

# Mathematical Tools for Neural and Cognitive Science

Fall semester, 2024

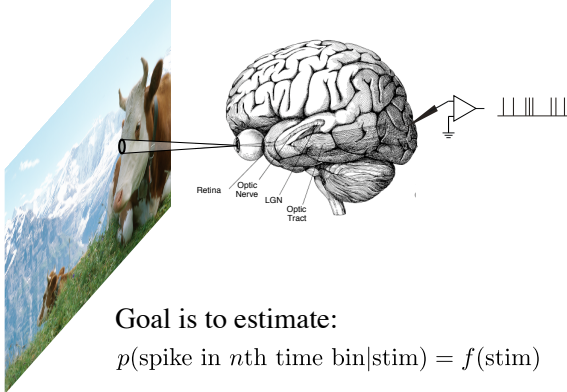
Section 6a:

Fitting simple neural models

## Fitting models to data

- How do we estimate parameters?
  - formulate model + objective function (common choice: ML)
  - optimize (closed form, gradient descent, etc)
- How good are parameter estimates?
- How well does model fit ?
  - likelihood or posterior comparisons
  - model failures

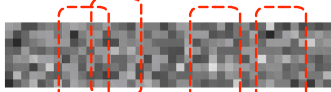
Example: modeling response of a sensory neuron



## Geometric view

1D stimulus over time  
(e.g., flickering bars)

