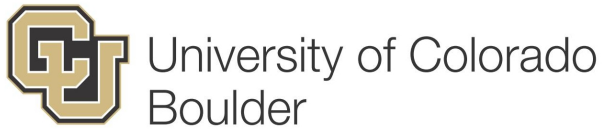
ECEN 4532: Digital Signal Processing Lab

Lecture Notes: Lab 4

Instructor: Prof. Farhad Pourkamali-Anaraki

University of Colorado at Boulder

Spring 2017



Introduction

- In this lab, you will implement the JPEG algorithm for image compression.
- In this lab, we will learn about the sparse representation model.
- Applications in a wide variety of domains, including but not limited to:
  - Multimedia technology
  - Medical imaging
  - Satellite imagery

Images

- A grayscale image M pixels tall and N pixels wide is represented as a matrix of size MxN

```
RGB = imread('autumn.tif');

size(RGB): 206   345     3
```

```
I = rgb2gray(RGB);

size(I):   206   345
```

Double precision

```
RGB = imread('autumn.tif');

I = rgb2gray(RGB);
```

Name ▲	Value
I	*206x345 uint8*
RGB	*206x345x3 uint8*

values range from 0 to $2^8 - 1 = 255$

```
J=im2double(I);
```

2D Filtering

```
img = imread('cameraman.tif');
imgd = im2double(img);
imgd = imnoise(imgd,'salt  & pepper',0.02);
f = ones(3,3)/9;
img1=filter2(f,imgd);
subplot(121);imshow(imgd);
subplot(122);imshow(img1);
```

Part 1: JPEG Compression

Structure of a compression system

- The **channel** is that part of the communication system that is out of the designer's control. Examples of channels are: wireless (radio), Ethernet, fiber, computer memory, etc.
- In this lab, we assume that the channel provides a reliable communication.

Encoder

- The encoder can be decomposed in a series of elementary stages:

1. Pre-processing. The goal of pre-processing is to **prepare** the image in order to minimize the coding complexity, and increase the compression efficiency. Examples of pre-possessing operations are:
   - Filtering
   - Padding with zeros
2. Transform. The goal of the transform is to reduce the correlation among image pixels, and describe with a smaller set of significant coefficients, most of the image content.

Encoder

3. Quantization. The output of the transform are real-valued coefficients. In order to decrease the amount of information needed to describe these coefficients, one represents the coefficients with a much smaller set of values.

Quantization is a non reversible process.

4. Entropy coding. After quantization, the image is approximated with a small set of codewords. Entropy coding permits to describe in the most efficient way the sequences of symbols from this alphabet with strings of ‘’0’’, and ‘’1’’.

Decoder

- The decoder reconstructs the image by applying in a reverse order the inverse of each of the processes 4), 3), and 2) described in the encoder.

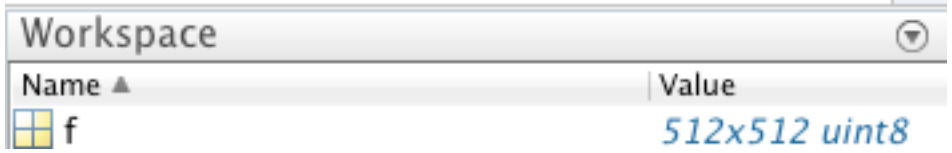
- While entropy coding is a reversible process, quantization is not invertible.

- A final **post-processing** is often performed on the reconstructed image in order to conceal coding artifacts, and improve the visual appearance.

Reading, writing, and displaying images

- In this lab, we will be using 6 standard test images available on D2L.

```
>>  f = imread ('barbara.pgm');
figure,imagesc(f, [0 255]),colormap gray,axis square, axis off;
g = f;
imwrite(g,'barbara2.pgm','pgm')
```



→ 8-bit unsigned integer

Values range from 0 to $2^8 - 1$.

