Fast Convolutional Sparse Coding (FCSC)

Bailey Kong
Department of Computer Science
University of California, Irvine
bhkong@ics.uci.edu

Charless C. Fowlkes
Department of Computer Science
University of California, Irvine
fowlkes@ics.uci.edu

1 Introduction

Convolutional sparse coding remedies the shortcomings of the traditional (patch-based) sparse coding by modeling shift invariance directly. This yields more parsimonious coding schemes but increases complexity by coupling the coding problem for neighboring image patches where they overlap. Here we describe an optimization approach which exploits the separability of convolution across bands in the frequency domain in order to make dictionary learning efficient. Our presentation follows the notation in the paper of [Bristow et al. IEEE CVPR 2013] but includes additional implementation details and corrects some inaccuracies and typos. [1]

2 Convolutional coding in the frequency domain

The objective for convolutional sparse coding is

$$\begin{aligned} \arg\min_{\mathbf{d},\mathbf{z}} \frac{1}{2}\|\mathbf{x} - \sum_{k=1}^{K} \mathbf{d}_k * \mathbf{z}_k\|_2^2 + \beta \sum_{k=1}^{K} \|\mathbf{z}_k\|_1 \\ \text{s.t. } \|\mathbf{d}_k\|_2^2 \leq 1 \quad \forall k = 1,\ldots,K, \end{aligned} \tag{1}$$

where $\mathbf{d}_k \in \mathbb{R}^M$ is the k-th filter, $\mathbf{z}_k \in \mathbb{R}^D$ is the corresponding sparse feature map, and $\mathbf{x} \in \mathbb{R}^{D-M+1}$ is an image. (Note that $M \ll D$.) For simplicity, we discuss using 1-dimensional image signals, but this generalizes to 2-dimensional image signals as well. It is well known that convolution corresponds to element-wise multiplication in the frequency domain, so it is more efficient to rewrite this objective in the frequency domain. Unfortunately this cannot be done due to the ℓ_1 penalty term, which is not rotation invariant. One way to work around this problem is by introducing two auxiliary variables $\mathbf{s}$ and $\mathbf{t}$ so that we can optimize part of the problem in the frequency domain and another part in the spatial domain. The objective can be rewritten as

$$\begin{aligned} \arg\min_{\hat{\mathbf{d}},\mathbf{s},\hat{\mathbf{z}},\mathbf{t}} \frac{1}{2D}\|\hat{\mathbf{x}} - \sum_{k=1}^{K} \hat{\mathbf{d}}_k \odot \hat{\mathbf{z}}_k\|_2^2 + \beta \sum_{k=1}^{K} \|\mathbf{t}_k\|_1 \\ \text{s.t. } \|\mathbf{s}_k\|_2^2 \leq 1 \quad \forall k = 1,\ldots,K \\ \mathbf{s}_k = \mathbf{F}^{-1}\hat{\mathbf{d}}_k \quad \forall k = 1,\ldots,K \\ \mathbf{t}_k = \mathbf{F}^{-1}\hat{\mathbf{z}}_k \quad \forall k = 1,\ldots,K \\ \mathbf{s}_k = \mathbf{C}\mathbf{s}_k \quad \forall k = 1,\ldots,K, \end{aligned} \tag{2}$$

where $\hat{\mathbf{x}}, \hat{\mathbf{d}}_k, \hat{\mathbf{z}}_k \in \mathbb{C}^D$, $\mathbf{s}_k, \mathbf{t}_k \in \mathbb{R}^D$, and $\mathbf{F}^{-1}$ is the inverse discrete Fourier transform matrix. The original variables $\mathbf{d}$ and $\mathbf{z}$ are moved to the frequency domain, and are now denoted as $\hat{\mathbf{d}}$ and $\hat{\mathbf{z}}$, while the auxiliary variables $\mathbf{s}$ and $\mathbf{t}$ take their place in the spatial domain and constraints are added to couple them together. The careful observer may also noticed that although originally $\mathbf{d}_k$ was M dimensional, $\mathbf{s}_k$ is now D dimensional. The mapping matrix $\mathbf{C}$ zeros out any entries of $\mathbf{s}_k$ which are outside the desired spatial support so the final constraint assures the learned dictionary elements have small spatial extent.

Notation: $\mathbf{F}$ is the discrete Fourier transform matrix and $\mathbf{F}^{-1}$ is the inverse DFT matrix defined as $\mathbf{F}^{-1} = \frac{1}{D}\mathbf{F}^{\dagger}$. Since the transform matrix $\mathbf{F}$ is orthogonal, the squared ℓ_2-norm in the spatial and frequency domains are equivalent up to a scale factor (i.e., $D\|\mathbf{s}\|_2^2 = \|\hat{\mathbf{s}}\|_2^2$). The matrix $\mathbf{C}$ is the mapping matrix that truncates a vector $\mathbb{R}^D \to \mathbb{R}^M$ and pads the truncation with zeros back to $\mathbb{R}^D$. This is can be defined as the product of two matrices $\mathbf{C} = \mathbf{P}\mathbf{T}$, where $\mathbf{P}$ is our padding matrix (a $D \times M$ identity matrix) and $\mathbf{T}$ is our truncation matrix (a $M \times D$ identity matrix). For the vectors $\mathbf{d}$, $\mathbf{s}$, $\mathbf{z}$, and $\mathbf{t}$ where the subscript

[1]This work was supported by NSF DBI-1262547 and NSF DBI-1053036

is dropped, we refer to a super vector with K components stacked together (e.g. $\mathbf{s} = [\mathbf{s}_1^T, \dots, \mathbf{s}_K^T]^T$). The same applies for the vectors in the frequency domain, $\hat{\mathbf{s}} = [(\mathbf{F}\mathbf{s}_1)^\dagger, \dots, (\mathbf{F}\mathbf{s}_K)^\dagger]^\dagger$.

