ECE532 Final Project

Dictionary Learning

Declan Gundrum **djgundrum@wisc.edu**
Spencer Du **sdu42@wisc.edu**
Atessa Amjadi **aamjadi@wisc.edu**

December 2019

1 Abstract

We will be introducing the concept of Dictionary Learning. Dictionary Learning is a very useful implementation of sparse coding. With this, we will be able to train an unsupervised model that is able to find the basic components of a given matrix $\mathbf{X}$. Recall that in an unsupervised model there is no need for any known solution data. Using this algorithm, we will be able to decompose matrix $\mathbf{X}$ into two matrices $\mathbf{D}$ and $\mathbf{R}$. The properties of these two matrices will be very unique. Matrix $\mathbf{D}$ will be a dictionary matrix holding the most important features of $\mathbf{X}$, and $\mathbf{R}$ will be the sparsest possible matrix that minimizes the two-norm difference of $\mathbf{X} - \mathbf{DR}$.

This seems very complicated, but the payoff is incredible. By finding these sparse solutions, we are able to apply much better pattern-recognition to data that is unknown to the system beforehand. This is applied to many different areas, but some of the most recognized are the uses in image classification (whether it is an image of a cat, or your grandma), and signal recovery (if something fits into a pattern, or if it doesn't and it just looks like it does). But how is it done? There are many different algorithms that have been created, but all have a similar concept implemented. From a very basic standpoint, what is happening behind the scenes is an iterative approach, where we first find a sparse solution of a random new sample, then follow that by updating and optimizing our dictionary to hold the information from this sample iteration. We will use multiple ideas from class like SVD decomposition, k-means algorithm to understand this new algorithm.

2 Background

This idea was originally proposed as something similar to what our own brain does when it sees something with a seemingly recognizable pattern. What they meant by this is, humans have a weirdly good knack at seeing passed noise in certain datum, and they wanted to be able to recreate that with a computer. For example, sometimes you will see an image such as the image 1 to the right, and even though there are a lot of lines through the figure, your brain is still able to easily tell that there are two pears in the image. All of this was also done even though your brain has never actually seen this image before. Bruno Olhausen took this inspiration and began to think intuitively about the problem at hand. What he eventually concluded on, is that our brain takes the image, and decomposes it to find the most important parts. After finding that, it reconstructs the image in your head with a different image that highlights key parts if it finds a pattern. This brought about the matrix decomposition talked about in the abstract intro. The dictionary $\mathbf{D}$ will be the matrix that holds k vectors containing the most parts of the image, and the matrix $\mathbf{R}$ will contain the information to correctly reconstruct our original matrix $\mathbf{R}$, this time with a little less noise in the way. One solution that they formulated for this uses LASSO to first find the matrix $\mathbf{R}$, then update $\mathbf{D}$ using K-SVD. They then repeat this until the difference between the original matrix and our regenerated matrix $\mathbf{DR}$ falls below a certain confidence threshold.

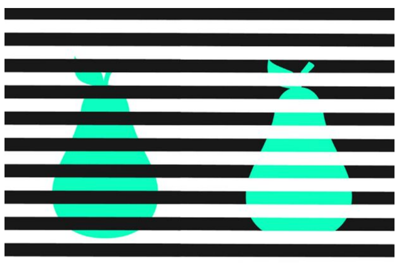
Figure 1: green pear[1]

