Neural Encoding Models

Maneesh Sahani

Gatsby Computational Neuroscience Unit
University College London

July 2016

Neural Coding

The brain appears to be modular. Different structures and cortical areas compute, represent and transmit separate pieces of information.

The coding questions:

- What information is represented by a particular neural population?

- How is that information encoded?

Neural Coding

The brain appears to be modular. Different structures and cortical areas compute, represent and transmit separate pieces of information.

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 be modular. Different structures and cortical areas compute, represent and transmit separate pieces of information.

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

- How is that information encoded?

Neural Coding

The brain appears to be modular. Different structures and cortical areas compute, represent and transmit separate pieces of information.

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
  - encoded quantities might not be obvious: inferred latent variables, uncertainty . . .
- How is that information encoded?

Neural Coding

The brain appears to be modular. Different structures and cortical areas compute, represent and transmit separate pieces of information.

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
  - 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 be modular. Different structures and cortical areas compute, represent and transmit separate pieces of information.

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
  - 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 be modular. Different structures and cortical areas compute, represent and transmit separate pieces of information.

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
  - 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 be modular. Different structures and cortical areas compute, represent and transmit separate pieces of information.

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
  - 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 be modular. Different structures and cortical areas compute, represent and transmit separate pieces of information.

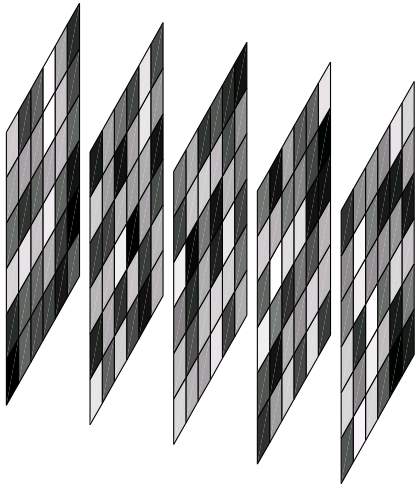
The coding questions:

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

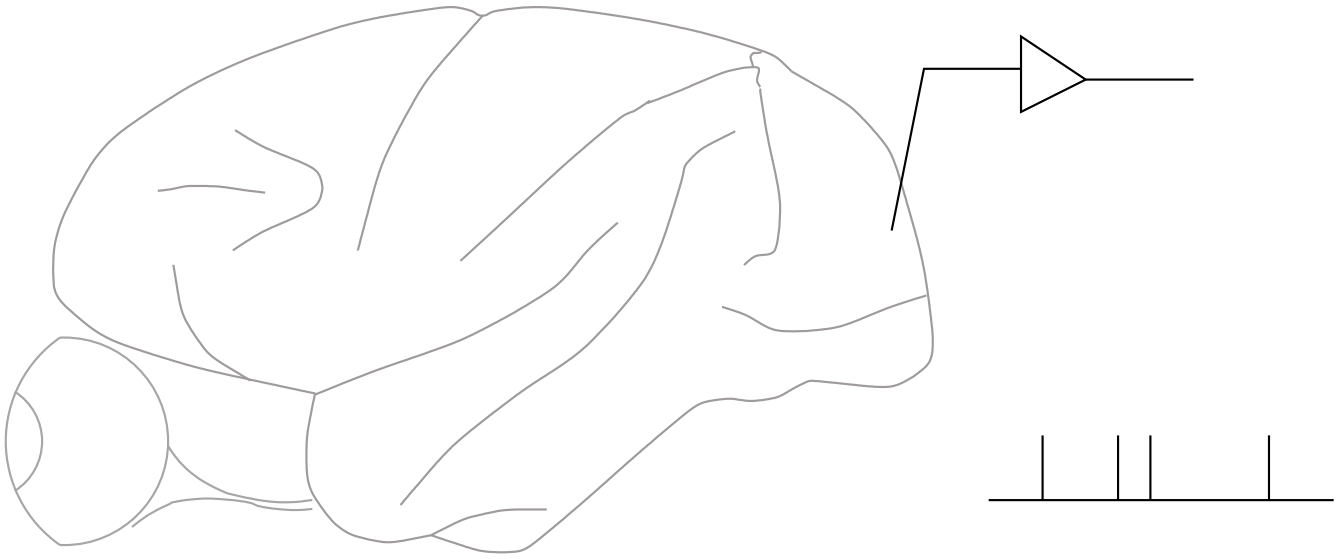
A complete answer will require convergence of theory and empirical results.

Computation plays a vital part in systematising empirical data.

Stimulus coding



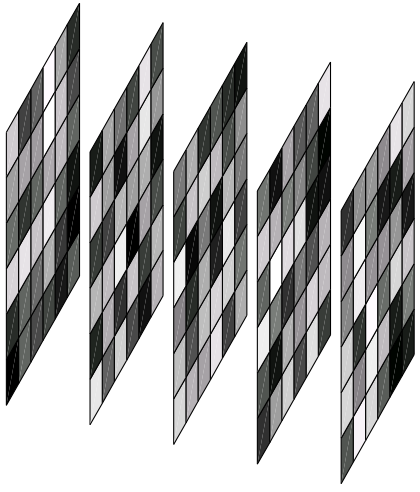
$s(t)$



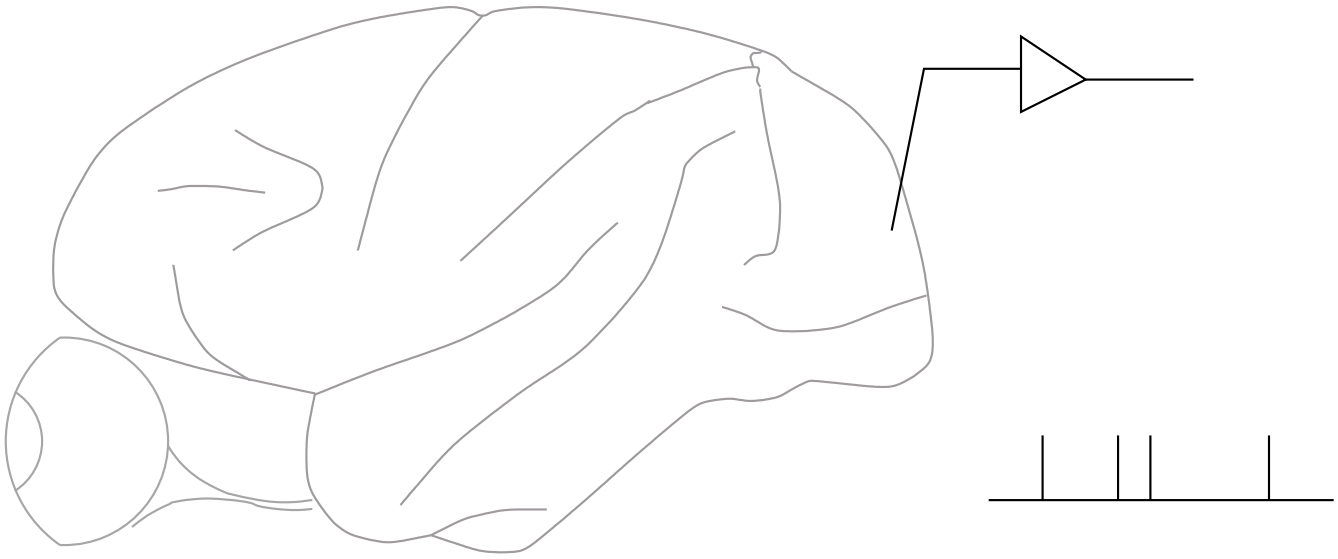
$r(t)$

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

Stimulus coding



$s(t)$



$r(t)$

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 $\lambda(t|s[0,t), H(t))$] from data.

General approach

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

Estimation theory

We will need a few ideas from estimation theory:

- estimators
- objective functions
- bias
- variance
- consistency
- validation
- regularisation

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:

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

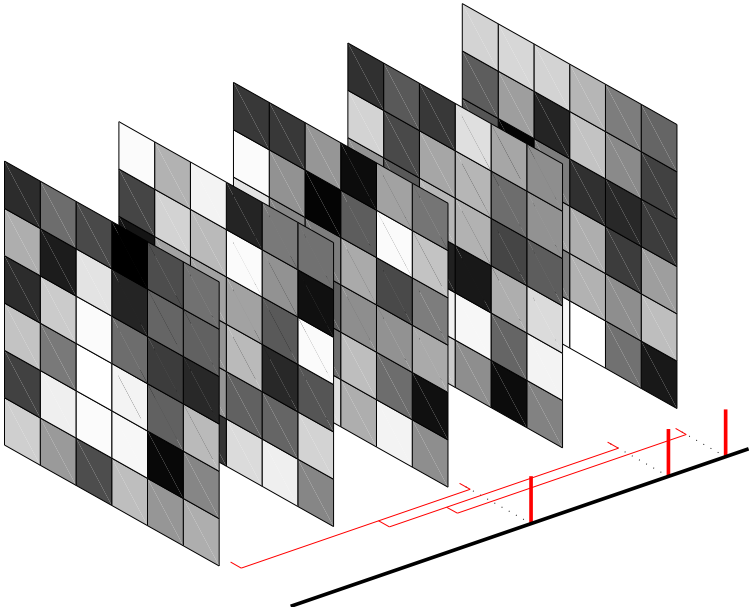
To fully model the response we need to identify λ. In general this depends on spike history $H(t)$ and network activity. Three options:

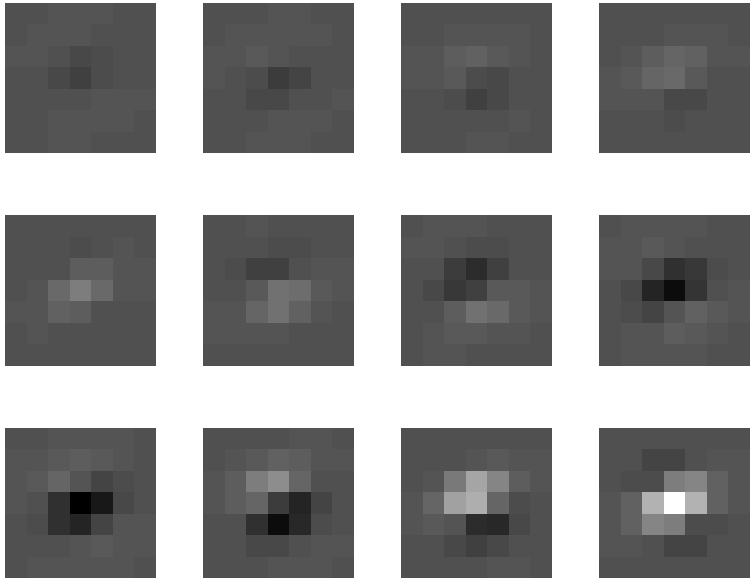
- Ignore the history dependence, take network activity as source of "noise" (i.e. assume firing is inhomogeneous Poisson or Cox process, conditioned on the stimulus).
- Average multiple trials to estimate the mean intensity (or PSTH)

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

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

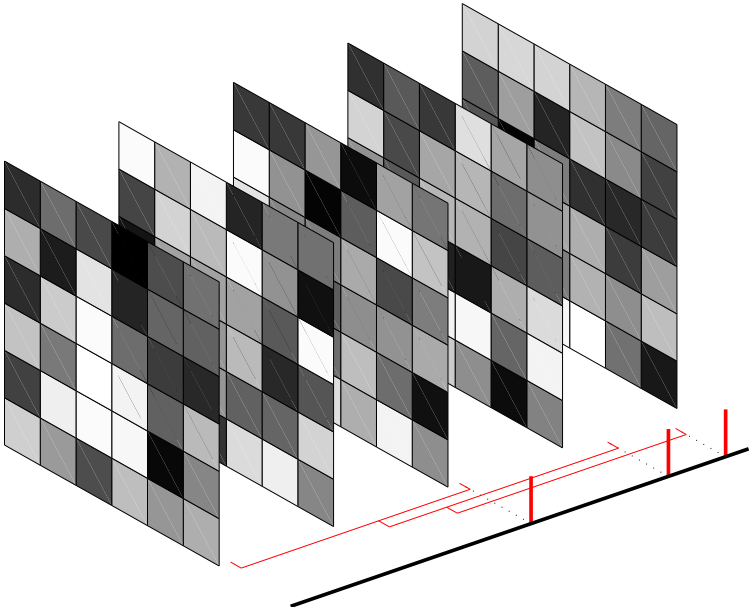
Spike-triggered average

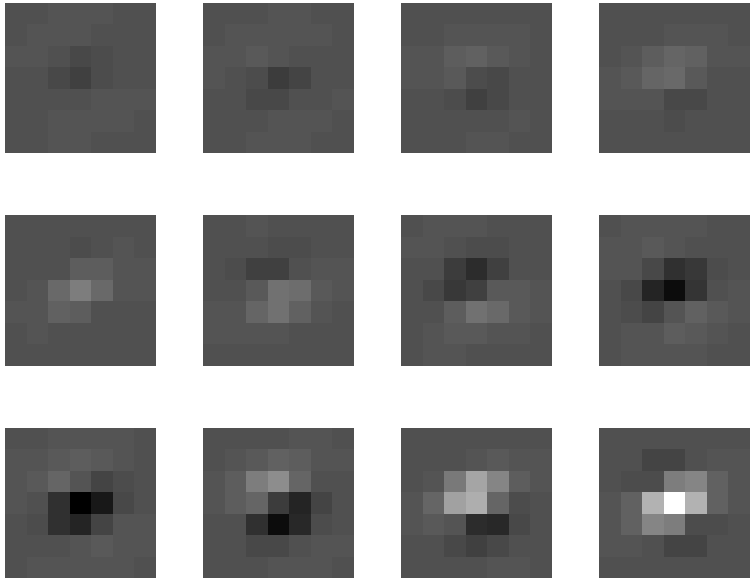




Decoding: mean of $\mathrm{P}\left(s \mid r = 1\right)$

Spike-triggered average





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

Encoding: predictive filter

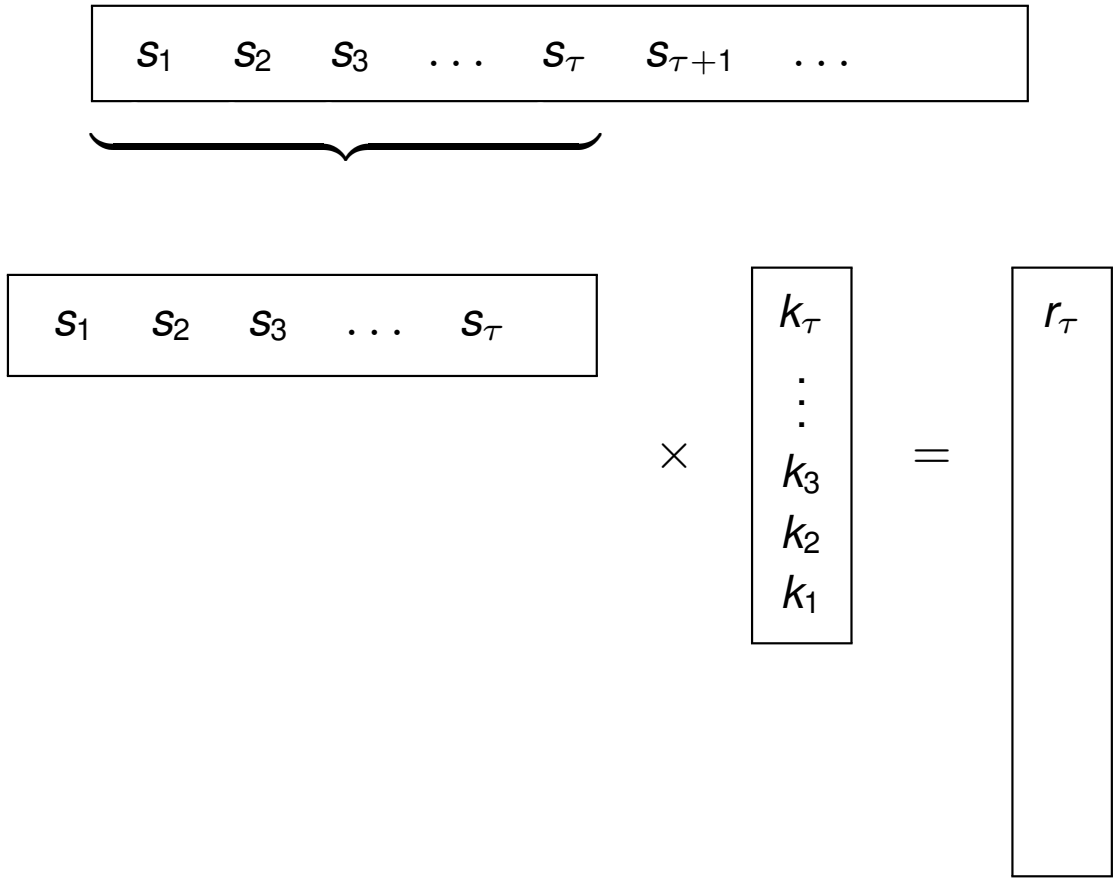
Linear regression

s_1	s_2	s_3	$\ldots$	s_τ	$s_{\tau+1}$	$\ldots$

$$r(t) = \int_0^\tau s(t - t')k(t')dt'$$

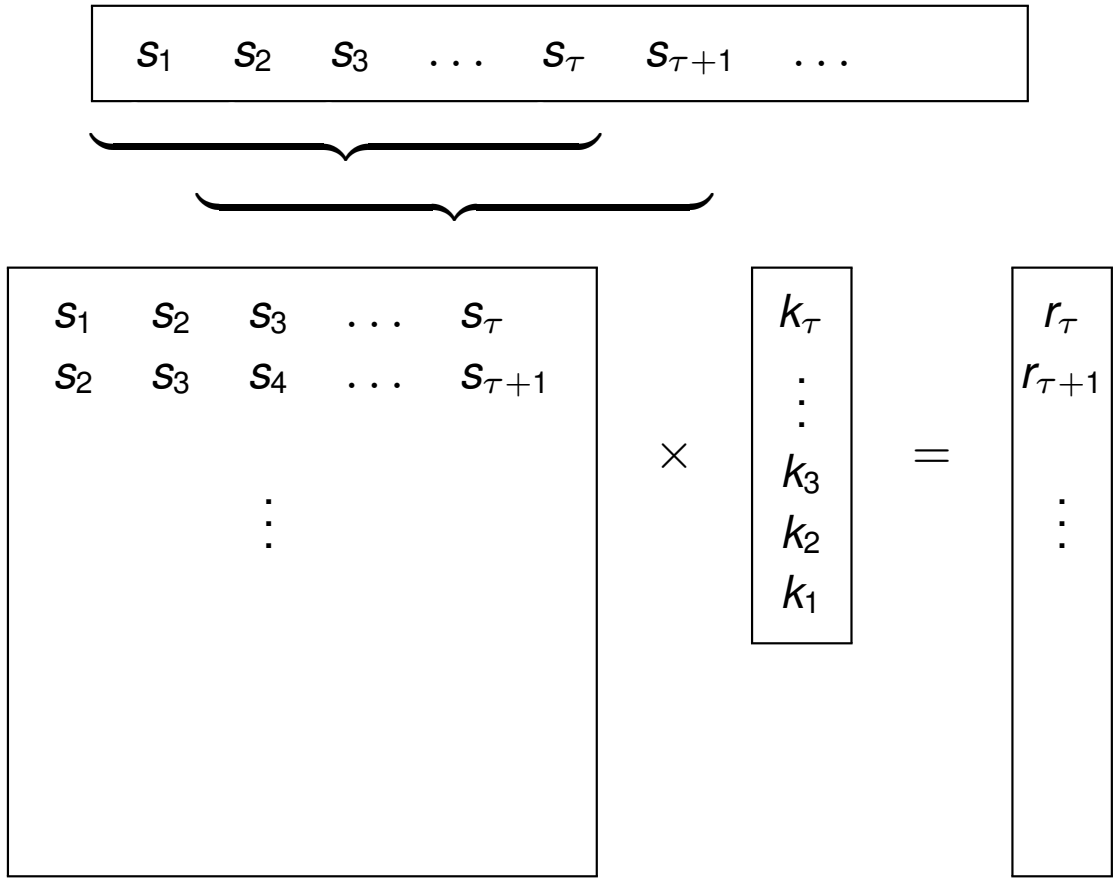
Linear regression

$$r(t) = \int_0^{\tau} s(t - t')k(t')dt'$$



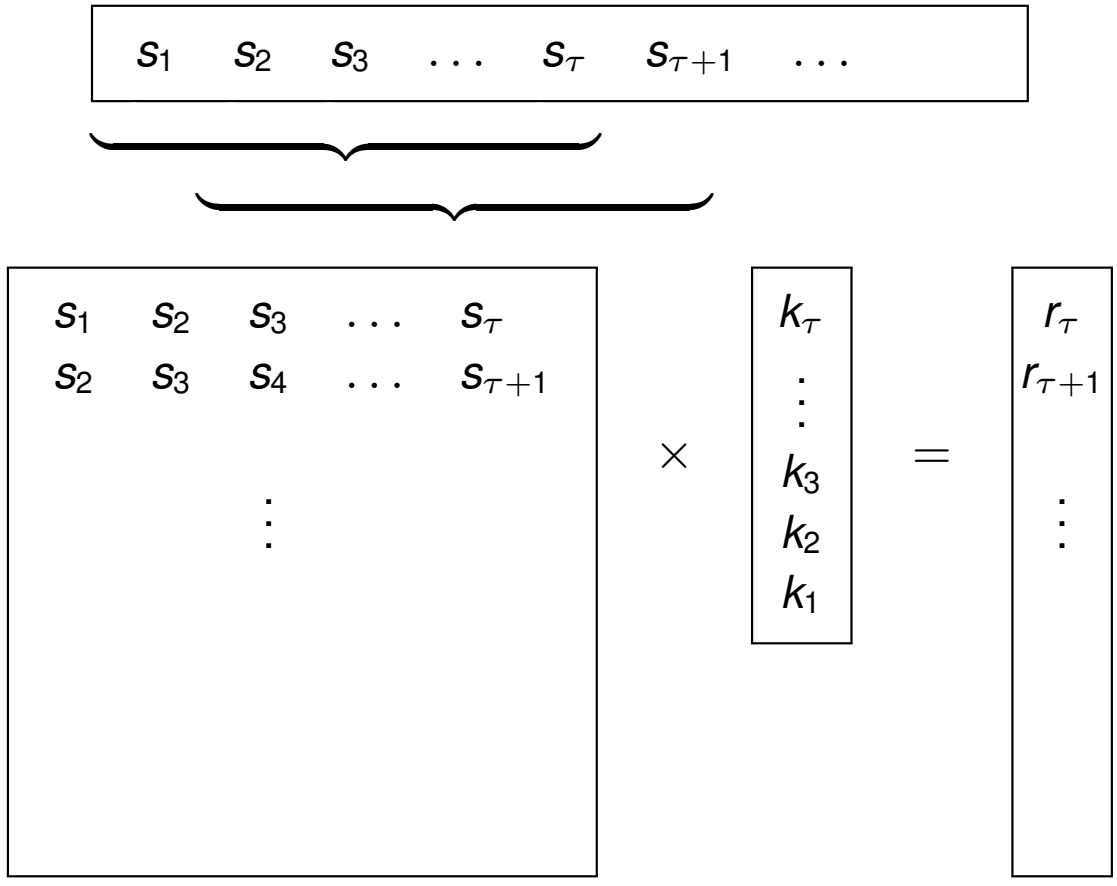
Linear regression

$$r(t) = \int_0^{\tau} s(t - t')k(t')dt'$$



Linear regression

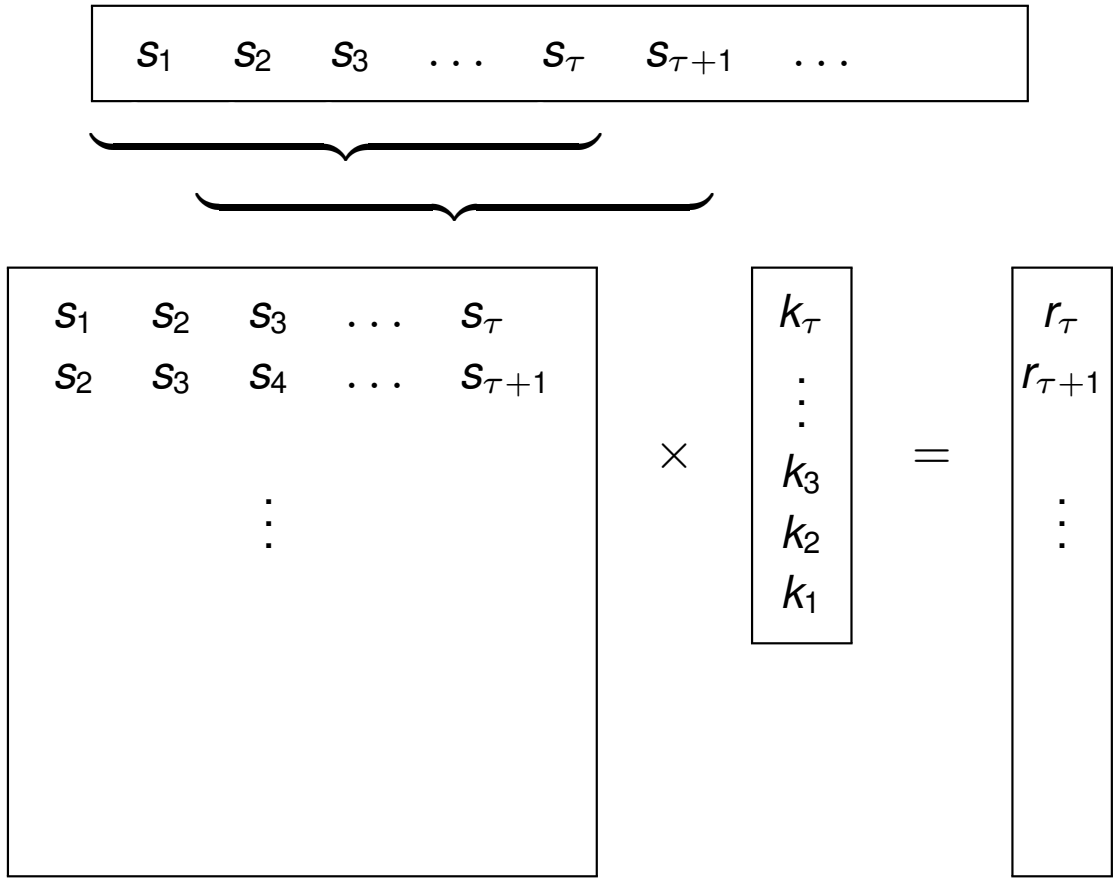
$$r(t) = \int_0^{\tau} s(t - t')k(t')dt'$$



$$Sk = R$$

Linear regression

$$r(t) = \int_0^{\tau} s(t - t')k(t')dt'$$



$$Sk = R$$

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

$$\hat{k} = \underbrace{(S^{\mathsf{T}} S)}_{\Sigma_{SS}}{}^{-1} \underbrace{(S^{\mathsf{T}} R)}_{\mathrm{STA}}$$

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:

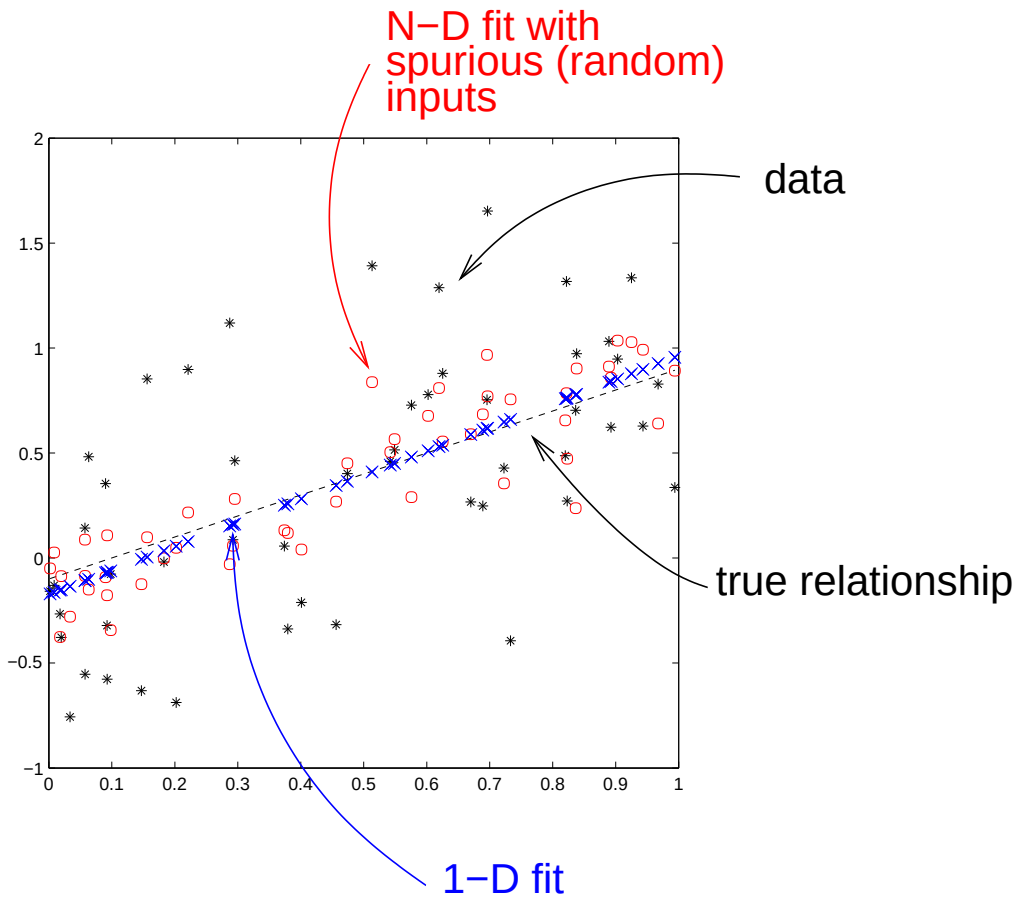
- overfitting and regularisation
  - standard methods for regression
- negative predicted rates
  - can model deviations from background
- real neurons aren't linear
  - models are still used extensively
  - interpretable suggestions of underlying sensitivity (but see later)
  - may provide unbiased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

Overfitting

Maximum-likelihood estimates often overfit to noise in the training data. Overfitting is a fundamental problem in data modelling. Even the correct model with the correct priors will overfit.

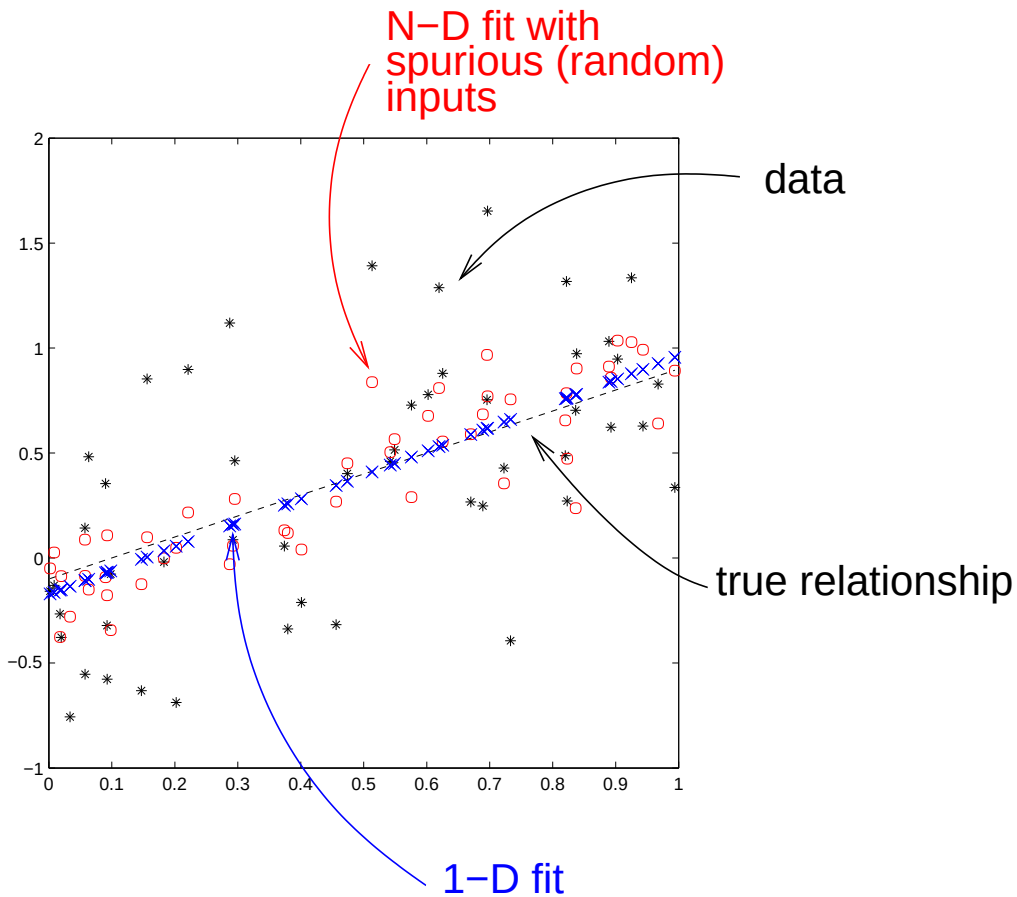
Overfitting

Maximum-likelihood estimates often overfit to noise in the training data. Overfitting is a fundamental problem in data modelling. Even the correct model with the correct priors will overfit.



Overfitting

Maximum-likelihood estimates often overfit to noise in the training data. Overfitting is a fundamental problem in data modelling. Even the correct model with the correct priors will overfit.



A common symptom is an unstable solution with large weights of opposite signs $\Rightarrow$ weight decay, ridge regression, or assume a prior distribution centered on zero weight.

