Using Statistics of Natural Images to Facilitate Automatic Receptive Field Analysis

Matthew Harrison
Division of Applied Mathematics
Brown University
Providence, RI 02912 USA
Matthew_Harrison@Brown.EDU

Stuart Geman
Division of Applied Mathematics
Brown University
Providence, RI 02912 USA
geman@dam.brown.edu

Elie Bienenstock
Division of Applied Mathematics
Brown University
Providence, RI 02912 USA
elie@dam.brown.edu

March 8, 2004

APPTS Report #04-2, Division of Applied Math, Brown University
http://www.dam.brown.edu/ptg/REPORTS/04-2.pdf

1 Introduction

A neuron can be conceptualized as performing a function on its inputs. This stimulus-response function characterizes "what the neuron does" and is the main object of interest for cellular-level neuroscience. Automated methods for discovering certain properties of stimulus-response functions have many advantages over traditional methods [5]. They can perform more exhaustive searches, use less-biased inputs and have tighter feedback loops for online, adaptive analysis. However, automated methods require many stimulus presentations, often many more than are feasible for a given physiology experiment. Finding ways to reduce the number of stimulus presentations is crucial for increasing the power and scope of these important techniques. We believe that one way to do this is to take advantage of the statistics of natural images.

Neurons in V1 appear to be tuned to the specific statistical properties of natural images. For example, a variety of techniques like sparse coding [13, 7], independent component analysis (ICA) [1, 8] and slow feature analysis (SFA) [17, 2], when applied to natural images, yield response properties strikingly similar to both simple and complex cells in V1. That neurons are tuned to natural inputs makes sense from both an evolutionary perspective and from a developmental perspective. We can use this knowledge to more efficiently probe the response properties of visual neurons.

A simple thought experiment partially illustrates our idea. Natural images can be significantly compressed using algorithms like JPEG without suffering any noticeable loss in quality. This suggests that neurons in the visual cortex ignore certain types of variation in images. If we want to somehow search for the stimulus that makes a neuron respond the strongest, then it seems reasonable to ignore these same types of variation. This constrains our search and makes it more efficient.

Here we focus on two particular types of analysis. The first is designed to identify the optimal stimuli for a neuron, that is, to find the maximizers of a neuron's stimulus-response function. The second attempts to approximate the entire stimulus-response function by using an appropriate parametric model, like linear or quadratic. We briefly describe each method and present some preliminary simulation results.

2 Stimulus Optimization

The goal is to find the input that maximizes the response of a given neuron using an adaptive, online search algorithm. Two basic methods are described in Földiák (2001; area V1, anesthetized monkey) [5] and Földiák et al. (2003; area STSa, awake monkey) [6]. In both cases they demonstrate the feasibility of these techniques in physiology experiments. We propose to increase the efficiency and thereby the utility of these methods. Here we will focus on the method in Földiák (2001).

2.1 Gradient ascent

The basic idea is to do gradient ascent ("hill-climbing") on the stimulus-response function. A stimulus $x(t)$ is presented on trial t and the neural response $y(t)$ is recorded. Now we update the stimulus by moving it in the direction of the response gradient. This gives a new stimulus $x(t+1)$, based on the previous, that should yield a higher response from the neuron. Repeatedly updating in this way allows us to climb the stimulus-response function until we find the optimal stimulus.

Of course, there are many caveats. The main problem is how to find the response gradient. We conceptualize the response y to the stimulus x as a noisy version of some ideal or mean response $f(x)$. The function f is the stimulus-response function of the neuron. We want to discover the gradient vector $\nabla f(x)$ of f at x. This can be done using a block of stimulus presentations, each of which is the original x plus noise. The gradient is (to a first approximation) proportional to the covariance of the noise and the response [5]. More details can be found in the Appendix.

The basic experimental design is thus a series of blocks of trials. Within each block the baseline stimulus is held constant and the animal is presented noisy versions of it. At the end of a block, the neural response gradient is estimated. Then the baseline stimulus is updated in the direction of the gradient before moving to the next block of trials. Because a block of trials are needed for each update of the stimulus, this method requires a large number of total trials for a given experiment. The computations can be carried out more or less instantaneously with a modern desktop computer and should add no further constraints on the time course of an experiment.

2.2 Dimensionality reduction

We can reduce the number of trials by reducing the dimensionality of the stimulus space. An arbitrary dimensionality reduction will strongly bias our search for the optimal stimulus. By using the properties of natural images to reduce the dimensionality in an intelligent way, however, we should be able to bias the search toward the optimum and not away from it.

One of the simplest types of dimensionality reduction is principal component analysis (PCA). Using a collection of natural image patches (16×16 pixels), we estimated the first 10 principal components. The gradient ascent method can be applied in this 10-dimensional subspace of principal components or in the original 256-dimensional pixel space. We compared the two methods in a simulation study.

In the first experiment we simulated the response of a V1 complex cell as a quadratic function of the pixel input, followed by a monotonic nonlinearity, followed by Poisson noise with this rate. That is, the mean of the Poisson observation was $f(x^T Ax+x^T B+C)$, where $f(z) = 10 \exp(z/2)/(1 + \exp(z/2))$. The parameters A, B and C of the quadratic were fit using slow feature analysis (SFA) trained on natural images. SFA produces cells with many of the properties of V1 complex cells including phase invariance, active inhibition and direction selectivity [2]. The form and parameters of f were chosen by hand.

Figure 1 shows three different full-dimensional searches for the optimal stimulus. This is the search that was used in Földiák (2001). Each graph shows the mean response of the neuron to the baseline image over 50 blocks. The response is normalized between 0 and 1 where 0 is the minimum possible mean response and 1 is the maximum possible mean response. (These values were obtained numerically using A, B and C.) The first graph uses 10 trials per block to estimate the response gradient. The second uses 25 and the third uses 50. In each case the search was started from the same random image patch. Beside each graph is the final estimate of the optimal stimulus.

For comparison, Figure 2 shows three different reduced-dimensional searches. The starting point and block structures are identical. Notice that the reduced-dimensional search performs better, especially when using only a few trials per block. Notice also that the three different estimates of the optimal response vary somewhat even though the response has been nearly maximized. This is because the complex cell shows some invariance.

The main quantities for comparison in Figures 1 and 2 are how quickly and how close the responses approach the maximal response. The images of the maximizing stimuli are somewhat misleading. Projecting the noisy stimuli for the full-dimensional search onto 10-dimensional PCA space will produce smooth, edge-like stimuli. In the case of 50 trials per block, where the full-dimensional search closely approached the maximum, the 10-dimensional projection of the final estimate looks similar to the estimates from the reduced-dimensional search.

