COMP 790-125: Goals for today

- Dictionaries
- Orthonormal dictionaries
- Sparse reconstructions using dictionaries

Dictionaries – setting up intuitions

A very simple dictionary for 3d data:

$$\mathbf{D} = \begin{bmatrix} \underbrace{\begin{matrix}1\\0\\0\end{matrix}}_{\text{dictionary element}} & \underbrace{\begin{matrix}0\\1\\0\end{matrix}}_{\text{dictionary element}} & \underbrace{\begin{matrix}0\\0\\1\end{matrix}}_{\text{dictionary element}} \end{bmatrix}$$

Representing a data vector $\mathbf{y} \in \mathbf{R}^3$ using dictionary $\mathbf{D}$

$$\mathbf{y} = \mathbf{D}\alpha = \sum_i \mathbf{d}_i \alpha_i$$

Somewhat boring since $\mathbf{D}$ contains a basis of $\mathbf{R}^3$ so any vector $\mathbf{y} \in \mathbf{R}^3$ is exactly representable using these dictionary elements.

Dictionaries – setting up intuitions

Generally we use dictionaries that are smaller than the dimensionality of vectors we try to represent.

An example

Data: $\mathbf{y} \in \mathbf{R}^{100}$

Dictionary: $\mathbf{D} \in \mathbf{R}^{100 \times 10}$
This is a dictionary with 10 dictionary elements (or words) used to represent 100 dimensional vectors.

Hence

$$\mathbf{y} = \mathbf{D}\boldsymbol{\alpha}$$

Q:What is the dimensionality of $\boldsymbol{\alpha}$? Contrast this to dimensionality of $\mathbf{y}$? What is the benefit ?

Sliding windows and image patches

Typically, but not always, dictionaries are trained on small fragments of the data that can be thought as words.

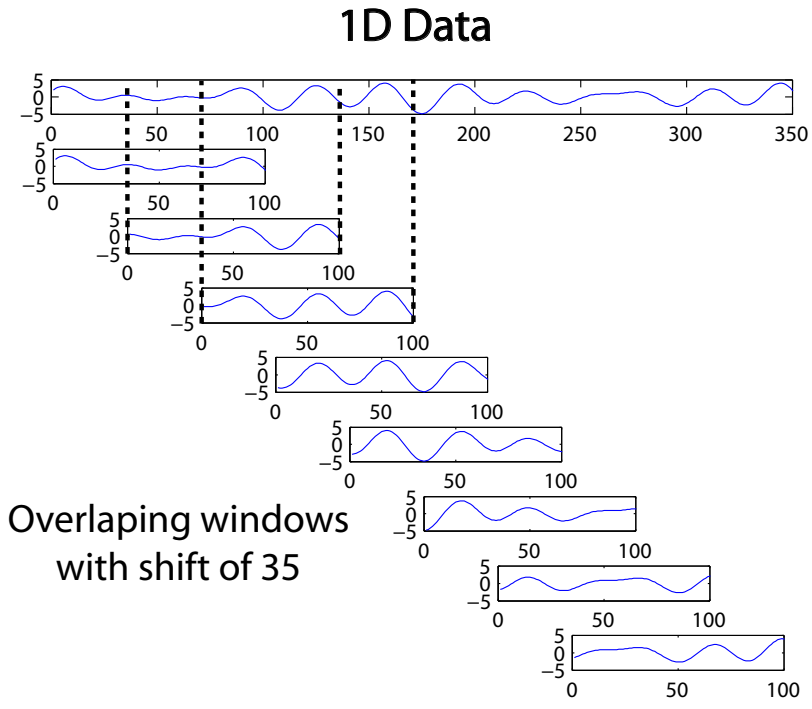
For example, in sequence data: MPIGSKERPTFEIG, we extract overlapping windows:

- MPIGSK
- PIGSKE
- IGSKER
- etc.

In comp.bio. these are called *k-mers*, example above are 6-mers.