Likelihood penalties

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

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

Multivariate Linear Regression

$$\begin{aligned}\ell &= \sum_i \log p(\mathbf{y}_i | \mathbf{x}_i, \mathrm{W}, \Sigma_y) \\ &= -\frac{N}{2} \log |2\pi\Sigma_y| - \frac{1}{2} \sum_i (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^{\mathsf{T}} \Sigma_y^{-1} (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)\end{aligned}$$

Multivariate Linear Regression

$$\begin{aligned}\ell &= \sum_i \log p(\mathbf{y}_i|\mathbf{x}_i, \mathsf{W}, \Sigma_y) \\ &= -\frac{N}{2}\log|2\pi\Sigma_y| - \frac{1}{2}\sum_i(\mathbf{y}_i - \mathsf{W}\mathbf{x}_i)^\mathsf{T}\Sigma_y^{-1}(\mathbf{y}_i - \mathsf{W}\mathbf{x}_i)\end{aligned}$$

$$\frac{\partial(-\ell)}{\partial \mathsf{W}} = \frac{\partial}{\partial \mathsf{W}}\left[\frac{N}{2}\log|2\pi\Sigma_y| + \frac{1}{2}\sum_i(\mathbf{y}_i - \mathsf{W}\mathbf{x}_i)^\mathsf{T}\Sigma_y^{-1}(\mathbf{y}_i - \mathsf{W}\mathbf{x}_i)\right]$$

Multivariate Linear Regression

$$\begin{aligned}\ell &= \sum_i \log p(\mathbf{y}_i|\mathbf{x}_i, \mathsf{W}, \Sigma_y) \\ &= -\frac{N}{2}\log|2\pi\Sigma_y| - \frac{1}{2}\sum_i (\mathbf{y}_i - \mathsf{W}\mathbf{x}_i)^\mathsf{T}\Sigma_y^{-1}(\mathbf{y}_i - \mathsf{W}\mathbf{x}_i)\end{aligned}$$

$$\begin{aligned}\frac{\partial(-\ell)}{\partial \mathsf{W}} &= \frac{\partial}{\partial \mathsf{W}}\left[\frac{N}{2}\log|2\pi\Sigma_y| + \frac{1}{2}\sum_i (\mathbf{y}_i - \mathsf{W}\mathbf{x}_i)^\mathsf{T}\Sigma_y^{-1}(\mathbf{y}_i - \mathsf{W}\mathbf{x}_i)\right] \\ &= \frac{1}{2}\sum_i \frac{\partial}{\partial \mathsf{W}}\left[(\mathbf{y}_i - \mathsf{W}\mathbf{x}_i)^\mathsf{T}\Sigma_y^{-1}(\mathbf{y}_i - \mathsf{W}\mathbf{x}_i)\right]\end{aligned}$$

Multivariate Linear Regression

$$\begin{aligned}\ell &= \sum_i \log p(\mathbf{y}_i|\mathbf{x}_i, \mathrm{W}, \Sigma_y) \\ &= -\frac{N}{2}\log|2\pi\Sigma_y| - \frac{1}{2}\sum_i (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^{\mathrm{T}}\Sigma_y^{-1}(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)\end{aligned}$$

$$\begin{aligned}\frac{\partial(-\ell)}{\partial \mathrm{W}} &= \frac{\partial}{\partial \mathrm{W}}\left[\frac{N}{2}\log|2\pi\Sigma_y| + \frac{1}{2}\sum_i (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^{\mathrm{T}}\Sigma_y^{-1}(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)\right] \\ &= \frac{1}{2}\sum_i \frac{\partial}{\partial \mathrm{W}}\left[(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^{\mathrm{T}}\Sigma_y^{-1}(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)\right] \\ &= \frac{1}{2}\sum_i \frac{\partial}{\partial \mathrm{W}}\left[\mathbf{y}_i^{\mathrm{T}}\Sigma_y^{-1}\mathbf{y}_i + \mathbf{x}_i^{\mathrm{T}}\mathrm{W}^{\mathrm{T}}\Sigma_y^{-1}\mathrm{W}\mathbf{x}_i - 2\mathbf{x}_i^{\mathrm{T}}\mathrm{W}^{\mathrm{T}}\Sigma_y^{-1}\mathbf{y}_i\right]\end{aligned}$$

Multivariate Linear Regression

$$\begin{aligned}\ell &= \sum_i \log p(\mathbf{y}_i|\mathbf{x}_i, \mathrm{W}, \Sigma_y) \\ &= -\frac{N}{2}\log|2\pi\Sigma_y| - \frac{1}{2}\sum_i (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^{\mathrm{T}}\Sigma_y^{-1}(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)\end{aligned}$$

$$\begin{aligned}\frac{\partial(-\ell)}{\partial \mathrm{W}} &= \frac{\partial}{\partial \mathrm{W}}\left[\frac{N}{2}\log|2\pi\Sigma_y| + \frac{1}{2}\sum_i (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^{\mathrm{T}}\Sigma_y^{-1}(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)\right] \\ &= \frac{1}{2}\sum_i \frac{\partial}{\partial \mathrm{W}}\left[(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^{\mathrm{T}}\Sigma_y^{-1}(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)\right] \\ &= \frac{1}{2}\sum_i \frac{\partial}{\partial \mathrm{W}}\left[\mathbf{y}_i^{\mathrm{T}}\Sigma_y^{-1}\mathbf{y}_i + \mathbf{x}_i^{\mathrm{T}}\mathrm{W}^{\mathrm{T}}\Sigma_y^{-1}\mathrm{W}\mathbf{x}_i - 2\mathbf{x}_i^{\mathrm{T}}\mathrm{W}^{\mathrm{T}}\Sigma_y^{-1}\mathbf{y}_i\right] \\ &= \frac{1}{2}\sum_i \left[\frac{\partial}{\partial \mathrm{W}}\mathrm{Tr}\left[\mathrm{W}^{\mathrm{T}}\Sigma_y^{-1}\mathrm{W}\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}}\right] - 2\frac{\partial}{\partial \mathrm{W}}\mathrm{Tr}\left[\mathrm{W}^{\mathrm{T}}\Sigma_y^{-1}\mathbf{y}_i\mathbf{x}_i^{\mathrm{T}}\right]\right]\end{aligned}$$

Multivariate Linear Regression

$$\begin{aligned}\ell &= \sum_i \log p(\mathbf{y}_i|\mathbf{x}_i, \mathrm{W}, \Sigma_y) \\ &= -\frac{N}{2}\log|2\pi\Sigma_y| - \frac{1}{2}\sum_i (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^{\mathrm{T}}\Sigma_y^{-1}(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)\end{aligned}$$

$$\begin{aligned}\frac{\partial(-\ell)}{\partial \mathrm{W}} &= \frac{\partial}{\partial \mathrm{W}}\left[\frac{N}{2}\log|2\pi\Sigma_y| + \frac{1}{2}\sum_i (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^{\mathrm{T}}\Sigma_y^{-1}(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)\right] \\ &= \frac{1}{2}\sum_i \frac{\partial}{\partial \mathrm{W}}\left[(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^{\mathrm{T}}\Sigma_y^{-1}(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)\right] \\ &= \frac{1}{2}\sum_i \frac{\partial}{\partial \mathrm{W}}\left[\mathbf{y}_i^{\mathrm{T}}\Sigma_y^{-1}\mathbf{y}_i + \mathbf{x}_i^{\mathrm{T}}\mathrm{W}^{\mathrm{T}}\Sigma_y^{-1}\mathrm{W}\mathbf{x}_i - 2\mathbf{x}_i^{\mathrm{T}}\mathrm{W}^{\mathrm{T}}\Sigma_y^{-1}\mathbf{y}_i\right] \\ &= \frac{1}{2}\sum_i \left[\frac{\partial}{\partial \mathrm{W}}\mathrm{Tr}\left[\mathrm{W}^{\mathrm{T}}\Sigma_y^{-1}\mathrm{W}\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}}\right] - 2\frac{\partial}{\partial \mathrm{W}}\mathrm{Tr}\left[\mathrm{W}^{\mathrm{T}}\Sigma_y^{-1}\mathbf{y}_i\mathbf{x}_i^{\mathrm{T}}\right]\right] \\ &= \frac{1}{2}\sum_i \left[2\Sigma_y^{-1}\mathrm{W}\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}} - 2\Sigma_y^{-1}\mathbf{y}_i\mathbf{x}_i^{\mathrm{T}}\right]\end{aligned}$$

Multivariate Linear Regression

$$\ell = \sum_i \log p(\mathbf{y}_i | \mathbf{x}_i, \mathrm{W}, \Sigma_y)$$
$$= -\frac{N}{2} \log |2\pi\Sigma_y| - \frac{1}{2} \sum_i (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^\mathrm{T} \Sigma_y^{-1} (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)$$

$$\frac{\partial(-\ell)}{\partial \mathrm{W}} = \frac{\partial}{\partial \mathrm{W}} \left[\frac{N}{2} \log |2\pi\Sigma_y| + \frac{1}{2} \sum_i (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^\mathrm{T} \Sigma_y^{-1} (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i) \right]$$
$$= \frac{1}{2} \sum_i \frac{\partial}{\partial \mathrm{W}} \left[(\mathbf{y}_i - \mathrm{W}\mathbf{x}_i)^\mathrm{T} \Sigma_y^{-1} (\mathbf{y}_i - \mathrm{W}\mathbf{x}_i) \right]$$
$$= \frac{1}{2} \sum_i \frac{\partial}{\partial \mathrm{W}} \left[\mathbf{y}_i^\mathrm{T} \Sigma_y^{-1} \mathbf{y}_i + \mathbf{x}_i^\mathrm{T} \mathrm{W}^\mathrm{T} \Sigma_y^{-1} \mathrm{W}\mathbf{x}_i - 2\mathbf{x}_i^\mathrm{T} \mathrm{W}^\mathrm{T} \Sigma_y^{-1} \mathbf{y}_i \right]$$
$$= \frac{1}{2} \sum_i \left[\frac{\partial}{\partial \mathrm{W}} \mathrm{Tr} \left[\mathrm{W}^\mathrm{T} \Sigma_y^{-1} \mathrm{W}\mathbf{x}_i \mathbf{x}_i^\mathrm{T} \right] - 2 \frac{\partial}{\partial \mathrm{W}} \mathrm{Tr} \left[\mathrm{W}^\mathrm{T} \Sigma_y^{-1} \mathbf{y}_i \mathbf{x}_i^\mathrm{T} \right] \right]$$
$$= \frac{1}{2} \sum_i \left[2\Sigma_y^{-1} \mathrm{W}\mathbf{x}_i \mathbf{x}_i^\mathrm{T} - 2\Sigma_y^{-1} \mathbf{y}_i \mathbf{x}_i^\mathrm{T} \right]$$

$$= 0 \Rightarrow \widehat{\mathrm{W}} = \sum_i \mathbf{y}_i \mathbf{x}_i^\mathrm{T} \left(\sum_i \mathbf{x}_i \mathbf{x}_i^\mathrm{T} \right)^{-1}$$

Bayesian Methods

Apply the basic rules of probability to learning from data.

Bayesian Methods

Apply the basic rules of probability to learning from data.

- Problem specification:

  Data: $\mathcal{D} = \{x_1, \ldots, x_n\}$ Models: $\mathcal{M}_1$, $\mathcal{M}_2$, etc. Parameters: θ_i (per model)

  Prior probability of models: $P(\mathcal{M}_i)$.

  Prior probabilities of model parameters: $P(\theta_i|\mathcal{M}_i)$

  Model of data given parameters (likelihood model): $P(x|\theta_i, \mathcal{M}_i)$

Bayesian Methods

Apply the basic rules of probability to learning from data.

- Problem specification:

  Data: $\mathcal{D} = \{x_1, \ldots, x_n\}$ Models: $\mathcal{M}_1$, $\mathcal{M}_2$, etc. Parameters: θ_i (per model)

  Prior probability of models: $P(\mathcal{M}_i)$.
  Prior probabilities of model parameters: $P(\theta_i|\mathcal{M}_i)$
  Model of data given parameters (likelihood model): $P(x|\theta_i, \mathcal{M}_i)$

- Data probability (likelihood)

$$P(\mathcal{D}|\theta_i, \mathcal{M}_i) = \prod_{j=1}^{n} P(x_j|\theta_i, \mathcal{M}_i) \equiv \mathcal{L}(\theta_i)$$

  (provided the data are independently and identically distributed (iid).)

Bayesian Methods

Apply the basic rules of probability to learning from data.

- Problem specification:

  Data: $\mathcal{D} = \{x_1, \ldots, x_n\}$ Models: $\mathcal{M}_1$, $\mathcal{M}_2$, etc. Parameters: θ_i (per model)

  Prior probability of models: $P(\mathcal{M}_i)$.
  Prior probabilities of model parameters: $P(\theta_i|\mathcal{M}_i)$
  Model of data given parameters (likelihood model): $P(x|\theta_i, \mathcal{M}_i)$

- Data probability (likelihood)

$$P(\mathcal{D}|\theta_i, \mathcal{M}_i) = \prod_{j=1}^{n} P(x_j|\theta_i, \mathcal{M}_i) \equiv \mathcal{L}(\theta_i)$$

  (provided the data are independently and identically distributed (iid).)

- Parameter learning (posterior):

$$P(\theta_i|\mathcal{D}, \mathcal{M}_i) = \frac{P(\mathcal{D}|\theta_i, \mathcal{M}_i)P(\theta_i|\mathcal{M}_i)}{P(\mathcal{D}|\mathcal{M}_i)}; \quad P(\mathcal{D}|\mathcal{M}_i) = \int d\theta_i \; P(\mathcal{D}|\theta_i, \mathcal{M}_i)P(\theta|\mathcal{M}_i)$$

Bayesian Methods

Apply the basic rules of probability to learning from data.

- Problem specification:

  Data: $\mathcal{D} = \{x_1, \ldots, x_n\}$ Models: $\mathcal{M}_1, \mathcal{M}_2$, etc. Parameters: θ_i (per model)

  Prior probability of models: $P(\mathcal{M}_i)$.
  Prior probabilities of model parameters: $P(\theta_i|\mathcal{M}_i)$
  Model of data given parameters (likelihood model): $P(x|\theta_i, \mathcal{M}_i)$

- Data probability (likelihood)

$$P(\mathcal{D}|\theta_i, \mathcal{M}_i) = \prod_{j=1}^{n} P(x_j|\theta_i, \mathcal{M}_i) \equiv \mathcal{L}(\theta_i)$$

  (provided the data are independently and identically distributed (iid).)

- Parameter learning (posterior):

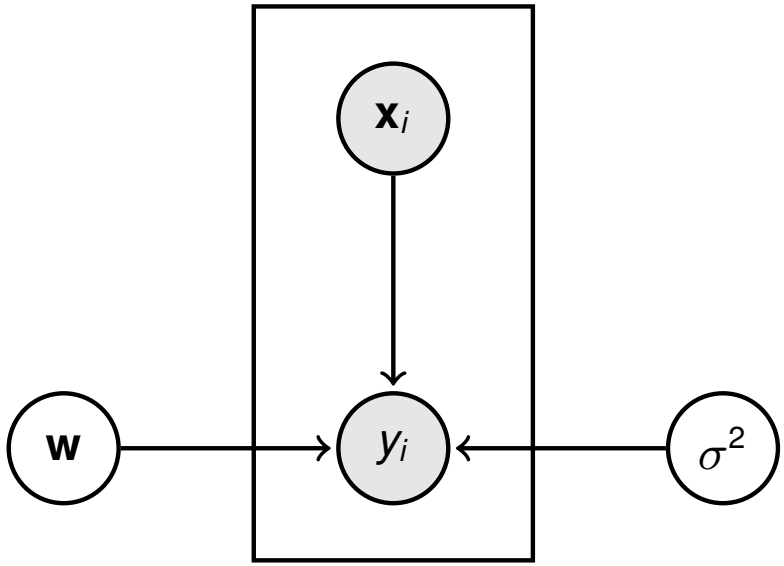
$$P(\theta_i|\mathcal{D}, \mathcal{M}_i) = \frac{P(\mathcal{D}|\theta_i, \mathcal{M}_i)P(\theta_i|\mathcal{M}_i)}{P(\mathcal{D}|\mathcal{M}_i)}; \quad P(\mathcal{D}|\mathcal{M}_i) = \int d\theta_i \; P(\mathcal{D}|\theta_i, \mathcal{M}_i)P(\theta|\mathcal{M}_i)$$

  $P(\mathcal{D}|\mathcal{M}_i)$ is called the marginal likelihood or evidence for $\mathcal{M}_i$. It is proportional to the posterior probability model $\mathcal{M}_i$ being the one that generated the data.

- Model selection:

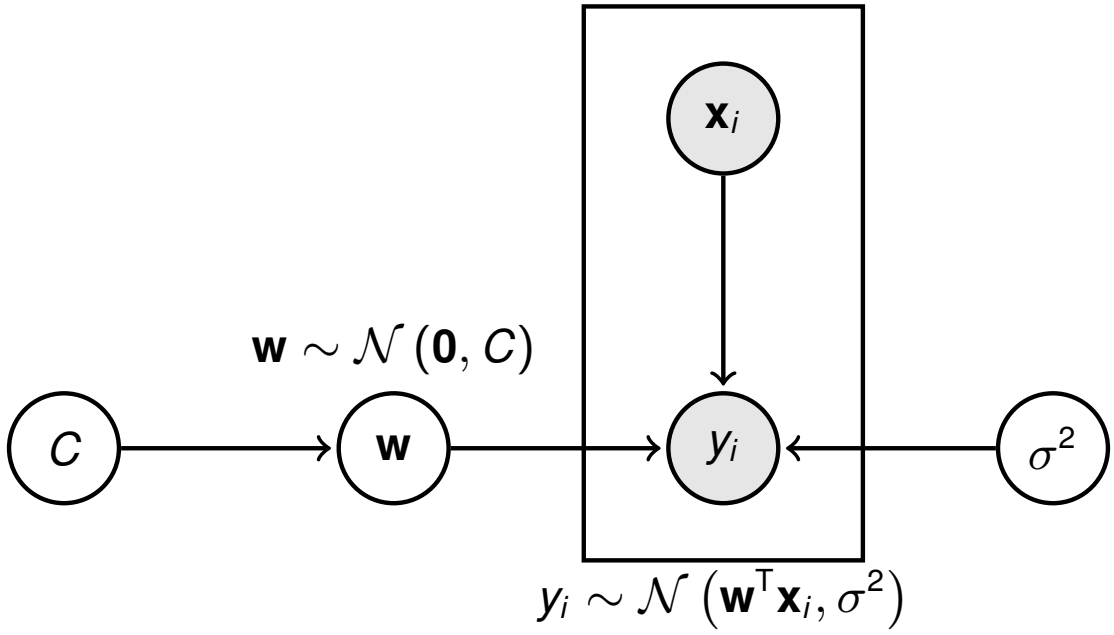
$$P(\mathcal{M}_i|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{M}_i)P(\mathcal{M}_i)}{P(\mathcal{D})}$$

Bayesian regularization



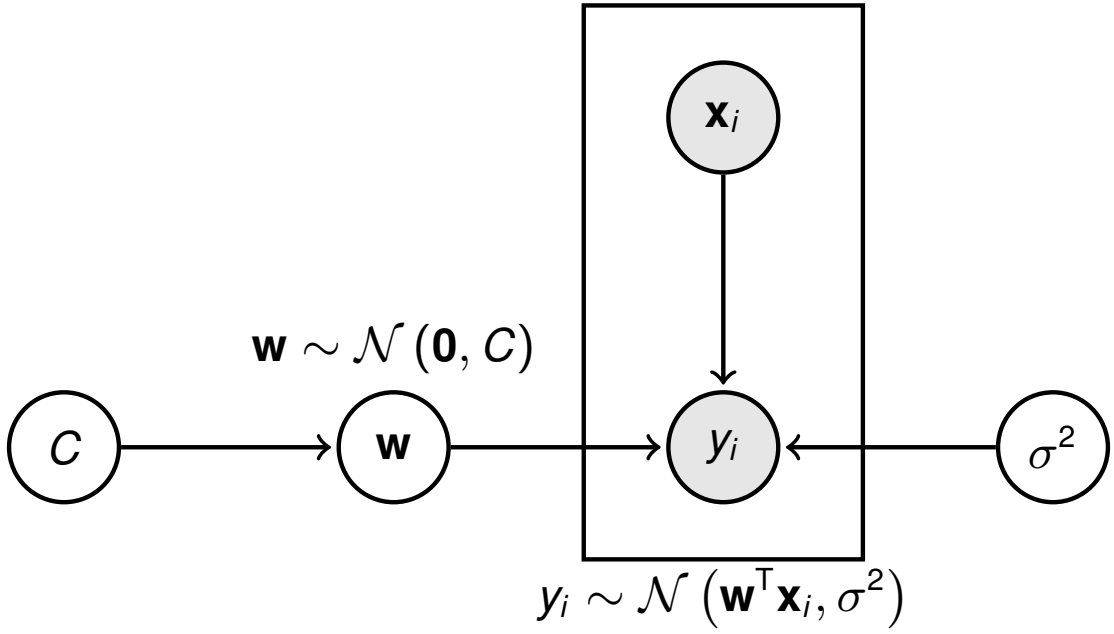
$$y_i \sim \mathcal{N}\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_i, \sigma^2\right)$$

Bayesian regularization



- Overfitting can be reduced by choosing appropriate “prior” distributions for the regression weights.

Bayesian regularization



- Overfitting can be reduced by choosing appropriate “prior” distributions for the regression weights.
- The distributions can be fine-tuned by choosing parameters for which there is greatest evidence or marginal likelihood.

Posterior estimation

Let y_i be scalar (so that W is a row vector) and write $\mathbf{w}$ for the column vector of weights.

Posterior estimation

Let y_i be scalar (so that W is a row vector) and write $\mathbf{w}$ for the column vector of weights.

A conjugate prior for $\mathbf{w}$ is

$$P(\mathbf{w}|C) = \mathcal{N}(\mathbf{0}, C)$$

Posterior estimation

Let y_i be scalar (so that W is a row vector) and write $\mathbf{w}$ for the column vector of weights.

A conjugate prior for $\mathbf{w}$ is

$$P(\mathbf{w}|C) = \mathcal{N}(\mathbf{0}, C)$$

Then the log posterior on $\mathbf{w}$ is

$$\log P(\mathbf{w}|\mathcal{D}, C, \sigma_y) = \log P(\mathcal{D}|\mathbf{w}, C, \sigma_y) + \log P(\mathbf{w}|C, \sigma_y) - \log P(\mathcal{D}|C, \sigma_y)$$

Posterior estimation

Let y_i be scalar (so that W is a row vector) and write $\mathbf{w}$ for the column vector of weights.

A conjugate prior for $\mathbf{w}$ is

$$P(\mathbf{w}|C) = \mathcal{N}(\mathbf{0}, C)$$

Then the log posterior on $\mathbf{w}$ is

$$\begin{aligned}\log P(\mathbf{w}|\mathcal{D}, C, \sigma_y) &= \log P(\mathcal{D}|\mathbf{w}, C, \sigma_y) + \log P(\mathbf{w}|C, \sigma_y) - \log P(\mathcal{D}|C, \sigma_y) \\ &= -\frac{1}{2}\mathbf{w}^{\mathsf{T}} C^{-1}\mathbf{w} - \frac{1}{2}\sum_i (y_i - \mathbf{w}^{\mathsf{T}}\mathbf{x}_i)^2 \sigma_y^{-2} + \text{const}\end{aligned}$$

Posterior estimation

Let y_i be scalar (so that W is a row vector) and write $\mathbf{w}$ for the column vector of weights.

A conjugate prior for $\mathbf{w}$ is

$$P(\mathbf{w}|C) = \mathcal{N}(\mathbf{0}, C)$$

Then the log posterior on $\mathbf{w}$ is

$$\begin{aligned}
\log P(\mathbf{w}|\mathcal{D}, C, \sigma_y) &= \log P(\mathcal{D}|\mathbf{w}, C, \sigma_y) + \log P(\mathbf{w}|C, \sigma_y) - \log P(\mathcal{D}|C, \sigma_y) \\
&= -\frac{1}{2}\mathbf{w}^{\mathsf{T}} C^{-1}\mathbf{w} - \frac{1}{2}\sum_i (y_i - \mathbf{w}^{\mathsf{T}}\mathbf{x}_i)^2 \sigma_y^{-2} + \text{const} \\
&= -\frac{1}{2}\mathbf{w}^{\mathsf{T}}(C^{-1} + \sigma_y^{-2}\sum_i \mathbf{x}_i\mathbf{x}_i^{\mathsf{T}})\mathbf{w} + \mathbf{w}^{\mathsf{T}}\sum_i (y_i\mathbf{x}_i)\sigma_y^{-2} + \text{const}
\end{aligned}$$

Posterior estimation

Let y_i be scalar (so that W is a row vector) and write $\mathbf{w}$ for the column vector of weights.

A conjugate prior for $\mathbf{w}$ is

$$P(\mathbf{w}|C) = \mathcal{N}(\mathbf{0}, C)$$

Then the log posterior on $\mathbf{w}$ is

$$\begin{aligned}
\log P(\mathbf{w}|\mathcal{D}, C, \sigma_y) &= \log P(\mathcal{D}|\mathbf{w}, C, \sigma_y) + \log P(\mathbf{w}|C, \sigma_y) - \log P(\mathcal{D}|C, \sigma_y) \\
&= -\frac{1}{2}\mathbf{w}^\mathsf{T} C^{-1}\mathbf{w} - \frac{1}{2}\sum_i (y_i - \mathbf{w}^\mathsf{T}\mathbf{x}_i)^2 \sigma_y^{-2} + \text{const} \\
&= -\frac{1}{2}\mathbf{w}^\mathsf{T} \underbrace{(C^{-1} + \sigma_y^{-2}\sum_i \mathbf{x}_i\mathbf{x}_i^\mathsf{T})}_{\Sigma_w^{-1}}\mathbf{w} + \mathbf{w}^\mathsf{T}\sum_i (y_i\mathbf{x}_i)\sigma_y^{-2} + \text{const}
\end{aligned}$$

Posterior estimation

Let y_i be scalar (so that W is a row vector) and write $\mathbf{w}$ for the column vector of weights.

A conjugate prior for $\mathbf{w}$ is

$$P(\mathbf{w}|C) = \mathcal{N}(\mathbf{0}, C)$$

Then the log posterior on $\mathbf{w}$ is

$$\begin{aligned}
\log P(\mathbf{w}|\mathcal{D}, C, \sigma_y) &= \log P(\mathcal{D}|\mathbf{w}, C, \sigma_y) + \log P(\mathbf{w}|C, \sigma_y) - \log P(\mathcal{D}|C, \sigma_y) \\
&= -\frac{1}{2}\mathbf{w}^\mathsf{T} C^{-1}\mathbf{w} - \frac{1}{2}\sum_i (y_i - \mathbf{w}^\mathsf{T}\mathbf{x}_i)^2 \sigma_y^{-2} + \text{const} \\
&= -\frac{1}{2}\mathbf{w}^\mathsf{T} \underbrace{(C^{-1} + \sigma_y^{-2}\sum_i \mathbf{x}_i\mathbf{x}_i^\mathsf{T})}_{\Sigma_w^{-1}}\mathbf{w} + \mathbf{w}^\mathsf{T}\sum_i (y_i\mathbf{x}_i)\sigma_y^{-2} + \text{const} \\
&= -\frac{1}{2}\mathbf{w}^\mathsf{T}\Sigma_w^{-1}\mathbf{w} + \mathbf{w}^\mathsf{T}\Sigma_w^{-1}\,\Sigma_w \sum_i (y_i\mathbf{x}_i)\sigma_y^{-2} + \text{const}
\end{aligned}$$

Posterior estimation

Let y_i be scalar (so that W is a row vector) and write $\mathbf{w}$ for the column vector of weights.

A conjugate prior for $\mathbf{w}$ is

$$P(\mathbf{w}|C) = \mathcal{N}(\mathbf{0}, C)$$

Then the log posterior on $\mathbf{w}$ is

$$\begin{aligned}
\log P(\mathbf{w}|\mathcal{D}, C, \sigma_y) &= \log P(\mathcal{D}|\mathbf{w}, C, \sigma_y) + \log P(\mathbf{w}|C, \sigma_y) - \log P(\mathcal{D}|C, \sigma_y) \\
&= -\frac{1}{2}\mathbf{w}^\mathsf{T} C^{-1}\mathbf{w} - \frac{1}{2}\sum_i (y_i - \mathbf{w}^\mathsf{T}\mathbf{x}_i)^2 \sigma_y^{-2} + \text{const} \\
&= -\frac{1}{2}\mathbf{w}^\mathsf{T} \underbrace{(C^{-1} + \sigma_y^{-2}\sum_i \mathbf{x}_i\mathbf{x}_i^\mathsf{T})}_{\Sigma_w^{-1}}\mathbf{w} + \mathbf{w}^\mathsf{T}\sum_i (y_i\mathbf{x}_i)\sigma_y^{-2} + \text{const} \\
&= -\frac{1}{2}\mathbf{w}^\mathsf{T}\Sigma_w^{-1}\mathbf{w} + \mathbf{w}^\mathsf{T}\Sigma_w^{-1}\underbrace{\Sigma_w \sum_i (y_i\mathbf{x}_i)\sigma_y^{-2}}_{\boldsymbol{\mu}_w} + \text{const}
\end{aligned}$$

Posterior estimation

Let y_i be scalar (so that W is a row vector) and write $\mathbf{w}$ for the column vector of weights.

A conjugate prior for $\mathbf{w}$ is

$$P(\mathbf{w}|C) = \mathcal{N}(\mathbf{0}, C)$$

Then the log posterior on $\mathbf{w}$ is

$$\begin{aligned}
\log P(\mathbf{w}|\mathcal{D}, C, \sigma_y) &= \log P(\mathcal{D}|\mathbf{w}, C, \sigma_y) + \log P(\mathbf{w}|C, \sigma_y) - \log P(\mathcal{D}|C, \sigma_y) \\
&= -\frac{1}{2}\mathbf{w}^{\mathsf{T}} C^{-1}\mathbf{w} - \frac{1}{2}\sum_i (y_i - \mathbf{w}^{\mathsf{T}}\mathbf{x}_i)^2 \sigma_y^{-2} + \text{const} \\
&= -\frac{1}{2}\mathbf{w}^{\mathsf{T}} \underbrace{(C^{-1} + \sigma_y^{-2}\sum_i \mathbf{x}_i\mathbf{x}_i^{\mathsf{T}})}_{\Sigma_w^{-1}}\mathbf{w} + \mathbf{w}^{\mathsf{T}}\sum_i (y_i\mathbf{x}_i)\sigma_y^{-2} + \text{const} \\
&= -\frac{1}{2}\mathbf{w}^{\mathsf{T}}\Sigma_w^{-1}\mathbf{w} + \mathbf{w}^{\mathsf{T}}\Sigma_w^{-1} \underbrace{\Sigma_w \sum_i (y_i\mathbf{x}_i)\sigma_y^{-2}}_{\boldsymbol{\mu}_w} + \text{const} \\
&= \log\ \mathcal{N}\left(\Sigma_w \textstyle\sum_i (y_i\mathbf{x}_i)\sigma_y^{-2}, \Sigma_w\right)
\end{aligned}$$

The evidence for linear regression

- The posterior on $\mathbf{w}$ is normal, with variance $\Sigma = (\frac{XX^{\mathsf{T}}}{\sigma^2} + C^{-1})^{-1}$ and mean $\mu = \Sigma \frac{XY^{\mathsf{T}}}{\sigma^2}$.

  Note: X is a matrix where columns are input vectors, and Y is a row vector of corresponding predicted outputs.

The evidence for linear regression

- The posterior on $\mathbf{w}$ is normal, with variance $\Sigma = (\frac{XX^{\mathsf{T}}}{\sigma^2} + C^{-1})^{-1}$ and mean $\mu = \Sigma \frac{XY^{\mathsf{T}}}{\sigma^2}$.

  Note: X is a matrix where columns are input vectors, and Y is a row vector of corresponding predicted outputs.

- The evidence, $\mathcal{E}(C, \sigma^2) = \int P(Y|X, \mathbf{w}, \sigma^2) P(\mathbf{w}|C)\, d\mathbf{w}$, is given by:

$$\mathcal{E}(C, \sigma^2) = \sqrt{\frac{|2\pi\Sigma|}{|2\pi\sigma^2 I|\,|2\pi C|}} \exp\left(-\frac{1}{2} Y \left(\frac{I}{\sigma^2} - \frac{X^{\mathsf{T}}\Sigma X}{\sigma^4}\right) Y^{\mathsf{T}}\right)$$

The evidence for linear regression

- The posterior on $\mathbf{w}$ is normal, with variance $\Sigma = (\frac{XX^T}{\sigma^2} + C^{-1})^{-1}$ and mean $\mu = \Sigma \frac{XY^T}{\sigma^2}$.

  Note: X is a matrix where columns are input vectors, and Y is a row vector of corresponding predicted outputs.

- The evidence, $\mathcal{E}(C, \sigma^2) = \int P(Y|X, \mathbf{w}, \sigma^2) P(\mathbf{w}|C)\, d\mathbf{w}$, is given by:

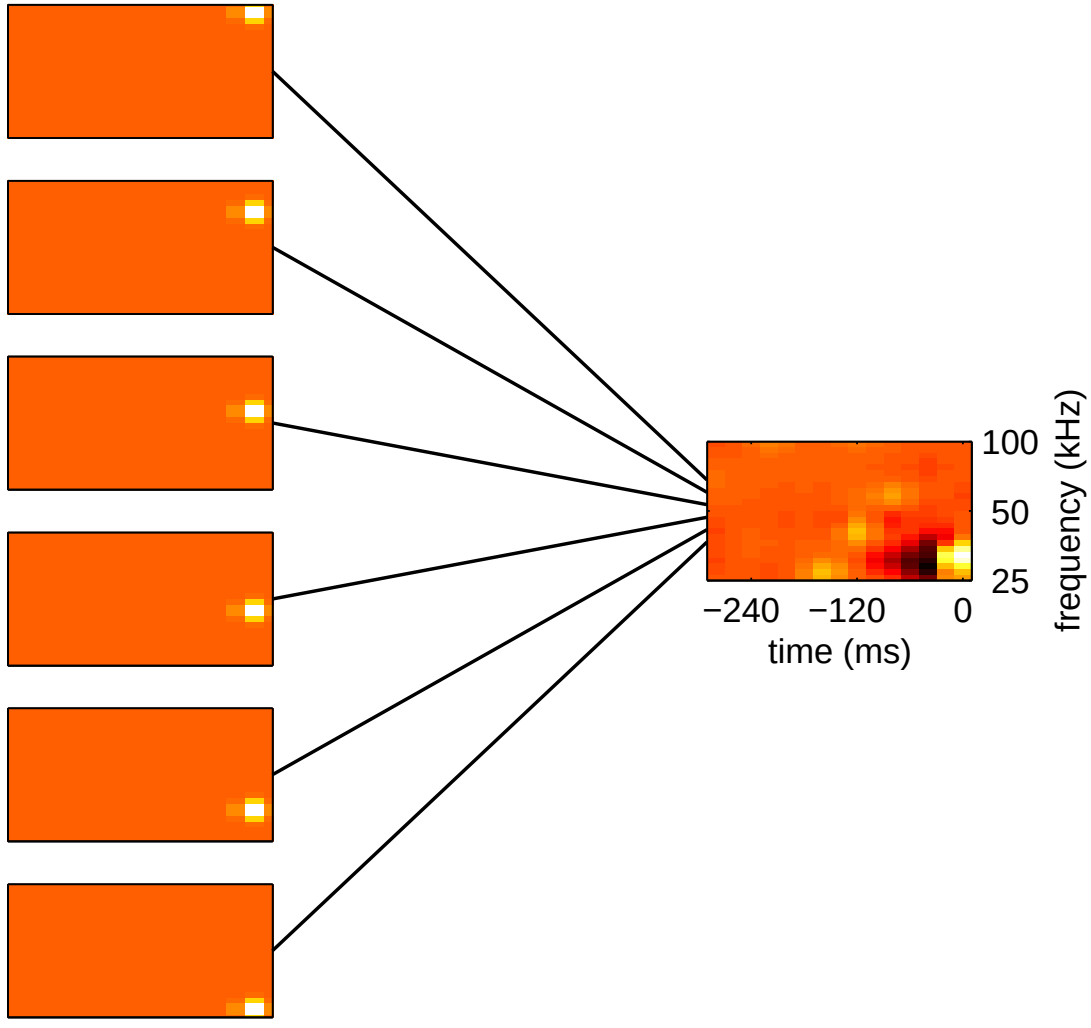
$$\mathcal{E}(C, \sigma^2) = \sqrt{\frac{|2\pi\Sigma|}{|2\pi\sigma^2 I|\,|2\pi C|}} \exp\left(-\frac{1}{2} Y \left(\frac{I}{\sigma^2} - \frac{X^T \Sigma X}{\sigma^4}\right) Y^T\right)$$

- For optimization, general forms for the gradients are available. If θ is a parameter in C:

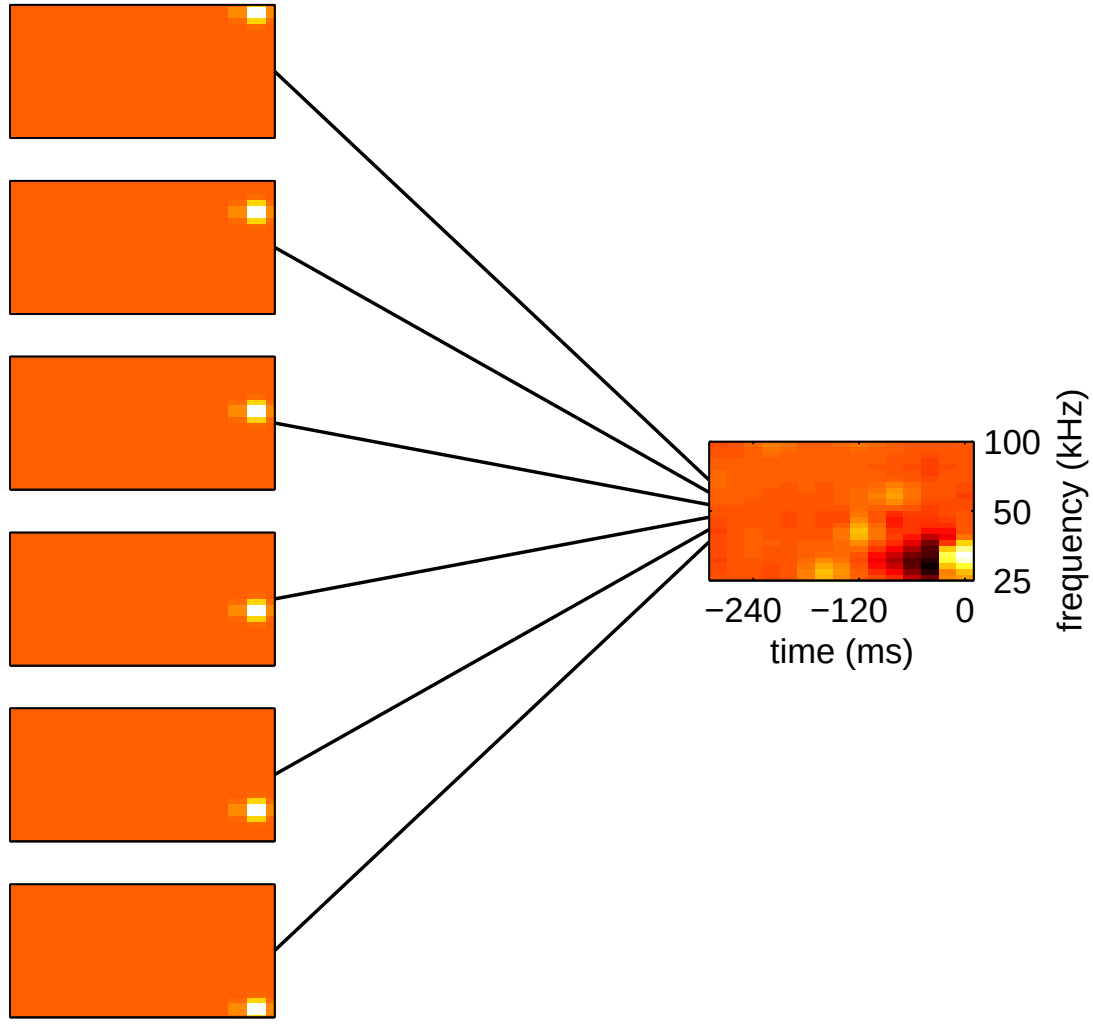
$$\frac{\partial}{\partial\theta} \log \mathcal{E}(C, \sigma^2) = \frac{1}{2} \mathrm{Tr}\left[(C - \Sigma - \mu\mu^T) \frac{\partial}{\partial\theta} C^{-1}\right]$$

$$\frac{\partial}{\partial\sigma^2} \log \mathcal{E}(C, \sigma^2) = \frac{1}{\sigma^2}\left(-N + \mathrm{Tr}\left[I - \Sigma C^{-1}\right] + \frac{1}{\sigma^2}(Y - \mu^T X)(Y - \mu^T X)^T\right)$$

Appropriate priors

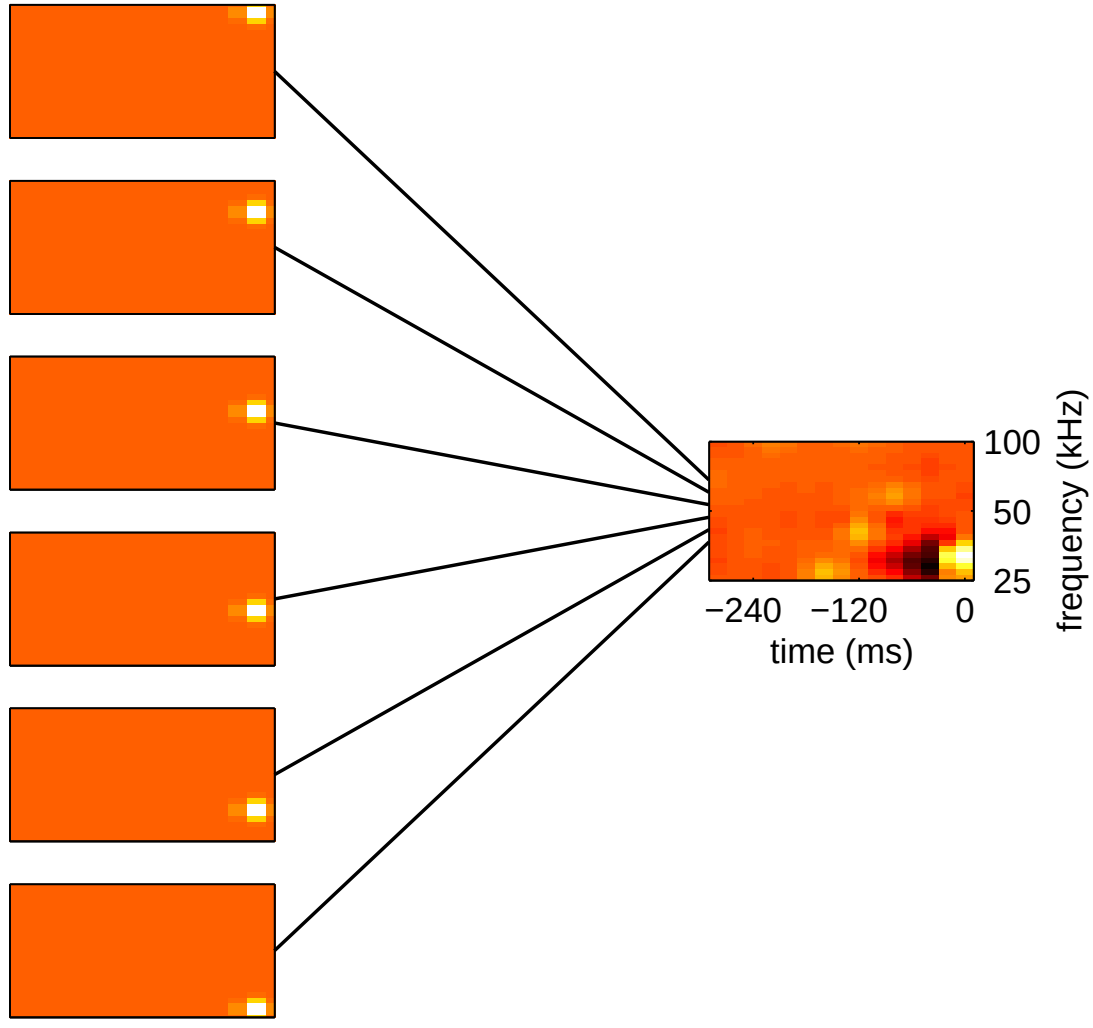


Appropriate priors



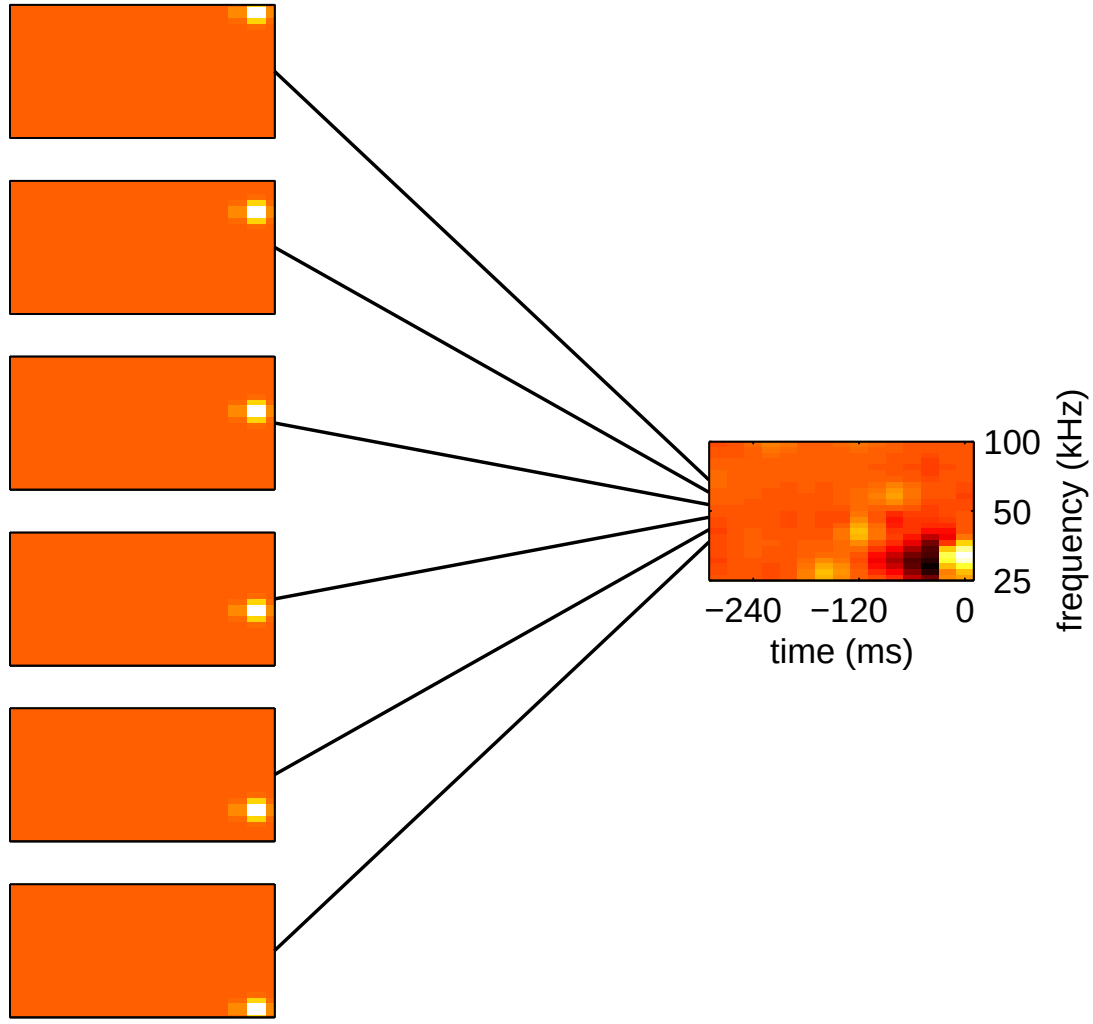
- sparsity [C_{ii} zero for many i] ARD

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

ARD

The most common form of evidence optimization for regression (due to MacKay and Neal) takes $C^{-1} = \text{diag}(\boldsymbol{\alpha})$ (i.e. $w_i \sim \mathcal{N}(0, \alpha_i^{-1})$) and then optimizes the precisions $\{\alpha_i\}$.

