Nonconvex Optimization

1 Structured matrix factorization

1.1 Bilinear models

Bilinear low-rank models of the form

$$y[i,j] \approx \sum_{l=1}^{r} a_l[i] b_l[j], \quad 1 \leq i \leq m,\ 1 \leq j \leq n, \tag{1}$$

or, in matrix form,

$$Y \approx AB, \qquad A \in \mathbb{R}^{m \times r}, \quad B \in \mathbb{R}^{r \times n}, \tag{2}$$

are useful in collaborative filtering, as discussed in the lecture notes on the SVD, and multiple other applications. The two model parameters $a[i]$ and $b[j]$ are combined multiplicatively. As a result, if we fit these parameters using observed data by minimizing a least-squares cost function, of the form

$$\min_{A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{r \times n}} ||Y - A\,B||_{\mathrm{F}} \tag{3}$$

the cost function is not convex. In addition, there is an identifiability issue, for any invertible $C \in \mathbb{R}^{r \times r}$ (A, B) and $(AC, C^{-1}B)$ yield the same model. This suggests fitting the model under the constraint that one of the factors has normalized columns,

$$\min_{A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{r \times n}} ||Y - A\,B||_{\mathrm{F}} \qquad \text{subject to} \quad \left|\left|\tilde{A}_i\right|\right|_2 = 1, \qquad 1 \leq i \leq r, \tag{4}$$

The feasibility set is nonconvex. Figure 1 illustrates this for the function

$$f\,(a, b) := (1 - ab)^2\,. \tag{5}$$

Despite the nonconvexity of the cost function and the feasibility set, in the lecture notes on the SVD we showed that the truncated SVD provides optimal factors: if USV^T is the SVD of Y then $A^* := U_{:,1:r}$ and $B := S_{1:r,1:r}V^T_{:,1:r}$ is a solution to Problem (4).

1.2 Nonnegative matrix factorization

The truncated SVD yields factors with entries that can be negative. In some applications, nonnegative factors have a more intuitive interpretation (we provide two examples below). Nonnegative

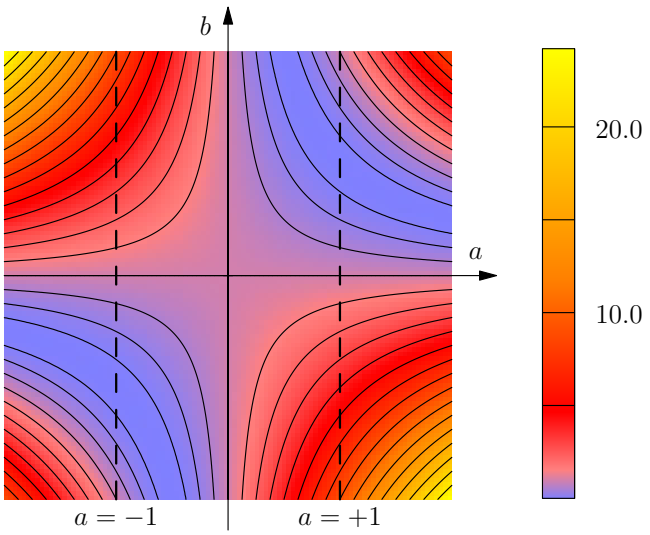

Figure 1: Heat map and contour map for the function $(1 - ab)^2$. The dashed line correspond to the set $|a| = 1$.

matrix factorization (NMF) is a method that fits a low-rank bilinear model where the factors are constrained to be nonnegative. The corresponding optimization problem is of the form

$$\begin{aligned} &\text{minimize} && \left|\left|X - \tilde{A}\,\tilde{B}\right|\right|_{\mathrm{F}}^2 && (6)\\ &\text{subject to} && \tilde{A}_{i,j} \geq 0, && (7)\\ & && \tilde{B}_{i,j} \geq 0, \qquad \text{for all } i,j && (8) \end{aligned}$$

where $\tilde{A} \in \mathbb{R}^{d\times r}$ and $\tilde{B} \in \mathbb{R}^{r\times n}$ for a fixed r. This is a nonconvex problem, which can no longer be solved using the SVD. Several methods to compute local optima have been suggested in the literature, as well as alternative cost functions to replace the Frobenius norm. We refer interested readers to [3].