To further illustrate how this method behaves with invariance, we created an artificial neuron that responds to a T-junction and is invariant to rotation. The cell returns the maximum of 8 linear filters, each of which looks like a T-junction but at a different rotation. (As before, this is followed by a non-linearity and Poisson noise.) Applying the method using 3 different random starting points shows some of the different maximizing stimuli. Figure 3 shows the method using all 256 pixel dimensions. Figure 4 shows the same thing using the first 25 principal components. In each case we used 100 trials per block. This cell responds in a complicated way to fine features in the input. Because of this, we needed more principal components and more trials per block to get good performance. Unlike the previous example, projecting the stimuli found by the full dimensional search onto the first 25 principal components does not typically show anything resembling a T.

2.3 Research directions

These methods open up several exciting areas of research. One line of research focuses on further efficiency improvements. PCA is perhaps the simplest dimensionality reduction technique. Other bases like wavelets or curvelets [3] may work better in practice. The hierarchical nature of these bases opens up the possibility of coarse-to-fine searches which have the potential of dramatically improving efficiency. There may also be room for improvement in the gradient ascent method itself. For example, adjusting the step size and the number of trials per block in an adaptive way seem like useful ways to hone in on the optimal stimuli.

Another line of research focuses on modifications of the technique. These methods can be easily altered to search for stimuli other than the optimal one. For example, in a cell that shows baseline firing, we can search for the least-optimal stimulus, that is, the stimulus that inhibits the cell the most. We can also add time as an input dimension and look for optimal spatio-temporal stimuli. One experiment that particularly interests us involves invariance. We would first use gradient ascent to find the optimal stimulus. A simple modification of the method would then allow us to vary the baseline stimulus in the direction of least response variation. This would map out what might be called an *invariance ridge* for the cell. Other invariance properties could be explored in a similar manner.

A further avenue for investigation is where the methods are applied. V1 is an obvious choice, but the methods should be applicable to higher levels of visual cortex, like inferior temporal cortex (IT). Földiák et al. (2003) have demonstrated that online, adaptive stimulus presentation is possible even in higher levels of visual cortex. It should also be possible to apply these methods in auditory cortex, using auditory stimuli. These methods are even applicable for computational vision. Already algorithms like SFA are producing "neurons" whose response characteristics are difficult to determine and visualize. These methods can be immediately applied to the cells produced by complex computational vision algorithms in order to gain understanding about how these algorithms perform.

3 Stimulus-Response Function Approximation

The goal of our second approach is to find not just the maxima, but the entire stimulus-response function. This is impossible for arbitrary functions, but perhaps we can find a good fit of the true stimulus-response function within some restricted parametric class.

A simple example is the linear-nonlinear-Poisson (LNP) model class, where a neuron's response function is characterized as a linear function of its inputs, followed by a nonlinear function. The output of the nonlinear function becomes the instantaneous firing rate of the neuron, where the neuron's firing is modeled as an inhomogeneous Poisson process. The goal is to estimate the linear function and the nonlinear function given the input stimulus and the output spiking process. A variety of techniques have been developed to address this problem. See for example, Simoncelli et al. (2004) [16].

Certain cells in V1 (simple cells) appear to be well approximated by the LNP model (although see [14]). Other cells in V1 (complex cells) and most cells in higher visual areas do not fit the LNP model. One way to extend the LNP model is called multidimensional LNP. In these models, the initial stage involves many linear filters. The Poisson firing rate is now a nonlinear combination of all of these filters. The techniques used for the simple LNP model can also be extended to handle the multidimensional LNP model, however, the analysis becomes more delicate and the number of stimulus presentations required increases dramatically [16].

Another way to extend the LNP model is to first transform the stimuli using a fixed, known collection of (nonlinear) functions and then apply the LNP model. This is a common technique for approaching nonlinear problems with linear methods, and is somewhat related to the Poisson regression used here and described in the Appendix. Again, the analysis becomes much more delicate. See Nykamp (2003) [11] for more details and for some interesting examples using more powerful models to quantify receptive field structure.

We believe that most, if not all of the current techniques used for neural response fitting can benefit from dimensionality reduction techniques like the ones demonstrated in the previous section. The extensions are obvious and we do not go into them here. Instead, we will describe another way to use the statistics of natural images that opens up exciting new possibilities for neural response fitting.

3.1 Filter response distributions

One of the difficulties of the LNP model is that both the linearity and the nonlinearity are unknown. Current techniques either try to estimate them simultaneously [12] or try to estimate the linearity (or linearities) first and then use this to infer the nonlinearity [16]. Not surprisingly, estimating the linearity in the presence of an unknown nonlinearity is difficult. Estimating more complicated models becomes even more difficult or perhaps impossible.

On the other hand, if somehow we had a good estimate of the nonlinearity, then the whole situation would be changed. Not only would it be straightforward to estimate linear models, but more complicated models would also be accessible. For example, the quadratic-nonlinear-Poisson (QNP) model used earlier could be estimated. In this hypothetical situation, only the quadratic part is unknown and can be easily fit using standard regression techniques. This is a very difficult model to estimate if the nonlinearity is not known. At first glance, it might seem impossible to estimate the nonlinearity first, but we think a surprising property of natural images will actually make this straightforward.

When a linear filter is applied to a random collection of natural image patches, the resulting distribution of filter responses often looks *sparse*, that is, it looks qualitatively similar to a double exponential distribution – symmetric, with a sharp peak and heavy tails [4]. This seems to be true for all zero-mean, local, linear filters. The reasons underlying this characteristic shape are not completely understood, but the phenomenon is remarkably robust.

A V1 simple cell is often approximated by an LNP model where the linear part is zero-mean and local. Thus, when presented with a collection of natural images, the output of the linear part (before the nonlinearity and the Poisson spike generation) will have a distribution that looks like a double-exponential, irrespective of the particular filter. We can use this to estimate the nonlinearity before estimating the linearity.

In fact, the method that we will propose does not rely on linearity in any way. The initial linear filter can be replaced by any (nonlinear) function of the input whose response to natural images shows this same characteristic double exponential distribution. In our experience, this includes several other models of visual neurons. It includes all linear models, as we mentioned, including overcomplete basis models like sparse coding and adaptive wavelets which seem linear but are actually nonlinear because of competition among units. It also includes units discovered by more modern techniques like slow feature analysis (SFA). Figure 5 shows the response distributions from two different types of functions applied to natural images. The second plot is the quadratic SFA cell used previously.

3.2 Estimating the (second) nonlinearity (first)

We want to generalize the LNP model to an NNP model – nonlinear-nonlinear-Poisson. We will call the first nonlinearity the response function and the second nonlinearity the rectifier. This reflects the intuitive notion that the response function characterizes how the neuron ideally responds to input and that the rectifier maps this ideal response into physiologically appropriate units, perhaps via a sigmoidal function. Of course, both functions are important for understanding the entire behavior of the neuron. Nonlinear rectification can drastically alter the properties of the response function.

The NNP model seems rather ill-defined. How do we distinguish between the two different types of nonlinearities? What is even the point of two nonlinearities? One will suffice. We can constrain things somewhat by requiring that the response nonlinearity has a specific distribution when presented with natural stimuli – namely, that the distribution is a double exponential. Without loss of generality, we can further assume that the distribution is mean 0 and variance 1, because centering and scaling constants can be incorporated into the rectifier. As we discussed earlier, several ideal models of response functions show distributions that are double-exponential like.

