SVD-for-digit-classification-CUDA

“IPSC Project”

Gauravdeep Bindra (2018201027)

Anjul Ravi Gupta (2018201021)

Kishan Shankar Singhal (2018201023)

Abstract

From linear classifiers to neural networks, image classification has been a widely explored topic in mathematics, and many algorithms have proven to be effective classifiers. However, the most accurate classifiers typically have significantly high storage costs, or require complicated procedures that may be computationally expensive. We present a novel (nonlinear) classification approach using truncation of local tensor singular value decompositions (tSVD) that robustly offers accurate results, while maintaining manageable storage costs. Our approach takes advantage of the optimality of the representation under the tensor algebra described to determine to which class an image belongs. We extend our approach to a method that can determine specific pairwise match scores, which could be useful in, for example, object recognition problems where pose/position are different. We demonstrate the promise of our new techniques on the MNIST data set.

Introduction

Image classification is a well-explored problem in which an image is identified as belonging to one of a known number of classes. Researchers seek to extract particular features from which to determine patterns comprising an image. Algo-rithms to determine these essential features include statistical methods such as centroid-based clustering, connectivity/graph-based clustering, distribution-based clustering, and density-based clustering as well as learning algorithms (linear discriminant analysis, support vector machines, neural networks) .

Our approach differs significantly from techniques in the literature in that it uses singular value decompositions (SVD) to form the feature space of an image.

The Singular-Value Decomposition, or SVD for short, is a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler.

For the case of simplicity we will focus on the SVD for real-valued matrices and ignore the case for complex numbers.

```
A = U . Sigma . V^T
```

where A is the real m x n matrix that we wish to decompose, U is an m x m matrix, Sigma (often represented by the uppercase Greek letter Sigma) is an m x n diagonal matrix, and V^T is the transpose of an n x n matrix where T is a superscript, and U is orthogonal, Σ is diagonal, and V is orthogonal.

- The diagonal values in the Sigma matrix are known as the singular values of the original matrix A. The columns of the U matrix are called the left-singular vectors of A, and the columns of V are called the right-singular vectors of A.

The SVD is calculated via iterative numerical methods. The SVD is used widely both in the calculation of other matrix operations, such as matrix inverse, but also as a data reduction method in machine learning. SVD can also be used in least squares linear regression, image compression, and denoising data.

SVD Basis:

The following functions help in creating a basis given a set of images. Applied to images of each digits, we obtain the left singular vectors representative of each digit, which can be considered to form a basis for the digits.
`loss` projects an image into the vectors defined by U and tries to reconstruct it back using the inverse operation. But the process results in loss of information, which we compare with the original sample. Whichever digit's basis leads to least loss of information captures the digit maximum, and can be treated as the label.

Formulation:

The SVD of a matrix $\mathbf{A}$ of dimension dxN, where d is the size of each input sample, and N the number of input samples is represented by:

$$A = U\Sigma V^T$$

where U is **d x d, Σ(sigma)** is a diagonal matrix of dimensions **d x N** and **V^T** of dimensions **N x N.**

Left Singular Vectors-

Let **U**k be the **k** best left singular vectors formed by the **U** matrix. i.e. ,

$$U_k = U[:, : k]$$

Let the input sample be z, then we have:

$$\alpha = U_k^T z$$
$$z' = U_k z$$

Classifying digits-

If **Uk(i)** is the **U** for the i-th digit:

$$z_i' = U_k(i) z$$
$$c = \arg\min_i \|z - z_i'\|_2$$

The concrete steps taken for digit classification are as follows:

1.Ingest the **binary** data files into arrays that can be visualized as digit images.

- We have two sets: Training images and Test images.

2.Make a linear vector space representation of the images by simple unfolding.

3.For each digit find the corresponding representation matrix and factorize it.

4.Store the matrix factorization results in a suitable data structure. (These results comprise the classifier training.)