3 Optimization by ADMM

To solve this problem with variables in both the frequency and spatial domains, the Alternating Direction Method of Multipliers (ADMM) approach is used. We begin by writing the augmented Lagrangian that incorporates the equality constraints coupling the auxiliary variables $\mathbf{s}, \mathbf{t}$ to the original variables $\mathbf{d}, \mathbf{z}$.

$$\begin{aligned}
\mathcal{L}(\hat{\mathbf{d}}, \mathbf{s}, \hat{\mathbf{z}}, \mathbf{t}, \boldsymbol{\lambda}_\mathbf{s}, \boldsymbol{\lambda}_\mathbf{t}) = &\frac{1}{2D}\|\hat{\mathbf{x}} - \sum_{k=1}^K \hat{\mathbf{d}}_k \odot \hat{\mathbf{z}}_k\|_2^2 + \beta \sum_{k=1}^K \|\mathbf{t}_k\|_1 \\
&+ \sum_{k=1}^K \frac{\mu_\mathbf{s}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k\|_2^2 + \sum_{k=1}^K \boldsymbol{\lambda}_{\mathbf{s}k}^T(\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k) \\
&+ \sum_{k=1}^K \frac{\mu_\mathbf{t}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k\|_2^2 + \sum_{k=1}^K \boldsymbol{\lambda}_{\mathbf{t}k}^T(\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k) \\
\text{s.t. } &\|\mathbf{s}_k\|_2^2 \le 1 \quad \forall k = 1, \dots, K \\
&\mathbf{s}_k = \mathbf{C}\mathbf{s}_k \quad \forall k = 1, \dots, K,
\end{aligned} \tag{3}$$

where $\boldsymbol{\lambda}_{\mathbf{s}k}$, $\mu_\mathbf{s}$, $\boldsymbol{\lambda}_{\mathbf{t}k}$, and $\mu_\mathbf{t}$ are the Lagrange multipliers associated with $\mathbf{s}$ and $\mathbf{t}$ respectively.

Subproblem $\mathbf{z}$:

$$\begin{aligned}
\hat{\mathbf{z}}^* &= \arg\min_{\hat{\mathbf{z}}} \mathcal{L}(\hat{\mathbf{z}}; \hat{\mathbf{d}}, \mathbf{s}, \mathbf{t}, \boldsymbol{\lambda}_\mathbf{s}, \boldsymbol{\lambda}_\mathbf{t}) \\
&= \arg\min_{\hat{\mathbf{z}}} \frac{1}{2D}\|\hat{\mathbf{x}} - \sum_{k=1}^K \hat{\mathbf{d}}_k \odot \hat{\mathbf{z}}_k\|_2^2 + \sum_{k=1}^K \frac{\mu_\mathbf{t}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k\|_2^2 + \sum_{k=1}^K \boldsymbol{\lambda}_{\mathbf{t}k}^T(\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k) \\
&= \arg\min_{\hat{\mathbf{z}}} \frac{1}{2D}\|\hat{\mathbf{x}} - \hat{\mathbf{D}}\hat{\mathbf{z}}\|_2^2 + \frac{\mu_\mathbf{t}}{2D}\|\hat{\mathbf{z}} - \hat{\mathbf{t}}\|_2^2 + \frac{1}{D}\hat{\boldsymbol{\lambda}}_\mathbf{t}^\dagger(\hat{\mathbf{z}} - \hat{\mathbf{t}}) \\
&= (\hat{\mathbf{D}}^\dagger\hat{\mathbf{D}} + \mu_\mathbf{t}\mathbf{I})^{-1}(\hat{\mathbf{D}}^\dagger\hat{\mathbf{x}} + \mu_\mathbf{t}\hat{\mathbf{t}} - \hat{\boldsymbol{\lambda}}_\mathbf{t}),
\end{aligned} \tag{4}$$

where $\hat{\mathbf{D}} = [\text{diag}(\hat{\mathbf{d}}_1), \dots, \text{diag}(\hat{\mathbf{d}}_K)]$ is $D \times KD$. Although the matrix $\hat{\mathbf{D}}^\dagger\hat{\mathbf{D}}$ is quite large, $KD \times KD$, it is sparse banded and has an efficient variable reordering such that $\hat{\mathbf{z}}^*$ can be found by solving D independent $K \times K$ linear systems.

