Statistical analysis of neural data:
Regression approaches for modeling neural responses and stimulus decoding

Liam Paninski
Department of Statistics and Center for Theoretical Neuroscience
Columbia University
http://www.stat.columbia.edu/∼liam

September 28, 2013

Contents

Before we attack the full neural coding problem of learning the full high-dimensional $p(\vec{n}|\vec{x})$, where $\vec{n}$ is a full spike train, or multivariate spike train, etc., and $\vec{x}$ is the observed signal with which we are trying to correlate $\vec{n}$ ($\vec{x}$ could be the stimulus, or observed motor behavior, etc.), it is conceptually easier to begin by trying to predict the scalar $p(n(t)|\vec{x})$, i.e., to predict the spike count in a single time bin t. From a statistical modeling point of view, we will therefore begin by discussing a simple first-order model for $p(\vec{n}|\vec{x})$:

$$p(\vec{n}|\vec{x}) = \prod_t p(n(t)|\vec{x}),$$

i.e., the responses $n(t)$ in each time bin are conditionally independent given the observed $\vec{x}$. (This model is typically wrong but it's a useful place to start; later we'll discuss a variety of ways to relax this conditional independence assumption (Paninski et al., 2004; Truccolo et al., 2005).)

Understanding $p(n(t)|\vec{x})$ is already a hard problem, due to the high dimensionality of $\vec{x}$, and the fact that, of course, we only get to observe a noisy version of this high-dimensional function of $\vec{x}$.

1 Nonparametric estimation of spike responses is straightforward in low-dimensional cases

In the simplest case, we may take dt, the width of the time bin in which $n(t)$ is observed, to be small enough that only at most one spike is observed per time bin. Then estimating $p(n(t) = 1|\vec{x})$ is equivalent to estimating $E(n(t)|\vec{x})$. We may begin by attempting to estimate this function $E(n|\vec{x})$ nonparametrically: this approach is attractive because it requires us to make fewer assumptions about the shape of $E(n|\vec{x})$ as a function of $\vec{x}$ (although as we will see, we still have to make some kind of assumption about how sharply $E(n|\vec{x})$ is allowed to vary as a function of $\vec{x}$). One simple method is based on kernel density estimation (Hastie et al., 2001; Devroye and Lugosi, 2001): we form the estimate

$$\hat{E}(n|\vec{x}) = \frac{\sum_t w(\vec{x}_t - \vec{x}) n_t}{\sum_t w(\vec{x}_t - \vec{x})},$$

where $w(.)$ is a suitable smoothing kernel; typically, $w(.)$ is chosen to be positive, integrable with respect to $\vec{x}$, and symmetric in $\vec{x}$ about $\vec{x} = 0$. See Fig. 1 for an illustration in the case that $\vec{x}$ is one-dimensional. A related approach is to simply form a histogram for $\vec{x}$, and set $\hat{E}(n|\vec{x})$ to be the mean of $n(t)$ for all time points t for which the corresponding $\vec{x}_t$ fall in the given histogram bin (Chichilnisky, 2001).

The wider $w(.)$ (or equivalently, the histogram bin) is chosen to be, the smoother the resulting estimate $\hat{E}(n|\vec{x})$ becomes; thus it is common to use an adaptive approach in the choice of $w(.)$, where $w(.)$ is chosen to be wider in regions of the $\vec{x}$-space where there are fewer samples $\vec{x}_t$ (where more smoothing is necessary) and narrower in regions where more data are available.

This simple smoothing approach is quite effective in the case that $\dim(\vec{x}) \leq 2$, where it is possible to visualize the estimated function $\hat{E}(n|\vec{x})$ directly. We will return to these smoothing methods in a later chapter, after we develop some theory for generalized linear models; as we will see, both the histogram and kernel smoother approaches can be understood in the

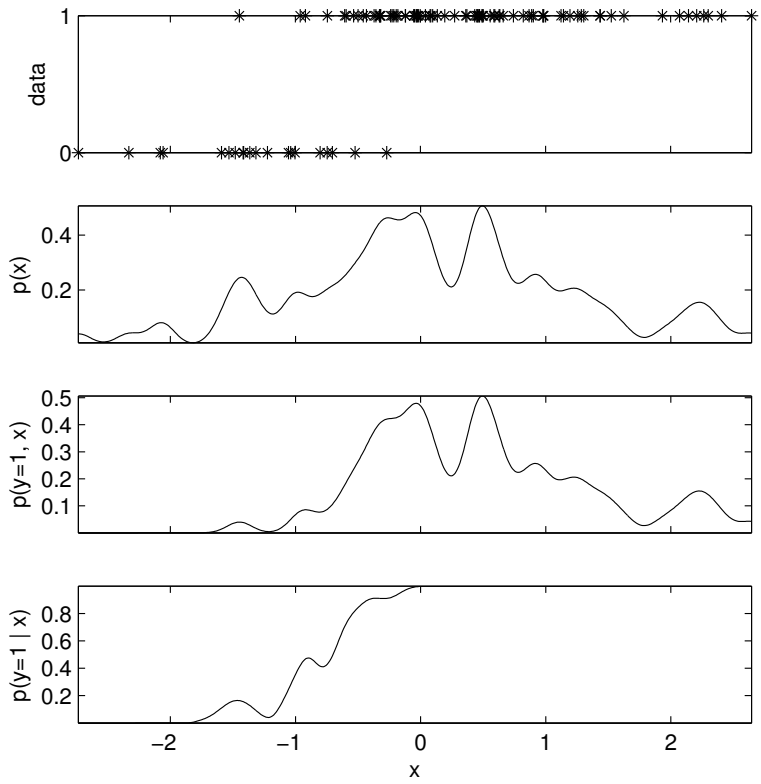

Figure 1: Illustration of the Gaussian smoothing kernel applied to simulated one-dimensional data x. **Top**: observed binary data. **Second panel**: Estimated density $\hat{p}(x) = \frac{1}{N}\sum_{i=1}^{N} w(x_i - x)$, with the smoother $w(.)$ chosen to be Gaussian with mean zero and standard deviation .1. **Third panel**: Estimated joint density $\hat{p}(x, y = 1) = \frac{1}{N}\sum_{i=1}^{N} 1(y_i = 1)w(x_i - x)$. **Bottom**: Estimated conditional density $\hat{p}(y = 1|x) = \hat{p}(y = 1, x)/\hat{p}(x)$.

context of likelihood-based methods. However, for higher-dimensional $\vec{x}$ this nonparametric approach becomes less useful, in effect because the number of samples needed to "fill in" a multidimensional histogram scales exponentially with $d = \dim(\vec{x})$: this is one example of the so-called "curse of dimensionality" (Duda and Hart, 1972; Hastie et al., 2001). Thus in the following subsections we will examine more parametric methods for estimating the firing rate.

2 Multiple linear regression provides the simplest approach for modeling the firing rate given higher-dimensional stimuli

A great variety of more involved nonparametric approaches have been developed in the statistics and machine learning community (Hastie et al., 2001). However, the approach emphasized here will be more model-based; this makes the results somewhat easier to interpret, and more importantly, allows us to build in more about what we know about biophysics, functional neuroanatomy, etc.

The simplest model-based approach is to employ classical linear multiple regression. We

model n_t as

$$n_t = \vec{k}^T \vec{x}_t + b + \epsilon_t,$$

where ϵ_t is taken to be an independent and identically distributed (i.i.d.) random variable with mean zero and variance $Var(\epsilon_t) = \sigma^2$. The solution to the problem of choosing the parameters $(\vec{k}, b)$ to minimize the mean-square error

$$\sum_t [\vec{k}^T \vec{x}_t + b - n_t]^2 \tag{1}$$

is well-known (Kutner et al., 2005): the best-fitting parameter vector $\hat{\theta}_{LS} = (\vec{k}^T \;\; b)^T_{LS}$ satisfies the "normal equations"

$$(X^T X)\hat{\theta}_{LS} = X^T \vec{n},$$

where the matrix X is defined as

$$X_t = (\vec{x}_t^T \;\; 1)$$

and

$$\vec{n} = \begin{pmatrix} n_1 & n_2 & \dots & n_t \end{pmatrix}^T.$$

The normal equations are derived by simply writing the mean square error in matrix form,

$$\sum_t [\vec{k}^T \vec{x}_t + b - n_t]^2 = ||X\theta - \vec{n}||_2^2 = \theta^T X^T X \theta - 2\theta^T X^T \vec{n} + \vec{n}^T \vec{n},$$

and setting the gradient with respect to the parameters $\theta = (\vec{k}^T \;\; b)^T$ equal to zero. In the case that the matrix $X^T X$ is invertible, we have the nice explicit solution

$$\hat{\theta}_{LS} = (X^T X)^{-1} X^T \vec{n};$$

more generally, the solution to the normal equations may be nonunique, and additional constraints may need to be imposed to obtain a unique solution, as we will discuss at more length below.

There is an important connection between the least-squares solution and the maximum likelihood estimator if the noise terms ϵ_t are Gaussian. In this case we can write the log-likelihood of the observed outputs $\{n_t\}$ given the parameters $(\vec{k}, b, \sigma^2)$ and the observed inputs $\{\vec{x}_t\}$ as

$$\begin{aligned} \log p(\{n_t\}|\{\vec{x}_t\}, \vec{k}, b, \sigma^2) &= \log \prod_t \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(\vec{k}^T \vec{x}_t + b - n_t)^2\right) \\ &= c - a \sum_t \left((\vec{k}^T \vec{x}_t + b - n_t)^2\right), \end{aligned}$$

where the scalars $a > 0$ and c do not depend on $(\vec{k}, b)$. Thus maximizing the log-likelihood leads to the same solution for $(\vec{k}, b)$ as does minimizing the mean square error.

