Neural Encoding Models

Maneesh Sahani

Gatsby Computational Neuroscience Unit
University College London

March 2022

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.

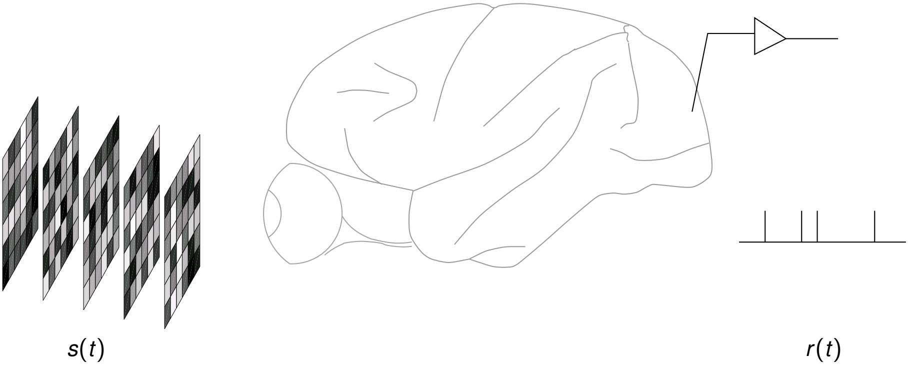
The coding questions:

- 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.

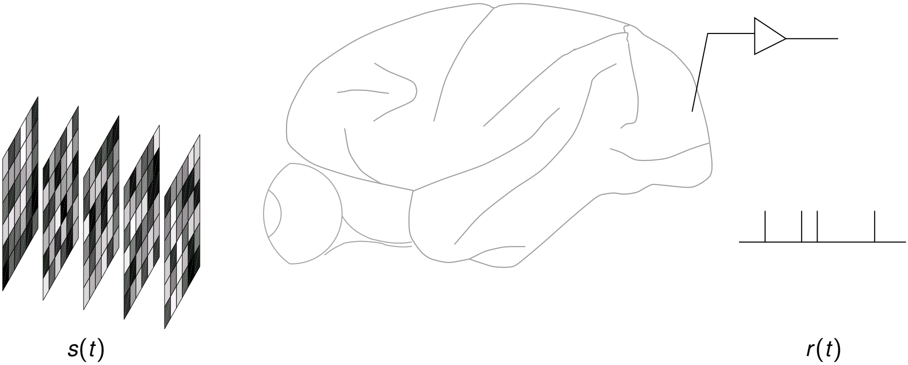
Computation plays a vital part in systematising empirical data.

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:

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

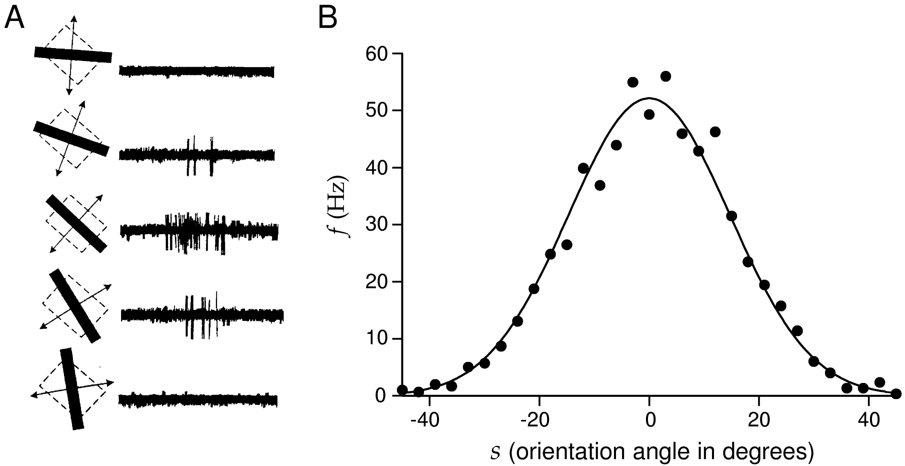
To fully model the response we need to identify λ. In general this depends on spike history $H(t)$ and network activity. Three options:

- 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)

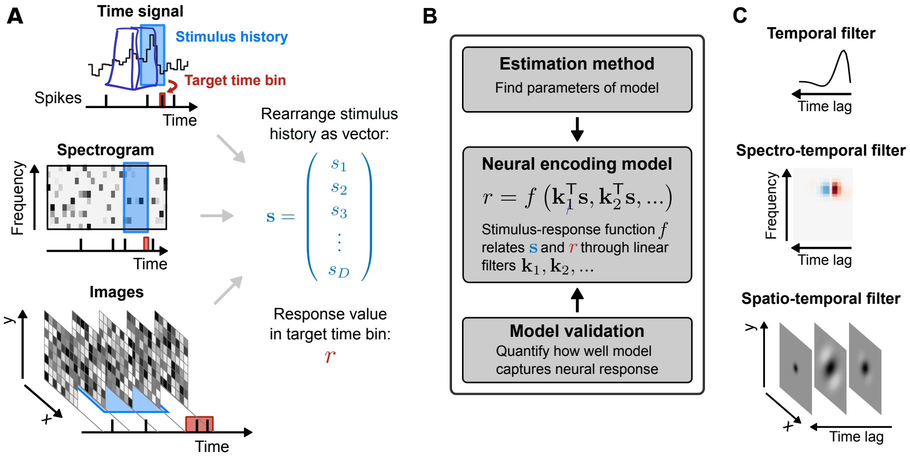
  $$\overline{\lambda}(t, \text{stimulus}) = \lim_{N\to\infty} \frac{1}{N} \sum_n \lambda(t|H_n(t), \text{stimulus}, \text{network}_n)\,,$$

