Learning a selectivity–invariance–selectivity feature extraction architecture for images

Michael U. Gutmann and Aapo Hyvärinen
Dept Computer Science, Dept Mathematics and Statistics, HIIT, University of Helsinki
michael.gutmann@helsinki.fi

Abstract

Selectivity and invariance are thought to be important ingredients in biological or artificial visual systems. A fundamental problem is, however, to know what the visual system should be selective to and what to be invariant to. Building a statistical model of images, we learn here a three-layer feature extraction system where the selectivity and invariance emerges from the properties of the images.

1. Introduction

Selectivity and invariance are two fundamental requirements for any feature extraction system. The system should detect specific patterns while being invariant, or tolerant, to possible variations.

Our visual system, for instance, is highly selective in recognizing faces. At the same time, however, it is tolerant to all kinds of variations. We recognize a familiar face when seen under different illuminations, when seen from the front or the side, or when it is partially covered with clothing. Selectivity and tolerance are thought to be relevant ingredients in biological and computer vision, see for example [4, 8, 1] and the references within. One interesting line of research considers hierarchical models that consist of canonical elements which perform elementary selectivity and tolerance (invariance) computations, see for example [7, 5]. A fundamental problem is, however, to know what the canonical elements should be selective and invariant to.

In this paper, we address this issue by learning from natural images what kind of features to be selective to and what kind of deviations to tolerate. We build a probabilistic model which consists of three feature-extraction layers. After learning, the first layer emphasizes selectivity, the second invariance, and the third one again selectivity. Moreover, learning increases the sparsity of the feature outputs. The learning itself is performed with an estimation method which guarantees consistent (converging) estimates [2].

We introduce the image data next before turning to the model in Section 3. Section 4 concludes the paper.

2. Image data and preprocessing

The modeling of natural images is often done with image patches. In this paper, we use instead the tiny images dataset [9], converted to gray scale. The data set consists of about eighty million images that show complete visual scenes downsampled to 32×32 pixels. Examples are shown in Figure 1.

Since we are interested in modeling spatial features, we removed the DC component from the images and normalized them to unit norm before the learning of the features. We compute the norm of the images after PCA-based whitening. Unlike the norm before whitening, this norm is not dominated by the low-frequency content of an image [3, Chapter 5]. Note that this normalization can be considered to provide a simple means to make the features invariant to different illumination conditions. After normalization, we reduced the dimensionality from $32 \cdot 32 = 1024$ to 200, which corresponds to low-pass filtering of the images. After dimension reduction, the images are elements in a 200 dimensional sphere.

Figure 1: Examples from the tiny images dataset.

3. Learning features in a three-layer model

We divide the learning of the three feature extraction layers into two phases: first, we estimate an intermediate model with two layers. Then, we learn the complete three-layer model.

3.1. Intermediate model

The intermediate model is the same as the two layer model in [2, Section 5.3] where we estimated it for natural image patches extracted from larger images. Unlike the image data which we use in this paper, the patches did not show complete visual scenes. We can thus expect some differences in the results.

In this intermediate model, the value of the log-pdf at an input image $\mathbf{x}$ is given by the overall activity of the second layer feature outputs $y_k^{(2)}$,

$$\ln p(\mathbf{x}) = \sum_{k=1}^{n_2} y_k^{(2)}(\mathbf{x}) + c, \tag{1}$$

where c is a scalar offset and $n_2 = 50$. The feature outputs are computed as follows. First, the input $\mathbf{x}$ is passed through a linear feature detection stage which gives the first-layer outputs $y_i^{(1)}$,

$$y_i^{(1)} = \mathbf{w}_i^{(1)T}\mathbf{x}, \quad i = 1 \dots n_1, \tag{2}$$

with $n_1 = 100$. The first-layer outputs are then rectified, passed through a second linear feature detection stage, and nonlinearly transformed to give the second-layer outputs $y_k^{(2)}$,

$$y_k^{(2)}(\mathbf{x}) = f_k\left(\sum_{i=1}^{n_1} w_{ki}^{(2)}(y_i^{(1)})^2\right), \quad k = 1 \dots n_2. \tag{3}$$