Sliding windows and image patches

In 1d continuous data, sliding window yields words

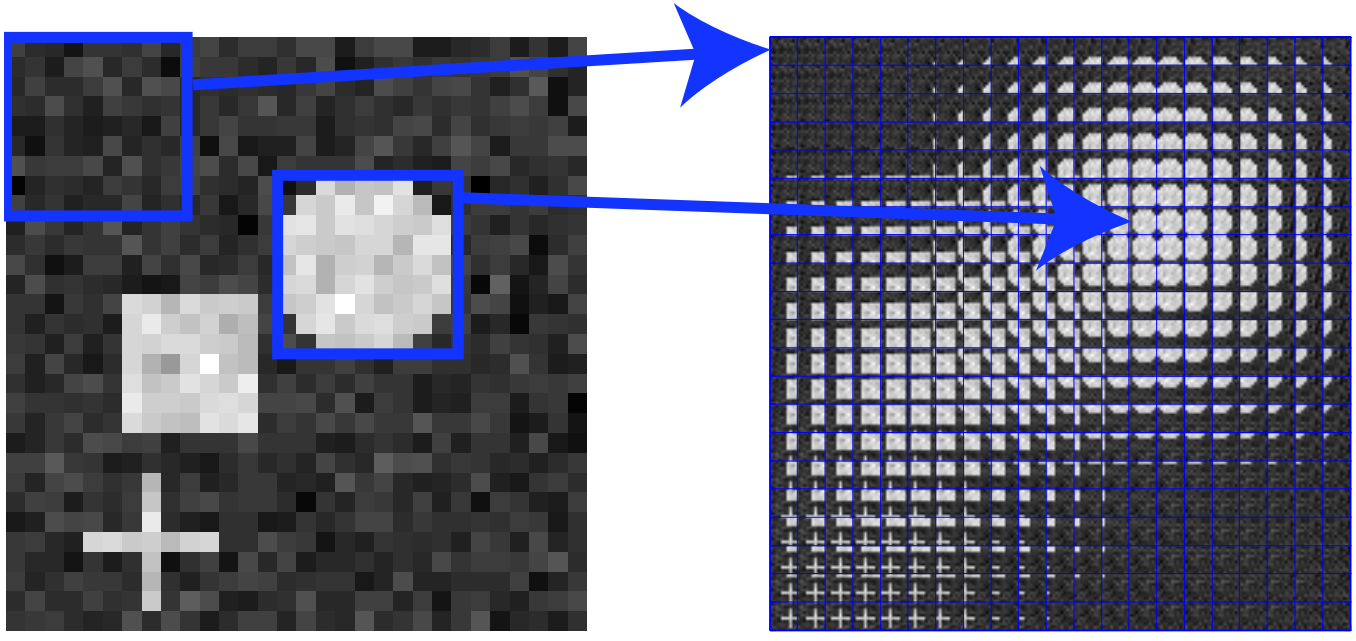


Sliding windows and image patches

In images or 2d data we extract patches

Original Image

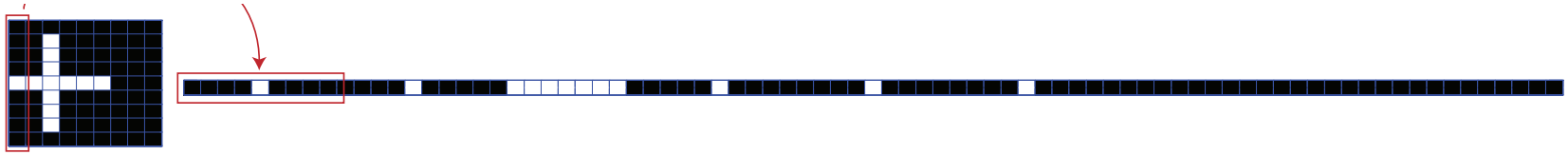
Set of all the extracted patches pasted together



Vectorizing 2d data

To make math easier, we assume words are always vectors.