(8 bits per pixel)

The JPEG compression algorithm

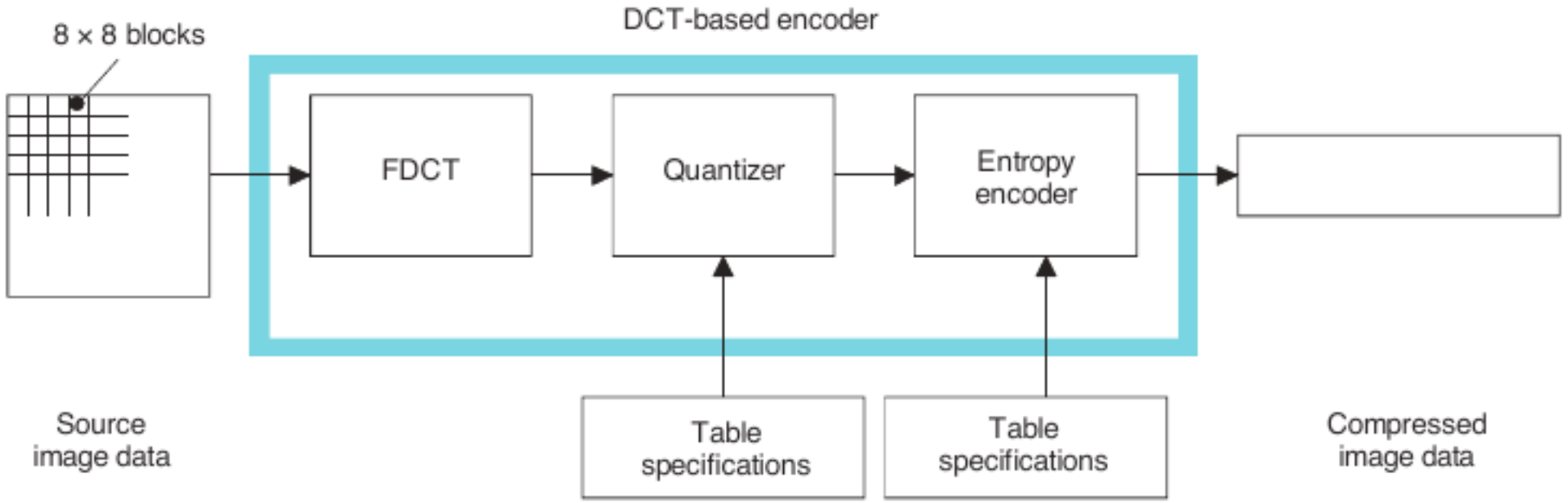


Figure 1: The different stages in the JPEG compression algorithm

The JPEG compression algorithm

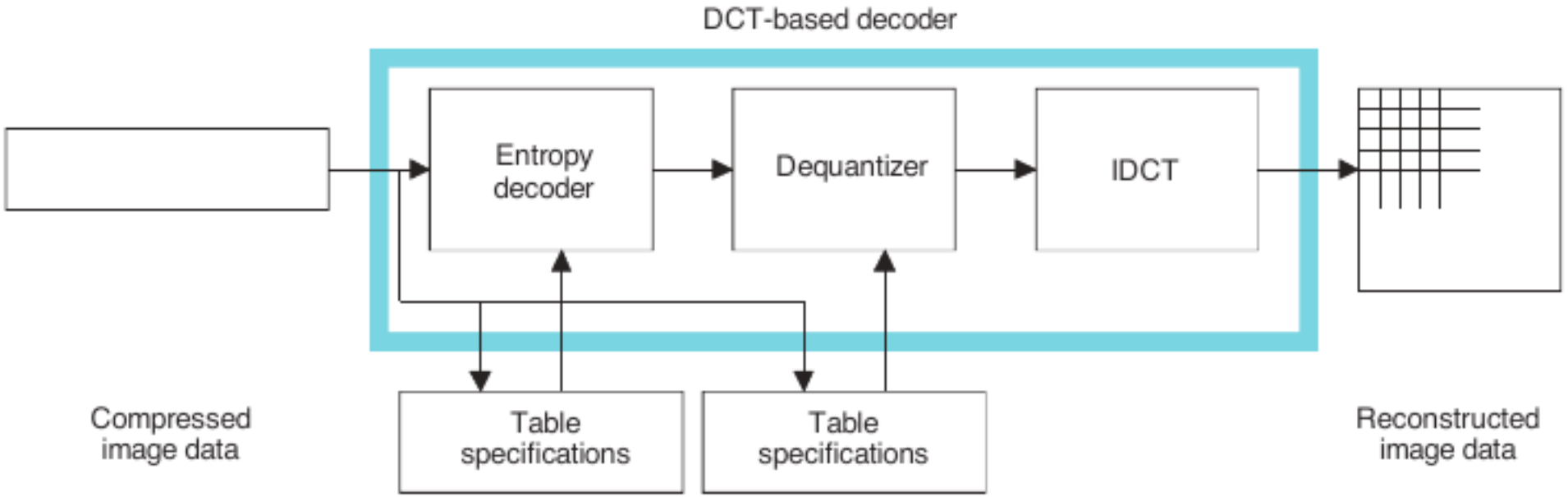


Figure 2: The different stages in the JPEG decoding algorithm

The forward and inverse discrete cosine transform

$$F_{u,v} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}, \quad u, v = 0, \ldots 7$$