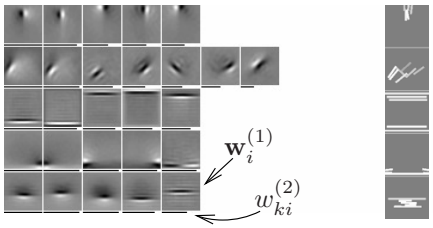
The nonlinearity f_k is like in [2, Section 5.3] given by

$$f_k(u) = f_{\text{th}}(\ln(u+1) + b_k), \tag{4}$$

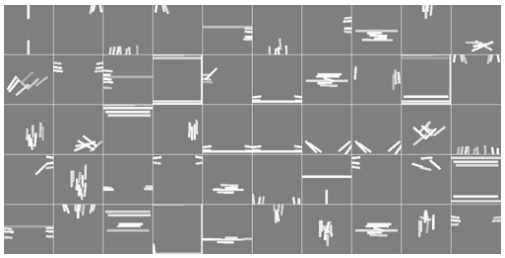
where $f_{\text{th}}(u) = 0.25\ln(\cosh(2u)) + 0.5u + 0.17$ is a smooth approximation of the thresholding function $\max(0, u)$. The term b_k sets the threshold. The parameters of the model are the first-layer feature detectors $\mathbf{w}_i^{(1)}$, the second-layer weights $w_{ki}^{(2)} \geq 0$, the thresholds b_k, and the scalar c which is needed to allow for proper normalization of the model.

The model in (1) is unnormalized. That is, it does not integrate to one except for the right value of c which we do, however, not know. This makes learning of the parameters by standard maximum likelihood estimation impossible. We use noise-contrastive estimation for the learning [2].[1]

[1] As noise distribution, we use the uniform distribution in the 200 dimensional sphere. We took ten times more noise than data.



(a) Subset of the features and their icons

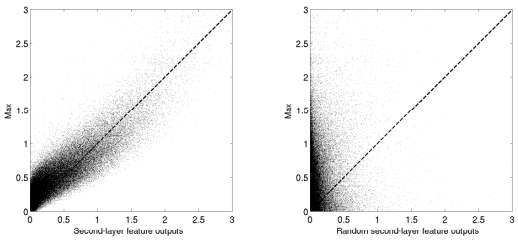


(b) All features shown as icons

Figure 2: Learned features of the second layer.

We visualize our results in the same way as in [2]. The first-layer features $\mathbf{w}_i^{(1)}$ are visualized by showing the image which yields the largest first-layer output $y_i^{(1)}$. After learning, the second-layer weights $w_{ki}^{(2)}$ are extremely sparse: 94.5% have values less than 10^{-6} while 5.1% are larger than 10. We can thus visualize each row of the $w_{ki}^{(2)}$ by showing the few $\mathbf{w}_i^{(1)}$ for which the weights are nonzero. This is done in Figure 2(a) for five randomly selected rows. The same figure shows on the right also a condensed visualization of the features by means of icons that we have created as in [2, Figure 12]. In Figure 2(b), we use the icons to show all the learned features of the first two layers.

The first layer is mostly sensitive to Gabor-like image features, and the second layer pools dominantly over similarly oriented or localized first-layer features. These results are similar to those obtained for image patches [2]. The pooling here is, however, less localized. The first layer implements a selectivity stage, with the Gabor-like image features being the preferred input of each $\mathbf{w}_i^{(1)}$. We show now that the second layer weights $w_{ki}^{(2)}$ can be interpreted to perform a max-like computation over the first-layer feature outputs $|y_i^{(1)}|$. Figure 3(a) shows a scatter plot between the outputs $y_k^{(2)}$ and the maximal value of $|y_i^{(1)}|$, taken over all i for which $w_{ki}^{(2)}$ is larger than 0.001. There is a clear correlation, and a clear difference to the baseline in Figure 3(b). We thus may consider the learned weights $w_{ki}^{(2)}$ as indices that select over which first-layer outputs to take the max operation. Hence, the first layer imple-



(a) With learning (b) Baseline

Figure 3: (a) For natural image input, we plot the second-layer outputs $y_k^{(2)}$ on the x-axis against $\max_{i:w_{ki}^{(2)}>0.001} |y_i^{(1)}|$ on the y-axis. The correlation coefficient is 0.81. (b) Instead of using the learned $w_{ki}^{(2)}$, we took a random matrix with positive elements with equal row sums as the learned matrix. This gives a correlation coefficient of 0.18.

ments a feature selection stage while the second layer corresponds to a feature invariance stage. Together with Figure 2(b), the invariance takes often the form of tolerance with respect to exact localization and orientation.

