Neural Encoding Models

Maneesh Sahani

Gatsby Computational Neuroscience Unit
University College London

February 2019

Neural coding

Neural Coding

The brain appears to process sensory information in a modular way. Different structures and cortical areas process, represent and transmit different aspects of the input.

Neural Coding

The brain appears to process sensory information in a modular way. Different structures and cortical areas process, represent and transmit different aspects of the input.

The coding questions:

- What information is represented by a particular neural population?
- How is that information encoded?

Neural Coding

The brain appears to process sensory information in a modular way. Different structures and cortical areas process, represent and transmit different aspects of the input.

The coding questions:

- What information is represented by a particular neural population?
  - easy (?) if we know the code
- How is that information encoded?

Neural Coding

The brain appears to process sensory information in a modular way. Different structures and cortical areas process, represent and transmit different aspects of the input.

The coding questions:

- What information is represented by a particular neural population?
  - easy (?) if we know the code
  - more generally, can search for selectivity / invariance (in invidual neurons in in populations)

- How is that information encoded?

Neural Coding

The brain appears to process sensory information in a modular way. Different structures and cortical areas process, represent and transmit different aspects of the input.

The coding questions:

- What information is represented by a particular neural population?
  - easy (?) if we know the code
  - more generally, can search for selectivity / invariance (in invidual neurons in in populations)
  - encoded quantities might not be obvious: inferred latent variables, uncertainty . . .
- How is that information encoded?

Neural Coding

The brain appears to process sensory information in a modular way. Different structures and cortical areas process, represent and transmit different aspects of the input.

The coding questions:

- What information is represented by a particular neural population?
  - easy (?) if we know the code
  - more generally, can search for selectivity / invariance (in invidual neurons in in populations)
  - encoded quantities might not be obvious: inferred latent variables, uncertainty . . .
- How is that information encoded?
  - firing rate, spiking timing (relative to other spikes, population oscillations, onset of time-invariant stimulus)?

Neural Coding

The brain appears to process sensory information in a modular way. Different structures and cortical areas process, represent and transmit different aspects of the input.

The coding questions:

- What information is represented by a particular neural population?
  - easy (?) if we know the code
  - more generally, can search for selectivity / invariance (in invidual neurons in in populations)
  - encoded quantities might not be obvious: inferred latent variables, uncertainty . . .
- How is that information encoded?
  - firing rate, spiking timing (relative to other spikes, population oscillations, onset of time-invariant stimulus)?
  - functional mapping of encoded variable to spikes?

Neural Coding

The brain appears to process sensory information in a modular way. Different structures and cortical areas process, represent and transmit different aspects of the input.

The coding questions:

- What information is represented by a particular neural population?
  - easy (?) if we know the code
  - more generally, can search for selectivity / invariance (in invidual neurons in in populations)
  - encoded quantities might not be obvious: inferred latent variables, uncertainty ...
- How is that information encoded?
  - firing rate, spiking timing (relative to other spikes, population oscillations, onset of time-invariant stimulus)?
  - functional mapping of encoded variable to spikes?
  - easy (?) if we know what is encoded

Neural Coding

The brain appears to process sensory information in a modular way. Different structures and cortical areas process, represent and transmit different aspects of the input.

The coding questions:

- What information is represented by a particular neural population?
  - easy (?) if we know the code
  - more generally, can search for selectivity / invariance (in invidual neurons in in populations)
  - encoded quantities might not be obvious: inferred latent variables, uncertainty ...
- How is that information encoded?
  - firing rate, spiking timing (relative to other spikes, population oscillations, onset of time-invariant stimulus)?
  - functional mapping of encoded variable to spikes?
  - easy (?) if we know what is encoded

A complete answer will require convergence of theory and empirical results.

Neural Coding

The brain appears to process sensory information in a modular way. Different structures and cortical areas process, represent and transmit different aspects of the input.

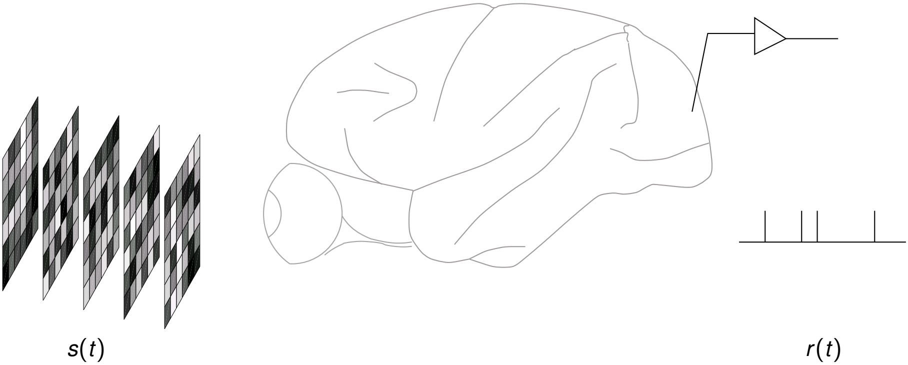
The coding questions:

- What information is represented by a particular neural population?
  - easy (?) if we know the code
  - more generally, can search for selectivity / invariance (in invidual neurons in in populations)
  - encoded quantities might not be obvious: inferred latent variables, uncertainty . . .
- How is that information encoded?
  - firing rate, spiking timing (relative to other spikes, population oscillations, onset of time-invariant stimulus)?
  - functional mapping of encoded variable to spikes?
  - easy (?) if we know what is encoded

A complete answer will require convergence of theory and empirical results.

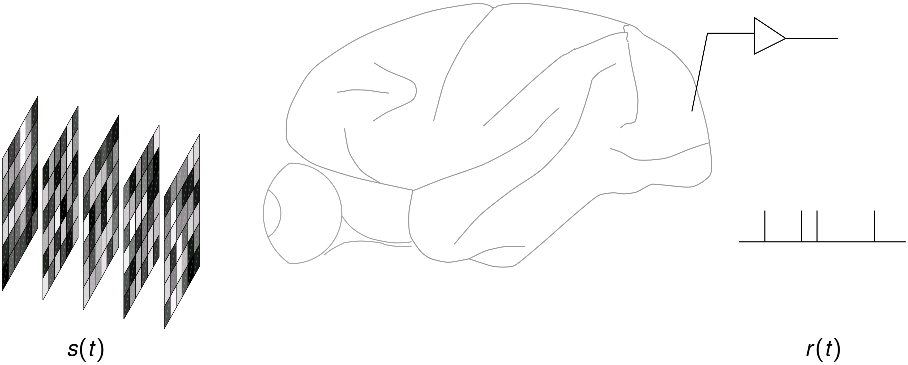
Computation plays a vital part in systematising empirical data.

Stimulus coding



Decoding: $\hat{s}(t) = G[r(t)]$ (reconstruction)

Stimulus coding



Decoding: $\hat{s}(t) = G[r(t)]$ (reconstruction)

Encoding: $\hat{r}(t) = F[s(t)]$ (systems identification)

Why?

The stimulus coding problem has sometimes been identified with the “neural coding” problem.

However, on the face of it, mapping *either* the decoding or encoding function does not by itself answer either of our basic questions about coding.

So why do we do it?

Why?

The stimulus coding problem has sometimes been identified with the “neural coding” problem.

However, on the face of it, mapping *either* the decoding or encoding function does not by itself answer either of our basic questions about coding.

So why do we do it?

- encapsulate and systematise the response so that we *can* ask the questions that we want answered.

Why?

The stimulus coding problem has sometimes been identified with the "neural coding" problem.

However, on the face of it, mapping *either* the decoding or encoding function does not by itself answer either of our basic questions about coding.

So why do we do it?

- encapsulate and systematise the response so that we *can* ask the questions that we want answered.
- design hypothesis-driven stimulus-coding models: evaluate coding reliability for different function(al)s of $s(t)$ and for different definitions of $r(t)$.

Why?

The stimulus coding problem has sometimes been identified with the "neural coding" problem.

However, on the face of it, mapping *either* the decoding or encoding function does not by itself answer either of our basic questions about coding.

So why do we do it?

- encapsulate and systematise the response so that we *can* ask the questions that we want answered.
- design hypothesis-driven stimulus-coding models: evaluate coding reliability for different function(al)s of $s(t)$ and for different definitions of $r(t)$.
- but correlation $\not\Rightarrow$ causation: in this case the *presence* of information about an aspect of the stimulus in a particular aspect of the response does not mean that the brain *uses* that information.

General approach

Goal: Estimate $p(\text{spike}|s, H)$ [or *intensity* $\lambda(t|s[0, t), H(t))$] from data.

General approach

Goal: Estimate $p(\text{spike}|s, H)$ [or *intensity* $\lambda(t|s[0, t), H(t))$] from data.

- Naive approach: measure $p(\text{spike}, H|s)$ directly for every setting of s.

General approach

Goal: Estimate $p(\text{spike}|s, H)$ [or *intensity* $\lambda(t|s[0, t), H(t))$] from data.

- Naive approach: measure $p(\text{spike}, H|s)$ directly for every setting of s.
  - too hard: too little data and too many potential inputs.

General approach

Goal: Estimate $p(\text{spike}|s, H)$ [or *intensity* $\lambda(t|s[0, t), H(t))$] from data.

- Naive approach: measure $p(\text{spike}, H|s)$ directly for every setting of s.
  - too hard: too little data and too many potential inputs.
- Estimate some functional $F[p]$ instead (e.g. mutual information)

General approach

Goal: Estimate $p(\text{spike}|s, H)$ [or *intensity* $\lambda(t|s[0,t), H(t))$] from data.

- Naive approach: measure $p(\text{spike}, H|s)$ directly for every setting of s.
  - too hard: too little data and too many potential inputs.
- Estimate some functional $F[p]$ instead (e.g. mutual information)
- Select stimuli efficiently

General approach

Goal: Estimate $p(\text{spike}|s, H)$ [or *intensity* $\lambda(t|s[0,t), H(t))$] from data.

- Naive approach: measure $p(\text{spike}, H|s)$ directly for every setting of s.
  - too hard: too little data and too many potential inputs.
- Estimate some functional $F[p]$ instead (e.g. mutual information)
- Select stimuli efficiently
- Fit models with smaller numbers of parameters

Spikes, or rate?

Most neurons communicate using action potentials — statistically described by a point process:

$$P\big(\text{spike} \in [t, t+dt)\big) = \lambda(t|H(t), \text{stimulus}, \text{network activity})dt$$

To fully model the response we need to identify λ. In general this depends on spike history $H(t)$ and network activity. Three options:

Spikes, or rate?

Most neurons communicate using action potentials — statistically described by a point process:

$$P\big(\text{spike} \in [t, t+dt)\big) = \lambda(t|H(t), \text{stimulus}, \text{network activity})dt$$

To fully model the response we need to identify λ. In general this depends on spike history $H(t)$ and network activity. Three options:

- Ignore the history dependence, take network activity as source of "noise" (i.e. assume firing is inhomogeneous Poisson or Cox process, conditioned on the stimulus).