Subproblem $\mathbf{t}$:

$$\begin{aligned}
\mathbf{t}_k^* &= \arg\min_{\mathbf{t}_k} \mathcal{L}(\mathbf{t}_k; \hat{\mathbf{d}}_k, \mathbf{s}_k, \hat{\mathbf{z}}_k, \boldsymbol{\lambda}_{\mathbf{s}k}, \boldsymbol{\lambda}_{\mathbf{t}k}) \\
&= \arg\min_{\mathbf{t}_k} \beta\|\mathbf{t}_k\|_1 + \frac{\mu_\mathbf{t}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k\|_2^2 + \boldsymbol{\lambda}_{\mathbf{t}k}^T(\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k) \\
&= \arg\min_{\mathbf{t}_k} \beta\|\mathbf{t}_k\|_1 + \frac{\mu_\mathbf{t}}{2}\|\mathbf{z}_k - \mathbf{t}_k\|_2^2 + \boldsymbol{\lambda}_{\mathbf{t}k}^T(\mathbf{z}_k - \mathbf{t}_k)
\end{aligned} \tag{5}$$

Given $\mathbf{z}_k$ and $\boldsymbol{\lambda}_{\mathbf{t}k}$ in the spatial domain, it turns out that each coefficient in $\mathbf{t}_k$ is independent of all other coefficients,

$$t_{ki}^* = \arg\min_{t_{ki}} \frac{\mu_\mathbf{t}}{2}(z_{ki} - t_{ki})^2 + \lambda_{\mathbf{t}ki}(z_{ki} - t_{ki}) + \beta|t_{ki}|. \tag{6}$$

Which can be efficiently compute in closed form using the shrinkage (soft-thresholding) function,

$$t_{ki}^* = \text{sign}\left(z_{ki} + \frac{\lambda_{\mathbf{t}ki}}{\mu_\mathbf{t}}\right) \cdot \max\left\{\left|z_{ki} + \frac{\lambda_{\mathbf{t}ki}}{\mu_\mathbf{t}}\right| - \frac{\beta}{\mu_\mathbf{t}}, 0\right\}. \tag{7}$$

Subproblem $\mathbf{d}$:

$$
\begin{aligned}
\hat{\mathbf{d}}^* &= \arg\min_{\hat{\mathbf{d}}} \; \mathcal{L}(\hat{\mathbf{d}}; \mathbf{s}, \hat{\mathbf{z}}, \mathbf{t}, \boldsymbol{\lambda}_{\mathbf{s}}, \boldsymbol{\lambda}_{\mathbf{t}}) \\
&= \arg\min_{\hat{\mathbf{d}}} \; \frac{1}{2D}\|\hat{\mathbf{x}} - \sum_{k=1}^{K} \hat{\mathbf{d}}_k \odot \hat{\mathbf{z}}_k\|_2^2 + \sum_{k=1}^{K} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k\|_2^2 + \sum_{k=1}^{K} \boldsymbol{\lambda}_{\mathbf{s}k}^T(\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k) \\
&= \arg\min_{\hat{\mathbf{d}}} \; \frac{1}{2D}\|\hat{\mathbf{x}} - \hat{\mathbf{Z}}\hat{\mathbf{d}}\|_2^2 + \frac{\mu_{\mathbf{s}}}{2D}\|\hat{\mathbf{d}} - \hat{\mathbf{s}}\|_2^2 + \frac{1}{D}\hat{\boldsymbol{\lambda}}_{\mathbf{s}}^{\dagger}(\hat{\mathbf{d}} - \hat{\mathbf{s}}) \\
&= (\hat{\mathbf{Z}}^{\dagger}\hat{\mathbf{Z}} + \mu_{\mathbf{s}}\mathbf{I})^{-1}(\hat{\mathbf{Z}}^{\dagger}\hat{\mathbf{x}} + \mu_{\mathbf{s}}\hat{\mathbf{s}} - \hat{\boldsymbol{\lambda}}_{\mathbf{s}})
\end{aligned}
\tag{8}
$$

where $\hat{\mathbf{Z}} = [\text{diag}(\hat{\mathbf{z}}_1), \ldots, \text{diag}(\hat{\mathbf{z}}_K)]$ is $D \times KD$. Like subproblem $\mathbf{z}$, there is an efficient variable reordering such that $\hat{\mathbf{d}}^*$ can be found by solving D independent $K \times K$ linear systems.

Subproblem $\mathbf{s}$:

$$
\begin{aligned}
\mathbf{s}_k^* &= \arg\min_{\mathbf{s}_k} \; \mathcal{L}(\mathbf{s}_k; \hat{\mathbf{d}}_k, \hat{\mathbf{z}}_k, \mathbf{t}_k, \boldsymbol{\lambda}_{\mathbf{s}k}, \boldsymbol{\lambda}_{\mathbf{t}k}) \\
&= \arg\min_{\mathbf{s}_k} \; \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k\|_2^2 + \boldsymbol{\lambda}_{\mathbf{s}k}^T(\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k) \\
&= \arg\min_{\mathbf{s}_k} \; \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{d}_k - \mathbf{s}_k\|_2^2 + \boldsymbol{\lambda}_{\mathbf{s}k}^T(\mathbf{d}_k - \mathbf{s}_k) \\
&\text{s.t. } \|\mathbf{s}_k\|_2^2 \leq 1 \\
&\quad\; \mathbf{s}_k = \mathbf{C}\mathbf{s}_k
\end{aligned}
\tag{9}
$$

An equivalent formulation is to drop the spatial support constraint on $\mathbf{s}$ and instead preserve only the M coefficients of $\mathbf{d}_k$ and $\boldsymbol{\lambda}_{\mathbf{s}k}$ associated with the small spatial support.

$$
\begin{aligned}
\mathbf{s}_k^* &= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{C}\mathbf{d}_k - \mathbf{s}_k\|_2^2 + (\mathbf{C}\boldsymbol{\lambda}_{\mathbf{s}k})^T(\mathbf{C}\mathbf{d}_k - \mathbf{s}_k) \\
&\text{s.t. } \|\mathbf{s}_k\|_2^2 \leq 1.
\end{aligned}
\tag{10}
$$

The coefficients of $\mathbf{s}_k$ outside the spatial support (where $\mathbf{C}\mathbf{d}_k$ is 0) are automatically driven to 0 by the ℓ_2 norm (see appendix for details).