2.1 Formulation

Mathematically speaking, given the input $\mathbf{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n], \boldsymbol{x}_i \in \mathbb{R}^m$, find a dictionary $\mathbf{D} \in \mathbb{R}^{m\times k} : \mathbf{D} = [\boldsymbol{d}_1, \cdots, \boldsymbol{d}_k]$ with k columns referred to as *atoms* and a representation $\mathbf{R} = [\boldsymbol{r}_1, \cdots, \boldsymbol{r}_n]$, $\boldsymbol{r}_i \in \mathbb{R}^k$ such that both $\|\mathbf{X} - \mathbf{DR}\|_F^2$ is minimized and the representations $\boldsymbol{r}_i$ are sparse enough. This can be formulated as the following optimization problem:

$$\begin{gathered}\min_{\mathbf{D},\mathbf{R}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \mathbf{D}\boldsymbol{r}_i\|_2^2 + \lambda\|\boldsymbol{r}_i\|_0 \\ \text{where } \|\boldsymbol{d}_j\|_2 \leq 1 \text{ for } j = 1, \cdots, k \text{ and } \lambda > 0\end{gathered} \tag{1}$$

[1]This figure is from https://www.dailystar.co.uk/real-life/green-pears-optical-illusion-goes-19936268.

$\|\boldsymbol{r}_i\|_0$ is the number of nonzero element in $\boldsymbol{r}_i$, i.e. $\|\boldsymbol{r}_i\|_0 = \sum_{j=1}^{k} \mathbb{1}_{\{\boldsymbol{r}_{ik} \neq 0\}}$. Since we know that l_0-norm is not convex, we can replace it with l_1-norm (same tactic when trying to find a sparse solution in video lecture 5.3). So We rewrite problem (1) as

$$\begin{gathered}\min_{\mathbf{D},\mathbf{R}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \mathbf{D}\boldsymbol{r}_i\|_2^2 + \lambda \|\boldsymbol{r}_i\|_1 \\ \text{where } \|\boldsymbol{d}_j\|_2 \leq 1 \text{ for } j = 1, \cdots, k \text{ and } \lambda > 0\end{gathered} \tag{2}$$

This problem can also be written as

$$\begin{aligned}&\min_{\mathbf{D},\mathbf{R}} \sum_{i=1}^{n} \|\boldsymbol{r}_i\|_1 \\ &\text{s.t. } \sum_{i=1}^{n} \|\boldsymbol{x}_i - \mathbf{D}\boldsymbol{r}_i\|_2^2 < \varepsilon \\ &\text{and } \|\boldsymbol{d}_j\|_2 \leq 1 \text{ for } j = 1, \cdots, k\end{aligned} \tag{3}$$

But we will focus on form (2).

2.2 Solve the problem

The problem above will be solved as a convex optimization problem. This will be done in two different steps, each step updates either $\mathbf{D}$ or $\mathbf{R}$, and holds the other matrix fixed. Take note that near the end of the description above, it is specified that the 2-norm of each column of $\mathbf{D}$ must fall under a constraint. This constraint is arbitrary in all cases but is necessary; if there is no constraint given to each of the columns, the algorithm specified will always be able to minimize $\mathbf{R}$ more while increasing the values in $\mathbf{D}$ to compensate. Here is a description of dictionary learning algorithm.

Algorithm 1 Dictionary learning

```
Require: Input dataset $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $k$.
Ensure: Dictionary $\mathbf{D} \in \mathbb{R}^{m \times k}$ and sparse representation $\mathbf{R} \in \mathbb{R}^{k \times n}$
1: Initialization: Randomly pick $k$ columns from $\mathbf{X}$ or pick the first $k$ left singular vectors of $\mathbf{X}$ to form dictionaty $\mathbf{D}^{(0)}$.
2: for $l = 1, 2, 3, \cdots$ do
3:     Sparse coding to get $\mathbf{R}^{(l)}$
4:     Update dictionary to get $\mathbf{D}^{(l)}$
5:     if $\|\mathbf{X} - \mathbf{DR}\|_F^2 < \varepsilon$ then
6:         Stop
7:     end if
8: end for
```

2.2.1 Sparse Coding

In this part, given the data $\mathbf{X}$ and the dictionary $\mathbf{D}$, we need to find $\mathbf{R}$ that minimize $\|\mathbf{X} - \mathbf{DR}\|_F^2$ and make $\mathbf{R}$ as sparse as possible. Obviously we can do that by applying LASSO regression using the cost function

$$\min_{\mathbf{R}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \mathbf{D}\boldsymbol{r}_i\|_2^2 + \lambda \|\boldsymbol{r}_i\|_1 \tag{4}$$

There are other ways to get a sparse representation. **OMP** (Orthogonal Matching Pursuit) is a popular choice. We won't delve into this. But if you are familiar with Python, you can give it a try in the main activity later.