Spikes, or rate?

Most neurons communicate using action potentials — statistically described by a point process:

$$P\big(\text{spike} \in [t, t + dt)\big) = \lambda(t|H(t), \text{stimulus}, \text{network activity})dt$$

To fully model the response we need to identify λ. In general this depends on spike history $H(t)$ and network activity. Three options:

- Ignore the history dependence, take network activity as source of "noise" (i.e. assume firing is inhomogeneous Poisson or Cox process, conditioned on the stimulus).
- Average multiple trials to estimate the mean intensity (or PSTH)

  $$\overline{\lambda}(t, \text{stimulus}) = \lim_{N\to\infty} \frac{1}{N} \sum_n \lambda(t|H_n(t), \text{stimulus}, \text{network}_n)\,,$$

  and try to fit this.

Spikes, or rate?

Most neurons communicate using action potentials — statistically described by a point process:

$$P\big(\text{spike} \in [t, t+dt)\big) = \lambda(t|H(t), \text{stimulus}, \text{network activity})dt$$

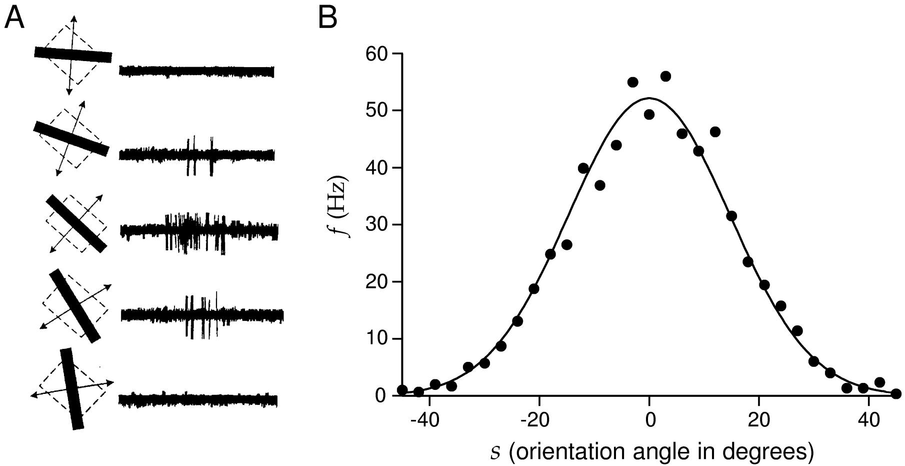
To fully model the response we need to identify λ. In general this depends on spike history $H(t)$ and network activity. Three options:

- Ignore the history dependence, take network activity as source of "noise" (i.e. assume firing is inhomogeneous Poisson or Cox process, conditioned on the stimulus).
- Average multiple trials to estimate the mean intensity (or PSTH)

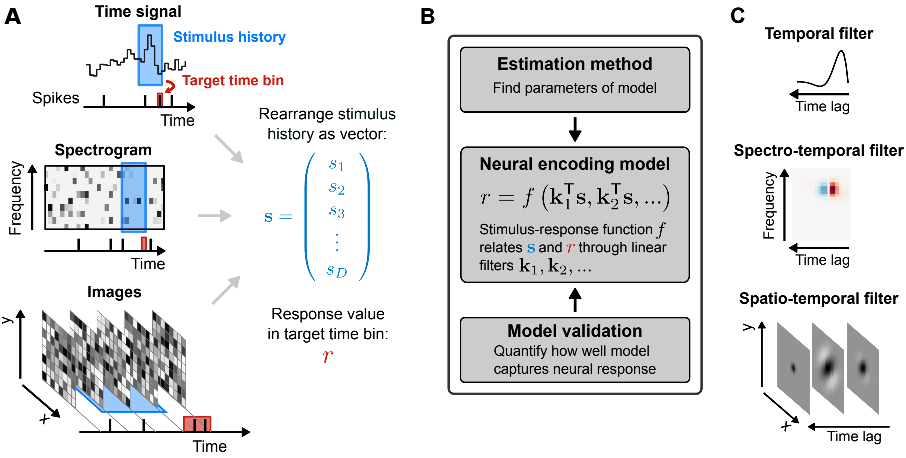
  $$\overline{\lambda}(t, \text{stimulus}) = \lim_{N\to\infty} \frac{1}{N} \sum_n \lambda(t|H_n(t), \text{stimulus}, \text{network}_n) \, ,$$

  and try to fit this.
- Attempt to capture history and network effects in simple models.

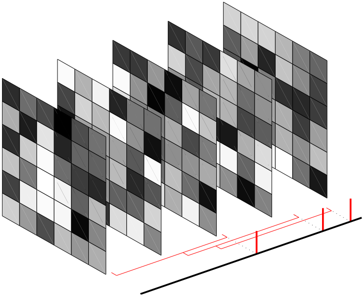
Tuning – stationary stimuli

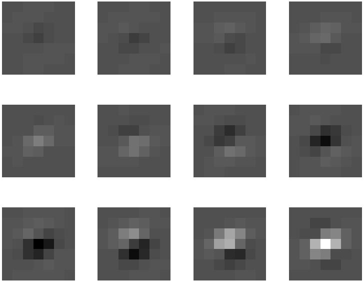


(Nonlinear) filtering – dynamic stimuli



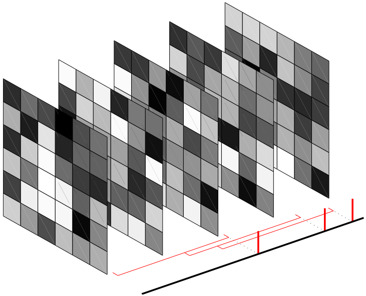
Spike-triggered average

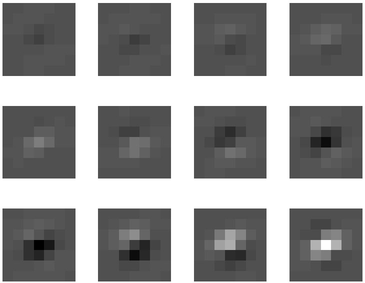




Decoding: mean of $\mathrm{P}\,(s \mid r = 1)$

Spike-triggered average





Decoding: mean of P $(s \mid r = 1)$

Encoding: predictive filter

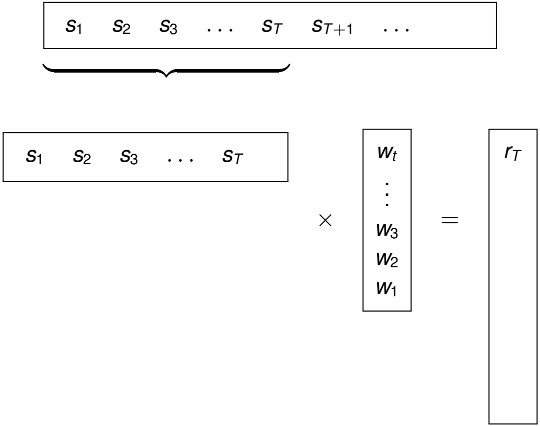
Linear regression

$$s_1 \quad s_2 \quad s_3 \quad \ldots \quad s_T \quad s_{T+1} \quad \ldots$$

$$r(t) = \int_0^T s(t-\tau)w(\tau)d\tau$$

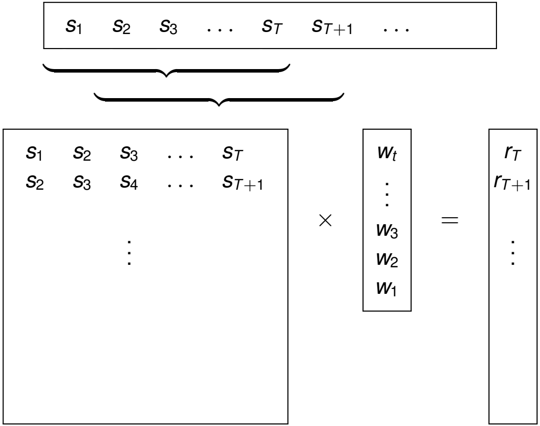
Linear regression

$$r(t) = \int_0^T s(t-\tau)w(\tau)d\tau$$



Linear regression

$$r(t) = \int_0^T s(t-\tau)w(\tau)d\tau$$



Linear regression

$$r(t) = \int_0^T s(t-\tau)w(\tau)d\tau$$

$$\begin{array}{|cccccccc|}\hline s_1 & s_2 & s_3 & \ldots & s_T & s_{T+1} & \ldots & \\ \hline \end{array}$$

$$\begin{pmatrix} s_1 & s_2 & s_3 & \ldots & s_T \\ s_2 & s_3 & s_4 & \ldots & s_{T+1} \\ & & \vdots & & \end{pmatrix} \times \begin{pmatrix} w_t \\ \vdots \\ w_3 \\ w_2 \\ w_1 \end{pmatrix} = \begin{pmatrix} r_T \\ r_{T+1} \\ \vdots \end{pmatrix}$$

$$SW = R$$

Linear regression

$$r(t) = \int_0^T s(t-\tau) w(\tau) d\tau$$

$$\begin{array}{|ccccccc|} \hline s_1 & s_2 & s_3 & \dots & s_T & s_{T+1} & \dots \\ \hline \end{array}$$

$$\begin{pmatrix} s_1 & s_2 & s_3 & \dots & s_T \\ s_2 & s_3 & s_4 & \dots & s_{T+1} \\ & & \vdots & & \end{pmatrix} \times \begin{pmatrix} w_t \\ \vdots \\ w_3 \\ w_2 \\ w_1 \end{pmatrix} = \begin{pmatrix} r_T \\ r_{T+1} \\ \vdots \end{pmatrix}$$

$$SW = R$$

$$W(\omega) = \frac{S(\omega)^* R(\omega)}{|S(\omega)|^2}$$

$$W = \underbrace{(S^T S)}_{\Sigma_{SS}}{}^{-1} \underbrace{(S^T R)}_{\text{STA}}$$

Linear models

So the (whitened) spike-triggered average gives the minimum-squared-error linear model.

Issues:

Linear models

So the (whitened) spike-triggered average gives the minimum-squared-error linear model.

Issues:

- overfitting and regularisation

Linear models

So the (whitened) spike-triggered average gives the minimum-squared-error linear model.

Issues:

- overfitting and regularisation
    - standard methods for regression

Linear models

So the (whitened) spike-triggered average gives the minimum-squared-error linear model.

Issues:

- overfitting and regularisation
  - standard methods for regression
- negative predicted rates

Linear models

So the (whitened) spike-triggered average gives the minimum-squared-error linear model.

