Homework #4: Face Recognition: Scattered Wavelet Network II

Due May 3rd , 2016

Professor Davi Geiger.

The goal of this assignment is to develop a face recognition machine. It is based on the work of Bruna and Mallat, 2012. The idea is to build a convolution-like Network, with various layers (typically 2 or 3 layers). At each layer an “image-like” is convolved with a low pass filter ϕ_σ (a blur-like filter at scale σ) and down sampled (reducing the size). We refer to “image-like” to structures that have the same pixels as the image and are non-negative quantities at each pixel. The “image-like” vary from layer to layer via a prescribed procedure as we describe next.

Let us develop and test these ideas on face recognition.

Faces will have size 92 x 112 pixels. Let us resize them to 96 x 96 pixels, using a resize function in python (this function basically smooth the image and resample it).

Morlet Wavelets: You have done all of this already. I am suggesting to use just Scales $\boldsymbol{\sigma} = 4, 8$ Angles, $\boldsymbol{\theta} = 0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}$

SCATTERED WAVELET NETWORK “SCHEME”

	Layer 0	**Layer 1**	**Layer 2**
Wavelet Modulus	$\boldsymbol{I(u)}$	$\left\lvert\boldsymbol{\psi_{\lambda_i} * I(u)}\right\rvert$ $\boldsymbol{i = 1, \dots, 8;}$	$\left\lvert\boldsymbol{\psi_{\lambda_j} *} \left\lvert\boldsymbol{\psi_{\lambda_i} * I(u)}\right\rvert\right\rvert$ $\boldsymbol{i = 1, \dots, 8;\ j = 1, \dots, 4}$
GaussianConvolution	$\boldsymbol{G_{\sigma=8} * I(u)}$	$\boldsymbol{G_{\sigma=8} *} \left\lvert\boldsymbol{\psi_{\lambda_i} * I(u)}\right\rvert$	$\boldsymbol{G_{\sigma=8} *} \left\lvert\boldsymbol{\psi_{\lambda_j} *} \left\lvert\boldsymbol{\psi_{\lambda_i} * I(u)}\right\rvert\right\rvert$
Downsampling			
OUTPUT:	$\boldsymbol{SI^0_{k=24}(u)}$	$\boldsymbol{SI^1_{k=24}}\left(\left\lvert\boldsymbol{\psi_{\lambda_i} * I(u)}\right\rvert\right)$	$\boldsymbol{SI^2_{k=24}}\left(\left\lvert\boldsymbol{\psi_{\lambda_j} *} \left\lvert\boldsymbol{\psi_{\lambda_i} * I(u)}\right\rvert\right\rvert\right)$

MORE PRECISELY

PROBLEM 2: FACE TRAINING AND RECOGNITION

a. **PCA to improve and simplify representation**

Our “first approximation” represented a class of all N images of faces by their average value, i.e.,

$$\overrightarrow{AS}_F = \frac{1}{N}\sum_{k=1}^{N} \vec{S}(I_k)$$

Consider a matrix of all vectors $\vec{X}(I_k) = \vec{S}(I_k) - \overrightarrow{AS}_F$ stacked together

$$\boldsymbol{X} = \begin{pmatrix} \vec{X}(I_1) \\ \vec{X}(I_2) \\ \dots \\ \vec{X}(I_n) \end{pmatrix}$$

So X is an N x144 matrix, where N is the number of face images. Then the 144 x 144 covariance matrix $\boldsymbol{C}$ is written as

$$\boldsymbol{C} = \frac{1}{N-1}\boldsymbol{X}^{\boldsymbol{T}}\,\boldsymbol{X}$$

The entry values of $\boldsymbol{C}$ (the covariance associated to the N image face vectors $\vec{S}(I_k)$ and average face $\overrightarrow{AS}_F$) is written as

$$\boldsymbol{C}_{mn} = \frac{1}{N-1}\sum_{k=1}^{N}(S_m(I_k) - AS_m)(S_n(I_k) - AS_n)$$