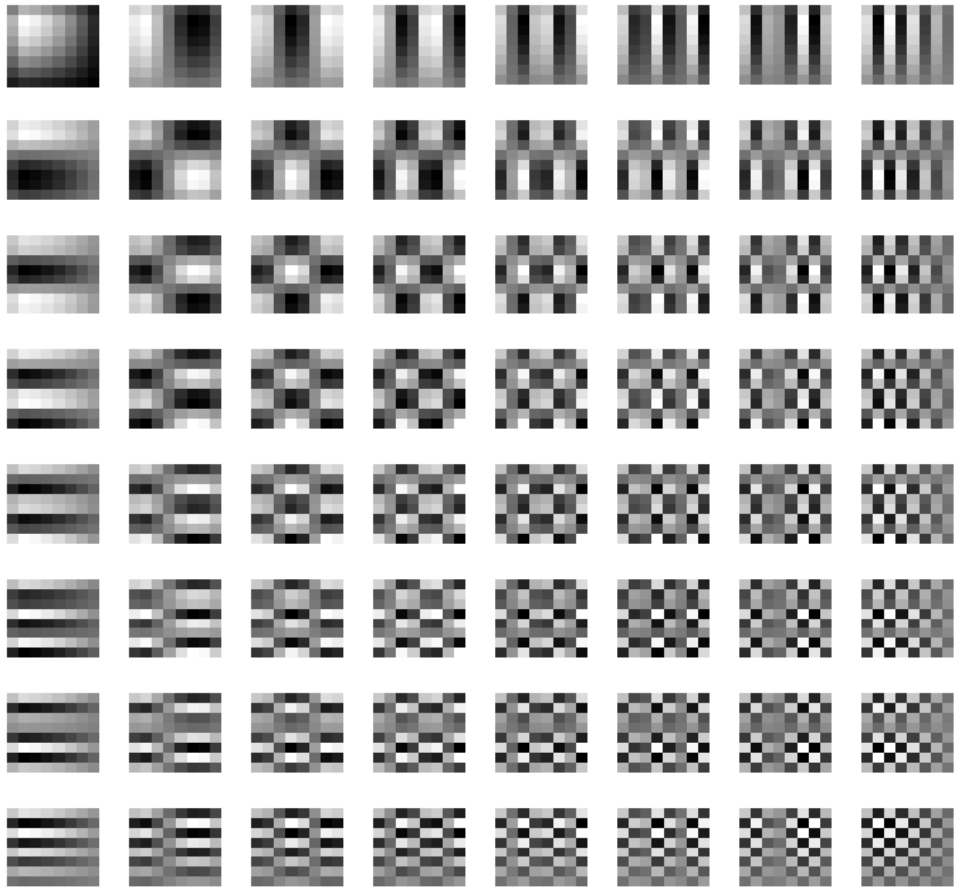
non overlapping blocks of size 8×8

$$C_u = \begin{cases} 1 & \text{if } u \neq 0, \\ \frac{1}{\sqrt{2}} & \text{if } u = 0. \end{cases}$$

- Given the coefficients, one can use the inverse DCT transform to reconstruct an estimate of the block:

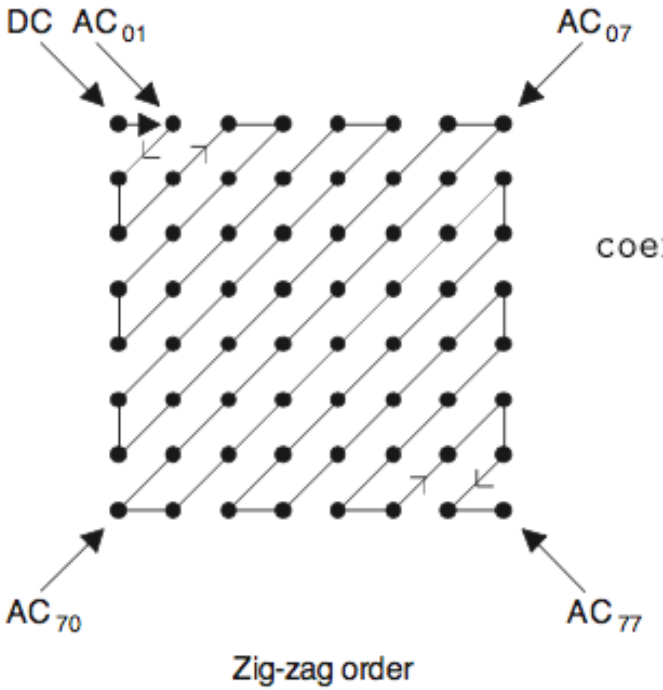
$$\widetilde{f}(x,y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u \, C_v \, \widetilde{F}_{u,v} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}, \quad x, y = 0, \ldots 7$$