To compare the results between models obtained via a truncated SVD and NMF, we fit both low-rank models to the Olivetti Faces data set[1]. Figure 2 shows the columns of the left matrix A, which we interpret as *atoms* that can be combined to form the faces. Due to the nonnegative constraint, the atoms resemble portions of faces (the black areas have values close to zero) which capture features such as the eyebrows, the nose, the eyes, etc. In contrast, the truncated SVD produces atoms with negative entries that cancel each other and are more difficult to interpret.

Example 1.1 (Topic modeling)**.** Topic modeling aims to learn the thematic structure of a text corpus automatically. We illustrate this application with a simple example. We take six newspaper articles and compute the frequency of a list of words in each of them. Our final goal is to separate the words into different clusters that hopefully correspond to different topics. The following matrix contains the counts for each word and article. Each entry contains the number of times that the

[1]Available at http://www.cs.nyu.edu/~roweis/data.html

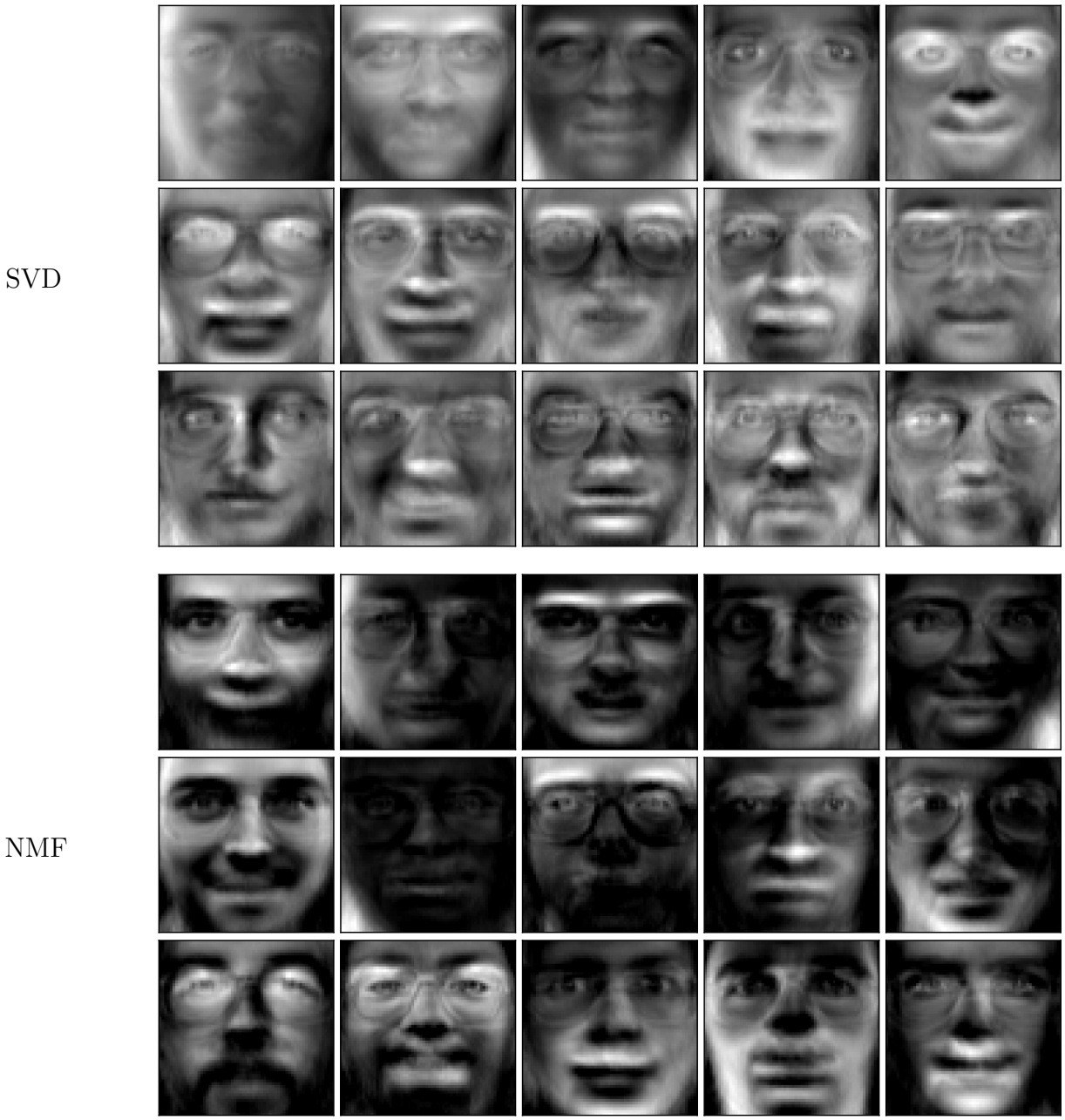

Figure 2: Face atoms obtained by a truncated SVD and by nonnegative matrix factorization (NMF) for the Olivetti Faces dataset. NMF avoids cancellations and results in more interpretable atoms.

word corresponding to column j is mentioned in the article corresponding to row i.

$$M := \begin{array}{c} \begin{array}{ccccccccc} \text{singer} & \text{GDP} & \text{senate} & \text{election} & \text{vote} & \text{stock} & \text{bass} & \text{market} & \text{band} \end{array} \\ \begin{pmatrix} 6 & 1 & 1 & 0 & 0 & 1 & 9 & 0 & 8 \\ 1 & 0 & 9 & 5 & 8 & 1 & 0 & 1 & 0 \\ 8 & 1 & 0 & 1 & 0 & 0 & 9 & 1 & 7 \\ 0 & 7 & 1 & 0 & 0 & 9 & 1 & 7 & 0 \\ 0 & 5 & 6 & 7 & 5 & 6 & 0 & 7 & 2 \\ 1 & 0 & 8 & 5 & 9 & 2 & 0 & 0 & 1 \end{pmatrix} \end{array} \begin{array}{c} \textit{Articles} \\ \text{a} \\ \text{b} \\ \text{c} \\ \text{d} \\ \text{e} \\ \text{f} \end{array}$$

Computing the singular-value decomposition of the matrix– after subtracting the mean of each entry– we determine that the matrix is approximately low rank