So the linear regression approach leads to a nice, computationally-tractable solution; moreover, the statistical properties of the estimated parameters $\hat{\theta}_{LS}$ are very well-understood: we can construct confidence intervals and do hypothesis testing using standard, well-defined techniques (again, see (Kutner et al., 2005) for all details). Finally, the components of the solution $(X^T X)^{-1} X^T \vec{n}$ turn out to have some useful, straightforward interpretations. For example,

$$X^T \vec{n} = \begin{pmatrix} \sum_t \vec{x}_t^T n_t & \sum_t n_t \end{pmatrix}^T;$$

forming the quotient of the two terms on the right, $[\sum_t \vec{x}_t n_t]/[\sum_t n_t]$, gives us the spike-triggered average (de Boer and Kuyper, 1968) — the conditional mean $\vec{x}$ given a spike — about which we will have much more to say in a moment. Similarly, the matrix $X^T X$ contains all the information we need to compute the correlation matrix of the stimulus.

2.1 Different loss functions may be used to obtain more robust estimators

The least-squares estimate is very non-robust to outliers: by changing a single (n_t, x_t) pair we can cause arbitrarily large changes in the estimate θ_{LS}. One way to fix this problem is to optimize a different objective function. (Another method is to include prior information about the true underlying parameter value θ in our estimator; we will discuss Bayesian approaches based on this idea in much more depth below.) For example, instead of eq. (1) we could minimize an objective function of the form

$$\sum_t G(\vec{k}^T \vec{x}_t + b - n_t), \tag{2}$$

where $G(u)$ is a convex function of u that is minimized for $u = 0$ and which grows more slowly than the quadratic function for large values of the error $|u|$. Common choices include the absolute error $G(u) = |u|$ or the "epsilon-insensitive" loss $G_\epsilon(u) = \max(0, |u| - \epsilon)$. In general, no analytic solution exists for minimizing the resulting objective function (2), and numerical convex minimization algorithms are required. For both the absolute-error and epsilon-insensitive loss functions, the problem of minimizing eq. (2) can be cast as a linear programming problem (i.e., minimize a linear objective function under linear inequality constraints), for which fast algorithms are available.

3 Including nonlinear terms enhances the flexibility of the regression technique

It is not clear that this simple linear regression model captures neural responses very well. Moreover, departures from the assumptions of the model might bias our estimates of the model parameters, or reduce the interpretability of the results.

A few such departures are obvious, even necessary; for example, the spike count n_t, and therefore $E(n|\vec{x})$, must be nonnegative. More importantly, the function $E(n|\vec{x})$ may be quite nonlinear, reflecting saturation, rectification, adaptation effects, etc. It is straightforward to include nonlinear terms in the regression analysis (Sahani, 2000; Kutner et al., 2005), simply by redefining the matrix X appropriately: instead of letting the t-th row X_t contain just the elements of $\vec{x}$ and 1, we may also include arbitrary functionals $\phi_i(\vec{x}_t)$:

$$X_t = \begin{pmatrix} \vec{x}^T & \phi_1(\vec{x}_t) & \phi_2(\vec{x}_t) & \dots & \phi_m(\vec{x}_t) & 1 \end{pmatrix}.$$

The resulting model of the response is now nonlinear:

$$n_t = \vec{k}^T \vec{x} + \sum_{i=1}^{m} a_i \phi_i(\vec{x}) + b + \epsilon_t,$$

with the maximum-likelihood (least-squares) parameters $(\vec{k}, \vec{a}, b)_{LS}$ determined by solving the normal equations (with the suitably redefined X) exactly as in the fully linear case.

We still need to make sure that the predicted firing rate $E(n|\vec{x})$ remains nonnegative. This nonnegativity constraint may be enforced with a collection of linear inequality constraints

$$\vec{k}^T\vec{x} + \sum_{i=1}^{m} a_i\phi_i(\vec{x}) + b \geq 0 \quad \forall\vec{x},$$

(i.e., one constraint for each value of $\vec{x}$; note that each constraint is linear as a function of the parameters $(\vec{k}, \vec{a}, b)$, despite the nonlinearity in $\vec{x}$). This converts the original unconstrained quadratic regression problem into a quadratic program[1], which retains much of the tractability of the original problem.

This nonlinear regression approach is useful in a number of contexts. One example involves the incorporation of known presynaptic nonlinearities: if we know that the neuron of interest receives input from presynaptic neurons which perform some well-defined nonlinear transformation on the stimulus $\vec{x}$, it is worth incorporating this knowledge into the model (Rust et al., 2006).

3.1 Volterra-Wiener series

Another common application is a kind of polynomial expansion referred to as a "Volterra-Wiener" series (Marmarelis and Marmarelis, 1978). The N-th order Volterra expansion involves all polynomials in $\vec{x}$ up to the N-th order: thus the zero-th order model is

$$n_t = b + \epsilon_t,$$

with a corresponding design matrix

$$X_t = (1);$$

the first order expansion is the linear model discussed above ($n_t = b + \vec{k}^T\vec{x}_t + \epsilon_t$); the second-order model is

$$n_t = b + \vec{k}^T\vec{x}_t + \sum_{ij} a_{ij}\vec{x}_t(i)\vec{x}_t(j) + \epsilon_t,$$

with

$$X_t = \begin{pmatrix} 1 & \vec{x}_t^T & \vec{x}_t(1)\vec{x}_t(1) & \vec{x}_t(2)\vec{x}_t(1) & \vec{x}_t(3)\vec{x}_t(1) & \dots & \vec{x}_t(2)\vec{x}_t(2) & \dots & \vec{x}_t(d)\vec{x}_t(d) \end{pmatrix},$$

while the third-order model includes all triplet terms $\vec{x}(i)\vec{x}(j)\vec{x}(l)$, and so on. The attraction of these expansion-based models is that, in principle, we may approximate an arbitrary smooth function $E(n|\vec{x})$ by using a sufficiently large expansion order N, while the order N provides a natural, systematic index of the complexity of the model.

[1] A quadratic program (QP) is a linearly-constrained quadratic optimization problem of the form

$$\max_\theta \frac{1}{2}\theta^T A\theta + a^T\theta, \; a_i^T\theta \geq c_i \quad \forall i,$$

for some negative semidefinite matrix A and some collection of vectors a and a_i and corresponding scalars c_i. Quadratic programs are do not in general have analytic solutions, but if the number of inequality constraints is small then we may numerically solve a QP in the same order of computational time as required to solve the unconstrained problem $\max_\theta \frac{1}{2}\theta^T A\theta + a^T\theta$, since we are maximizing a particularly simple concave function on a particularly simple convex space. However, if the number of constraints is large then solving the QP efficiently may become more difficult.

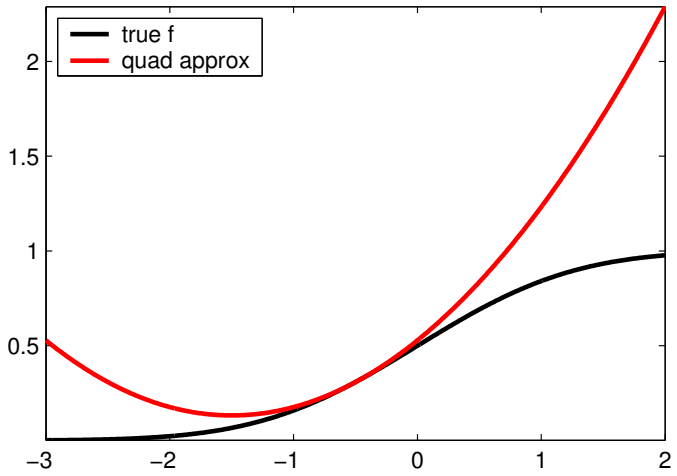

Figure 2: A simple toy example illustrating some flaws in the Volterra expansion approach. In this case we are approximating the true firing rate function $f(.)$ by its second-order Taylor series. The problem here is that the function $f(x)$ saturates for large values of x, while of course the x^2 term increases towards infinity, thus making a poor approximation.

However, several problems are evident in this nonlinear regression approach. The key problem is that it is often difficult to determine *a priori* what nonlinearities $\phi(\vec{x})$ to include in the analysis. In the Volterra-Wiener approach described above, for example, the polynomial expansion works poorly to approximate a saturating function $E(n|\vec{x})$, in the sense that a large N is required to obtain a reasonable degree of accuracy, and (more importantly) the resulting approximation is unstable, with delicately balanced oscillatory terms and unbounded behavior at the boundary of the $\vec{x}$ space (poor extrapolation). In general, moreover, the number of terms required in the expansion scales unfavorably with both the expansion order N and the dimension d of $\vec{x}$. A complementary problem is that the inclusion of many terms in any regression model will lead to overfitting effects, as we discuss below (Machens et al., 2003; Smyth et al., 2003; Paninski, 2004): that is, poor generalization ability even in cases when the training error may be made small.

3.2 The kernel trick can be used to fit some very high-dimensional nonlinear models

It is interesting to note that the regression problem can be reformulated such that we never need to explicitly compute the nonlinear feature functions $\phi(x_i)$; instead, we only need to be able to compute the "kernel" matrix consisting of the dot products between all the $\phi(x_i)$ vectors: $K(i,j) =< \phi(x_i), \phi(x_j) >$. (We skip the derivation here; see e.g. (Scholkopf and Smola, 2002) for details.) This observation is useful because in some cases we can compute $K(i,j)$ directly, without having to compute $\phi(x_i)$ and $\phi(x_j)$ at all. This is especially helpful in cases for which $\phi(x)$ is very high-dimensional or infinite-dimensional. (See (Scholkopf and Smola, 2002) for a wide variety of examples.) This "kernel trick" (evaluate $K(i,j)$ directly, not the nonlinear feature functions $\phi(x_i)$) leads to faster computation when the number of samples is much smaller than the dimensionality of $\phi(x_i)$. In addition, this trick is applicable

in many other cases, not just linear regression. For example, if we replace the squared error with absolute error or the epsilon-insensitive error (recall section 2.1) and incorporate a quadratic regularizer (we will discuss regularization in depth below), the resulting quadratic program can be reformulated to only involve kernel evaluations $K(i, j)$. This kernel trick can also be applied to other classical multivariate methods, e.g. principal or canonical correlations analysis, discriminant analysis, etc. (Scholkopf and Smola, 2002).