2D DCT



The coefficient matrix

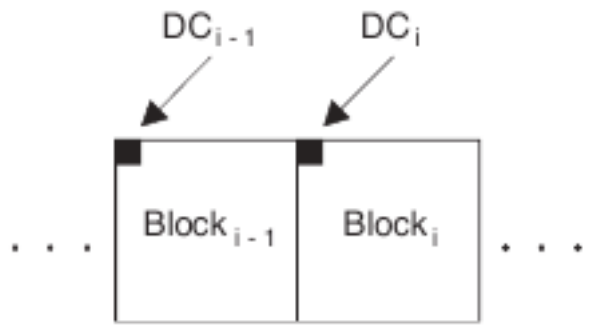
- The DCT coefficients for the entire image are returned into a matrix coeff of size 64xN, where N is the number of 8x8 blocks inside the image.




```
coeff(2,b) = F(1,2), coeff(3,b) = F(2,1), coeff(4,b) = F(3,1),...
```

The coefficient matrix

$F_i(1,1)$ is the zero frequency DCT coefficient of block i.

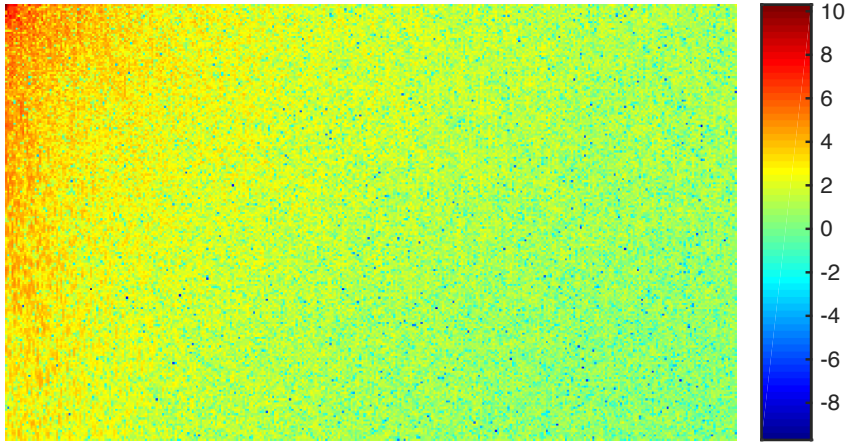


Differential DC encoding

$$\begin{aligned}
\texttt{coeff(1,1)} &= F_1(1,1),\\
\texttt{coeff(1,2)} &= F_2(1,1) - F_1(1,1),\\
\texttt{coeff(1,3)} &= F_3(1,1) - F_2(1,1),\\
&\ldots
\end{aligned}$$

Example

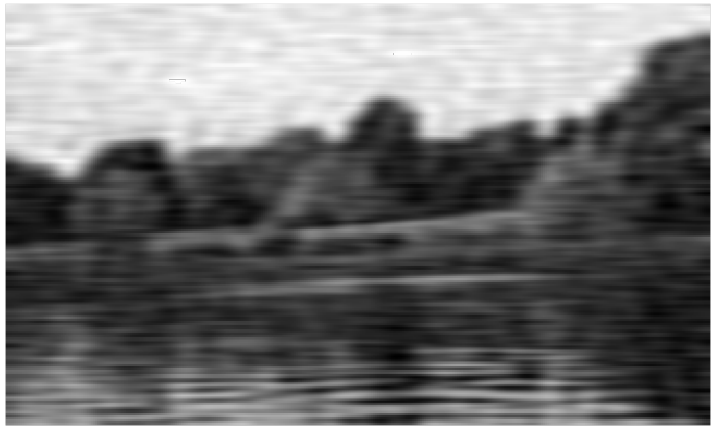
```
>> RGB = imread('autumn.tif');
>> I = rgb2gray(RGB);
>> J = dct2(I);
>> imshow(log(abs(J)),[]), colormap(jet(64)), colorbar
```