$$M = USV^T = U \begin{bmatrix} 23.64 & 0 & 0 & 0 & & \\ 0 & 18.82 & 0 & 0 & 0 & 0 \\ 0 & 0 & 14.23 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3.63 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.03 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.36 \end{bmatrix} V^T \tag{9}$$

Unfortunately the singular vectors do not have an intuitive interpretation. In particular, they do not allow to cluster the words

$$\begin{array}{rcccccccc} & & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} \\ U_1 & = & (-0.24 & -0.47 & -0.24 & -0.32 & -0.58 & -0.47) \\ U_2 & = & (\ 0.64 & -0.23 & 0.67 & -0.03 & -0.18 & -0.21) \\ U_3 & = & (-0.08 & -0.39 & -0.08 & 0.77 & 0.28 & -0.40) \end{array} \tag{10}$$

or the articles

$$\begin{array}{rcccccccccc} & & \text{singer} & \text{GDP} & \text{senate} & \text{election} & \text{vote} & \text{stock} & \text{bass} & \text{market} & \text{band} \\ V_1 & = & (-0.18 & -0.24 & -0.51 & -0.38 & -0.46 & -0.34 & -0.2 & -0.3 & -0.22) \\ V_2 & = & (\ 0.47 & 0.01 & -0.22 & -0.15 & -0.25 & -0.07 & 0.63 & -0.05 & 0.49\ ) \\ V_3 & = & (-0.13 & 0.47 & -0.3 & -0.14 & -0.37 & 0.52 & -0.04 & 0.49 & -0.07) \end{array} \tag{11}$$

A problem here is that the singular vectors have negative entries that are difficult to interpret. In the case of rating prediction, negative ratings mean that a person does not like a movie. In contrast articles either are about a topic or they are not: it makes sense to add atoms corresponding to different topics to approximate the word count of a document but not to subtract them. Following this intuition, we apply NMF to obtain two matrices $W \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times n}$ such that

$$M \approx WH, \quad W_{i,j} \geq 0, \qquad 1 \leq i \leq m,\ 1 \leq j \leq k, \tag{12}$$

$$H_{i,j} \geq 0, \qquad 1 \leq i \leq k,\ 1 \leq i \leq n. \tag{13}$$

In our example, we set $k = 3$. H_1, H_2 and H_3 can be interpreted as word-count atoms, but also

as coefficients that weight the contribution of W_1, W_2 and W_3.

