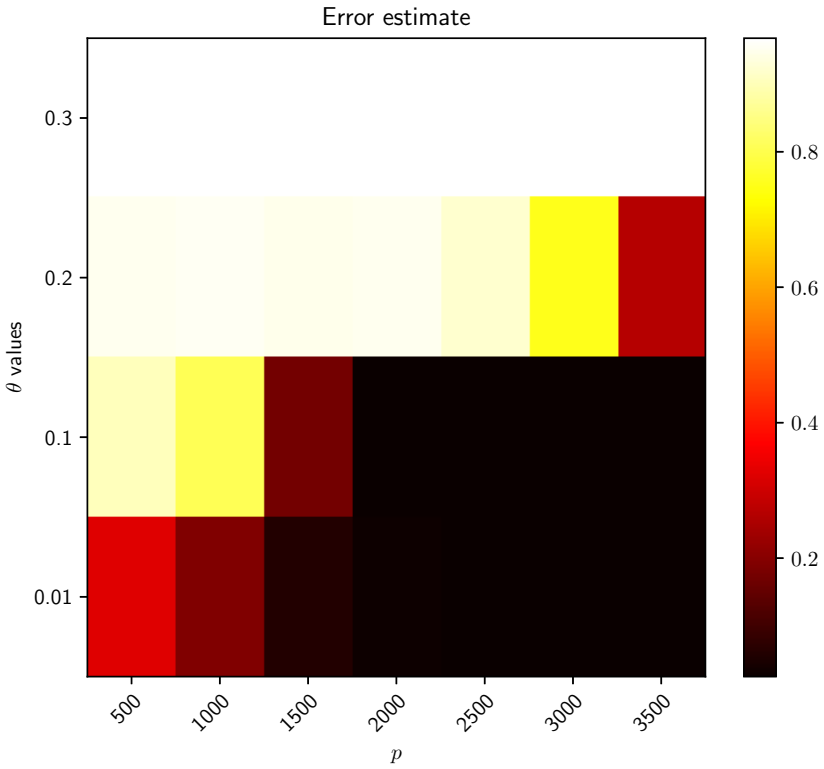

Figure 2: Error rate as the sparsity θ and the number of samples p increase and $n = 100$.

We also decided to sweep θ while holding n steady at 100, the results of which are seen in Figure 2. This allowed us to see the error rate as p and θ increased. As p increases and n and θ are steady, the error rate drops as expected. However, any change in θ requires a significant increase in p to attain a similarly low error rate. At $\theta = 0.1$, where we did most of our testing, our experiment showed that we needed around 2000 samples to get a decently low error rate. Decreasing the sparsity ten percent by increasing θ to 0.2 meant that the MSP algorithm needed more than 3500 samples before the error dropped to a reasonable level. This is a significant increase in the number of samples needed. Clearly, sparsity is beneficial.

3.2 Results on Benchmark Data

We chose the MNIST [6] dataset as our benchmark dataset and attempted to record the training error between the generated dictionary and the dataset using Algorithm 2 of [4]. The below graphs are for our runtime and expected error.

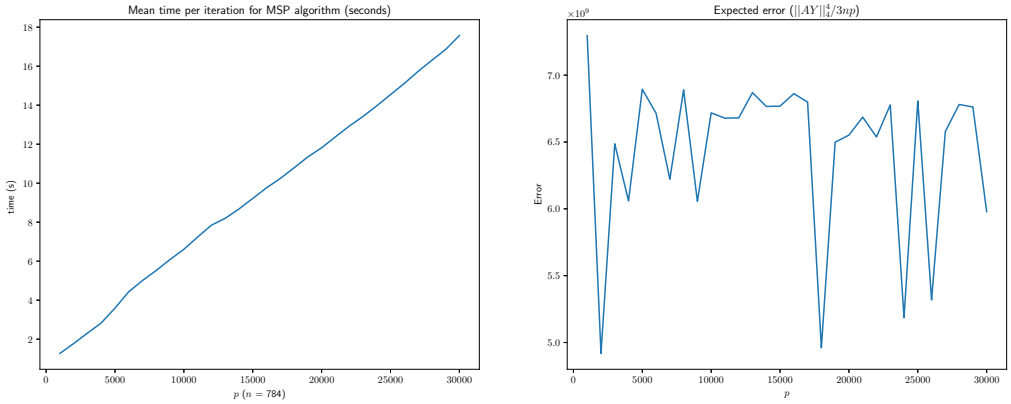

Figure 3: Mean iteration time over p and the calculated expected error.

In total, we bounded our run of the MNIST dataset to 3 hours and managed to collect data for input matrices up to size 30000×784. Given the line of the graph, the entire 240,000 image set would take around 2.5 minutes per iteration to complete, but sweeping p up to that point would be time prohibitive.

This algorithm notably did not converge while it was iterating, with the powers of 4 tending to blow up the expected result, which should have been around 1. After going back to the source paper and double checking equations it simply never came down to a resonable level.

Our chosen algorithms performance did not match what was reported in the paper. We believe that either our understanding of the algorithm implementation is incomplete, we missed an assumption on how to condition the input data, or the original authors overstated the performance of their algorithm.

References

[1] J. Sulam, B. Ophir, M. Zibulevsky, and M. Elad, "Trainlets: Dictionary learning in high dimensions," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3180–3193, 2016.

[2] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 791–804, 2012.

[3] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[4] Y. Zhai, Z. Yang, Z. Liao, J. Wright, and Y. Ma, "Complete dictionary learning via ℓ^4-norm maximization over the orthogonal group," *Journal of Machine Learning Research*, vol. 21, no. 165, pp. 1–68, 2020. [Online]. Available: http://jmlr.org/papers/v21/19-755.html

[5] I. Tosic and P. Frossard, "Dictionary learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, 2011.

[6] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

A Appendix

Our code repository is located here.