We can solve the unconstrained version of this problem by pulling the linear term into the quadratic, completing the square

$$
\begin{aligned}
\tilde{\mathbf{s}}_k^* &= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{C}\mathbf{d}_k - \mathbf{s}_k\|_2^2 + (\mathbf{C}\boldsymbol{\lambda}_{\mathbf{s}k})^T(\mathbf{C}\mathbf{d}_k - \mathbf{s}_k) \\
&= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\bar{\mathbf{d}}_k - \mathbf{s}_k\|_2^2 + \bar{\boldsymbol{\lambda}}_{\mathbf{s}k}^T(\bar{\mathbf{d}}_k - \mathbf{s}_k) \qquad \text{where } \bar{\mathbf{d}}_k = \mathbf{C}\mathbf{d}_k, \bar{\boldsymbol{\lambda}}_{\mathbf{s}k} = \mathbf{C}\boldsymbol{\lambda}_{\mathbf{s}k} \\
&= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{s}_k - \big(\bar{\mathbf{d}}_k + \frac{1}{\mu_{\mathbf{s}}}\bar{\boldsymbol{\lambda}}_{\mathbf{s}k}\big)\|_2^2 - \frac{\mu_{\mathbf{s}}}{2}\|\bar{\mathbf{d}}_k + \frac{1}{\mu_{\mathbf{s}}}\bar{\boldsymbol{\lambda}}_{\mathbf{s}k}\|_2^2 + \frac{\mu_{\mathbf{s}}}{2}\bar{\mathbf{d}}_k^T\bar{\mathbf{d}}_k + \bar{\boldsymbol{\lambda}}_{\mathbf{s}k}^T\bar{\mathbf{d}}_k \\
&= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{s}_k - \big(\bar{\mathbf{d}}_k + \frac{1}{\mu_{\mathbf{s}}}\bar{\boldsymbol{\lambda}}_{\mathbf{s}k}\big)\|_2^2
\end{aligned}
$$

The optimal solution to the unconstrained problem (denoted by $\tilde{\mathbf{s}}_k$) is thus given in closed form by

$$
\tilde{\mathbf{s}}_k = \mathbf{C}\mathbf{d}_k + \frac{1}{\mu_{\mathbf{s}}}\mathbf{C}\boldsymbol{\lambda}_{\mathbf{s}k},
\tag{11}
$$

Since this objective is isotropic, we can simply enforce the norm constraint by projecting the unconstrained solution back on to the constraint set:

$$
\mathbf{s}_k^* = \begin{cases} \|\tilde{\mathbf{s}}_k\|_2^{-1}\tilde{\mathbf{s}}_k & \text{if } \|\tilde{\mathbf{s}}_k\|_2^2 \geq 1 \\ \tilde{\mathbf{s}}_k & \text{otherwise} \end{cases}.
\tag{12}
$$

Lagrange Multiplier Update:

$$
\boldsymbol{\lambda}_{\mathbf{s}}^{(i+1)} = \boldsymbol{\lambda}_{\mathbf{s}}^{(i)} + \mu_{\mathbf{s}}(\mathbf{d}^{(i+1)} - \mathbf{s}^{(i+1)})
\tag{13}
$$

$$
\boldsymbol{\lambda}_{\mathbf{t}}^{(i+1)} = \boldsymbol{\lambda}_{\mathbf{t}}^{(i)} + \mu_{\mathbf{t}}(\mathbf{z}^{(i+1)} - \mathbf{t}^{(i+1)})
\tag{14}
$$

Penalty Update:

$$\mu^{(i+1)} = \begin{cases} \tau\mu^{(i)} & \text{if } \mu^{(i)} < \mu_{\max} \\ \mu^{(i)} & \text{otherwise} \end{cases} \tag{15}$$