$$\begin{array}{rcccccccccc} & & \text{singer} & \text{GDP} & \text{senate} & \text{election} & \text{vote} & \text{stock} & \text{bass} & \text{market} & \text{band} \\ H_1 & = & (\ 0.34 & 0 & 3.73 & 2.54 & 3.67 & 0.52 & 0 & 0.35 & 0.35\ ) \\ H_2 & = & (\ 0 & 2.21 & 0.21 & 0.45 & 0 & 2.64 & 0.21 & 2.43 & 0.22\ ) \\ H_3 & = & (\ 3.22 & 0.37 & 0.19 & 0.2 & 0 & 0.12 & 4.13 & 0.13 & 3.43\ ) \end{array} \tag{14}$$

The latter interpretation allows to cluster the words into topics. The first topic corresponds to the entries that are not zero (or very small) in H_1: senate, election and vote. The second corresponds to H_2: GDP, stock and market. The third corresponds to H_3: singer, bass and band.

The entries of W allow to assign the topics to articles. b, e and f are about politics (topic 1), d and e about economics (topic 3) and a and c about music (topic 3)

$$\begin{array}{rcccccccc} & & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} \\ W_1 & = & (0.03 & 2.23 & 0 & 0 & 1.59 & 2.24) \\ W_2 & = & (\ 0.1 & 0 & 0.08 & 3.13 & 2.32 & 0\ ) \\ W_3 & = & (2.13 & 0 & 2.22 & 0 & 0 & 0.03) \end{array} \tag{15}$$

Finally, we check that the factorization provides a good fit to the data. The product WH is equal to

$$\begin{array}{ccccccccccc} \text{singer} & & \text{GDP} & \text{senate} & \text{election} & \text{vote} & \text{stock} & \text{bass} & \text{market} & \text{band} & \textit{Art.} \\ \begin{pmatrix} 6.89\ (6) \\ 0.75\ (1) \\ 7.14\ (8) \\ 0\ (0) \\ 0.53\ (0) \\ 0.86\ (1) \end{pmatrix} & & \begin{pmatrix} 1.01\ (1) \\ 0\ (0) \\ 0.99\ (1) \\ 7\ (6.91) \\ 5.12\ (5) \\ 0.01\ (0) \end{pmatrix} & \begin{pmatrix} 0.53\ (1) \\ 8.32\ (9) \\ 0.44\ (0) \\ 0.67\ (1) \\ 6.45\ (6) \\ 8.36\ (8) \end{pmatrix} & \begin{pmatrix} 0.54\ (0) \\ 5.66\ (5) \\ 0.47\ (1) \\ 1.41\ (0) \\ 5.09\ (7) \\ 5.69\ (5) \end{pmatrix} & \begin{pmatrix} 0.10\ (0) \\ 8.18\ (8) \\ 0\ (0) \\ 0\ (0) \\ 5.85\ (5) \\ 8.22\ (9) \end{pmatrix} & \begin{pmatrix} 0.53\ (1) \\ 1.15\ (1) \\ 0.47\ (0) \\ 8.28\ (9) \\ 6.97\ (6) \\ 1.16\ (2) \end{pmatrix} & \begin{pmatrix} 8.83\ (9) \\ 0\ (0) \\ 9.16\ (9) \\ 0.65\ (1) \\ 0.48\ (0) \\ 0.14\ (0) \end{pmatrix} & \begin{pmatrix} 0.53\ (0) \\ 0.78\ (1) \\ 0.48\ (1) \\ 7.60\ (7) \\ 6.19\ (7) \\ 0.79\ (0) \end{pmatrix} & \begin{pmatrix} 7.36\ (8) \\ 0.78\ (0) \\ 7.62\ (7) \\ 0.69\ (0) \\ 1.07\ (2) \\ 0.9\ (1) \end{pmatrix} & \begin{matrix} \text{a} \\ \text{b} \\ \text{c} \\ \text{d} \\ \text{e} \\ \text{f} \end{matrix} \end{array}$$

For ease of comparison the entries of M are shown in brackets. △

1.3 Sparse coding

In the lecture notes on frequency representations we studied translation-invariant signals. We explained that performing PCA on this class of signals, which models images, video, speech, etc. tends to yield sinusoidal principal directions, due to the approximately-convolutional structure of the covariance matrix. This means that one can compress such signals effectively using a predefined frequency representation, but it also means that we cannot uncover specific structure using PCA. Sparse coding is an alternative that does not require the basis functions (the columns of the left factor) to be orthogonal, but instead forces them to reproduce the data parsimoniously. This is