4 *Analysis-of-variance methods may be used to determine when to include additional terms in a regression model

5 "Overfitting" is the bane of high-dimensional model estimation: training error, generalization, and cross-validation

The key thing to remember about high-dimensional data analysis is that we are looking for models that *predict* the data well, rather than *fit* the data well. For example, we typically measure the quality of a model's fit to data D by the maximum of the likelihood function,

$$L \equiv \max_{\theta \in \Theta} p(D|\theta).$$

Here Θ indexes the parameter set, corresponding to all possible models in the model class under consideration. Clearly, we can always make L larger (in principle) simply by expanding Θ, since adding elements to Θ can never decrease the maximum in the definition of L; for example, in the regression setting, we could increase L by fitting models including both linear and nonlinear terms, rather than just linear terms. Typically, by adding more and more terms in our regression we can fit any data we'd like, in the sense that L becomes arbitrarily large.

There are many problems with this approach of simply iteratively expanding the parameter space Θ to increase the fit quality L. First, of course, the higher the dimensionality of the parameter space (e.g., the more nonlinear, poorly physiologically-justified terms we include in our regression analysis), the more uninterpretable and overly complex our models become. For similar reasons, higher-dimensional models often pose greater computational difficulties than do simpler models. The most important problem with this approach from a statistical point of view, though, is that poor control over the complexity of one's model typically leads to poor predictions; this is the statistical justification for "Occam's razor," the principle that simple explanations are preferred over complex.

A classical example of this phenomenon is shown in Fig. 3. We observe data generated by a smooth curve $g(.)$ (a sum of a few low-fequency sinusoids, in this case) plus i.i.d. Gaussian noise: thus, the i-th sample was given by

$$n_i = g(x_i) + \sigma\epsilon_i,$$

where ϵ_i is i.i.d. standard Gaussian noise. Then we fit a series of models of monotonically increasing complexity to this data: the p-th model class, Θ_p, is the set of all sums of sinusoids of integer frequency less than p. We see, as expected, that the *training error*

$$\frac{1}{N}\sum_{i=1}^{N} Err(g_p(x_i), n_i) = \frac{1}{N}\sum_{i=1}^{N} [\hat{g}_p(x_i) - n_i]^2$$

(with the sum taken over the observed samples n_i and the estimate $\hat{g}_p(.)$ constructed by linear least squares from sinusoids of maximal frequency p) decreases monotonically with the model complexity p, while the *generalization error*

$$E\left[Err(g_p(x), n)\right] = E\left[\hat{g}_p(x) - n\right]^2 = E_{x,\epsilon}\left[\hat{g}_p(x) - (g(x) + \sigma\epsilon)\right]^2$$

(where the expectation is now taken over the true underlying distribution of the data x and noise ϵ, instead of the observed samples) reaches its minimum at the true p (5 Hz in this case) and then increases for larger p. The explanation is that models with larger p fit the data very well at the observed sample points x_i at the expense of large oscillations where no data are observed (i.e., the *noise* has been fit well, not the underlying true function g). Thus the training error curve is completely misleading if we want to understand how well our estimator is actually generalizing, rather than just fitting the data.

A geometric analysis of this phenomenon is useful; our discussion here will be in terms of the linear regression model, but these ideas hold more generally. We may understand linear regression as an intersection of soft constraints, in the following sense. As we saw above, the loglikelihood is simply the quadratic form

$$\log(D|X, \vec{k}) = c - \frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(n_i - \vec{k}^t\vec{x}_i\right)^2,$$

which may be rewritten as

$$-\vec{k}^T A\vec{k} + \vec{b}^T\vec{k} + c,$$

where

$$A \propto \sum_i \vec{x}_i\vec{x}_i^T = X^T X,$$

$$\vec{b} \propto 2\sum_i n_i\vec{x}_i,$$

and c is a constant which does not affect the location of the optimal $\vec{k}$. Note that each observation $\vec{x}_i$ contributes a rank-one matrix to the resulting $X^T X$ matrix. The geometric interpretation of this sum of rank-one matrices is illustrated in Fig. 4: each sample contributes a term $(\vec{k}^T\vec{x}_i - n_i)^2$ to the overall cost function which serves to constrain the optimal $\vec{k}$ along a single direction in $\vec{k}$-space (because the matrix $\vec{x}_i\vec{x}_i^T$ is of rank one), and we obtain the optimal solution by forming a weighted intersection of these constraints. The directions of low curvature of the resulting quadratic surface are the directions for which $\vec{k}$ is poorly constrained, and which will result in an estimate of $\vec{k}$ which will be highly variable in these directions (in the sense that slight changes in $\vec{n}$ will cause large variations in the least-squares estimate $\hat{k}_{LS}$). Unconstrained directions correspond to zero curvature — infinitely long flat valleys in the cost function. The overfitting phenomenon we observed in Fig. 3 results in exactly this unconstrained case (e.g., when the dimension of $\vec{k}$ is large compared to the number of available samples).

We can translate this geometric intuition into algebra fairly easily: $\hat{k}_{LS}$ is defined as $\hat{k}_{LS} = (X^T X)^{-1}(X^T\vec{n})$, so if $cov(\vec{n}) = \sigma^2 I$, then

$$Cov(\hat{k}_{LS}) = (X^T X)^{-1}X^T\sigma^2 I X(X^T X)^{-1} = \sigma^2(X^T X)^{-1}.$$

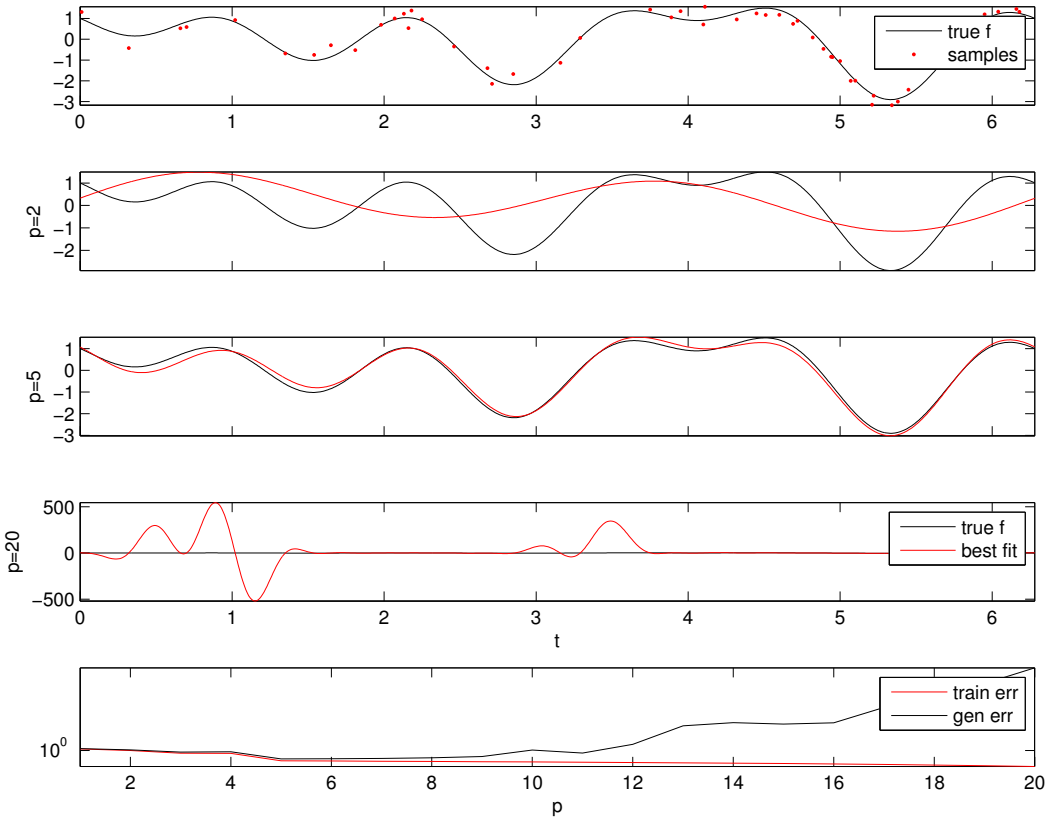

Figure 3: Overfitting demonstration: test vs. generalization error. **Top**: the true function $g(.)$ (solid black) was a sum of three sinusoids, of frequency 2, 3, and 5 Hz. We observe 50 noisy (zero-mean Gaussian; sd=0.3) samples from this true function (red dots). **Middle**: best-fitting function $\hat{g}_p(.)$ (red trace) versus true function $g(.)$ (black), using all sines and cosines of nonnegative integer frequencies, up to and including a maximum frequency $p = 2$, 5, and 20 Hz. Note that the estimate is oversmoothed when the maximal frequency $p = 2$ Hz, and badly overfit for $p = 20$. **Bottom**: training and generalization error as a function of maximal frequency p. Note that the generalization error achieves a minimum at the true value $p = 5$, and increases for higher p (overfitting). The training error decreases monotonically, as it must. (Error curves averaged over 100 i.i.d. experiments; note log axis.)

If we express this covariance matrix in terms of its eigenvectors (principal components), $(X^T X)^{-1} = OD^{-1}O^T$, then we see again that the directions of low curvature (small eigenvalues of $X^T X$, i.e., small values of the diagonal matrix D) will correspond exactly to directions of large variance.