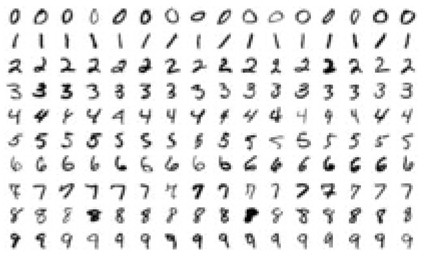
- One of the matrix factors is seen as a new basis.

5.For a given test image (and its linear vector space representation) find the basis that approximates it best. The corresponding digit is the classifier prediction for the given test image.

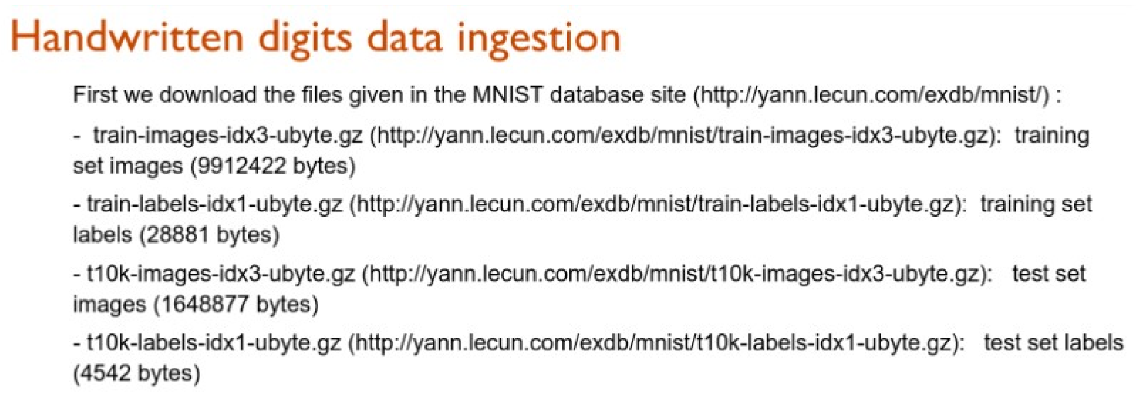
6.Evaluate the classifier(s) over all test images and compute accuracy.

About the dataset:

The **MNIST database** is a large database of handwritten digits that is commonly used for training various image processing systems.Consists of the black and white images with 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.



The MNIST database contains 60,000 training images and 10,000 testing images.



Handwritten digits data ingestion

First we download the files given in the MNIST database site (http://yann.lecun.com/exdb/mnist/) :

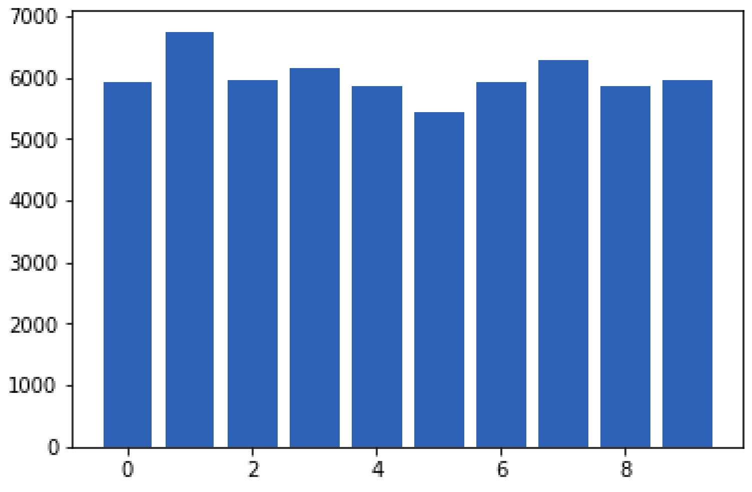
- train-images-idx3-ubyte.gz (http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz): training set images (9912422 bytes)

- train-labels-idx1-ubyte.gz (http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz): training set labels (28881 bytes)