Algorithm 1 Convolutional Sparse Coding using Fourier Subproblems and ADMM

```
1:  Initialize $\mathbf{s}^{(0)}, \mathbf{z}^{(0)}, \mathbf{t}^{(0)}, \boldsymbol{\lambda}_{\mathbf{s}}^{(0)}, \boldsymbol{\lambda}_{\mathbf{t}}^{(0)}, \mu_{\mathbf{s}}^{(0)}, \mu_{\mathbf{t}}^{(0)}$
2:  Perform FFT $\mathbf{s}^{(0)}, \mathbf{z}^{(0)}, \mathbf{t}^{(0)}, \boldsymbol{\lambda}_{\mathbf{s}}^{(0)}, \boldsymbol{\lambda}_{\mathbf{t}}^{(0)} \to \hat{\mathbf{s}}^{(0)}, \hat{\mathbf{z}}^{(0)}, \hat{\mathbf{t}}^{(0)}, \hat{\boldsymbol{\lambda}}_{\mathbf{s}}^{(0)}, \hat{\boldsymbol{\lambda}}_{\mathbf{t}}^{(0)}$
3:  $i = 0$
4:  **repeat**
5:      **dictionary update:**
6:      Solve for $\hat{\mathbf{d}}^{(i+1)}$ given $\hat{\mathbf{s}}^{(i)}, \hat{\mathbf{z}}^{(i)}, \hat{\boldsymbol{\lambda}}_{\mathbf{s}}^{(i)}$ using Eq. 8
7:      Perform inverse FFT $\hat{\mathbf{d}}^{(i+1)} \to \mathbf{d}^{(i+1)}$
8:      Solve for $\tilde{\mathbf{s}}^{(i+1)}$ given $\mathbf{d}^{(i+1)}$ using Eq. 11
9:      Solve for $\mathbf{s}^{(i+1)}$ by projecting $\tilde{\mathbf{s}}^{(i+1)}$ using Eq. 12
10:     **code update:**
11:     Solve for $\hat{\mathbf{z}}^{(i+1)}$ given $\hat{\mathbf{t}}^{(i)}, \hat{\boldsymbol{\lambda}}_{\mathbf{t}}^{(i)}, \hat{\mathbf{d}}^{(i+1)}$ using Eq. 4
12:     Perform inverse FFT $\hat{\mathbf{z}}^{(i+1)} \to \mathbf{z}^{(i+1)}$
13:     Solve for $\mathbf{t}^{(i+1)}$ given $\mathbf{z}^{(i+1)}, \boldsymbol{\lambda}_{\mathbf{t}}^{(i)}$ using Eq. 5
14:     Update Lagrange multiplier vectors using Eq. 13 and Eq. 14
15:     Update penalties using Eq. 15
16:     Perform FFT $\mathbf{s}, \mathbf{t}, \boldsymbol{\lambda}_{\mathbf{s}}, \boldsymbol{\lambda}_{\mathbf{t}} \to \hat{\mathbf{s}}, \hat{\mathbf{t}}, \hat{\boldsymbol{\lambda}}_{\mathbf{s}}, \hat{\boldsymbol{\lambda}}_{\mathbf{t}}$
17:     $i = i + 1$
18: **until** $\hat{\mathbf{d}}, \mathbf{s}, \hat{\mathbf{z}}, \mathbf{t}$ has converged
```

4 Effect of ADMM parameters on optimization

This section aims to provide some intuition regarding the hyper-parameters of FCSC optimization, specifically we want to understand how the hyper-parameters interact with one another in terms of the convergence rate and the objective value. In all our experiments, we use images that are contrast normalized. The convergence criteria is that we've reach a feasible solution (i.e., $\mathbf{s} = \mathbf{F}^{-1}\hat{\mathbf{d}}$ and $\mathbf{t} = \mathbf{F}^{-1}\hat{\mathbf{z}}$).

The FCSC algorithm has three hyper-parameters, $\mu_{\mathbf{s}}$, $\mu_{\mathbf{t}}$, and β. $\mu_{\mathbf{s}}$ can be thought of as the parameter that control the convergence rate of the dictionary learning, as it is the step-size at which we move towards a feasible solution. Likewise, $\mu_{\mathbf{t}}$ is the same for the coding. The hyper-parameter β in our original objective controls the sparsity of the codes found.

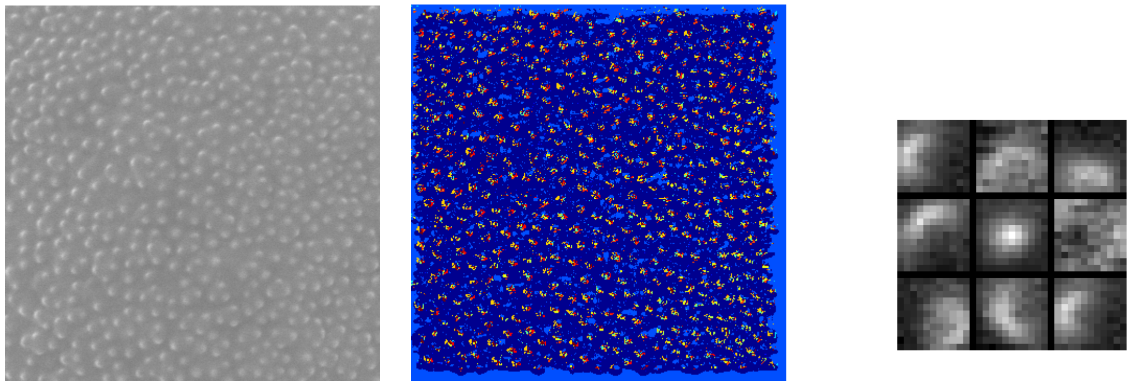
(a) (b) (c)

Figure 1: (a) Example *A. odoratum* grass pollen grain image (SEM, 6000x). (b) Visualization of sparse coding of pollen texture. Pseudo-color indicates dictionary element with the largest magnitude coefficient (c) Corresponding dictionary learned from images of pollen texture using convolutional sparse coding with $\beta = 1$

4.1 Effect of step-size parameters on coding

First we look at the coding hyper-parameters; using a pre-trained dictionary, we vary β and $\mu_{\mathbf{t}}$. Since β controls the sparsity of the codes, we expect (and the results show) that as it increases the reconstruction error would also increase. Perhaps not surprisingly, the same behavior is observed w.r.t. the objective value. As β increases the codes become more sparse to the point where they're all zeros. At that point the objective value is just the reconstruction error with a vector of zeros. We see that both the reconstruction error and the objective value at convergence increase slowly as $\mu_{\mathbf{t}}$ increases. In another experiments (not shown) we have seen the objective value be roughly the same for all values of $\mu_{\mathbf{t}}$ and $\beta = 0$, when a minimum number of iterations is enforced (in those experiments it was 10). These iterations are necessary to overcome the "memory" of the initial value of $\mathbf{t}$ in the ADMM iteration.

