STK-MAT2011
Dictionary learning for sparse coding

Student: Emil Haugen
Supervisor: Odd Kolbjørnsen

14th February 2019

Abstract

In this paper we examine dictionary learning for sparse encoding, an unsupervised technique used for representing a vector as a sparse linear combination of basis vectors. After presenting the initial optimization problem, we look closely at the two subroutines that we alternate between to solve the problem. Finally, we summarize the results in an algorithm which can be used to learn a dictionary and a sparse code from the data.

1 Introduction

To begin with, we introduce the concepts of dictionary learning for sparse coding. This is an unsupervised representation learning method for extracting key features from a data vector encoded in a sparse representation.

More precisely, given a set of data vectors $\{\mathbf{u}^{(t)}\}_{t=1}^{T} \subset \mathbb{R}^m$, we want to find a set of vectors $\{\mathbf{x}^{(t)}\}_{t=1}^{T} \subset \mathbb{R}^k$ and a matrix $\mathbf{D} \in \mathbb{R}^{m\times k}$ such that we solve the optimization problem

$$\min_{D} \frac{1}{T} \sum_{t=1}^{T} \min_{\mathbf{x}^{(t)}} \frac{1}{2} \left\| \mathbf{u}^{(t)} - \mathbf{D}\mathbf{x}^{(t)} \right\|_2^2 + \lambda \left\| \mathbf{x}^{(t)} \right\|_1 \tag{1}$$

where $\|\cdot\|_1$ and $\|\cdot\|_2$ are the ℓ_1 and ℓ_2-norms respectively. The first term in the sum above is small when the difference between the true data $\mathbf{u}^{(t)}$ and the reconstructed data $\mathbf{D}\mathbf{x}^{(t)}$ is small. The second term is a regularization term that is small when the $\mathbf{x}^{(t)}$ vectors are sparse, i.e. they have many zeros. Thus (1) is a trade-off between getting an accurate reconstruction of the data and keeping the $\mathbf{x}^{(t)}$ sparse.

The parameter λ is a positive real number, which acts as a regularization tuning parameter. The matrix $\mathbf{D}$ is referred to as the *dictionary*. We may have $k > m$, such that the representation $\mathbf{x}^{(t)}$ has higher dimension, but the ℓ_1-regularization term will force this vector to be sparse, i.e. it will have many zero entries. Finally, we constrain the columns of $\mathbf{D}$ to be of unit ℓ_2-norm. If not, we could solve the minimization problem (1) by taking the entries of $\mathbf{x}^{(t)}$ to be arbitrarily small and increase the entries of $\mathbf{D}$ to decrease the first term of (1).

We will tackle the minimization problem (1) one variable at a time, alternating between optimizing with respect to the dictionary $\mathbf{D}$ and the sparse representation $\mathbf{x}^{(t)}$, keeping one fixed while varying the other at each step. An online algorithm for solving the problem is proposed by (Mairal et al. 2009). We will follow the same strategy, but do it offline instead.

In Section 2.1 and Section 2.2 we look closely at the subroutines where we optimize (1) with respect to $\mathbf{x}^{(t)}$ and $\mathbf{D}$ respectively. In Section 2.3, we observe that the dictionary update step can be done more efficiently than the naive way suggested immediately. Finally, in Section 2.4, we give a global algorithm where we bring together the previous sections and use the entire data set to produce a learned dictionary and sparse code.

2 Dictionary learning and sparse coding

2.1 Sparse coding using the LASSO

Creating a sparse code $\mathbf{x}^{(t)}$ is the subproblem of minimizing the loss function

$$l(\mathbf{u}^{(t)}, \mathbf{D}) = \frac{1}{2}\left\|\mathbf{u}^{(t)} - \mathbf{D}\mathbf{x}^{(t)}\right\|_2^2 + \lambda\|\mathbf{x}^{(t)}\|_1 \tag{2}$$

given an input vector $\mathbf{u}^{(t)}$ and a dictionary $\mathbf{D}$ which are considered fixed. Thus we only minimizie (2) with respect to the vector $\mathbf{x}^{(t)}$ and view it as independent from $\mathbf{D}$. In reality, $\mathbf{x}^{(t)}$ naturally depends on $\mathbf{D}$, but assuming independence is a useful heuristic for solving the problem approximately, which is originally NP-hard (Tillmann 2015). Then, (2) is the well known LASSO regression problem from statistics (Tibshirani 1996). The non-diferentiable ℓ_1 penalty term means that we can not use e.g. gradient descent in a straightforward manner. However, both terms of the objective function are convex so the sum is convex. Thus we are guaranteed to have a unique global optimum. One widely used algorithm for finding an optimal $\mathbf{x}^{(t)}$ is the least-angle regression (LARS) algorithm (Efron et al. 2004). This is also the one suggested by (Mairal et al. 2009) in their online algorithm.