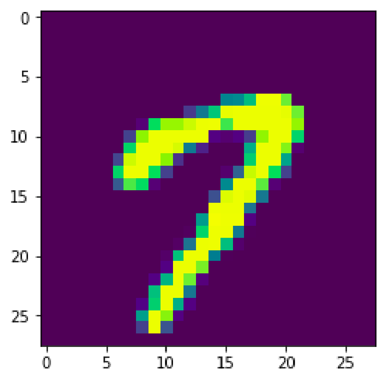
- t10k-images-idx3-ubyte.gz (http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz): test set images (1648877 bytes)

- t10k-labels-idx1-ubyte.gz (http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz): test set labels (4542 bytes)

Distribution of digits in dataset:-



Sample input digit:-



Classification of digits

For classification, we treat the digits in three different but equivalent formats:
1. As 16 × 16 gray scale images,

2. As functions of two variables, s = s(x, y), and
3. As vectors in R^256 .

A simple classification algo (k-means clustering):

1. Training:

Given the manually classified training set, compute the means (centroids) m i , i = 0, . . . , 9, of all the 10 classes.

2. Classification:

For each digit in the test set, classify it as k if m k is the closest mean.

Disadvantage:

It turns out that for our test set, the success rate of this algorithm is around 75%, which is not good enough.

The reason for this relatively bad performance is that the algorithm does not use any information about the variation within each class of digits.

Classification Using SVD Bases

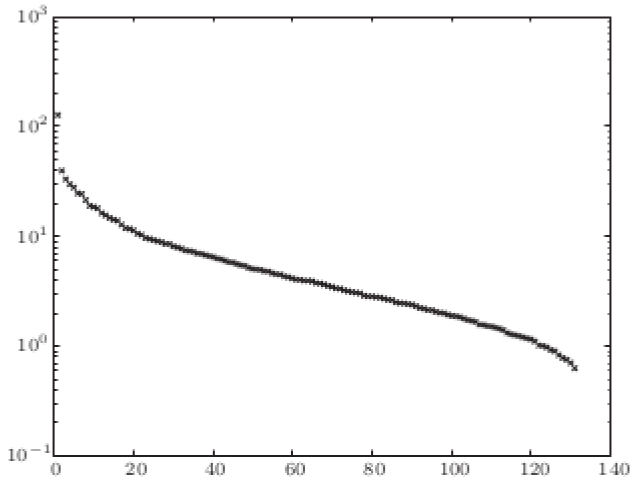
Let A ∈ R m×n , with m = 256, be the matrix consisting of all the training digits of one kind, the 3s, say. The columns of A span a linear subspace of R m.

The idea now is to 'model' the variation within the set of training digits of one kind using an orthogonal basis of the subspace. An orthogonal basis can be computed using the SVD, and A can be approximated by a sum of rank-one matrices,

$$A = \sum_{i=1}^{k} \sigma_i u_i v_i^T,$$

for some value of k. Each column in A is an image of a digit 3, and therefore the left singular vectors u i are an orthogonal basis in the 'image space of 3s'.

We will refer to the left singular vectors as 'singular images'. From the matrix approximation properties of the SVD, we know that the first singular vector represents the 'dominating' direction of the data matrix. Therefore, if we fold the vectors u i back to images, we expect the first singular vector to look like a 3, and the following singular images should represent the dominating variations of the training set around the first singular image. Following is the figure to illustrate the singular values and the first three singular images for the training set 3s.



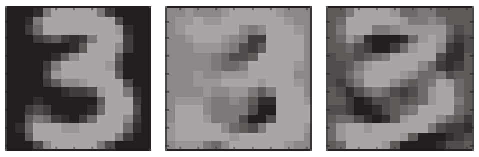


Fig: Singular values plot and top 3 singular values image retrieval

The SVD basis classification algorithm will be based on the following assumptions:-

(1) Each digit (in the training and test sets) is well characterized by a few of the first singular images of its own kind. The more precise meaning of 'a few' should be investigated by experiment.