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

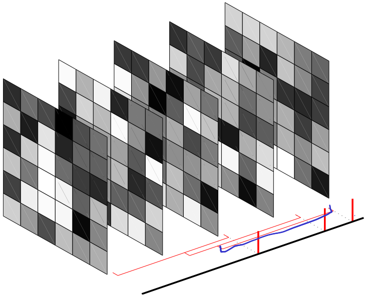
Tuning – stationary stimuli

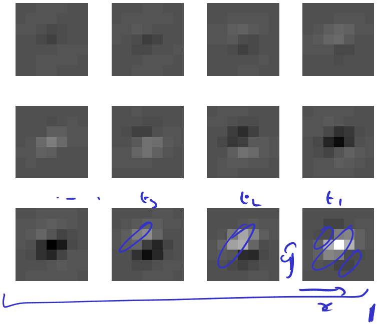


(Nonlinear) filtering – dynamic stimuli



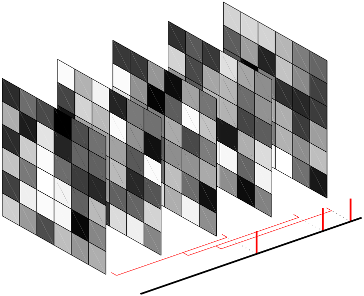
Spike-triggered average

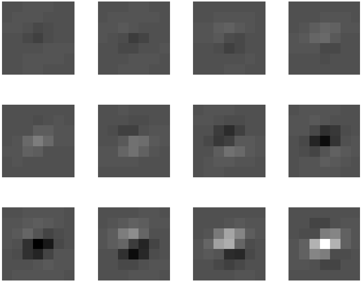




Decoding: mean of $P\left(s \mid r = 1\right)$

Spike-triggered average





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

Encoding: predictive filter

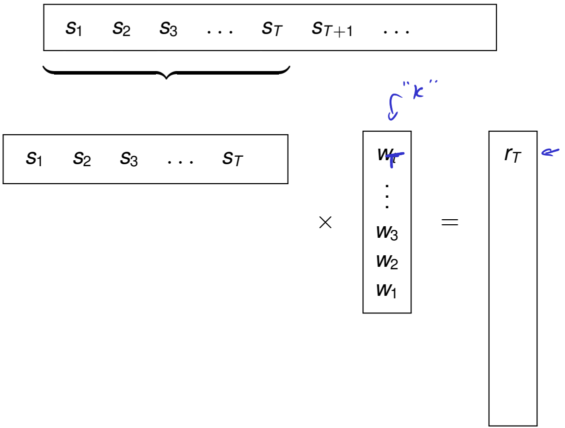
Linear regression

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

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

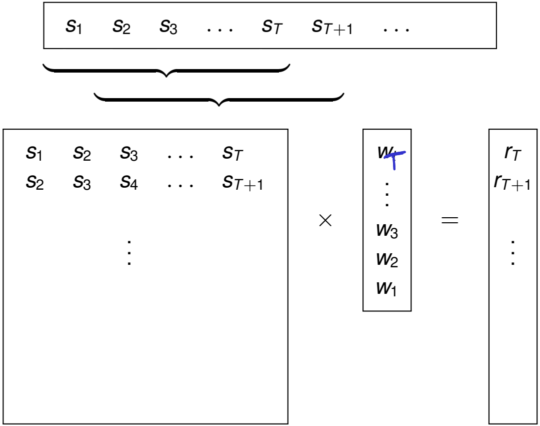
Linear regression

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



Linear regression

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



Linear regression

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

$$\begin{bmatrix} s_1 & s_2 & s_3 & \dots & s_T & s_{T+1} & \dots \end{bmatrix}$$

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

$$SW = R$$

Linear regression

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

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

minimise:

$$\int (r(t) - \hat{r}(t))^2 \, dt$$

$$\underset{\sim}{s}_t^{\top} \quad \begin{pmatrix} s_1 & s_2 & s_3 & \dots & s_T \\ s_2 & s_3 & s_4 & \dots & s_{T+1} \\ & & \vdots & & \end{pmatrix} \times \begin{pmatrix} w_t \\ \vdots \\ w_3 \\ w_2 \\ w_1 \end{pmatrix} = \begin{pmatrix} r_T \\ r_{T+1} \\ \vdots \end{pmatrix}$$

$$SW = R$$

$$\sum_t \underset{\sim}{s}_t \underset{\sim}{s}_t^{\top}$$

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

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

$\Sigma_{SS} = I$?

