Image classification using dictionary on patches

Pradyot Prakash

130050008

Yash Sanghvi

13D070042

CS 663: Digital Image Processing

Course Project

Department of Computer Science and Engineering

Indian Institute of Technology, Bombay

Abstract

Classification is an important problem in Machine Learning. It has been applied to numerous areas including images. This project is an exploratory endeavor to classify images using patch based dictionaries learned on the same. The basic intuition is to learn a dictionary for all the classes and then sparse code the data to learn its coefficients. These coefficients acts as indicators of the label of class to which the data point belongs to. A classification algorithm is built using these. We were able to get accuracies to the tune of 95% with this approach. It does not beat the state-of-the-art systems, but it does really well compared to basic feed forward neural network systems.

Contents

Chapter 1

Dictionaries

Dictionaries can be understood as a generalization of the idea of bases in linear algebra. The notion of dictionaries comes while trying to come up with a solution for the following problem.

Suppose that we have a set of training data points, labeled as $\mathbf{X_1}, \mathbf{X_2}, .., \mathbf{X_n}$ in some `d` dimensional space. Define,

$$\mathbf{X} = [\mathbf{X_1}\mathbf{X_2} \dots \mathbf{X_n}]$$

to be a `dxn` matrix. Our goal is to find a set called **dictionary**,

$$\mathbf{A} = [\mathbf{A_1}\mathbf{A_2} \dots \mathbf{A_m}]$$

containing `m` vectors in the `d` dimensional space such that X can be written as a linear combination of $\mathbf{A_1}, \mathbf{A_2}, \dots, \mathbf{A_m}$, i.e.,

$$\mathbf{X_i} = w_{1i}\mathbf{A_1} + w_{2i}\mathbf{A_2} + \cdots + w_{mi}\mathbf{A_m}$$

which is equivalent to

$$\mathbf{X_i} = \mathbf{A}\mathbf{W_i}$$

where

$$\mathbf{W_i} = [w_{1i}w_{2i} \dots w_{mi}]^T$$

For the sake of brevity, the entire composition can be written as,

$$\mathbf{X} = \mathbf{AW}$$

where

$$\mathbf{W} = [\mathbf{W_1 W_2} \dots \mathbf{W_n}]$$

Note that the components of the set A are called **atoms**. Also, $\mathbf{X} \in \mathbb{R}^{dxn}$, $\mathbf{A} \in \mathbb{R}^{dxm}$ and $\mathbf{W} \in \mathbb{R}^{mxn}$. Our task at hand is to estimate both $\mathbf{A}$ and $\mathbf{W}$ given $\mathbf{X}$. This can be approximated by trying to minimize the error, $\|\mathbf{X} - \mathbf{AW}\|_F^2$ where $\|.\|_F$ is the Frobenius norm.

This is done by modeling this as the following optimization problem,

$$\arg\min_{\mathbf{A},\mathbf{W}} \sum_{i=1}^{n} \|\mathbf{X_i} - \mathbf{AW_i}\|_2^2$$

One may wonder as to how does a dictionary differ from the basis set. One can simply choose A to be the `d` dimensional basis set and the coefficients can be accordingly computed. Dictionaries differ in the sense that `m` may or may not be equal to `d`. The cardinality of the basis set is what we call the dimension of the data, in our case, `d`. This basis set can be used to represent all the vectors in the `d` dimensional space. However, for many problems, we do not need to cover the entire wide space, but only a small subset of it. For such cases, we consider a reduced basis of sorts, commonly called the dictionary. For some cases, a small number of atoms, $m < d$ can suffice and for other scenarios, we may need more than `d` atoms.