Setting the gradients to 0 and solving gives

$$\alpha_i^{\text{new}} = \frac{1 - \alpha_i \Sigma_{ii}}{\mu_i^2}$$

$$(\sigma^2)^{\text{new}} = \frac{(Y - \mu^{\mathsf{T}} X)(Y - \mu^{\mathsf{T}} X)^{\mathsf{T}}}{N - \sum_i (1 - \Sigma_{ii}\alpha_i)}$$

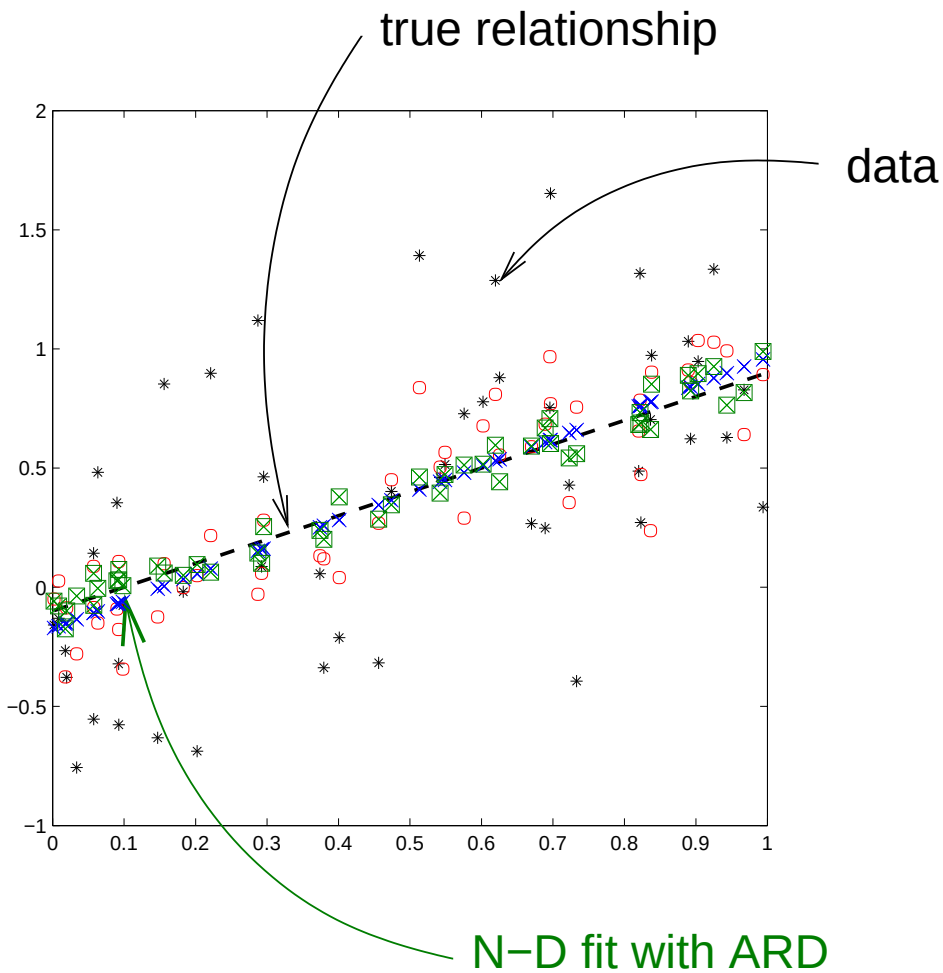
During optimization the α_is meet one of two fates

$\alpha_i \to \infty$	$\Rightarrow$	$w_i = 0$	irrelevant input x_i
α_i finite	$\Rightarrow w_i =$	$\text{argmax}\, \mathrm{P}\,(w_i \mid X, Y, \alpha_i)$	relevant input x_i

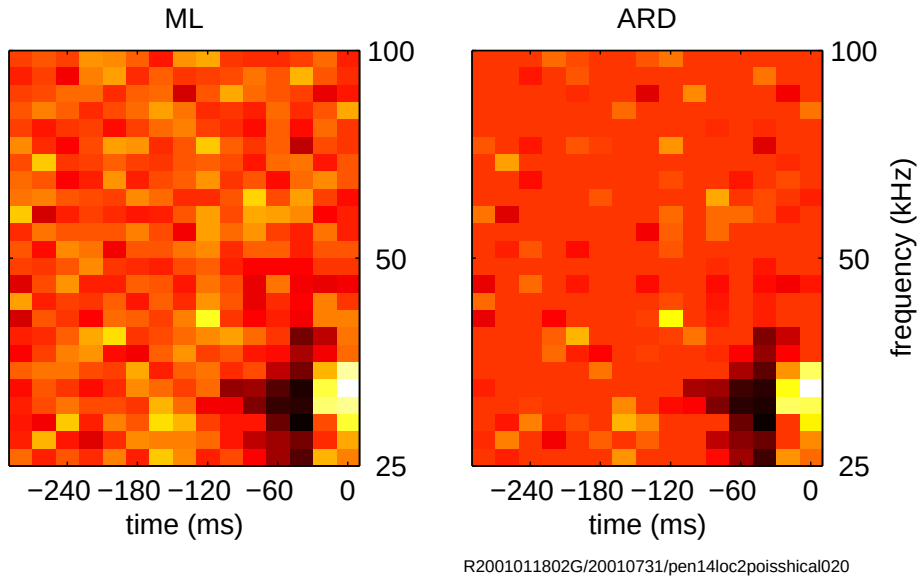
This procedure, Automatic Relevance Determination (ARD), yields sparse solutions that improve on ML regression. (cf. L_1-regression or LASSO).

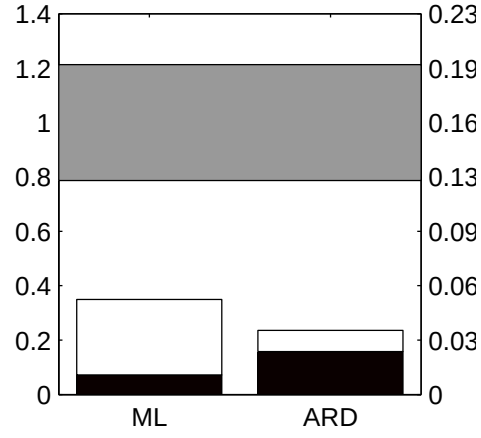
Evidence optimisation is also called maximum marginal likelihood or ML-2 (Type 2 maximum likelihood).

ARD can reduce overfitting

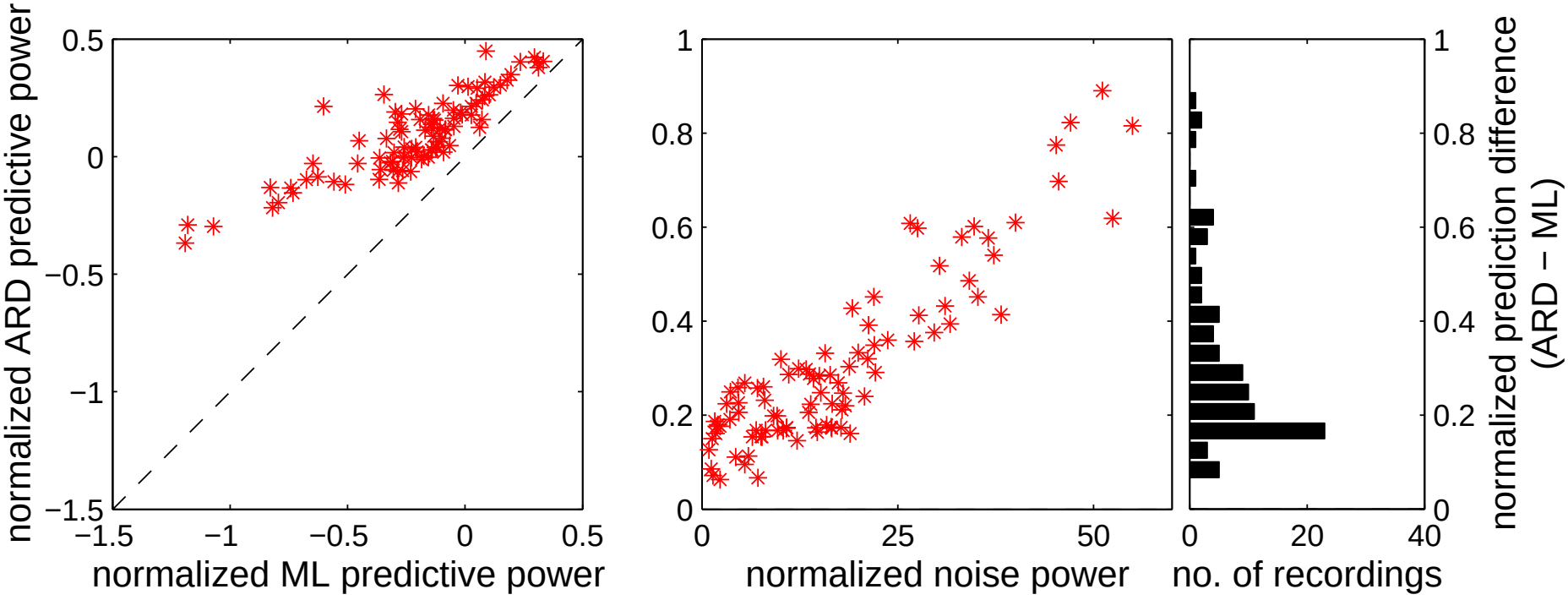


ARD improves STRF predictions





ARD population performance



Smoothness (ASD)

Let:

- Δ_s be matrix of spectral "distances" between weights.
- Δ_t be matrix of temporal "distances" between weights.

Define prior covariance on weights:

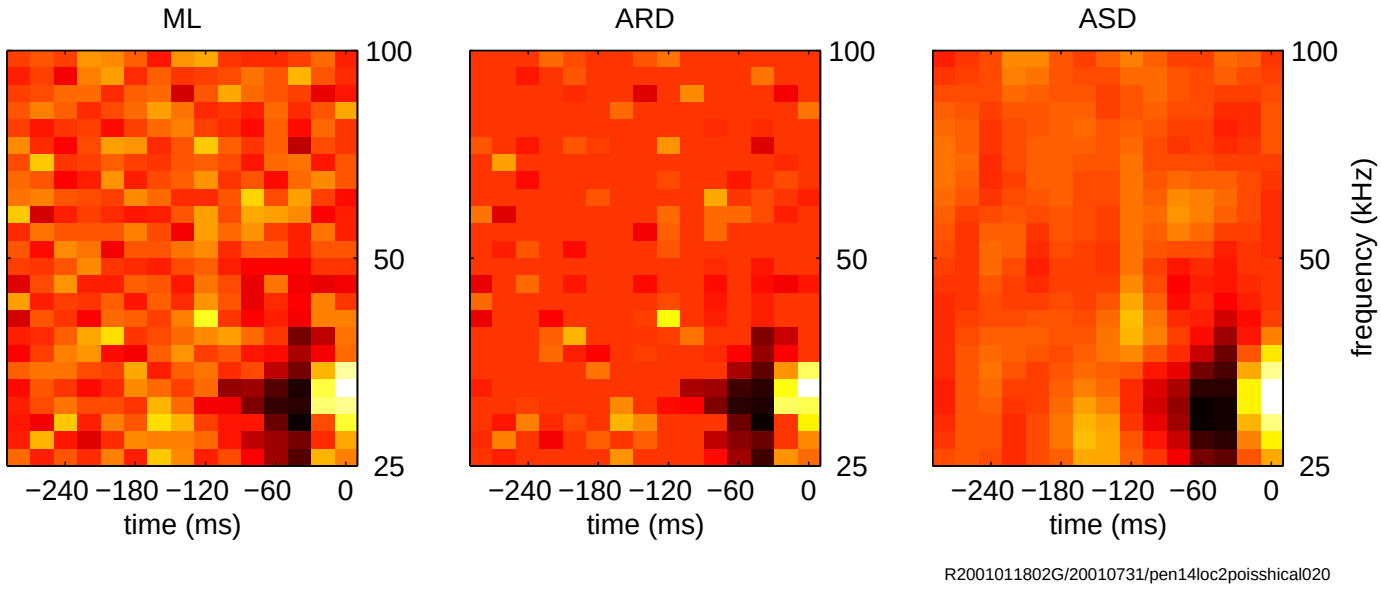
$$C = \exp\left(-\rho - \frac{1}{2}\left(\frac{\Delta_s}{\delta_s^2} + \frac{\Delta_t}{\delta_t^2}\right)\right)$$

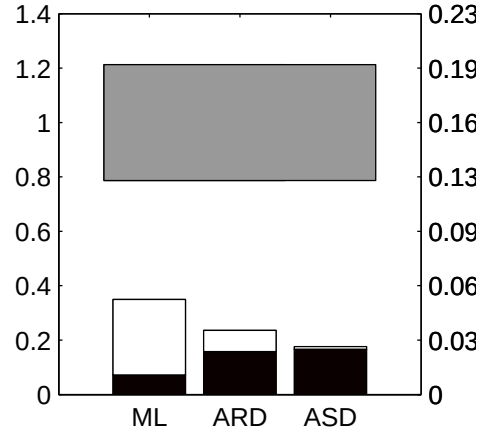
Optimise:

$$\frac{\partial}{\partial \rho} \log L_2 = \frac{1}{2}\mathrm{Tr}\left[(C - \Sigma - \mu\mu^{\mathsf{T}})C^{-1}\right]$$
$$\frac{\partial}{\partial \delta_s} \log L_2 = -\frac{1}{2}\mathrm{Tr}\left[(C - \Sigma - \mu\mu^{\mathsf{T}})C^{-1}(C \circ \frac{\Delta_s}{\delta_s^3})C^{-1}\right]$$

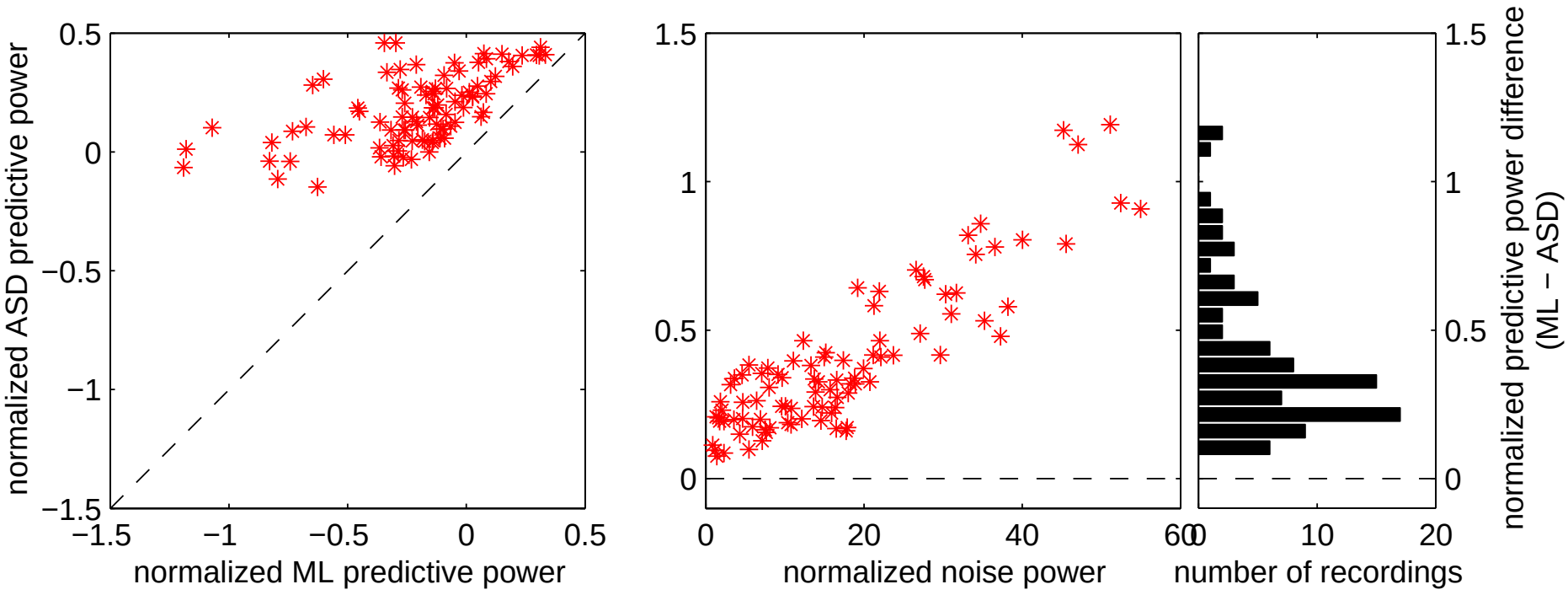
(where μ and Σ are the current posterior mean and variance of weights).

Smoothness (ASD)

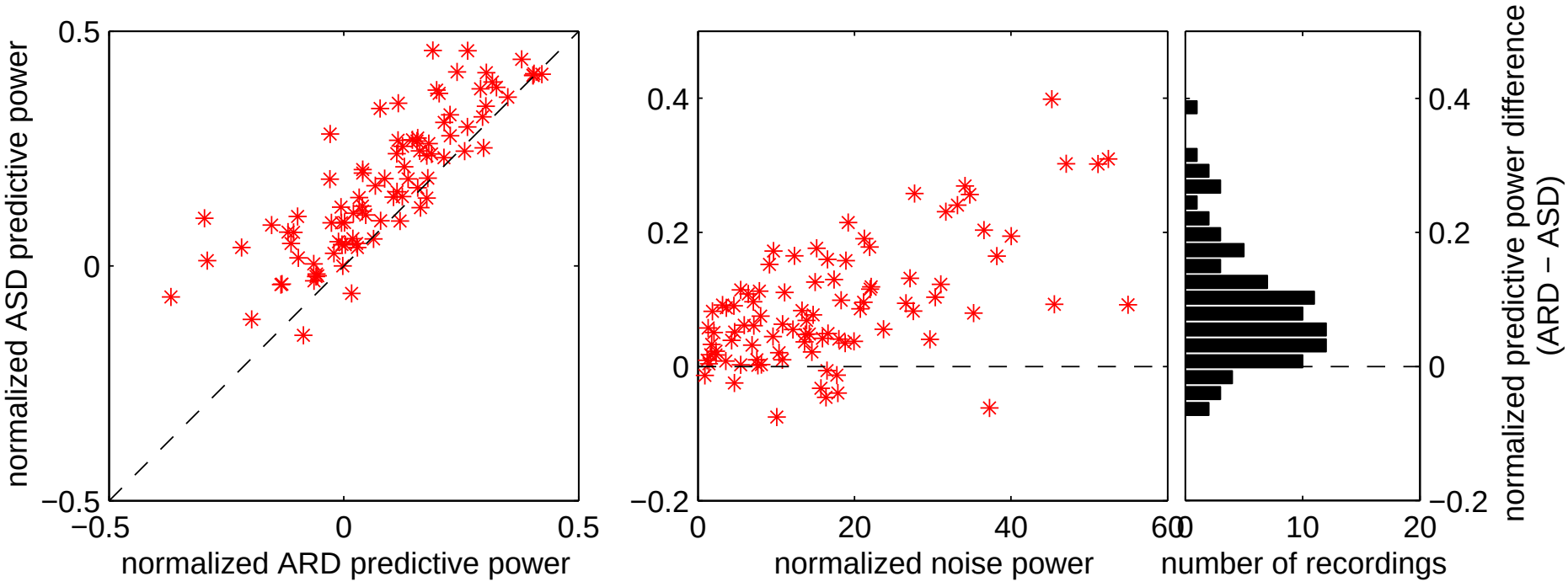




ASD population performance (vs ML)



ASD population performance (vs ARD)



Smoothness and sparsity (ASD/RD)

ASD defines a basis:

$$C_{\text{ASD}} = RR^{\text{T}}$$

Smoothness and sparsity (ASD/RD)

ASD defines a basis:

$$C_{\text{ASD}} = RR^{\mathsf{T}}$$

We can rotate to that basis, such that prior on $\mathbf{w}$ is $\mathcal{N}(0, I)$:

$$Y = \mathbf{w}^{\mathsf{T}} X \qquad \log P(\mathbf{w}) = Z - \frac{1}{2}\mathbf{w}^{\mathsf{T}} C^{-1}\mathbf{w}$$

Smoothness and sparsity (ASD/RD)

ASD defines a basis:

$$C_{\text{ASD}} = RR^{\mathsf{T}}$$

We can rotate to that basis, such that prior on $\mathbf{w}$ is $\mathcal{N}(0, I)$:

$$Y = \mathbf{w}^{\mathsf{T}} {R^{-1}}^{\mathsf{T}} R^{\mathsf{T}} X \qquad \log P(\mathbf{w}) = Z - \frac{1}{2}\mathbf{w}^{\mathsf{T}}(RR^{\mathsf{T}})^{-1}\mathbf{w}$$

Smoothness and sparsity (ASD/RD)

ASD defines a basis:

$$C_{\text{ASD}} = RR^{\mathsf{T}}$$

We can rotate to that basis, such that prior on $\mathbf{w}$ is $\mathcal{N}(0, I)$:

$$Y = \mathbf{w}^{\mathsf{T}} R^{-1^{\mathsf{T}}} R^{\mathsf{T}} X \qquad \log P(\mathbf{w}) = Z - \frac{1}{2} \mathbf{w}^{\mathsf{T}} (RR^{\mathsf{T}})^{-1} \mathbf{w}$$

$$Y = (R^{-1}\mathbf{w})^{\mathsf{T}} (R^{\mathsf{T}} X) \qquad \log P(\mathbf{w}) = Z - \frac{1}{2} (R^{-1}\mathbf{w})^{\mathsf{T}} (R^{-1}\mathbf{w})$$

Smoothness and sparsity (ASD/RD)

ASD defines a basis:

$$C_{\text{ASD}} = RR^{\mathsf{T}}$$

We can rotate to that basis, such that prior on $\mathbf{w}$ is $\mathcal{N}(0, I)$:

$$Y = \mathbf{w}^{\mathsf{T}} R^{-1^{\mathsf{T}}} R^{\mathsf{T}} X \qquad \log P(\mathbf{w}) = Z - \frac{1}{2}\mathbf{w}^{\mathsf{T}}(RR^{\mathsf{T}})^{-1}\mathbf{w}$$

$$Y = \underbrace{(R^{-1}\mathbf{w})}_{\widetilde{\mathbf{w}}}^{\mathsf{T}} \underbrace{(R^{\mathsf{T}} X)}_{\widetilde{X}} \qquad \log P(\mathbf{w}) = Z - \frac{1}{2}\underbrace{(R^{-1}\mathbf{w})}_{\widetilde{\mathbf{w}}}^{\mathsf{T}} \underbrace{(R^{-1}\mathbf{w})}_{\widetilde{\mathbf{w}}}$$

Smoothness and sparsity (ASD/RD)

ASD defines a basis:

$$C_{\text{ASD}} = RR^{\mathsf{T}}$$

We can rotate to that basis, such that prior on $\mathbf{w}$ is $\mathcal{N}(0, I)$:

$$Y = \mathbf{w}^{\mathsf{T}} R^{-1^{\mathsf{T}}} R^{\mathsf{T}} X \qquad \log P(\mathbf{w}) = Z - \frac{1}{2}\mathbf{w}^{\mathsf{T}}(RR^{\mathsf{T}})^{-1}\mathbf{w}$$

$$Y = \underbrace{(R^{-1}\mathbf{w})}_{\widetilde{\mathbf{w}}}^{\mathsf{T}} \underbrace{(R^{\mathsf{T}} X)}_{\widetilde{X}} \qquad \log P(\mathbf{w}) = Z - \frac{1}{2}\underbrace{(R^{-1}\mathbf{w})}_{\widetilde{\mathbf{w}}}^{\mathsf{T}} \underbrace{(R^{-1}\mathbf{w})}_{\widetilde{\mathbf{w}}}$$

$$Y = \widetilde{\mathbf{w}}^{\mathsf{T}} \widetilde{X} \qquad \log P(\mathbf{w}) = Z - \frac{1}{2}\widetilde{\mathbf{w}}^{\mathsf{T}}\widetilde{\mathbf{w}}$$

Smoothness and sparsity (ASD/RD)

ASD defines a basis:

$$C_{\text{ASD}} = RR^{\mathsf{T}}$$

We can rotate to that basis, such that prior on $\mathbf{w}$ is $\mathcal{N}(0, I)$:

$$Y = \mathbf{w}^{\mathsf{T}} R^{-1^{\mathsf{T}}} R^{\mathsf{T}} X \qquad \log P(\mathbf{w}) = Z - \frac{1}{2}\mathbf{w}^{\mathsf{T}}(RR^{\mathsf{T}})^{-1}\mathbf{w}$$

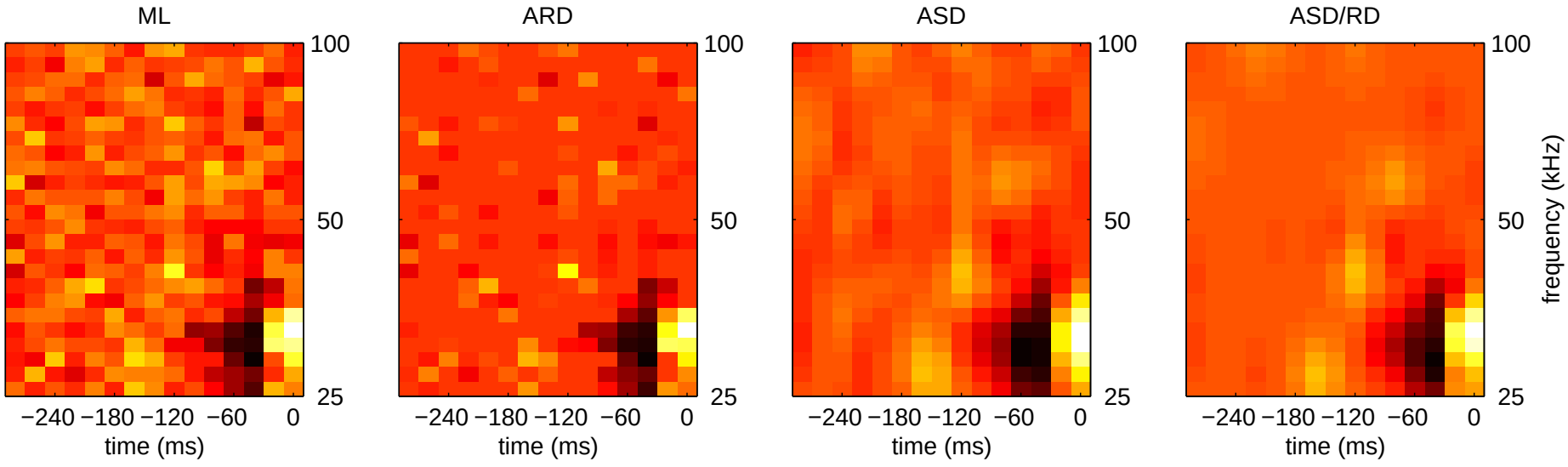
$$Y = \underbrace{(R^{-1}\mathbf{w})}_{\widetilde{\mathbf{w}}}^{\mathsf{T}} \underbrace{(R^{\mathsf{T}} X)}_{\widetilde{X}} \qquad \log P(\mathbf{w}) = Z - \frac{1}{2}\underbrace{(R^{-1}\mathbf{w})}_{\widetilde{\mathbf{w}}}^{\mathsf{T}} \underbrace{(R^{-1}\mathbf{w})}_{\widetilde{\mathbf{w}}}$$

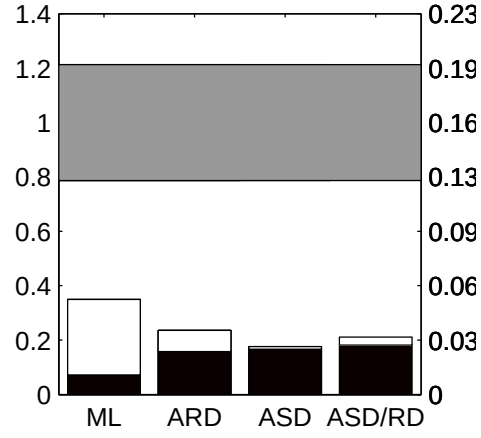
$$Y = \widetilde{\mathbf{w}}^{\mathsf{T}} \widetilde{X} \qquad \log P(\mathbf{w}) = Z - \frac{1}{2}\widetilde{\mathbf{w}}^{\mathsf{T}}\widetilde{\mathbf{w}}$$

Now, add a new *diagonal* prior A and perform ARD in the rotated space.

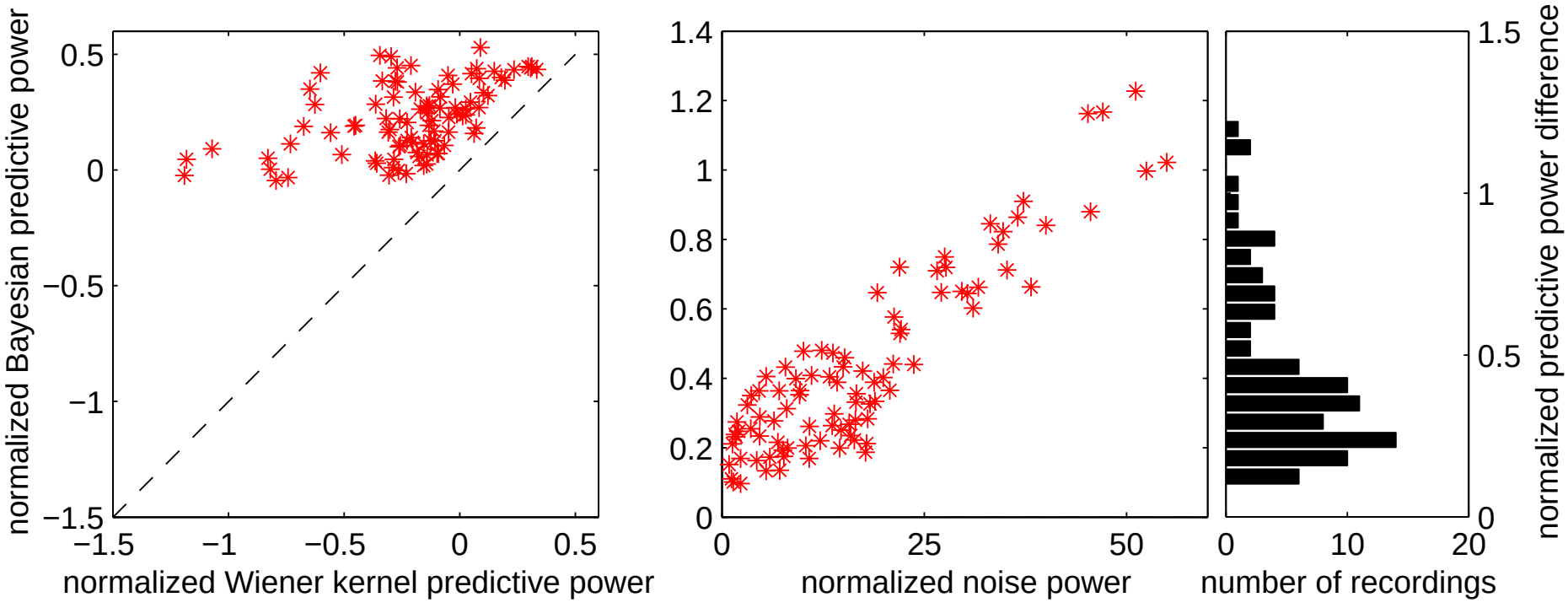
This enforces sparsity in the basis defined by ASD.

Smoothness and sparsity (ASD/RD)





Performance



How good are linear predictions?

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

How good are linear predictions?

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.

How good are linear predictions?

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.

How good are linear predictions?

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.

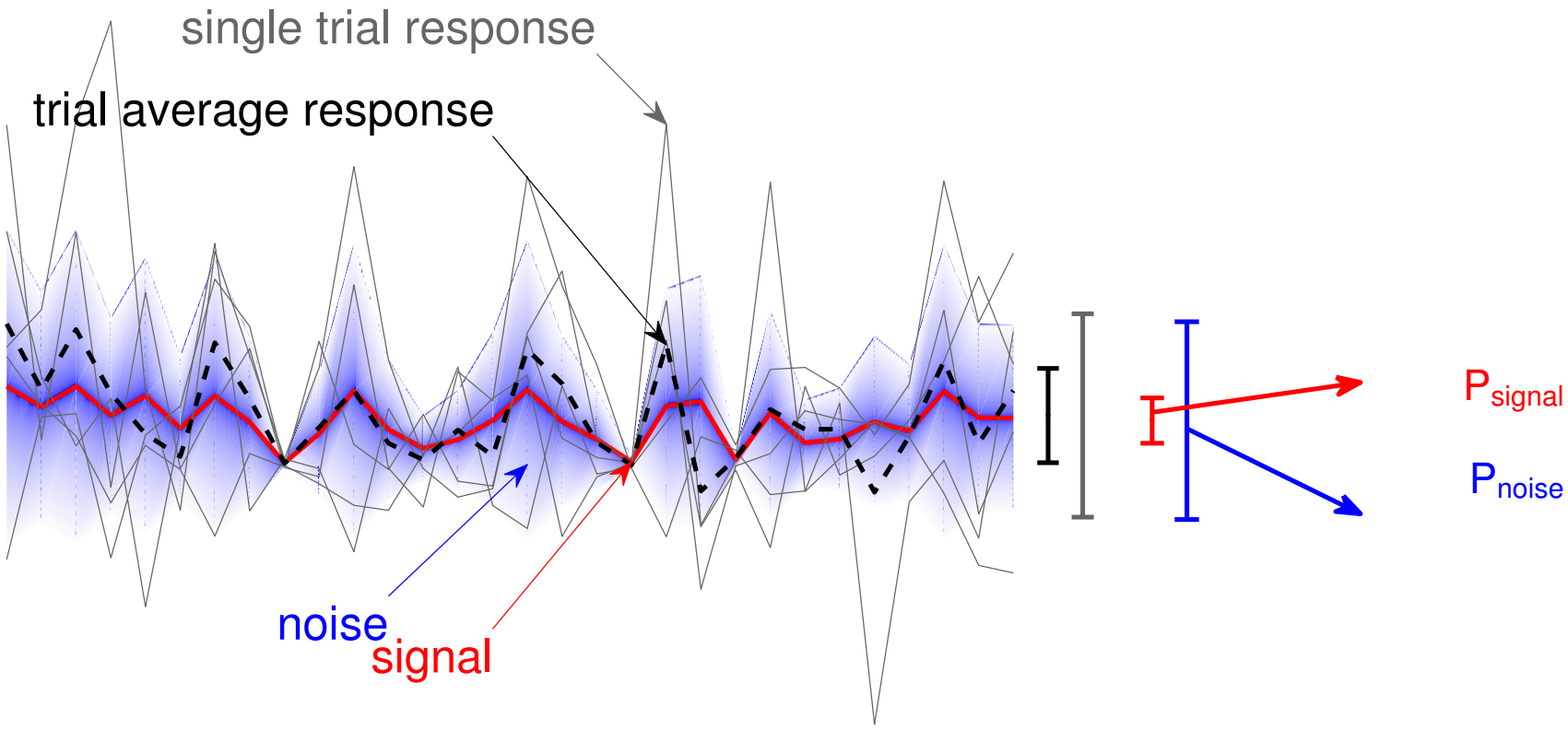
Even if we can discount this, a model may predict poorly because either:

- 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
- 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 *predicatable power* (similar to ANOVA).