We vectorize multidimensional objects by just unrolling the data:



In MATLAB:

- `x(:)` will produce a vector regardless of structure of x
- `reshape(v,[m n])` will reshape a vector v into a matrix of size $m \times n$

In math we will use $\mathrm{vec}(\cdot)$ and $\mathrm{mat}_{m \times n}(\cdot)$ to refer to these operations.

Getting vectorized patches out of vectorized images

We are going to vectorize a 3×3 image and illustrate a linear mapping that extracts a patch from vectorized image

$$\mathbf{y} = \mathrm{vec}\left(\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \right) = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9]^T$$

Mapping matrix that extracts the blue patch

$$\mathbf{R}_{1,2} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and

$$\mathbf{x} = \mathbf{R}_{1,2}\mathbf{y} = \begin{bmatrix} 4 & 5 & 7 & 8 \end{bmatrix}^T$$

A little bit more on patch extraction mappings

$$\mathbf{R}_{1,2} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{R}_{1,2}\mathbf{y} = \begin{bmatrix} 4 & 5 & 7 & 8 \end{bmatrix}^T$$

Q1: What do you obtain when you do: $\mathrm{mat}_{3\times 3}(\mathbf{R}_{1,2}^T\mathbf{x})$?

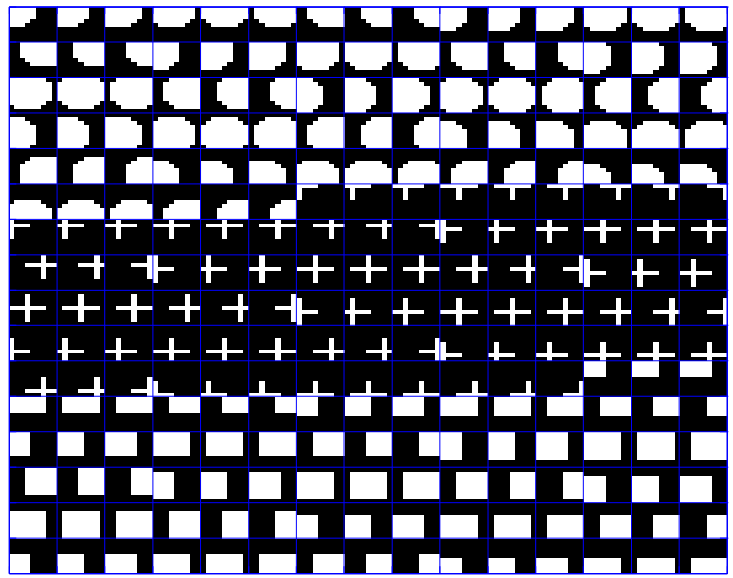
Q2: What is the size of $\mathbf{R}_{1,2}^T\mathbf{R}_{1,2}$? What is it equal to?

Q3: In general for a matrix $\mathbf{R}$ that has single 1 per column, at most single 1 per row, and otherwise it is all 0, what does $\mathbf{R}^T\mathbf{R}$ look like?

Q4: What would $\sum_{i,j}\mathbf{R}_{i,j}^T\mathbf{R}_{i,j}$ be equal to? What would inverse of that matrix be equal to?

Modeling patch sets

What would be the dictionary that explains images built up out of crosses, squares, and circles look like?



Note that the patches we extract from images are usually overlapping, hence our dictionary needs to be able to cope with shifts.

What can we do with a dictionary?