2.2 Dictionary update

For the dictionary update step, we consider $\mathbf{x}^{(t)}$ to be fixed in (1) and vary only $\mathbf{D}$. Thus, the ℓ_1 term is constant, and so we get the least squares problem. Unlike in ordinary least squares, we are optimizing with respect to a matrix.

$$\min_D \frac{1}{T}\sum_{t=1}^{T} \frac{1}{2}\left\|\mathbf{u}^{(t)} - \mathbf{D}\mathbf{x}^{(t)}\right\|_2^2. \tag{3}$$

One way to do this, is to iterate through the columns $\mathbf{D}_{\cdot,j}$ of $\mathbf{D}$, for $j = 1, \dots, k$. Thinking of each column of $\mathbf{D}$ as a *block* of variables, this method is a variation of *block coordinate descent*. Unlike gradient descent, this method has the advantage of being non-parametric, in that it does not require a *learning rate* that must be tuned in order to control the convergence. More precisely, we consider

$$\frac{1}{T}\sum_{t=1}^{T} \frac{1}{2}\left\|\mathbf{u}^{(t)} - \mathbf{D}\mathbf{x}^{(t)}\right\|_2^2 \tag{4}$$

as a function of $\mathbf{D}_{\cdot,j}$ i.e. column j of $\mathbf{D}$, and all other columns are considered constant. We first take the gradient and set it to zero:

$$0 = \frac{1}{T}\sum_{t=1}^{T}\left(\mathbf{u}^{(t)} - \mathbf{D}\mathbf{x}^{(t)}\right)x_j^{(t)}$$

We then isolate the variable $\mathbf{D}_{\cdot,j}$ by noting that the matrix product (and our estimate of the original vector $\mathbf{u}^{(t)}$) may be written as a sum over the columns: $\mathbf{D}\mathbf{x}^{(t)} = \sum_i \mathbf{D}_{\cdot,i}x_i^{(t)}$ where $x_i^{(t)}$ is the i-th component of the vector $\mathbf{x}^{(t)}$.

$$\begin{aligned} 0 &= \sum_{t=1}^{T}\left(\mathbf{u}^{(t)} - \left(\sum_{i\neq j}\mathbf{D}_{\cdot,i}x_i^{(t)}\right) - \mathbf{D}_{\cdot,j}x_j^{(t)}\right)x_j^{(t)} \\ \sum_{t=1}^{T}\mathbf{D}_{\cdot,j}(x_j^{(t)})^2 &= \sum_{t=1}^{T}\left(\mathbf{u}^{(t)} - \left(\sum_{i\neq j}\mathbf{D}_{\cdot,i}x_i^{(t)}\right)\right)x_j^{(t)} \qquad (5) \end{aligned}$$

Finally, we note that in the sum on the left hand side of (5), we can take $\mathbf{D}_{\cdot,j}$ outside to get

$$\mathbf{D}_{\cdot,j} = \frac{1}{\sum_{t=1}^{T}(x_j^{(t)})^2}\sum_{t=1}^{T}\left(\mathbf{u}^{(t)} - \left(\sum_{i\neq j}\mathbf{D}_{\cdot,i}x_i^{(t)}\right)\right)x_j^{(t)}. \qquad (6)$$

Repeating the calculation (6) for each column $j = 1, \dots, k$ gives us the updated dictionary. We then normalize the columns and use this updated dictionary and go back to the LASSO sparse coding problem, keeping D fixed to the new value to improve the estimate for $\mathbf{x}^{(t)}$. Then we update the dictionary again, and alternate between solving these two problems until some convergence criterion is satisfied.

2.3 Improving dictionary update efficiency

We can rewrite the calculation in (6) to significantly reduce the computational burden. Namely, we can rewrite (6) by distributing the sum in the numerator and the term $x_j^{(t)}$ at the end of the expression to obtain

$$\begin{aligned} \mathbf{D}_{\cdot,j} &= \frac{1}{\sum_{t=1}^{T}(x_j^{(t)})^2}\left[\left(\sum_{t=1}^{T}\mathbf{u}^{(t)}x_j^{(t)}\right) - \sum_{i\neq j}\mathbf{D}_{\cdot,i}\left(\sum_{t=1}^{T}x_i^{(t)}x_j^{(t)}\right)\right] \qquad (7) \\ &= \frac{1}{A_{j,j}}\left(\mathbf{B}_{\cdot,j} - \mathbf{D}\mathbf{A}_{\cdot,j} + \mathbf{D}_{\cdot,j}A_{j,j}\right) \qquad (8) \end{aligned}$$