Estimating predictable power



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

$$\left.\begin{aligned} \overline{P(\mathbf{r}^{(n)})} &= P_{signal} + P_{noise} \\ P(\overline{\mathbf{r}^{(n)}}) &= P_{signal} + \frac{1}{N} P_{noise} \end{aligned}\right\} \Rightarrow \begin{cases} \widehat{P}_{signal} = \dfrac{1}{N-1}\left(NP(\overline{\mathbf{r}^{(n)}}) - \overline{P(\mathbf{r}^{(n)})}\right) \\ \widehat{P}_{noise} = \overline{P(\mathbf{r}^{(n)})} - \widehat{P}_{signal} \end{cases}$$

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

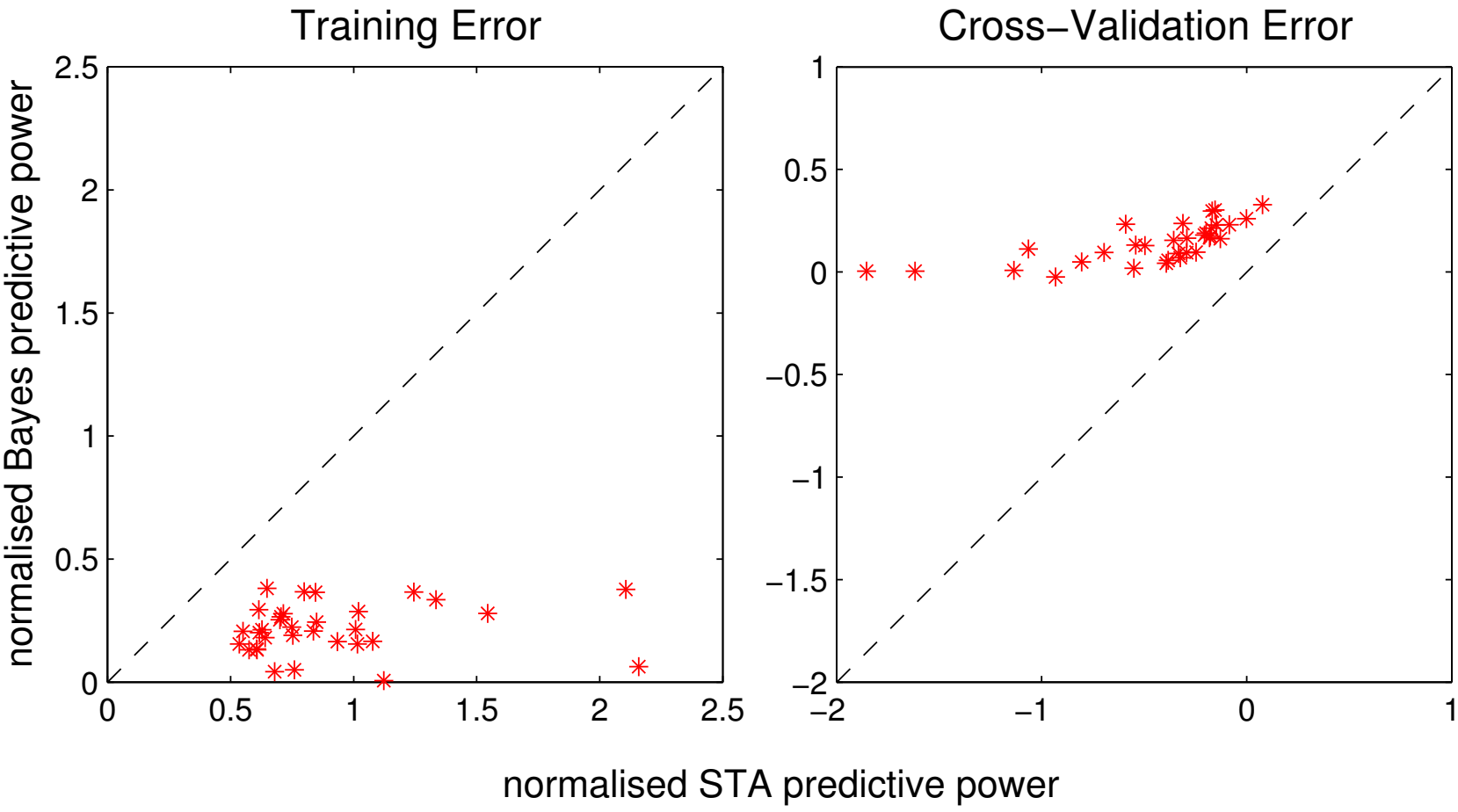
$$\left\langle \mathrm{P}(\overline{\mathrm{trial}}) - \mathrm{P}(\mathrm{residual}) \right\rangle = \mathrm{P}(\mathrm{signal})$$

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

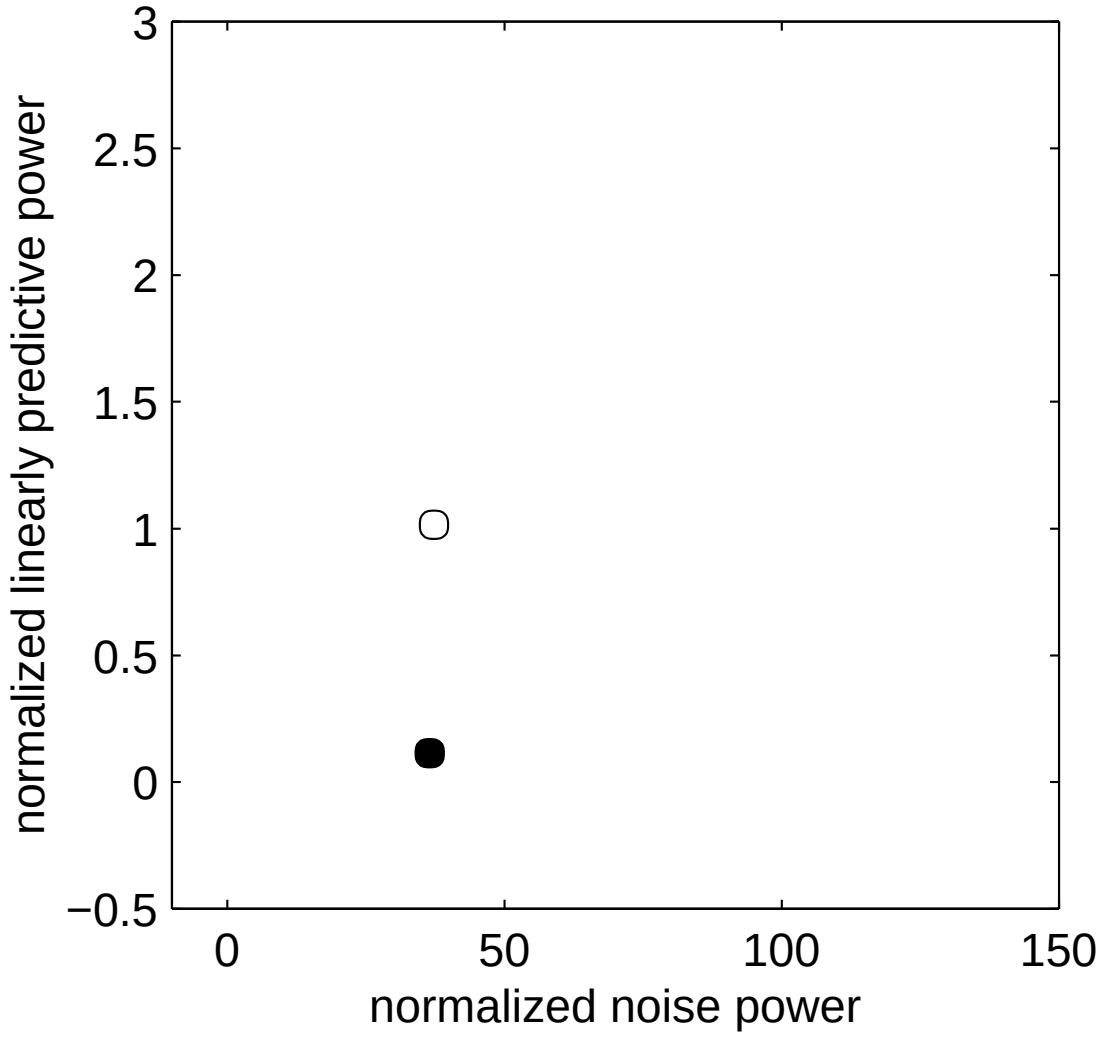
$$\frac{\mathrm{P}(\overline{\mathrm{trial}}) - \mathrm{P}(\mathrm{residual})}{\widehat{\mathrm{P}}(\mathrm{signal})}$$

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

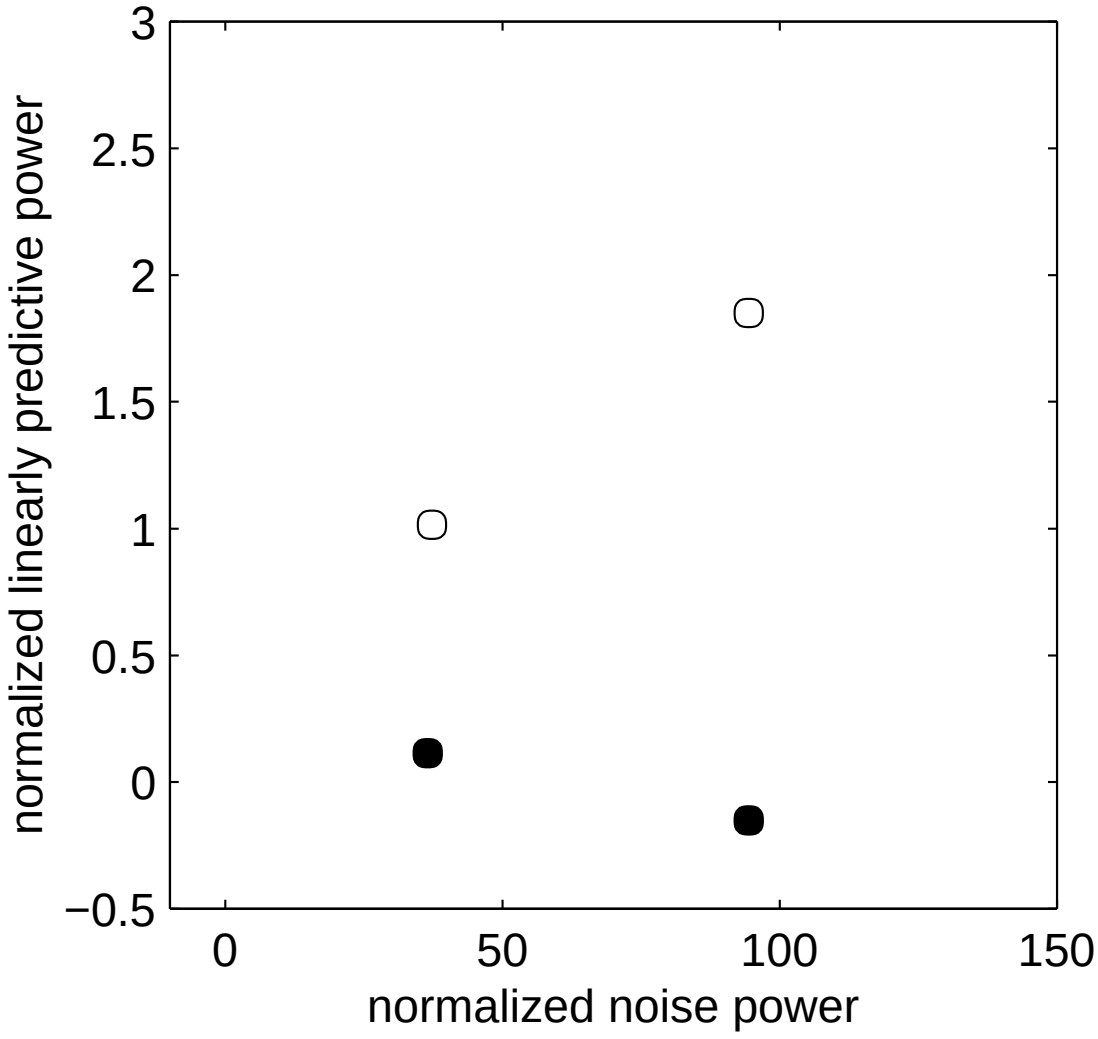
Predictive performance



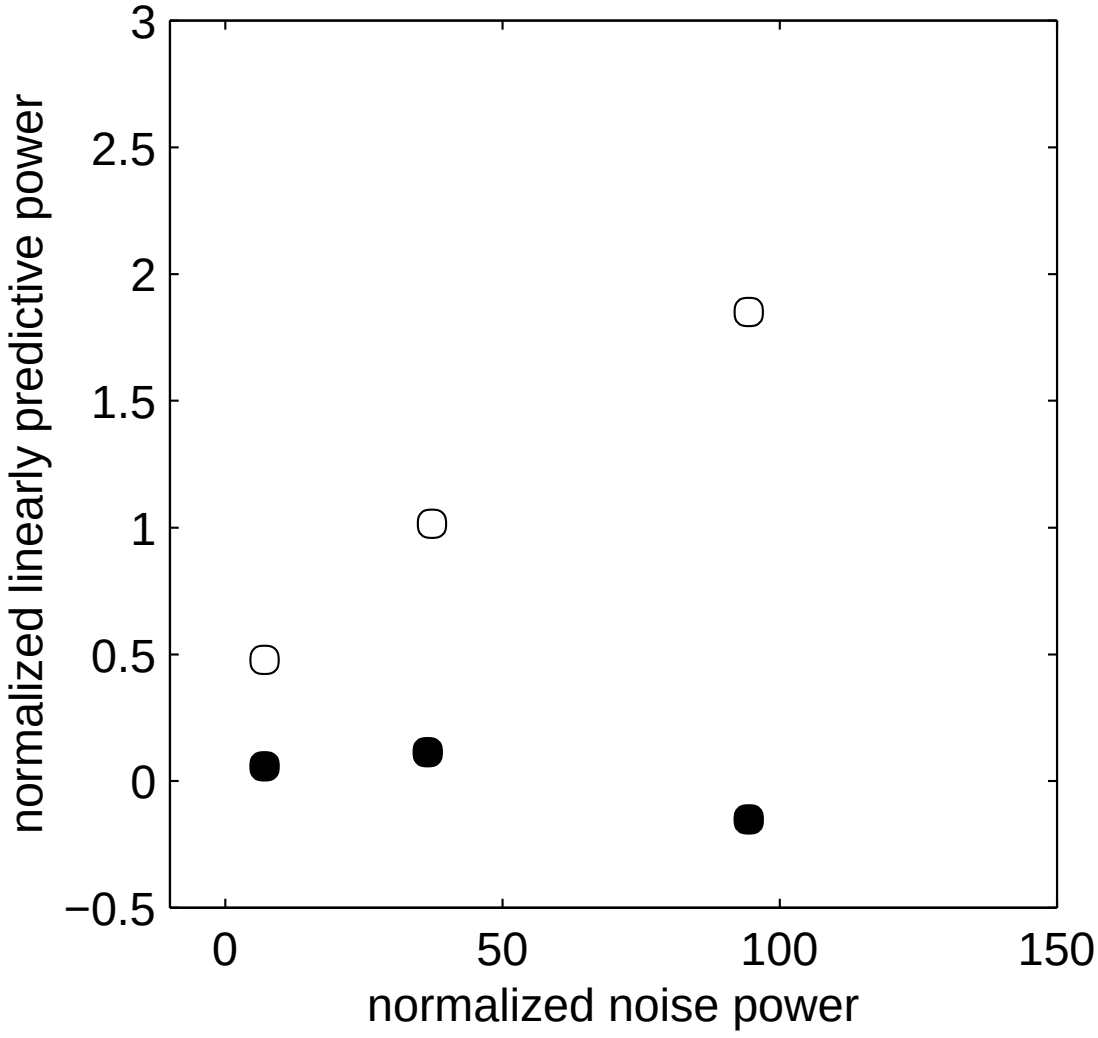
Extrapolating the model performance



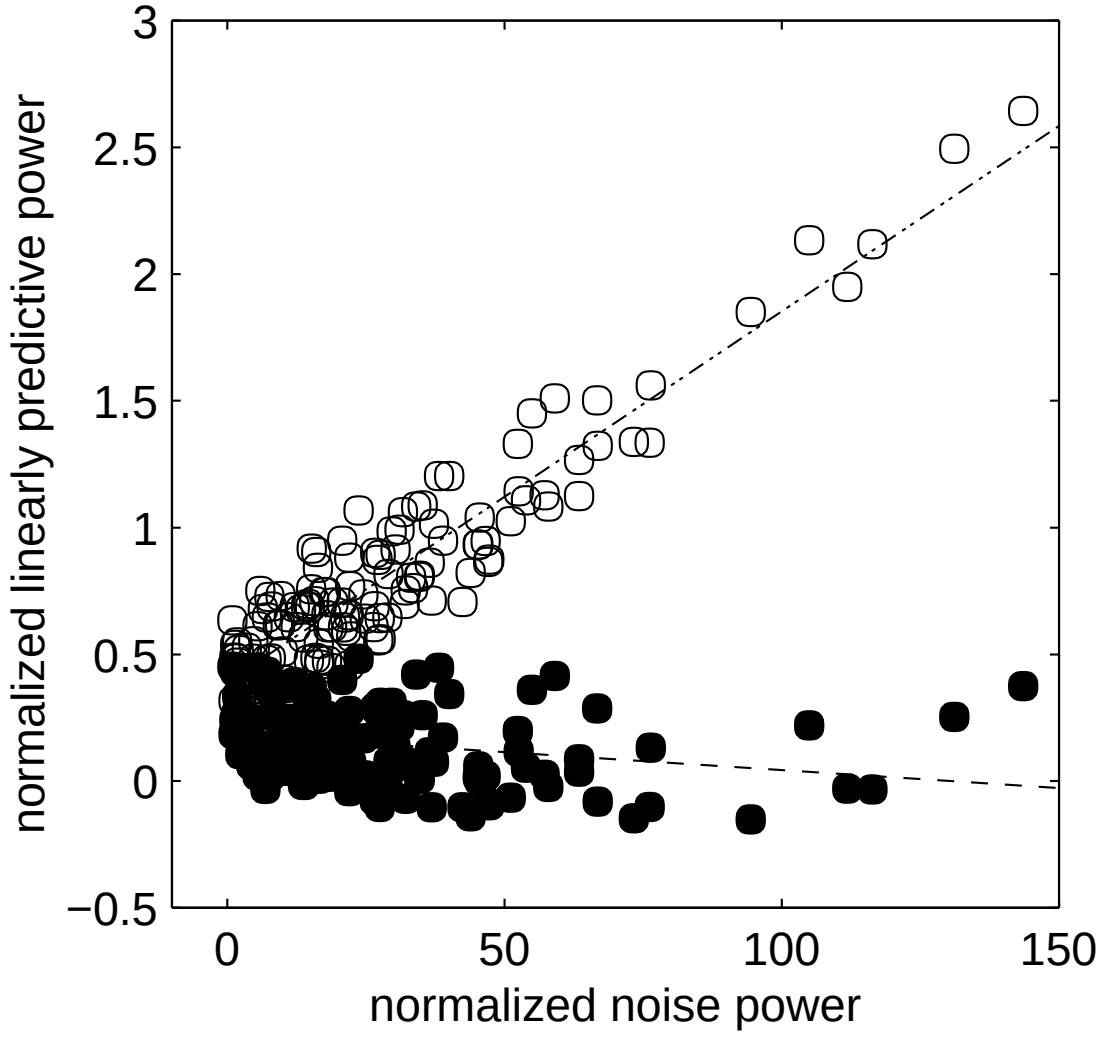
Extrapolating the model performance



Extrapolating the model performance



Extrapolating the model performance



Jacknife bias correction

Estimate bias by extrapolation in data size:

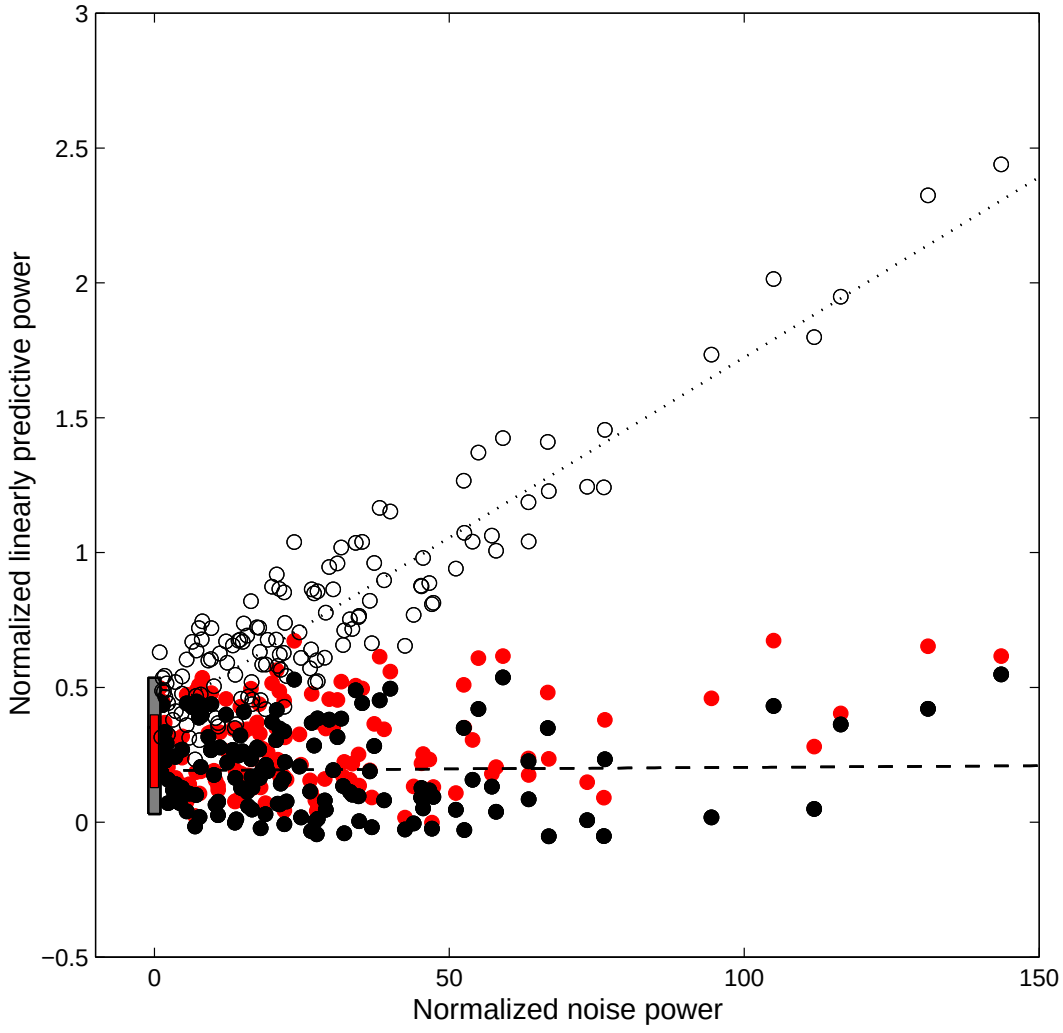
$$\mathcal{T}_{\text{jn}} = N\mathcal{T} - (N-1)\mathcal{T}_{\text{loo}}$$

where $\mathcal{T}$ is the training error on all data and $\mathcal{T}_{\text{loo}}$ is the average training error on all sets of $N-1$ data.

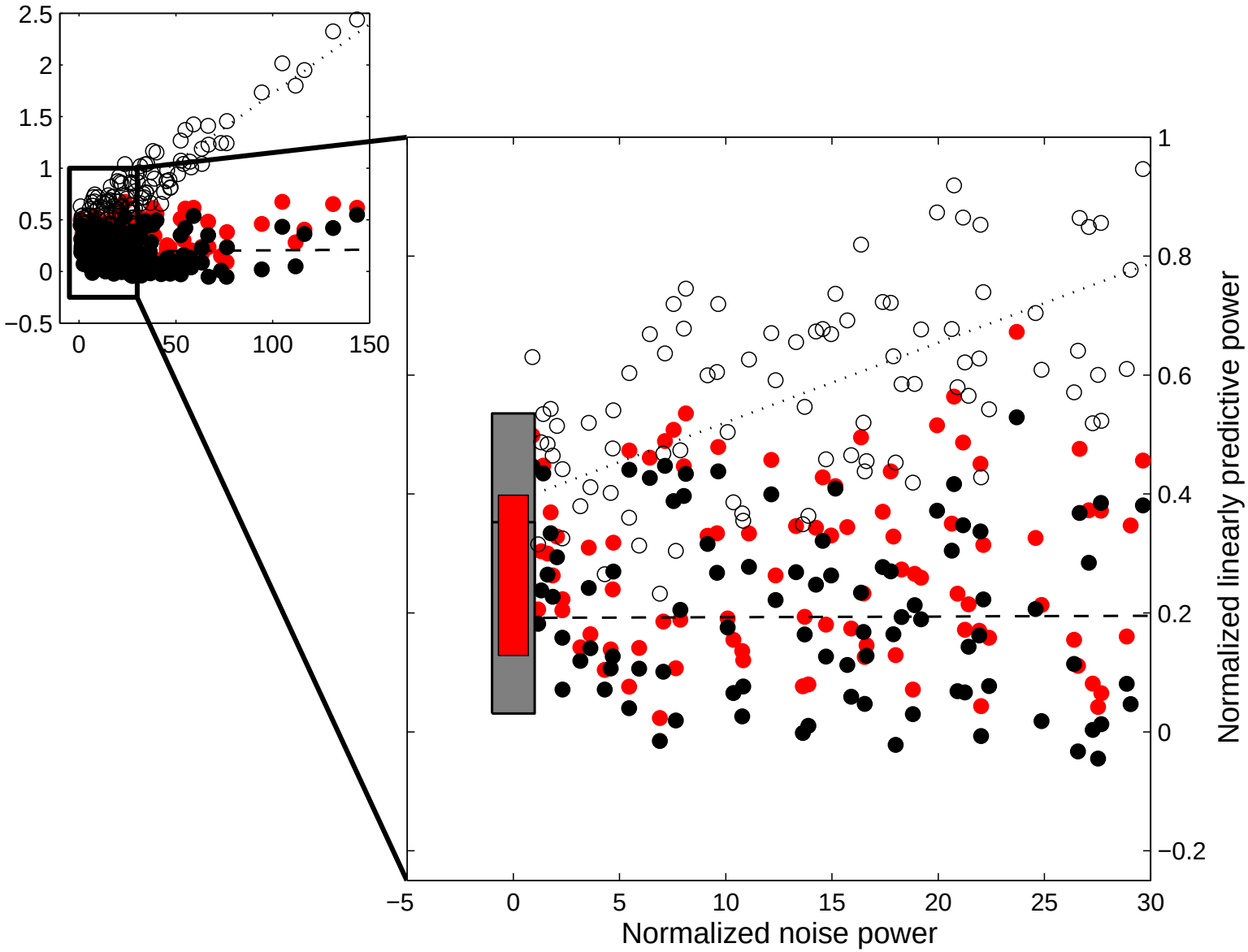
For a linear model we can find this in closed form:

$$\mathcal{T}_{\text{jn}} = \frac{1}{N}\sum_i \left(\frac{(r_i - \mathbf{s}_i \mathbf{w}^{\text{ML}})^2}{1 - \mathbf{s}_i (S^{\text{T}} S)^{-1} \mathbf{s}_i^{\text{T}}} \right)$$

Jackknifed estimates

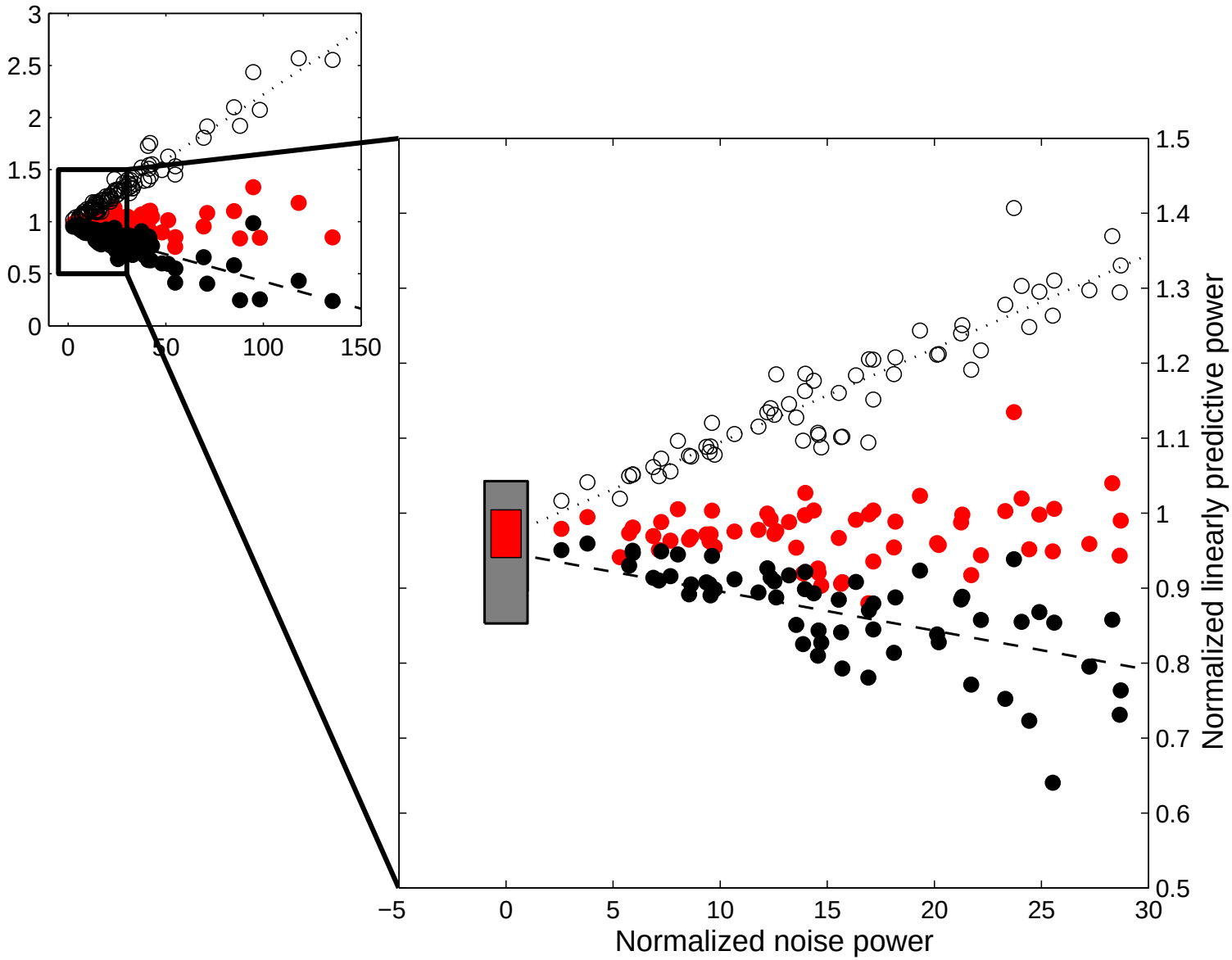


Extrapolated linearity



[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]

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

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* $x(t)$.

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

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

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

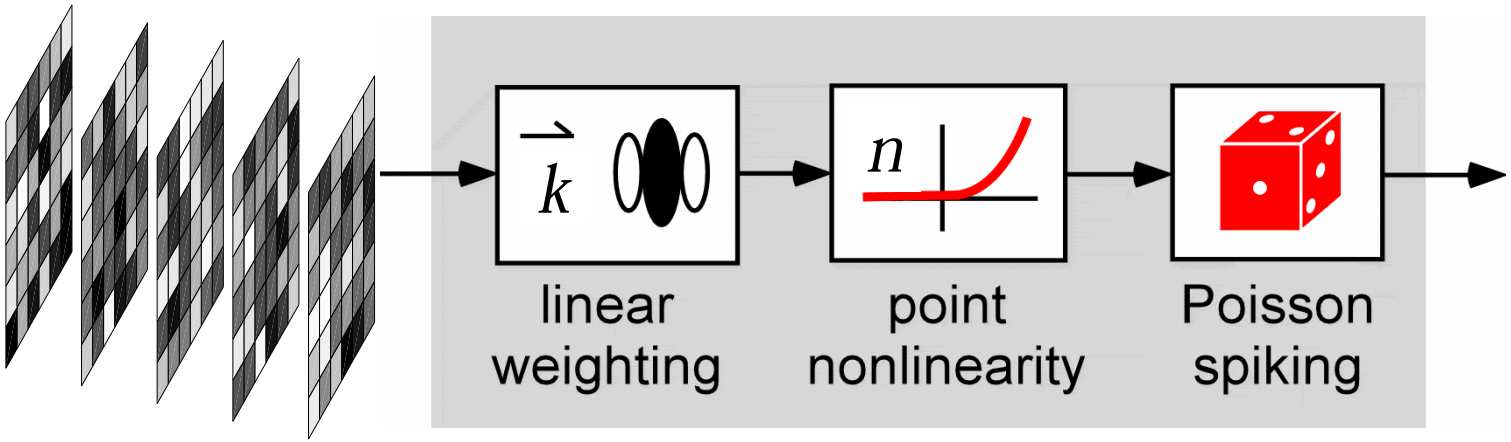
$$G_3[x(t); h^{(3)}] = \iiint d\tau_1\, d\tau_2\, d\tau_3\, h^{(3)}(\tau_1, \tau_2, \tau_3) x(t-\tau_1) x(t-\tau_2) x(t-\tau_3)$$
$$- 3P \iint d\tau_1\, d\tau_2\, h^{(3)}(\tau_1, \tau_2, \tau_2) x(t-\tau_1)$$

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.

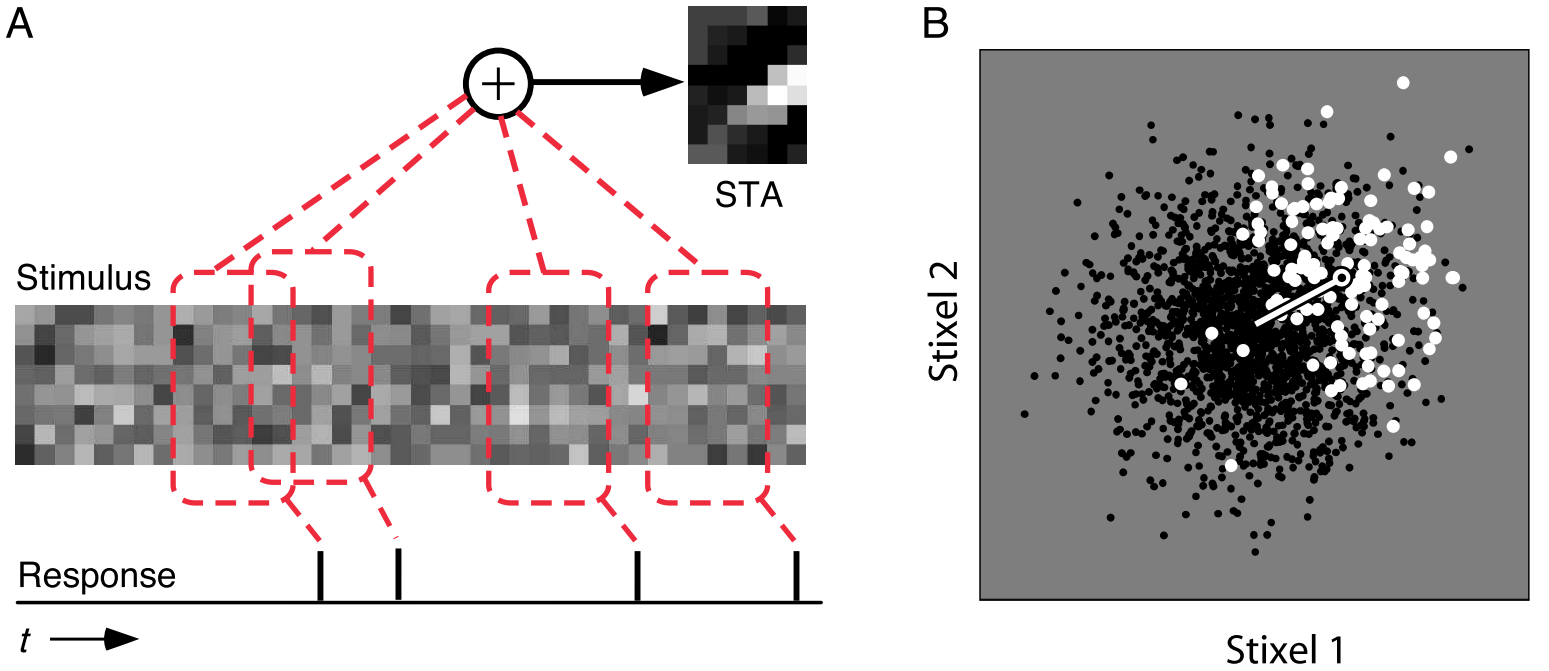
Cascade models

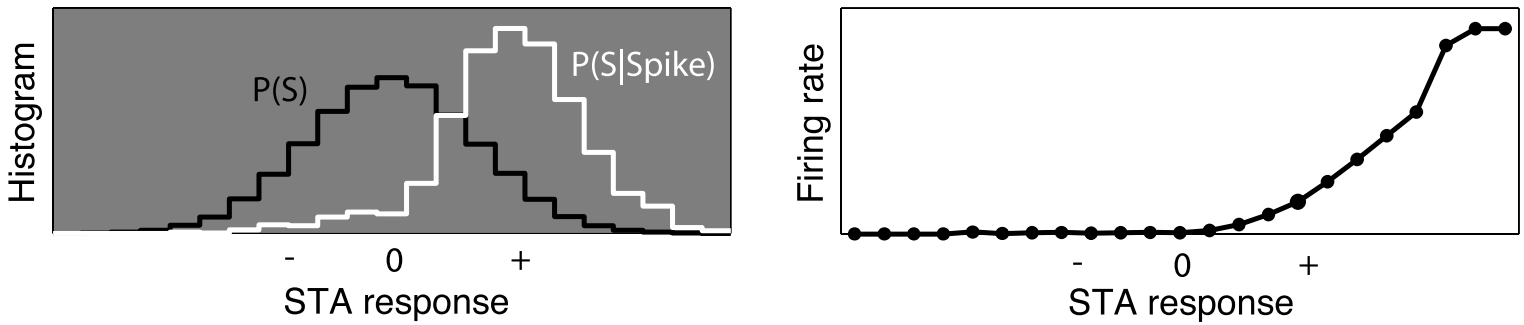
The LNP (Wiener) cascade



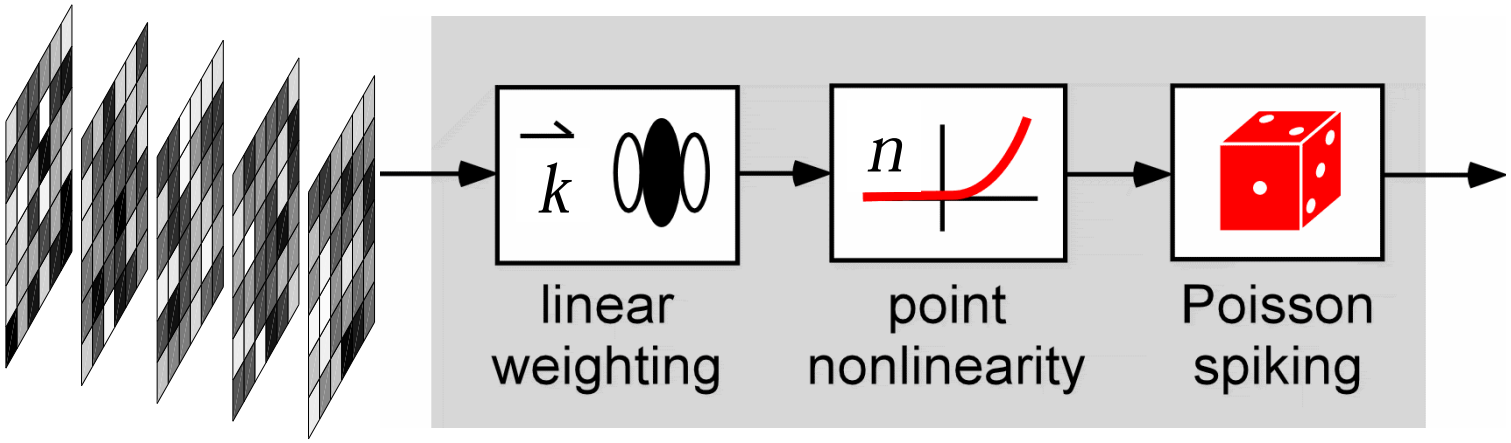
- Rectification addresses negative firing rates.
- Loose biophysical correspondance.