Issues:

- overfitting and regularisation
  - standard methods for regression
- negative predicted rates
  - can model deviations from background

Linear models

So the (whitened) spike-triggered average gives the minimum-squared-error linear model.

Issues:

- overfitting and regularisation
  - standard methods for regression
- negative predicted rates
  - can model deviations from background
- real neurons aren't linear

Linear models

So the (whitened) spike-triggered average gives the minimum-squared-error linear model.

Issues:

- overfitting and regularisation
  - standard methods for regression
- negative predicted rates
  - can model deviations from background
- real neurons aren't linear
  - models are still used extensively

Linear models

So the (whitened) spike-triggered average gives the minimum-squared-error linear model.

Issues:

- overfitting and regularisation
  - standard methods for regression
- negative predicted rates
  - can model deviations from background
- real neurons aren't linear
  - models are still used extensively
  - interpretable suggestions of underlying sensitivity (but see later)

Linear models

So the (whitened) spike-triggered average gives the minimum-squared-error linear model.

Issues:

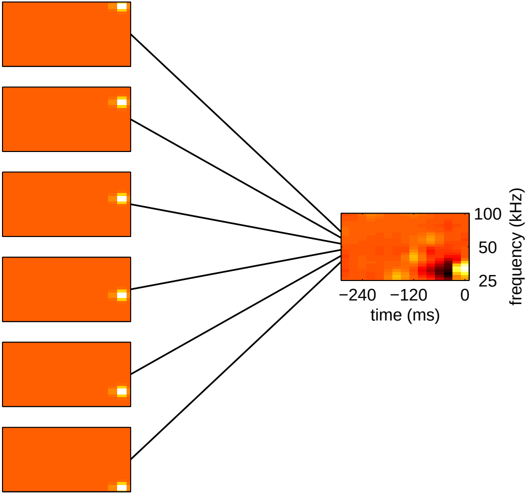
- overfitting and regularisation
  - standard methods for regression
- negative predicted rates
  - can model deviations from background
- real neurons aren't linear
  - models are still used extensively
  - interpretable suggestions of underlying sensitivity (but see later)
  - may provide unbiased estimates of cascade filters (see later)

Likelihood penalties for regularisation

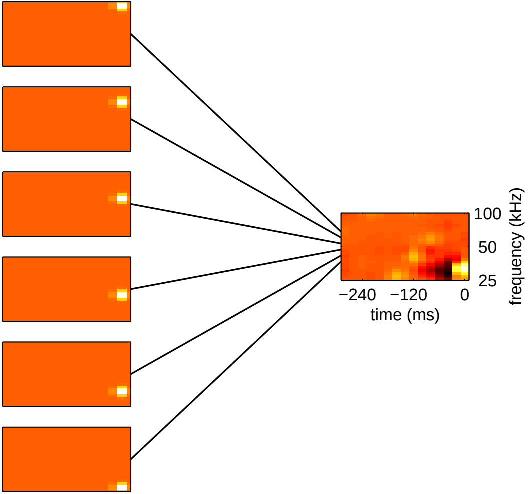
$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\text{argmax}}\, \underbrace{\mathcal{L}(\mathbf{w}; \mathit{Data})}_{\text{Likelihood}} \quad - \underbrace{\mathcal{R}(\mathbf{w})}_{\text{Regulariser}}$$

$\mathcal{R}$ may penalise large values of $\mathbf{w}$ (e.g. $\|\mathbf{w}\|^2$ or $\sum_i |w_i|$) or may promote smoothness or other properties.

Appropriate priors

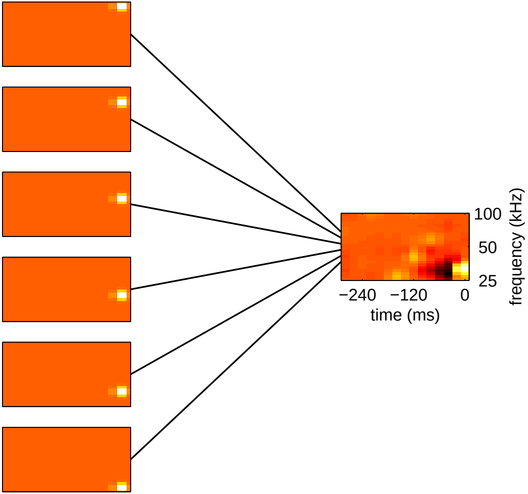


Appropriate priors



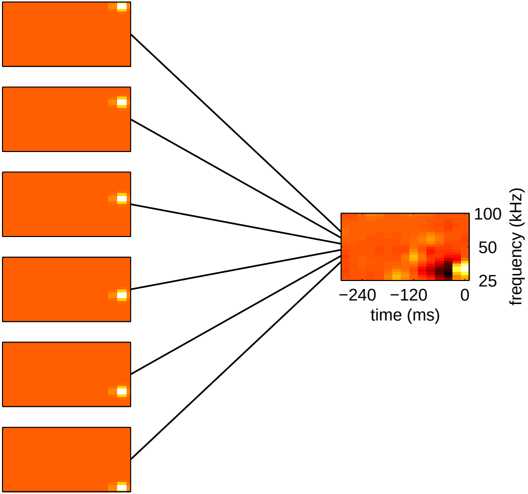
- sparsity [C_{ii} zero for many i] ARD

Appropriate priors



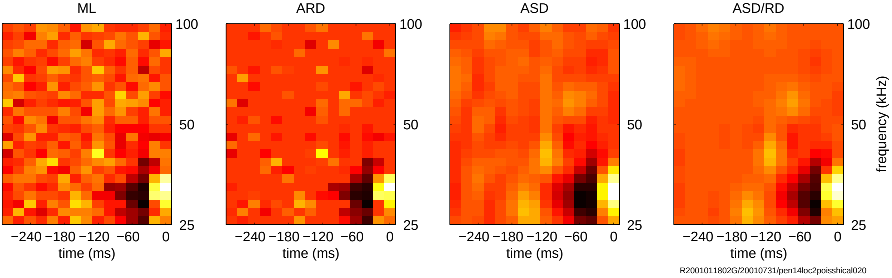
- sparsity [C_{ii} zero for many i] ARD
- smoothness [C_{ij} high for close i and j] ASD

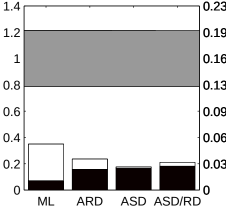
Appropriate priors



- sparsity [C_{ii} zero for many i] ARD
- smoothness [C_{ij} high for close i and j] ASD
- locality [C_{ii} high in a single region] ALD

Smoothness and sparsity (ASD/RD)





Beyond linearity

Beyond linearity

Linear models often fail to predict well. Alternatives?

- Wiener/Volterra functional expansions
  - M-series
  - Linearised estimation
  - Kernel formulations
- LN (Wiener) cascades
  - Spike-trigger covariance (STC) methods
  - "Maximimally informative" dimensions (MID) $\Leftrightarrow$ ML nonparametric LNP models
  - ML Parametric GLM models
- NL (Hammerstein) cascades
  - Multilinear formulations
- LNLN and more ...

The Volterra functional expansion

A polynomial-like expansion for functionals (or operators).

Let $y(t) = F[x(t)]$. Then:

$$y(t) \approx k^{(0)} + \int d\tau\, k^{(1)}(\tau)x(t-\tau) + \iint d\tau_1\, d\tau_2\, k^{(2)}(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)$$
$$+ \iiint d\tau_1\, d\tau_2\, d\tau_3\, k^{(3)}(\tau_1,\tau_2,\tau_3)x(t-\tau_1)x(t-\tau_2)x(t-\tau_3) + \dots$$

or (in discretised time)

$$y_t = K^{(0)} + \sum_i K_i^{(1)} x_{t-i} + \sum_{ij} K_{ij}^{(2)} x_{t-i}x_{t-j} + \sum_{ijk} K_{ijk}^{(3)} x_{t-i}x_{t-j}x_{t-k} + \dots$$

For finite expansion, the kernels $k^{(0)}, k^{(1)}(\cdot), k^{(2)}(\cdot,\cdot), k^{(3)}(\cdot,\cdot,\cdot), \dots$ are not straightforwardly related to the functional F. Indeed, values of lower-order kernels change as the maximum order of the expansion is increased.

Estimation: model is linear in kernels, so can be estimated just like a linear (first-order) model with expanded "input".

- Kernel trick: polynomial kernel $K(x_1, x_2) = (1 + x_1 x_2)^n$.
- M-series.

Wiener Expansion

The Wiener expansion gives functionals of different orders that are orthogonal *for white noise input* $x(t)$.

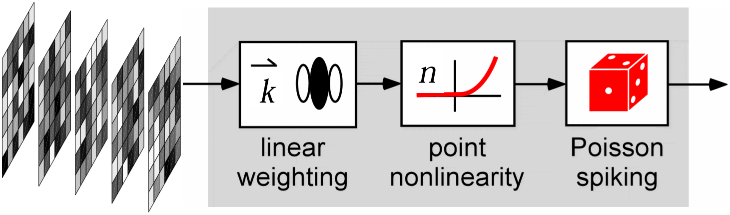
$$
\begin{aligned}
G_0[x(t); h^{(0)}] &= h^{(0)} \\
G_1[x(t); h^{(1)}] &= \int d\tau \, h^{(1)}(\tau) x(t-\tau) \\
G_2[x(t); h^{(2)}] &= \iint d\tau_1 \, d\tau_2 \, h^{(2)}(\tau_1, \tau_2) x(t-\tau_1) x(t-\tau_2) - P \int d\tau_1 \, h^{(2)}(\tau_1, \tau_1) \\
G_3[x(t); h^{(3)}] &= \iiint d\tau_1 \, d\tau_2 \, d\tau_3 \, h^{(3)}(\tau_1, \tau_2, \tau_3) x(t-\tau_1) x(t-\tau_2) x(t-\tau_3) \\
&\quad - 3P \iint d\tau_1 \, d\tau_2 \, h^{(3)}(\tau_1, \tau_2, \tau_2) x(t-\tau_1)
\end{aligned}
$$

Easy to verify that $\mathbb{E}[G_i[x(t)] G_j[x(t)]] = 0$ for $i \neq j$.

Thus, these kernels can be estimated independently. But, they depend on the stimulus.