Furthermore, the notion of orthogonality is often missing in dictionaries. Even in the case of $m = d$, orthogonality may be missing. The number m is a hyper-parameter of the model and needs to be guessed. The dictionary $\mathbf{A}$ is called **undercomplete** if $m < d$ or **overcomplete** in case $m > d$. The first case is similar to representing the data using a smaller number of dimensions. The PCA algorithm tries to do something similar. The second case motivates the sparse dictionary learning problem. The usual notion of dictionaries is extended by to include sparsity in the $\mathbf{W}$ matrix. We wish to

be able to represent $\mathbf{X_i}$ using only a subset of $\mathbf{A}$. To account for this, the optimization problem is tweaked slightly to look like,

$$\arg\min_{\mathbf{A},\mathbf{W}} \sum_{i=1}^{n} \|\mathbf{X_i} - \mathbf{A}\mathbf{W_i}\|_2^2 + \lambda f(\mathbf{W})$$

Here, $f(.)$ is some function defining the sparsity for $\mathbf{W}$. The l_0, l_1 norms are popular choices for $f(.)$.

Chapter 2

Non-Negative Sparse Coding (NNSC)

The notation has been borrowed from [1].

The problem of non-negative matrix factorization can be represented succinctly as the minimization of,

$$C(\mathbf{A}, \mathbf{W}) = \frac{1}{2} \|\mathbf{X} - \mathbf{AW}\|_F^2$$

where $C(.,.)$ is the cost function. Here, the non-negativity constraints are on $\mathbf{A}$ and $\mathbf{W}$, i.e. $\forall ij : A_{ij} \geq 0, W_{ij} \geq 0$.

The problem of non-negative sparse coding adds a specific sparsity function to this function along with some other constraints. Sparsity is added through the addition of

$$f(\mathbf{W}) = \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij}$$

giving us the resultant optimization function

$$C(\mathbf{A}, \mathbf{W}) = \frac{1}{2} \|\mathbf{X} - \mathbf{AW}\|_F^2 + \lambda \sum_{i,j} W_{ij}$$

subject to the constraints $\forall ij : A_{ij} \geq 0, W_{ij} \geq 0$ and $\forall i : \|\mathbf{A_i}\| = 1$. The hyper-parameter λ is assumed to be non-negative.

The problem boils down to minimizing this error function. Due to the lack of a closed form solution to the solution, an iterative approach is taken. Usually, an algorithm identical to gradient descent is used to iteratively reduce the value of the function to a minima.

However, the authors of the paper show that under the above choice of $C(.,.)$, the iteration can be done in a specific way which has certain good properties.

2.1 Non-increasing updates to W

The objective function is non-increasing under the following update rule:

$$\mathbf{W}^{t+1} = \mathbf{W}^{t}.*(\mathbf{A}^{T}\mathbf{X})./(\mathbf{A}^{T}\mathbf{A}\mathbf{W}^{t} + \lambda)$$

where .* and ./ notations are MATLAB compatible. The addition of λ is to all the elements of the matrix. Note that `T` represents the transpose of a matrix and `t` represents time.

2.2 NNSC algorithm

Algorithm 1 NNSC

t $\leftarrow 0$

Initialize $\mathbf{A}^0$ to strictly positive values

$\mathbf{A}^0 \leftarrow \texttt{normc}(\mathbf{A}^0)$

Initialize $\mathbf{W}^0$ to strictly positive values

repeat

  $\mathbf{A1} \leftarrow \mathbf{A}^t - \mu(\mathbf{A}^t\mathbf{W}^t - X)(\mathbf{W}^t)^T$

  $\mathbf{A2} \leftarrow \texttt{max}(0, \mathbf{A1})$

  $\mathbf{A}^{t+1} \leftarrow \texttt{normc}(\mathbf{A2})$

  $\mathbf{W}^{t+1} \leftarrow \mathbf{W}^t .* ((\mathbf{A}^{t+1})^T\mathbf{X})./((\mathbf{A}^{t+1})^T(\mathbf{A}^{t+1})\mathbf{W}^t + \lambda)$

  t $\leftarrow$ t + 1

until convergence

Chapter 3

Experiments

We conducted the experiments using the above mentioned coding approach on the MNIST dataset of handwritten digits. This data set contains 28 x 28 images the digits 0 to 9 written by people in different styles. There are 60,000 images for testing and 10,000 for testing. For the purpose of this experiment, the images were resized to 14 x 14 and the random Forest classifier with 100 trees was used.