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

$$\hat{r}(t) = \int s(t-\tau)\, w(\tau) + b$$

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:

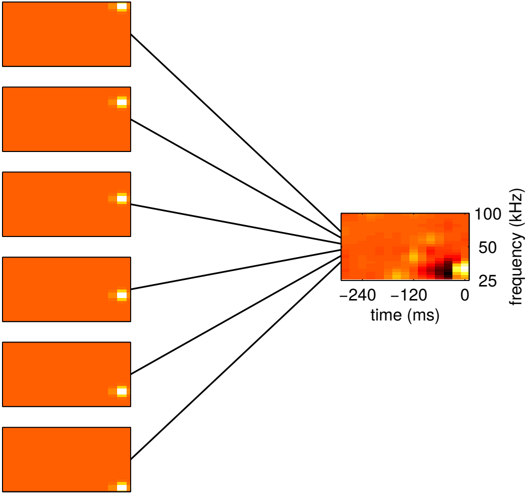
- overfitting and regularisation
  - standard methods for regression
- negative predicted rates
  - can model deviations from background
- real neurons aren't linear
  - models are still used extensively
  - interpretable suggestions of underlying sensitivity (but see later)
  - may provide unbiased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

Likelihood penalties for regularisation

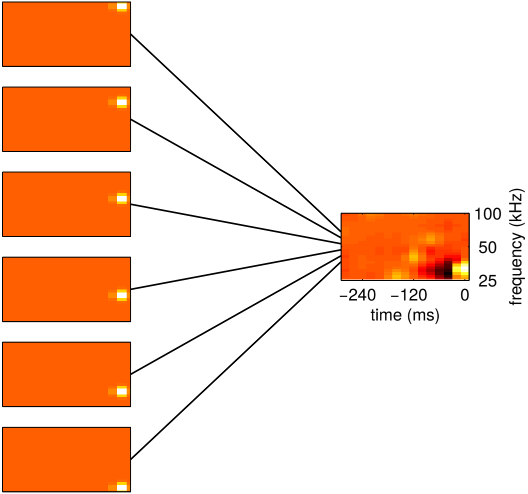
$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \underbrace{\mathcal{L}(\mathbf{w}; \mathit{Data})}_{\text{Likelihood}} \quad - \underbrace{\mathcal{R}(\mathbf{w})}_{\text{Regulariser}}$$

$\mathcal{R}$ 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

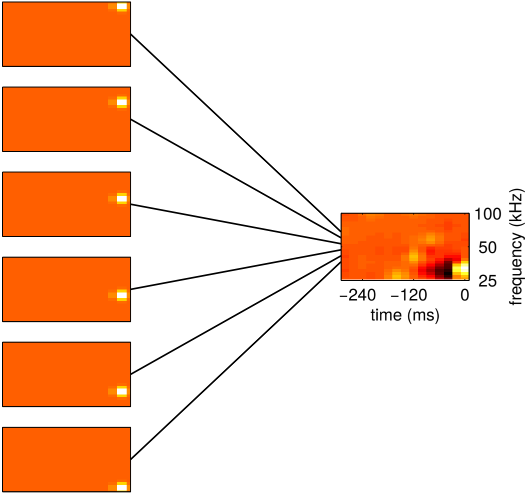


Appropriate priors



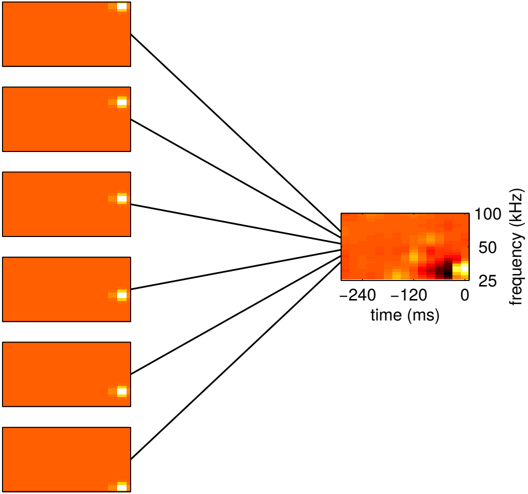
- sparsity [C_{ii} zero for many i] ARD

Appropriate priors



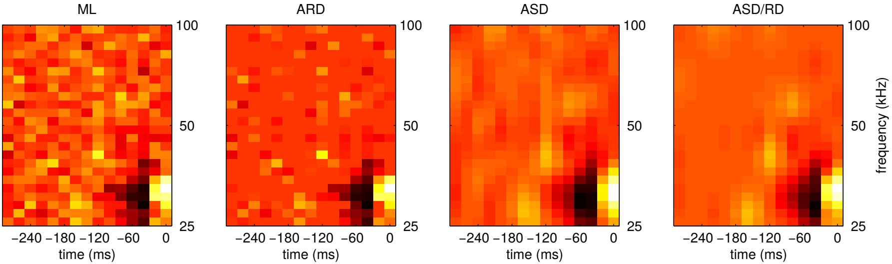
- sparsity [C_{ii} zero for many i] ARD
- smoothness [C_{ij} high for close i and j] ASD