The NNP model is now constrained enough to estimate the rectifier using standard statistical techniques. We present a neuron with a random collection of natural images and use the expectation-maximization (EM) algorithm to approximate a maximum-likelihood estimate (MLE) of the entire nonlinear rectifier. Details can be found in the Appendix. Figure 6 shows the results of this estimating procedure applied to a simulated neuron. Each of the three plots shows a different number of stimulus presentations used for the estimation. The middle plot shows 1000 stimulus presentations and seems like a reasonable trade-off between goodness of fit and experimental duration.

The simulated neuron was the same quadratic SFA function used previously. Figure 7 shows the method applied to an LNP neuron with the same linear filter that was used for Figure 5. Note that the method is agnostic to the form of the response function, as long as its distribution looks double exponential on natural images. As shown in Figure 5 the distributions of these two cells are actually only approximately double exponential, but the method still works. One possible explanation for this robustness is that a sigmoidal-like rectifier, which seems physiologically reasonable, helps to mitigate the effects of outliers in the tails of the distributions. Another possible explanation is implicit smoothing in our approximation of the MLE. The true MLE of the rectifier is probably much less regular than the estimates we found.

3.3 Fitting the response function

Once the rectifier has been estimated, fitting a model to the remaining response function is conceptually simple. Theoretically, any model can be estimated (at least in the range over which the rectifier is not constant). Practically, the dimensionality of the model needs to be small enough to obtain a meaningful estimate. The same dimensionality reduction techniques that we advocated earlier can be used here.

For example, consider the simulated quadratic SFA cell used throughout. In the previous section we used natural stimuli to estimate the nonlinear rectifier. Now we present the cell with (artificial, noisy) stimuli and use standard Poisson regression techniques to estimate the parameters of the quadratic function. The Poisson regression techniques rely explicitly on our estimate of the nonlinear rectifier and cannot be used when the rectifier is not specified. Details can be found in the Appendix.

Figure 8 shows two examples of fitting the QNP model. The left example corresponds to the quadratic SFA unit used throughout. The right is another quadratic SFA unit. The parameters were estimated in 10-dimensional PCA space and the figures show the true parameters projected into this space. Figure 8 clearly shows that the qualitative properties of the parameters in the QNP model can be estimated, at least in these simulations. Quantitatively, the fit is not bad. The (normalized) inner products of the true and fitted parameter vectors are 0.9821 and 0.9409 for the left and right examples, respectively. More training examples (10000) improves the estimates until the inner product is essentially 1.

Figure 9 compares the true QNP model to the fitted QNP model on natural image data (not the training data). The fitted QNP model includes both the estimated rectifier and the estimated quadratic parameters. On natural images, the true and fitted cells behave quite similarly. Again, more training examples makes this fit almost perfect. Note that Figure 9 compares the mean response of the cells. Since this is only observed in practice through a Poisson process, which is quite noisy, these true and fitted units would be nearly indistinguishable with limited data.

3.4 Research directions

These preliminary simulations are promising and suggest several methods of possible improvement. The nonparametric rectifier estimation can probably be improved dramatically by switching to a parametric model. Not only will fewer training examples be required, but physiological experiments and biophysical theories may be able to provide insights into the form of the model. As far as fitting the response function, we have only used the simplest of methods. Regression is well understood and there are undoubtedly better methods for experimental design, estimation and validation.

The double exponential is only a crude approximation of the response distributions of linear and quadratic filters. Furthermore, other functions, like classical energy models of complex cells, have one-sided distributions that are better approximated by a (one-sided) exponential. Better models of the response distribution would improve our estimation of the rectifier and can easily be incorporated into the methods used here. It may even be possible to simultaneously estimate the rectifier and the response distribution if each is restricted to a small parametric class.

Modeling the entire stimulus / response function of a neuron provides a wealth of information about how the neuron behaves. This information can be used to investigate things like functional connectivity, which are crucial for understanding the computational strategies used by the brain. If this modeling program is successful, it will certainly create many more questions and directions for further research.

A Mathematical Appendix

A.1 Response gradient approximation

Let x be an n-dimensional vector and let $g : \mathbb{R}^n \to \mathbb{R}$ be any real-valued function of x. We want to approximate the gradient ∇g at a fixed point $\tilde{x}$. We do not know g or ∇g, but for any point x we can observe independent realizations of a random variable $Y(x)$ with mean $g(x)$. For example, x is an image patch, $g(x)$ is the (unobservable) average response of a neuron to that image patch and $Y(x)$ is the observed spike count in some window after a single presentation of the stimulus x.

Let ΔX be an n-dimensional random vector (noise) with mean 0 and nonsingular covariance matrix Σ. Let $\Delta Y = Y(\tilde{x}+\Delta X) - E[Y(\tilde{x}+\Delta X)]$ be the response to $\tilde{x}+\Delta X$, shifted to have 0 mean. (E is expected value and in this case is taken over both ΔX and Y.) We claim that to a first order approximation

$$\nabla g(\tilde{x}) \approx \Sigma^{-1} E[\Delta X \Delta Y]. \tag{A.1}$$

The covariance $E[\Delta X \Delta Y]$ is an n-dimensional vector because ΔX is a vector and ΔY is a scalar.

In practice, we create a sequence of independent realizations of ΔX, say $\Delta X_1, \ldots, \Delta X_S$. We add this noise to the baseline stimulus $\tilde{x}$ and collect the sequence of responses $Y(\tilde{x}+\Delta X_1), \ldots, Y(\tilde{x}+\Delta X_S)$. We subtract the empirical mean

$$\Delta Y_s = Y(\tilde{x}+\Delta X_s) - \langle Y(\tilde{x}+\Delta X_t)\rangle_t$$

and use (A.1) to approximate

$$\nabla g(\tilde{x}) \approx \Sigma^{-1} \langle \Delta X_s \Delta Y_s \rangle_s .$$

(The empirical mean is $\langle c_s \rangle_s = S^{-1} \sum_{s=1}^{S} c_s$.) In the context of gradient ascent, we would then update the baseline stimulus $\tilde{x}$ by

$$\tilde{x} \mapsto \tilde{x} + \epsilon \Sigma^{-1} \langle \Delta X_s \Delta Y_s \rangle_s$$

for some small positive constant ϵ. (If we take $\epsilon < 0$, then this is gradient descent.) In the simulations in the text we take $\epsilon = .1$ and $\Sigma = .25I$, where I is the identity matrix.

In many situations is makes more sense to perform an online search subject to some constraint. In this case we follow each gradient ascent by a projection back into the constrain space [15]. For example, in the simulations in the text we did gradient ascent subject to a constant norm (intensity) constraint on the stimulus. So we updated the baseline stimulus as before but then projected back to the appropriate norm:

$$\tilde{x} \mapsto \tilde{x} + \epsilon \Sigma^{-1} \langle \Delta X_s \Delta Y_s \rangle_s$$
$$\tilde{x} \mapsto C \frac{\tilde{x}}{\|\tilde{x}\|}.$$

The norm constraint was $C = 2.0030$, which was the average norm from the (centered) PCA training data.