where $\mathbf{A}$ and $\mathbf{B}$ are sums of outer products:

$$\begin{aligned} \mathbf{A} &= \sum_{t=1}^{T}\mathbf{x}^{(t)}(\mathbf{x}^{(t)})^T \\ \mathbf{B} &= \sum_{t=1}^{T}\mathbf{u}^{(t)}(\mathbf{x}^{(t)})^T. \end{aligned}$$

In (8) we have to add $\mathbf{D}_{\cdot,j}A_{j,j}$ because the sum being subtracted in (7) is over all columns but j.

The formulation (8) for updating column j of the dictionary $\mathbf{D}$ gives an improved efficiency because we only need to calculate the matrices $\mathbf{A}$ and $\mathbf{B}$ once, then we can store them in memory while iterating over the columns in $\mathbf{D}$. This saves a lot of computation compared to calculating every term of (7) in each iteration of the while loop in Algorithm 1 below.

2.4 Global algorithm

Before giving a final algorithm, we formalize the dictionary update step from Section 2.2 in terms of its input and output.

Algorithm 1 Dictionary Update

```
Input: Previous dictionary $\mathbf{D} \in \mathbb{R}^{m\times k}$. Matrices $\mathbf{A} = \sum_{t=1}^{T} \mathbf{x}^{(t)}(\mathbf{x}^{(t)})^T \in \mathbb{R}^{k\times k}$, $\mathbf{B} = \sum_{t=1}^{T} \mathbf{u}^{(t)}(\mathbf{x}^{(t)})^T \in \mathbb{R}^{m\times k}$.
1: while $\mathbf{D}$ has not converged do
2:     for $j = 1, \dots, k$ do
3:         let $\mathbf{D}_{\cdot,j} = \frac{1}{A_{j,j}}\left(\mathbf{B}_{\cdot,j} - \mathbf{D}\mathbf{A}_{\cdot,j} + \mathbf{D}_{\cdot,j}A_{j,j}\right)$
4:         let $\mathbf{D}_{\cdot,j} = \mathbf{D}_{\cdot,j}/\|\mathbf{D}_{\cdot,j}\|_2$
Return: Updated dictionary $\mathbf{D}$
```

Now we can formulate the global algorithm

Algorithm 2 Global Dictionary Learning

```
Input: Data vector $\mathbf{u}^{(t)} \in \mathbb{R}^m$ for $t = 1, \dots, T$. Initial dictionary $\mathbf{D}_0 \in \mathbb{R}^{m\times k}$.
1: let $\mathbf{D} = \mathbf{D}_0$
2: while $\mathbf{D}$ has not converged do
3:     for $t = 1, \dots, T$ do
4:         let $\mathbf{x}^{(t)} = \arg\min_{\mathbf{x}} \frac{1}{2}\left\|\mathbf{u}^{(t)} - \mathbf{D}\mathbf{x}\right\|_2^2 + \lambda\|\mathbf{x}\|_1$ (using LARS)
5:     let $\mathbf{A} = \sum_{t=1}^{T} \mathbf{x}^{(t)}(\mathbf{x}^{(t)})^T$
6:     let $\mathbf{B} = \sum_{t=1}^{T} \mathbf{u}^{(t)}(\mathbf{x}^{(t)})^T$
7:     Update $\mathbf{D}$ using Algorithm 1 with $\mathbf{D}, \mathbf{A}$ and $\mathbf{B}$ as inputs.
Return: Final dictionary $\mathbf{D}$ and sparse codes $\mathbf{x}^{(t)}$ for $t = 1, \dots, T$.
```

References

Efron, Bradley et al. (2004). ‘Least angle regression’. In: *Ann. Statist.* 32.2, pp. 407–499.

Mairal, Julien et al. (2009). ‘Online Dictionary Learning for Sparse Coding’. In: *Proceedings of the 26th Annual International Conference on Machine Learning.* ICML ’09. Montreal, Quebec, Canada: ACM, pp. 689–696.

Tibshirani, Robert (1996). ‘Regression Shrinkage and Selection via the Lasso’. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1, pp. 267–288.

Tillmann, A. M. (2015). ‘On the Computational Intractability of Exact and Approximate Dictionary Learning’. In: *IEEE Signal Processing Letters* 22.1, pp. 45–49.