The approach taken is as follows:

1. We consider images of one class at a time, $label \in 0, 1, \ldots, 9$. Label the set of images $\mathbf{X}^{label}$

2. Set the number of atoms to be learned for each class, say $\mathbf{A}^{label}$

3. Use NNSC to learn the atoms $\mathbf{A}^{label}$ along with sparsity regularization. The hyper-parameter λ dictates this. Too high a sparsity, will lead to fewer atoms being used to determine the image reconstruction, and vice-versa

4. Now that we have the learned atoms $\{\mathbf{A}^0, \mathbf{A}^1, \ldots \mathbf{A}^9\}$

5. Following this we combine all the atoms and the data points together, i.e., set

$$\mathbf{A} = [\mathbf{A}^0 \mathbf{A}^1 \ldots \mathbf{A}^9]$$

$$\mathbf{X} = [\mathbf{X}^0 \mathbf{X}^1 \ldots \mathbf{X}^9]$$

6. Using these $\mathbf{A}$, we fit $\mathbf{X}$ to it, i.e., we keep $\mathbf{A}$ constant and find the $\mathbf{W}$ which minimizes the cost function. In other words, we sparse code the data. Refer to Algorithm 2 for a detailed description. This step gives us the coefficients that

Algorithm 2 Sparse coding the learned dictionary

1: $t \leftarrow 0$

2: Initialize $\mathbf{W}^0$ to strictly positive values

3: **repeat**

4: $\quad \mathbf{W}^{t+1} \leftarrow \mathbf{W}^t . * (\mathbf{A}^T\mathbf{X})./(\mathbf{A}^T\mathbf{A}\mathbf{W}^t + \lambda)$

5: $\quad t \leftarrow t + 1$

6: **until** convergence

define the reconstruction of images from the atoms. Note that it is possible that an image of, say class 7, may have components from different classes. The algorithm does nothing to make this distinction. However, we would expect that the atoms corresponding to class 7 will have a major contribution to the data points from class 7. Our project is trying to validate the same hypothesis, and as it turns out, it does it really well.

7. Now, we fit a Random Forest classifier with 100 trees. The input to the classifier are the sparse coded coefficients and the labels are their corresponding classes.

8. The validation of the model is done in a similar manner. The testing data is sparse coded using the dictionary matrix, $\mathbf{A}$, and the corresponding sparse coefficient vectors are fed to the trained classifier. The classifier tells us which class the coefficient vector should belong to.