2.2.2 Update the dictionary

After the sparse coding task, the next is to search for a better dictionary $\mathbf{D}$. However, finding the whole dictionary all at a time is impossible, so the process is to update only one column of

$\mathbf{D}$ each time, while fixing $\mathbf{R}$. The update of the j-th column is done by rewriting the penalty term as

$$\begin{aligned}\|\mathbf{X}-\mathbf{D}\mathbf{R}\|_F^2 &= \left\|\mathbf{X}-\sum_{t=1}^{k}\boldsymbol{d}_t\boldsymbol{r}_{[t]}\right\|_F^2 \\ &= \left\|\left(\mathbf{X}-\sum_{t\neq j}\boldsymbol{d}_t\boldsymbol{r}_{[t]}\right)-\boldsymbol{d}_j\boldsymbol{r}_{[j]}\right\|_F^2 \\ &= \|\mathbf{E}_j-\boldsymbol{d}_j\boldsymbol{r}_{[j]}\|_F^2\end{aligned} \tag{5}$$

where $\boldsymbol{r}_{[j]}$ denotes the j-th row of $\mathbf{R}$.

By decomposing the multiplication $\mathbf{DR}$ into sum of k rank-1 matrices, we can assume the other $k-1$ terms are assumed fixed, and the j-th remains unknown. After this step, we can solve the minimization problem by approximating the $\mathbf{E}_j$ term with a rank-1 matrix using singular value decomposition, then update $\boldsymbol{d}_j$ with it. However, the new solution of vector $\boldsymbol{r}_{[j]}$ is very likely to be filled, because the sparsity constraint is not enforced.

To cure this problem, Define w_j as

$$w_j = \{i|\boldsymbol{r}_{[j]}(i) \neq 0\}$$

where $\boldsymbol{r}_{[j]}(i)$ refers to i-th entry in row vector $\boldsymbol{r}_{[j]}$. w_j points to examples that use atom $\boldsymbol{d}_j$. Then define $\mathbf{E}'_j$ as a matrix of size $m \times |w_j|$ whose columns are chosen from columns of $\mathbf{E}_j$ according to w_j. So the minimization problem (5) becomes

$$\|\mathbf{E}'_j - \boldsymbol{d}_j\boldsymbol{r}'_{[j]}\|_F^2 = \|\mathbf{E}_j(w_j) - \boldsymbol{d}_j \times \boldsymbol{r}_{[j]}(w_j)\|_F^2$$

and can be done by directly using SVD. SVD decomposes $\mathbf{E}'_j$ into $\mathbf{U\Sigma V}^T$. Then use the **best rank-1 approximation** to update $\boldsymbol{d}_j$ and $\boldsymbol{r}_{[j]}$.

```
Algorithm 2 Dictionary Update
Require: Input dataset $\mathbf{X}$, input dictionary $\mathbf{D}$ and representation $\mathbf{R}$.
Ensure: Updated dictionary $\mathbf{D}$.
1: for $j = 1, \cdots, k$ do
2:     $\mathbf{E}_j \leftarrow \mathbf{X} - \sum_{t\neq j} \boldsymbol{d}_t\boldsymbol{r}_{[t]}$
3:     $w_j \leftarrow \{i|\boldsymbol{r}_{[j]}(i) \neq 0\}$
4:     $\mathbf{r}'_{[j]} \leftarrow \boldsymbol{r}_{[j]}(w_j) = \{\boldsymbol{r}_{[j]}(i)|\boldsymbol{r}_{[j]}(i) \neq 0\}$
5:     $\mathbf{E}'_j \leftarrow \mathbf{E}_j(w_j)$
6:     Perform SVD on $\mathbf{E}'_j$ and use the best rank-1 approximation to update
       $\boldsymbol{d}_t$ and corresponding entries $\mathbf{r}'_{[j]}$ in $\boldsymbol{r}_{[j]}$
7: end for
```

3 Warm Up

(1) Write down K-means algorithm and compare with our dictionary learning algorithm 2.