These arguments help to explain how overfitting arises, and give us some intuition into what is going on. But how can we avoid overfitting? We describe several methods below. But the major concept to keep in mind (c.f. Fig. 3) is that the training error is misleading; when fitting a model to data we need to quantify the performance of the model in predicting data that was not used to train the model. This practice of quantifying the model's performance on a "test" set of data which is held completely distinct and independent of the "training"

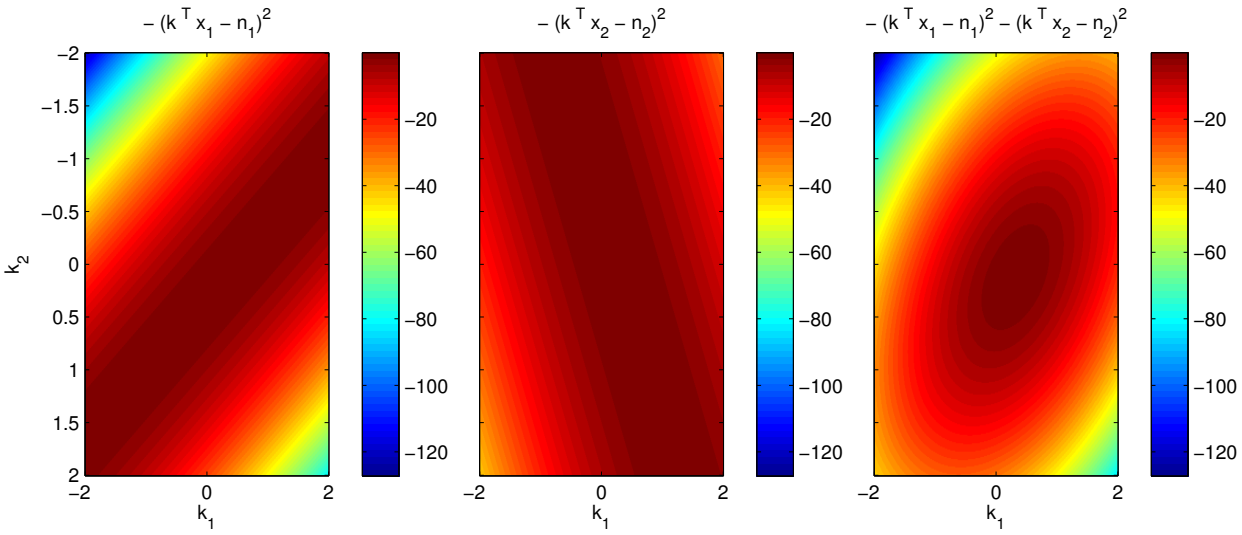

Figure 4: Geometry of least-squares: soft constraints. The left two panels show the cost function $-(\vec{k}^T\vec{x}_i - n_i)^2$ as a function of $\vec{k}$ given two observed data points, $(\vec{x}_1, n_1)$ and $(\vec{x}_2, n_2)$, while the right panel shows the sum of these two terms. Each individual sample $(\vec{x}_i, n_i)$ acts as a soft consraint, restricting $\vec{k}$ in the direction parallel to $\vec{x}_i$ (but not restricting $\vec{k}$ at all in any of the $\dim(\vec{k}) - 1$ other directions); by combining the available data, (i.e., forming what amounts to the weighted intersection of these soft constraints), we obtain a well-defined minimum. Note that the resulting confidence ellipse is tilted in the direction of the first constraint, which was stronger in this case; despite the fact that the constraints were nonorthogonal, the principal axes of the resulting confidence ellipse are orthogonal (as they must be, since these correspond to he eigenvectors of the symmetric matrix $X^T X$).

set is called "cross-validation." Thus, to compare the performance of two distinct models, we might fit both models on the same training set, then compute the likelihood of a completely distinct test data set under each of the two models: the model with the higher test likelihood may be considered a better model, in that it is able to predict the responses of the neuron to novel stimuli more accurately.

6 Reducing the number of free parameters by choosing a suitable parameter subspace can increase the prediction accuracy

As discussed above, it is well-known that estimates of the receptive field $\vec{k}$ based on spike-triggered averaging can be quite noisy when $\vec{k}$ has many parameters (Sahani and Linden, 2003; Smyth et al., 2003); the noisiness of the estimate $\vec{k}_{LS}$ is roughly proportional to the dimensionality of $\vec{k}$ (the number of parameters in $\vec{k}$ that we need to estimate from data) divided by the total number of observed samples (Paninski, 2003). A variety of methods have been introduced to "regularize" the estimated $\vec{k}$, to incorporate prior knowledge about the shape and/or magnitude of the true $\vec{k}$ to reduce the noise in $\vec{k}_{LS}$. In each case, the goal is to reduce the variance associated with estimating a large number of parameters, at the possible

expense of an increase in bias due to a reduction in the flexibility of the model.

One basic idea is to restrict $\vec{k}$ to lie within a lower-dimensional subspace,

$$\vec{k} = \sum_l a_l \vec{k}_l,$$

where $\vec{k}_l$ denotes the l-th basis element (fixed *a priori*); we then employ the same fitting procedure to estimate the coefficients a_l of $\vec{k}$ within this lower-dimensional basis. Plugging in this formula for $\vec{k}$, we have

$$\vec{k}^T \vec{x}_t = (\sum_l a_l \vec{k}_l)^T \vec{x}_t = \sum_l a_l (\vec{k}_l^T \vec{x}_t) = \vec{a}^T \vec{y}_t,$$

where

$$\vec{y}_l = \vec{k}_l^T \vec{x}.$$

Thus fitting these new parameters $\vec{a}$ proceeds in exactly the same fashion as before: we set up a design matrix, $X_t = \vec{y}_t$, and optimize the loglikelihood with respect to the parameters $\theta = \vec{a}$. The goal is to choose a basis whose span contains the "shapes" we might expect $\vec{k}$ to take (in order to minimize the bias associated with restricting our attention to a lower-dimensional subspace of possible $\vec{k}$), with as few basis elements (smallest dimensionality) as possible (since the variance of the estimated $\vec{k}$ is roughly proportional to the dimensionality). Of course, this restriction also increases the computational efficiency of the fitting procedure, since computation time increases with the dimensionality of X_t.

The choice of a suitable basis is problem dependent, but some basic principles are often followed in practice. For example, we often have good a priori knowledge about the smoothness of the filter $\vec{k}$: in this case, it is common to represent $\vec{k}$ in a Fourier or wavelet basis, with the very high- (and/or low-) frequency elements removed from the basis. We may take a similar approach using an orthogonal basis defined by principal components analysis (PCA): the idea is to match our basis for $\vec{k}$ to those directions in $\vec{x}$-space with high variance, which may be described in terms of the eigenvectors of the prior covariance matrix of $\vec{x}$. (In the case that $\vec{x}$ are drawn from a shift-stationary distribution, we will see below that the Fourier- and PCA-based approaches coincide.) Some other bases which have proven useful in practice include the Hermite basis (Victor et al., 2006), the stretched-cosine basis introduced by Keat et al. for the representation of temporal receptive fields (Keat et al., 2001; Pillow et al., 2005; Pillow et al., 2008), and the Zernike basis for receptive fields defined on a circle (Barbieri et al., 2004).

In general, it is helpful to choose the basis in such a way that the resulting design matrix X (expressed in the new basis) is close to orthogonal, i.e., that the matrix $(X^T X)$ has a small condition number. This increases the numerical stability as well as interpretability of the resulting estimate $\hat{\theta}_{ML}$.

7 Regularization provides a "softer" method for incorporating prior information and avoiding overfitting: maximum penalized likelihood and maximum *a posteriori* estimation

Above we discussed one tractable way to avoid overfitting, by restricting our attention to a linear subspace or submanifold of the full parameter space. This corresponds to enforcing

"hard" constraints on the acceptable parameter values. A slightly less restrictive approach is to use "soft" constraints instead — that is, to penalize some parameters but not to disallow them entirely. This penalization may be interpreted easily in Bayesian terms: instead of maximizing the loglikelihood $\log p(D|X,\vec{k})$ directly (which can lead to overfitting), we maximize the logarithm of the *posterior*

$$\log p(\vec{k}|X,D) = c + \log p(D|X,\vec{k}) + \log p(\vec{k})$$

(with $\vec{k}$ allowed to take values in the full original parameter space, i.e., no hard constraints have been imposed); here $p(\vec{k})$ encodes our *a priori* beliefs about the true underlying $\vec{k}$, and if we set $-Q(\vec{k}) = \log p(\vec{k})$, we see that $Q(\vec{k})$ acts as a kind of "penalty function," encoding our preferences in more *a priori* probable values of $\vec{k}$ (or equivalently, penalizing less probable values).

In the linear regression case, the computationally-simplest prior is a zero-mean Gaussian,

$$\log p(\vec{k}) = c - \vec{k}^T A\vec{k}/2,$$

where A is a positive definite matrix (the inverse prior covariance matrix); maximizing the corresponding log-posterior

$$\log p(\vec{k}|X,D) = c + \log p(D|X,\vec{k}) + \log p(\vec{k}) = c - \frac{1}{2\sigma^2}||X^T\vec{k} - \vec{n}||_2^2 - \frac{1}{2}\vec{k}^T A\vec{k}$$

analytically leads directly to the regularized least-square estimator

$$\vec{k}_{RLS} = (X^T X + \sigma^2 A)^{-1} X^T \vec{n}$$

(Sahani and Linden, 2003; Smyth et al., 2003).