Proof of (A.1). This calculation is outlined in [5]. The formula is based on the first order, linear approximation of g as

$$g(x+\Delta x) \approx g(x) + \Delta x^T \nabla g(x), \tag{A.2}$$

where $\cdot^T$ denotes transpose. Throughout x is fixed and known.

Using this we first approximate

$$\begin{aligned} &E[Y(x+\Delta X)] = E[E[Y(x+\Delta X)|\Delta X]] = E[g(x+\Delta X)] \\ &\approx E[g(x) + \Delta X^T \nabla g(x)] = g(x) + E[\Delta X^T]\nabla g(x) = g(x), \end{aligned} \tag{A.3}$$

since ΔX has mean 0. Now we have

$$\begin{aligned} &E[\Delta X \Delta Y] = E[E[\Delta X \Delta Y|\Delta X]] = E[\Delta X E[\Delta Y|\Delta X]] \\ &\approx E[\Delta X E[Y(x+\Delta X) - g(x)|\Delta X]] = E[\Delta X(g(x+\Delta X) - g(x))] \\ &\approx E[\Delta X \Delta X^T \nabla g(x)] = \Sigma \nabla g(x), \end{aligned}$$

where the first approximation comes from (A.3) and the second from (A.2). Multiplying by Σ^{-1} gives (A.1). □

A.2 Rectifier estimation

Let X be a random n-dimensional vector with unknown distribution and $r : \mathbb{R} \to \mathbb{R}$ be an unknown function. Suppose however that $Z = r(X)$ has a known distribution, say with density p_Z. In the text we take this distribution to be double exponential distribution with mean 0 and variance 1, that is

$$p_Z(z) = \frac{1}{\sqrt{2}} e^{-\sqrt{2}|z|}, \quad -\infty < z < \infty,$$

but here the specific form of p_Z is not important. We are given a sequence of i.i.d. r.v.'s $X_1, \ldots, X_S$ with the same distribution as X. We also get to observe Poisson counts from an NNP model, namely $Y_1, \ldots, Y_S$ where Y_s is a Poisson random variable with mean $f(r(X_s)$. We want to estimate the (rectifier) function $f : \mathbb{R} \to [0, \infty)$, which is unknown. As we mentioned, we do not know r, but we know the distribution of $Z = r(X)$.

Let $Z_s = r(X_s)$. We will completely ignore our knowledge of the X_s (this is obscured by r anyway) and use only the fact that Z_s has known density p_Z. The model becomes: $Y_1, \ldots, Y_S$ are independent Poisson random variables and Y_s has mean $f(Z_s)$ for unknown f and unobserved $Z_1, \ldots, Z_S$. Since we know the distribution of the Z_s we can use maximum likelihood estimation to estimate f. The log likelihood is

$$\begin{aligned} \log p(Y_1, \ldots, Y_S|f) &= \sum_{s=1}^{S} \log p(Y_s|f) = \sum_{s=1}^{S} \log \int_{\mathbb{R}} p(Y_s|Z_s = z; f) p_Z(z) dz \\ &= \sum_{s=1}^{S} \log \int_{\mathbb{R}} \frac{e^{-f(z)} f(z)^{Y_s}}{Y_s!} p_Z(z) dz. \end{aligned} \tag{A.4}$$

It is not clear that this can be maximized analytically.

Instead, we frame this as a missing data problem – the Z_s are missing – and use the expectation maximization (EM) algorithm (see McLachlan and Krishnan, 1997, for details and references [9]). The EM update equation is

$$f_{k+1}(z) = \frac{\sum_{s=1}^{S} Y_s p_Z(z|Y_s; f_k)}{\sum_{s=1}^{S} p_Z(z|Y_s; f_k)}. \tag{A.5}$$

This calculation is detailed below. The functions $p_Z(z|Y_s; f_k)$ can be determined using Bayes' Rule

$$p_Z(z|Y_s; f_k) = \frac{p(Y_s|Z_s = z; f_k)p_Z(z)}{\int_{\mathbb{R}} p(Y_s|Z_s = \tilde{z}; f_k)p_Z(\tilde{z})d\tilde{z}} = \frac{e^{-f_k(z)}f_k(z)^{Y_s}p_Z(z)}{\int_{\mathbb{R}} e^{-f_k(\tilde{z})}f_k(\tilde{z})^{Y_s}p_Z(\tilde{z})d\tilde{z}}. \tag{A.6}$$

All of these calculations can be carried out on a grid (in z) over the effective range of p_Z, which is known. The computations can be sped up significantly by taking advantage of the multiplicities of the Y_s which are Poisson counts. For example, the integrals in the denominator of (A.6) need only be evaluated for each distinct value of the Y_s. If the distinct values of the Y_s are $Y_{(1)}, \ldots, Y_{(M)}$, with multiplicities $N_1, \ldots, N_M$, then (A.5) becomes

$$f_{k+1}(z) = \frac{\sum_{m=1}^{M} N_m Y_{(m)} p_Z(z|Y_{(m)}; f_k)}{\sum_{m=1}^{M} N_m p_Z(z|Y_{(m)}; f_k)}.$$

For the simulations in the text, we take p_Z to be a double exponential and we initialize the EM algorithm with

$$f_1(z) = \begin{cases} \alpha & \text{if } z \leq -10 \\ (\beta - \alpha)(z + 10)/20 + \alpha & \text{if } -10 < z < 10 \\ \beta & \text{if } z \geq 10 \end{cases},$$

which is linear over the effective range of p_Z and then held constant outside of that range. α and β are determined from the data. We take α to be the 5th percentile of the Y_s and β to be the 95th percentile. We estimate f on a .1 grid from -10 to 10.

The EM algorithm is run until the successive estimates of f are not changing much at any point on the grid, specifically, until $\max_z |f_{k+1}(z) - f_k(z)| < .01$ or 100 iterations, which ever comes first (usually the former). The entire process typically takes under 1/2 second on a desktop PC. The stopping criterion for the EM algorithm appears to be crucial in this context. The MLE estimate of f is probably not very smooth. Stopping the EM algorithm earlier than computationally necessary effectively introduces some smoothing into the estimate. This is what we did in the text. Allowing the EM algorithm to iterate longer generates worse estimates of f. It is likely that an explicit regularization term or a parameterized model of f would help to create better and more robust estimates than those used here.

In our experience, a single large value for one of the Y_s can raise the far right side of the estimated f much higher than the true rectifier. This is easy to spot visually because of a discontinuity and can be remedied by removing the outlier from the data. Outlier removal could be automated, but may not be necessary in practice because of physiological upper limits on the number of spikes that can occur in a given time window. These are the sorts of issues that cannot be adequately addressed by simulation experiments.