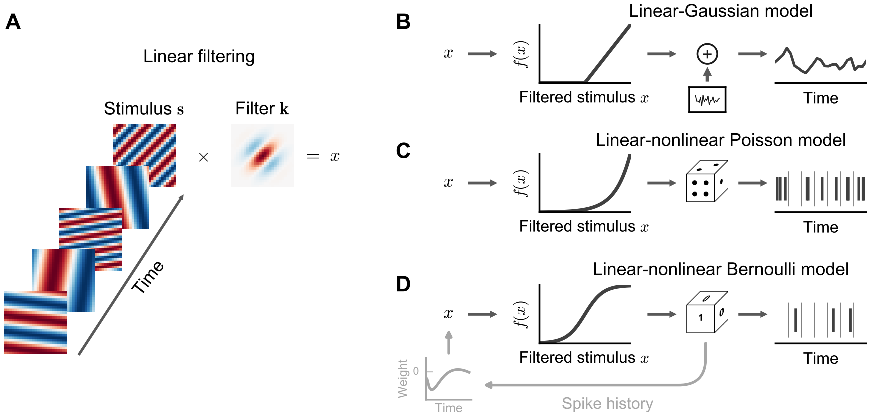
Cascade models

The LNP (Wiener) cascade

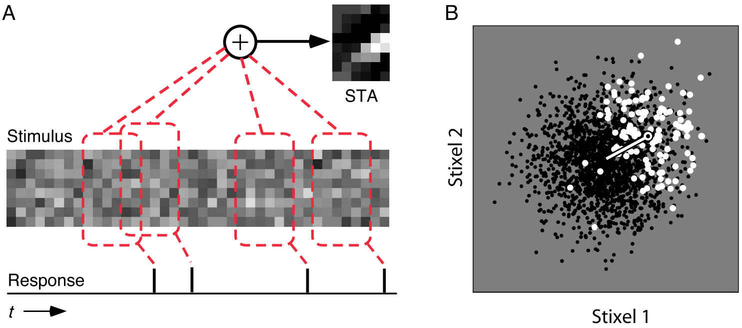


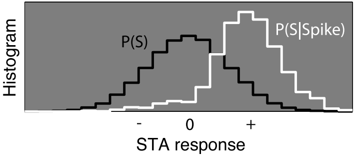
- Rectification addresses negative firing rates.
- Loose biophysical correspondance.

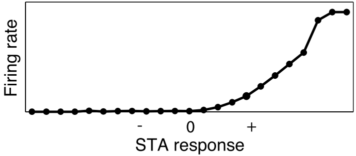
LNP cascades and noise



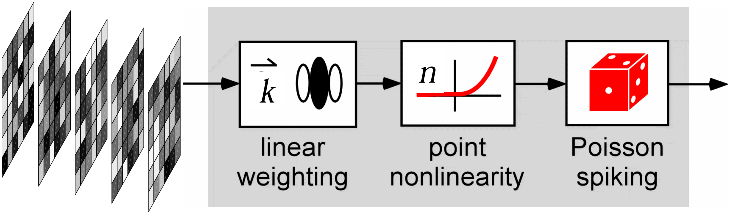
LNP estimation – the Spike-triggered ensemble





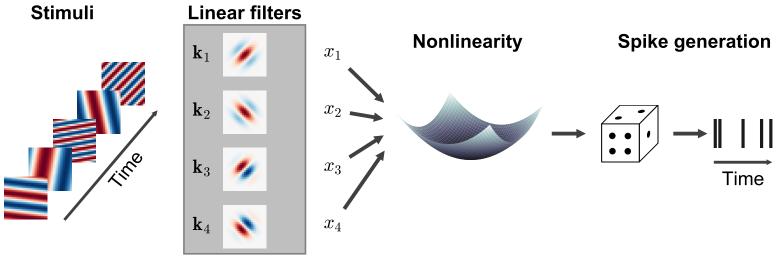


Single linear filter



- STA is unbiased estimate of filter for spherical input distribution. (Bussgang's theorem)
- Elliptically-distributed data can be whitened $\Rightarrow$ linear regression weights are unbiased.
- Linear weights are not necessarily maximum-likelihood (or otherwise optimal), even for spherical/elliptical stimulus distributions.
- Linear weights may be biased for general stimuli (binary/uniform or natural).

Multiple filters

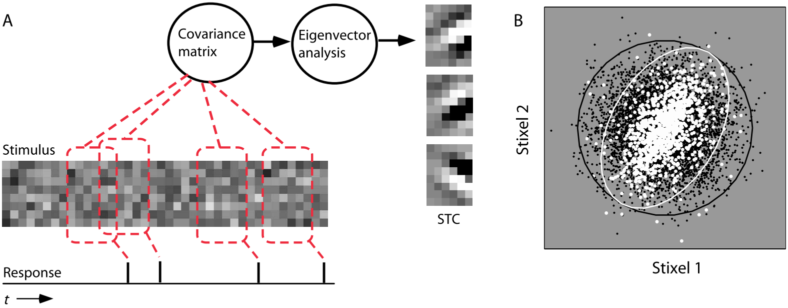


Distribution changes along relevant directions (and, usually, along all linear combinations of relevant directions).

Proxies to measure change in distribution:

- mean: STA (can only reveal a single direction)
- variance: STC
- binned (or kernel) KL divergence: MID "maximally informative directions" (equivalent to ML in LNP model with binned nonlinearity)

STC



Project out STA:

$$\widetilde{S} = S - (S\mathbf{k}_{sta})\mathbf{k}_{sta}^{T}; \quad C_{prior} = \frac{\widetilde{S}^{T}\widetilde{S}}{N}; \, C_{spike} = \frac{\widetilde{S}^{T}\text{diag}(R)\widetilde{S}}{N_{spike}}$$

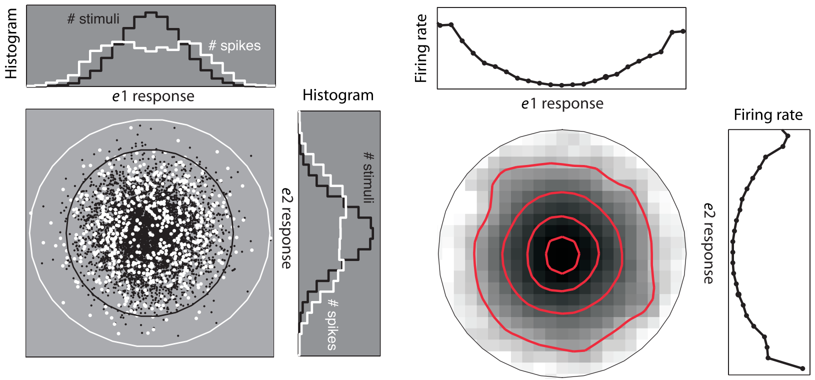
Choose directions with greatest change in variance:

$$\text{k-}\underset{\|\mathbf{v}\|=1}{\text{argmax}}\, \mathbf{v}^{T}(C_{prior} - C_{spike})\mathbf{v}$$

$\Rightarrow$ find eigenvectors of $(C_{prior} - C_{spike})$ with large (absolute) eigvals.

STC

Reconstruct nonlinearity (may assume separability)



Biases

STC (obviously) requires that the nonlinearity alter variance.
If so, subspace is unbiased provided distribution is

- radially (elliptically) symmetric
- AND independent

$\Rightarrow$ Gaussian.

May be possible to correct for non-Gaussian stimulus by transformation, subsampling or weighting (latter two at cost of variance).

More LNP methods

- Non-parametric non-linearities:

  "Maximally informative dimensions" (MID) $\Leftrightarrow$ "non-parametric" maximum likelihood.

  - Intuitively, extends the variance difference idea to arbitrary differences between marginal and spike-conditioned stimulus distributions.

$$\mathbf{k}_{\text{MID}} = \operatorname*{argmax}_{\mathbf{k}} \mathbf{KL}\big[P(\mathbf{k}\cdot\mathbf{x})\big\|P(\mathbf{k}\cdot\mathbf{x}|\text{spike})\big]$$

  - Measuring KL requires binning or smoothing—turns out to be equivalent to fitting a non-parametric nonlinearity by binning or smoothing (Williamson, Sahani, Pillow PLoSCB 2015).
  - Difficult to use for high-dimensional LNP models (but ML viewpoint suggests separable or "cylindrical" basis functions – see Williamson et al.).

- Parametric non-linearities: the "generalised linear model" (GLM).

Generalised linear models

LN models with specified nonlinearities and exponential-family noise.

In general (for monotonic g):

$$y \sim \text{ExpFamily}[\mu(\mathbf{x})]; \qquad g(\mu) = \beta\mathbf{x}$$

For our purposes easier to write

$$y \sim \text{ExpFamily}[f(\beta\mathbf{x})]$$

(Continuous time) point process likelihood with GLM-like dependence of λ on covariates is approached in limit of bins $\to 0$ by either Poisson or Bernoulli GLM.

Mark Berman and T. Rolf Turner (1992) Approximating Point Process Likelihoods with GLIM
Journal of the Royal Statistical Society. Series C (Applied Statistics), 41(1):31-38.

Generalised linear models

Poisson distribution $\Rightarrow f = \exp()$ is *canonical* (*natural params* $= \beta\mathbf{x}$).

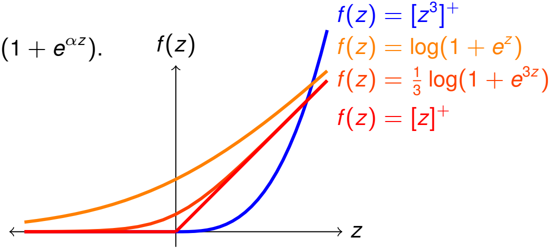
Canonical link functions give concave likelihoods $\Rightarrow$ unique maxima.

Generalises (for Poisson) to any f which is convex and log-concave:

$$\text{log-likelihood} = c - f(\beta\mathbf{x}) + y \log f(\beta\mathbf{x})$$

Includes:

- threshold-linear
- threshold-polynomial
- "soft-threshold" $f(z) = \alpha^{-1} \log(1 + e^{\alpha z})$.



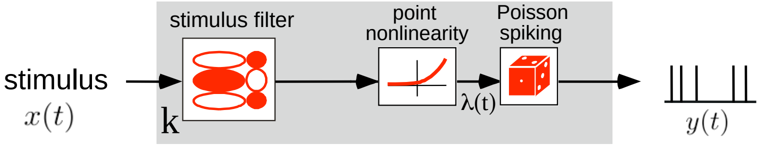
Generalised linear models

ML parameters found by

- gradient ascent
- IRLS

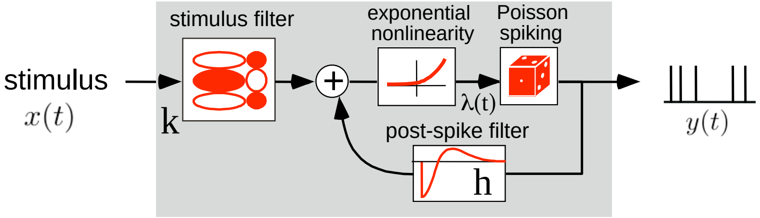
Regularisation by L_2 (quadratic) or L_1 (absolute value – sparse) penalties (MAP with Gaussian/Laplacian priors) preserves concavity.