Where m,n=1, …, 144, and AS_m is the m-th component of $\overrightarrow{AS}_F$ Likewise, $S_m(I_k)$ is the m-th component of $\vec{S}(I_k)$. From the covariance matrix we can compute the principal components as the normalized eigenvectors of it, say $\{\boldsymbol{e_j}; j = 1, \dots, 144\}$ are all the normalized and orthogonal eigenvectors of $\boldsymbol{C}$ and $\{\alpha_j; j = 1, \dots 144\}$ the corresponding eigenvalues**,** i.e.,

$$\boldsymbol{C}\ \boldsymbol{e_j} = \alpha_j\,\boldsymbol{e_j}$$

For the covariance matrix the eigenvectors are also orthogonal. The eigenvalues capture the variance of the data along the eigenvectors. The SVD decomposition for the covariance matrix allow us to write $\boldsymbol{C}$ as

$$\boldsymbol{C} = \boldsymbol{V}\,\boldsymbol{\alpha}\,\boldsymbol{V^T} = \begin{pmatrix}\boldsymbol{e_1} & \boldsymbol{e_2} & \dots\ \boldsymbol{e_j} & \dots.\ \boldsymbol{e_{144}}\end{pmatrix}\begin{pmatrix} \alpha_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 & \dots & 0 \\ & \dots & \dots & \dots & \dots & \\ 0 & 0 & \dots & \alpha_j & \dots & 0 \\ & \dots & \dots & \dots & \dots & \\ 0 & 0 & \dots & 0 & \dots & \alpha_{144} \end{pmatrix}\begin{pmatrix} \boldsymbol{e_1} \\ \boldsymbol{e_2} \\ \dots \\ \boldsymbol{e_j} \\ \dots. \\ \boldsymbol{e_{144}} \end{pmatrix}$$

$\boldsymbol{V}$ is a matrix of eigenvectors (each column is an eigenvector $\boldsymbol{e_j}$) and $\boldsymbol{\alpha}$ a diagonal matrix with the eigenvalues α_j in the decreasing order on the diagonal. The eigenvectors are the *principal axes* or *principal directions* of the data. Projections of the data on the *principal axes* are called principal components or *PC scores.* The *j-th* principal component is given by the *j-th* column of

.

A new data representation is then created, $\boldsymbol{Y} = \boldsymbol{X}\,\boldsymbol{V}$, replacing $\boldsymbol{X}$. The coordinate of the *k-th* data point in the new PC space are given ty the *k-th* row of $\boldsymbol{Y}$. Writing component by component

$$\vec{X}(I_k) \xrightarrow{yields} \vec{Y}(I_k) = \left(\vec{X}(I_k).e_1, \vec{X}(I_k).e_2, \dots, \vec{X}(I_k).e_j, \dots, \vec{X}(I_k).e_{144}\right) = (y_1^k, y_2^k, \dots, y_{144}^k)$$

Since the eigenvectors are orthogonal, this also implies that we recover the original representation via

$$\begin{aligned}\vec{X}(I_k) &= Y_1(I_k)\,\boldsymbol{e_1} + Y_2(I_k)\,\boldsymbol{e_2} + \dots + Y_j(I_k)\,\boldsymbol{e_j} + \dots + Y_{144}(I_k)\,\boldsymbol{e_{144}} \\ &= y_1^k\,\boldsymbol{e_1} + y_2^k\,\boldsymbol{e_2} + \dots + y_j^k\,\boldsymbol{e_j} + \dots + y_{144}^k\,\boldsymbol{e_{144}}\end{aligned}$$

b. Reducing the representation dimensionality to L << 144

Up to now, the vectors $\vec{Y}(I_k)$ have the same dimensionality as $\vec{X}(I_k)$. If the small eigenvalues are near zero we can reduce the representation of $\boldsymbol{C}$ as follows.

$$\boldsymbol{C} = \left(\boldsymbol{e_1}\ \ \boldsymbol{e_2}\ \ \dots\ \boldsymbol{e_j}\ \ \dots.\ \boldsymbol{e_{144}}\right) \begin{pmatrix} \alpha_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & & \\ 0 & 0 & \dots & \alpha_L & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 \dots & 0 \\ \dots & \dots & \dots & \dots & & \\ 0 & 0 & \dots & 0 & 0 \dots & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{e_1} \\ \boldsymbol{e_2} \\ \dots \\ \boldsymbol{e_j} \\ \dots. \\ \boldsymbol{e_{144}} \end{pmatrix}$$

$$\approx \left(\boldsymbol{e_1}\ \ \boldsymbol{e_2}\ \ \dots\ \boldsymbol{e_L}\ \ \boldsymbol{0}\ \dots.\ \boldsymbol{0}\right) \begin{pmatrix} \alpha_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & & \\ 0 & 0 & \dots & \alpha_L & \dots & 0 \\ 0 & 0 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & & \\ 0 & 0 & \dots & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{e_1} \\ \boldsymbol{e_2} \\ \dots \\ \boldsymbol{e_L} \\ \boldsymbol{0} \\ \dots. \\ \boldsymbol{0} \end{pmatrix}$$