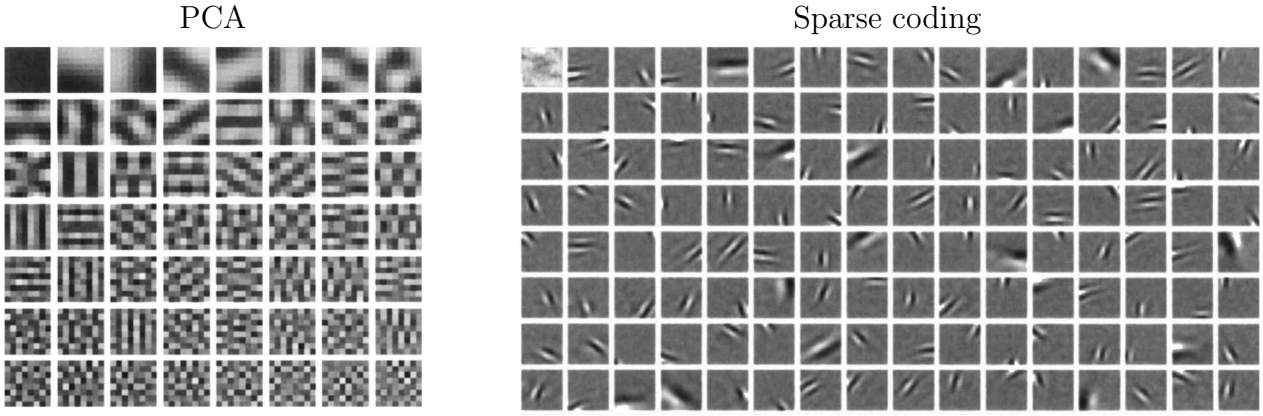

Figure 3: Basis functions learnt by PCA (left) and sparse coding (right) from patches extracted from natural images. The image is borrowed from [5].

achieved by incorporating a sparsity constraint on the coefficients or *codes* of the right factor:

$$\text{minimize} \quad \left|\left|X - \tilde{A}\,\tilde{B}\right|\right|_2^2 + \lambda \sum_{i=1}^{r} \left|\left|\tilde{B}_i\right|\right|_1 \tag{16}$$

$$\text{subject to} \quad \left|\left|\tilde{A}_i\right|\right|_2 = 1, \qquad 1 \leq i \leq r. \tag{17}$$

The technique is called dictionary learning or sparse coding in the literature. As in the case of NMF, the factorization is fit by optimizing the cost function until a local minimum is reached [4]. It was originally proposed in neuroscience, as an explanation of the Gabor-like receptive fields of neurons in the visual cortex [5]. Figure 3, borrowed from [5] shows the basis functions learned by sparse coding. They look like edge detectors. Edges are crucial features in images that are ignored by PCA.

2 Deep learning for image denoising

In the lecture notes on Fourier we described how to denoise signals using a linear translation-invariant filter. In the lecture notes on signal representations we showed that thresholding-based techniques produce better results. These techniques apply a linear transformation to the signal, which yields a sparse representation that is thresholded to remove the noise. In this section we describe an approach to learn more general nonlinear denoisers based on deep learning.

Convolutional neural networks (CNNs) have been notoriously successful in image classification [2]. They are translation-invariant nonlinear models implemented by a series of convolutions with different filters interleaved by pointwise nonlinearities. This yields a model with a large number of parameters, which are learned by minimizing a cost function related to the task of interest. In the case of denoising, the cost function is of the form

$$\min_{W_1,\ldots,W_L} \sum_{j=1}^{n} \left|\left|\vec{x}^{[j]} - W_L \rho\left(W_{L-1}\rho\left(\ldots W_2 \rho\left(W_1 \vec{y}^{[j]}\right)\right)\right)\right|\right|_2^2, \tag{18}$$