Linear-Nonlinear-Poisson (GLM)



GLM with history-dependence

(Truccolo et al 04)

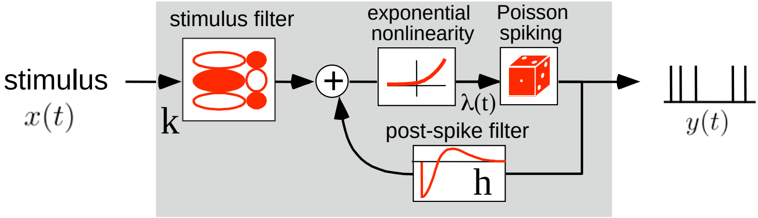


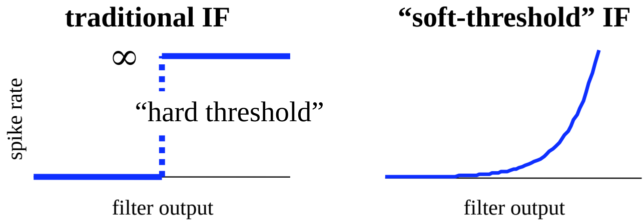
conditional intensity
(spike rate)

$$\lambda(t) = f(k \cdot x(t) \; + \; h \cdot y(t))$$
$$= e^{k \cdot x(t)} \; \cdot \; e^{h \cdot y(t)}$$

- rate is a product of stim- and spike-history dependent terms
- output no longer a Poisson process
- also known as “soft-threshold” Integrate-and-Fire model

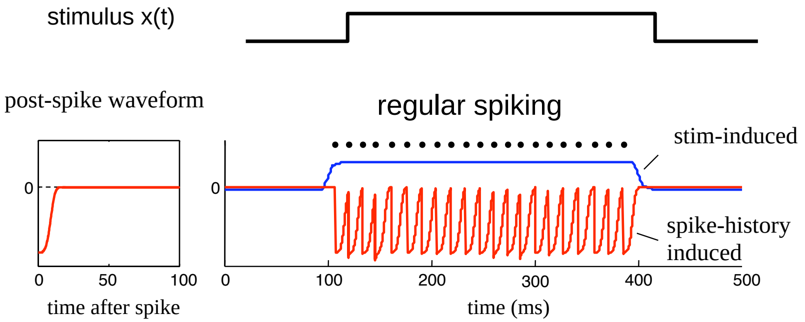
GLM with history-dependence



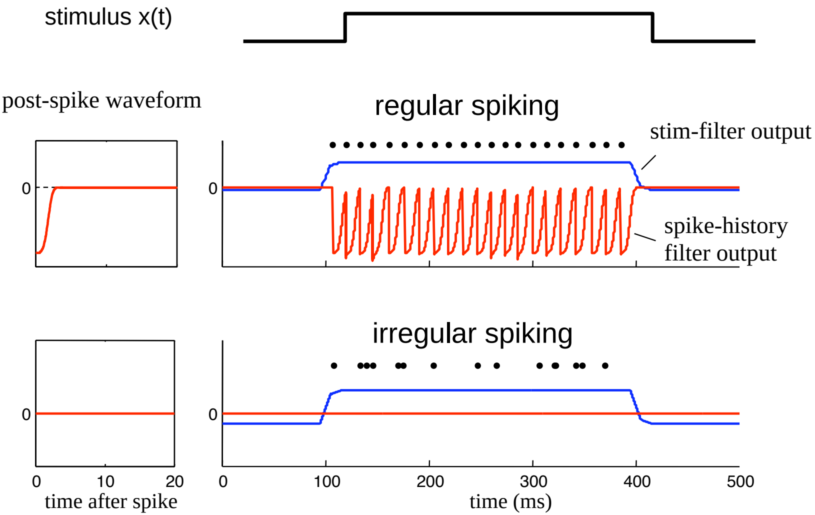


- "soft-threshold" approximation to Integrate-and-Fire model

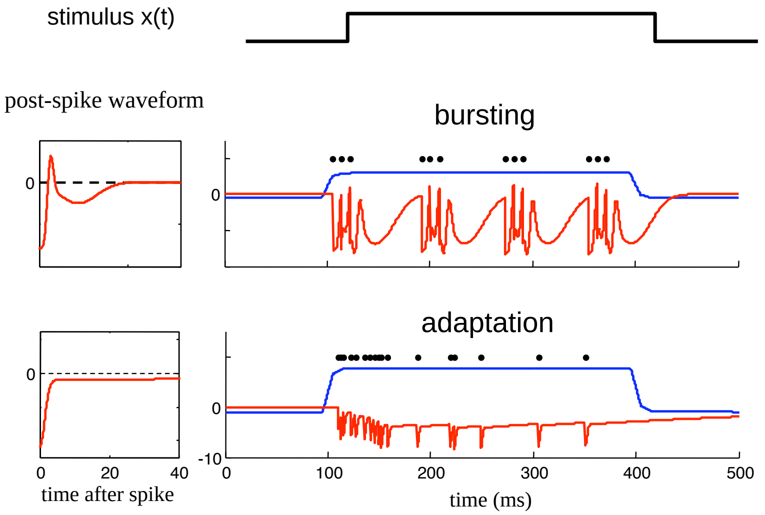
GLM dynamic behaviors



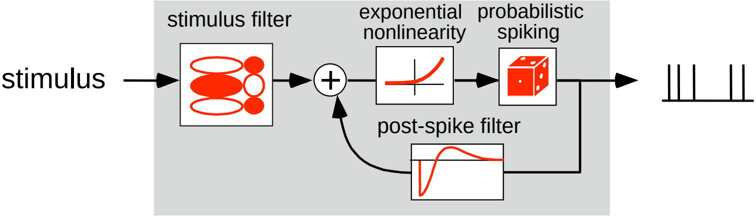
GLM dynamic behaviors



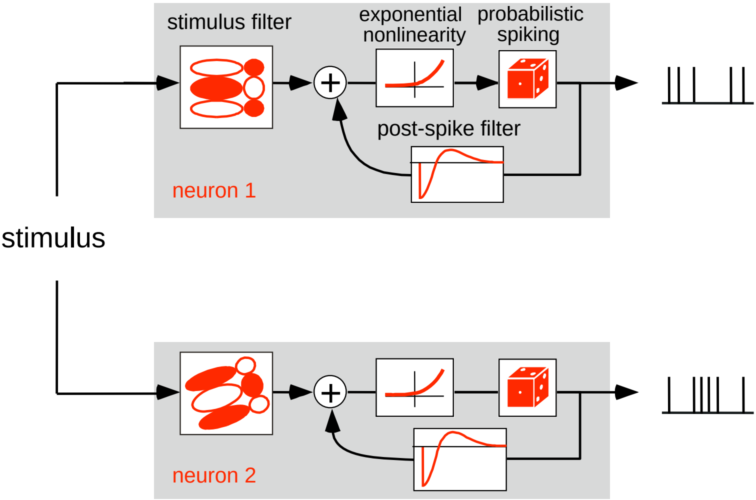
GLM dynamic behaviors



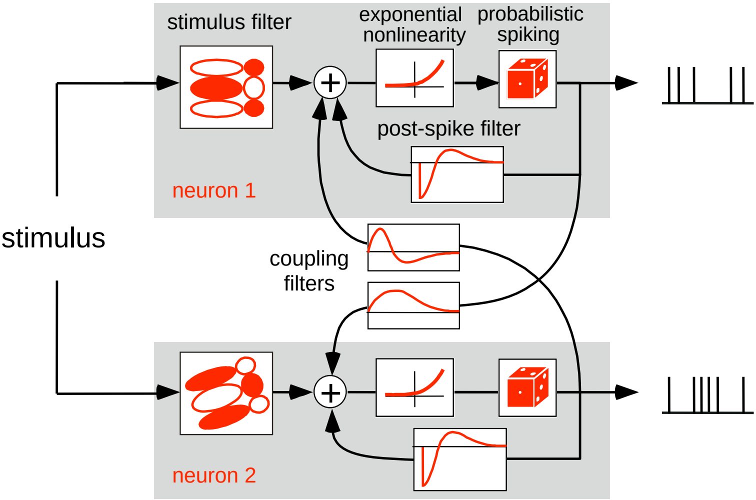
Generalized Linear Model (GLM)



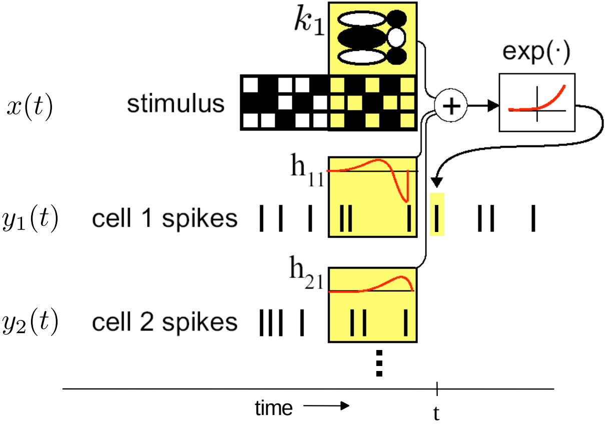
multi-neuron GLM



multi-neuron GLM



GLM equivalent diagram:



conditional intensity
(spike rate)

$$\lambda_i(t) = \exp(k_i \cdot x(t) \; + \; \sum_j h_{ij} \cdot y(t))$$

Non-LN models?

The idea of responses depending on one or a few linear stimulus projections has been dominant, but cannot capture all non-linearities.

- Contrast sensitivity might require normalisation by $\|\mathbf{s}\|$.
- Linear weighting may depend on *units* of stimulus measurement: amplitude? energy? logarithms? thresholds? (NL models – Hammerstein cascades)
- Neurons, particularly in the auditory system are known to be sensitive to combinations of inputs: forward suppression; spectral patterns (Young); time-frequency interactions (Sadogopan and Wang).
- Experiments with realistic stimuli reveal nonlinear sensivity to parts/whole (Bar-Yosef and Nelken).

Many of these questions can be tackled using a multilinear (cartesian tensor) framework.

Input nonlinearities

The basic linear model (for sounds):

$$\underbrace{\widehat{r}(i)}_{\text{predicted rate}} = \sum_{jk} \underbrace{w^{\mathbf{tf}}_{jk}}_{\text{STRF weights}} \underbrace{s(i-j,k)}_{\text{stimulus power}} ,$$

Input nonlinearities

The basic linear model (for sounds):

$$\underbrace{\hat{r}(i)}_{\text{predicted rate}} = \sum_{jk} \underbrace{w^{\mathbf{tf}}_{jk}}_{\text{STRF weights}} \underbrace{s(i-j,k)}_{\text{stimulus power}} ,$$

How to measure s? (pressure, intensity, dB, thresholded, . . .)