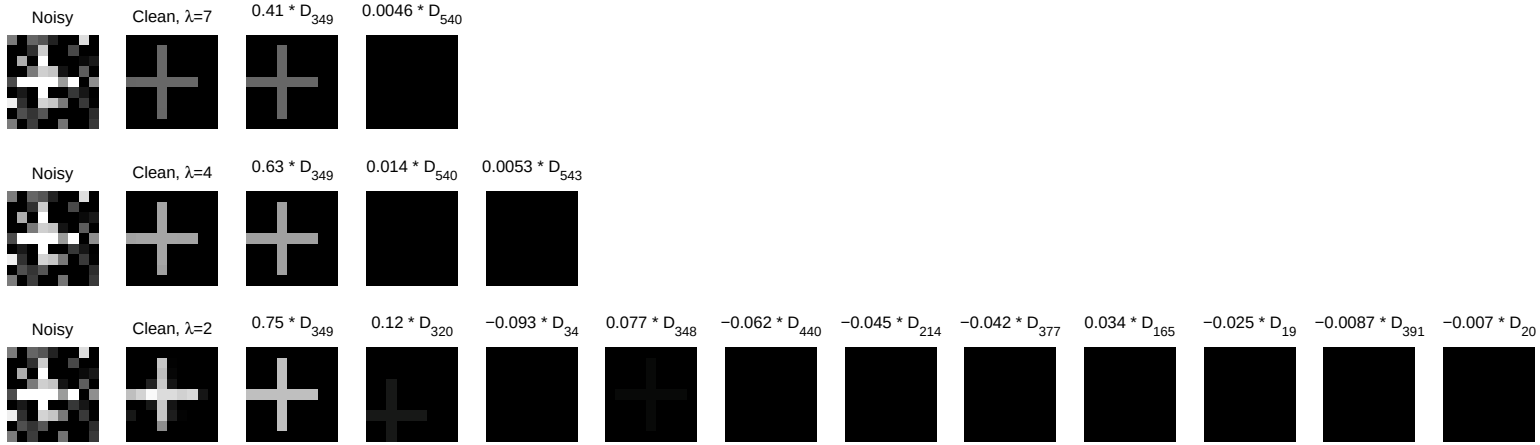
Say we take a $\mathbf{D}$ and a corrupted image $\mathbf{y}$ and solve following optimization problem

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2} \left\| \underbrace{\mathbf{R}_{i,j}\mathbf{y}}_{\text{extracted patch}} - \underbrace{\mathbf{D}\boldsymbol{\alpha}}_{\substack{\text{combination of} \\ \text{dictionary elements}}} \right\|_2^2 + \lambda \underbrace{\|\boldsymbol{\alpha}\|_1}_{\text{sparsity penalty}}$$

Q:What does this optimization problem accomplish?

What can we do with a dictionary?

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2} \left\| \underbrace{\mathbf{R}_{i,j}\mathbf{y}}_{\text{extracted patch}} - \underbrace{\mathbf{D}\boldsymbol{\alpha}}_{\substack{\text{combination of} \\ \text{dictionary elements}}} \right\|_2^2 + \lambda \underbrace{\|\boldsymbol{\alpha}\|_1}_{\text{sparsity penalty}}$$