Proof of (A.5). The complete data log likelihood is

$$\log p(Y_1, Z_1, \ldots, Y_S, Z_S|f) = \sum_{s=1}^{S} \log p(Y_s, Z_s|f) = \sum_{s=1}^{S} \log \left[\frac{e^{-f(Z_s)} f(Z_s)^{Y_s}}{Y_s!} p_Z(Z_s) \right]$$

$$= \sum_{s=1}^{S} \big[-f(Z_s) + Y_s \log f(Z_s) - \log Y_s! + \log p_Z(Z_s) \big]. \tag{A.7}$$

The EM algorithm creates a sequence of estimates $f_1, f_2, \dots$ that increase the likelihood in (A.4). Given f_k we find f_{k+1} by maximizing over f the expected value of (A.7) given the observations $Y_1, \dots, Y_S$ and using f_k. This conditional expectation is

$$\begin{aligned} &E\big[\log p(Y_1, Z_1, \dots, Y_S, Z_S|f)\big|Y_1, \dots, Y_S; f_k\big] \\ &\quad = \sum_{s=1}^{S} E\big[-f(Z_s) + Y_s \log f(Z_s) - \log Y_s! + \log p_Z(Z_s)\big|Y_s; f_k\big] \\ &\quad = \sum_{s=1}^{S} \int_{\mathbb{R}} \big[-f(z) + Y_s \log f(z) - \log Y_s! + \log p_Z(z)\big] p_Z(z|Y_s; f_k) dz. \end{aligned}$$

To maximize this over f we can ignore the parts that do not depend on f and choose

$$f_{k+1} = \arg\max_f \sum_{s=1}^{S} \int_{\mathbb{R}} \big[-f(z) + Y_s \log f(z)\big] p_Z(z|Y_s; f_k) dz. \tag{A.8}$$

The argument of (A.8) is concave in f because of the concavity of the logarithm, so any critical point of the (functional) derivative will be a global maximizer. Perturbing f by $f + \epsilon\eta$ for an arbitrary function η, taking the derivative w.r.t. ϵ, evaluating at $\epsilon = 0$ and setting the result equal to zero gives the following equation for critical points:

$$\sum_{s=1}^{S} \int_{\mathbb{R}} p_Z(z|Y_s; f_k) \left[-\eta(z) + Y_s \frac{\eta(z)}{f_{k+1}(z)}\right] dz = 0 \quad \text{for all functions } \eta,$$

or equivalently,

$$\int_{\mathbb{R}} \eta(z) \sum_{s=1}^{S} p_Z(z|Y_s; f_k) \left[\frac{Y_s}{f_{k+1}(z)} - 1\right] dz = 0 \quad \text{for all functions } \eta. \tag{A.9}$$

Since η is arbitrary, we must have

$$\sum_{s=1}^{S} p_Z(z|Y_s; f_k) \left[\frac{Y_s}{f_{k+1}(z)} - 1\right] = 0 \quad \text{for all } z,$$

which has the unique solution given in (A.5) (except for a few pathological cases like when all the $p_Z(z|Y_s; f_k)$ are 0 for some z and then $f_{k+1}(z)$ can be anything). □

A.3 Poisson regression

Poisson regression is well studied [10]. The simplest form is that the observations Y_s are independent Poisson random variables with mean $f(\beta^T h(X_s))$, where $\beta^T = (\beta_0, \beta_1, \dots, \beta_n)$ is the vector of parameters to be estimated, X_s is the sth stimulus and

$$h(X_s) = (h_0(X_s), h_1(X_s), \dots, h_n(X_s))^T$$

is the vector of predictor variables for the sth stimulus. The functions f and h are known.

Typically $h_0 \equiv 1$ and allows a constant term to enter the model. If X_s is a vector (like pixels in a receptive field), then h_j can be X_{sj}, the jth element of X_s. This corresponds to the LNP model. Often, there will be many more predictor variables than stimulus dimensions. For example, in the QNP model Y_s has mean $f(X_s^T A X_s + X_s^T B + C)$ for matrix A, vector B and constant C. We can write this as $f(\beta^T h(X_s))$ by including not only the elements of X_s in h but also all of the interaction terms like $h_\ell(X_s) = X_{sj}X_{sk}$. The entries in β will thus correspond to certain elements of A, B or C.

Since h is known and X_s is observed, we can write $Z_s = h(X_s)$, where Z_s is a vector, and forget about h and X_s. Y_s has mean $f(\beta^T Z_s)$. The log likelihood equation is

$$\log p(Y_1, \ldots, Y_S|\beta) = \sum_{s=1}^{S} \log p(Y_s|\beta) = \sum_{s=1}^{S} \log \frac{e^{-f(\beta^T Z_s)} f(\beta^T Z_s)^{Y_s}}{Y_s!}$$
$$= \sum_{s=1}^{S} \big[-f(\beta^T Z_s) + Y_s \log f(\beta^T Z_s) - \log Y_s!\big].$$

Since f, the Y_s and the Z_s are known, this can be maximized with standard nonlinear optimization tools. The gradient vector is easy to calculate and this can be used to speed the convergence. The gradient vector requires an estimate of the derivative of the rectifier f. For this we just approximated $f'(z) \approx (f(z+\Delta) - f(z-\Delta))/(2\Delta)$ at grid points z, where Δ is the grid resolution. We took both f and f' to be linear between grid points.

For the simulations in the text, we initialized the optimization at $\beta^T = (1, 0, \ldots, 0)$, that is, only a constant term. Using Matlab's unconstrained nonlinear optimization typically took under 2 minutes for the 65-parameter QNP problem.

For stimuli, we first chose standard Gaussian (white) noise in the 10-dimensional PCA parameter space. To get a better sampling of the input space, we self-normalized each noise vector and then scaled by a random amount (Gaussian, mean 0, standard deviation 3). Fitting with white noise or with natural stimuli seemed to work, but not quite as well as with this method. There are probably much better experimental design methods in the literature, but we did not investigate this possibility.

References

[1] Anthony J. Bell and Terrence J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[2] Pietro Berkes and Laurenz Wiskott. Slow feature analysis yields a rich repertoire of complex-cell properties. Cognitive Sciences EPrint Archive (CogPrints) 2804, http://cogprints.ecs.soton.ac.uk/archive/00002804/ (12/08/2003), February 2003.

[3] David L. Donoho and Ana Georgina Flesia. Can recent innovations in harmonic analysis 'explain' key findings in natural image statistics? *Network: Computation in Neural Systems*, 12(3):371–393, 2001.

[4] D. J. Field. What is the goal of sensory coding? *Neural Computation*, 6:559–601, 1994.

[5] Peter Földiák. Stimulus optimisation in primary visual cortex. *Neurocomputing*, 38–40:1217–1222, 2001.

[6] Peter Földiák, Dengke Xiao, Christian Keyers, Robin Edwards, and David Ian Perrett. Rapid serial visual presentation for the determination of neural selectivity in area STSa. *Progress in Brain Research*, 144:107–116, 2003.

[7] Patrik O. Hoyer and Aapo Hyvaärinen. A multi-layer sparse coding network learns contour coding from natural images. *Vision Research*, 42:1593–1605, 2002.