Example

```
>> J(abs(J) < 10) = 0;
>> K = idct2(J);
>> figure, imshow(K,[0 255])
```

```
>> J(abs(J) < 10^2) = 0;
>> K = idct2(J);
>> figure, imshow(K,[0 255])
```



Quantization and inverse quantization

- The effect of the DCT is to create many small coefficients that are close to zero, and a small number of large coefficients.
- In order to benefit from this redistribution of the energy, we need to simplify the representation of the **floating numbers** that are used to describe the DCT coefficients.
- In the JPEG compression standard, the quantization of the DCT coefficients is performed as follows:

$$F^q_{u,v} = \text{round}\left(\frac{F_{u,v}}{\texttt{loss-factor} \times Q_{u,v}}\right)$$

a user supplied parameter

quantization table

Quantization and inverse quantization

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Figure 4: The quantization table Q

Quantization and inverse quantization

- The inverse quantization of the DCT coefficients is defined by the map:

$$\widetilde{F}_{u,v} = F^{q}_{u,v} \times \texttt{loss-factor} \times Q_{u,v}.$$

- In principle quantization is the only mechanism in a compression system where irrecoverable loss is introduced.

Peak Signal to Noise Ratio (PSNR)

- The Peak Signal to Noise Ratio (PSNR) between the original image, and the reconstructed image:

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\frac{1}{N^2} \sum_{i,j=0}^{N-1} |f(i,j) - \widetilde{f}(i,j)|^2} \right)$$

Part 2: Sparse Representation Model

Introduction

- Previously: We looked at **Discrete Cosine Transform (DCT)** for images.
- However, one might wonder if there is a **best basis** for a **given class of images** and, if so, how to find it.

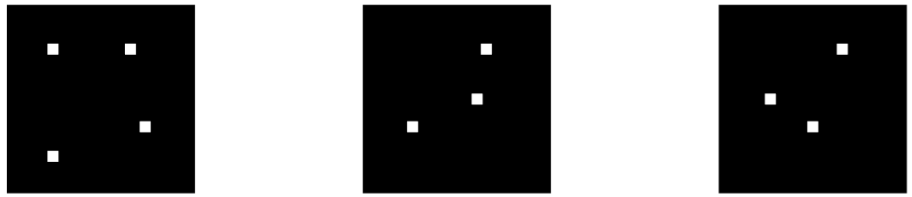
Dictionary learning

- For a given population of images, find the set of **building blocks** in which they will be represented most efficiently.

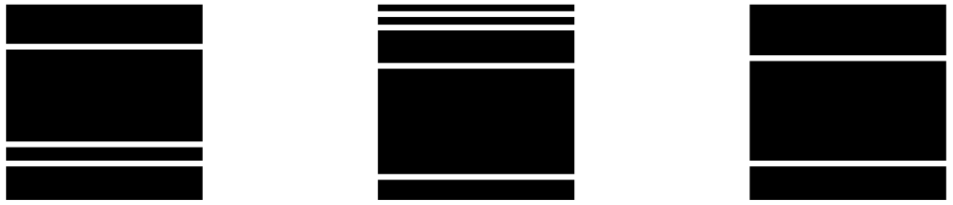
- This idea of **trying to find an optimal dictionary for a given class of images** was originally called **sparse coding** and more recently has been called **dictionary learning**.

Which dictionary is best?

- Of course, which dictionary is best depends on the class of signals being represented:
- Example: What dictionary might you use to represent images like these?



- For images like these?



Dictionary learning: Goals

- Given a set of training signals $y_1, \dots, y_n$ in R^p we aim to **find dictionary elements $D_1, \dots, D_d$** to optimize two criteria:
  - How **few of these dictionary elements** are needed to approximate each of $y_1, \dots, y_n$
  - How **accurate** (in terms of MSE) each of the resulting approximations is

Dictionary learning: Important note

- Generally, we need n > d (i.e. more training images than dictionary elements) to have a meaningful result.
- Otherwise, we could just set the dictionary elements equal to the training signals, yielding perfect 1-element approximations for each signal!