One of the most common penalties acts to smooth the resulting estimate (Smyth et al., 2003). For example, our prior might express that smooth $\vec{k}$ are more common than rapidly changing or highly fluctuating $\vec{k}$. If we express this prior in the Gaussian form described above, $\log p(\vec{k}) = c - \vec{k}^T A\vec{k}/2$, then we might choose the matrix A such that

$$\vec{k}^T A\vec{k} = \sum_i [k(i) - k(i+1)]^2 = ||D\vec{k}||_2^2 = \vec{k}^T D^T D\vec{k},$$

with D denoting the discrete difference matrix, i.e., large changes between adjacent elements $\vec{k}(i)$ of $\vec{k}$ are penalized. Clearly A here may be written as $A = D^T D$, which turns out to correspond to the symmetric second difference matrix. (Of course it is also possible to penalize higher-order derivatives.)

It is also worth mentioning how to implement this penalty in the case that $\vec{k}$ is expressed in some alternative basis, $\vec{k} = \sum_l a_l \vec{k}_l = K\vec{a}$ for a suitable basis matrix K. If we write out the penalty in this case,

$$\vec{k}^T A\vec{k} = \vec{k}^T D^T D\vec{k} = \vec{a}^T K^T D^T DK\vec{a} = \vec{a}^T B\vec{a},$$

where the elements of the matrix $B = K^T D^T DK$ are given by the inner product of the differenced basis elements,

$$B_{l,l'} = (D\vec{k}_l)^T (D\vec{k}_{l'}).$$

Note that precomputing B and maximizing the posterior with respect to the parameters $\vec{a}$ is typically more computationally efficient than recomputing $\vec{k} = K\vec{a}$ on each iteration.

More generally, if $\log p(\vec{k})$ is maximized at the point $\vec{k} = \vec{0}$, the MAP estimator will basically be a more "conservative" version of the MLE, with the chosen coefficients shrunk nonlinearly towards zero. This type of "shrinkage" estimator has been extremely well-studied, from a variety of viewpoints (James and Stein, 1960; Donoho et al., 1995; Klinger, 1998; Tipping, 2001; Ng, 2004), and is known, for example, to perform strictly better than the MLE in certain contexts: again, because this shrinkage can effect a large decrease in the variance of our estimator, at the expense of a small increase in the bias. See (Sahani and Linden, 2003; Machens et al., 2003; Harris et al., 2003) for some illustrations of this effect.

The simplest version of this shrinkage idea is to choose the matrix A in the quadratic form for Q to be proportional to the identity. Thus we are penalizing the magnitude $\vec{k}^T\vec{k}$ directly instead of the magnitude of $D\vec{k}$, as in the smoothing case. This form of direct "shrinkage" has been studied extensively and is also known as "ridge regression" or Tikhonov regularization, depending on the literature. Note that the eigenstructure of $X^TX + \lambda I$ is easily derived from that of X^TX: the eigenvectors are exactly the same, and the eigenvalues are merely changed by the constant value λ. The key fact is that any zero eigenvalues in X^TX which might have caused problems in computing the inverse $(X^TX)^{-1}$ are now strictly positive, making the inverse much more stable and insensitive to noise.

Another very common penalizer is based on the L_1 norm of $\vec{k}$, $Q(\vec{k}) = \sum_i |\vec{k}(i)|$, instead of the L_2 norms we have discussed above[2]. This L_1 penalty is often used as a "sparseness" penalty: in many cases, we might believe that many of the elements of $\vec{k}$ are exactly zero, i.e., that only a "sparse" subset of $\vec{k}$ are actually active. One reasonable penalty to enforce sparseness would be the so-called L_0 norm,

$$||\vec{k}||_0 = \sum_i \delta(\vec{k}_i) = \lim_{p \to 0} ||\vec{k}||_p$$

(where as usual the Dirac delta function $\delta(.)$ is one at zero and zero everywhere else); unfortunately, this L_0 norm is nonconvex and the resulting minimization problem is often plagued by multiple local optima. The absolute value function $|x|$ is in a sense the closest convex function to the discontinuous function $1(x = 0)$, and so we often use the L_1 norm to impose sparseness. A great deal of recent research has focused on the properties of this L_1 penalty (also known as the "LASSO" in the statistics literature (Donoho et al., 1995; Tibshirani, 1996)); for example, it has recently been established that, under certain circumstances, the L_0- and L_1-penalized regression problems have exactly the same solution, i.e., the L_1 term really does serve to sparsen (Donoho and Elad, 2003).

It is worth comparing the L_1 versus L_2 penalization in a simple one-dimensional case, in order to gain some intuition into the behavior of these penalizers. As always, we turn to the quadratic-loss case for simplicity: if our original loglikelihood can be written in the form

$$-\frac{a}{2}\theta^2 + b\theta + c$$

for some coefficients (a, b, c), with $a > 0$, then adding an L_2 penalty term on θ, $-(a_0/2)\theta^2$,

[2]The L_p norm of a vector $\vec{k}$ is a measure of the magnitude of $\vec{k}$, defined as

$$||\vec{k}||_p = \left(\sum_i |k(i)|^p \right)^{1/p}.$$

For $p \geq 1$, this is a convex function of $\vec{k}$.

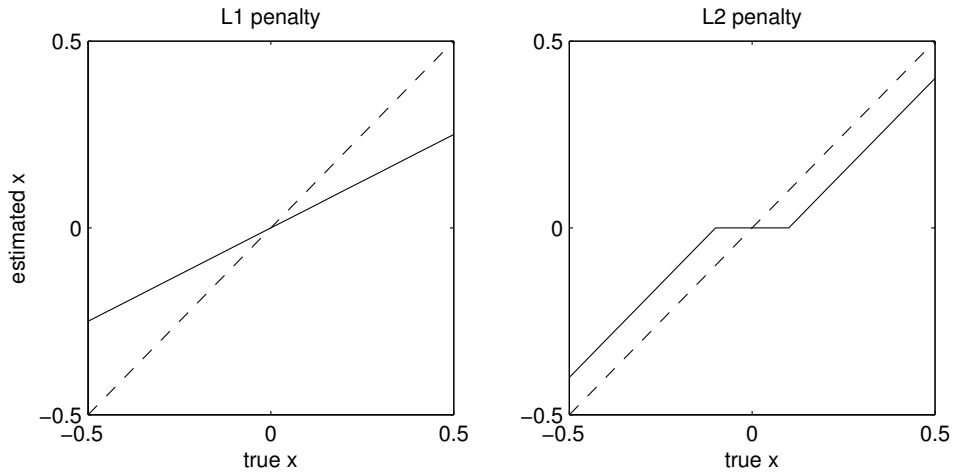

Figure 5: Comparison of L_2 and L_1 penalties for one-dimensional observations x. Note that the L_2 penalty changes the slope of the line $\hat{x}(x)$ (i.e., larger x are shrunk more), while the L_1 penalty leads to threshold behavior in $\hat{x}$ (estimates corresponding to small x are set to zero, but medium and large x are shrunk equally). Identity line (dashed trace) shown for comparison.

corresponds to changing the objective function to

$$-\frac{a+a_0}{2}\theta^2 + b\theta + c,$$

with the penalized optimum $\theta = b/(a+a_0)$, as compared to the unpenalized optimal $\theta = b/a$. Thus we see that the L_2 penalty term simply shrinks the optimal θ by a multiplicative factor of $a/(a+a_0)$ — and therefore, the larger the optimal original θ is, the larger the absolute shrinkage will be.

The L_1 penalty term behaves differently in two respects. First, instead of a multiplicative shrinkage we have an additive shrinkage: the total shrinkage does not increase as a function of the absolute value of the unpenalized optimizer. Second, and related, it is easy to see that the L_1 optimizer has a threshold nature: if the unpenalized θ has small enough magnitude, θ will be shrunk all the way to zero. (Clearly the L_2 penalty will never set θ to zero exactly, unless the original unpenalized θ is itself zero.) The optimizer may easily be computed analytically in this one-dimensional case; see Fig. 5 for a comparison between the L_1 and L_2 solutions in the one-dimensional setting.

In the case of multidimensional $\vec{k}$, the geometric interpretation of the two penalties is helpful and fairly natural. See Fig. 6 for an illustration: the key point is that the L_2 is radially symmetric — no elements of $\vec{k}$ are preferred, and thus in a sense all directions are shrunk equally. In the L_1 case, this symmetry no longer holds: it is clear that sparse solutions are favored, since the L_1 penalty is smaller along the coordinate axes (where some components of $\vec{k}$ are set to zero) than along other directions.