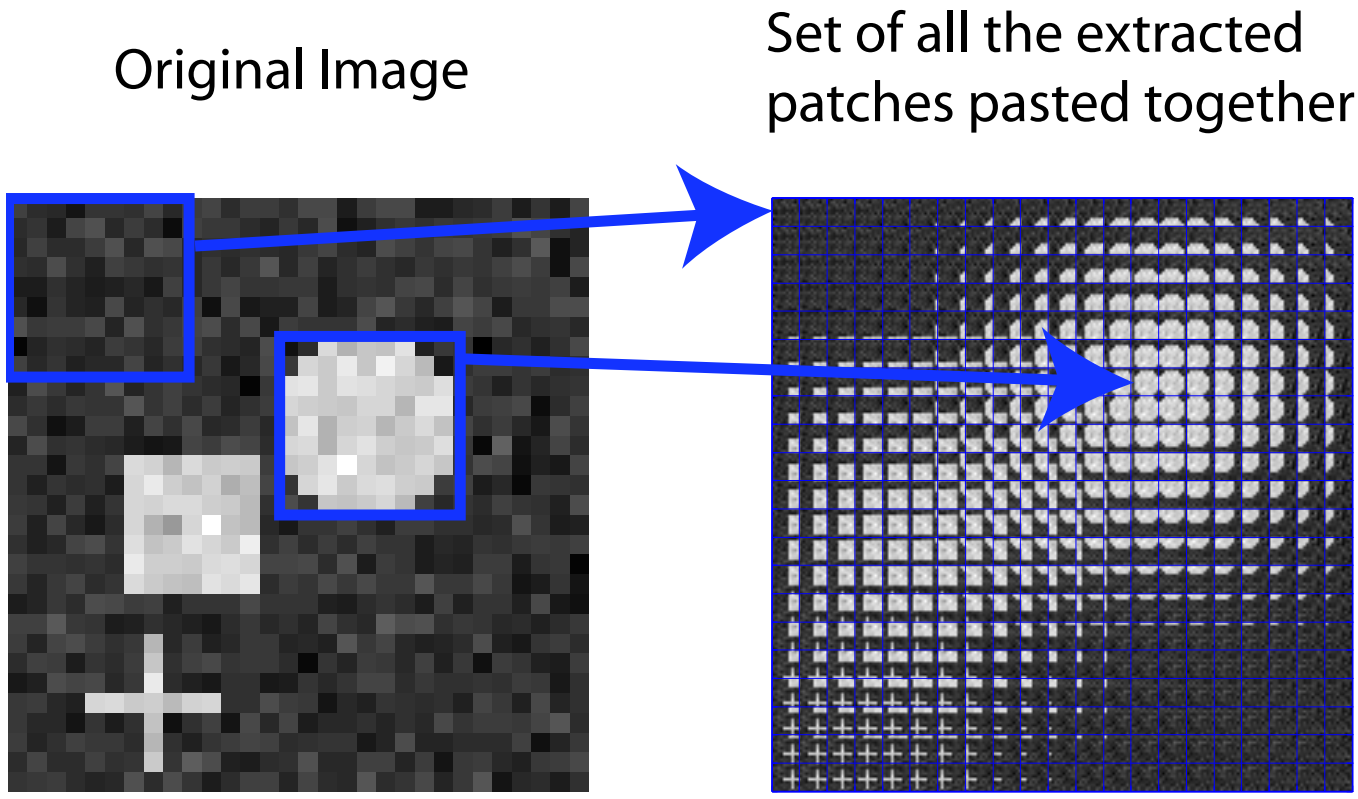
Weights $\boldsymbol{\alpha}$ of dictionary elements, $\mathbf{D}$, encode a clean version of the patch, $\mathbf{D}\alpha$.

Put differently we represented patch by weights $\boldsymbol{\alpha}$ instead of gray intensity values.

Denoising the whole image

Previous optimization problem just yielded weights for different dictionary elements.

Remember we broke the image down into patches



Q: If we denoise each patch separately, how do we reconstruct the denoised image?

Reconstructing denoised image

We can create a different optimization problem that explicitly encodes a denoised image $\mathbf{x}$

$$\underset{\boldsymbol{\alpha},\mathbf{x}}{\operatorname{argmin}} \frac{\mu}{2} \|\mathbf{x}-\mathbf{y}\|_2^2 + \sum_{i,j} \left\{ \frac{1}{2} \|\mathbf{R}_{i,j}\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}_{i,j}\|_2^2 + \lambda \|\boldsymbol{\alpha}_{i,j}\|_1 \right\}$$

How do we solve for $\mathbf{x}$ in closed form?

Taking a gradient with respect to $\mathbf{x}$ and equating to 0 gives

$$\mathbf{x} = (\mu\mathbf{I} + \sum_{i,j} \mathbf{R}_{i,j}^T \mathbf{R}_{i,j})^{-1} (\mu\mathbf{y} + \sum_{i,j} \mathbf{R}_{i,j}^T \mathbf{D}\alpha_{i,j})$$

Derivation is given at the end of slides.