Chapter 4

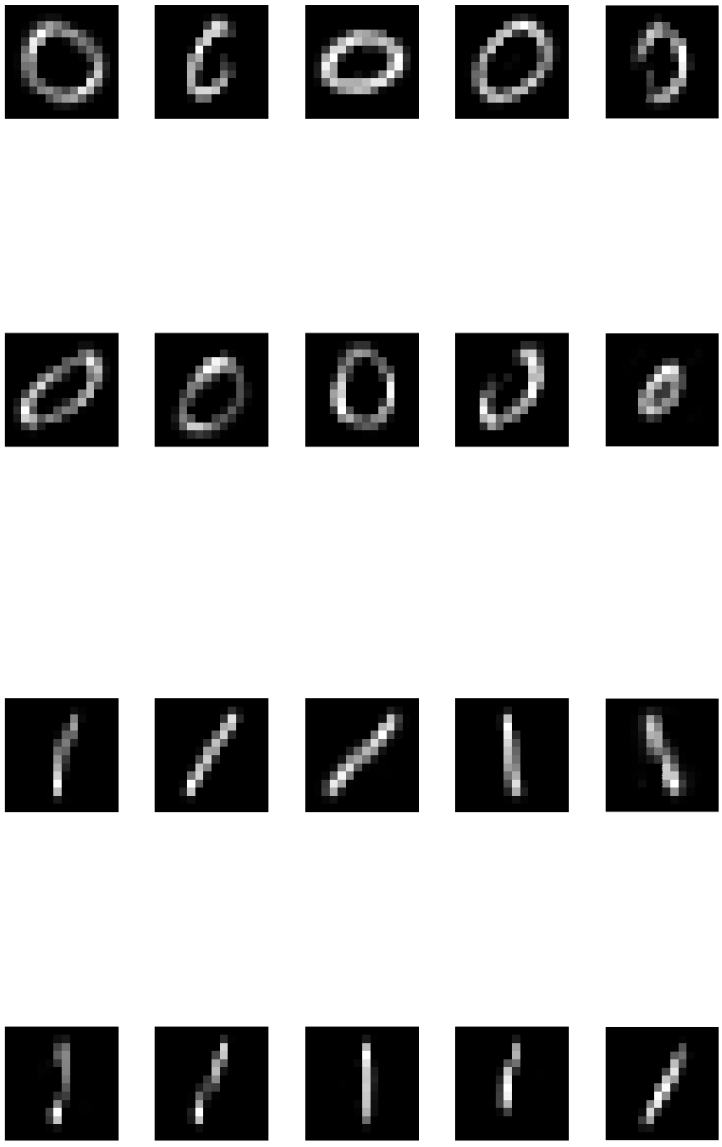
Results

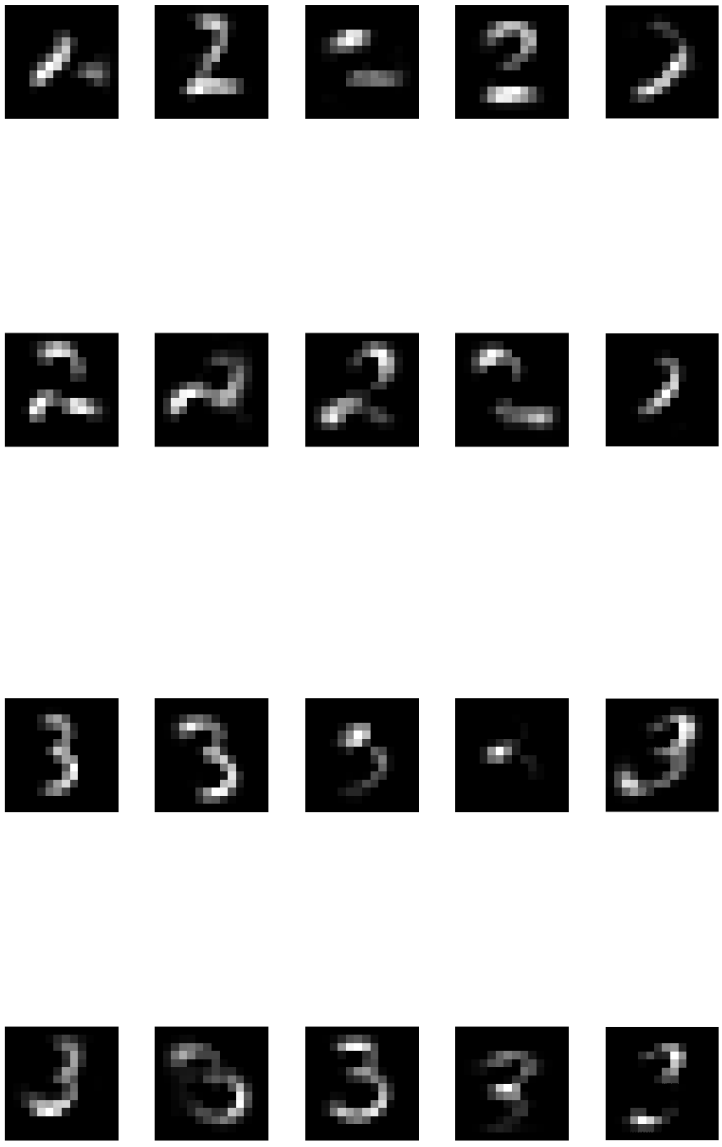
The average classification accuracy was 94.610000% for the 10,000 testing samples. The class-wise breakdown is as follows:

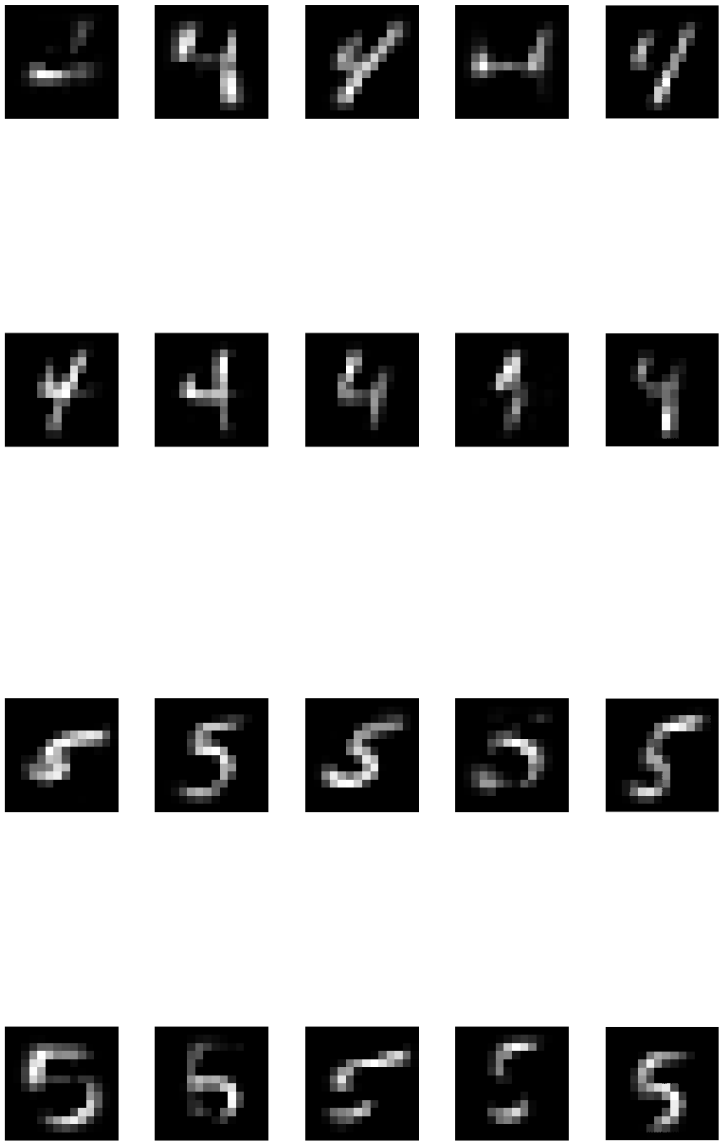
Digit	Accuracy(%)
0	98.36
1	98.85
2	96.31
3	94.65
4	95.11
5	92.15
6	97.07
7	94.55
8	87.16
9	90.98