Appropriate priors

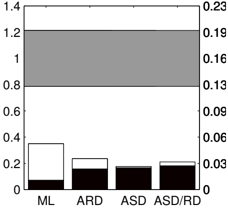


- sparsity [C_{ii} zero for many i] ARD
- smoothness [C_{ij} high for close i and j] ASD
- locality [C_{ii} high in a single region] ALD

Smoothness and sparsity (ASD/RD)



R2001011802G/20010731/pen14loc2poisshical020



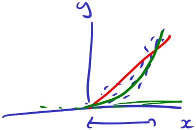
Beyond linearity

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* $x(t)$.

$$G_0[x(t); h^{(0)}] = h^{(0)}$$

$$G_1[x(t); h^{(1)}] = \int d\tau\, h^{(1)}(\tau) x(t-\tau)$$

stimulus power

$$G_2[x(t); h^{(2)}] = \iint d\tau_1\, d\tau_2\, h^{(2)}(\tau_1, \tau_2) x(t-\tau_1) x(t-\tau_2) - P \int d\tau_1\, h^{(2)}(\tau_1, \tau_1)$$

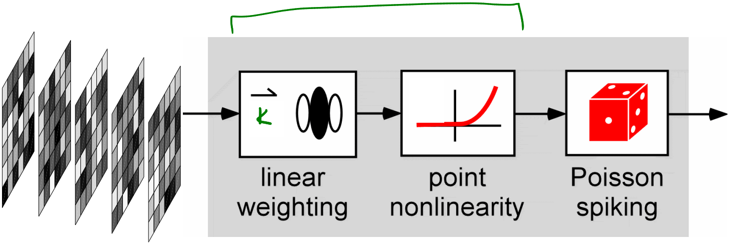
$$G_3[x(t); h^{(3)}] = \iiint d\tau_1\, d\tau_2\, d\tau_3\, h^{(3)}(\tau_1, \tau_2, \tau_3) x(t-\tau_1) x(t-\tau_2) x(t-\tau_3) - 3P \iint d\tau_1\, d\tau_2\, h^{(3)}(\tau_1, \tau_2, \tau_2) x(t-\tau_1)$$

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.

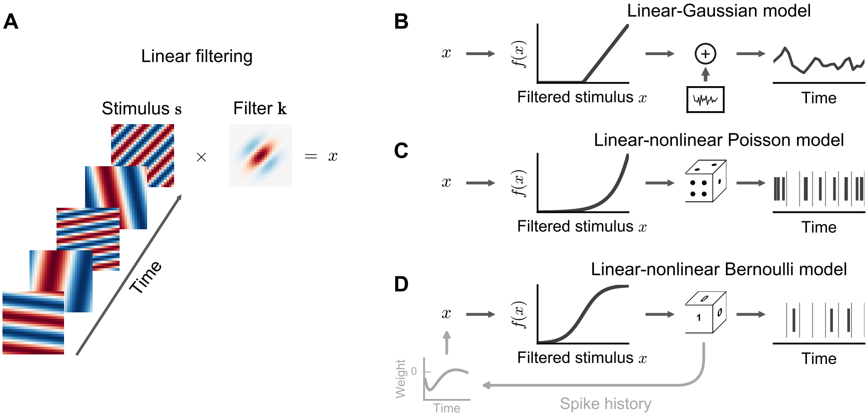
Cascade models

The LNP (Wiener) cascade



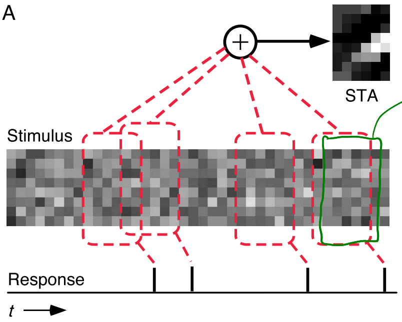
- Rectification addresses negative firing rates.
- Loose biophysical correspondance.

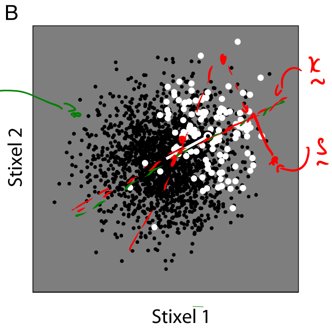
LNP cascades and noise

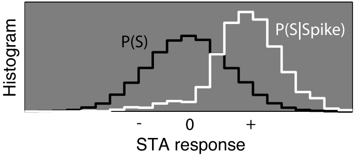


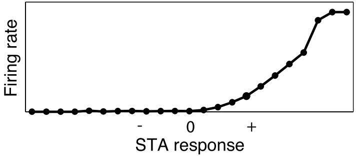
LNP estimation – the Spike-triggered ensemble