[8] Aapo Hyvaärinen and Patrik O. Hoyer. Topographic independent component analysis as a model of V1 organization and receptive fields. *Neurocomputing*, 38–40:1307–1315, 2001.

[9] Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, New York, 1997.

[10] John Neter, Michael H. Kutner, Christopher J. Nachtsheim, and William Wasserman. *Applied Linear Regression Models*. Irwin, Chicago, 3rd edition, 1996.

[11] Duane Q. Nykamp. Measuring linear and quadratic contributions to neuronal response. *Network: Computation in Neural Systems*, 14:673–703, 2003.

[12] Duane Q. Nykamp and Dario L. Ringach. Full identification of a linear-nonlinear system via cross-correlation analysis. *Journal of Vision*, 2:1–11, 2002.

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

[14] Nicole C Rust, Odelia Schwartz, J Anthony Movshon, and Eero Simoncelli. Spike-triggered characterization of excitatory and suppressive stimulus dimensions in monkey V1. *Neurocomputing*, 2004 (to appear).

[15] L. E. Scales. *Introduction to Non-Linear Optimization*. Springer-Verlag, New York, 1985.

[16] Eero P. Simoncelli, Liam Paninski, Jonathan Pillow, and Odelia Schwartz. Characterization of neural responses with stochastic stimuli. In M. Gazzaniga, editor, *The New Cognitive Sciences*. MIT Press, 3rd edition, 2004 (to appear).

[17] Laurenz Wiskott and Terrence J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14:715–770, 2002.

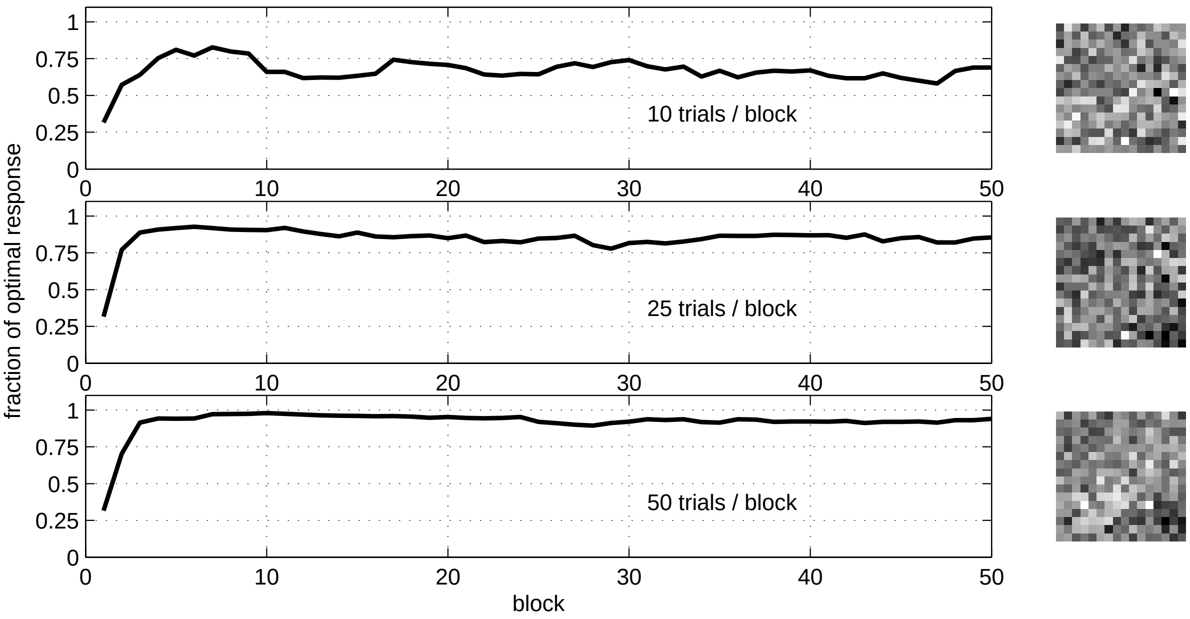

Figure 1: Full-dimensional search.

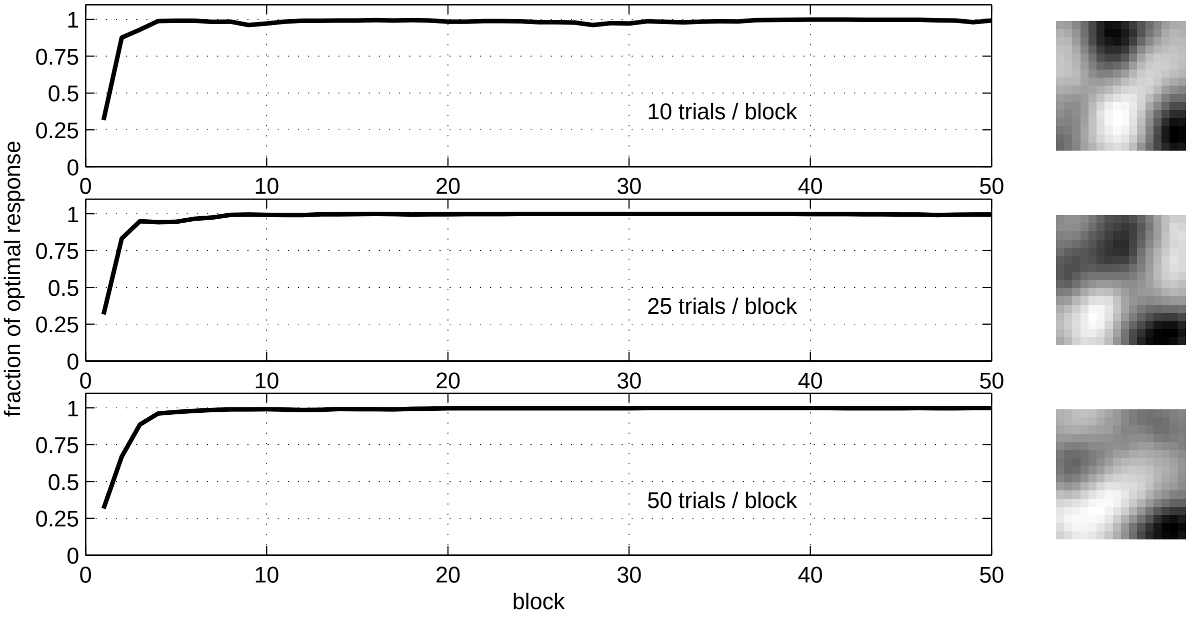

Figure 2: Reduced-dimensional search.