Definition: Sparsity

- We say that a vector is **T-sparse** if it has **at most** T nonzero entries:

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \pi \\ 0 \end{bmatrix}$$

Example: The vector at left is a 2-sparse vector, it is also 3-sparse and 4-sparse, but not 1-sparse.

Dictionary learning goals revisited

- Given: A set of training examples: $y_1, \ldots, y_n \in \mathbb{R}^p$
- Goal: To learn a set of dictionary elements

  $D_1, \ldots, D_d \in \mathbb{R}^p$ so that for each i:

$$y_i \approx \begin{bmatrix} | & | & & | \\ D_1 & D_2 & \ldots & D_d \\ | & | & & | \end{bmatrix} c_i = Dc_i$$

for some T-sparse: $c_i \in \mathbb{R}^d$

Dictionary learning: Optimization set-up

- We turn these goals into an optimization problem:

Given: $y_1, \ldots, y_n \in \mathbb{R}^p$

Solve: $$\min_{D \in \mathbb{R}^{p \times d}, c_1, \ldots, c_n \in \mathbb{R}^d} \sum_{i=1}^{n} ||y_i - Dc_i||^2$$

subject to: c_i is T-sparse for all i

Signal approximation error using the dictionary

Optimizing over **both** the dictionary and the coefficients

This is a hard optimization problem. Different dictionary learning algorithms attempt to optimize in different ways.

Variations on the dictionary learning setup

- Alternatively, some dictionary learning algorithms, such as the original sparse coding algorithm, solve instead:

$$\min_{D \in \mathbb{R}^{p \times d}, c_1, \ldots, c_n \in \mathbb{R}^d} \sum_{i=1}^{n} ||y_i - Dc_i||^2 + \underbrace{\sum_{i=1}^{n} f_{penalty}(c_i)}$$

Penalty function designed to encourage (but not strictly require) sparsity in c

Examples:

$$f_{penalty}(c_i) = \sum_{j=1}^{d} \log(1 + ((c_i)_j)^2)$$

or

$$f_{penalty}(c_i) = \sum_{j=1}^{d} |(c_i)_j|$$

The K-SVD dictionary learning algorithm

- K-SVD is a particularly successful algorithm for the optimization problem below:

$$\min_{D \in \mathbb{R}^{p \times d}, c_1, \ldots, c_n \in \mathbb{R}^d} \sum_{i=1}^{n} ||y_i - D c_i||^2$$

$$\text{subject to: } c_i \text{ is T-sparse for all } i$$

- It alternates between
  - optimizing c for fixed D and
  - updating the elements of D (with their coefficients) one at a time while fixing everything else

K-SVD coefficient update

- First: Coefficient update step for fixed D:
  - Looking at:

$$\min_{c_1,\ldots,c_n \in \mathbb{R}^d} \sum_{i=1}^{n} ||y_i - Dc_i||^2$$
$$\text{subject to: } c_i \text{ is T-sparse for all } i$$

Not optimizing over D in this step

  - It is clear that we can consider the vectors c_i individually:

$$\min_{c_i \in \mathbb{R}^d} ||y_i - Dc_i||^2$$
$$\text{subject to: } c_i \text{ is T-sparse}$$

Solve this for each c_i

K-SVD coefficient update

- But this:

$$\min_{c_i \in \mathbb{R}^d} ||y_i - Dc_i||^2$$

$$\text{subject to: } c_i \text{ is T-sparse}$$

is just a regular sparse approximation problem!

- We are approximating a known signal y_i using at most T elements from a fixed dictionary D.
- Can use orthogonal matching pursuit or other techniques.

Orthogonal Matching Pursuit (OMP)

Algorithm 1 ORTHOGONAL MATCHING PURSUIT

1: Input: Dictionary $\mathbf{D}$, signal $\underline{x}$, target sparsity K *or* target error ϵ
2: Output: Sparse representation $\underline{\gamma}$ such that $\underline{x} \approx \mathbf{D}\underline{\gamma}$
3: Init: Set $I := (\,)$, $\underline{r} := \underline{x}$, $\underline{\gamma} := \underline{0}$
4: **while** (*stopping criterion not met*) **do**
5: $\quad \hat{k} := \underset{k}{\text{Argmax}} \; |\, \underline{d}_k^T \, \underline{r} \,|$
6: $\quad I := (\, I, \, \hat{k} \,)$
7: $\quad \underline{\gamma}_I := (\mathbf{D}_I)^+ \underline{x}$
8: $\quad \underline{r} := \underline{x} - \mathbf{D}_I \underline{\gamma}_I$
9: **end while**