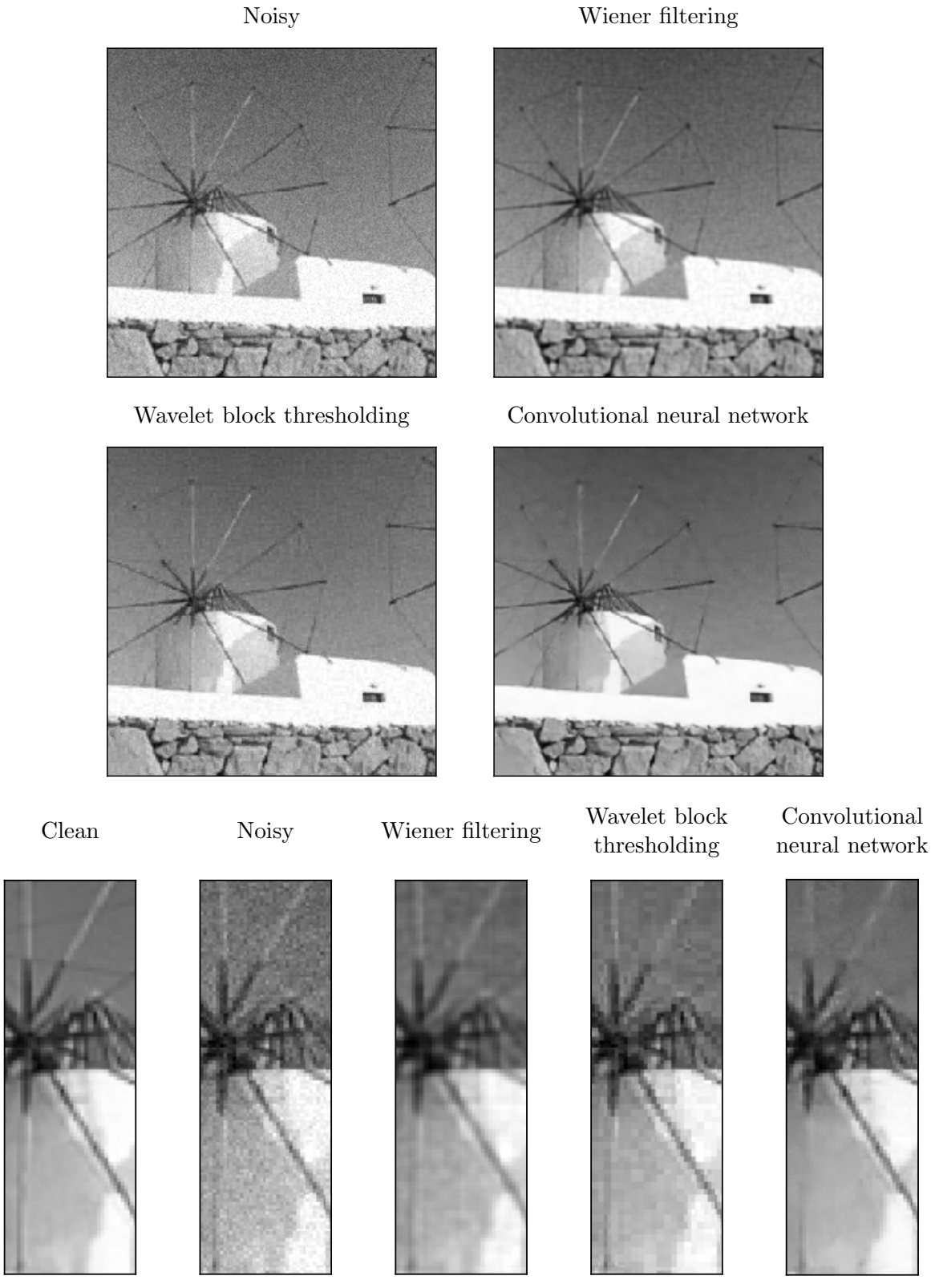

Figure 4: Denoising results for Wiener filtering, wavelet block thresholding and a convolutional neural network for iid Gaussian noise with standard deviation $\sigma := 0.04$.