LNP estimation – the Spike-triggered ensemble



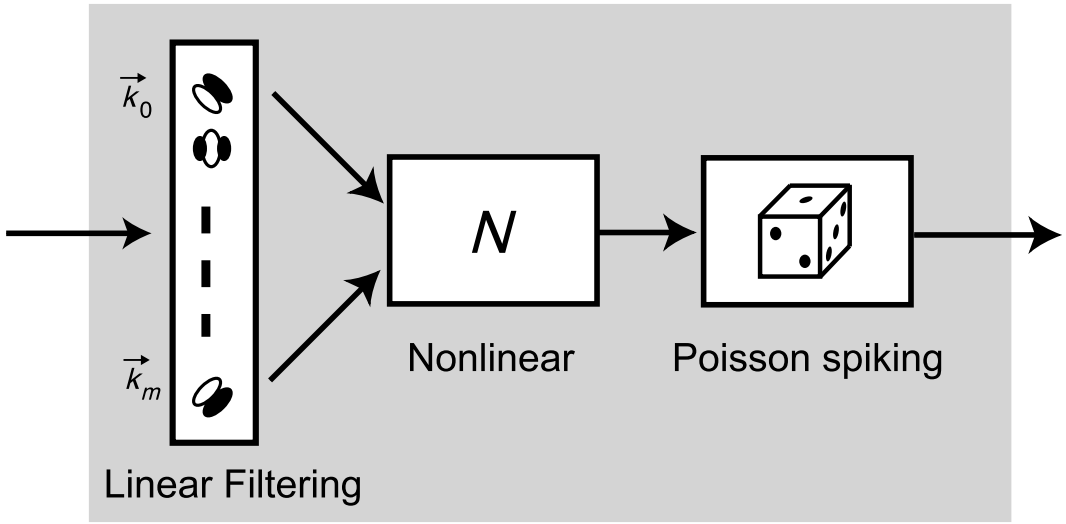


Single linear filter



- 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

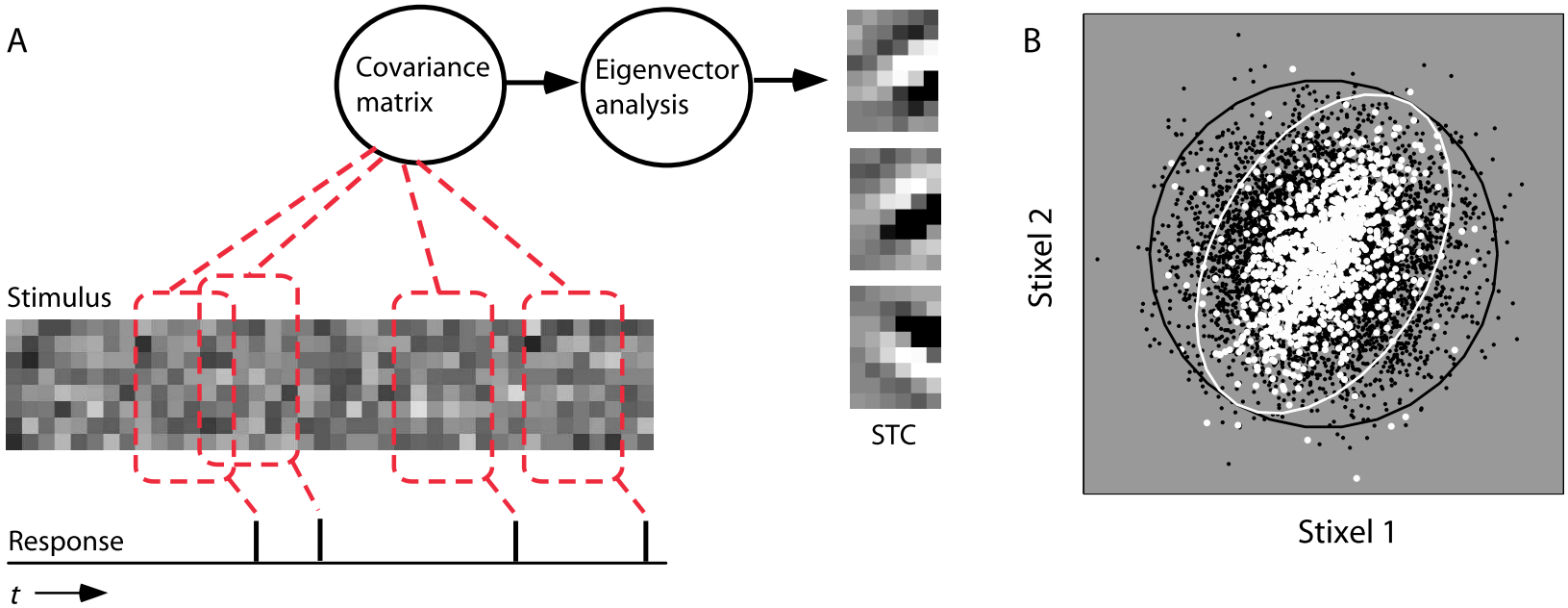


Distribution changes along relevant directions (and, usually, along all linear combinations of relevant directions).

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:

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

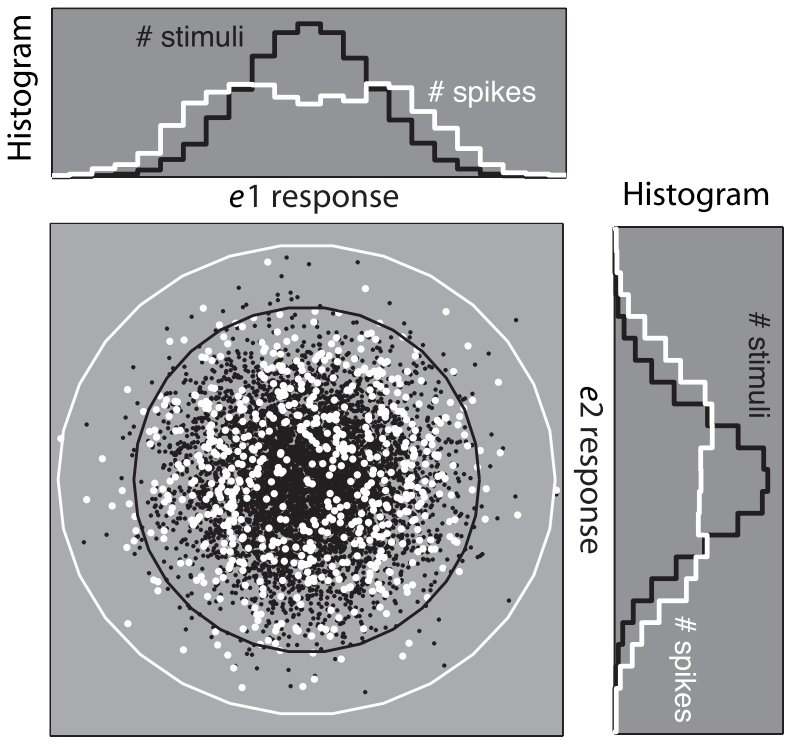
Choose directions with greatest change in variance:

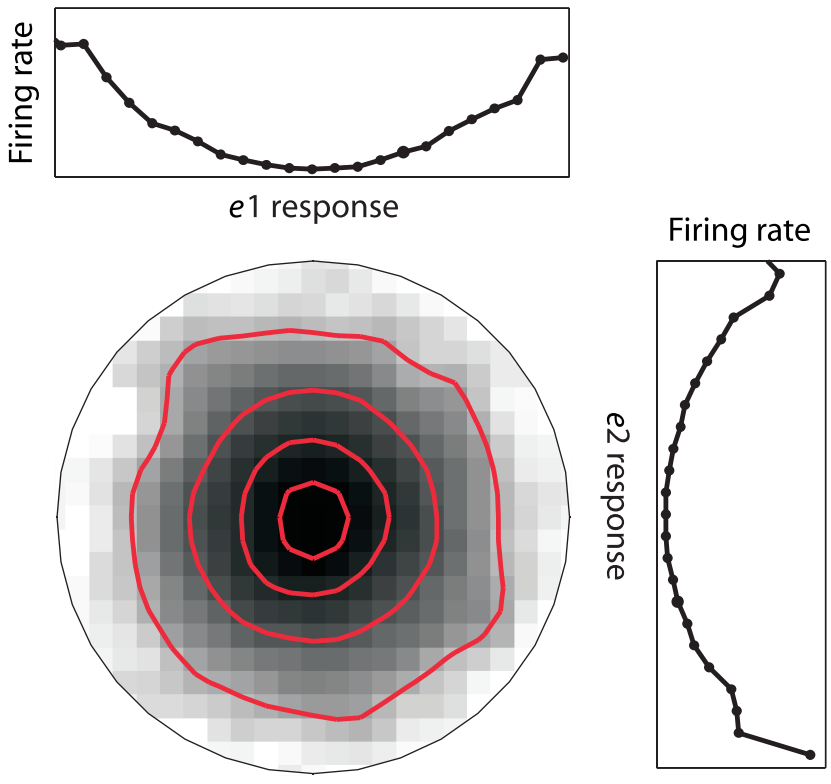
$$\underset{\|\mathbf{v}\|=1}{\text{k-argmax}}\, \mathbf{v}^{\mathsf{T}}(C_{\text{prior}} - C_{\text{spike}})\mathbf{v}$$

$\Rightarrow$ 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

$\Rightarrow$ 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.

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

  - Measuring KL requires binning or smoothing—turns out to be equivalent to fitting a non-parametric nonlinearity by binning or smoothing.
  - Difficult to use for high-dimensional LNP models (but ML viewpoint suggests separable or "cylindrical" basis functions).

- 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* $= \beta\mathbf{x}$).

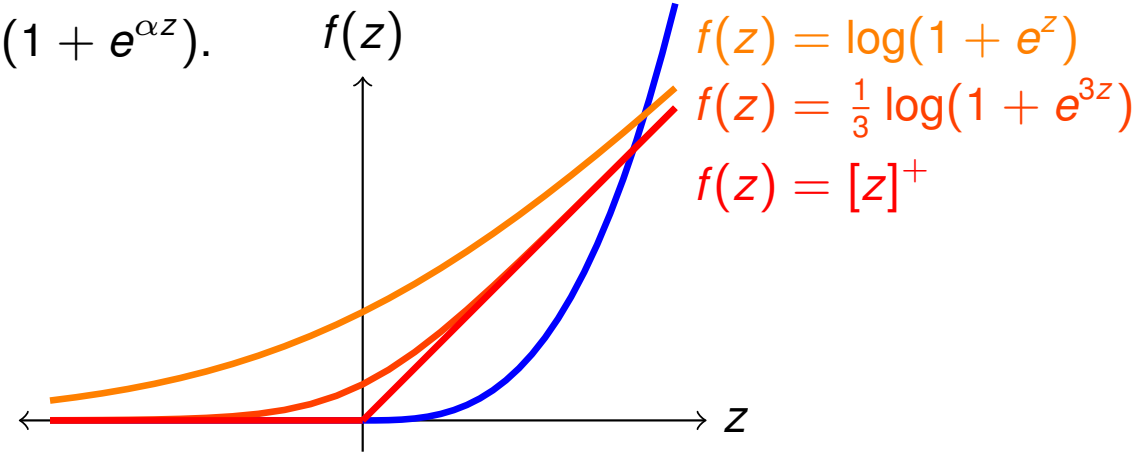
Canonical link functions give concave likelihoods $\Rightarrow$ unique maxima.

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



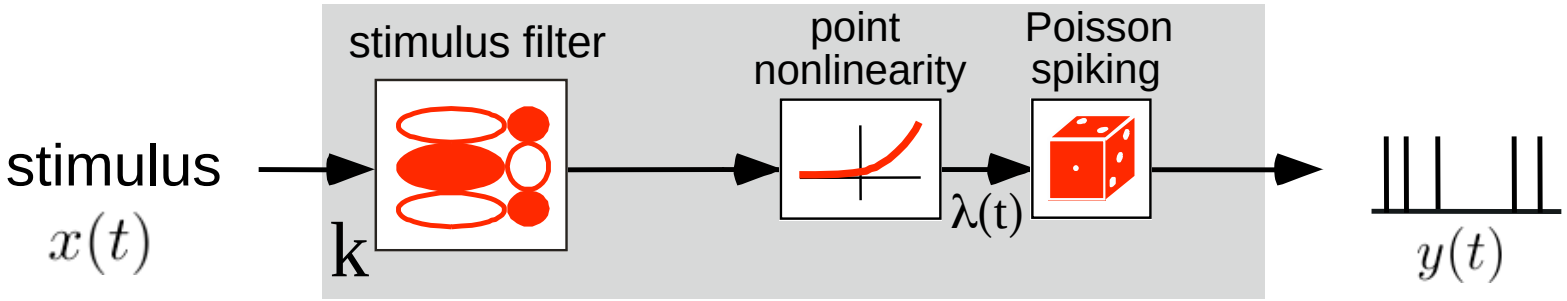
Generalised linear models

ML parameters found by

- gradient ascent
- IRLS

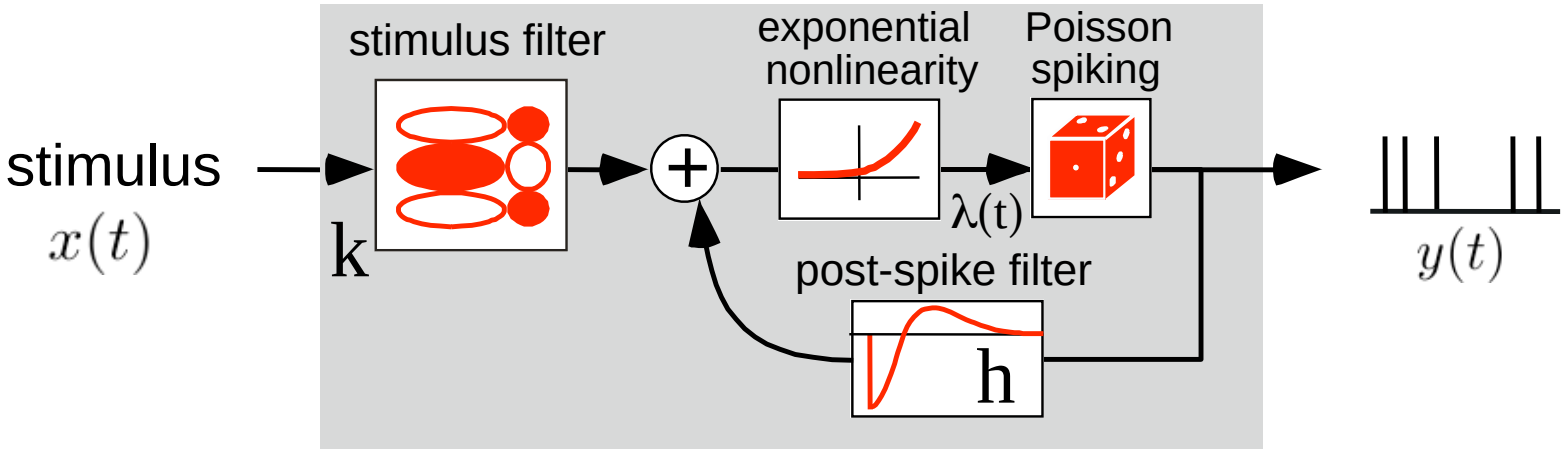
Regularisation by L_2 (quadratic) or L_1 (absolute value – sparse) penalties (MAP with Gaussian/Laplacian priors) preserves concavity.

Linear-Nonlinear-Poisson (GLM)



GLM with history-dependence

(Truccolo et al 04)



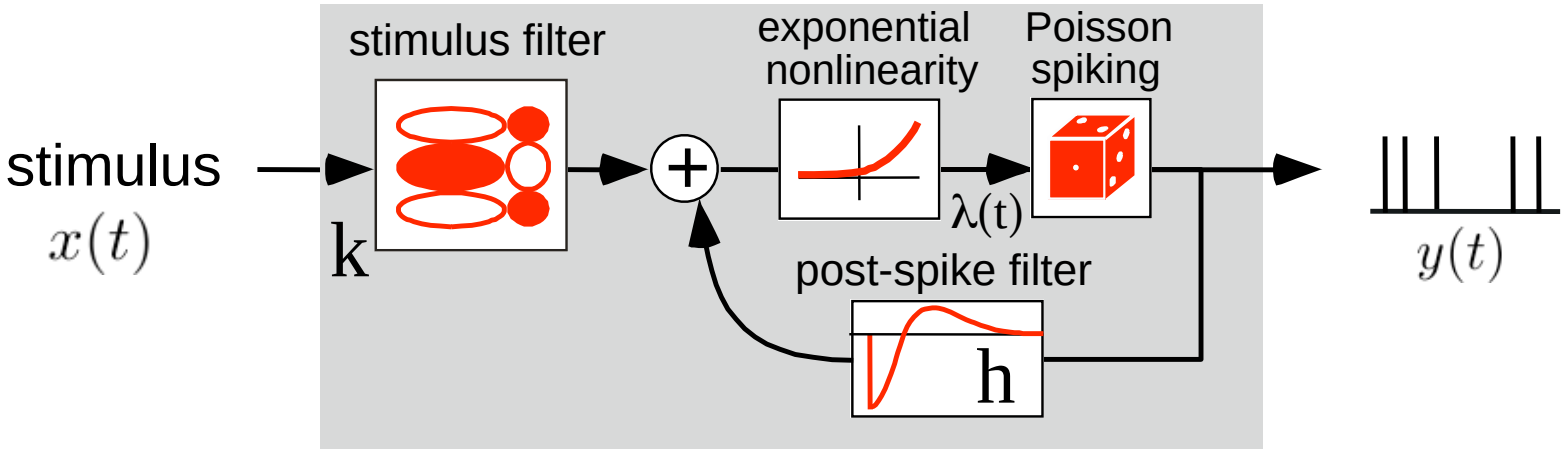
conditional intensity (spike rate)

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

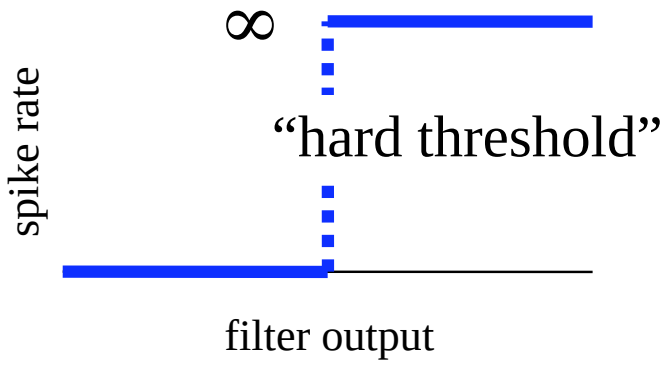
$$= e^{k \cdot x(t)} \cdot e^{h \cdot y(t)}$$

- 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

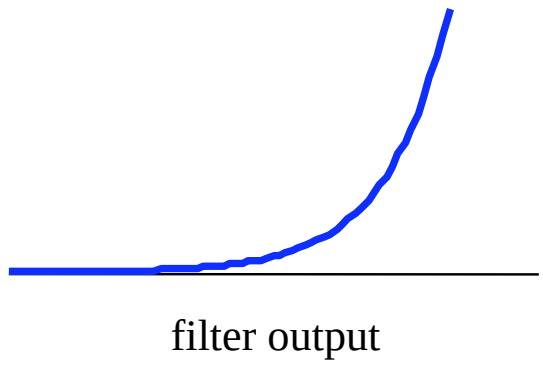
GLM with history-dependence



traditional IF

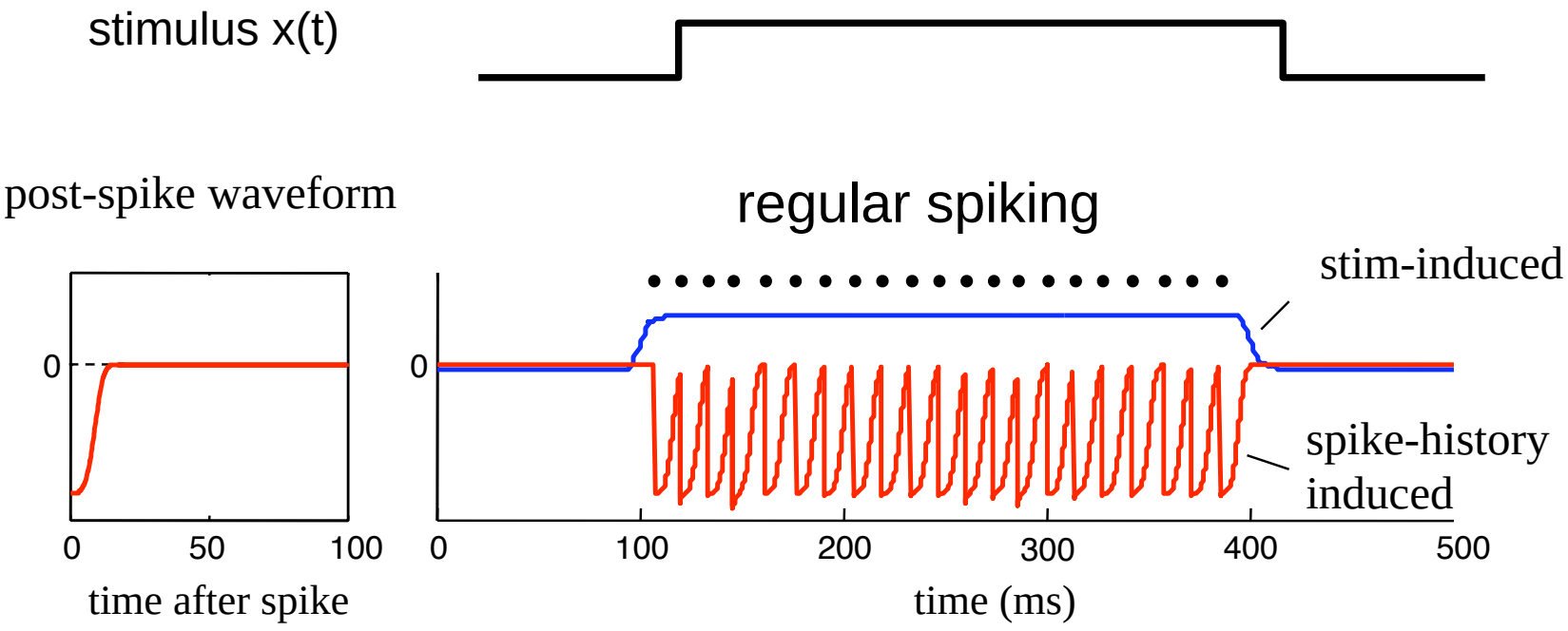


"soft-threshold" IF

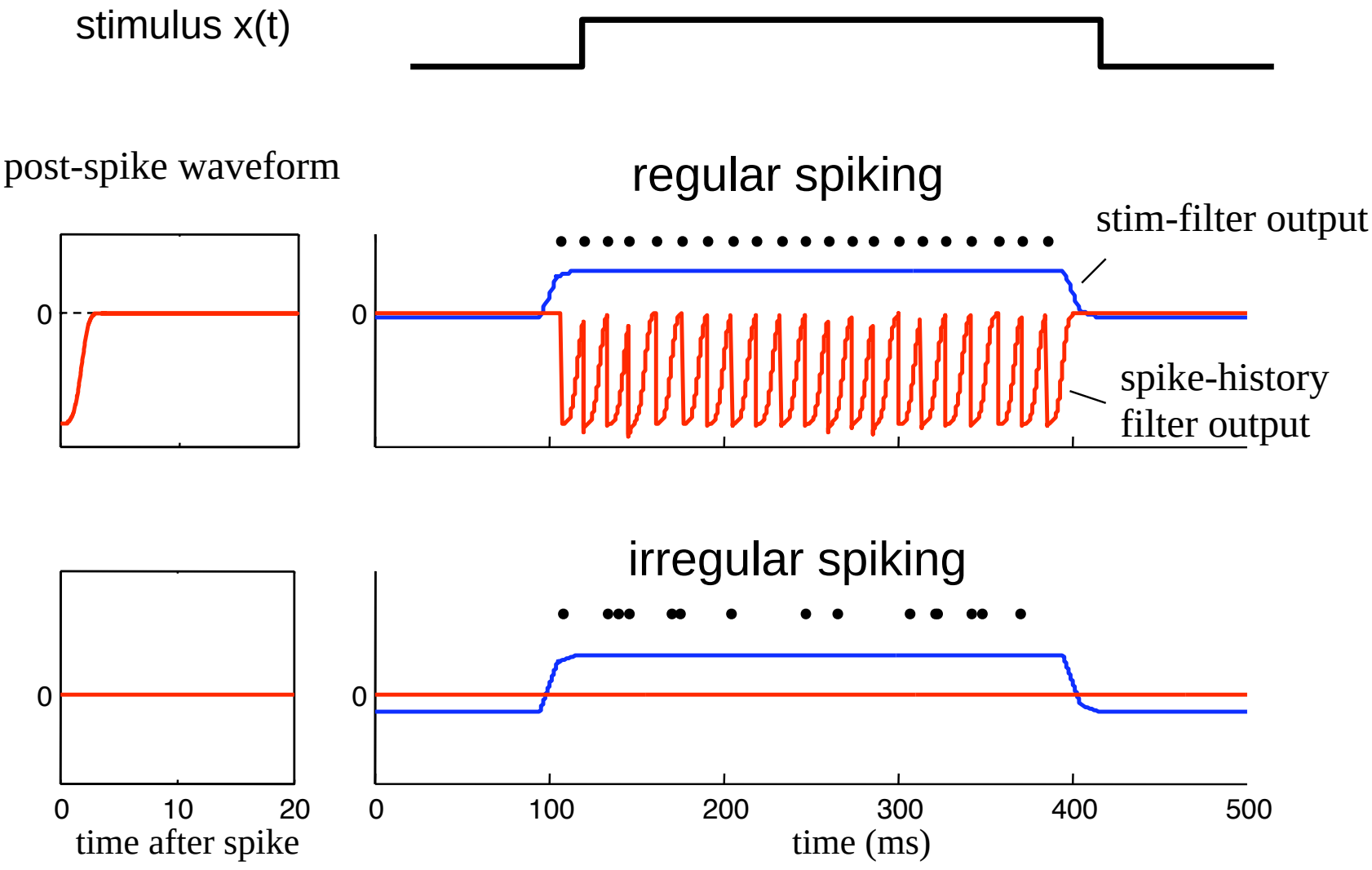


- "soft-threshold" approximation to Integrate-and-Fire model

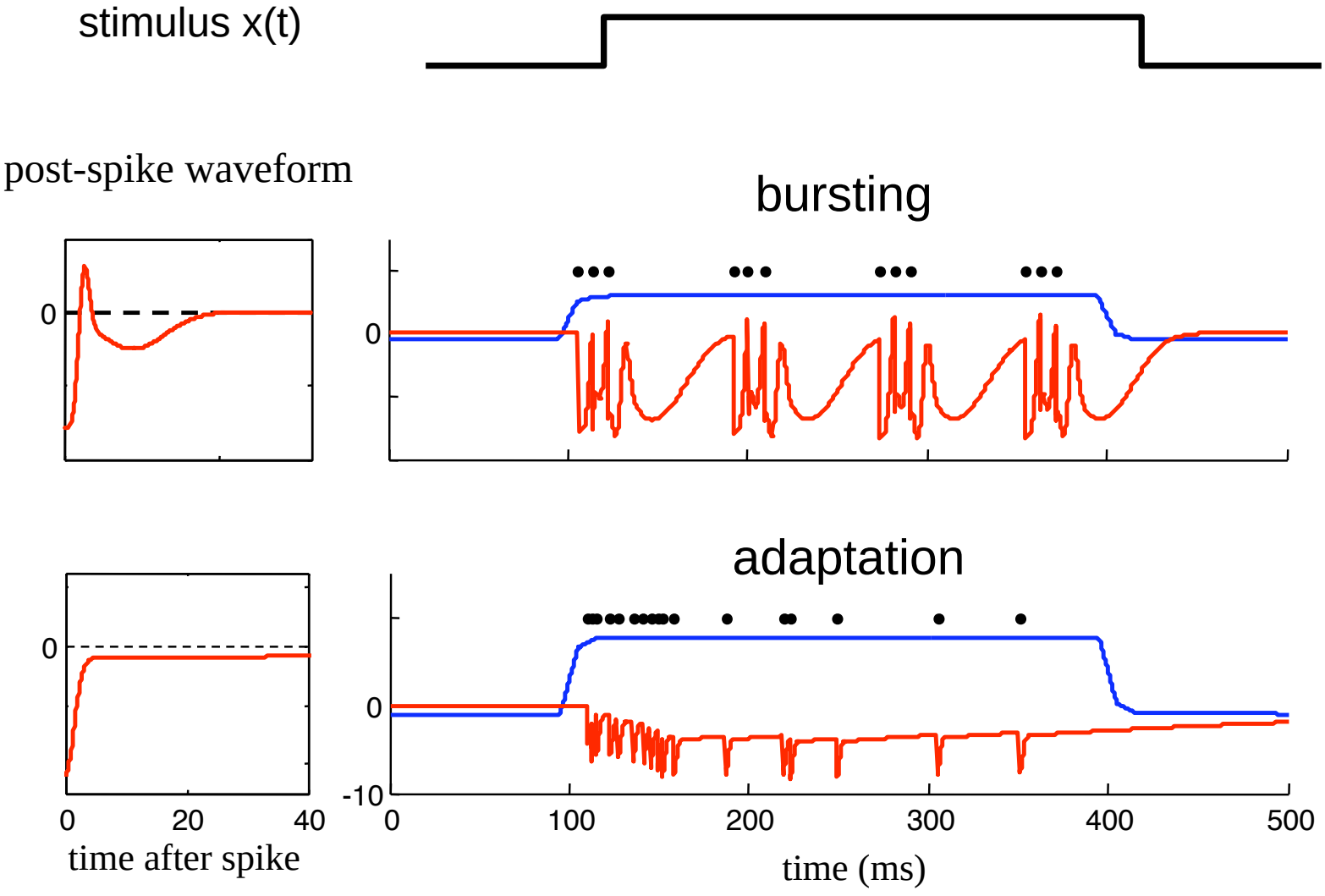
GLM dynamic behaviors



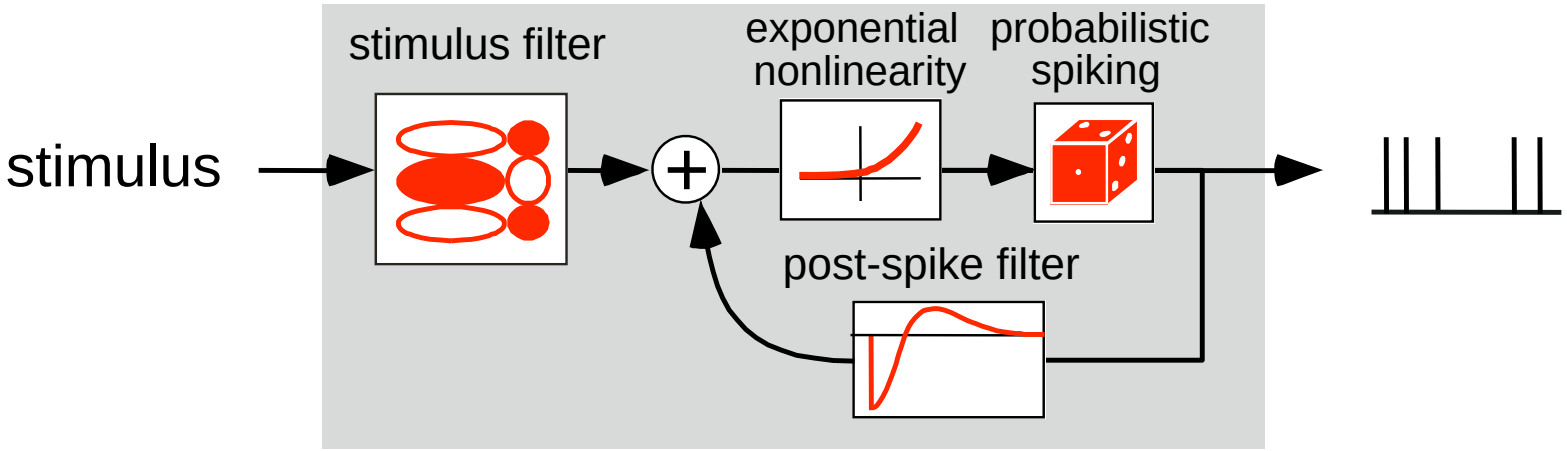
GLM dynamic behaviors



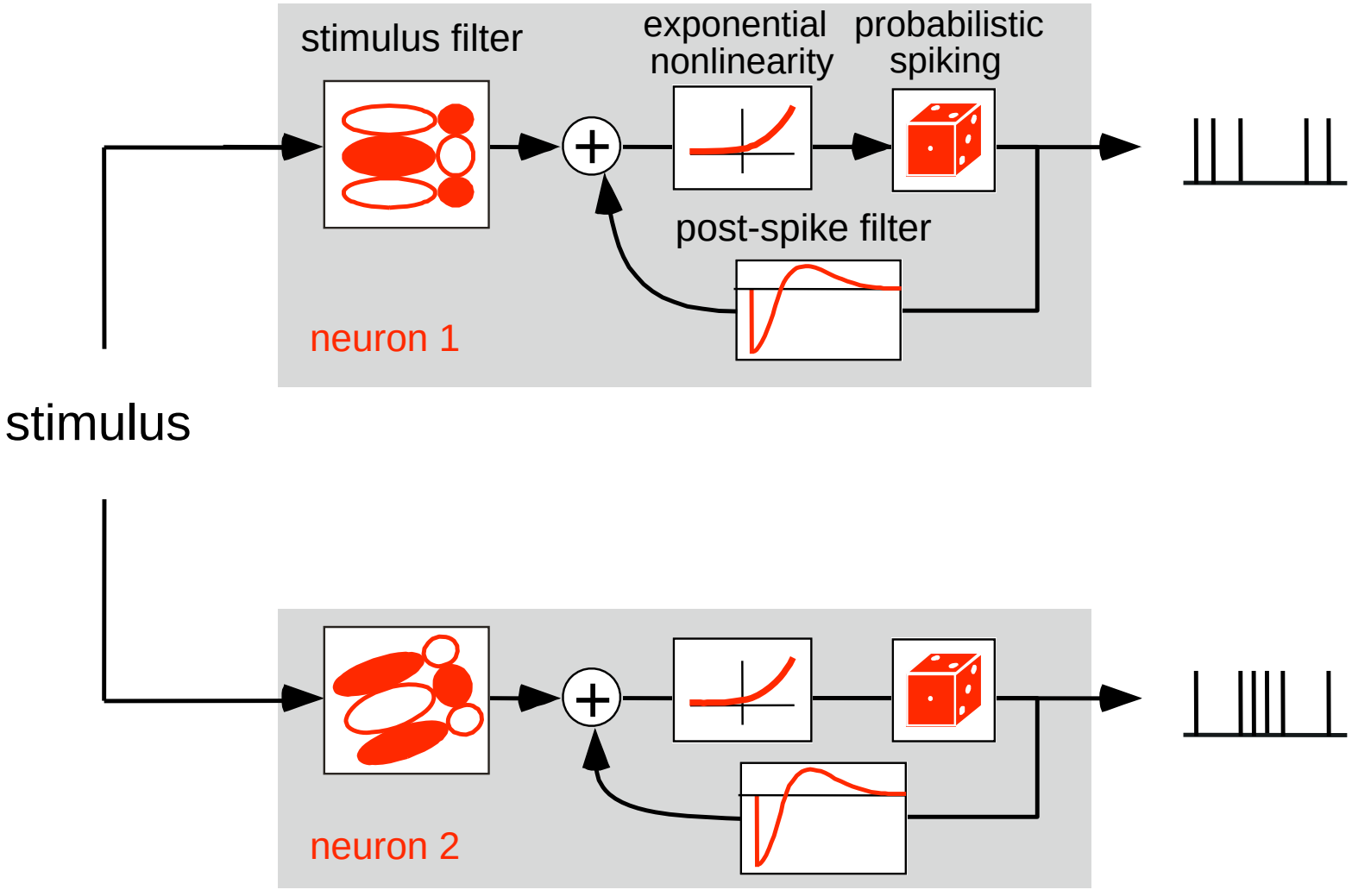
GLM dynamic behaviors



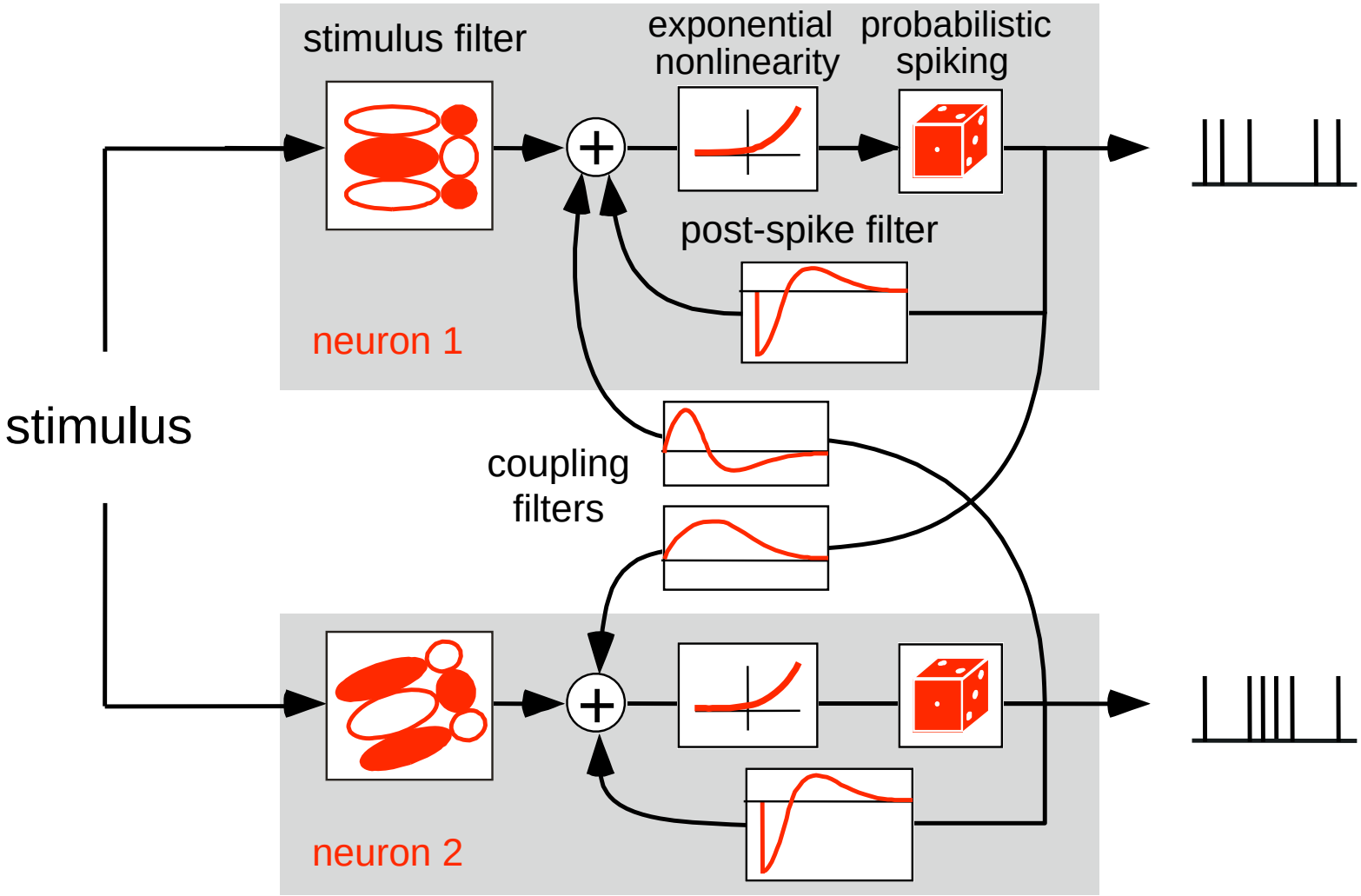
Generalized Linear Model (GLM)