Reconstructing denoised image

$$\mathbf{x} = (\mu\mathbf{I} + \sum_{i,j} \mathbf{R}_{i,j}^T \mathbf{R}_{i,j})^{-1} (\mu\mathbf{y} + \sum_{i,j} \mathbf{R}_{i,j}^T \mathbf{D}\alpha_{i,j})$$

Making sense of the equation

- $\mu\mathbf{y}$ scaled version of original image
- $\mathbf{D}\alpha_{i,j}$ reconstructed patch (i,j) using dictionary
- $\mathbf{R}_{i,j}^T \mathbf{D}\alpha_{i,j}$ reconstructed patch (i,j) plugged into an otherwise empty image
- $(\mu\mathbf{y} + \sum_{i,j} \mathbf{R}_{i,j}^T \mathbf{D}\alpha_{i,j})$ a weighted average of original image and patch based reconstructions
- $(\mu\mathbf{I} + \sum_{i,j} \mathbf{R}_{i,j}^T \mathbf{R}_{i,j})^{-1}$ a scaling factor that accounts for number of patches that cover a particular pixel in image

A simple dictionary

The most boring dictionary ever, diagonal matrix:

$$\mathbf{D} = \mathbf{I}$$

It has one nice property. It is orthonormal, that is to say $\mathbf{d}_i^T \mathbf{d}_j = 0, i \neq j$ and $\mathbf{d}_i^T \mathbf{d}_i = 1$.

$$\mathbf{y} = [1\ 2\ 3]^T \ \mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Q1: If we have to minimize $\frac{1}{2} \|\mathbf{y} - \mathbf{D}\mathbf{w}\|_2^2$ with respect to $\mathbf{w}$ what is the solution? Equivalently, how do we combine columns of $\mathbf{D}$ to obtain $\mathbf{y}$?

Representation with orthonormal dictionaries

Let's look at the general problem again

$$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{D}\mathbf{w}\|_2^2 = \frac{1}{2}\sum_k (y_i - \sum_l d_{i,k} w_k)^2$$

and take a partial derivative with respect to w_j

$$\begin{aligned}
\frac{\partial f(\mathbf{w})}{\partial w_k} &= -\sum_i (y_i - \sum_k d_{i,k} w_k) d_{i,j} \\
&= -\sum_i y_i d_{i,j} + \sum_i \sum_k d_{i,k} d_{i,j} w_k \\
&= -\sum_i y_i d_{i,j} + \sum_k w_k \sum_i d_{i,k} d_{i,j} \\
&= -\sum_i y_i d_{i,j} + \sum_k w_k \mathbf{d}_k^T \mathbf{d}_j \\
&= -\sum_i y_i d_{i,j} + w_j \mathbf{d}_j^T \mathbf{d}_j = -\sum_i y_i d_{i,j} + w_j
\end{aligned}$$

So, equating the partial derivative to zero $w_j = \mathbf{y}^T \mathbf{d}_j$

Representation with orthonormal dictionaries

Unpenalized objective

$$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{D}\mathbf{w}\|_2^2 = \frac{1}{2}\sum_k (y_i - \sum_l d_{i,k} w_k)^2$$

update is:

$$w_j = \mathbf{y}^T \mathbf{d}_j$$

A little test, if

$$\mathbf{y} = \begin{bmatrix} 0.25 & 2 & -3 \end{bmatrix}^T, \ \mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

what is optimal $\mathbf{w}$?

Representation with orthonormal dictionaries and L1 penalty

L1 Penalized objective

$$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{D}\mathbf{w}\|_2^2 = \frac{1}{2}\sum_k (y_i - \sum_l d_{i,k} w_k)^2 + \lambda \|\mathbf{w}\|_1$$

If the dictionary is orthonormal, the closed form solution is:

$$w_j = S(\mathbf{y}^T \mathbf{d}_j, \lambda)$$

where $S(x, \lambda) \equiv \mathrm{sign}(x)\max(|x| - \lambda, 0)$

Same little test but with penalties

Little test:

$$\mathbf{y} = \begin{bmatrix} 0.25 & 2 & -3 \end{bmatrix}^T, \ \mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \lambda = 0.5$$

What is

$$\underset{\mathbf{w}}{\operatorname{argmax}} \frac{1}{2} \|\mathbf{y} - \mathbf{D}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1 ?$$

Because $\mathbf{D}$ is orthonormal: $w_j = S(\mathbf{y}^T \mathbf{d}_j, \lambda)$

$$\mathbf{y} = \begin{bmatrix} 0 & 1.5 & -2.5 \end{bmatrix}^T$$

The shrinkage-and-thresholding operator arose in context of penalized regression on orthonormal directories.