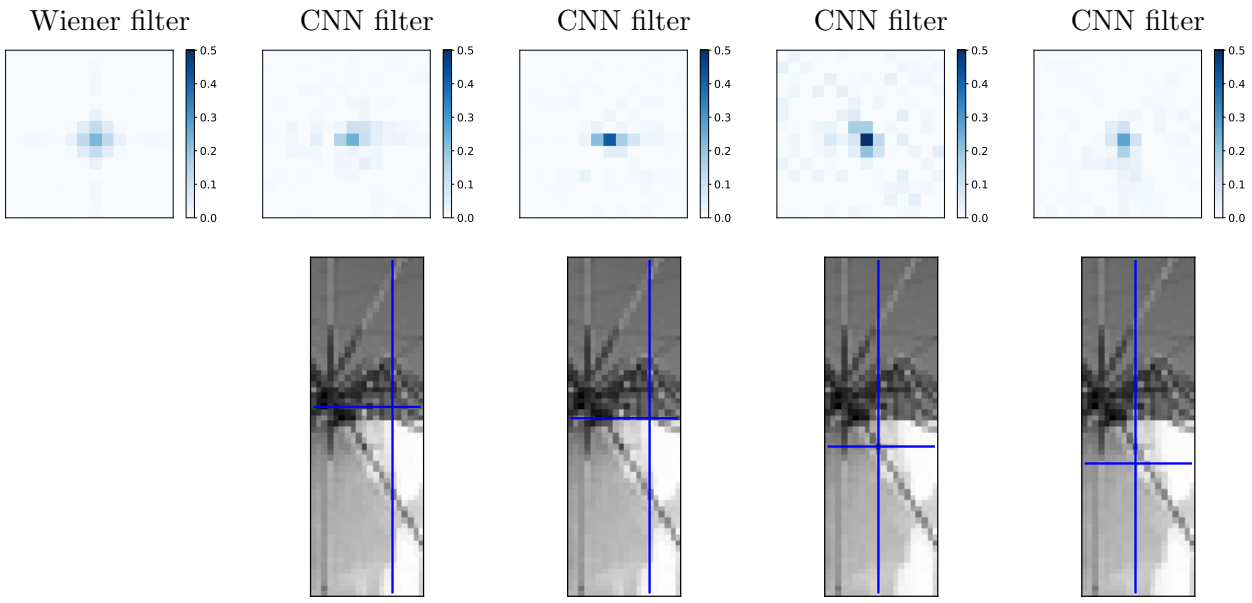

Figure 5: The top row shows a Wiener filter together with rows of the Jacobian of a CNN, which can be interpreted as nonlinear filters tied to the particular image. The second row shows the location of the pixels corresponding to each of the rows of the Jacobian.

where $(\vec{x}^{[1]}, \vec{y}^{[1]}), \ldots, (\vec{x}^{[n]}, \vec{y}^{[n]})$ is a training set of n pairs of clean and noisy signals of dimension N. The nonlinearity ρ is chosen to be a pointwise rectifier linear unit, for any $\vec{v} \in \mathbb{R}^N$ $\rho(\vec{v})[i] := \max\{0, \vec{v}[i]\}$ for $1 \leq i \leq N$. The matrices $W_1, \ldots, W_L$ contain convolutional filters that are applied to the output of the previous layer after rectification. The number of parameters in the model is very large (around half a million), and therefore requires a large training set of image patches (note however that these can be extracted from a set of clean images of just a few hundreds). To make this tractable, stochastic gradient descent (see the notes on convex optimization) is applied to solve the minimization problem. For more details we refer to [6] (see also [1] for an excellent introduction to deep learning).

The results of deep-learning based denoising are compared in Figure 4 to the Wiener filtering technique described in the notes on Fourier representations, and to the block-thresholding approach from the notes on signal representations. The CNN produces significantly better results, removing more noise while preserving image structure very effectively. In order to understand more intuitively what the DNCNN is doing, we can consider the Jacobian of the function $W_L \rho\left(W_{L-1} \rho\left(\ldots W_2 \rho\left(W_1 \vec{y}\right)\right)\right)$ for a fixed image $\vec{y}$. The Jacobian corresponds to a matrix such that $J\vec{y} = W_L \rho\left(W_{L-1} \rho\left(\ldots W_2 \rho\left(W_1 \vec{y}\right)\right)\right)$. The rows of the matrix can be interpreted as denoising filters adapted to the specific structure of the image. Figure 5 shows some of these filters for different locations in an image. The filters adapt to edges and other features.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[2] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[3] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[4] J. Mairal, F. Bach, J. Ponce, et al. Sparse modeling for image and vision processing. *Foundations and Trends® in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

[5] B. A. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[6] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.