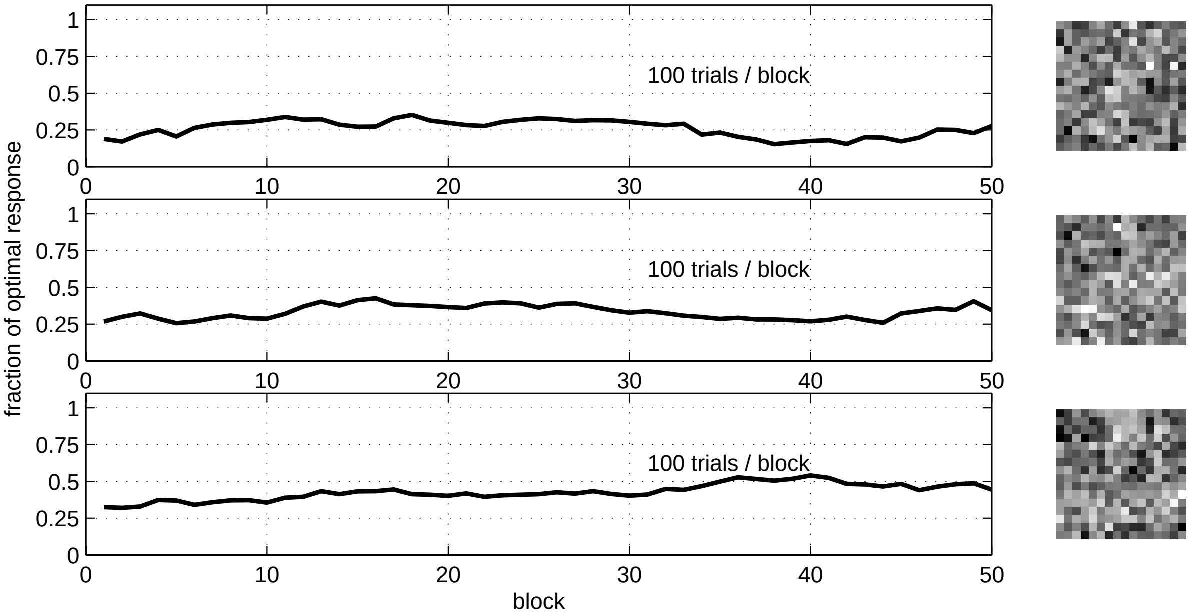

Figure 3: Full-dimensional search.

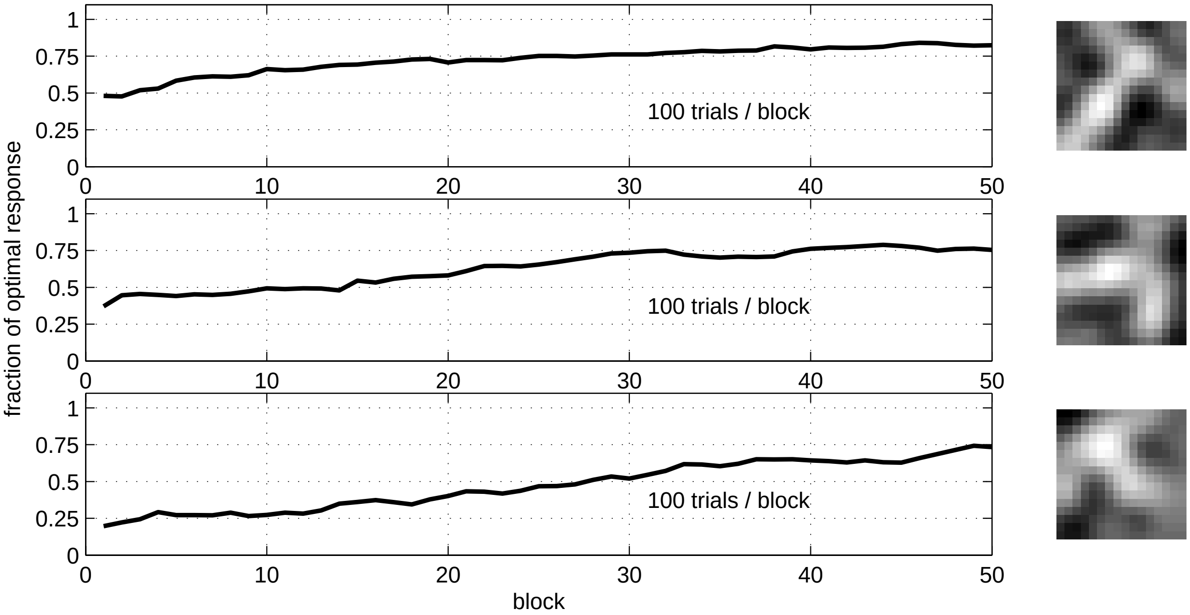

Figure 4: Reduced-dimensional search.

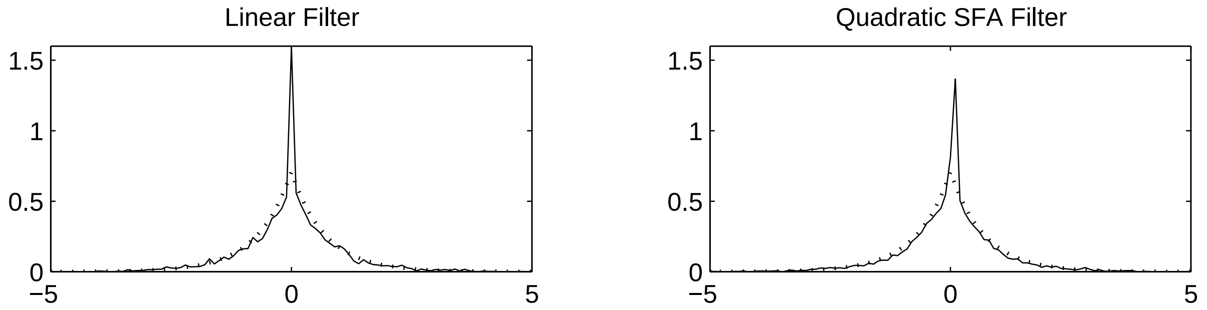

Figure 5: Solid line: filter distributions over natural images normalized to mean 0 and variance 1. Dotted line: double exponential distribution.