Popular orthonormal dictionaries

Other popular orthonormal dictionaries

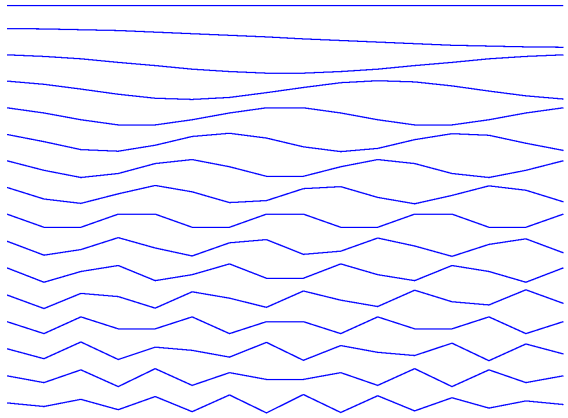
- Discrete Fourier Transform dictionaries
- Discrete Cosine Transform dictionaries
- Wavelets

We will look at a 2D DCT dictionary because they get used in JPEG, MPEG, H.261, H.263 compression.[1]

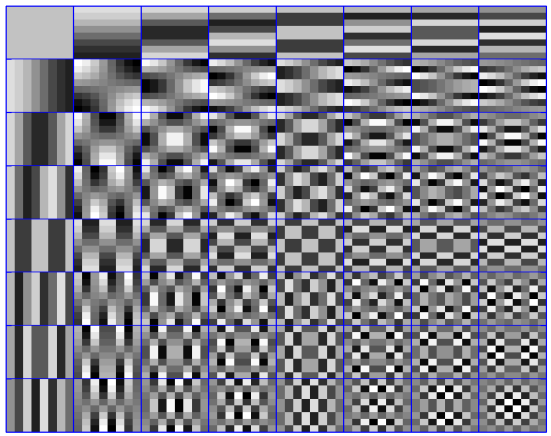
[1] And it is easy for us to talk about it right now

One popular dictionary for natural images

Highly popular dictionary used for natural image patches is called DCT (discrete cosine transform) dictionary 1D DCT dictionary with 16 elements (`dctmtx(16)'`)



2D DCT dictionary with 64 elements (`D2=kron(dctmtx(8),dctmtx(8))'`).



Kronecker product – almost straight from MATLAB help on kron

Kronecker product of two matrices is a large matrix obtained by taking all possible products of entries of the two matrices.

$$\mathbf{X} \circ \mathbf{Y} = \begin{bmatrix} x_{1,1} * \mathbf{Y} & x_{1,2} * \mathbf{Y} & \dots & x_{1,m} * \mathbf{Y} \\ x_{2,1} * \mathbf{Y} & x_{2,2} * \mathbf{Y} & \dots & x_{2,m} * \mathbf{Y} \\ \dots & & & \\ x_{n,1} * \mathbf{Y} & x_{n,2} * \mathbf{Y} & \dots & x_{n,m} * \mathbf{Y} \end{bmatrix}$$

where $\mathbf{X}$ is an $n \times m$ matrix.

Quick numerical check for orthonormality

If $\mathbf{D}$ is orthonormal, what does

$$\mathbf{G} = \mathbf{D}^T \mathbf{D}$$

look like?

$\mathbf{G}$ is called Gramian matrix.

How would you measure how far from orthonormality a dictionary is?