In terms of the convergence rate, for all values of β we generally see that the number of iterations required before convergence decreases as $\mu_{\mathbf{t}}$ increases. This is also to be expected as we previously described $\mu_{\mathbf{t}}$ as the step-size towards a feasible solution. Ignoring $\beta = 0$, we generally observe that the number of iterations goes down as β increases for any particular value of $\mu_{\mathbf{t}}$. This is a bit surprising, but can possibly be explained by considering how sparsity is induced. We know that the shrinkage function is used to solve the $\mathbf{t}$-subproblem, and that at each iteration the codes are shrunk by $\frac{\beta}{\mu_{\mathbf{t}}}$ towards zero. If β is large enough, then $\mu_{\mathbf{t}}$ does not grow enough at the early iterations such that additional coefficients of $\mathbf{z}$ are not zeroed out. This in turn magnifies the coefficients of $\boldsymbol{\lambda}_{\mathbf{t}}$ to make $\mathbf{z}$ more like $\mathbf{t}$, thus leading to faster convergence.

4.2 Effect of step-size and sparsity parameters on dictionary learning

In the second experiment, we fix the step-size $\mu_{\mathbf{s}}$ and simultaneously learn the dictionary and the codes. Here the results are quite different from before. We see that $\mu_{\mathbf{t}}$ must be strictly greater than β for FCSC to learn anything meaningful. When $\beta = 1$ both the reconstruction error and the objective value are largely unchanged w.r.t. $\mu_{\mathbf{t}}$. Finally in terms of convergence, we continue to observe that as $\mu_{\mathbf{t}}$ increases the number of iterations to convergence decreases for all values of β, consistent with our first experiment.

4.3 Effect of step-size on runtime and accuracy

In the third experiment, we fix the β hyper-parameter equal to 1 and look at how $\mu_{\mathbf{s}}$ and $\mu_{\mathbf{t}}$ interact. With β fixed, we see that the objective values increase as $\mu_{\mathbf{t}}$ and $\mu_{\mathbf{s}}$ increase in the domain $\mu_{\mathbf{s}} > 2000$ and $\mu_{\mathbf{t}} > 5$. This seems to be a result of the optimization enforcing constraints "too quickly". For small values of $\mu_{\mathbf{t}}$ and $\mu_{\mathbf{s}}$ (not shown) we also see erratic behavior. If we ignore the region where $\mu_{\mathbf{t}}$ is less than 5, we see that both the objective and the reconstruction error are smooth w.r.t. the hyper-parameters. This is quite nice, as we can choose a computational trade-off between lowering the objective value and the number of iterations to convergence.

We may also ask whether the optimization produces qualitatively "good" filters. A good learned filter shouldn't resemble the initialization and in general we expect to see Gabor-like oriented edge filters. After examining some learned filters that were subjectively good/bad, we defined goodness as having a standard deviation greater than 0.7. Much higher than the flat initialization which has a standard deviation of 0.01 and other bad looking filters that had standard deviation less than 0.6 in the cases we examined. While not a perfect measure by any means, we can see in the next set of figures that the reconstruction error is indeed correlated with the goodness measure.

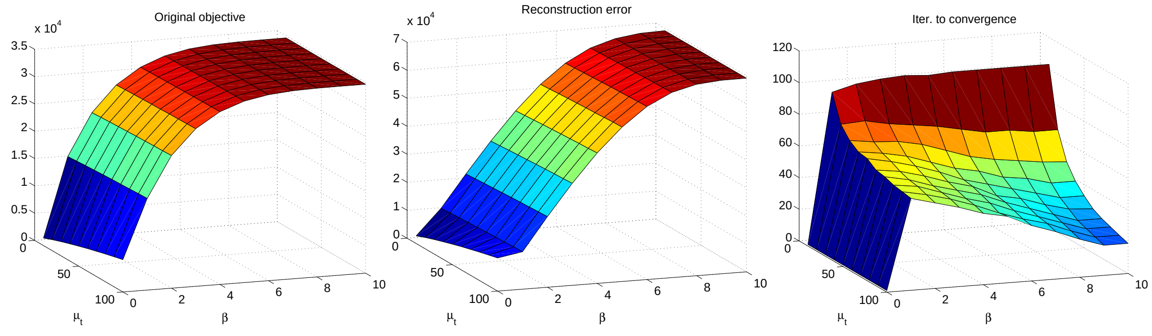

Figure 2: Effect of $\mu_{\mathbf{t}}, \beta$ on coding with fixed dictionary.