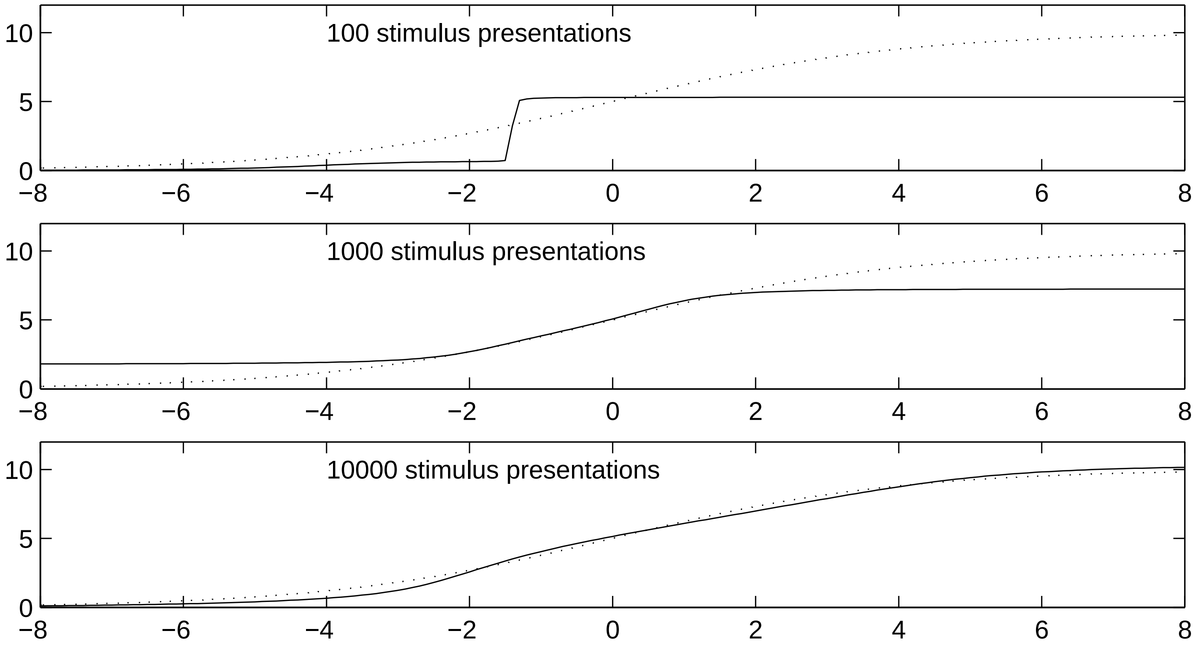

Figure 6: Quadratic SFA unit. Solid line: fitted rectifier. Dotted line: true rectifier.

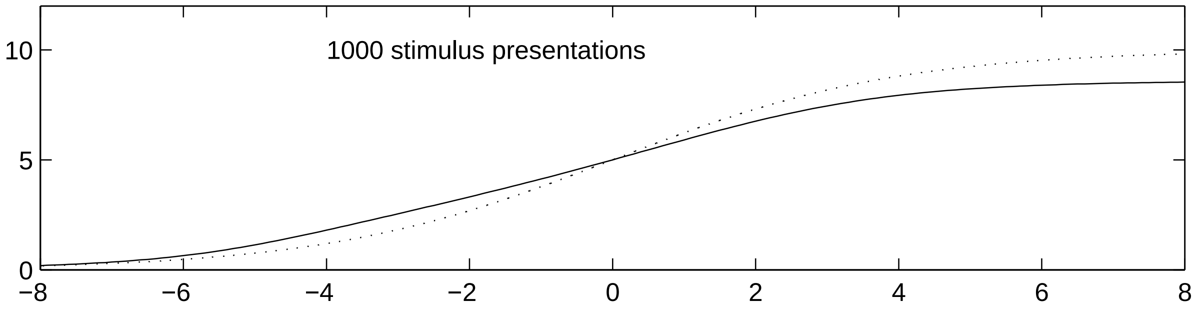

Figure 7: Linear Unit. Solid line: fitted rectifier. Dotted line: true rectifier.

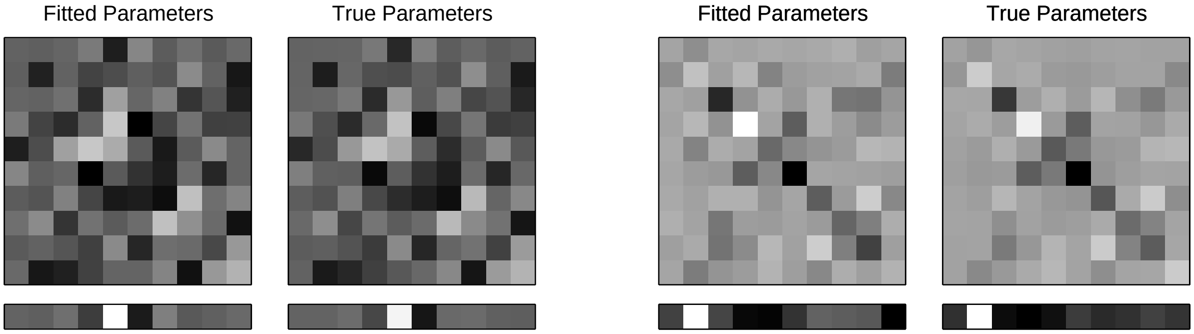

Figure 8: Fitted versus true parameters in the quadratic model. The upper, square images show the A matrices in the quadratic form. The lower, rectangular images show the B vectors. The left two plots are from the SFA unit used in the text. The right two are from another quadratic SFA unit. The true parameters have been projected into the 10-dimensional PCA space for easy comparison with the fitted parameters.

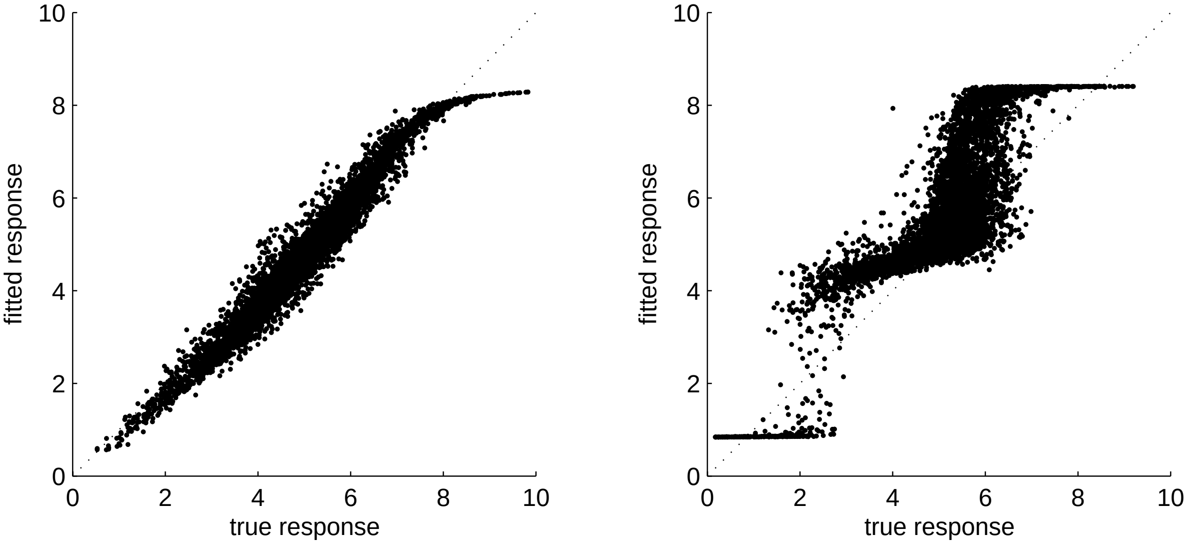

Figure 9: Comparison of true and fitted models on natural images. The left and right models are the same as in Figure 8. If the models were identical, all the points would lie on the diagonal dotted line because these plots show the mean (ideal) response, not the noisy Poisson observations. The estimated rectifier in the model on the right was wavy, unlike the true rectifier.