(2) Complete algorithm 2 step 6.

a) Assume $\mathbf{E}'_j$ has SVD $\mathbf{E}'_j = \mathbf{U\Sigma V}^T$. Remember the best rank-1 approximation for a matrix $\mathbf{E}'_j$ and write it down in terms of $\mathbf{U}, \mathbf{\Sigma}$ and $\mathbf{V}$.

b) Write down how to update $\boldsymbol{d}_j$ and $\boldsymbol{r}'_{[j]}$ using the representation of the best rank-1 approximation for $\mathbf{E}'_j$.

4 Main Activity

(1) The first step in Dictionary Learning is using LASSO regression to make a sparse matrix $\mathbf{R}$, such that it minimizes $\sum_{i=1}^{n} \|\boldsymbol{x}_i - \mathbf{D}\boldsymbol{r}_i\|_2^2 + \lambda\|\boldsymbol{r}_i\|_1$. In this step we will be holding the dictionary matrix $\mathbf{D}$ constant.

a) We choose LASSO regression because it is able to produce very sparse matrices given by the $\sum_{i=1}^{n} \lambda \|\boldsymbol{r}_i\|_1$ regularizer. Here we will explore this. Let $\boldsymbol{w}$ be $\begin{bmatrix} w_1 & w_2 \end{bmatrix}^T$, draw a graph in the $w_1 - w_2$ plane highlighting the region contained by $\|w\|_1 \leq 2$.

b) Given the ellipse $\dfrac{(w_1-3)^2}{16} + \dfrac{(w_2-3)^2}{4} = 1$, find the point that minimizes the $l1$-norm for this contour.

c) From the last part, we will move to Python script provided in `LASSO.ipynb`.

   i) Run through all the sections and comment on what happens to the line of the best fit as we increase λ.

   ii) Use your intuition on making a guess to which of the graphs have the sparsest solution.

   iii) A big part of LASSO and dictionary learning is finding the best λ value for your data. Which of the λ values shows the best fit while not overfitting.

(2) We are performing dictionary learning on images. The starter file `dictionary_learning.ipynb` or `dictionary_learning_start.m` will compute dictionary learning for a specified number of atoms for solving problem (2)

$$\min_{\mathbf{D},\mathbf{R}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \mathbf{D}\boldsymbol{r}_i\|_2^2 + \lambda \|\boldsymbol{r}_i\|_1$$
$$\text{where } \|\boldsymbol{d}_j\|_2 \leq 1 \text{ for } j = 1, \cdots, k \text{ and } \lambda > 0$$

Run the code and answer

a) You are given an image. Using 50 atoms, find the dimensions of matrix **D** and **R** based on the size of the image.

b) Compare what storage is required to hold **X** in comparison to **D** and **R**, then make a comment in relation to dictionary learning.
(We have the storage output in Jupyter Notebook. Matlab users can use command `whos` to find the bytes needed)

References

[1] Boaz B. & David S. (2016). Proofs, beliefs, and algorithms through the lens of sum-of-squares. Retrieved from https://www.sumofsquares.org/public/lec-sphere-dictionary.pdf

[2] Julien Mairal. (2000). Sparse Coding and Dictionary Learning for Image Analysis. Retrieved from http://lear.inrialpes.fr/people/mairal/resources/pdf/ERMITES10.pdf

[3] Wikipedia contributors. (2019). Sparse dictionary learning. In Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/Sparse_dictionary_learning

[4] Morgan. (2016). Sparse coding: A simple exploration.
Retrived from https://blog.metaflow.fr/sparse-coding-a-simple-exploration-152a3c900a7c

[5] Wikipedia contributors. (2019). K-SVD. In Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/K-SVD

[6] Bruno A. Olshausen, & David J. Field. . Sparse coding with an overcomplete basis set: a strategy employed by v1?. Vision Research, 37(23), 0-3325.