3.2. Complete three-layer model

We extend here the model in (1) by looking for features in the second layer outputs $y_k^{(2)}$. The features analyze the relationships between the different $y_k^{(2)}$. Note that in (1), only the average (DC component) of the $y_k^{(2)}$ enters into the computation of $\ln p$. That is, the relation between the different second-layer outputs does not matter. Here, we modify the model so that dependencies between the different second-layer outputs enter into the model. For that purpose, we remove the DC value $1/n_2 \sum_k y_k^{(2)}$ from the response vector $\mathbf{y}^{(2)} = (y_1^{(2)}, \dots y_{n_2}^{(2)})$, whiten it, and normalize its norm. We denote the whitened normalized vector by $\tilde{\mathbf{y}}^{(2)}$. Figure 2 shows that some of the $y_k^{(2)}$ are duplicates of each other. Hence, with the whitening, we are also performing dimension reduction from 50 to 46 dimensions. Finding the right amount of dimension reduction was straightforward since the eigenvalues of the covariance matrix of $\mathbf{y}^{(2)}$ (computed using natural image input) dropped abruptly from a level of 10^{-3} to 10^{-7}.

Keeping the computation in the first two layers, as specified in (2) and (3), fixed, the three-layer model is

$$\ln p(\mathbf{x}) = \sum_{j=1}^{n_3} y_j^{(3)} + c, \quad (5)$$

$$y_j^{(3)} = f_{\text{th}}\left(\mathbf{w}_j^{(3)T} \tilde{\mathbf{y}}^{(2)} + b_j^{(3)}\right), \quad (6)$$

where f_{th} is the same smooth approximation of $\max(0, u)$ as in (4). The parameters of the model are the third-layer features $\mathbf{w}_j^{(3)}$, the thresholds $b_j^{(3)}$, and the scalar c. We learned the parameters of the model when the number n_3 of third-layer features was 10 and 100, using noise-contrastive estimation as in the previous section.

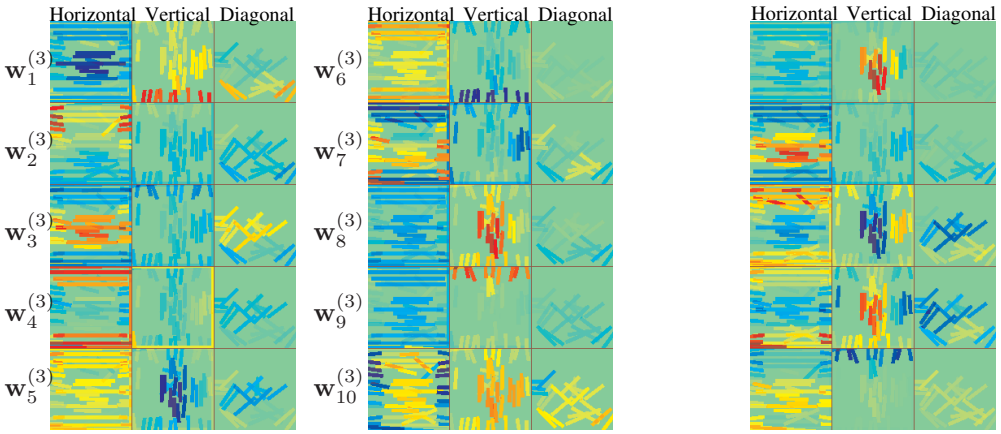
After learning, the thresholds $b_j^{(3)}$ were all negative (results not shown). The third layer implements thus another selectivity layer since a third-layer output $y_j^{(3)}$ is only nonzero if the inner product $\mathbf{w}_j^{(3)T} \tilde{\mathbf{y}}^{(2)}$ is larger than $|b_j^{(3)}|$.