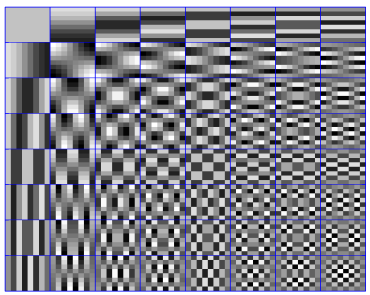
One way

$$\mu(\mathbf{D}) = \max_{i,j} |\mathbf{G}_{i,j}| = \max_{i,j} |\mathbf{d}_i^T \mathbf{d}_j^T|$$

Using DCT dictionary to denoise an image

We will use DCT dictionary[2]: `D = kron(dctmtx(8),dctmtx(8))'`

Dictionary



Original image

Noisy image

$\lambda = 0.1$

$\lambda = 0.2$

$\lambda = 0.3$

$\lambda = 0.4$

$\lambda = 0.5$

Q: Which parts of the image get denoised well and which get denoised poorly? Relate that to the dictionary elements.

[2] Btw. to see one of the elements do `imagesc(reshape(D(:,10),[8 8]))`

Learning dictionaries

Note that the dictionary was not adapted to data. Could we do better?

We will try a simple experiment:

1. Learn dictionary on one half of the image

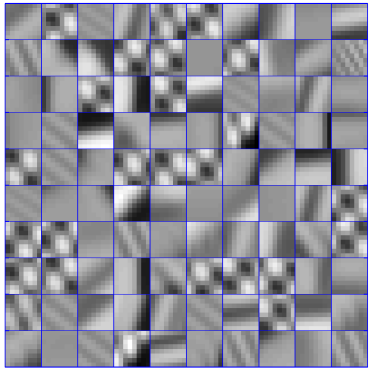
2. Denoise the other half

This is not a *natural* scenario[3], but a step towards one.

[3] Because, you do not often do you see an image that is half noisy and half clean.

Learning dictionaries

Learned Dictionary



Train image

Test image

$\lambda = 0.25$

$\lambda = 0.5$

$\lambda = 0.75$

$\lambda = 1.0$

$\lambda = 1.25$

We covered

- Dictionaries
- Orthonormal dictionaries
- Sparse reconstructions using dictionaries

Next time

- Learning dictionaries
- Using representations in learned dictionaries for classification

Appendix: Solving for reconstruction image

Optimization problem that recovers reconstruction $\mathbf{x}$ is given by minimizing

$$f(\mathbf{x}) = \frac{\mu}{2}\left\|\mathbf{x}-\mathbf{y}\right\|_2^2 + \sum_{i,j}\left\{\frac{1}{2}\left\|\mathbf{R}_{i,j}\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}_{i,j}\right\|_2^2 + +\lambda\left\|\boldsymbol{\alpha}_{i,j}\right\|_1\right\}$$

We will use: $\nabla_{\mathbf{x}}\left\|\mathbf{A}\mathbf{x}-\mathbf{y}\right\|_2 = (\mathbf{A}\mathbf{x}-\mathbf{y})^T\mathbf{A}$

Solve for $\mathbf{x}$

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \mu(\mathbf{x}-\mathbf{y})^T\mathbf{I} + \sum_{i,j}(\mathbf{R}_{i,j}\mathbf{x}-\mathbf{D}\alpha_{i,j})^T(\mathbf{R}_{i,j})$$

After equating gradient to zero and some rearrangement

$$\begin{aligned}
\mathbf{x}^T(\mu\mathbf{I}+\sum_{i,j}\mathbf{R}_{i,j}^T\mathbf{R}_{i,j} &= \mu\mathbf{y}^T\mathbf{I}+\sum_{i,j}(\alpha_{i,j}^T\mathbf{D}^T\mathbf{R}_{i,j}) \\
(\mu\mathbf{I}+\sum_{i,j}\mathbf{R}_{i,j}^T\mathbf{R}_{i,j})\mathbf{x} &= \mathbf{y}\mu\mathbf{I}+\sum_{i,j}\mathbf{R}_{i,j}^T\mathbf{D}\alpha_{i,j}) \\
\mathbf{x} &= (\mu\mathbf{I}+\sum\mathbf{R}_{i,j}^T\mathbf{R}_{i,j})^{-1}(\mathbf{y}\mu\mathbf{I}+\sum\mathbf{R}_{i,j}^T\mathbf{D}\alpha_{i,j})
\end{aligned}$$