$$= \left(\boldsymbol{e_1}\ \ \boldsymbol{e_2}\ \ \dots\ \boldsymbol{e_L}\ \right) \begin{pmatrix} \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \alpha_L \end{pmatrix} \begin{pmatrix} \boldsymbol{e_1} \\ \boldsymbol{e_2} \\ \dots \\ \boldsymbol{e_L} \end{pmatrix} = \boldsymbol{W}\,\boldsymbol{\alpha^L}\,\boldsymbol{W^T}$$

where $\boldsymbol{W} = (e_1, \dots, e_L)$ and each column of **W** represents one eigenvector. It is a reduction of the matrix **V**. The matrix **W** is of size **144** x **L**. So this representation $\boldsymbol{C} \approx \boldsymbol{W}\,\boldsymbol{\alpha^L}\,\boldsymbol{W^T}$ consider the eigenvectors associated to the higher eigenvalues, the top **L** eigenvalues and eigenvectors, since if the variance is very small we can neglect these directions. This matrix provides an approximation the Covariance matrix

$$\boldsymbol{C} \; = \boldsymbol{V}\, \boldsymbol{\alpha}\, \boldsymbol{V}^{\boldsymbol{T}} \approx \; \boldsymbol{W}\, \boldsymbol{\alpha}_{\boldsymbol{L}}\, \boldsymbol{W}^{\boldsymbol{T}}$$

The new vector representation is reduced to dimension **L**, namely

$$\vec{Y}(I_k) = \left(\vec{X}(I_k).\, e_1, \vec{X}(I_k).\, e_2, \ldots, \vec{X}(I_k).\, e_L\right) = (y_1^k, y_2^k, \ldots, y_L^k)$$

and in matrix form we can write

$$\boldsymbol{Y} = \boldsymbol{X}\,\boldsymbol{W}$$

Thus, we now approximately recover the original $\vec{X}(I_k)$ as

$$\vec{X}(I_k) \approx Y_1(I_k)\, \boldsymbol{e_1} + Y_2(I_k)\, \boldsymbol{e_2} + \cdots + Y_L(I_k)\, \boldsymbol{e_L} = y_1^k\, \boldsymbol{e_1} + y_2^k\, \boldsymbol{e_2} + \cdots + y_L^k\, \boldsymbol{e_L}$$

c. Using the reduced representation

Each new image I can then be processed by the scattered network. First we produce a vector $\vec{S}(I)$. Then, for each vector $\vec{S}(I)$ we remove the average vector $\overrightarrow{AS}_F$ i.e., create $\vec{X}(I_k) = \vec{S}(I_k) - \overrightarrow{AS}_F$Then, we transform $\vec{X}(I_k)$ to the new basis given by the matrix **W**, i.e.,

$$\vec{Y}(I_k) = \boldsymbol{W}^T \;\; \vec{X}(I_k) = \begin{pmatrix} e_1 \\ \cdots \\ e_L \end{pmatrix} \vec{X}(I_k)$$

The vector $\vec{Y}(I_k)$ has dimensions L and each component is the dot product of the vector $\vec{X}(I_k)$ with the PCA eigenvector associated to that component.

Evaluate if indeed $\vec{Y}(I_k)$ describes well the vector $\vec{X}(I_k)$, by computing the error

$$\epsilon^2 = \left[\vec{X}(I_k) - (Y_1(I_k)\, \boldsymbol{e_1} + Y_2(I_k)\, \boldsymbol{e_2} + \cdots + Y_L(I_k)\, \boldsymbol{e_L})\right]^2$$

If this error is small (compared to some threshold), then we also expect the following distance

$$d_F^2 = Y^T(I_k) \begin{bmatrix} \frac{1}{\alpha_1} & \ldots\; 0 & \ldots.\; 0 \\ & \ldots. & \\ 0 \ldots & \frac{1}{\alpha_j} & \ldots\; 0 \\ 0 \ldots & 0 \;\ldots & \frac{1}{\alpha_L} \end{bmatrix} Y(I_k)$$

to also be small, and so if it is below some threshold T, we accept as a face.

How to choose L ? one may simply keep only the top L=20 eigenvectors. Or threshold the eigenvalues that are below a quarter of the third largest eigenvalue (so that the first and second eigenvalue don't make such decision). So if $\alpha_j > 0.25\, \alpha_3$ then e_j is included.