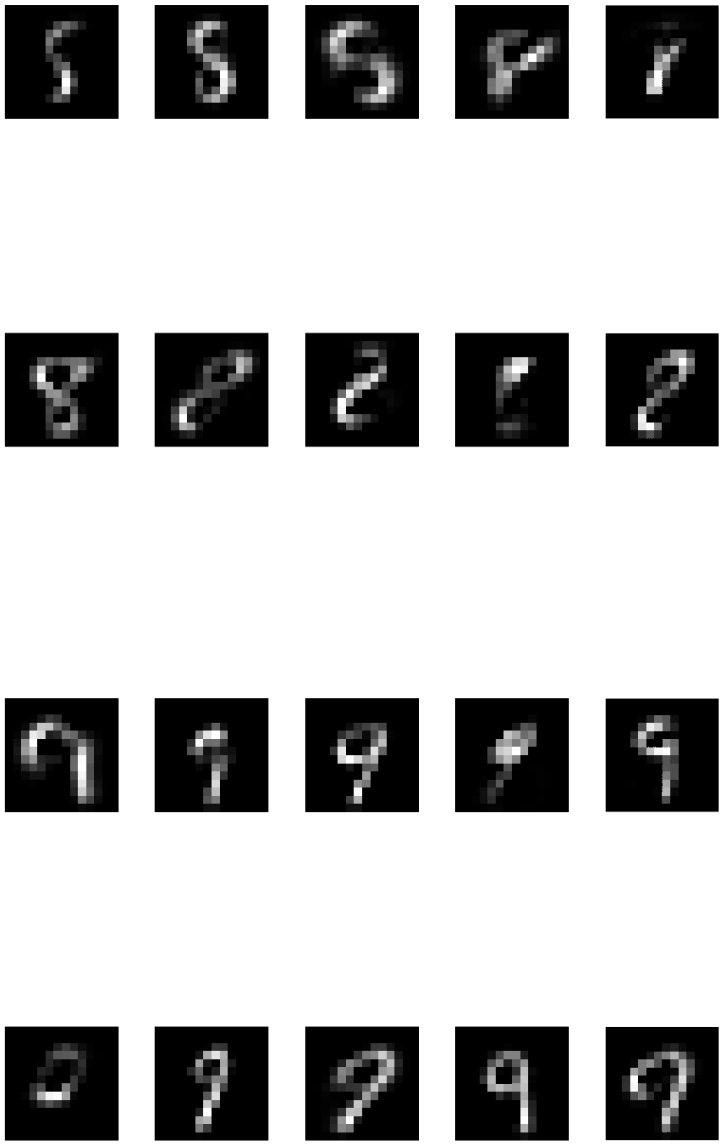
In this model, 10 atoms were used for each class. The coefficient λ was set to 0.

The shapes of the atoms are:









On analysis, we observed that on increasing λ, the classification accuracies continually decrease. The number of atoms was fixed to 10 for each class. This can be attributed to the fact that the coefficients that the model learned were mostly localized to the atoms representing the specific point. So, on penalizing the coeffi-

cients through regularization, the coefficients attained small values and were not able to accurately give an indication of the class they belong to. The average accuracies corresponding to different values of λ are as follows.

λ	Accuracy(%)
0	94.61
1	71.42
2	57.76
5	45.18
7	35.38
10	31.97
15	33.36
20	31.47
25	19.85

The trend in accuracies as a function of the number of atoms has been summarized in the next table.

# of atoms for each class	Accuracy(%)
1	91.04
2	94.21
5	95.00
8	93.78
10	94.71
12	95.16
15	94.14
20	95.53

This tells us the the accuracy as a function of atoms is more-or-less constant. The change in sparsity regulator has quite a significant effect on the accuracies, but the number of atoms has negligible effect. This may be because of the fact that the shape

of the digits differ quite a lot from each other and the random forest classifier is able to distinguish the coefficient vectors even from a fewer number of atoms. The small variation of 1-2% in the accuracies may be because trees are randomly constructed in the random forest algorithm and some ups and downs are because of that.

We also conducted an experiment in which we added some noise to the source images. The idea behind this experiment was to see if the learned atoms are able to get rid of the noise present in the original images. If that works correctly, then the classification accuracies should be near to the actual results.

Bibliography

[1] P. O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 557–565. IEEE, 2002.