[7] Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2009, June). Online dictionary learning for sparse coding. In Proceedings of the 26th annual international conference on machine learning (pp. 689-696). ACM.

[8] Francis Bach. (2010). Sparse methods for machine learning Theory and algorithms. Retrieved from https://www.di.ens.fr/ fbach/Cours_peyresq_2010.pdf

A Activity Solution

Warm Up

(1) Write down K-means algorithm and compare with our dictionary learning algorithm 2.
The algorithm is below

```
Algorithm 3 K-means Algorithm
Require: Input dataset $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $k$.
Ensure: $k$ centroids.
1: Initialization: choose $k$ centroids $\boldsymbol{d}_j^{(0)}, j = 1, 2, \cdots, k$ from columns of $\mathbf{X}$
2: for $l = 1, 2, 3, \cdots$ do
3:     Set the $j$-th entry of $\boldsymbol{r}_i^{(l)}$ equal to 1 with other entries 0 if $\boldsymbol{x}_i$ is closest to $\boldsymbol{d}_j^{(l-1)}$ for every $\boldsymbol{r}_i^{(l)}, i = 1, 2, \cdots, n$
4:     Update $\boldsymbol{d}_j^{(l)}$ equal to mean of the data in this cluster for $j = 1, 2, \cdots, k$
5:     if $\|\mathbf{D}^{(l)} - \mathbf{D}^{(l-1)}\|_F^2 \le \varepsilon$ then
6:         Stop
7:     end if
8: end for
```

Compared K-means with our algorithm 2, atoms in dictionary $\mathbf{D}$ are like centroids in K-means. Step 3 is re-grouping/sparse coding and step 4 is updating the atoms/centroids. Actually the two algorithms are very similar.

(2) Complete algorithm 2 step 6.

a) Assume $\mathbf{E}'_j$ has SVD $\mathbf{E}'_j = \mathbf{U\Sigma V}^T$. Remember the best rank-1 approximation for a matrix $\mathbf{E}'_j$ and write down in terms of $\mathbf{U}, \mathbf{\Sigma}$ and $\mathbf{V}$.
According to Eckart–Young theorem in video lecture 3.3, the best rank-1 approximation for a matrix $\mathbf{E}'_j$ is $\sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T$.

b) Write down how to update $\boldsymbol{d}_j$ and $\boldsymbol{r}'_{[j]}$ using the representation of the best rank-1 approximation for $\mathbf{E}'_j$.
To minimize $\|\mathbf{E}'_j - \boldsymbol{d}_j \boldsymbol{r}'_{[j]}\|_F^2$, we will assign like below:

$$\boldsymbol{d}_j = \boldsymbol{u}_1, \boldsymbol{r}'_{[j]} = \sigma_1 \boldsymbol{v}_1^T$$

Main Activity

(1) The first step in Dictionary Learning is using LASSO regression to make a sparse matrix $\mathbf{R}$, such that it minimizes $\sum_{i=1}^n \|\boldsymbol{x}_i - \mathbf{D}\boldsymbol{r}_i\|_2^2 + \lambda\|\boldsymbol{r}_i\|_1$. In this step we will be holding the dictionary matrix $\mathbf{D}$ constant.

a) We choose LASSO regression because it is able to produce very sparse matrices given by the $\sum_{i=1}^n \lambda\|\boldsymbol{r}_i\|_1$ regularizer. Here we will explore this. Let $\boldsymbol{w}$ be $\begin{bmatrix} w_1 & w_2 \end{bmatrix}^T$, draw a graph in the $w_1 - w_2$ plane highlighting the region contained by $\|w\|_1 \le 2$.
See pictures next page.

b) Given the ellipse $\dfrac{(w_1-3)^2}{16} + \dfrac{(w_2-3)^2}{4} = 1$, find the point that minimizes the $l1$-norm for this contour.
See pictures next page.