Input nonlinearities

The basic linear model (for sounds):

$$\underbrace{\widehat{r}(i)}_{\text{predicted rate}} = \sum_{jk} \underbrace{w^{\mathbf{tf}}_{jk}}_{\text{STRF weights}} \underbrace{s(i-j,k)}_{\text{stimulus power}} ,$$

How to measure s? (pressure, intensity, dB, thresholded, ...)

We can *learn* an optimal representation $g(.)$:

$$\hat{r}(i) = \sum_{jk} w^{\mathbf{tf}}_{jk} g(s(i-j,k)).$$

Input nonlinearities

The basic linear model (for sounds):

$$\underbrace{\hat{r}(i)}_{\text{predicted rate}} = \sum_{jk} \underbrace{w^{\mathbf{tf}}_{jk}}_{\text{STRF weights}} \underbrace{s(i-j,k)}_{\text{stimulus power}} ,$$

How to measure s? (pressure, intensity, dB, thresholded, ...)

We can *learn* an optimal representation $g(.)$:

$$\hat{r}(i) = \sum_{jk} w^{\mathbf{tf}}_{jk} g(s(i-j,k)).$$

Define: basis functions $\{g_l\}$ such that $g(s) = \sum_l w^{\mathbf{l}}_l g_l(s)$
and stimulus array $M_{ijkl} = g_l(s(i-j,k))$. Now the model is

$$\hat{r}(i) = \sum_{jkl} w^{\mathbf{tf}}_{jk} w^{\mathbf{l}}_l M_{ijkl}$$

Input nonlinearities

The basic linear model (for sounds):

$$\underbrace{\widehat{r}(i)}_{\text{predicted rate}} = \sum_{jk} \underbrace{w^{\mathbf{tf}}_{jk}}_{\text{STRF weights}} \underbrace{s(i-j,k)}_{\text{stimulus power}} ,$$

How to measure s? (pressure, intensity, dB, thresholded, ...)

We can *learn* an optimal representation $g(.)$:

$$\hat{r}(i) = \sum_{jk} w^{\mathbf{tf}}_{jk} g(s(i-j,k)).$$

Define: basis functions $\{g_l\}$ such that $g(s) = \sum_l w^{\mathbf{l}}_l g_l(s)$
and stimulus array $M_{ijkl} = g_l(s(i-j,k))$. Now the model is

$$\hat{r}(i) = \sum_{jkl} w^{\mathbf{tf}}_{jk} w^{\mathbf{l}}_l M_{ijkl} \quad \text{or} \quad \widehat{\mathbf{r}} = (\mathbf{w}^{\mathbf{tf}} \otimes \mathbf{w}^{\mathbf{l}}) \bullet \mathbf{M}.$$

Multilinear models

Multilinear forms are straightforward to optimise by alternating least squares.

Cost function:

$$\mathcal{E} = \left\| \mathbf{r} - (\mathbf{w}^{\mathsf{tf}} \otimes \mathbf{w}^{\mathsf{l}}) \bullet \mathbf{M} \right\|^2$$

Minimise iteratively, defining *matrices*

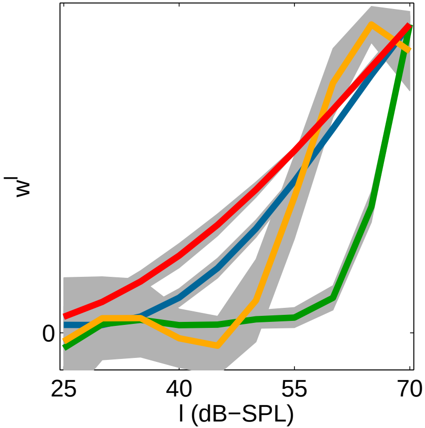
$$\mathbf{B} = \mathbf{w}^{\mathsf{l}} \bullet \mathbf{M} \qquad \text{and} \qquad \mathbf{A} = \mathbf{w}^{\mathsf{tf}} \bullet \mathbf{M}$$

and updating

$$\mathbf{w}^{\mathsf{tf}} = (\mathbf{B}^{\mathsf{T}}\mathbf{B})^{-1}\mathbf{B}^{\mathsf{T}}\mathbf{r} \qquad \text{and} \qquad \mathbf{w}^{\mathsf{l}} = (\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{r}.$$

Each linear regression step can be regularised by evidence optimisation (suboptimal), with uncertainty propagated approximately using *variational* methods.

Some input non-linearities



Variable (combination-dependent) input gain

- Sensitivities to different points in sensory space are not independent.

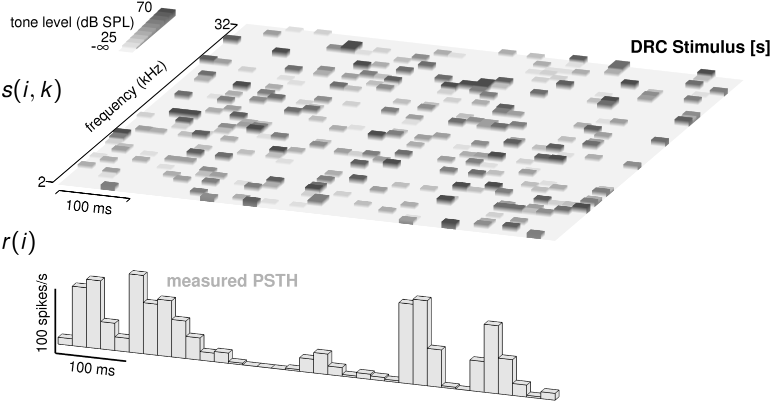
Variable (combination-dependent) input gain

- Sensitivities to different points in sensory space are not independent.
- Rather, the sensitivity at one point depends on other elements of the stimulus that create a *local* sensory context.

Variable (combination-dependent) input gain

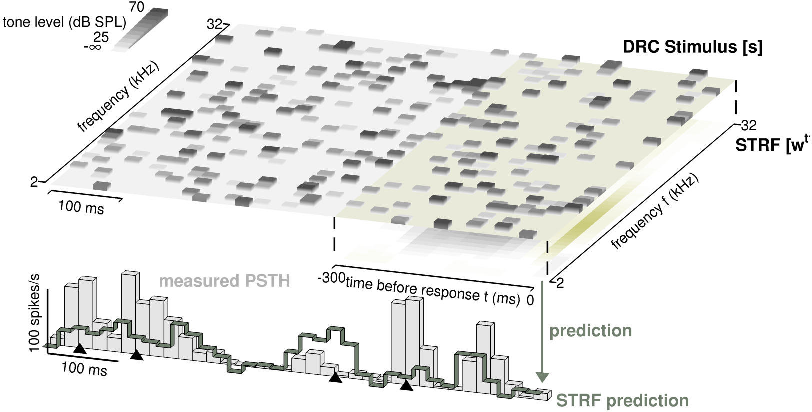
- Sensitivities to different points in sensory space are not independent.
- Rather, the sensitivity at one point depends on other elements of the stimulus that create a *local* sensory context.
- This context adjusts the input gain of the cell from moment to moment, dynamically refining the shape of the weighted receptive field.

Context-sensitive gain



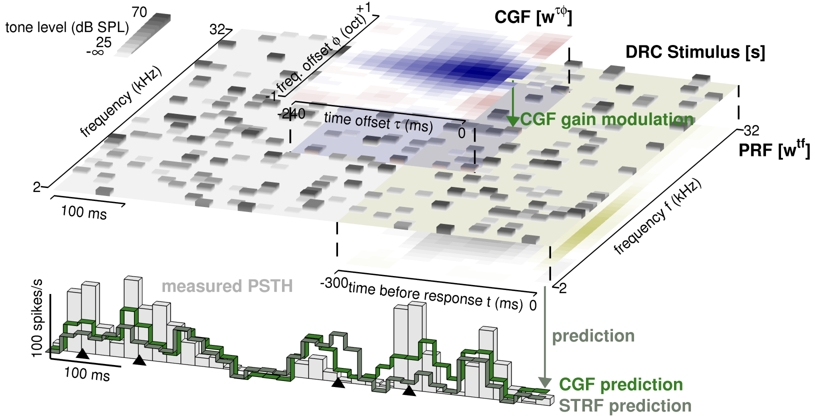
Context-sensitive gain

$$\hat{r}(i) = c + \sum_{j=0}^{J} \sum_{k=1}^{K} w^{tf}_{j+1,k} s(i-j,k)$$



Context-sensitive gain

$$\hat{r}(i) = c + \sum_{j=0}^{J} \sum_{k=1}^{K} w^{tf}_{j+1,k} s(i-j,k) \left(1 + \sum_{m=0}^{M} \sum_{n=-N}^{N} w^{\tau\phi}_{m+1,n+N+1} s(i-j-m, k+n) \right)$$



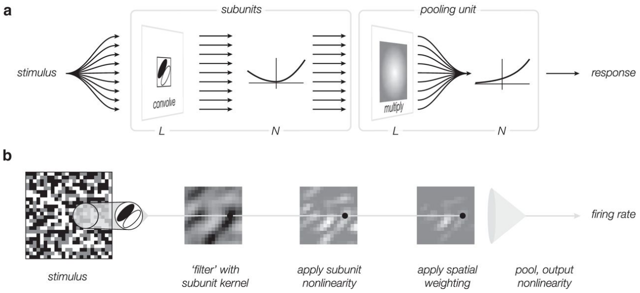
LNLN cascades

- Limited description of 'layered' structure of sensory pathways:

$$\hat{r}(t) = f\left(\sum_{n=1}^{N} w_n g_n\left(\mathbf{k}_n^{\mathsf{T}}\mathbf{s}(t)\right)\right)$$

- $\mathbf{k}_n$ describes the linear filter and g_n the output nonlinearity of each of N input subunits. The g_n are usually fixed half-wave rectifiers.
- Called a generalised nonlinear model (GNM; Butts *et al*. 2007, 2011; Schinkel-Bielefeld *et al*. 2012)
- Or a nonlinear input model (NIM; McFarland *et al*. 2013).
- Parameters estimated by maximum-likelihood using inhomogeneous Poisson noise – often by alternation (following Ahrens et al. 2008).
- Resembles a (perceptron) “neural network”.