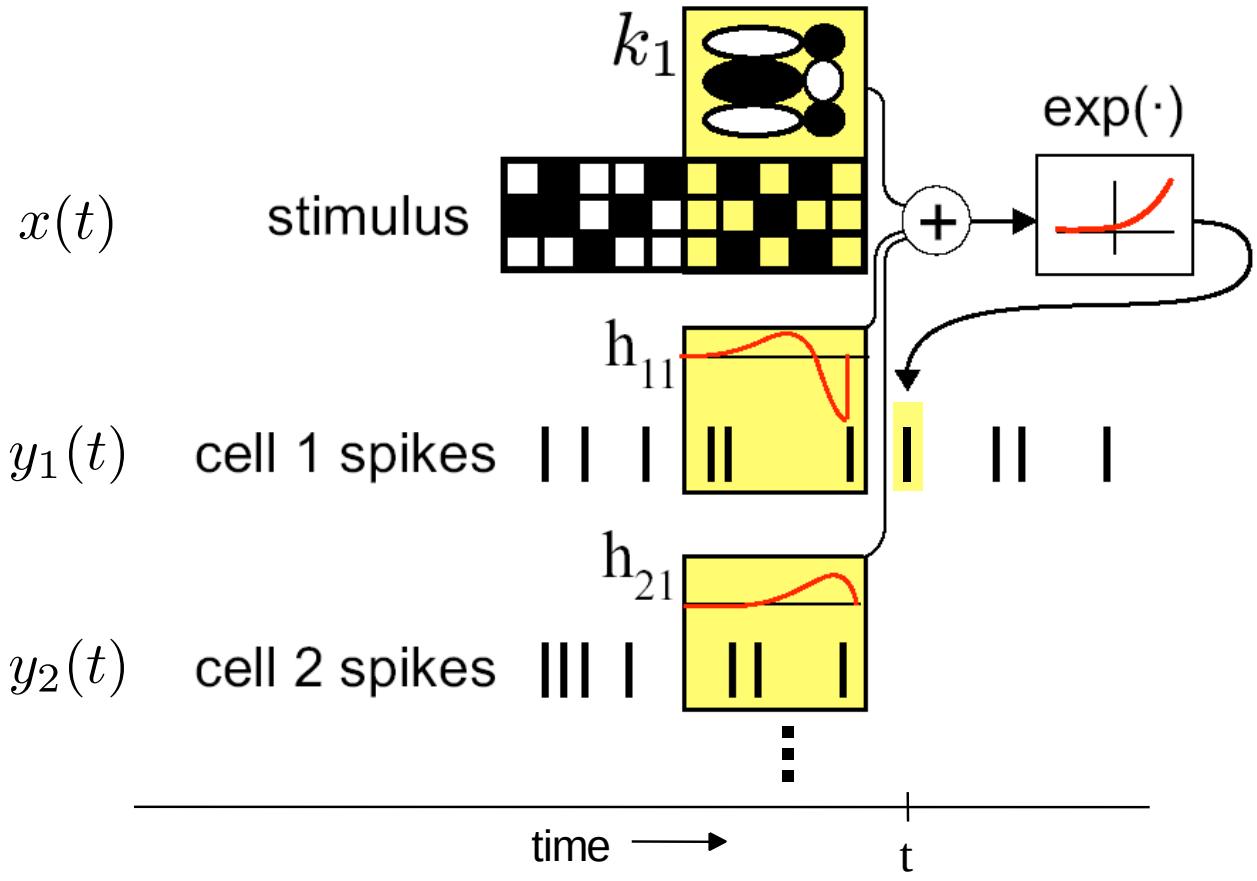
multi-neuron GLM



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):

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

Input nonlinearities

The basic linear model (for sounds):

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

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

Input nonlinearities

The basic linear model (for sounds):

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

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

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

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

Input nonlinearities

The basic linear model (for sounds):

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

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

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

Input nonlinearities

The basic linear model (for sounds):

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

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

$$\hat{r}(i) = \sum_j w_{jk}^{\mathbf{tf}} w_l^{\mathbf{l}} M_{ijkl} \quad \text{or} \quad \widehat{\mathbf{r}} = (\mathbf{w}^{\mathbf{tf}} \otimes \mathbf{w}^{\mathbf{l}}) \bullet \mathbf{M}.$$

Multilinear models

Multilinear forms are straightforward to optimise by alternating least squares.

Cost function:

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

Minimise iteratively, defining *matrices*

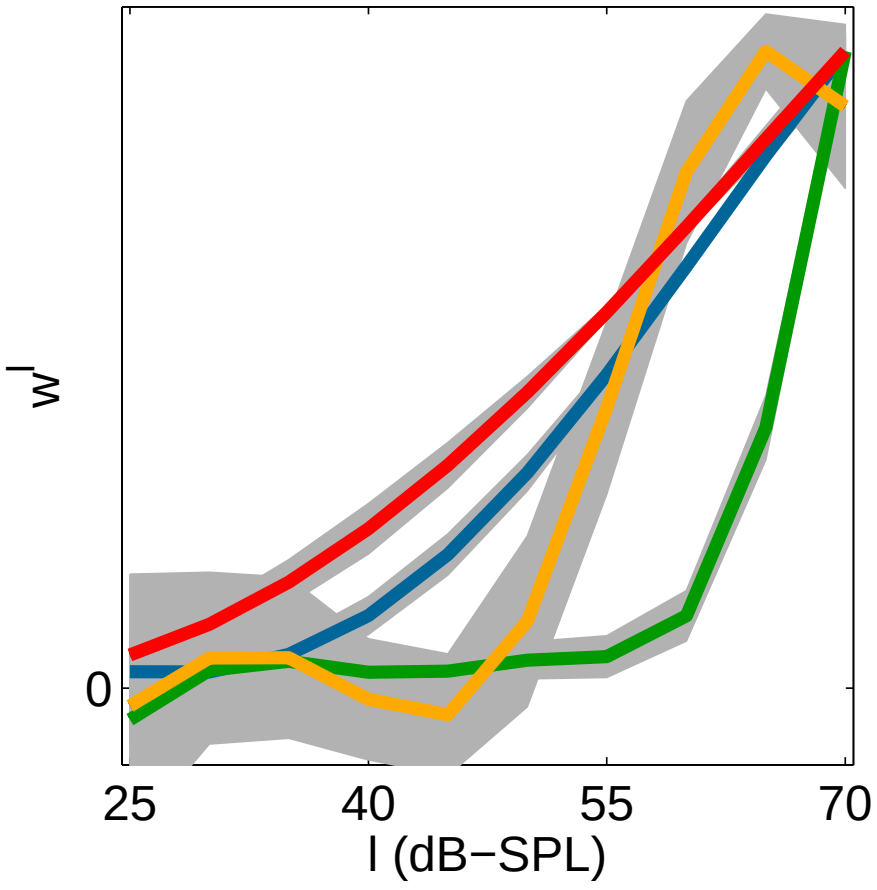
$$\mathbf{B} = \mathbf{w}^{\mathbf{l}} \bullet \mathbf{M} \qquad \text{and} \qquad \mathbf{A} = \mathbf{w}^{\mathbf{tf}} \bullet \mathbf{M}$$

and updating

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

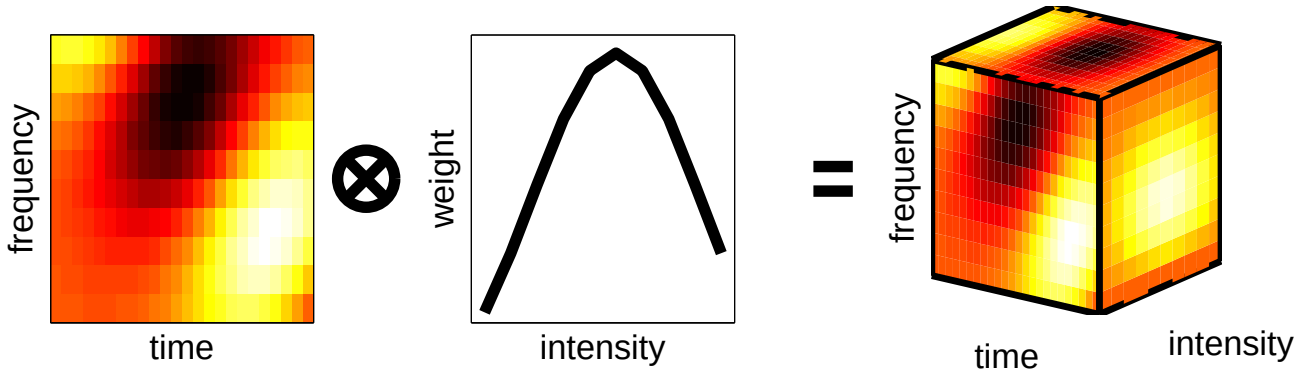
Each linear regression step can be regularised by evidence optimisation (suboptimal), with uncertainty propagated approximately using *variational* methods.

Some input non-linearities



Parameter grouping

Separable models: (time) $\otimes$ (frequency). The input nonlinearity model is separable in another sense: (time, frequency) $\otimes$ (sound level).

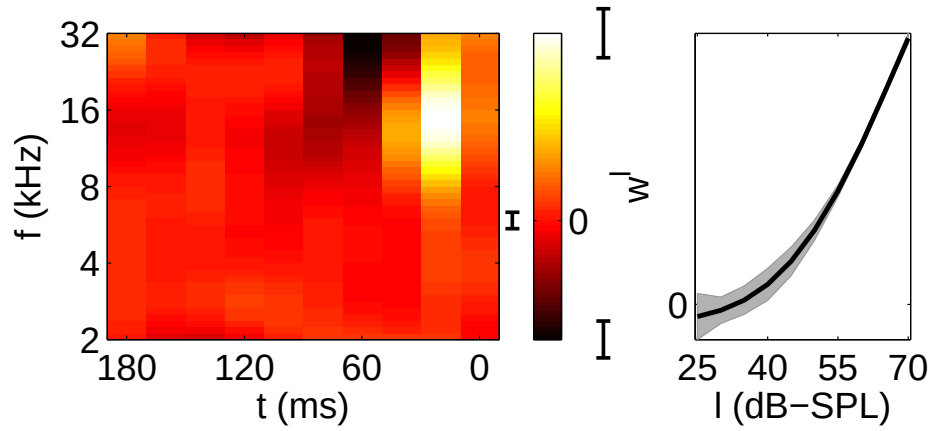


Other separations:

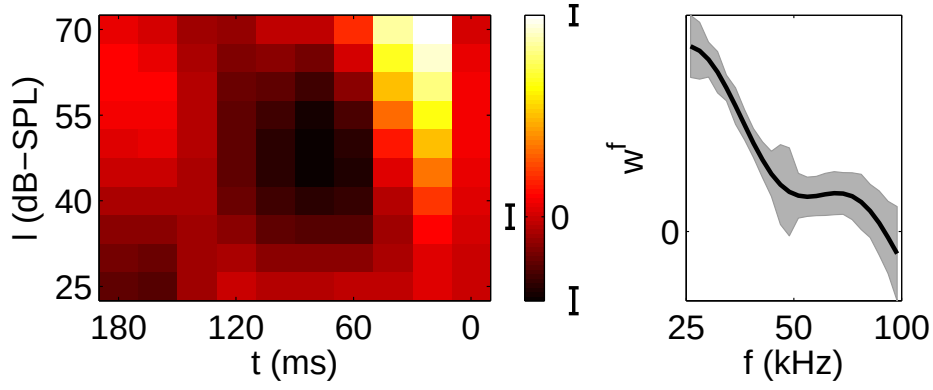
- (time, sound level) $\otimes$ (frequency): $\widehat{\mathbf{r}} = (\mathbf{w}^{\mathbf{tl}} \otimes \mathbf{w}^{\mathbf{f}}) \bullet \mathbf{M}$,
- (frequency, sound level) $\otimes$ (time): $\widehat{\mathbf{r}} = (\mathbf{w}^{\mathbf{fl}} \otimes \mathbf{w}^{\mathbf{t}}) \bullet \mathbf{M}$,
- (time) $\otimes$ (frequency) $\otimes$ (sound level): $\widehat{\mathbf{r}} = (\mathbf{w}^{\mathbf{l}} \otimes \mathbf{w}^{\mathbf{f}} \otimes \mathbf{w}^{\mathbf{l}}) \bullet \mathbf{M}$.

Some examples

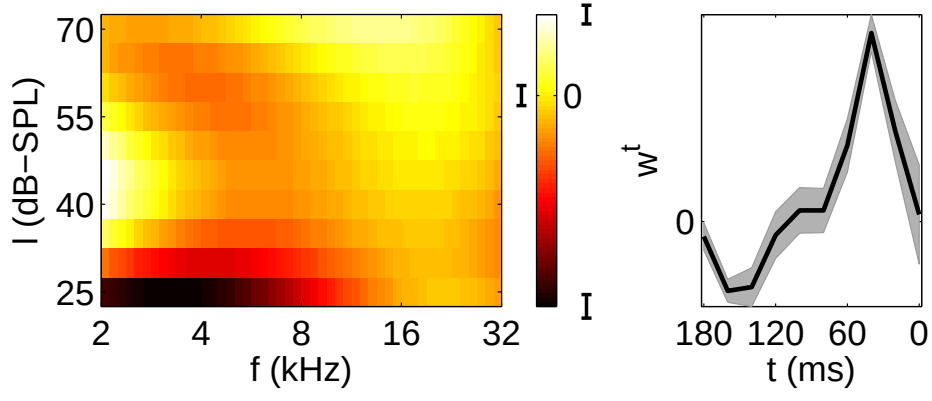
(time, frequency) $\otimes$ (sound level):



(time, sound level) $\otimes$ (frequency):



(frequency, sound level) $\otimes$ (time):



Variable (combination-dependent) input gain

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

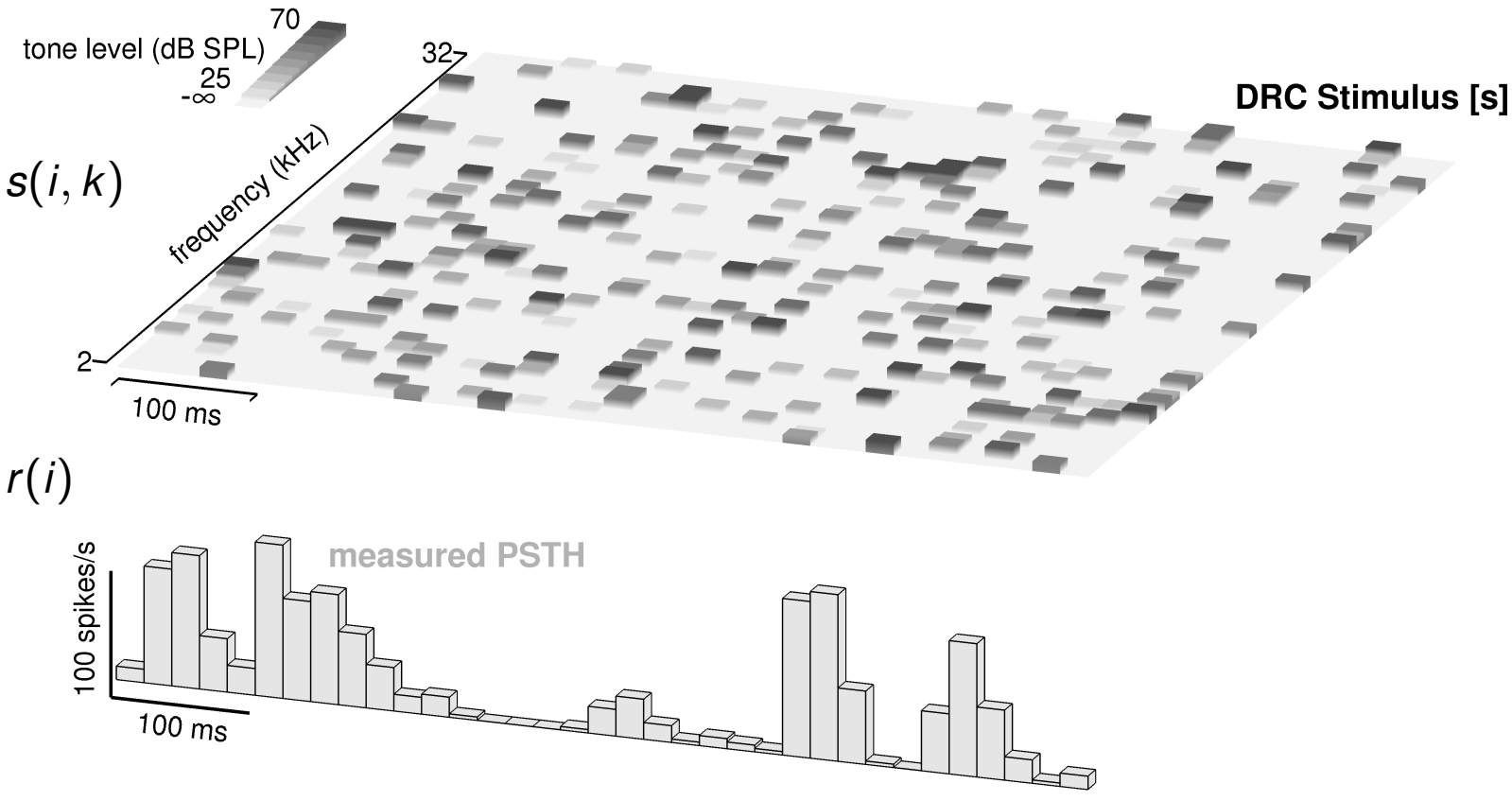
Variable (combination-dependent) input gain

- 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

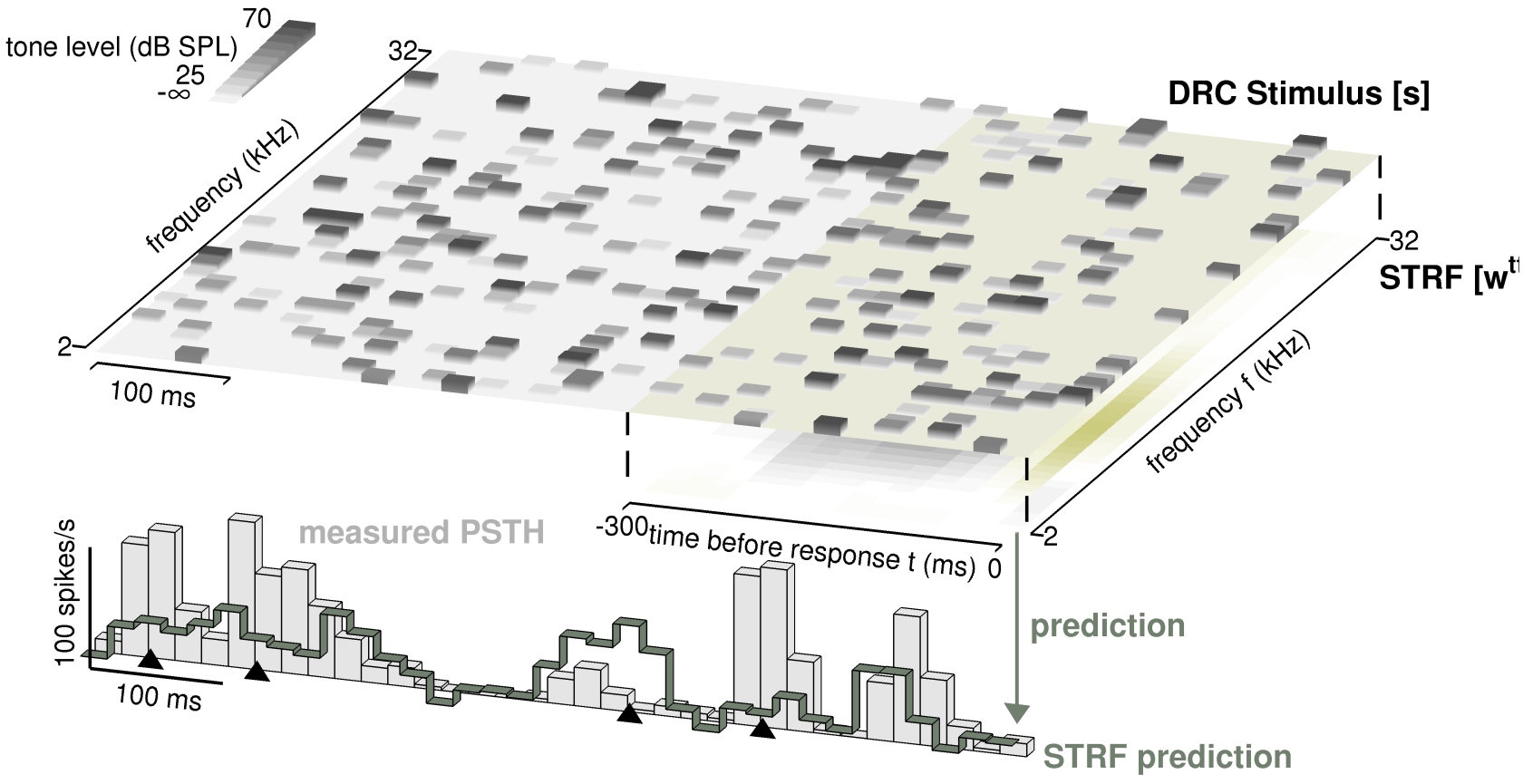
- Sensitivities to different points in sensory space are not independent.
- Rather, the sensitivity at one point depends on other elements of the stimulus that create a *local* sensory context.
- This context adjusts the input gain of the cell from moment to moment, dynamically refining the shape of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

A context-sensitive model



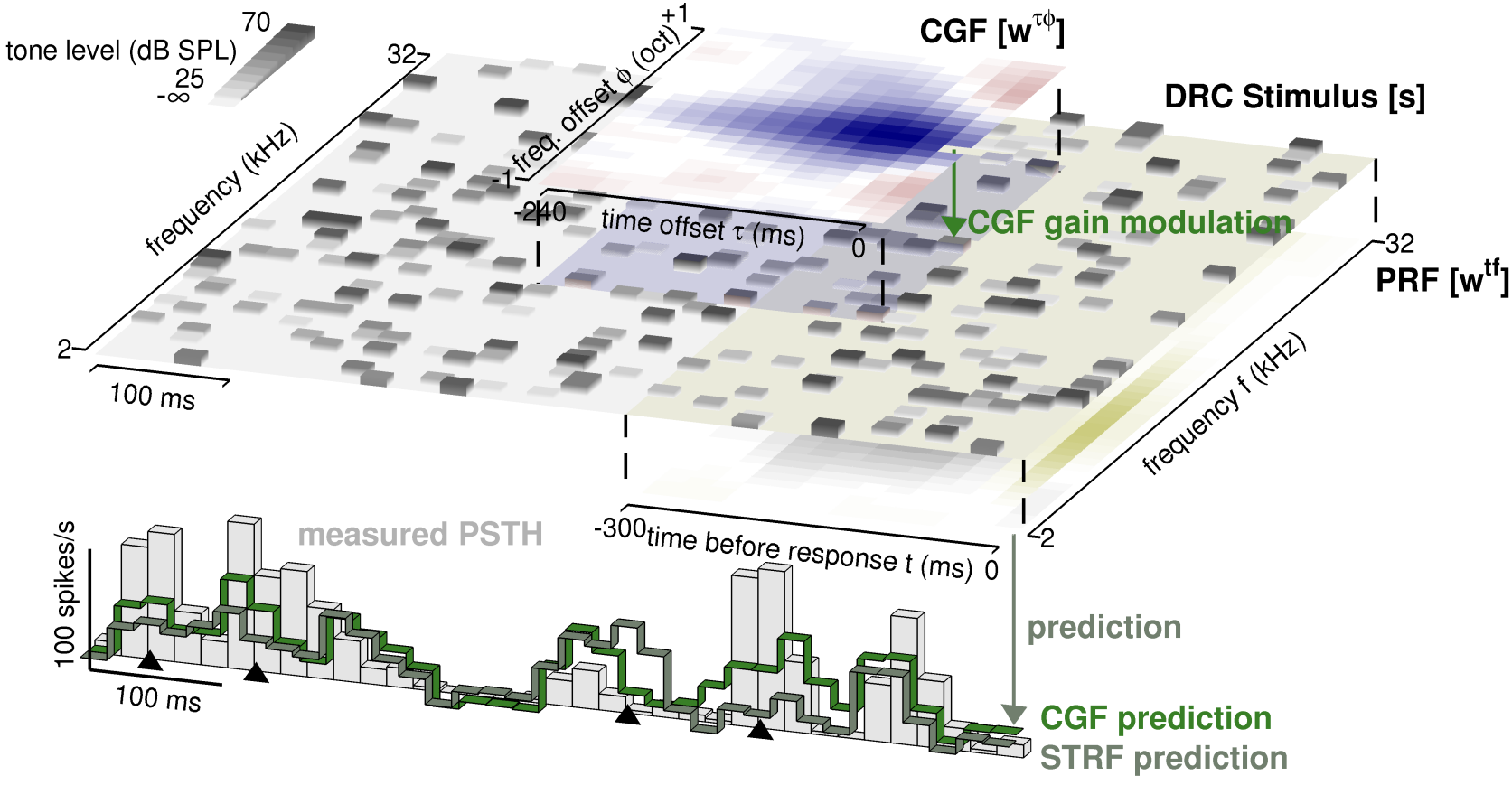
A context-sensitive model

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

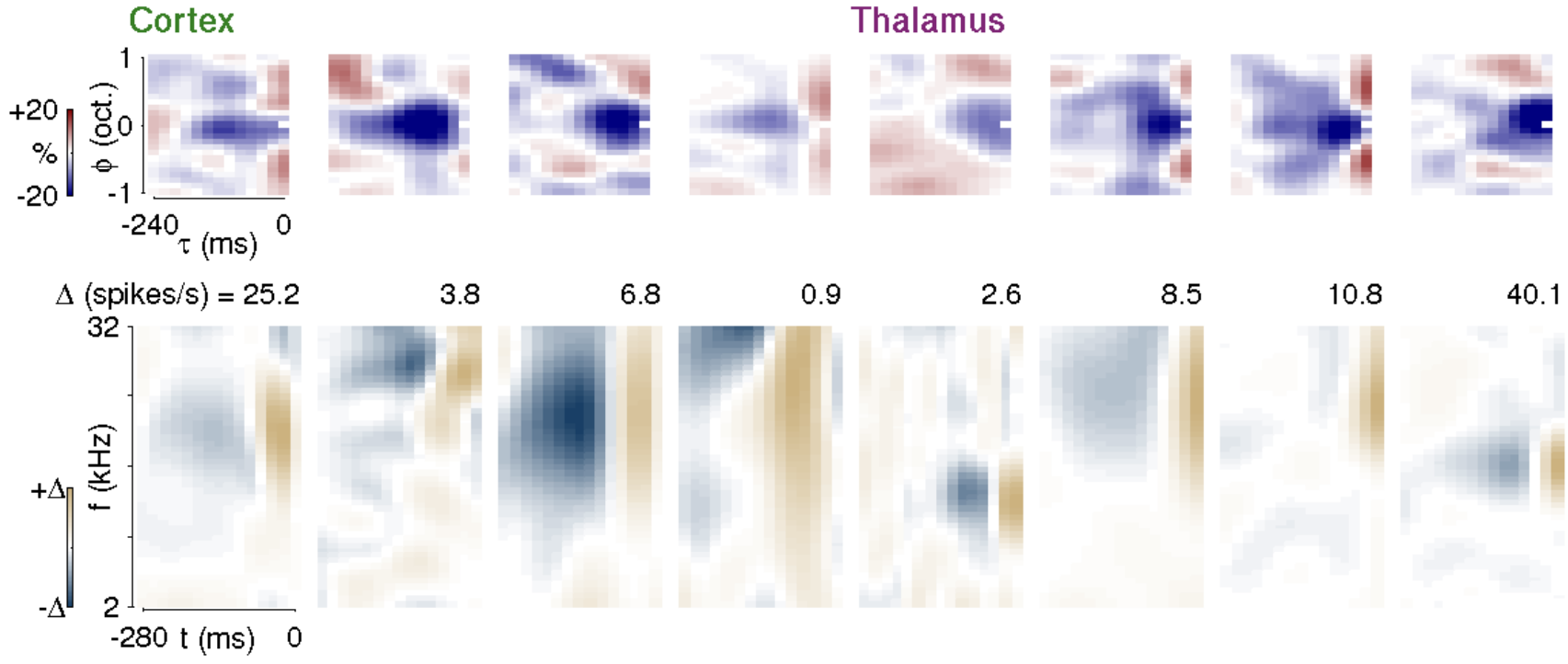


A context-sensitive model

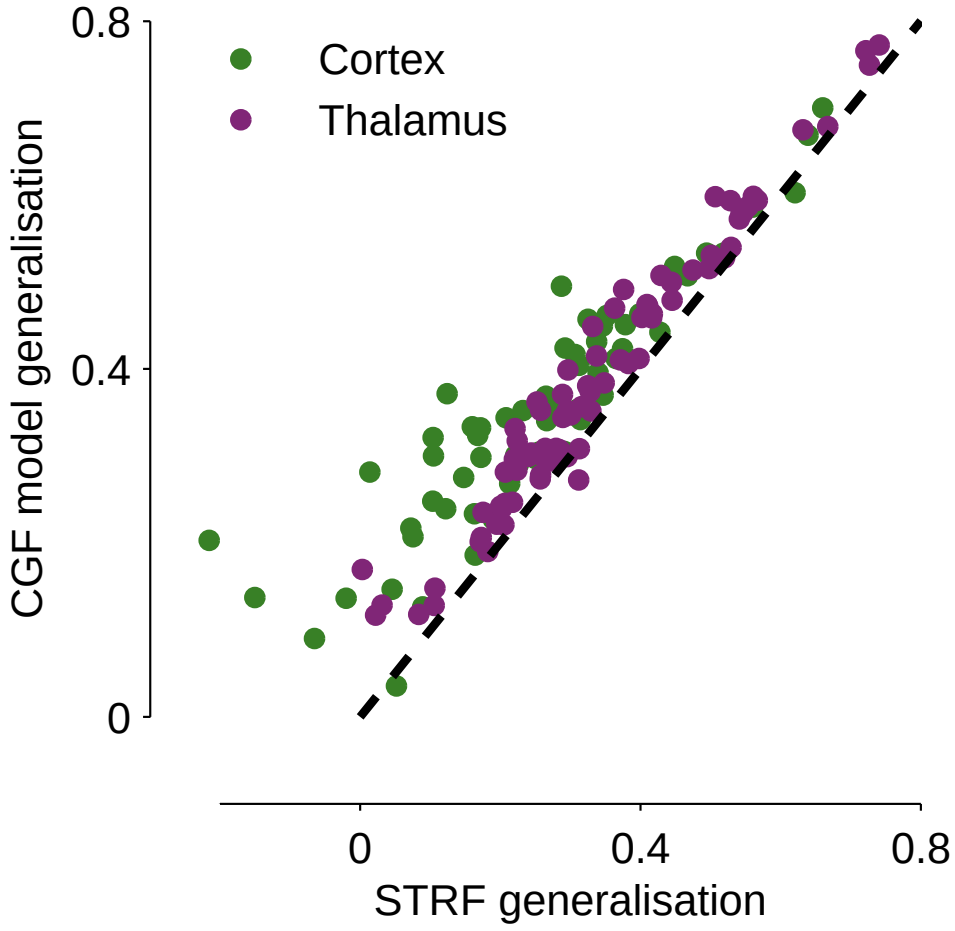
$$\hat{r}(i) = c + \sum_{j=0}^{J} \sum_{k=1}^{K} w^{tf}_{j+1,k} s(i-j,k) \left(1 + \sum_{m=0}^{M} \sum_{n=-N}^{N} w^{\tau\phi}_{m+1,n+N+1} s(i-j-m,k+n) \right)$$