This regularization approach does have a somewhat unpleasant side effect. While a smoothing penalty can greatly improve the shape of the resulting estimates, and the L_1 approach can perform feature selection (by "turning off" those features $k(i)$ that do not contribute any predictive power), both of these approaches also result in a "shrunk" estimate: the

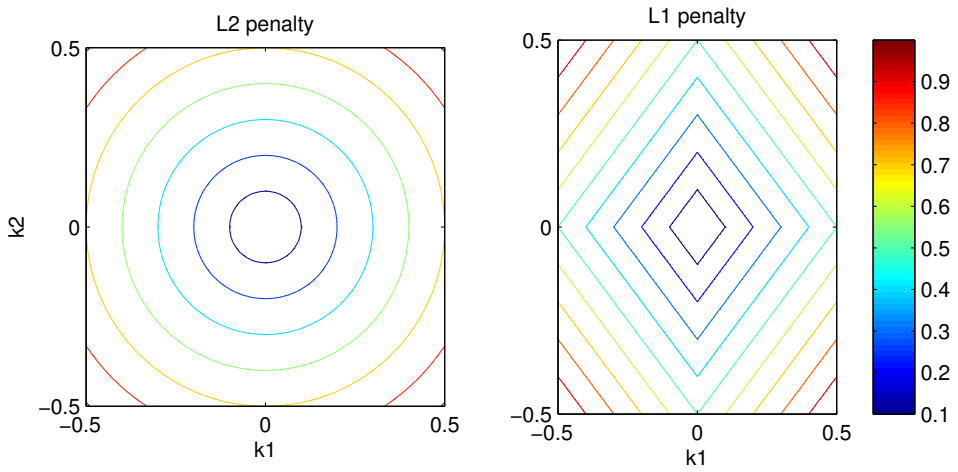

Figure 6: Comparison of L_2 and L_1 penalties for a two-dimensional $\vec{k}$. Note the radial symmetry of the L_2 penalty and the preference for the coordinate axes (the "corners") in the L_1 case.

overall magnitude of the estimate is often reduced. We can often achieve better generalization performance if we undo this shrinkage, while maintaining the shape or the "correct" features obtained by the regularization approach. (Of course, in the simplest L_2 case, undoing this shrinkage is a bad idea, since it undoes the symmetric shrinkage which was the whole point of the L_2 penalty.) Luckily, undoing this unwanted shrinkage is often fairly straightforward. For example, in the context of a smoothing penalty, we simply optimize the likelihood along a one-dimensional line corresponding to a magnitude $\alpha > 0$: that is, we choose $\hat{k}_0$ as the optimizer of the log-posterior $\log p(\vec{k}|X, D)$, but then choose our final estimate $\hat{k} = \alpha_{opt}\hat{k}_0$, where a_{opt} is the solution to the one-dimensional concave optimization problem

$$\alpha_{opt} = \arg\min_{\alpha>0} \log p(D|X, \alpha\hat{k}),$$

where note we are performing this last linesearch over the unpenalized likelihood, not the full posterior. This retains the smooth shape of the estimate but does not result in a reduced magnitude. Similarly, if we have used an L_1 penalty to choose a predictive subset of features $k(i)$, we may undo the L_1 shrinkage by performing an unpenalized "post-fit" on the subspace spanned by this reduced subset of features. See (Bühlmann and van de Geer, 2011) for a much more detailed discussion of these issues.

Finally, it is quite common to solve a slightly more general problem: instead of maximizing the log-posterior we might instead minimize

$$\log p(D|X, \vec{k}) - \lambda Q(\vec{k}), \tag{3}$$

where $\lambda > 0$ is a free "regularization parameter": for λ large, we penalize strongly, while for $\lambda \to 0$ we recover the unregularized maximum likelihood solution, and $\lambda = 1$ gives us the original MAP solution. Varying this parameter λ gives us some extra flexibility, but of course we need some way of selecting the best value of this parameter: this may be done either by cross-validation (Machens et al., 2003; Smyth et al., 2003) or by an approach known as

"evidence optimization" (Tipping, 2001; Sahani and Linden, 2003), a somewhat more involved technique based on integrating out hyperparameters in a Bayesian hierarchical framework; we will discuss related methods at more length below.

A full discussion of the computational problem of finding good solutions to problem (3), when $Q(\vec{k})$ is some convex penalty function, is outside the scope of this chapter. However, a few points are worth noting here. As emphasized above, in the case that both $Q(\vec{k})$ and $\log p(D|X,\vec{k})$ are quadratic functions of $\vec{k}$, an analytic solution is available, and more generally if both $Q(\vec{k})$ and $\log p(D|X,\vec{k})$ are concave and smooth then standard approaches based on conjugate gradient ascent or Newton's method typically suffice. However, in the case that $Q(\vec{k})$ imposes an L_1 penalty the resulting objective function is not everywhere differentiable: it has "corners" that cause problems for optimization techniques that assume that the objective function is twice-differentiable. This type of problem has attracted a good deal of attention in the recent optimization theory literature (Boyd and Vandenberghe, 2004). Three general approaches have proven useful. First, we can always replace a nonsmooth objective function f with a sequence of smooth functions f_i that approximate the desired objective: i.e., $f_i \rightarrow f$, in some suitable sense. Then we can use standard smooth methods to optimize each of the approximate functions; it is often easy to show that the optimizers to the approximate smooth functions will approach the desired optimizer of the original objective function, i.e., $\arg\max f_i \rightarrow \arg\max f$. This smooth approximation approach is especially useful in the context of constrained optimization; see (Boyd and Vandenberghe, 2004) and the examples in the later chapters for further discussion of this approach. Second, in some cases a coordinate ascent approach — in which we sequentially optimize along just one coordinate axis at a time — turns out to be surprisingly effective, particularly in cases where each coordinate optimization can be computed analytically and certain sparse features of the underlying problem can be exploited; see, e.g., (Friedman et al., 2010) for further discussion. Finally, in many cases we would like to solve problem (3) not just for one value of the regularization parameter λ, but for many values of λ simultaneously, so that the corresponding solutions, with different degrees of regularization, can be compared. We can always use a "warm start" technique, i.e., to initialize our search for an optimizer for a given value of λ at the precomputed value of $\vec{k}$ which maximizes (3) for a nearby value of λ. It turns out that in some cases we can exploit the structure of $Q(\vec{k})$ or $\log p(D|X,\vec{k})$ to follow the "solution path" $\vec{k}_\lambda$ in a semi-analytical and very computationally-efficient fashion, where $\vec{k}_\lambda = \arg\max \log p(D|X,\vec{k}) - \lambda Q(\vec{k})$; (Efron et al., 2004) discusses a very influential example of this idea.

8 Rank-penalizing and group LASSO penalties provide a useful method for regularizing matrix-valued parameters

In many cases the parameters we are interested in estimating can be best organized in terms of a matrix. Specialized penalization methods are often appropriate here. As a first example, imagine for concreteness that we are fitting a regression model for the firing rate of cell i, in which we are incorporating inputs from other cells i'. We might believe that the connectivity from neurons i' to i is sparse in the sense that only a few cells i' are connected to i. We might fit a matrix $h_i(i',\tau)$ of inputs, indexed by time delay τ and cell i'. Enforcing sparseness in this matrix by an L_1 penalty $Q(h) = \sum_{i',\tau} |h_i(i',\tau)|$ does not give us exactly what we want, unfortunately: this might lead to a sparse $h_i(.,.)$ matrix overall (in the sense that only a few values $h(i',\tau)$ are nonzero), but may leave many cells i' connected to i (albeit at only a sparse

subset of delays τ). A better penalty in this case is

$$Q(h) = \sum_{i'} ||h_i(i', .)||_2 = \sum_{i'} \left(\int |h_i(i', t)|^2 dt \right)^{1/2} :$$

i.e., enforce sparseness on the number of i' terms for which $h_i(i', .)$ is nonzero at any time t (the L_2 norm in this case provides a nice, radially-symmetric way to detect departures from zero in the vectors $h_i(i', .)$). Because this penalty is just a sum of convex functions (the p-norm is convex for any $p \geq 1$), the penalty remains convex. This technique is referred as a "groupwise LASSO" in the statistics literature.

Several more matrix examples appear in the subsections below.

8.1 Example: low-rank approximations for spatiotemporal receptive fields

One common case that leads to a very large number of parameters involves the estimation of a spatiotemporal receptive field in vision (Sharpee et al., 2006; Butts and Paninski, 2006) or somatosensory studies, or a spectrotemporal receptive field in audition (Sahani and Linden, 2003; Gill et al., 2006). In each case, it is often a reasonable approximation to represent the STRF as a "separable" function of space and time (Fig. 7), that is, the product form

$$k(x, y, t) = k_s(x, y)k_t(t).$$

The gain here is that the number of parameters is reduced from ST to $S + T$, where S and T denote the number of parameters required to describe the spatial component $k_s(x, y)$ and the temporal component $k_t(t)$, respectively; typically $ST \ll S + T$.

To fit such a separable model, it is reasonable to employ a simple alternating maximization strategy: if we hold $\vec{k}_t$ fixed, then the log-likelihood is concave with respect to $\vec{k}_s$, and vice versa. In fact, in the linear regression setting, we can write the optimization with respect to $\vec{k}_t$ with $\vec{k}_s$ held fixed (or vice versa) as a regression problem, with a quadratic objective function, and solve each optimization via the corresponding normal equations. Unfortunately, we are no longer guaranteed to find a global maximum using this strategy, despite the fact that the loglikelihood is concave as a function of $\vec{k}$, since the class of receptive fields of this separable form does not form a convex set: the sum of two separable functions is typically not separable. In addition, we must place restrictions on the model to ensure identifiability, since clearly $\left(c\vec{k}_t(t), \frac{1}{c}\vec{k}_s(x, y)\right)$ specifies the same model as $\left(\vec{k}_t(t), \vec{k}_s(x, y)\right)$, for any $c \neq 0$; see (Ahrens et al., 2008) for details.

This separable receptive field idea can be quite useful. However, in many cases the receptive field is highly non-separable. (For example, consider a motion-detecting receptive field, $k(x, t) = g(x - vt)$, for some function $g(.)$ and velocity $v \neq 0$.) In such cases we may generalize the separable idea slightly. Consider a one-dimensional spatial variable for simplicity (we may always concatenate the x and y variables in general): we may represent $k(x, t)$ as a matrix K. A separable receptive field corresponds to a matrix of rank one:

$$K = \vec{k}_s \vec{k}_t^T.$$

A natural generalization is to consider a matrix of higher rank r:

$$K = K_s K_t^T,$$

where the vectors $\vec{k}_s$ and $\vec{k}_t$ above have been replaced by the matrices K_s and K_t, of size $S \times r$ and $T \times r$, respectively. As in the separable case, we may fit the model parameters by straightforward alternating maximization: with K_t held fixed, solve a regression to optimize the likelihood for K_s, and vice versa. (Once again, certain restrictions on the model are necessary to ensure identifiability.) If $(S + T)r \ll ST$, clearly we achieve a reduction in the number of parameters, often at no great loss of accuracy in the resulting model.