$\lambda(\underset{\sim}{s}) = f(\underset{\sim}{k}^T \cdot \underset{\sim}{s})$

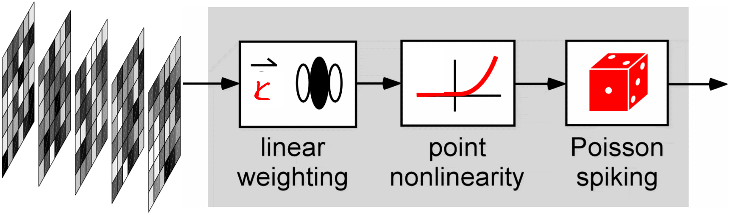






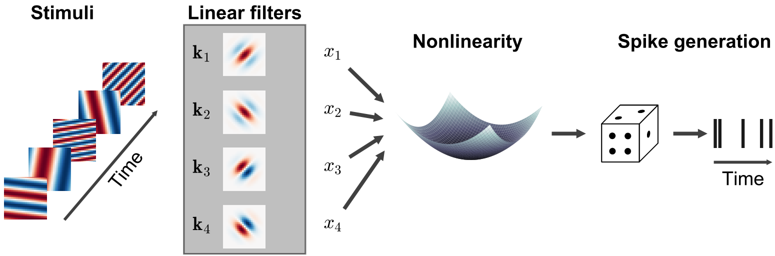


Single linear filter



- 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

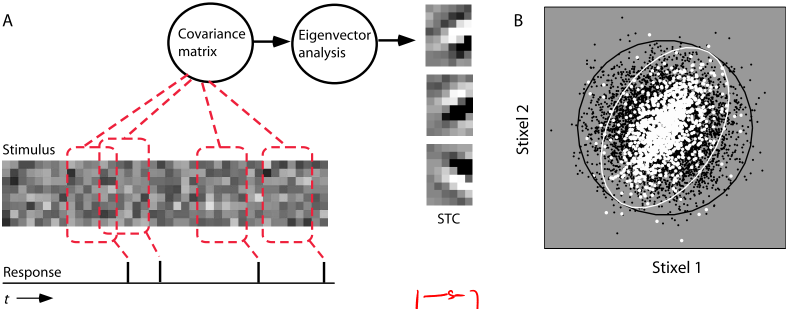


Distribution changes along relevant directions (and, usually, along all linear combinations of relevant directions).

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:

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

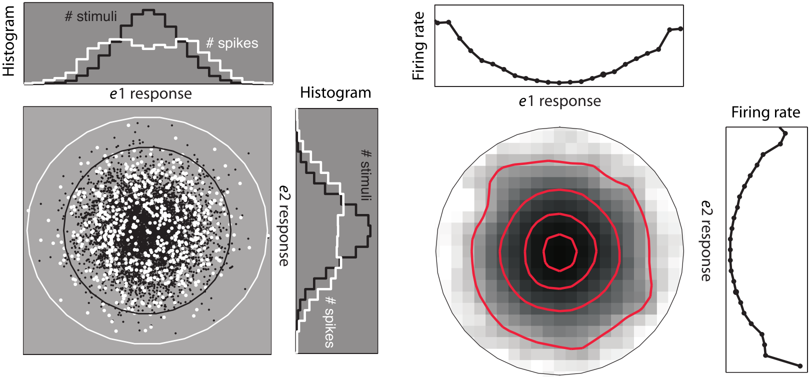
Choose directions with greatest change in variance:

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

$\Rightarrow$ find eigenvectors of $(C_{\text{prior}} - C_{\text{spike}})$ 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

⇒ 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.

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

  - 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

$\mathbb{E}[y] = f(\beta x)$

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

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

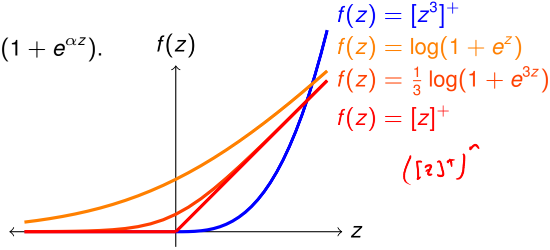
Generalises (for Poisson) to any f which is convex and log-concave:

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

$\frac{e^{-\mu}\mu^y}{y!}$

Includes:

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



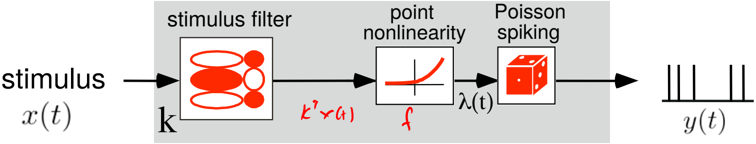
Generalised linear models

ML parameters found by

- gradient ascent
- IRLS → "Fisher scoring"

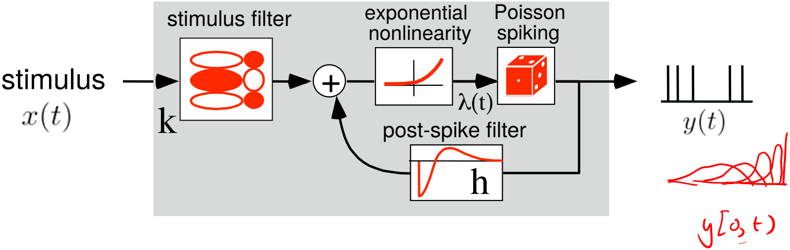
Regularisation by L_2 (quadratic) or L_1 (absolute value – sparse) penalties (MAP with Gaussian/Laplacian priors) preserves concavity.