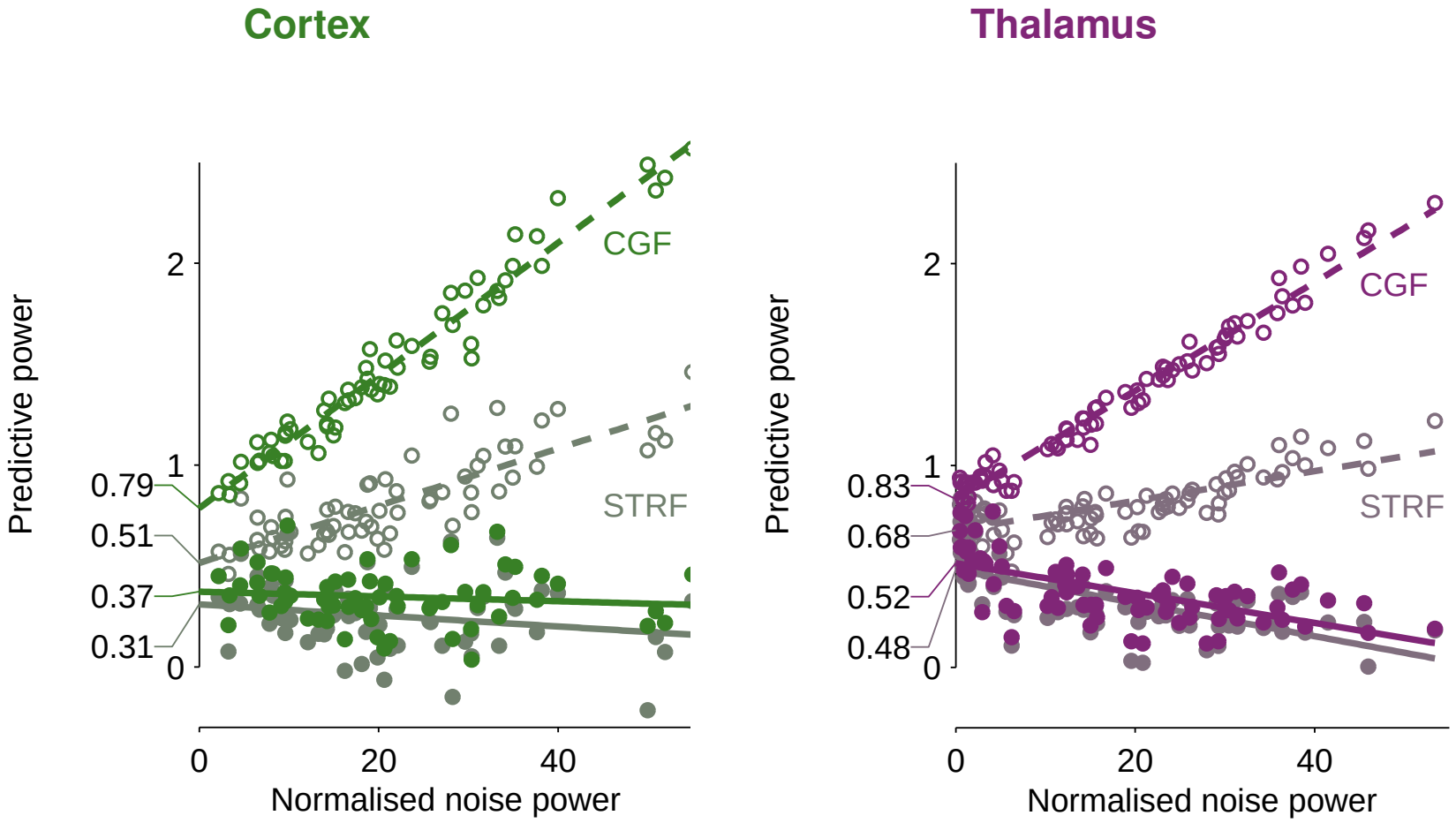
Some examples



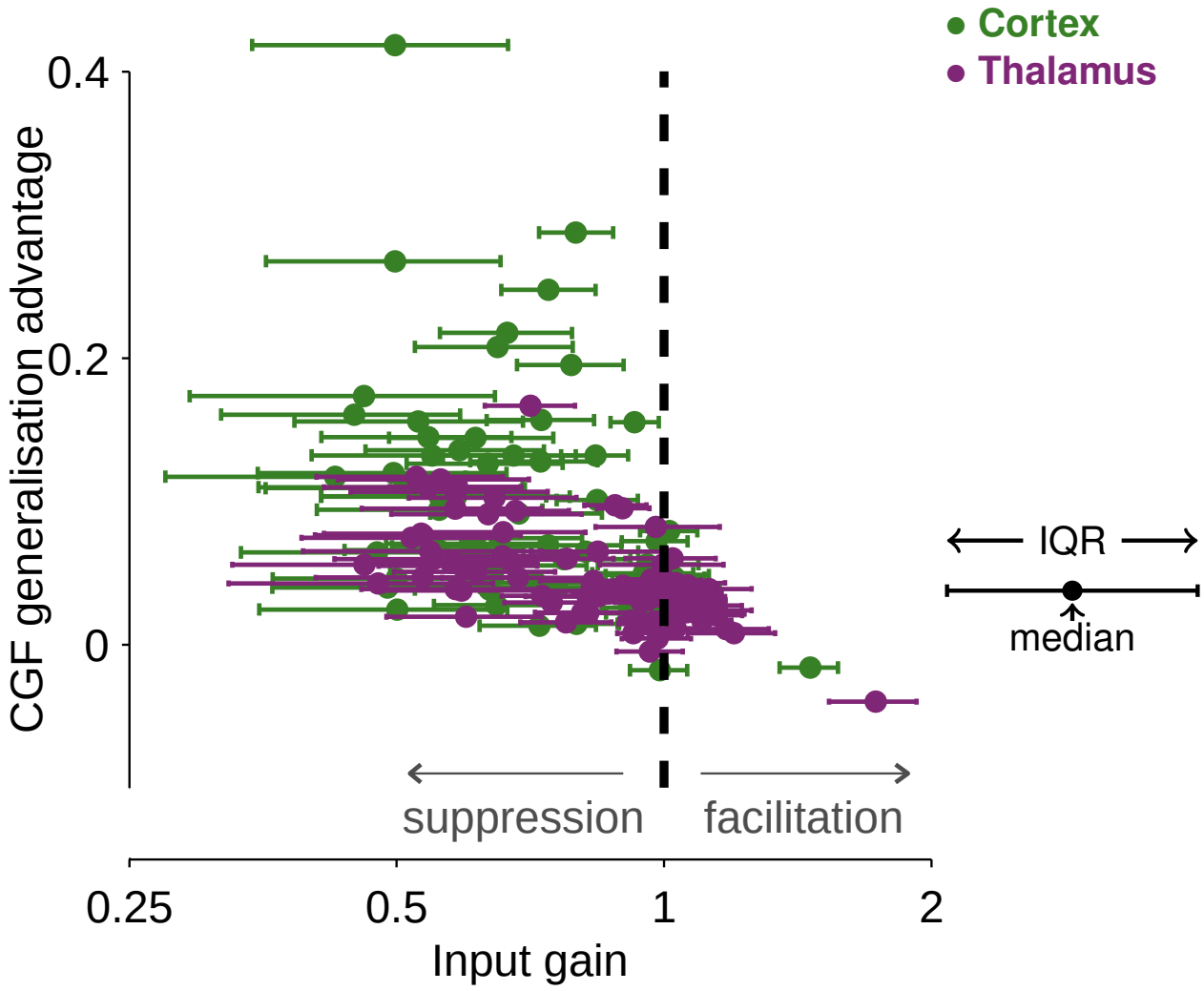
Predictive performance



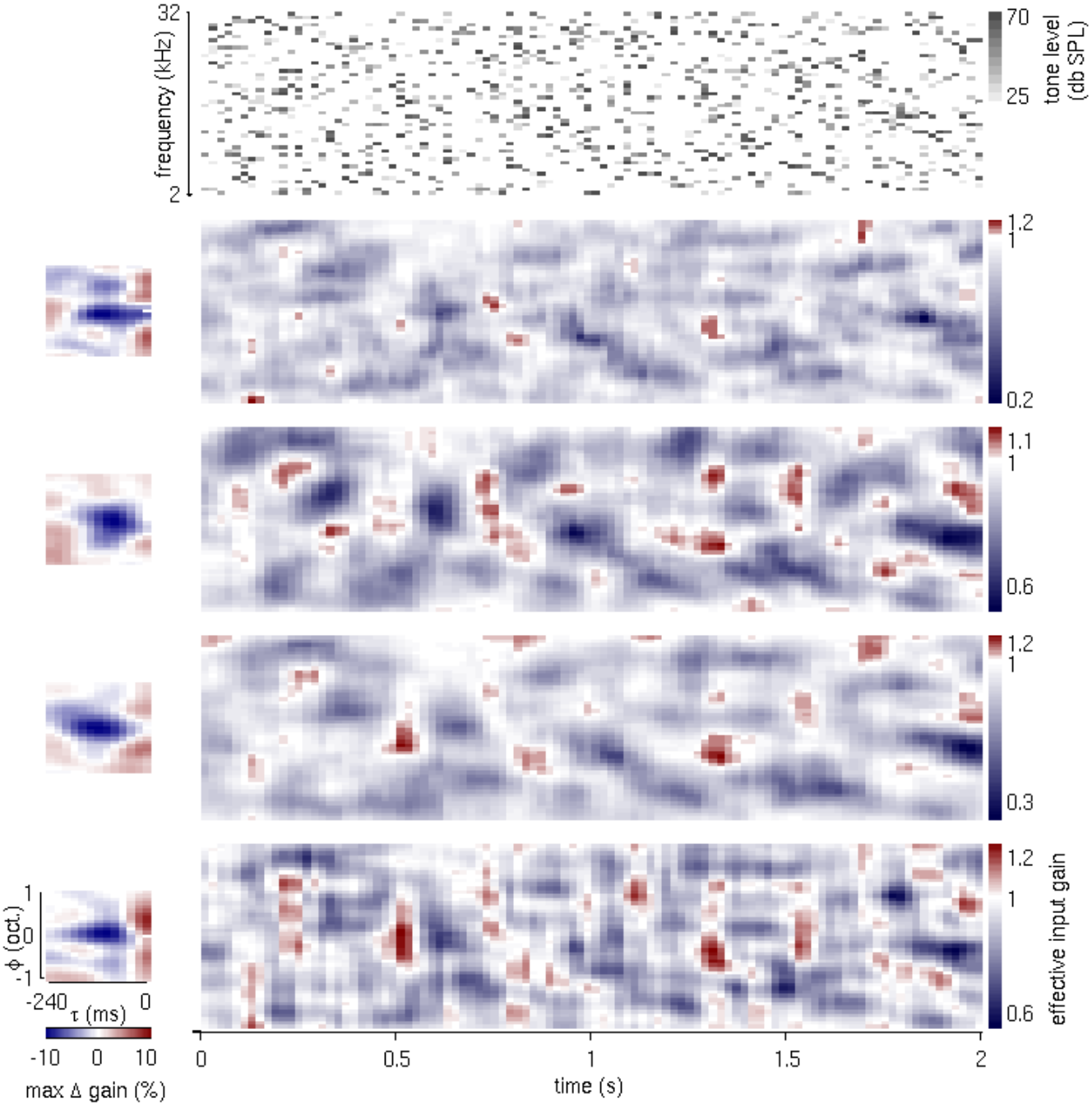
Predictive performance



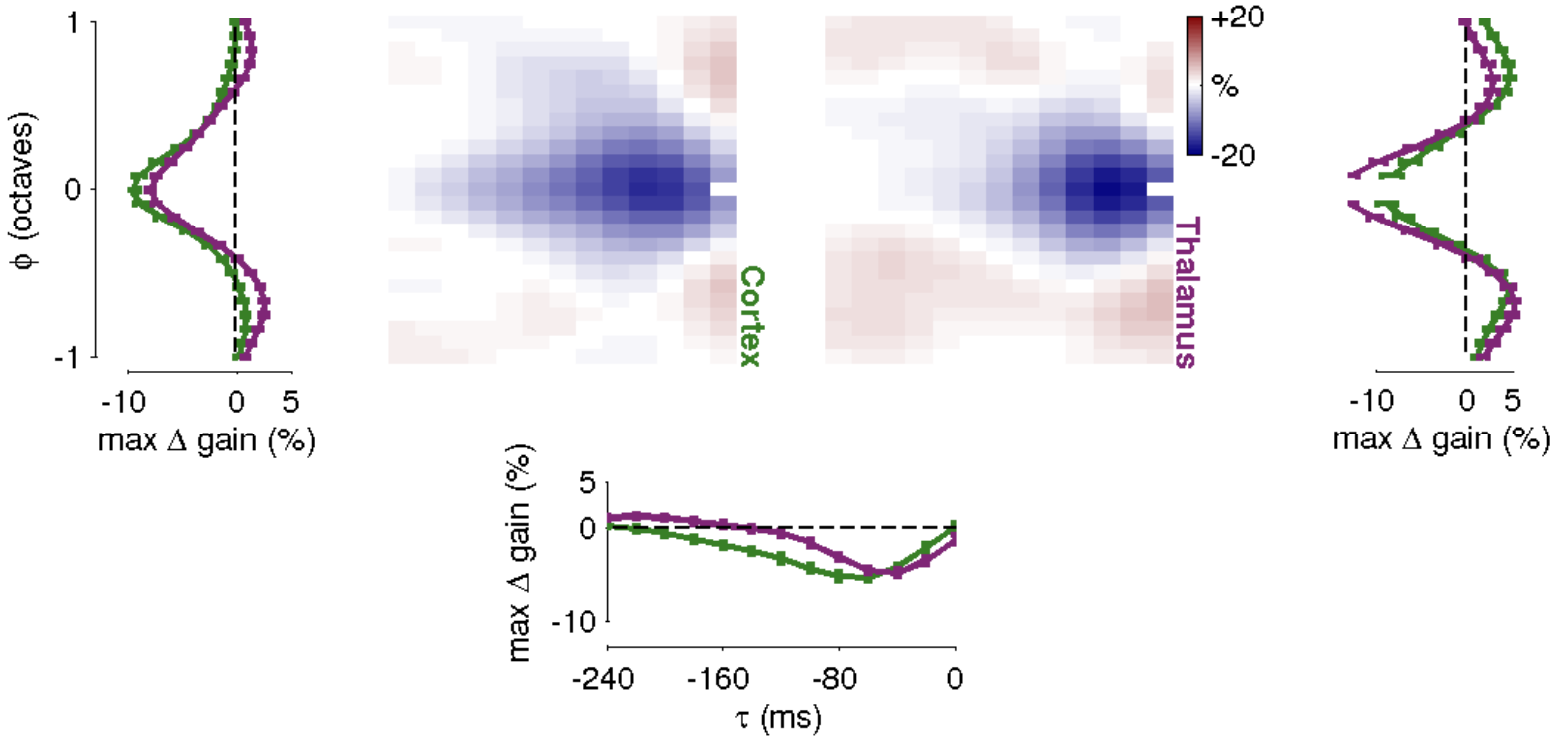
Range of input gain



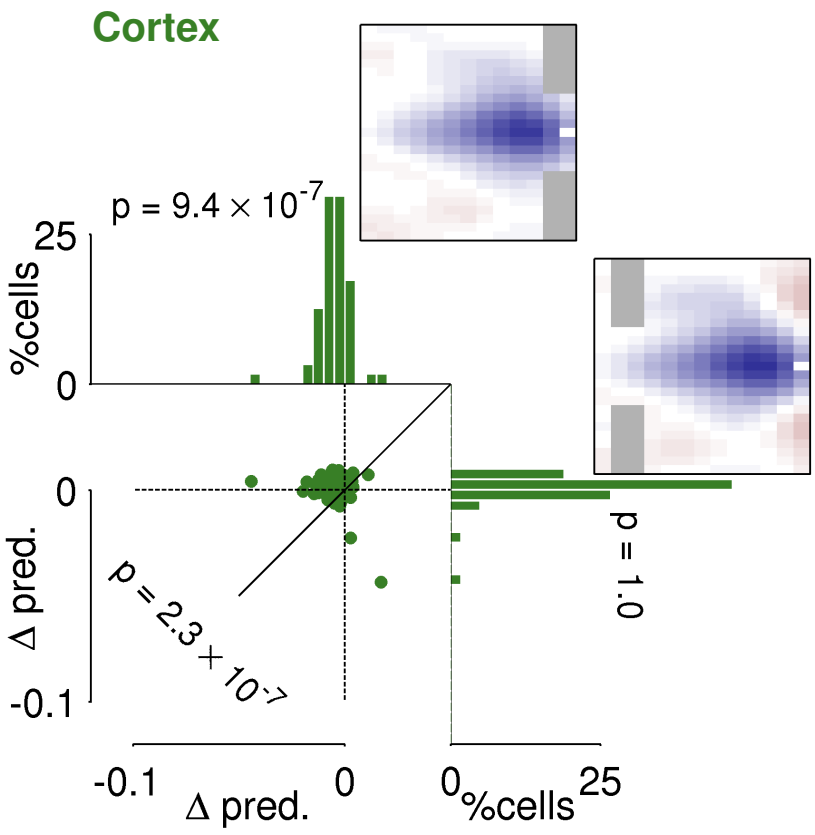
Input gain fluctuates rapidly

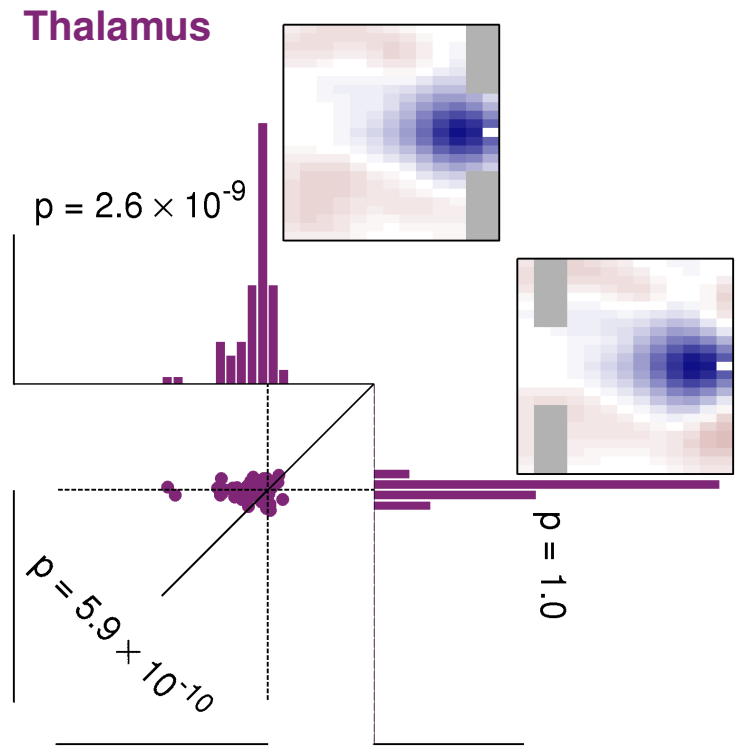


Mean CGFs

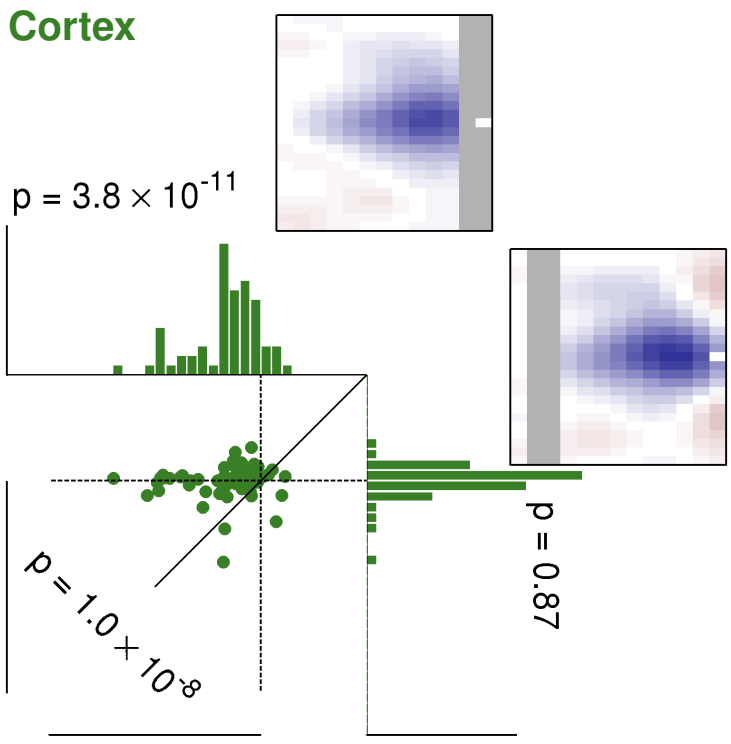

1
0
-1
ϕ (octaves)
-10
0
5
max Δ gain (%)
Cortex
Thalamus
+20
%
-20
-10
0
5
max Δ gain (%)
max Δ gain (%)
5
0
-10
-240
-160
-80
0
τ (ms)

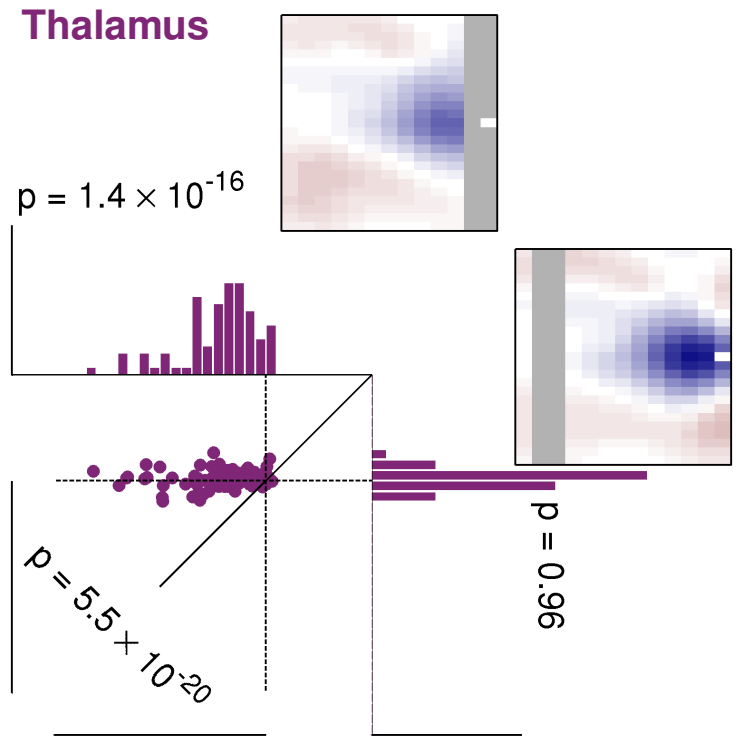
Component significance



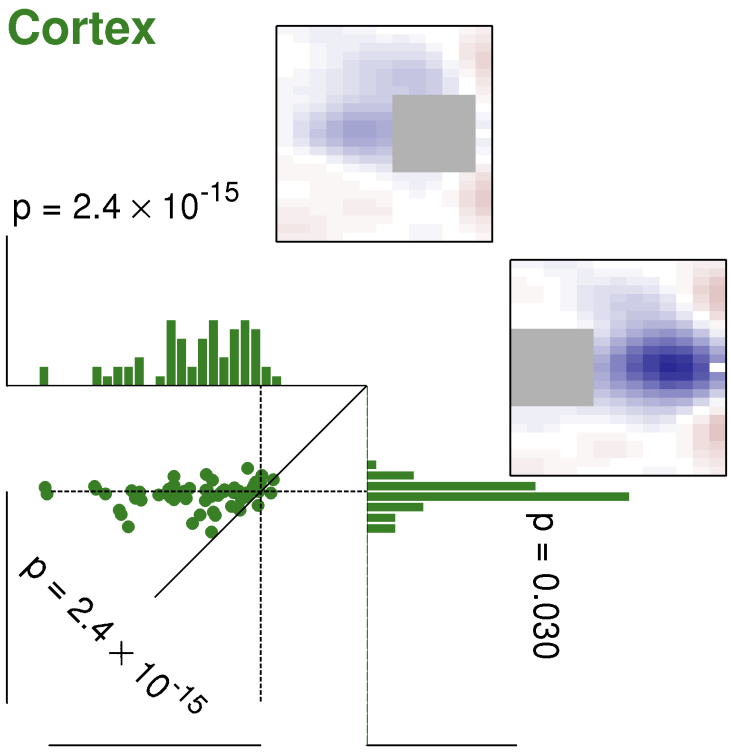


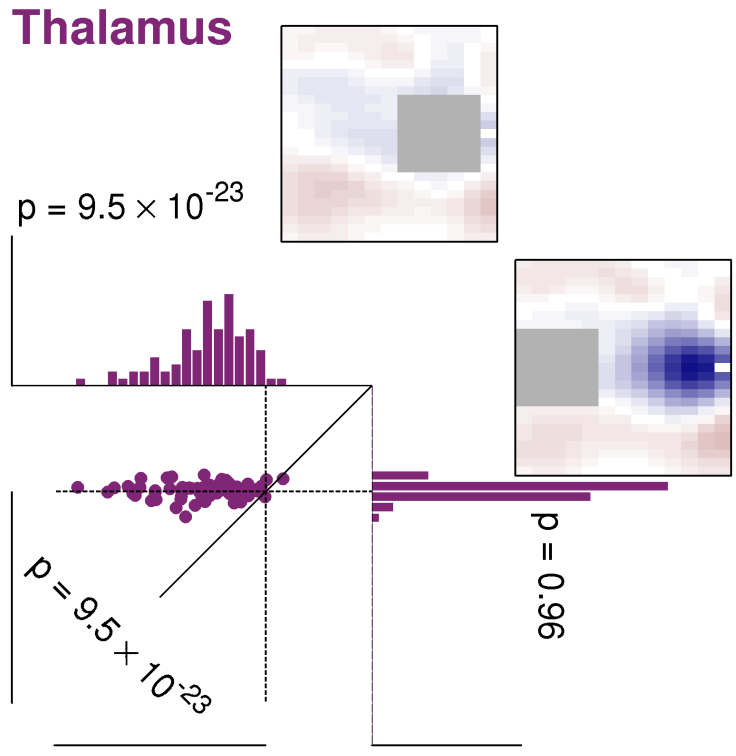
Component significance



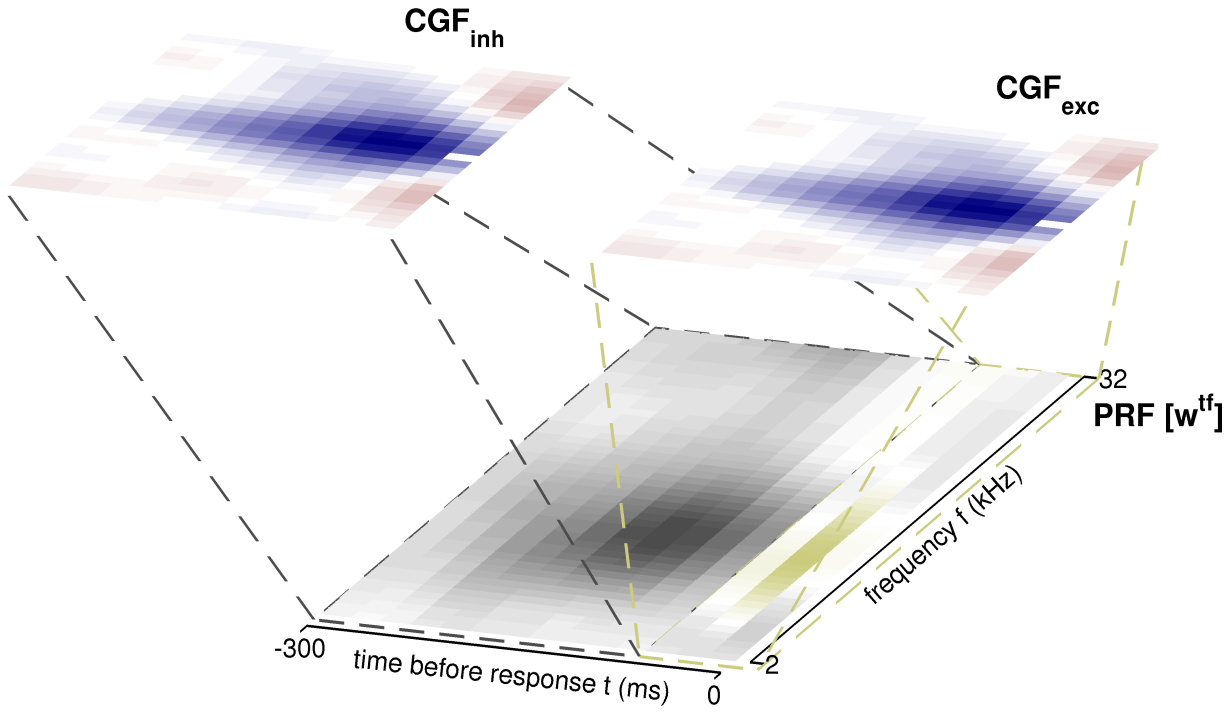


Component significance



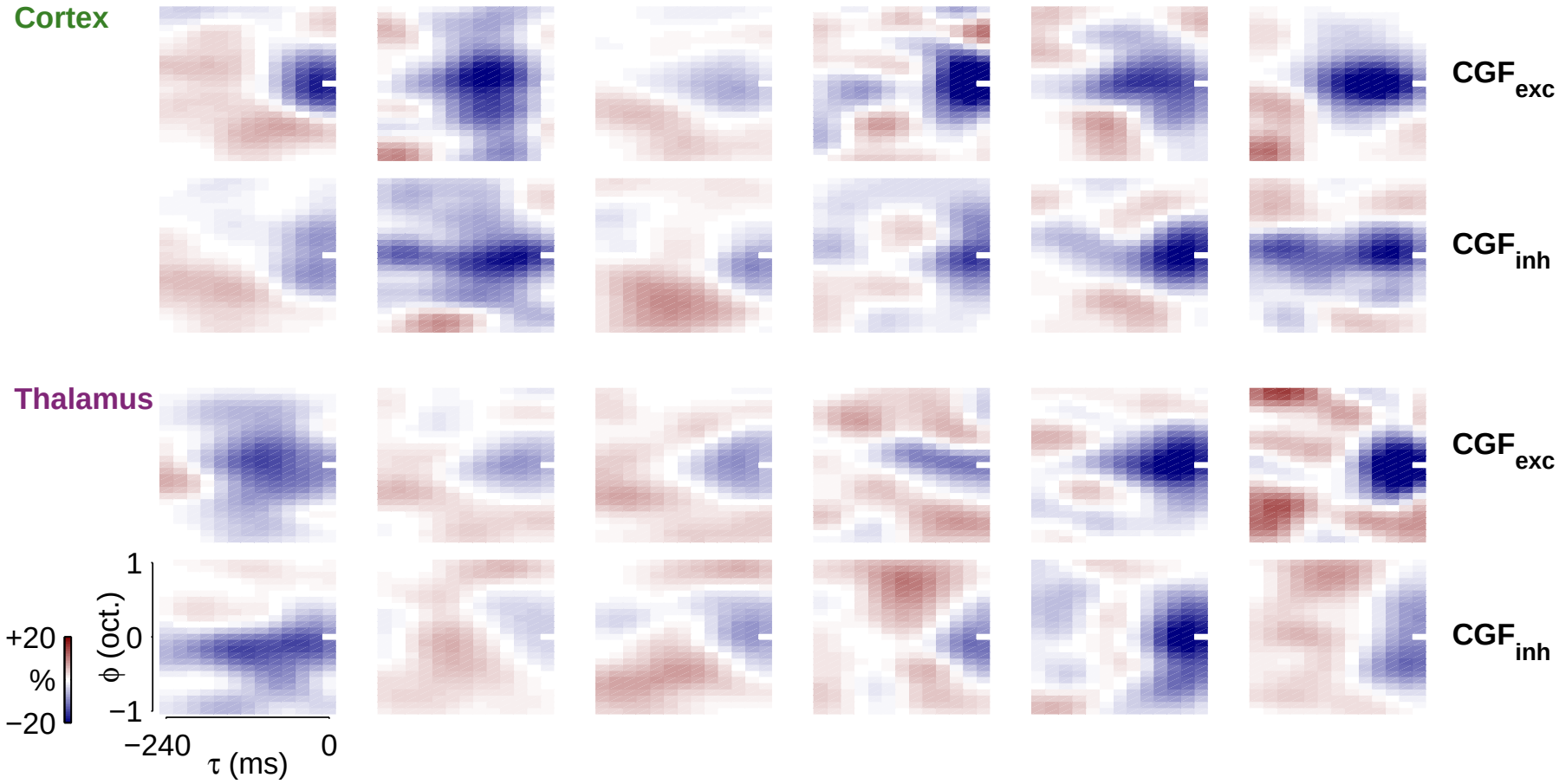


CGF consistency across the PRF

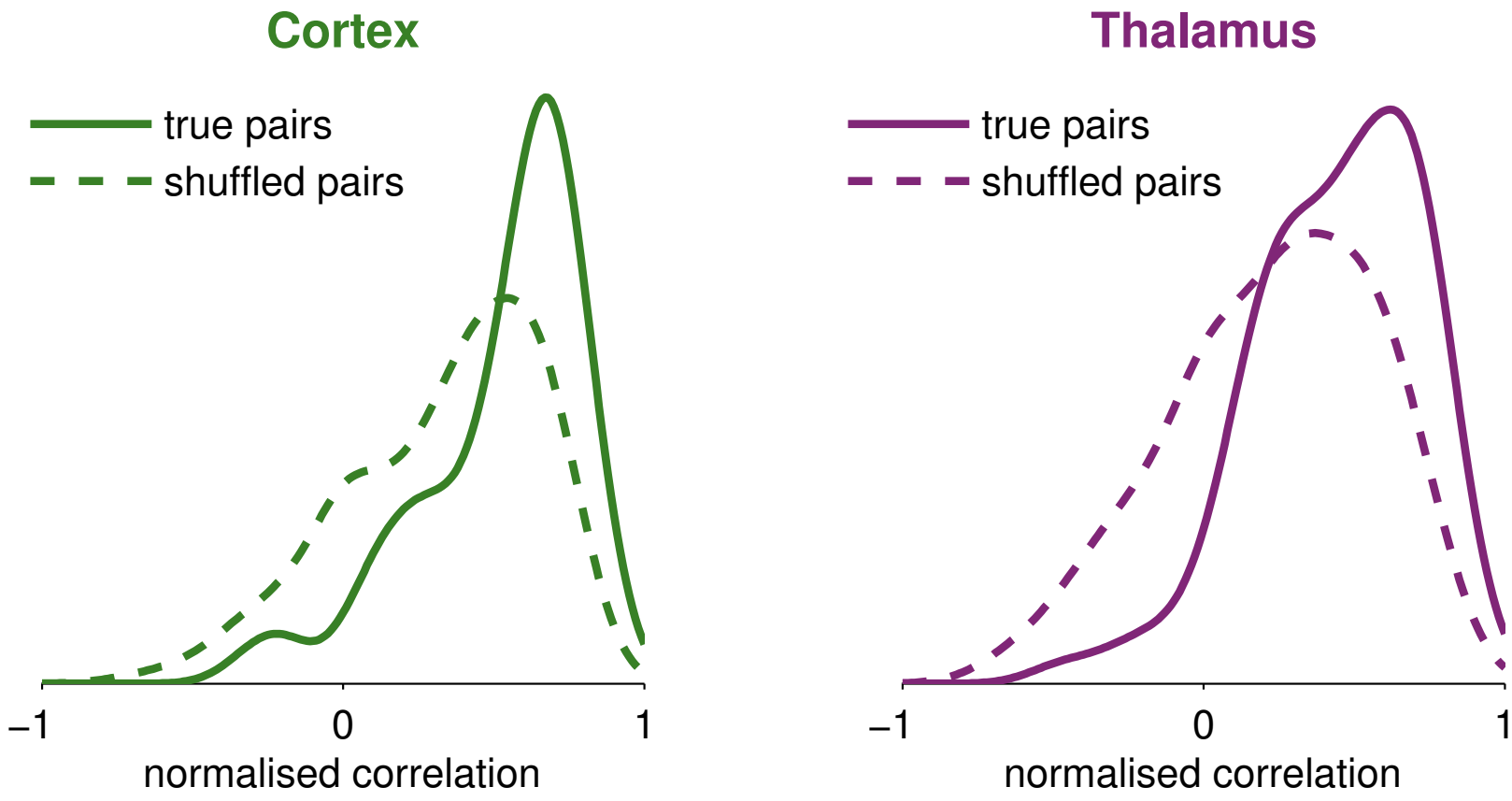


- As the CGF can be associated with the PRF weights rather than the stimulus, we can apply different CGFs to different PRF domains.