Because of the loss of our convergence guarantees in this low-rank model, the choice of initialization of the parameter search becomes somewhat more important. One useful approach is to take a preliminary estimate of the full-rank K and then to perform a singular value decomposition of the matrix $K = UWV$; then the first r rows of U serve to initialize K_s, while the first r columns (multiplied by the first r singular values in W) serve to initialize K_t.

An alternative approach is to optimize K directly, but use a penalization approach to force the solution to have low rank. One such penalty that has received quite a bit of recent attention is the "nuclear norm" $||K||_*$ of the matrix K, which is simply the sum of the singular values of K. This is a convex function of K, and acts to sparsen the singular values of K in much the same way as the L_1 penalty acts to sparsen vectors (Candès and Tao, 2010), as discussed in the previous section. Thus we proceed by maximizing the penalized log-likelihood $\log p(D|K, X) - \lambda||K||_*$; the advantage of this approach is that, in the cases of interest in this section, the resulting objective function is concave over a convex set (the set of matrices K), and therefore we do not have to worry about getting caught in local optima; a number of authors argue that this is a major practical advantage over the (non-convex) alternating maximization approach (see, e.g., (Mazumder et al., 2010) for further discussion). The disadvantage is that, as emphasized above, the dimensionality of K is much larger than that of K_s and K_t, and in addition the function $||K||_*$ is not everywhere differentiable as a function of K, which complicates the optimization somewhat. As in the L_1 case, nuclear-norm-penalized maximization problems have recently enjoyed a great deal of attention in the optimization and machine learning literatures. Again, it is beyond our scope to discuss algorithms for this optimization problem in depth, but one flexible strategy is to decompose the full penalized objective function into two simpler functions which can be optimized easily; typically in this approach one problem corresponds to a smoothly penalized likelihood optimization (which can be solved using standard techniques such as Newton's method or conjugate gradient), while the second my be solved by a simple SVD shrinkage operation. See (Goldfarb et al., 2010) and (Mazumder et al., 2010) for further details.

8.2 Example: "energy" models as low-rank Volterra series models

We may also apply this low-rank idea to the Volterra series analysis discussed above. Recall that the second-order Volterra model is

$$E[n_t] = b + \vec{k}^T\vec{x} + \vec{x}^T A\vec{x},$$

for some matrix A. As we discussed previously, we require a good deal of data to fit all the elements of the matrix A accurately; instead, we may assume that A is of low rank, $A = A_l A_r$, where A_l and A_r are of size $\dim(x) \times r$ and $r \times \dim(x)$, respectively. Once again, the loglikelihood is concave in the parameters $(b, \vec{k}, A_l)$ with A_r held fixed, or in $(b, \vec{k}, A_r)$ with A_l held fixed, and so either the alternating maximization or the nuclear-norm-penalty

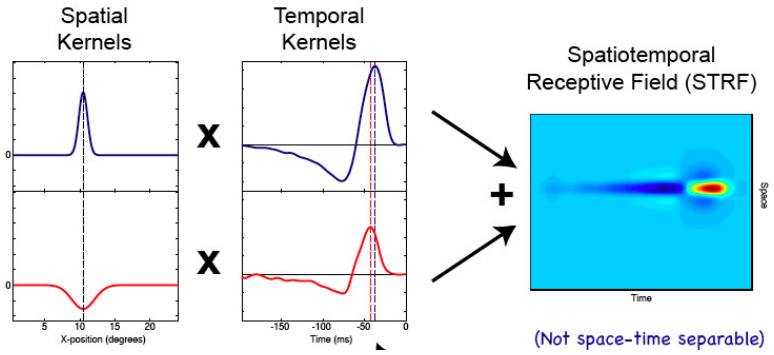

Figure 7: Low-rank models for the spatiotemporal visual receptive field of a neuron in the lateral geniculate nucleus. We model the full spatiotemporal receptive field as a sum of two rank-1 (separable) matrices, corresponding to a product of a temporal and spatial kernel (only one dimension of the 2-d spatial kernel is shown). This allows us to estimate the spatial and temporal kernels reliably in a generalized linear model framework using relatively few samples.

approach may be applied. (One small note: since

$$\vec{x}^T A_l A_r \vec{x} = \vec{x}^T A_r A_l \vec{x} = \vec{x}^T \left(\frac{A_l A_r + A_r A_l}{2} \right) \vec{x},$$

we are actually fitting a rank-$2r$ model for A here, instead of the usual rank-r model, since the matrix $(A_l A_r + A_r A_l)/2$ has rank $2r$.)

One key example of this low-rank Volterra model is the classical "energy model" for complex cells in primary visual cortex (Adelson and Bergen, 1985; Okajima and Imaoka, 2001). In this model the firing rate is given by

$$E[n_t] = (\vec{k}_1^T \vec{x})^2 + (\vec{k}_1^T \vec{x})^2,$$

where the linear filters $\vec{k}_1$ and $\vec{k}_2$ are in quadrature pair (and therefore the response of the model is invariant with respect to the phase of the stimulus $\vec{x}$). This may be rewritten in more standard Volterra form as

$$(\vec{k}_1^T \vec{x})^2 + (\vec{k}_1^T \vec{x})^2 = \vec{x}^T A \vec{x} + b^T \vec{x},$$

with $A = \vec{k}_1 \vec{k}_1^T + \vec{k}_2 \vec{k}_2^T$ and $b = 0$; thus the energy model may be considered a rank-two Volterra model.

8.3 Example: estimating input nonlinearities

Another useful application appears in (Ahrens et al., 2008). In many cases we might not know a neuron's "preferred units" a priori (Gill et al., 2006): for example, it might make more sense to represent $\vec{x}$ in logarithmic instead of linear units, or perhaps the neuron's response is invariant with respect to the sign of $\vec{x}$; we would like to learn this representation directly from data. Thus we might fit an "input nonlinearity" model (Fig. 8), of the form

$$E[n_t | \vec{x}_t] = \sum_i a_i g(x_{t-i}),$$

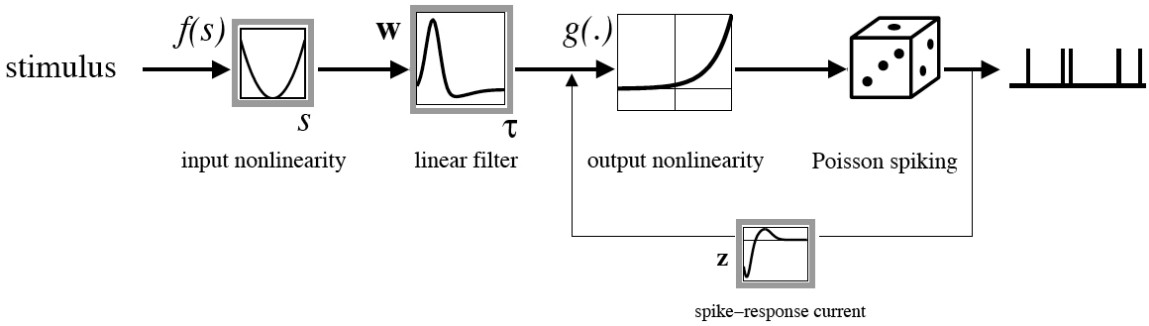

Figure 8: Schematic view of the bilinear "input nonlinearity" model. The parameters in the gray boxes are learnt from the data.

where x_t is the scalar input at time t, $g(.)$ is an unknown nonlinearity which transforms this input (for example, $g(.)$ could apply a logarithmic or squaring transformation), $\vec{a}$ is a temporal filter, and $f(.)$, as usual, is a convex and log-concave scalar function. To fit this model we represent the input nonlinearity function $g(.)$ as a weighted sum of some set of known functions $g_l(.)$,

$$g(u) = \sum_l b_l g_l(u),$$

and rewrite

$$E[n_t|\vec{x}_t] = \sum_i a_i g(x(t-i)) = \sum_i a_i \sum_l b_l g_l(x(t-i)) = \sum_{il} a_i b_l g_l(x(t-i)).$$

Now if we think of the fixed (known) stimulus terms $g_l(x(t))$ as elements of a matrix indexed by t and l, we may reinterpret the double sum

$$\sum_{il} a_{t-i} b_l g_l(x(t)) = \sum_{il} K_{il} g_l(x(t))$$

as a sum over a rank-one matrix $K = \vec{a}\vec{b}^T$, just as in the examples given above. Of course, it is now straightforward to generalize further, to let the matrix K be of rank r, for example, or to use the same trick to infer more complex linear-nonlinear-linear-etc. cascade models. See (Ahrens et al., 2008) for details.

8.4 Example: finding a good basis for estimating multiple receptive fields simultaneously

As a final example, imagine we have observed the responses of many neurons in a given brain region. We might expect many of these neurons to have similar tuning characteristics. Indeed, we would like to exploit any such similarities (for example, the more we know about a brain area *a priori*, the easier it should be to estimate the receptive fields of any new neurons we enounter in this area) and to quantify the heterogeneity of tuning properties in a given brain area, cortical layer, etc. How can we effectively "share" this kind of information across neurons?