Linear-Nonlinear-Poisson (GLM)



GLM with history-dependence

(Truccolo et al 04)



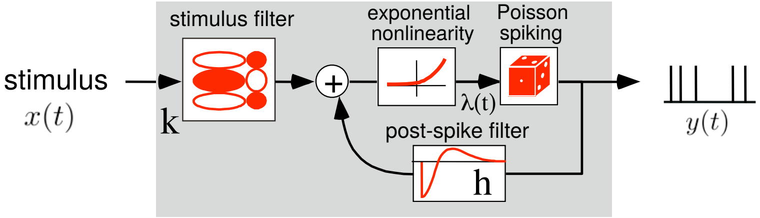
conditional intensity
(spike rate)

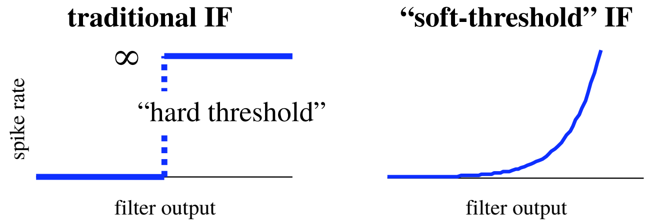
$$\lambda(t) = f(k \cdot x(t) + h \cdot y(t))$$