Convolutional LNLN



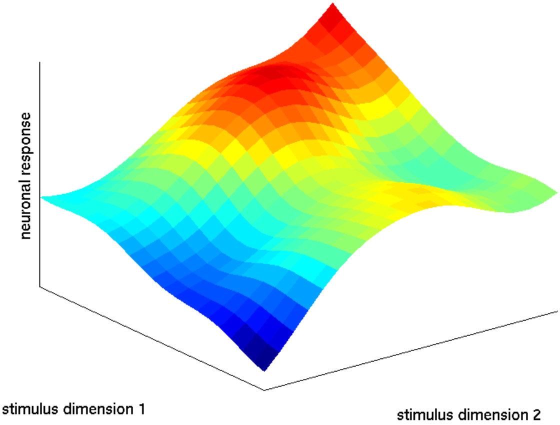
$$\hat{r}(t) = f\left(\sum_{c=1}^{C}\sum_{n=1}^{N} w_{c,n} \sum_{i=1}^{B} b_{c,i} g_i\left(\mathbf{k}_{c,n}^{\mathsf{T}}\mathbf{s}(t)\right)\right)$$

- C "channels" – each uses same kernel $\mathbf{k}_c$ translated to a different location (convolution).
- Input nonlinearities learned using basis expansion and alternation (Ahrens et al. 2008).
- Output nonlinearity f fixed.

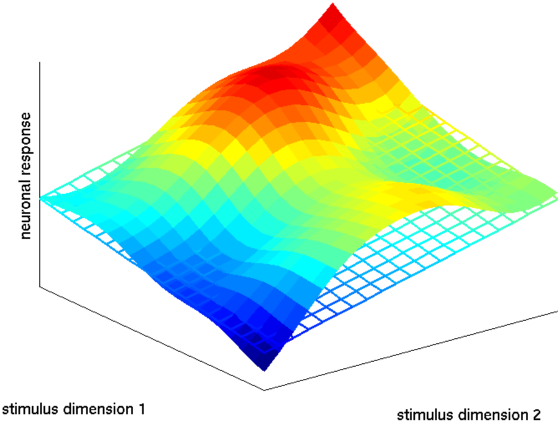
Limitations of linear approximations

What are the consequences of nonlinearities in the stimulus-response function for interpretation of structure in linear models like STRFs?

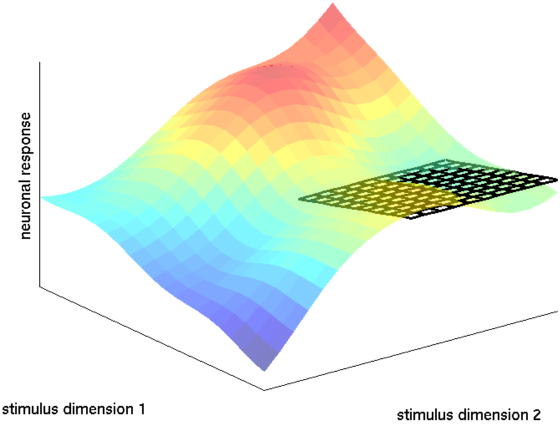
Linear fits to non-linear functions



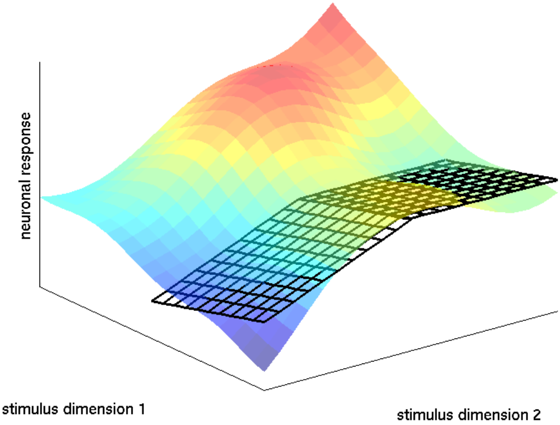
Linear fits to non-linear functions



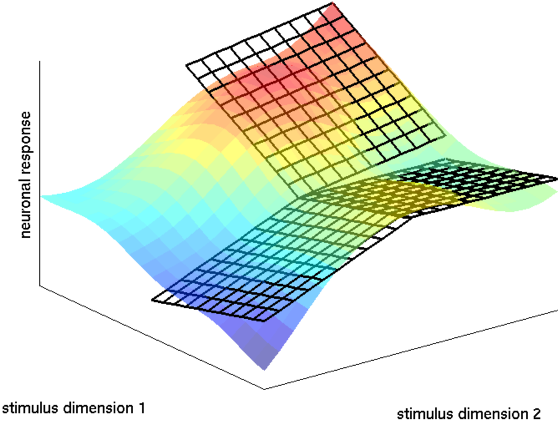
Approximations are stimulus dependent



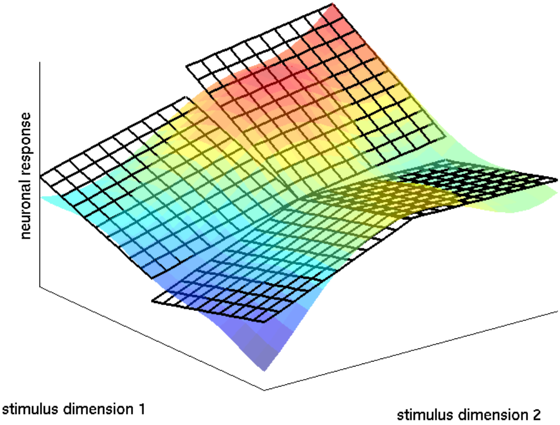
Approximations are stimulus dependent



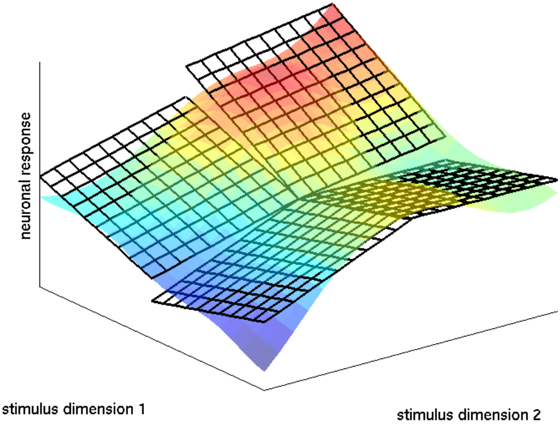
Approximations are stimulus dependent



Approximations are stimulus dependent



Approximations are stimulus dependent



(Stimulus dependence does not always signal response adaptation)

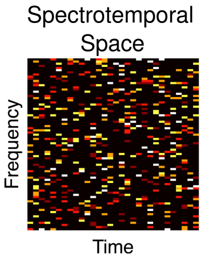
Consequences

Local fitting can have counterintuitive consequences on the interpretation of a “receptive field”.

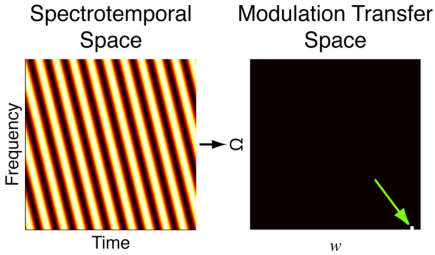
“Independently distributed” stimuli

Knowing stimulus power at any set of points in analysis space provides no information about stimulus power at any other point.

DRC:

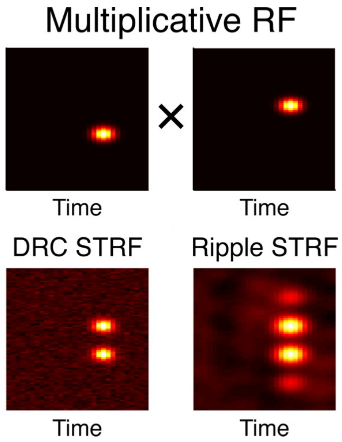


Ripple:



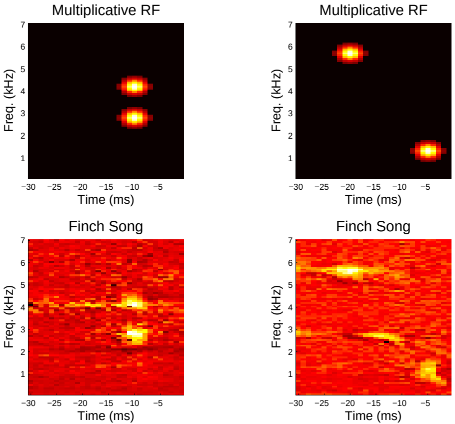
Independence is a property of stimulus *and* analysis space.

Nonlinearity & non-independence distort RF estimates



Stimulus may have higher-order correlations in other analysis spaces — and interaction with nonlinearities can produce misleading “receptive fields.” (Christianson, Sahani and Linden 2008 J Neurosci)

What about natural sounds?



Usually not independent in any space — so STRFs may not be conservative estimates of receptive fields.

Summary

How can we use linear models of neuronal stimulus-response functions most effectively to answer biological questions?

Pay a lot of attention to three key issues:

1. nature of stimulus
   - ethological/physiological relevance?
   - second-order and/or higher-order autocorrelations?
2. choice of stimulus representation
   - appropriate to the biology?
   - appropriate to the question?
3. limitations of linear approximation
   - consequences of likely nonlinearities in stimulus-response function?
   - interaction with higher-order autocorrelation in stimulus?

Linear modelling can be a simple and useful tool for answering specific questions about neural coding of stimuli, but results must be interpreted carefully.

How good are linear models?

Model evaluation

We would like an absolute measure of model performance. Two things make this difficult:

Model evaluation

We would like an absolute measure of model performance. Two things make this difficult:

Measured responses can never be predicted perfectly, even in principle:

- The measurements themselves are noisy.

Model evaluation

We would like an absolute measure of model performance. Two things make this difficult:

Measured responses can never be predicted perfectly, even in principle:

- The measurements themselves are noisy.

Even if we can discount this, a model may predict poorly because either:

- It is the wrong model.
- The parameters are mis-estimated due to noise.

Model evaluation

We would like an absolute measure of model performance. Two things make this difficult:

Measured responses can never be predicted perfectly, even in principle:

- The measurements themselves are noisy.

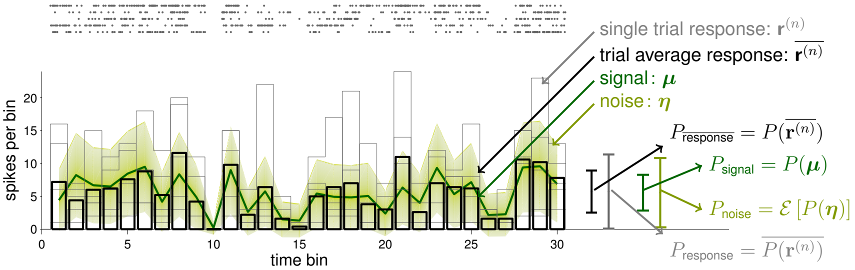
Even if we can discount this, a model may predict poorly because either:

- It is the wrong model.
- The parameters are mis-estimated due to noise.