(2) An expansion in terms of the first few singular images discriminates well between the different classes of digits.
(3) If an unknown digit can be better approximated in one particular basis of singular images, the basis of 3s say, than in the bases of the other classes, then it is likely that the unknown digit is a 3.

Thus, we should compute how well an unknown digit can be represented in the ten different bases. This can be done by computing the residual vector in least squares problems of the type-

$$\min_{\alpha_i} \left\| z - \sum_{i=1}^{k} \alpha_i u_i \right\|,$$

where z represents an unknown digit, and u i the singular images. We can write this problem in the form -

$$\min_{\alpha} \|z - U_k \alpha\|_2,$$

where U k = u 1 u 2 · · · u k . Since the columns of U k are orthogonal, the solution of this problem is given by α = U k T z, and the norm of the residual vector of the least squares problems is -

$$\|(I - U_k U_k^T) z\|_2.$$

An SVD basis classification algorithm:

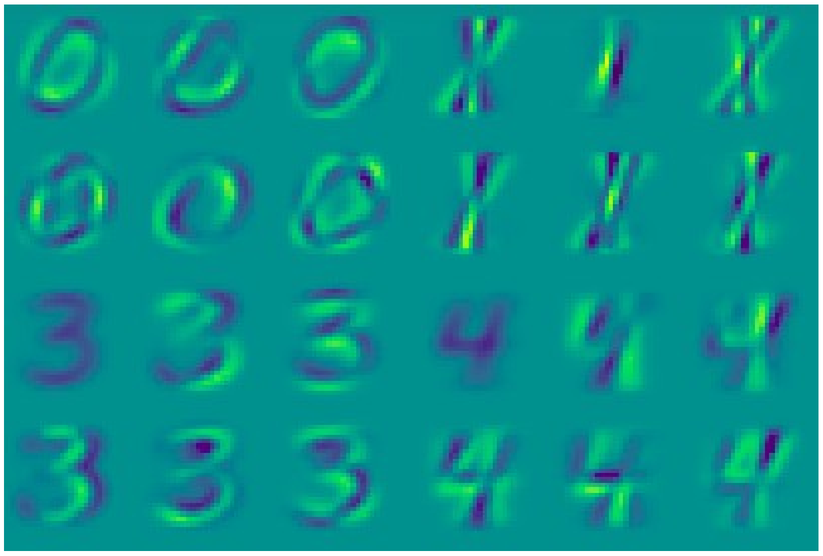
1. Training:

For the training set of known digits, compute the SVD of each class of digits, and use k basis vectors for each class.

2. Classification:

For a given test digit, compute its relative residual in all ten bases. If one residual is significantly smaller than all the others, classify as that. Otherwise give up.

Retrieving and classifying images using k basis=9 (i.e. top 9 singular vectors of U are taken) :-



Results:-

To measure the accuracy of our classification, we compute the recognition rate for the entire test data as follows:

For k = 9, **classification accuracy = 94%** on test data,
where k is top k singular vectors of U matrix.

Code Github Link:-

https://github.com/kishan811/SVD-for-digit-classification-CUDA

Research Papers and References:-

1. Image classification using local tensor singular value decompositions
by- Elizabeth Newman, Misha Kilmer and Lior Horesh

Link:- https://github.com/kishan811/SVD-for-digit-classification-CUDA/blob/master/Research_paper.pdf

2. Matrix-Methods-in-Data-Mining-and-Pattern-Recognition (Watkins book)
Chapter 10- Classification of Handwritten Digits

Link:- https://github.com/kishan811/SVD-for-digit-classification-CUDA/blob/master/image_classification_using_svd(Watkins%20book).pdf

3. Factorization strategies for third-order tensors
by- Misha E. Kilmer and Carla D. Martin

Link:- https://github.com/kishan811/SVD-for-digit-classification-CUDA/blob/master/research_paper_2.pdf