K-SVD dictionary update

- Second: Optimize individual dictionary elements (with their coefficients) while fixing everything else:
  - We'll try to solve:

$$\min_{D_k \in \mathbb{R}^p, (c_1)_k, \ldots, (c_n)_k \in \mathbb{R}} \sum_{i=1}^{n} ||y_i - Dc_i||^2$$

$$\text{subject to: the } (c_i)_k \text{ that are zero stay zero}$$

K-SVD dictionary update

- To solve this, we'll rewrite the objective a bit:

$$\min_{D_k \in \mathbb{R}^p, (c_1)_k, \ldots, (c_n)_k \in \mathbb{R}} \sum_{i=1}^{n} ||y_i - Dc_i||^2$$

columns of matrix D

$$\min_{D_k \in \mathbb{R}^p, (c_1)_k, \ldots, (c_n)_k \in \mathbb{R}} \sum_{i=1}^{n} ||y_i - \sum_{j=1}^{d} D_j (c_i)_j||^2$$

$$\min_{D_k \in \mathbb{R}^p, (c_1)_k, \ldots, (c_n)_k \in \mathbb{R}} \sum_{i=1}^{n} \left|\left| \left(y_i - \sum_{j=1, j \neq k}^{d} D_j (c_i)_j \right) - D_k (c_i)_k \right|\right|^2$$

K-SVD dictionary update

$$\min_{D_k \in \mathbb{R}^p, (c_1)_k, \ldots, (c_n)_k \in \mathbb{R}} \sum_{i=1}^{n} \left|\left| \underbrace{\left(y_i - \sum_{j=1, j \neq k}^{d} D_j (c_i)_j \right)}_{} - D_k (c_i)_k \right|\right|^2$$

Remainder of y_i when approximated by all the other fixed dictionary elements. (Doesn't depend on any of our optimization variables). Call this $R_i^{(k)}$

$$\min_{D_k \in \mathbb{R}^p, (c_1)_k, \ldots, (c_n)_k \in \mathbb{R}} \sum_{i=1}^{n} \left|\left| R_i^{(k)} - D_k (c_i)_k \right|\right|^2$$

K-SVD dictionary update

- Now rewrite

$$\min_{D_k \in \mathbb{R}^p, (c_1)_k, \ldots, (c_n)_k \in \mathbb{R}} \sum_{i=1}^{n} \left\| \begin{bmatrix} | \\ R_i^{(k)} \\ | \end{bmatrix} - \begin{bmatrix} | \\ D_k \\ | \end{bmatrix} (c_i)_k \right\|^2$$

as

$$\min_{D_k \in \mathbb{R}^p, (c_1)_k, \ldots, (c_n)_k \in \mathbb{R}} \left\| \begin{bmatrix} | & | & & | \\ R_1^{(k)} & R_2^{(k)} & \cdots & R_n^{(k)} \\ | & | & & | \end{bmatrix} - \begin{bmatrix} | \\ D_k \\ | \end{bmatrix} \begin{bmatrix} (c_1)_k & (c_2)_k & \cdots & (c_n)_k \end{bmatrix} \right\|_F^2$$

(Switched from a sum of vector norms to one big matrix norm.)

Here, if no restrictions on c, optimal D_k and $(c_i)_k$ can be found as best rank-1 approximation to the R matrix via singular value decomposition of R.

K-SVD dictionary update

$$\min_{D_k \in \mathbb{R}^p, (c_1)_k, \ldots, (c_n)_k \in \mathbb{R}} \left\| \begin{bmatrix} | & | & & | \\ R_1^{(k)} & R_2^{(k)} & \cdots & R_n^{(k)} \\ | & | & & | \end{bmatrix} - \begin{bmatrix} | \\ D_k \\ | \end{bmatrix} \begin{bmatrix} (c_1)_k & (c_2)_k & \cdots & (c_n)_k \end{bmatrix} \right\|_F^2$$

$$\text{subject to: the } (c_i)_k \text{ that are zero stay zero}$$

- But there are some restrictions that some $(c_i)_k$ stay zero, so we remove these (and corresponding columns of the R matrix) from the expression above, and do the SVD without them.