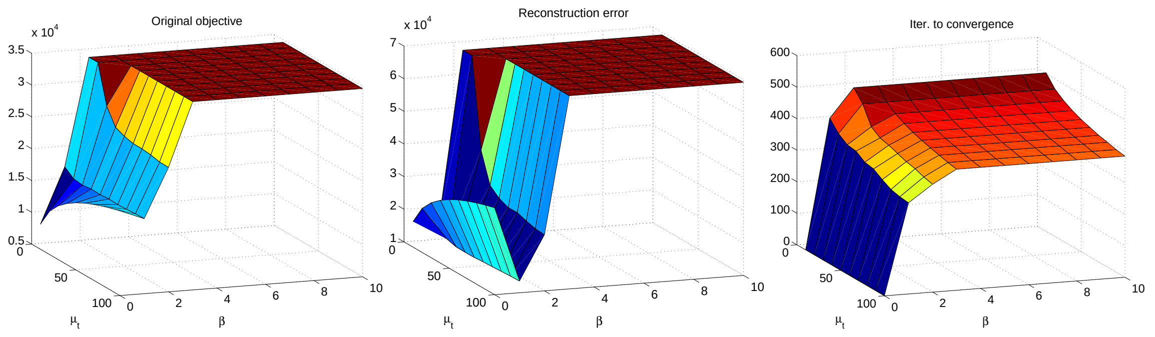

Figure 3: Effect of $\mu_\mathbf{t}, \beta$ on joint coding and dictionary learning ($\mu_\mathbf{s} = 515$).

Appendix A - Least-squares filters with constrained spatial support

To see why Eq. 10 is equivalent to the previous argmin (Eq. 9), consider the following problem

$$\begin{aligned} \mathbf{s}^* &= \arg\min_{\mathbf{s}} \|\mathbf{d} - \mathbf{s}\|_2^2 + \boldsymbol{\lambda}^T(\mathbf{d} - \mathbf{s}) \\ &\quad \text{s.t. } \mathbf{s} = \mathbf{C}\mathbf{s} \\ \mathbf{s}^* = \begin{bmatrix} \mathbf{s}_1^* \\ \mathbf{s}_0^* \end{bmatrix} &= \arg\min_{\mathbf{s}_1, \mathbf{s}_0} \|\mathbf{d}_1 - \mathbf{s}_1\|_2^2 + \|\mathbf{d}_0 - \mathbf{s}_0\|_2^2 + \boldsymbol{\lambda}_1^T(\mathbf{d}_1 - \mathbf{s}_1) + \boldsymbol{\lambda}_0^T(\mathbf{d}_0 - \mathbf{s}_0) \\ &\quad \text{s.t. } \mathbf{s}_0 = \mathbf{0}, \end{aligned}$$

where we split each vector into two sub-vectors: the part that corresponds to small spatial support (denoted with $_1$) and the part that does not (denoted with $_0$). By splitting s into $\mathbf{s}_1$ and $\mathbf{s}_0$, we can think about each as separate problems as there are no terms that contain both.

$$\begin{aligned} \mathbf{s}_1^* &= \arg\min_{\mathbf{s}_1} \|\mathbf{d}_1 - \mathbf{s}_1\|_2^2 + \boldsymbol{\lambda}_1^T(\mathbf{d}_1 - \mathbf{s}_1) \\ &= \arg\min_{\mathbf{s}_1} \|\mathbf{T}\mathbf{d} - \mathbf{s}_1\|_2^2 + \mathbf{T}\boldsymbol{\lambda}^T(\mathbf{T}\mathbf{d} - \mathbf{s}_1) \end{aligned}$$

$$\begin{aligned} \mathbf{s}_0^* &= \arg\min_{\mathbf{s}_0} \|\mathbf{d}_0 - \mathbf{s}_0\|_2^2 + \boldsymbol{\lambda}_0^T(\mathbf{d}_0 - \mathbf{s}_0) \\ &\quad \text{s.t. } \mathbf{s}_0 = \mathbf{0} \\ &= \arg\min_{\mathbf{s}_0} \|\mathbf{s}_0\|_2^2 \end{aligned}$$

So the $\mathbf{s}_1$ problem amounts to an unconstrained linear least squares; the $\mathbf{s}_0$ problem can be transformed into an equivalent unconstrained problem with a trivial solution when certain conditions hold. These conditions are that $\mathbf{d}_0$ and $\boldsymbol{\lambda}_0$ are null vectors. The matrix $\mathbf{C}$ does exactly this, as it preserves the coefficients of any vector corresponding to the small spatial support and zeros out the rest.

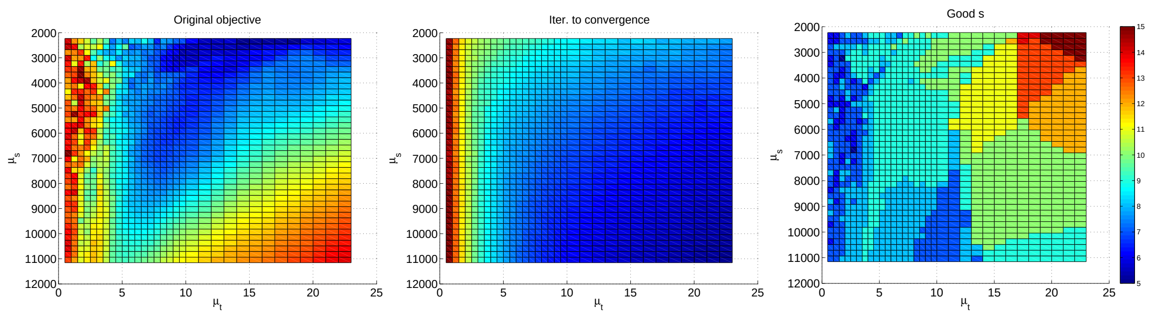

Figure 4: Joint effect of $\mu_\mathbf{t}, \mu_\mathbf{s}$ on solution quality and run time ($\beta = 1$)