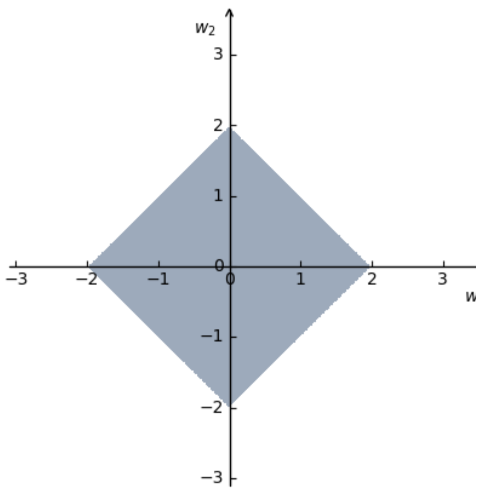

Figure 2: a)

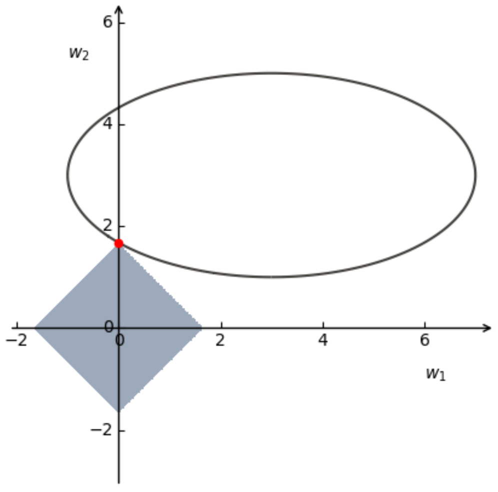

Figure 3: b)

c) From the last part, we will move to Python script provided in `LASSO.ipynb`.

  i) Run through all the sections and comment on what happens to the line of the best fit as we increase λ.
  As λ increases, the line of the best fit represents the data worse.

  ii) Use your intuition on making a guess to which of the graphs have the sparsest solution.
  The highest λ value has the most sparse solution, all the features are zero in the solution vector.

  iii) A big part of LASSO and dictionary learning is finding the best λ value for your data. Which of the λ values shows the best fit while not overfitting.
  $\lambda = 0.01$ seems to fit most of the data very well. Other answers can be accepted with a good explanation.

(2) We are performing dictionary learning on images. The starter file `dictionary_learning.ipynb` or `dictionary_learning_start.m` will compute dictionary learning for a specified number of atoms for solving problem 2

$$\min_{\mathbf{D},\mathbf{R}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \mathbf{D}\boldsymbol{r}_i\|_2^2 + \lambda\|\boldsymbol{r}_i\|_1$$
$$\text{where } \|\boldsymbol{d}_j\|_2 \leq 1 \text{ for } j = 1, \cdots, k \text{ and } \lambda > 0$$

Run the code and answer

a) You are given an image. Using 50 atoms, find the dimensions of matrix $\mathbf{D}$ and $\mathbf{R}$ based on the size of the image.
$\mathbf{D}$ is 907×50 and $\mathbf{R}$ is 50×736.

b) Compare what storage is required to hold $\mathbf{X}$ in comparison to $\mathbf{D}$ and $\mathbf{R}$, then make a comment in relation to dictionary learning.
(We have the storage output in Jupyter Notebook. Matlab users can use command `whos` to find the bytes needed)
By comparison, it is concluded that less storage is required for $\mathbf{D}$ and $\mathbf{R}$ in comparison to $\mathbf{X}$ since dictionary learning finds a sparser representation.