$$= e^{k \cdot x(t)} \cdot e^{h \cdot y(t)}$$

- 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

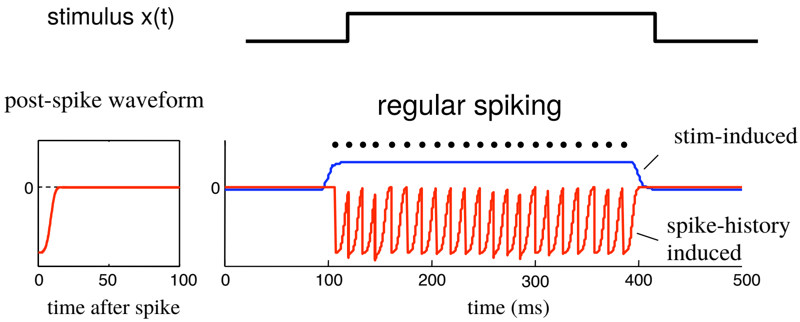
GLM with history-dependence



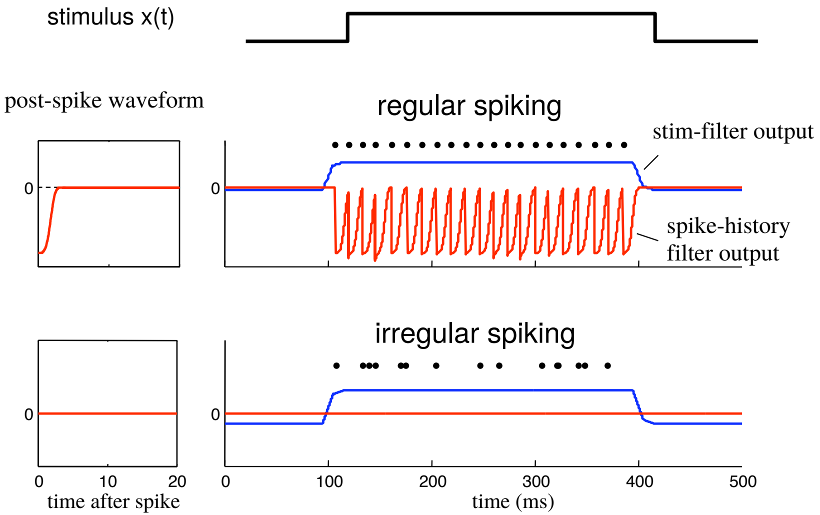


- "soft-threshold" approximation to Integrate-and-Fire model

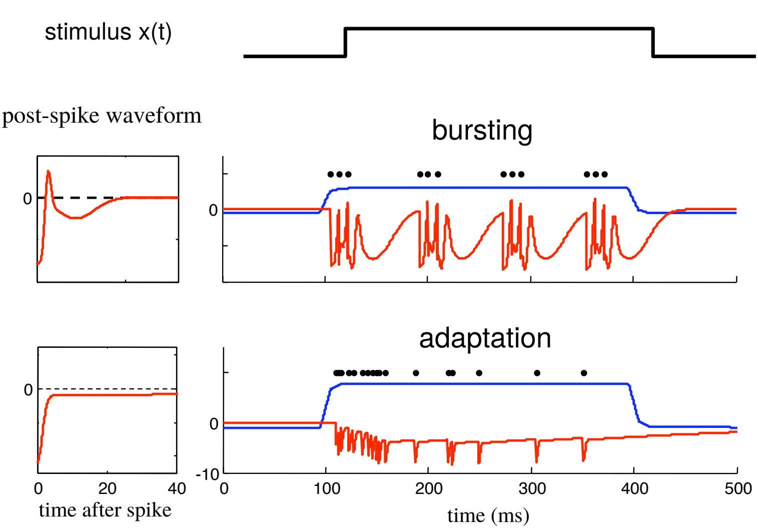
GLM dynamic behaviors



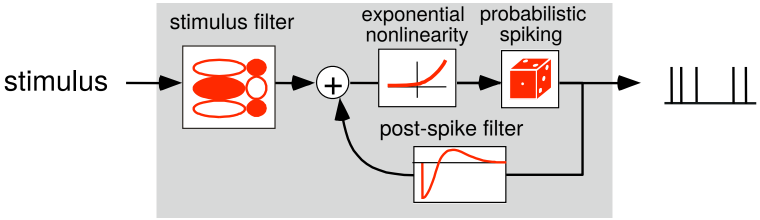
GLM dynamic behaviors



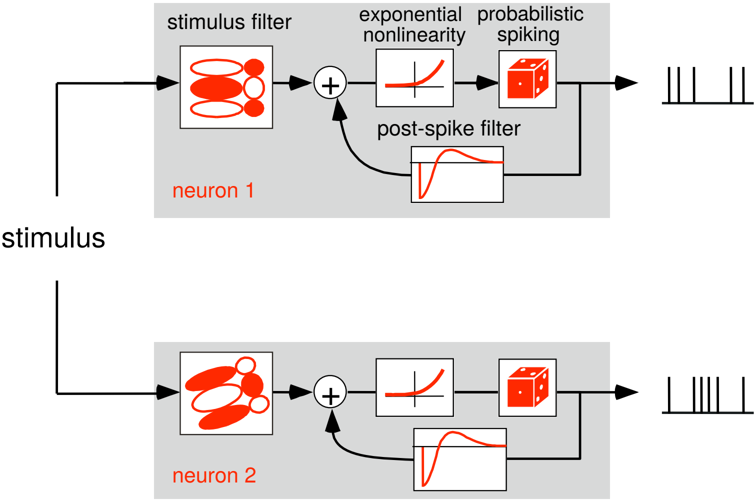
GLM dynamic behaviors



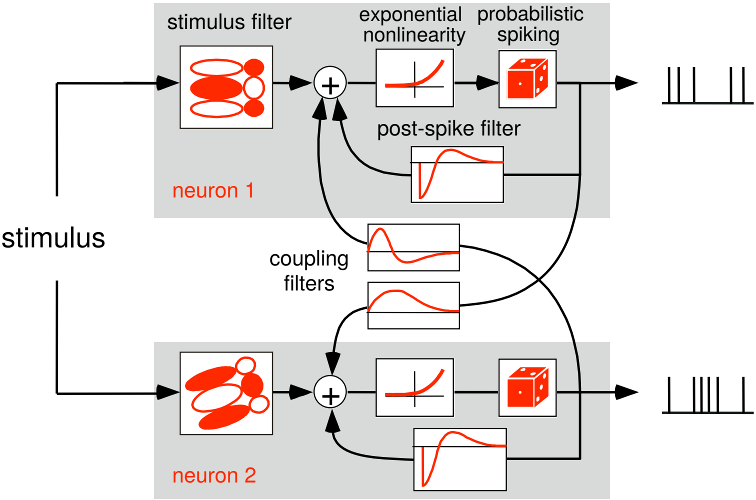
Generalized Linear Model (GLM)



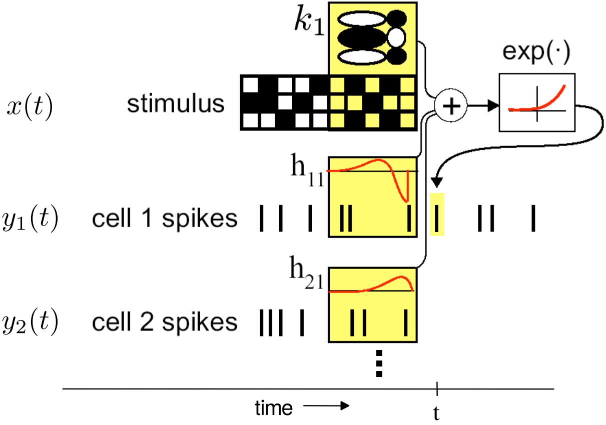
multi-neuron GLM



multi-neuron GLM



GLM equivalent diagram:



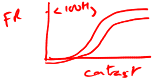
conditional intensity
(spike rate)

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

Non-LN models?

$\beta \cdot x \quad \beta \cdot f(x)$

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):

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

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, ...)

Input nonlinearities

The basic linear model (for sounds):

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

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

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

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

Input nonlinearities

The basic linear model (for sounds):

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

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

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

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

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

$$\hat{r}(i) = \sum_{jkl} w_{jk}^{\mathbf{tf}} w_l^{\mathbf{l}} M_{ijkl}$$

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

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

Multilinear models

Multilinear forms are straightforward to optimise by alternating least squares.