Approaches:

- Compare $I(\text{resp}; \text{pred})$ to $I(\text{resp}; \text{stim})$.
  - mutual information estimators are biased (and may not be what we really want)
- Compare $E(\text{resp} - \text{pred})$ to $E(\text{resp} - \text{psth})$ where psth is gathered over a very large number of trials.
  - may require impractical amounts of data to estimate the psth
- Compare the *predictive power* to the *predictable power* (similar to ANOVA).

Estimating predictable power



$$\underbrace{\text{response}}_{\mathbf{r}^{(n)}} = \underbrace{\text{signal}}_{\boldsymbol{\mu}} + \underbrace{\text{noise}}_{\boldsymbol{\eta}^{(n)}}$$

$$\left.\begin{aligned} \mathcal{E}\left[P_{\text{response}}\right] &= P_{\text{signal}} + P_{\text{noise}} \\ \mathcal{E}\left[P_{\overline{\text{response}}}\right] &= P_{\text{signal}} + \frac{1}{N} P_{\text{noise}} \end{aligned}\right\} \Rightarrow \left\{\begin{aligned} \widehat{P}_{\text{signal}} &= \frac{1}{N-1}\left(N P(\overline{\mathbf{r}^{(n)}}) - \overline{P(\mathbf{r}^{(n)})}\right) \\ \widehat{P}_{\text{noise}} &= \overline{P(\mathbf{r}^{(n)})} - \widehat{P}_{\text{signal}} \end{aligned}\right.$$

Testing a model

For a perfect prediction

$$\left\langle \mathrm{P}(\overline{\mathrm{trial}}) - \mathrm{P}(\mathrm{residual}) \right\rangle = \mathrm{P}(\mathrm{signal})$$

Testing a model

For a perfect prediction

$$\left\langle \mathrm{P}(\overline{\mathrm{trial}}) - \mathrm{P}(\mathrm{residual}) \right\rangle = \mathrm{P}(\mathrm{signal})$$

Thus, we can judge the performance of a model by the normalized predictive power

$$\frac{\mathrm{P}(\overline{\mathrm{trial}}) - \mathrm{P}(\mathrm{residual})}{\widehat{\mathrm{P}}(\mathrm{signal})}$$

Testing a model

For a perfect prediction

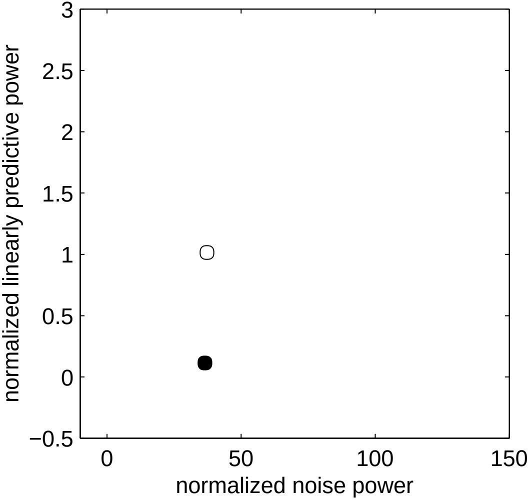
$$\left\langle \mathrm{P}(\overline{\mathrm{trial}}) - \mathrm{P}(\mathrm{residual}) \right\rangle = \mathrm{P}(\mathrm{signal})$$

Thus, we can judge the performance of a model by the normalized predictive power

$$\frac{\mathrm{P}(\overline{\mathrm{trial}}) - \mathrm{P}(\mathrm{residual})}{\widehat{\mathrm{P}}(\mathrm{signal})}$$

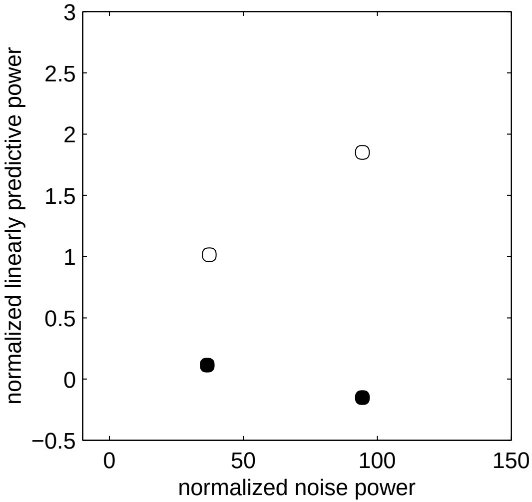
Similar to coefficient of determination (r^2), but the denominator is the predictable variance.

Extrapolating the model performance



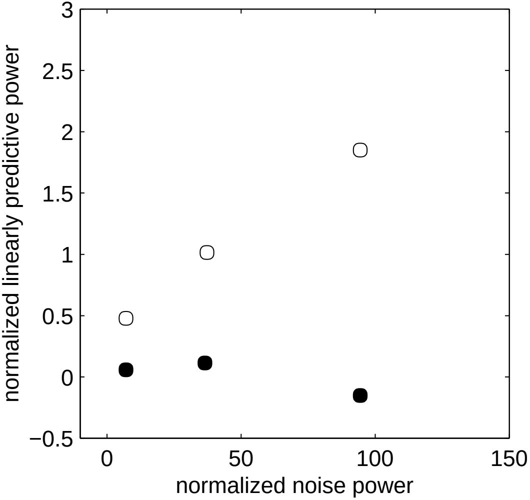
(Sahani and Linden 2003 NIPS)

Extrapolating the model performance



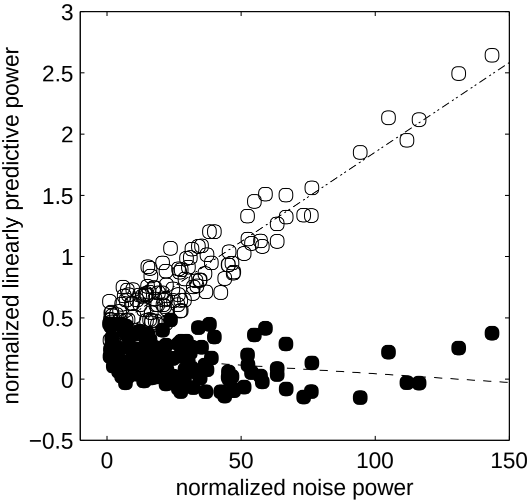
(Sahani and Linden 2003 NIPS)

Extrapolating the model performance



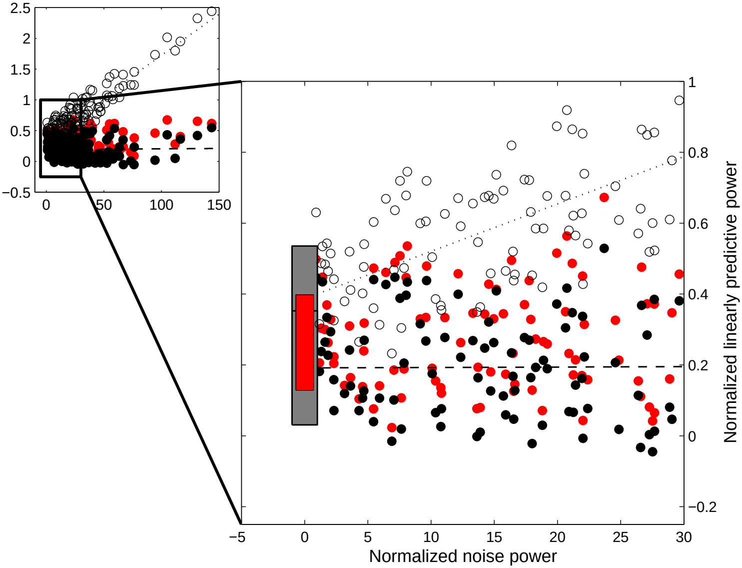
(Sahani and Linden 2003 NIPS)

Extrapolating the model performance



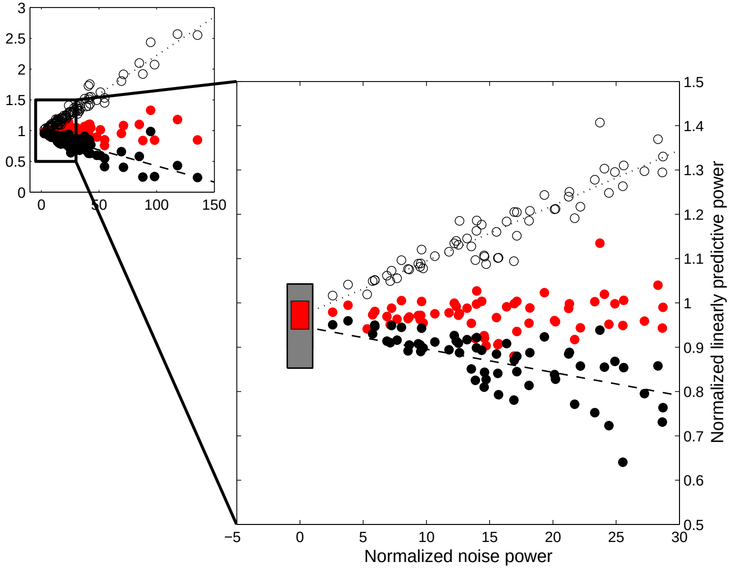
(Sahani and Linden 2003 NIPS)

Extrapolated linearity



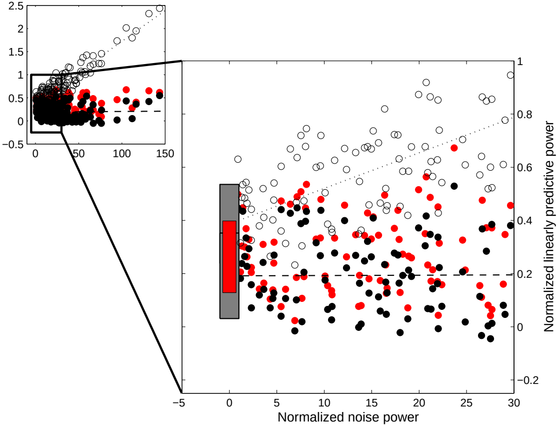
[extrapolated range: (0.19,0.39); mean Jackknife estimate: 0.29]

Simulated (almost) linear data



[extrapolated range: (0.95,0.97); mean Jackknife estimate: 0.97]

Linearity and nonlinearity in auditory cortical responses



So, spectrogram-linear models capture approximately 20–40% of the variability in auditory cortical responses to random chord stimuli (Sahani and Linden 2003 NIPS).

For natural sounds, performance is no better (Machens et al. 2004 J Neurosci).

Linearity in thalamus versus cortex

Spectrogram-linear models perform better in the thalamus than in the cortex (more on this later).

Not just because cortex is noisier but because cortical representations are more nonlinear!

Other studies likewise indicate that linearity of stimulus representation generally decreases as we ascend the auditory pathway (e.g., Chechik and Nelken 2012 PNAS; Atencio et al. 2012 J Neurosci; Williamson et al. 2016 Neuron).