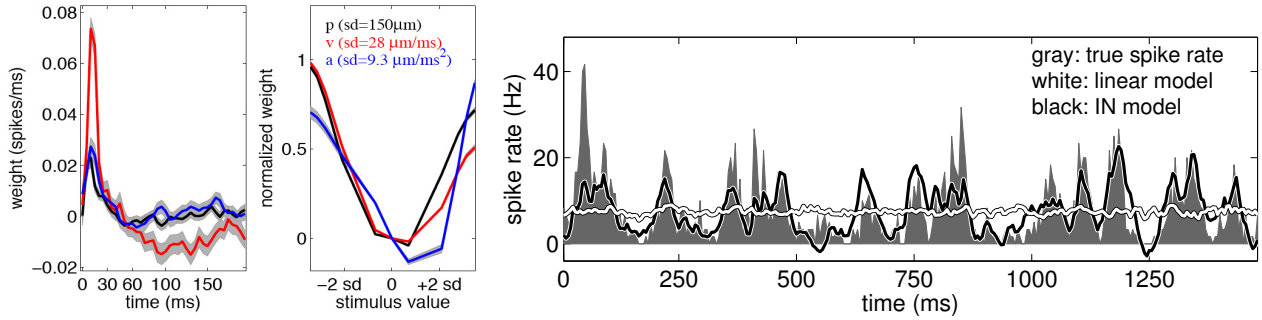

Figure 9: Input-nonlinearity models of a cortical whisker barrel neuron's responses (Ahrens et al., 2008) comparing three terms: position, velocity and acceleration. **Left**: temporal filters; pos., vel. and acc. terms in black, red, and blue, respectively. Velocity is the dominant term. **Middle**: inferred input nonlinearities; gray areas show 1 s.d. errorbar. Note that inferred nonlinearities are close to quadratic. **Right**: Predicted firing rates given a novel stimulus. Note accuracy of input-nonlinearity model predictions; linear model fails completely.

This basic idea has been quite successfully exploited in the statistics literature, in the context of "hierarchical" or "multilevel" models (Gelman and Hill, 2006). We will discuss such approaches at more length in a later chapter. However, for now we restrict our attention to a simple illustrative case. Let's introduce a simple linear model for each observed neuron:

$$E[n_t^i|\vec{x}_t] = \vec{k}_i\vec{x}_t,$$

where n_t^i denotes the i-th neuron's response at time t. (Nonlinear generalizations will be discussed at more length later.) We can represent the collection of linear filters $\{\vec{k}_i\}$ in matrix form, by simply concatenating the vectors $\vec{k}_i$ into a matrix K. Now we can impose a low-rank structure on K using methods basically identical to those discussed above; see (Yuan et al., 2007) for some statistical applications of this basic idea. In other words, we model K in terms of a low-rank projection, $K = UV$; V projects the stimulus $\vec{x}_t$ onto a low-dimensional subspace, and U weights the basis vectors of this subpsace appropriately to form the filters $\vec{k}_i$. The attractive feature of this approach is that we do not have to predefine the subspace V; instead, an optimal subspace is estimated directly from the data. (Geffen et al., 2009) discuss an application of related ideas, in which the filters $\vec{k}_i$ are estimated one-by-one, by standard regression (without sharing any information across neurons), and then PCA is applied to the matrix of the estimated filters $\vec{k}_i$ to obtain an interesting low-dimensional subspace in which the filters seem to concentrate most of their power.

9 *Regression methods are often used for neural decoding

Up to now we've been talking about encoding; to decode, just turn the regression around (Humphrey et al., 1970; Bialek et al., 1991).

10 *When decoding temporally-varying signals, it is useful to analyze the errors in the frequency domain

10.1 *The discrete Fourier transform performs harmonic regression across all available harmonic frequencies

10.2 *For a stationary time series, smoothing the periodogram produces an estimate of the spectrum

10.3 *Uncertainty following the discrete Fourier transform may be propagated to produce surrogate time series

References

Adelson, E. and Bergen, J. (1985). Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Am. A*, 2:284–299.

Ahrens, M., Paninski, L., and Sahani, M. (2008). Inferring input nonlinearities in neural encoding models. *Network: Computation in Neural Systems*, 19:35–67.

Barbieri, R., Frank, L., Nguyen, D., Quirk, M., Solo, V., Wilson, M., and Brown, E. (2004). Dynamic analyses of information encoding in neural ensembles. *Neural Computation*, 16:277–307.

Bialek, W., Rieke, F., de Ruyter van Steveninck, R., and Warland, D. (1991). Reading a neural code. *Science*, 252:1854–1857.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Oxford University Press.

Bühlmann, P. and van de Geer, S. (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer series in statistics. Springer.

Butts, D. and Paninski, L. (2006). Contrast adaptation in descriptions of visual neurons that incorporate spike-history dependence. *CNS*06 Meeting, Edinburgh*.

Candès, E. J. and Tao, T. (2010). The power of convex relaxation: near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56:2053–2080.

Chichilnisky, E. (2001). A simple white noise analysis of neuronal light responses. *Network: Computation in Neural Systems*, 12:199–213.

de Boer, E. and Kuyper, P. (1968). Triggered correlation. *IEEE Transactions on Biomedical Engineering*, 15:159–179.

Devroye, L. and Lugosi, G. (2001). *Combinatorial Methods in Density Estimation*. Springer-Verlag, New York.

Donoho, D. and Elad, M. (2003). Optimally sparse representation in general (nonorthogonal) dictionaries via L^1 minimization. *PNAS*, 100:2197–2202.

Donoho, D. L., Johnstone, I. M., Kerkyacharian, G., and Picard, D. (1995). Wavelet shrinkage: Asymptopia? *J. R. Statist. Soc. B.*, 57(2):301–337.

Duda, R. and Hart, P. (1972). *Pattern classification and scene analysis*. Wiley, New York.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, 32:407–499.

Friedman, J. H., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33:1–22.

Geffen, M. N., Broome, B. M., Laurent, G., and Meister, M. (2009). Neural encoding of rapidly fluctuating odors. *Neuron*, 61:570–586.

Gelman, A. and Hill, J. (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge.

Gill, P., Zhang, J., Woolley, S., Fremouw, T., and Theunissen, F. (2006). Sound representation methods for spectro-temporal receptive field estimation. *Journal of Computational Neuroscience*, 21:5–20.

Goldfarb, D., Ma, S., and Scheinberg, K. (2010). Fast alternating linearization methods for minimizing the sum of two convex functions. *Columbia University IEOR Technical Report*.

Harris, K., Csicsvari, J., Hirase, H., Dragoi, G., and Buzsaki, G. (2003). Organization of cell assemblies in the hippocampus. *Nature*, 424:552–556.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer.

Humphrey, D., Schmidt, E., and Thompson, W. (1970). Predicting measures of motor performance from multiple cortical spike trains. *Science*, 170:758–762.

James, W. and Stein, C. (1960). Estimation with quadratic loss. *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, 1:361–379.

Keat, J., Reinagel, P., Reid, R., and Meister, M. (2001). Predicting every spike: a model for the responses of visual neurons. *Neuron*, 30:803–817.

Klinger, A. (1998). *High-dimensional generalized linear models*. PhD thesis, University of Munich.

Kutner, M., Nachtsheim, C., Neter, J., and Li, W. (2005). *Applied Linear Statistical Models*. McGraw-Hill.

Machens, C., Wehr, M., and Zador, A. (2003). Spectro-temporal receptive fields of subthreshold responses in auditory cortex. *NIPS*.

Marmarelis, P. and Marmarelis, V. (1978). *Analysis of physiological systems: the white-noise approach*. Plenum Press, New York.

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *J. Mach. Learn. Res.*, 11:2287–2322.

Ng, A. (2004). Feature selection, L_1 vs. L_2 regularization, and rotational invariance. *ICML*, 21.

Okajima, K. and Imaoka, H. (2001). A Complex Cell-Like Receptive Field Obtained by Information Maximization. *Neural Computation*, 13(3):547–562.

Paninski, L. (2003). Convergence properties of some spike-triggered analysis techniques. *Network: Computation in Neural Systems*, 14:437–464.

Paninski, L. (2004). Maximum likelihood estimation of cascade point-process neural encoding models. *Network: Computation in Neural Systems*, 15:243–262.

Paninski, L., Pillow, J., and Simoncelli, E. (2004). Maximum likelihood estimation of a stochastic integrate-and-fire neural model. *Neural Computation*, 16:2533–2561.

Pillow, J., Paninski, L., Uzzell, V., Simoncelli, E., and Chichilnisky, E. (2005). Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model. *Journal of Neuroscience*, 25:11003–11013.

Pillow, J., Shlens, J., Paninski, L., Sher, A., Litke, A., Chichilnisky, E., and Simoncelli, E. (2008). Spatiotemporal correlations and visual signaling in a complete neuronal population. *Nature*, 454:995–999.

Rust, N., Mante, V., Simoncelli, E., and Movshon, J. (2006). How MT cells analyze the motion of visual patterns. *Nature Neuroscience*, 11:1421–1431.

Sahani, M. (2000). Kernel regression for neural systems identification. Presented at NIPS00 workshop on Information and statistical structure in spike trains; abstract available at http://www-users.med.cornell.edu/∼jdvicto/nips2000speakers.html.

Sahani, M. and Linden, J. (2003). Evidence optimization techniques for estimating stimulus-response functions. *NIPS*, 15.

Scholkopf, B. and Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press.

Sharpee, T., Sugihara, H., Kurgansky, A., Rebrik, S., Stryker, M., and Miller, K. (2006). Adaptive filtering enhances information transmission in visual cortex. *Nature*, 439:936–942.

Smyth, D., Willmore, B., Baker, G., Thompson, I., and Tolhurst, D. (2003). The receptive-field organization of simple cells in primary visual cortex of ferrets under natural scene stimulation. *Journal of Neuroscience*, 23:4746–4759.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, 58:267–288.

Tipping, M. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244.

Truccolo, W., Eden, U., Fellows, M., Donoghue, J., and Brown, E. (2005). A point process framework for relating neural spiking activity to spiking history, neural ensemble and extrinsic covariate effects. *Journal of Neurophysiology*, 93:1074–1089.

Victor, J., Mechler, F., Repucci, M., Purpura, K., and Sharpee, T. (2006). Responses of V1 neurons to two-dimensional hermite functions. *Journal of Neurophysiology*, 95:379–400.

Yuan, M., Ekici, A., Lu, Z., and Monteiro, R. (2007). Dimension reduction and coefficient estimation in multivariate linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69:329–346.