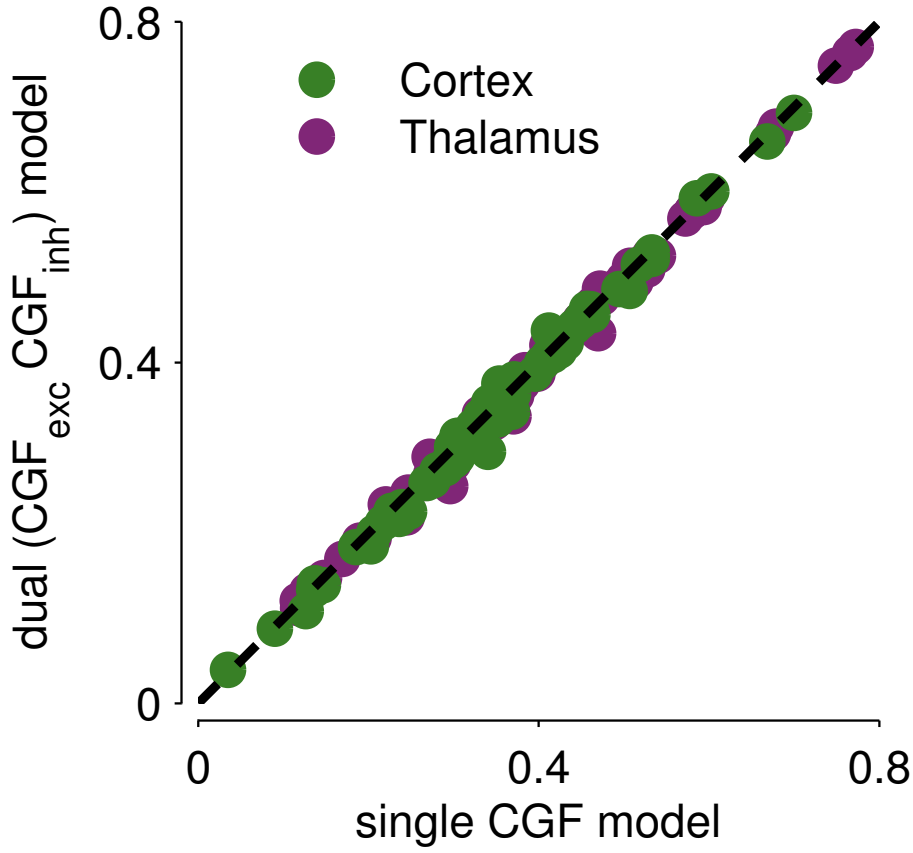
CGF consistency across the PRF



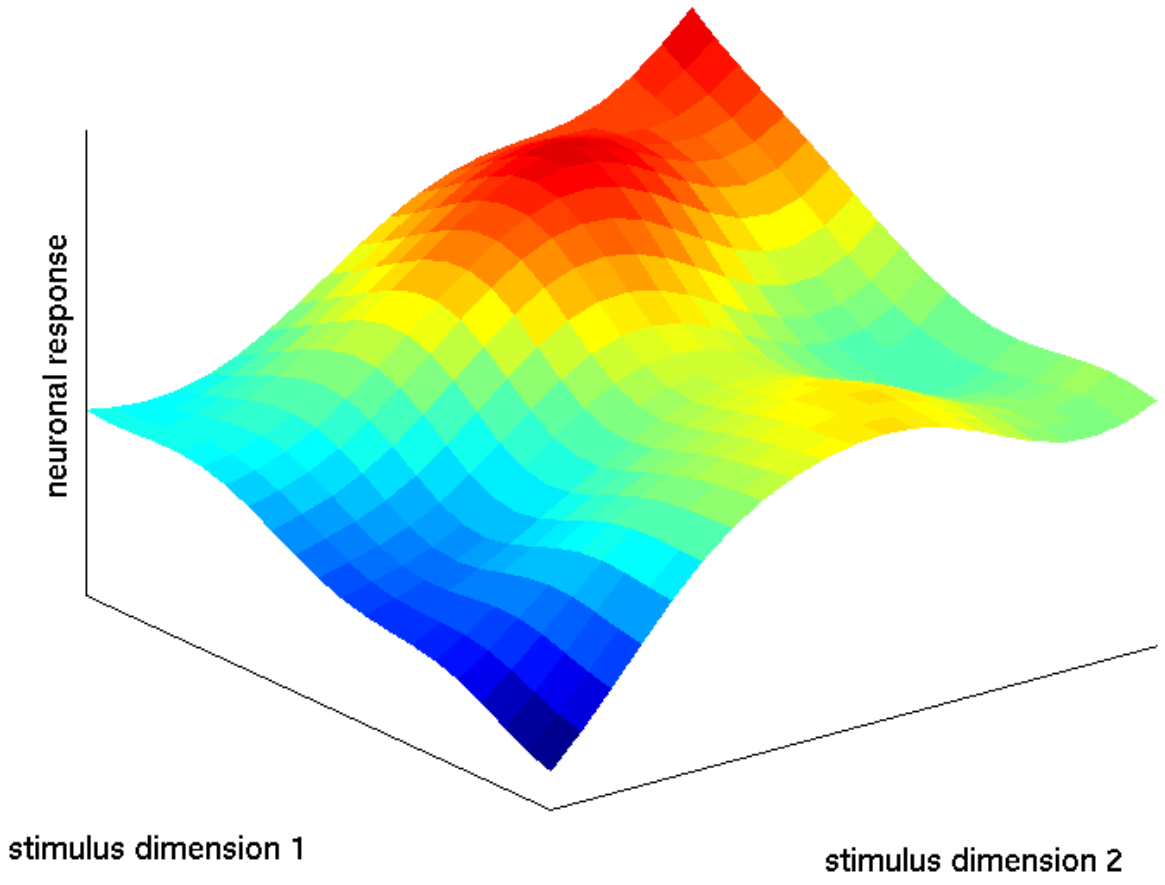
CGF consistency across the PRF



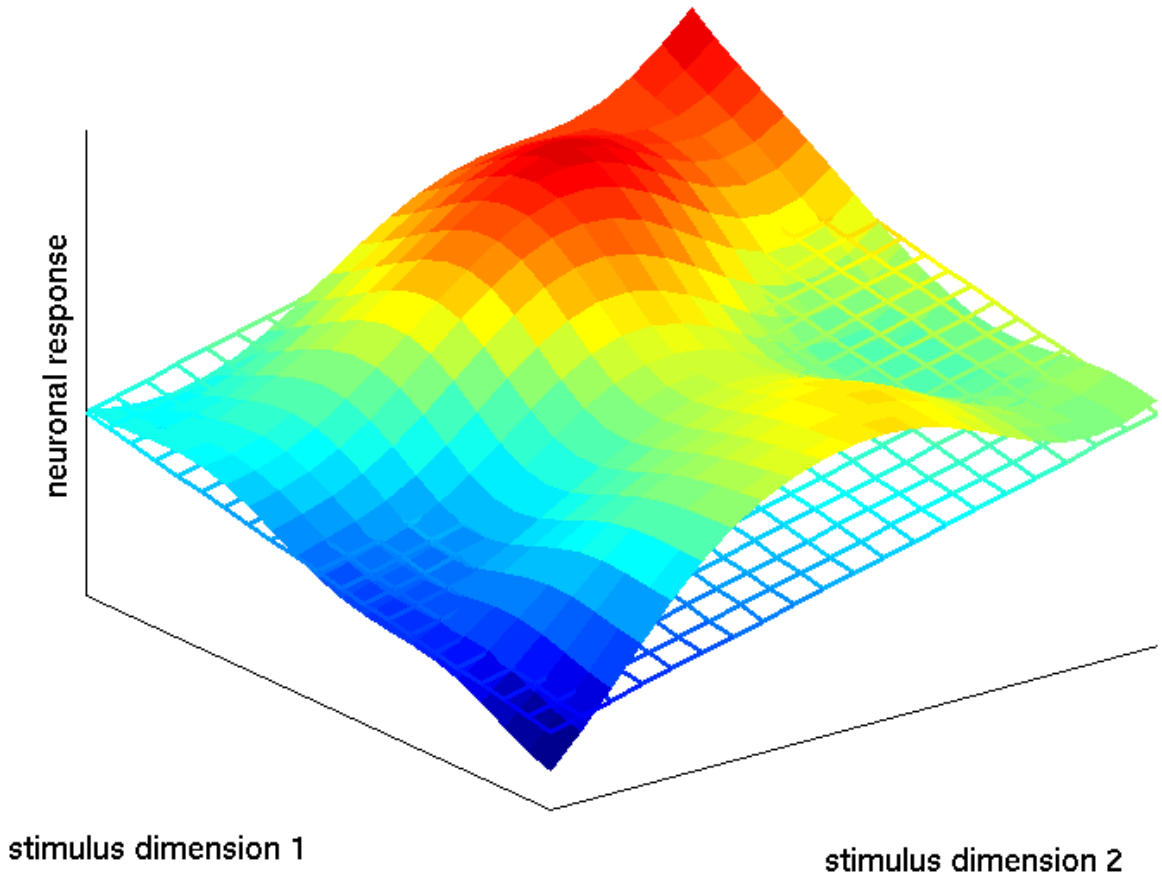
CGF consistency across the PRF



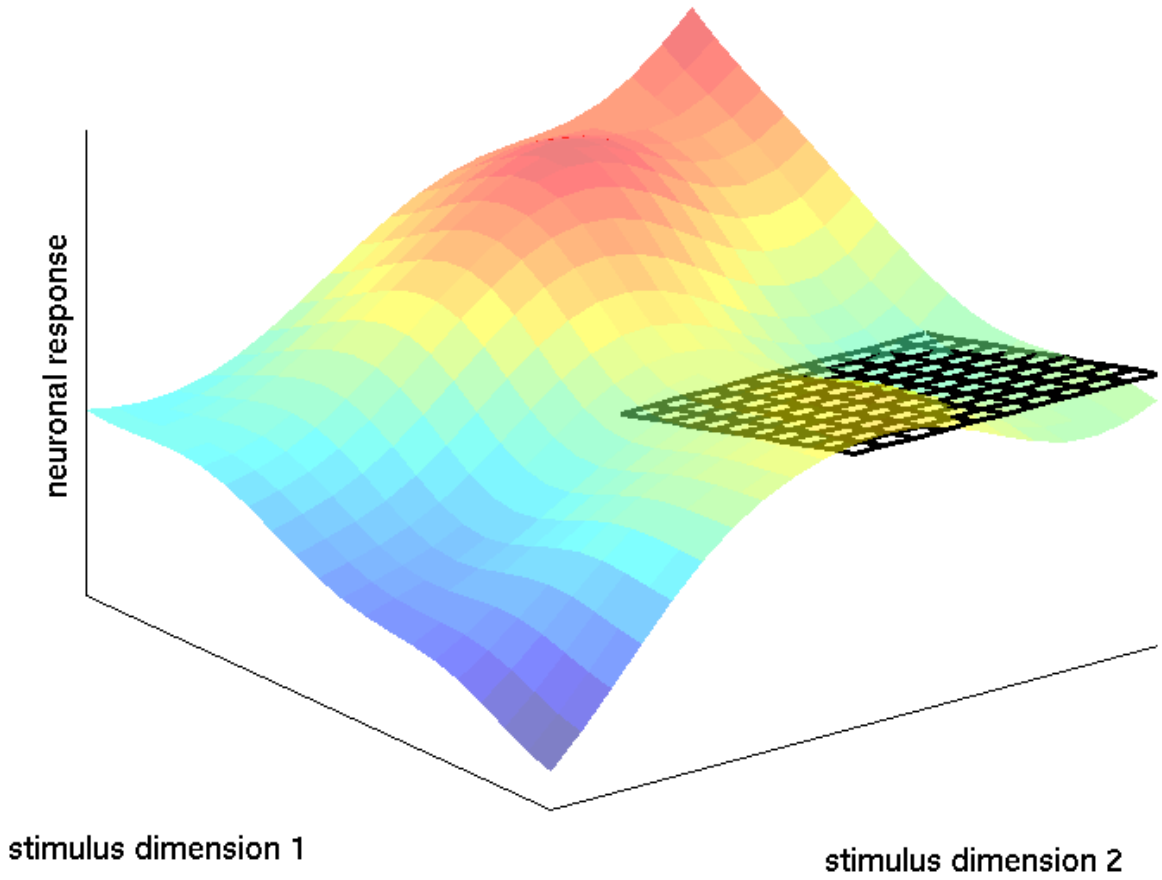
Linear fits to non-linear functions



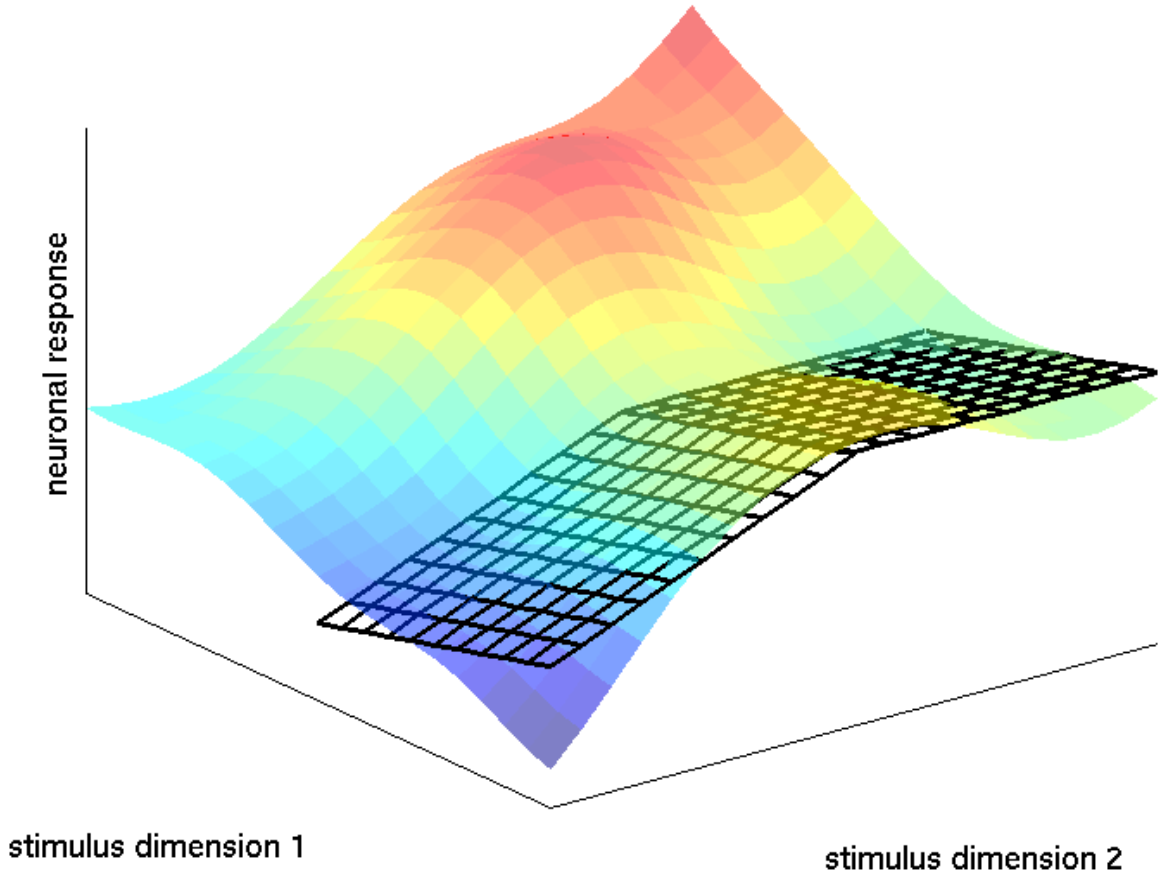
Linear fits to non-linear functions



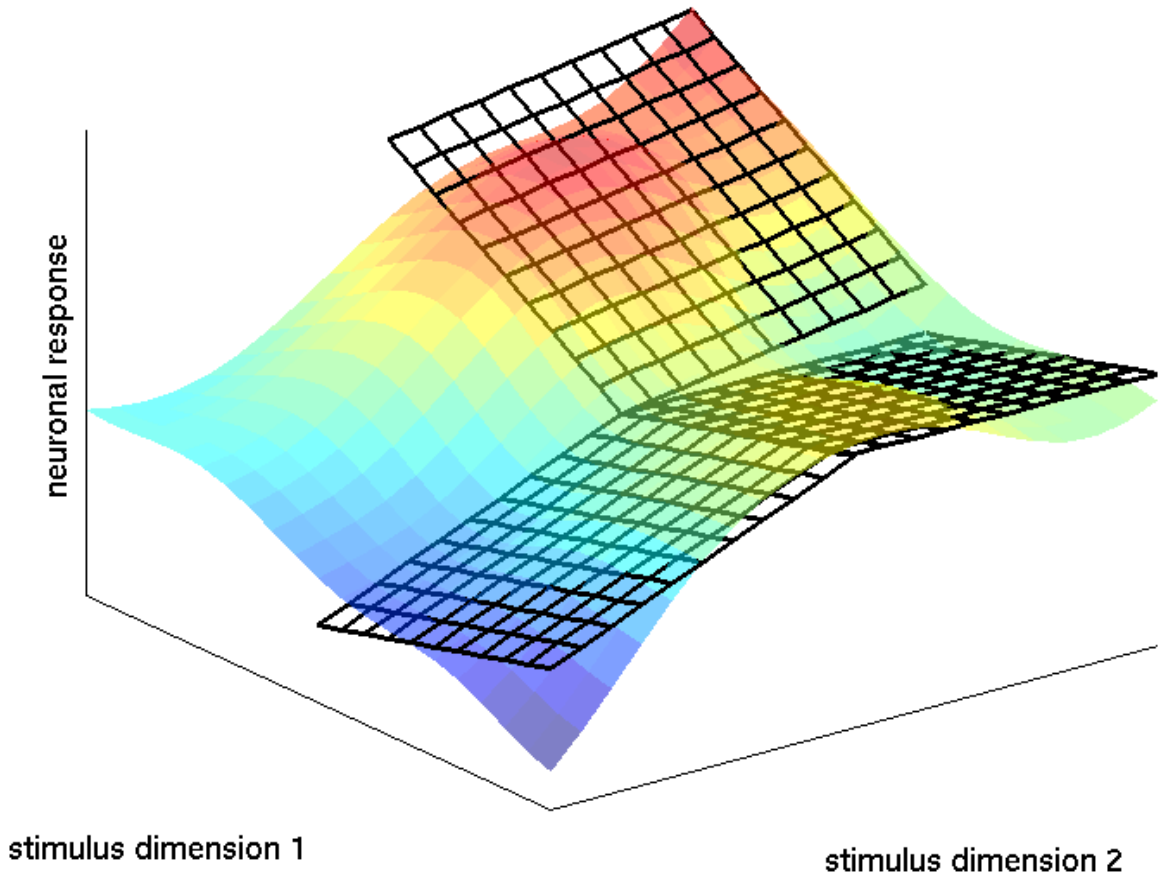
Approximations are stimulus dependent



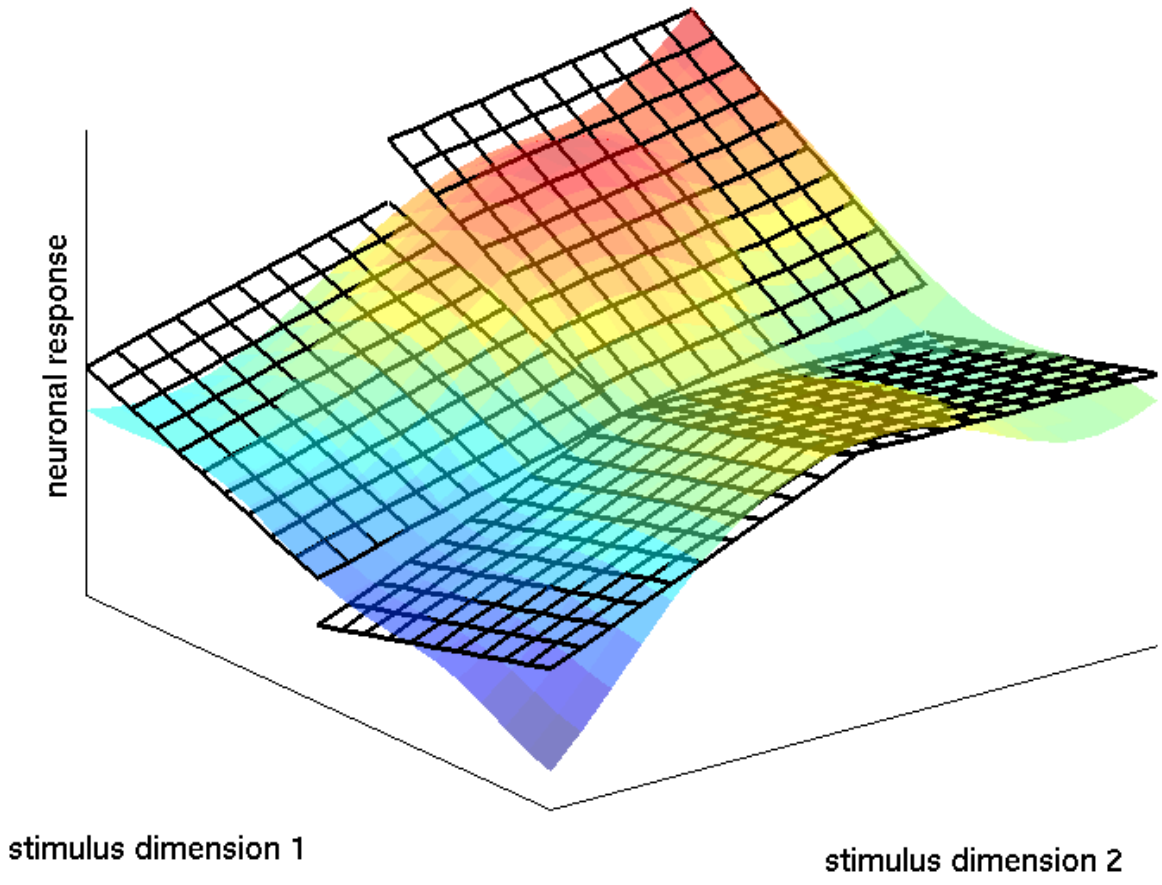
Approximations are stimulus dependent



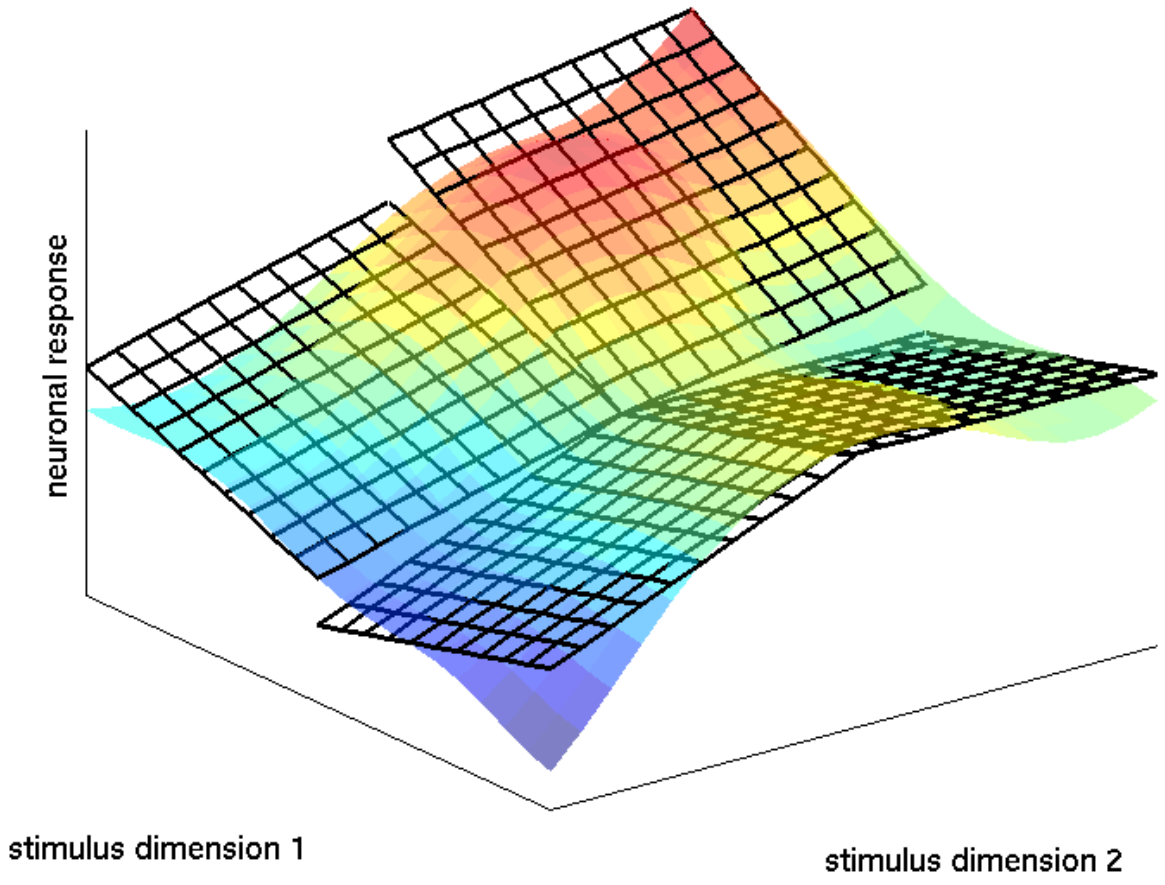
Approximations are stimulus dependent



Approximations are stimulus dependent



Approximations are stimulus dependent



(Stimulus dependence does not always signal response adaptation)

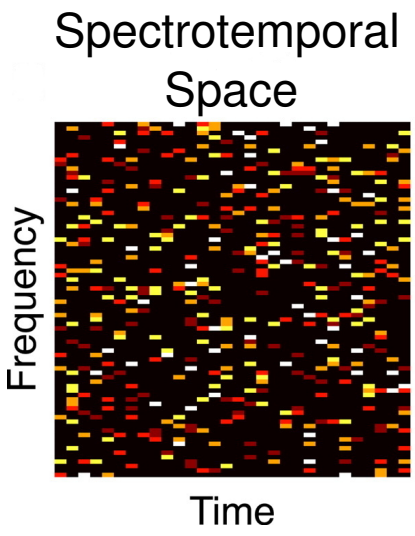
Consequences

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

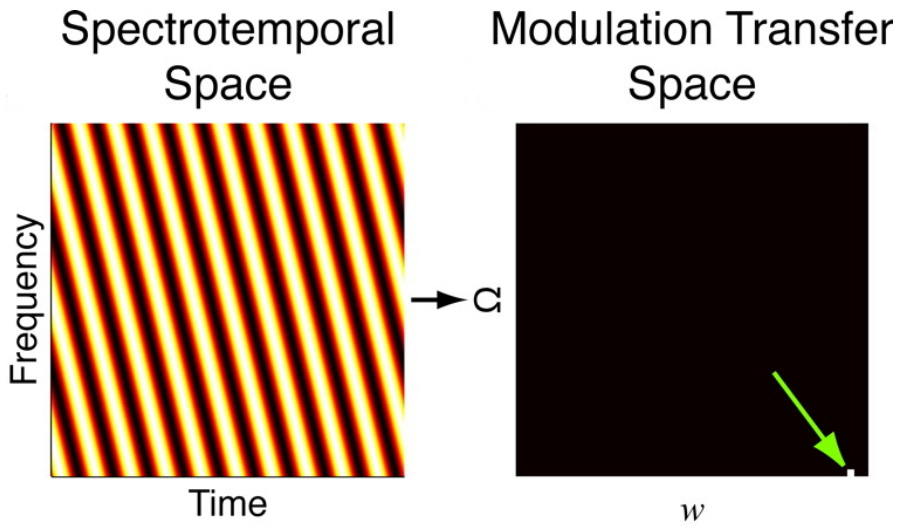
“Independently distributed” stimuli

Knowing stimulus power at any set of points in analysis space provides noinformation about stimulus power at any other point.

DRC:

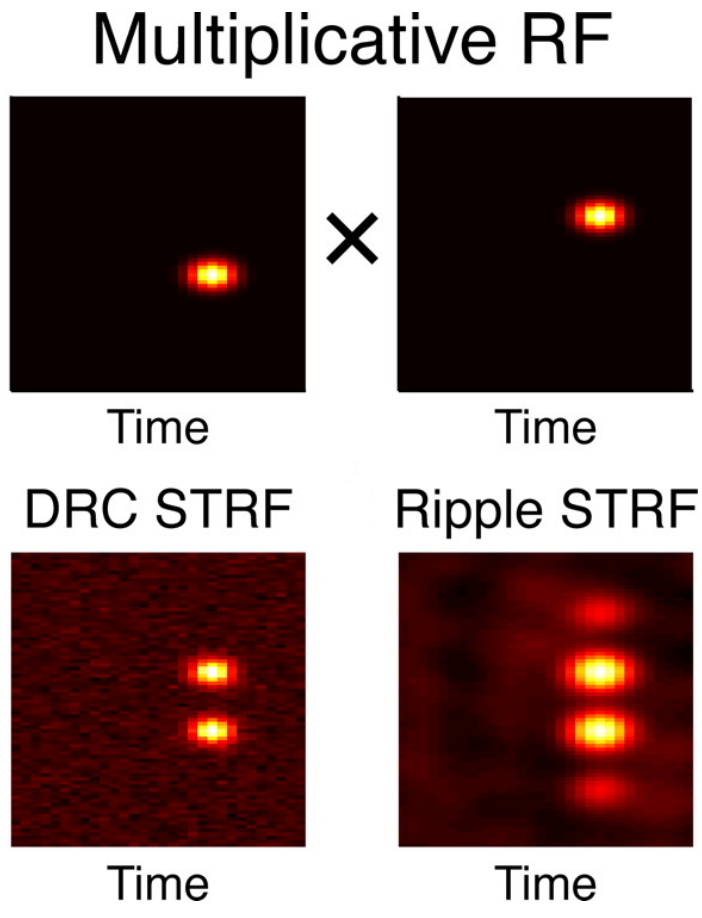


Ripple:



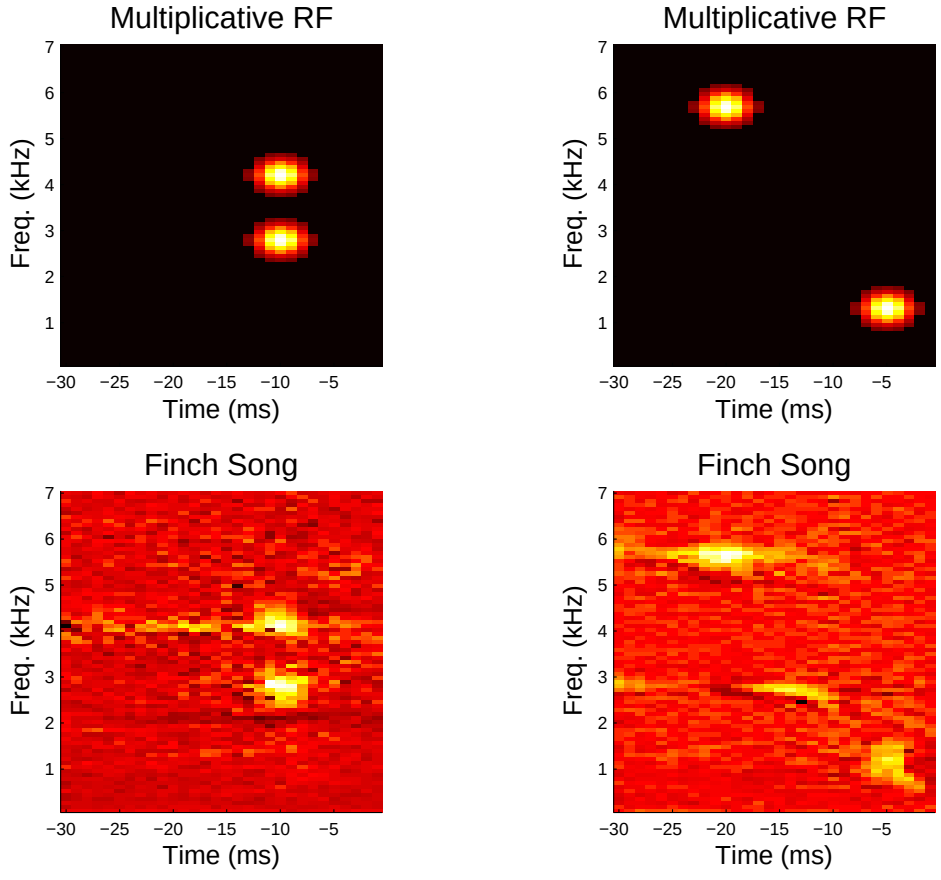
Independence is a property of stimulus *and* analysis space.

Nonlinearity & non-independence distort RF estimates



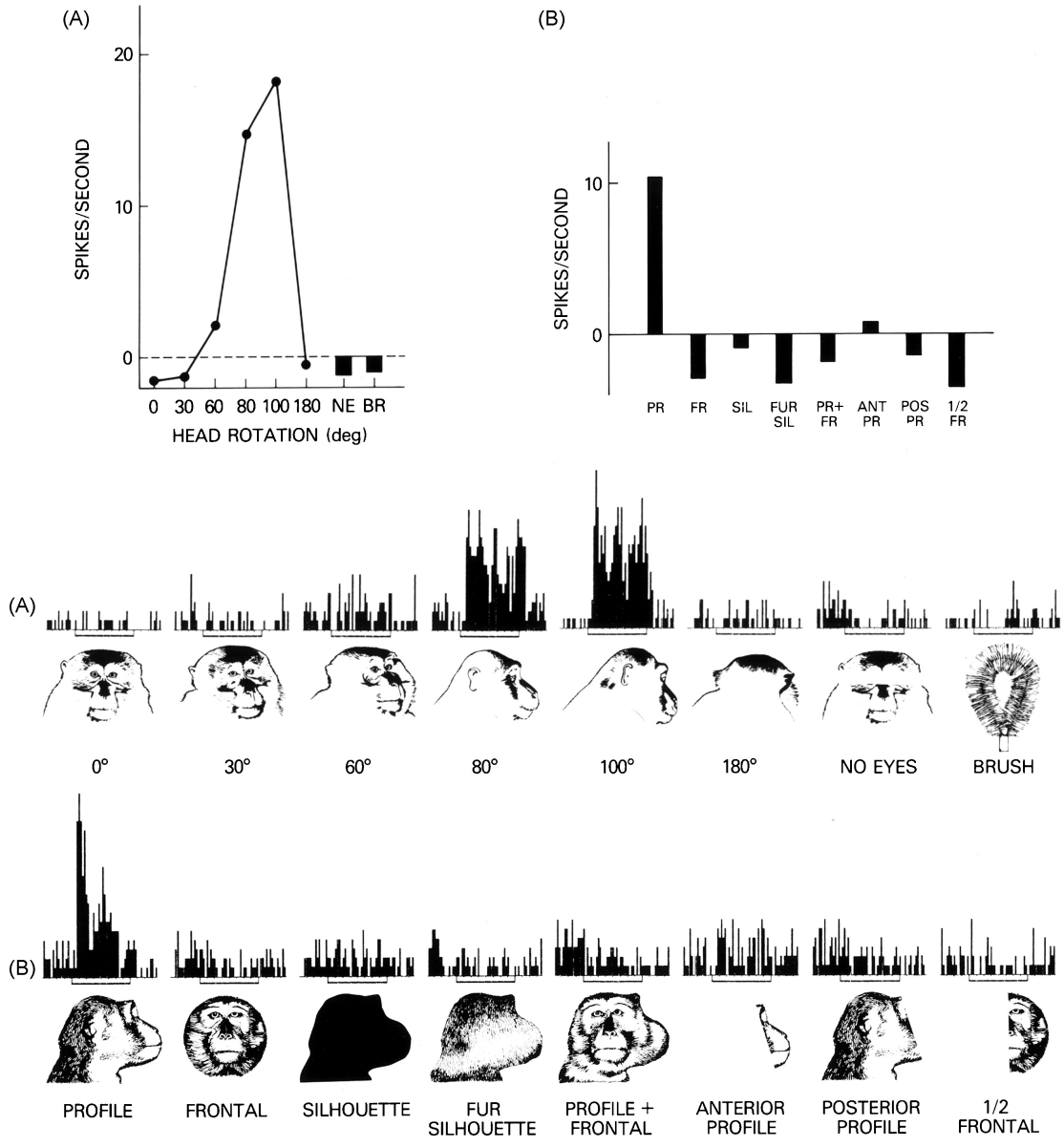
Stimulus may have higher-order correlations in other analysis spaces
— interaction with nonlinearities can produce misleading “receptive fields.”

What about natural sounds?



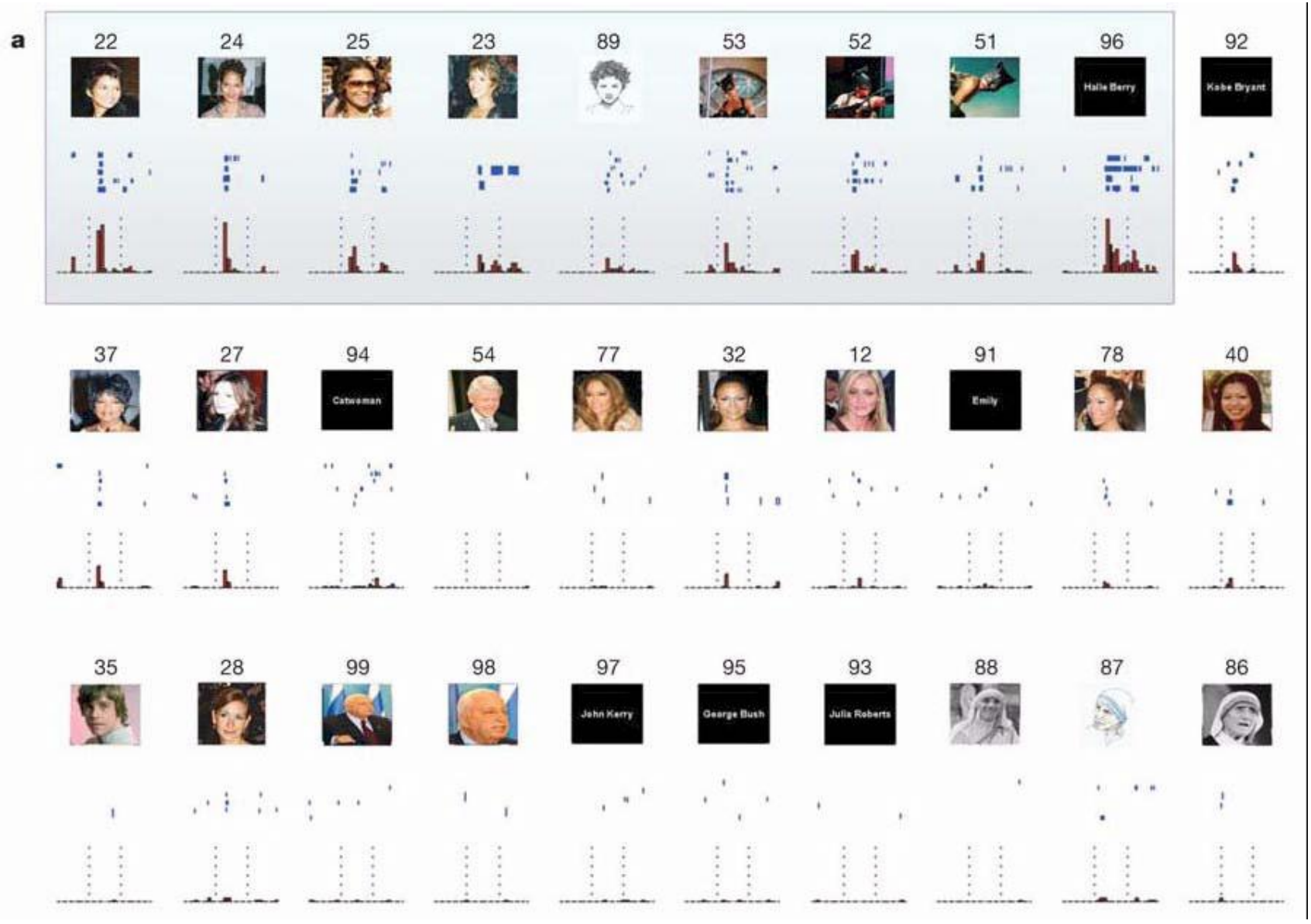
Usually not independent in any space — so STRFs may not be conservative estimates of receptive fields.

Issues: complex selectivity

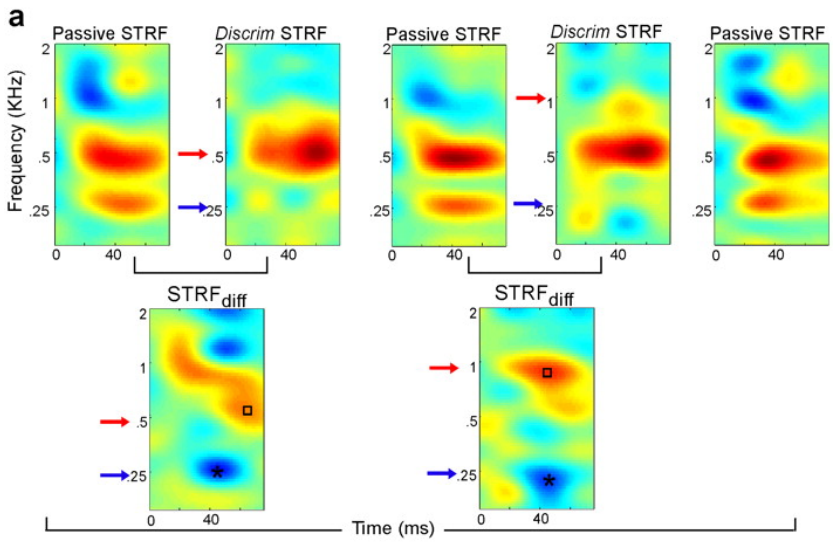


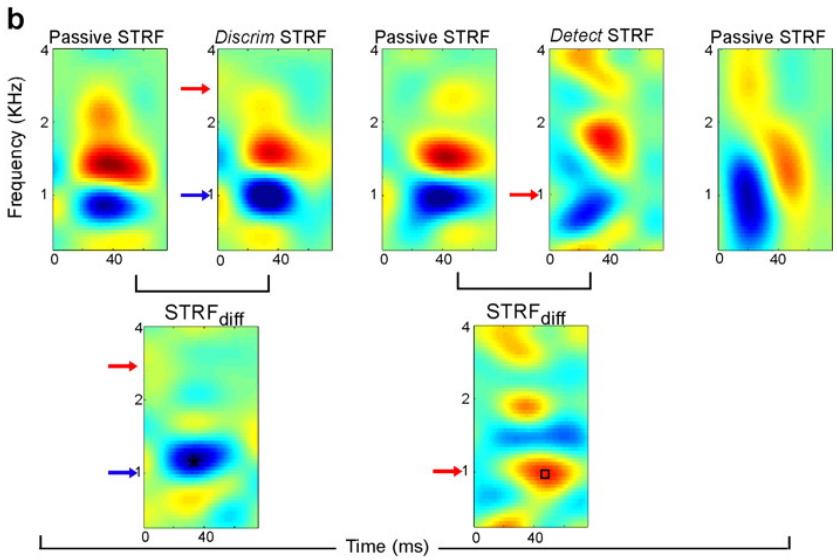
Issues: complex selectivity

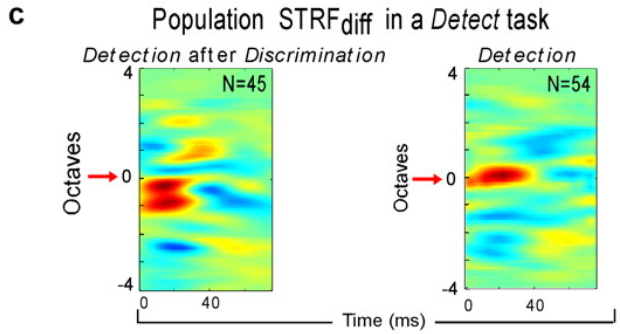
Issues: complex selectivity



Issues: adaptation, task-dependence







The “agnostic” coding approach can only take us so far. Eventually, we need solid scientifically (and probably theoretically) motivated hypotheses.