Appendix B - Implementation details

During our discussion of subproblems $\mathbf{z}$ and $\mathbf{d}$, we mentioned an efficient variable reordering to find the optimal solution by solving D independent $K \times K$ linear systems. First we define the following matrices

$$\hat{\mathbf{Z}} = [\hat{\mathbf{z}}_1, \dots, \hat{\mathbf{z}}_K]^T, \hat{\mathbf{T}} = \left[\hat{\mathbf{t}}_1, \dots, \hat{\mathbf{t}}_K\right]^T, \hat{\mathbf{D}} = \left[\hat{\mathbf{d}}_1, \dots, \hat{\mathbf{d}}_K\right]^T, \hat{\mathbf{S}} = [\hat{\mathbf{s}}_1, \dots, \hat{\mathbf{s}}_K]^T$$
$$\hat{\mathbf{\Lambda}}_{\mathbf{t}} = \left[\hat{\boldsymbol{\lambda}}_{\mathbf{t}1}, \dots, \hat{\boldsymbol{\lambda}}_{\mathbf{t}K}\right]^T, \hat{\mathbf{\Lambda}}_{\mathbf{s}} = \left[\hat{\boldsymbol{\lambda}}_{\mathbf{s}1}, \dots, \hat{\boldsymbol{\lambda}}_{\mathbf{s}K}\right]^T$$
$$\hat{\mathbf{Dx}} = \left[\overline{\hat{\mathbf{d}}_1} \odot \hat{\mathbf{x}}, \dots, \overline{\hat{\mathbf{d}}_K} \odot \hat{\mathbf{x}}\right]^T, \hat{\mathbf{Zx}} = \left[\overline{\hat{\mathbf{z}}_1} \odot \hat{\mathbf{x}}, \dots, \overline{\hat{\mathbf{z}}_K} \odot \hat{\mathbf{x}}\right]^T$$

and denote $\hat{\mathbf{Z}}_i, \hat{\mathbf{T}}_i, \hat{\mathbf{D}}_i, \hat{\mathbf{S}}_i, \hat{\mathbf{\Lambda}}_{\mathbf{t}i}, \hat{\mathbf{\Lambda}}_{\mathbf{s}i}, \hat{\mathbf{Dx}}_i$, and $\hat{\mathbf{Zx}}_i$ to be the i-th column of their respective matrices.[2] $\overline{\hat{\mathbf{d}}}$ and $\overline{\hat{\mathbf{z}}}$ denote the conjugate of $\hat{\mathbf{d}}$ and $\hat{\mathbf{z}}$ respectively. We can then solve the subproblem $\mathbf{z}$ with

$$\hat{\mathbf{Z}}_i^* = (\hat{\mathbf{D}}_i \hat{\mathbf{D}}_i^\dagger + \mu_{\mathbf{t}} \mathbf{I})^{-1} (\hat{\mathbf{Dx}}_i + \mu_{\mathbf{t}} \hat{\mathbf{T}}_i + \hat{\mathbf{\Lambda}}_{\mathbf{t}i}),$$

and subproblem $\mathbf{d}$ with

$$\hat{\mathbf{D}}_i^* = (\hat{\mathbf{Z}}_i \hat{\mathbf{Z}}_i^\dagger + \mu_{\mathbf{s}} \mathbf{I})^{-1} (\hat{\mathbf{Zx}}_i + \mu_{\mathbf{s}} \hat{\mathbf{S}}_i + \hat{\mathbf{\Lambda}}_{\mathbf{s}i}).$$

In Eq. 8, we solved for $\hat{\mathbf{d}}_k$ which the inverse Fourier transform needs to be applied to get $\mathbf{d}_k$. However, since only a sub-vector of $\mathbf{d}_k$ is needed computing the inverse of the entire vector is wasteful. If we look at the definition of $\mathbf{C}$, we see that we can define a new matrix $\mathbf{\Phi} = \mathbf{T}\mathbf{F}^{-1}$ that will invert just the part of the vector we need. And the smaller M is with respect to D, the greater the savings we gain. We can redefine Eq. 11 and Eq. 12 as

$$\tilde{\mathbf{s}}_k = \mathbf{\Phi} \hat{\mathbf{d}}_k + \frac{1}{\mu_{\mathbf{s}}} \mathbf{\Phi} \hat{\boldsymbol{\lambda}}_{\mathbf{s}k}. \tag{16}$$

$$\mathbf{s}_k^* = \mathbf{P} \begin{cases} \|\tilde{\mathbf{s}}_k\|_2^{-1} \tilde{\mathbf{s}}_k & \text{if } \|\tilde{\mathbf{s}}_k\|_2^2 \geq 1 \\ \tilde{\mathbf{s}}_k & \text{otherwise} \end{cases} \tag{17}$$

Our entire discussion thus far has revolved around a single image, to handle multiple images is quite straightforward and only requires us to change Eq. 8,

$$\hat{\mathbf{d}}^* = (\sum_{x=1}^{N} \hat{\mathbf{Z}}_x^\dagger \hat{\mathbf{Z}}_x + \mu_{\mathbf{s}} \mathbf{I})^{-1} (\sum_{x=1}^{N} \hat{\mathbf{Z}}_x^\dagger \hat{\mathbf{x}}_x + \mu_{\mathbf{s}} \hat{\mathbf{s}} - \hat{\boldsymbol{\lambda}}_{\mathbf{s}}). \tag{18}$$

[2]Note that $\hat{\mathbf{T}}$ is unrelated to th truncation matrix $\mathbf{T}$