Cost function:

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

Minimise iteratively, defining *matrices*

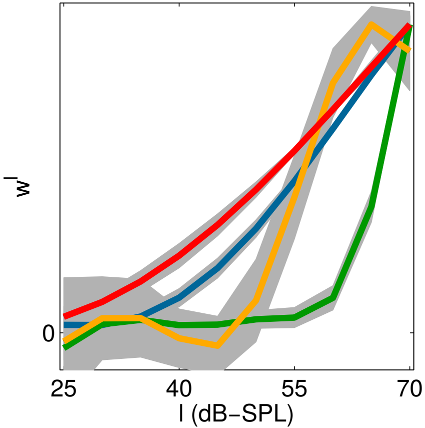
$$\mathbf{B} = \mathbf{w}^{\mathrm{l}} \bullet \mathbf{M} \quad \text{and} \quad \mathbf{A} = \mathbf{w}^{\mathrm{tf}} \bullet \mathbf{M}$$

and updating

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

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.

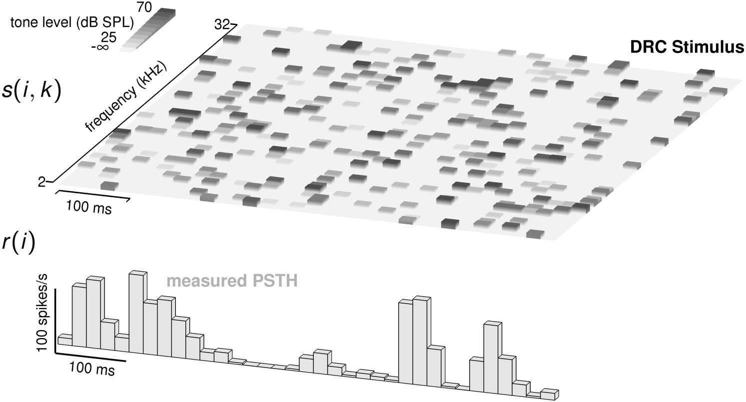
Variable (combination-dependent) input gain

- 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

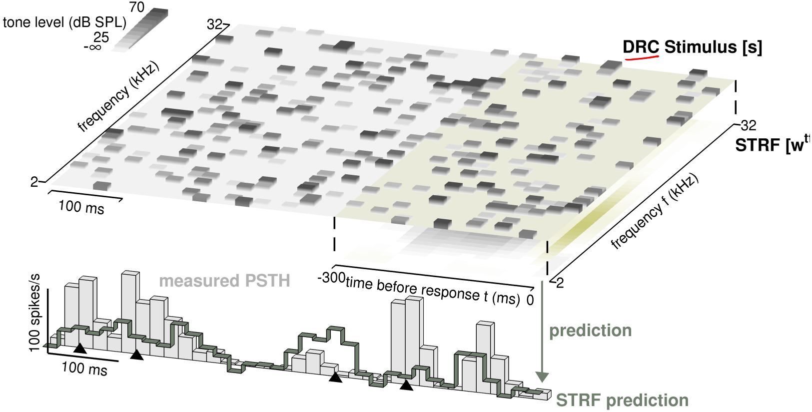
- Sensitivities to different points in sensory space are not independent.
- Rather, the sensitivity at one point depends on other elements of the stimulus that create a *local* sensory context.
- This context adjusts the input gain of the cell from moment to moment, dynamically refining the shape of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

Context-sensitive gain



Context-sensitive gain

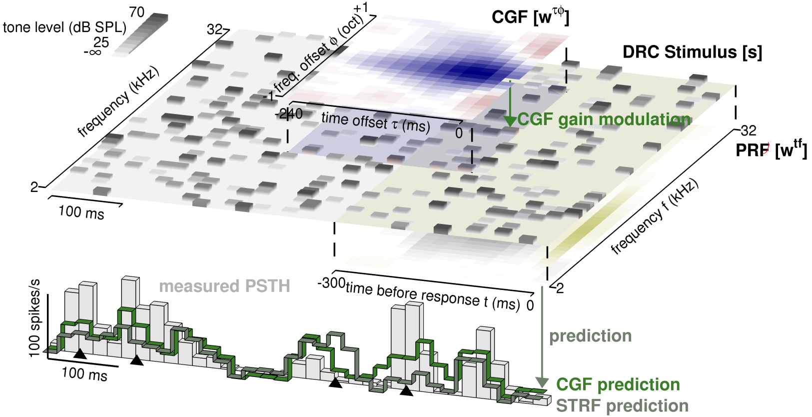
$$\hat{r}(i) = c + \sum_{j=0}^{J} \sum_{k=1}^{K} w^{tf}_{j+1,k} s(i-j,k)$$



Context-sensitive gain

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

$$M_{ijkmn} \approx s(i-j,k)\, s(i-j-m, k+n)$$



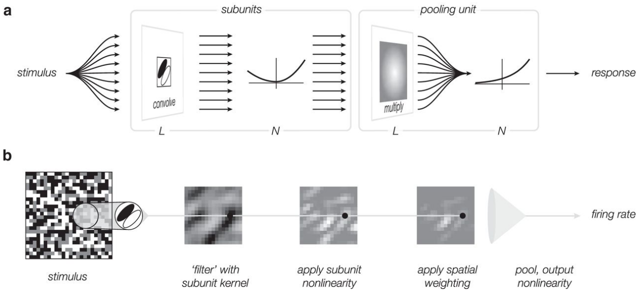
LNLN cascades

- 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.