In Figure 4(a) and (b), we show all the features for $n_3 = 10$ and a selection for $n_3 = 100$, respectively. Similarly to the visualization of the first-layer features, we visualize the third-layer features by showing the vector $\mathbf{y}^{(2)}$ which yields the largest output $\mathbf{w}_j^{(3)T} \tilde{\mathbf{y}}$. Visualization of the optimal $\mathbf{y}^{(2)}$ is not straightforward though. We chose to make use of the icons in Figures 2. Each icon represents a second-layer feature, and we weighted it proportionally to the k-th element of the optimal $\mathbf{y}^{(2)}$. In the colormap used, positively weighted icons appear reddish while negatively weighted icons appear in shades of blue. Green corresponds to a zero weight. For clarity of visualization, we separately show the weighted icons for the horizontally, vertically, and diagonally oriented second-layer features.

Figure 4 shows that activity of horizontally tuned second-layer features is often paired with inactivity of vertically tuned ones, and vice versa; see for example $\mathbf{w}_5^{(3)}$ and $\mathbf{w}_8^{(3)}$. Such a property is known as orientation inhibition. Some features also detect inactivity of co-oriented neighbors of activated features, see for example $\mathbf{w}_3^{(3)}$ and $\mathbf{w}_{10}^{(3)}$. This behavior is known as sharpening of orientation tuning and end-stopping. The properties of some third-layer features might be related to the fact that the tiny images are complete visual scenes where center and surround have distinct characteristics. For example, the third feature in (b) prefers activity on the top, bottom, and right side but none in the middle, while $\mathbf{w}_{10}^{(3)}$ prefers to have no horizontal activity on the sides.

In Figure 5, we show images which result in maximal and minimal values (activations) of selected $y_j^{(3)}$.[2] There is a good correspondence to the visualization of the features in Figure 4. Moreover, the images which activate each feature most all belong to a well defined category. More investigation is needed but this may suggest that the feature outputs could act as descriptors of the overall properties of the scene shown [6].

[2]The outputs were computed for 50000 tiny images.



(a) Complete set of third-layer features $\mathbf{w}_j^{(3)}$ for $n_3 = 10$

(b) Selection for $n_3 = 100$

Figure 4: Visualization of the learned third-layer features. A feature is tuned to detect activity of the second-layer features colored in red and inactivity of those colored in blue. See text body for further explanation on the visualization.

(a) Images for $\mathbf{w}_3^{(3)}$

(b) Images for $\mathbf{w}_8^{(3)}$

(c) Images for $\mathbf{w}_{10}^{(3)}$

Figure 5: Images with maximal and minimal activation of the features. Top: max activation. Bottom: min activation.

4. Conclusions

In this paper, we have learned a selectivity–invariance–selectivity architecture to extract features from tiny images. In the first layer, Gabor-like structures are detected. The second layer learned to compute a max-operation over the outputs of the first layer. In this way, an invariance to exact orientation or localization of the stimulus was learned from the data. The features on the third layer often detect activity of aligned first-layer features in combination with inactivity of their spatial neighbors, or inactivity of differently oriented features. Thus, the third layer learned enhanced selectivity to orientation and/or space.

While some of the features on the third-layer can be considered to reflect properties of complete visual scenes, they do not correspond to parts of objects. Increasing the number of features might lead to the emergence of such properties; increasing the number of layers might, however, also be necessary. We are hopeful that the approach in this paper can be extended to learn further selectivity and invariance layers.

References

[1] C. Cadieu and B. Olshausen. Learning Intermediate-Level Representations of Form and Motion from Natural Movies. *Neural Computation*, 24(4):827–866, 2012.

[2] M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361, 2012.

[3] A. Hyvärinen, J. Hurri, and P. Hoyer. *Natural Image Statistics*. Springer, 2009.

[4] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the Best Multi-Stage Architecture for Object Recognition? In *International Conference on Computer Vision (ICCV)*, 2009.

[5] M. Kouh and T. Poggio. A Canonical Neural Circuit for Cortical Nonlinear Operations. *Neural Computation*, 20(6):1427–1451, 2008.

[6] A. Oliva and A. Torralba. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[7] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature*, 2(11):1019, 1999.

[8] N. C. Rust and A. A. Stocker. Ambiguity and invariance: two fundamental challenges for visual processing. *Current Opinion in Neurobiology*, 20(3):382–388, 2010.

[9] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.