The K-SVD Algorithm

1 Introduction

K-SVD is an iterative algorithm used for dictionary learning - it finds a set of basis vectors (atoms of the dictionary) that can represent image patches sparsely. It consists of two alternating steps:

1. **Sparse Coding** - Finding sparse representations of patches using the current dictionary
2. **Dictionary Update** - Updating each dictionary atom using SVD

The process repeats for multiple iterations to refine both the dictionary and sparse representations.

2 Step 1: Extracting Image Patches

We start with an image I of size $H \times W$, from which we extract small overlapping patches.

1. Choose a patch size $P \times P$ (ex. 8×8).
2. Each patch is flattened into a column vector of size $P^2 \times 1$ (ex. 64×1).
3. Stack all patches to form the **data matrix**:

   $$Y = [y_1, y_2, ..., y_m] \in \mathbb{R}^{P^2 \times m} \tag{1}$$

   where:

   - P^2 is the vectorized patch size.
   - m is the total number of patches.

Before dictionary learning begins, the dictionary D is initialized, which consists of n atoms:

$$D = [d_1, d_2, ..., d_n] \in \mathbb{R}^{P^2 \times n} \tag{2}$$

Common Initialization Methods:

- **Random Gaussian vectors** (normalized to unit norm)

- **Random Image Patches** - Using randomized patches from the image itself

Each column d_k is normalized to unit ℓ_2-norm:

$$d_k = \frac{d_k}{\|d_k\|_2} \tag{3}$$

3 Step 2: Sparse Coding (Solving for X)

In this step, we represent each patch y_i as a sparse linear combination of the dictionary atoms:

$$y_i \approx Dx_i \tag{4}$$

where:

- $x_i \in \mathbb{R}^n$ is the sparse coefficient vector for y_i.
- Most elements of x_i are zero (sparse representation).

To find $X = [x_1, x_2, ..., x_m]$, we solve:

$$\min_{x_i} \|y_i - Dx_i\|_2^2 \quad \text{s.t.} \quad \|x_i\|_0 \leq T \tag{5}$$

where:

- T is the sparsity level (max number of non-zero coefficients per patch).
- This is solved **independently for each** y_i.

Each x_i is solved for using an algorithm called **Orthogonal Matching Pursuit (OMP)**. Roughly, we find which d_j correlates the best with the given patch (y_i) by calculating $\max_{d_j} \langle d_j, y_i \rangle$, and similarly find a second d_j to make up the difference, and so on, until we either hit T, or the difference becomes negligible.

After solving for all patches, we obtain:

$$X = [x_1, x_2, ..., x_m] \in \mathbb{R}^{n \times m} \tag{6}$$

where each column is a sparse vector.

4 Step 3: Dictionary Update Using SVD

Once we have X, we update D atom by atom.

For each dictionary atom d_k:

1. Compute the residual matrix by removing the contribution of all other dictionary atoms:

$$E_k = Y - \sum_{j \neq k} d_j x_j, \tag{7}$$

where x_j is the j^{th} row of X

2. **Filtering down E_k - Select patches where d_k is used**: From E_k, we select only those patches where d_k originally had a contribution.

   That is, if $x_k j = 0$, then the j^{th} column from E_k is discarded.

   This new matrix, $E_k^{\omega_k}$, is the filtered residual matrix.

3. **Perform SVD on $E_k^{\omega_k}$**:
$$E_k^{\omega_k} = U\Sigma V^T \tag{8}$$

4. **Update dictionary atom d_k**:
$$d_k = u_1 \tag{9}$$

5. **Update the corresponding sparse coefficients**:
$$X_k^{\omega_k} = \sigma_1 v_1^T \tag{10}$$

This process is repeated for each dictionary atom d_k.

5 Step 4: Repeat Until Convergence

- Sparse Coding and Dictionary Update are repeated for multiple iterations (typically 10-50).
- The process stops when:
  - The dictionary stabilizes (small change in D).
  - The reconstruction error stops decreasing.

6 Step 5: Image Reconstruction

After dictionary learning:

1. **Reconstruct patches** using D and X.
2. **Overlap and average** patches to form the final denoised/reconstructed image.