K-SVD dictionary update

$$\min_{D_k \in \mathbb{R}^p, (c_1)_k \cancel{,} \ldots, (c_n)_k \in \mathbb{R}} \left\| \begin{bmatrix} | & \cancel{|} & & | \\ R_1^{(k)} & \cancel{R_2^{(k)}} & \cdots & R_n^{(k)} \\ | & \cancel{|} & & | \end{bmatrix} - \begin{bmatrix} | \\ D_k \\ | \end{bmatrix} \begin{bmatrix} (c_1)_k & \cancel{(c_2)_k} & \cdots & (c_n)_k \end{bmatrix} \right\|_F^2$$

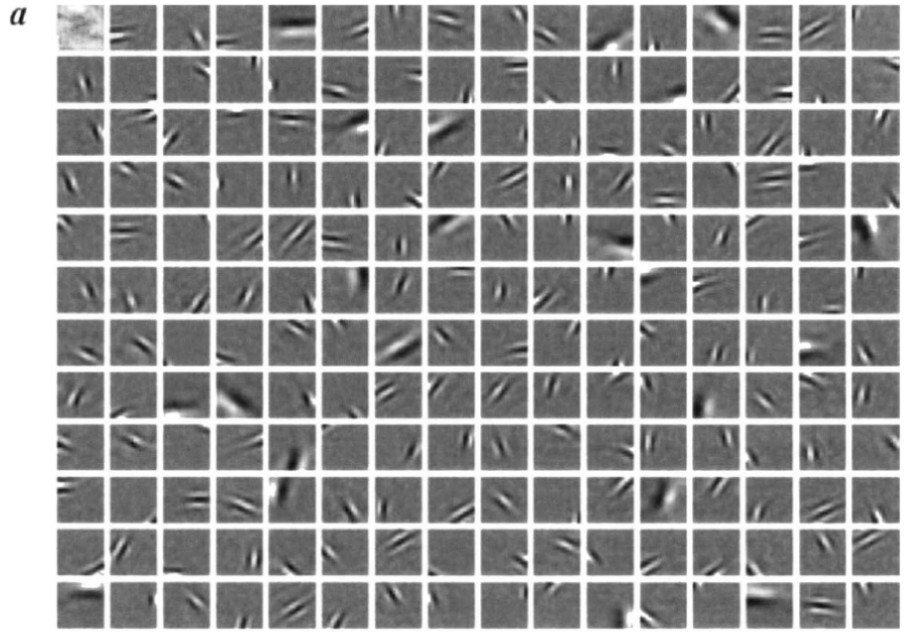
(No longer need constraint because zero coefficients have been removed from the problem.)

- But there are some restrictions that some $(c_i)_k$ stay zero, so we remove these (and corresponding columns of the R matrix) from the expression above, and do the SVD without them.

Example

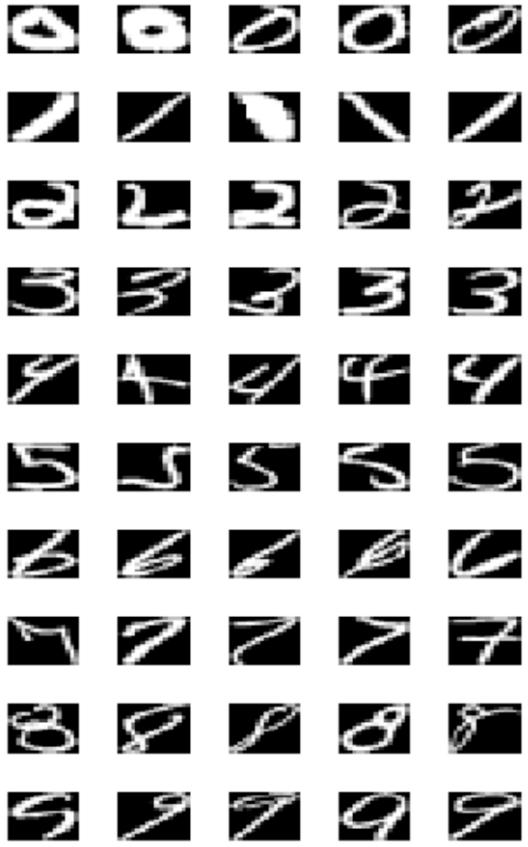
- Sparse Coding on Small Patches of Natural Images (scenes of landscapes):

Resulting dictionary elements $D_1, \ldots, D_d$:



Notice anything?

Example: K-SVD on USPS Dataset



USPS dataset:
United States Postal Service images of digits 0 through 9 taken from zip codes handwritten on mail

Example: K-SVD on USPS dataset