Stimulus



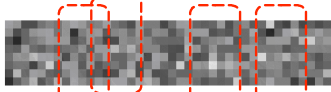
Response

time →

- 8 x 6 stimulus block  
= 48-dimensional vector

## Geometric picture

Stimulus



Response

time →

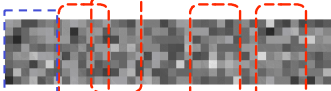
S<sub>1</sub>

S<sub>2</sub>

- non-spiking stimuli
- spiking stimuli

## Geometric picture

Stimulus



Response

time →

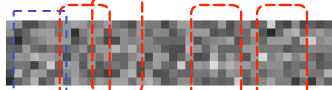
S<sub>1</sub>

S<sub>2</sub>

- non-spiking stimuli
- spiking stimuli

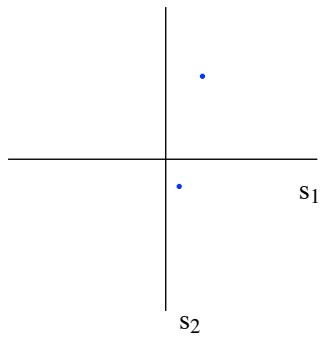
### Geometric picture

Stimulus



Response

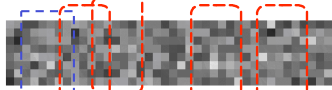
time →



- non-spiking stimuli
- spiking stimuli

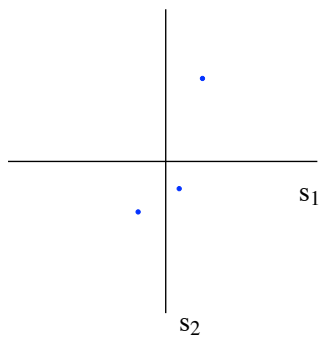
### Geometric picture

Stimulus



Response

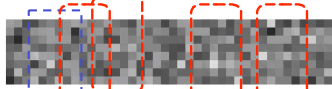
time →



- non-spiking stimuli
- spiking stimuli

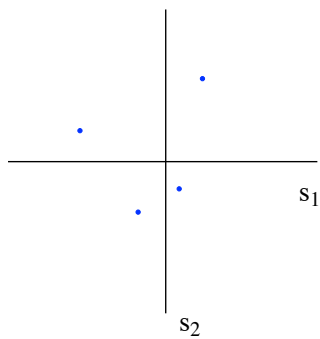
### Geometric picture

Stimulus



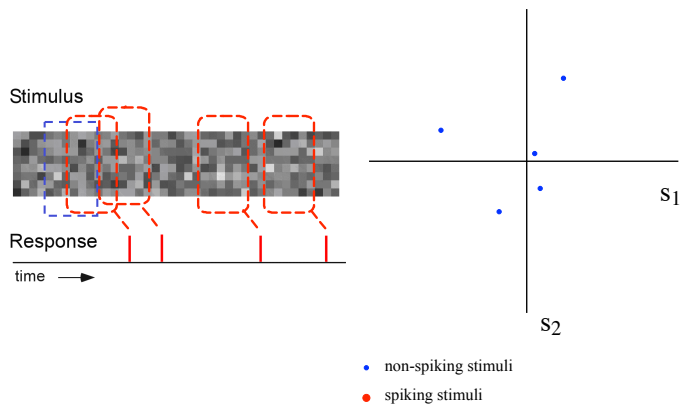
Response

time →

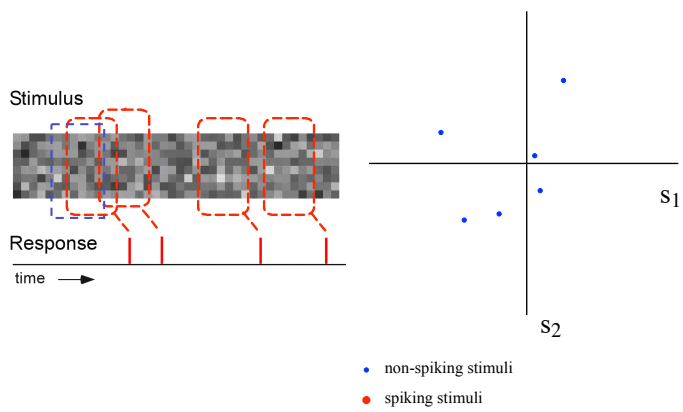


- non-spiking stimuli
- spiking stimuli

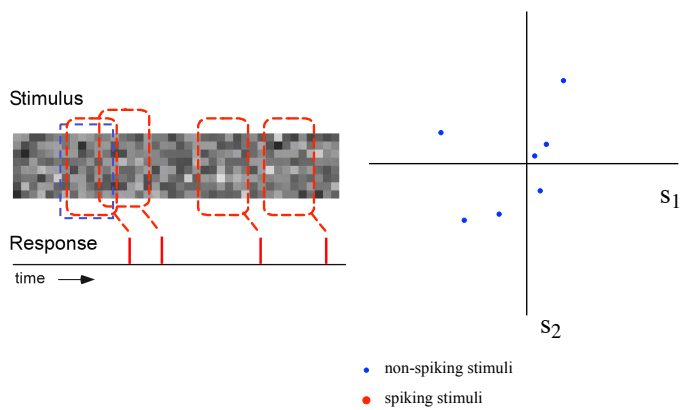
### Geometric picture



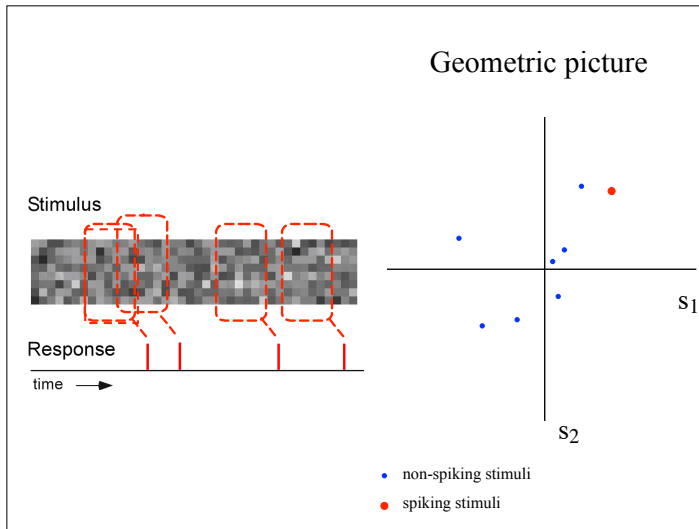
### Geometric picture



### Geometric picture








---

---

---

---

---

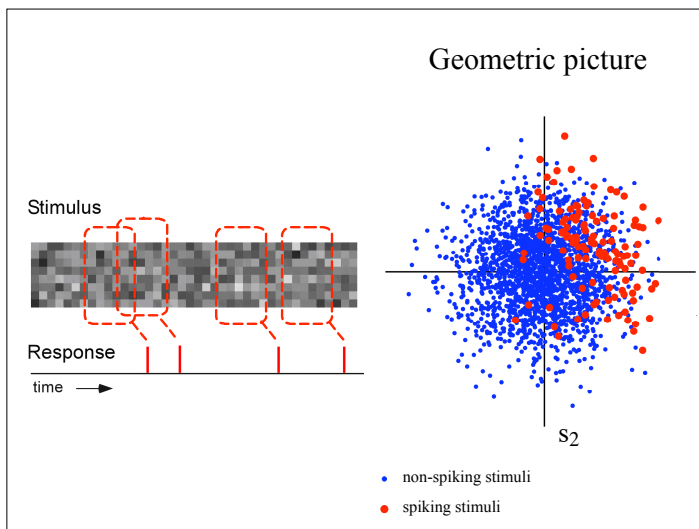
---

---

---

---

---




---

---

---

---

---

---

---

---

---

---

Response is captured by relationship between the distribution of red points (spiking stim) and blue+red points (all stim), expressed in terms of Bayes' rule:

$$P(\text{spike}|\vec{s}) = \frac{P(\vec{s}|\text{spike})P(\text{spike})}{P(\vec{s})}$$

Cannot estimate directly ("curse of dimensionality").  
We need a **model**

$S_1$

$S_2$

---

---

---

---

---

---

---

---

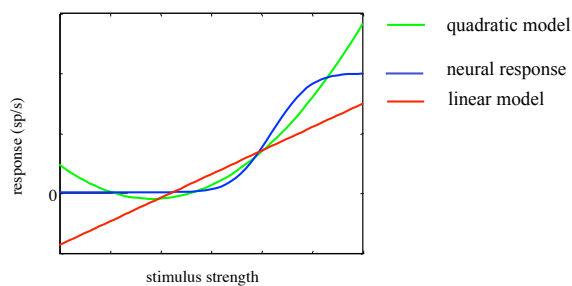
---

---

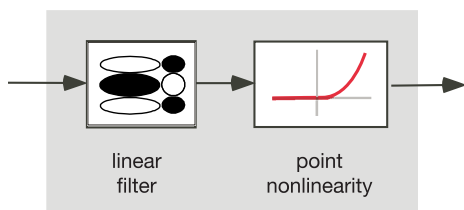
## Some tractable model options

- **Low-order polynomial** [Volterra '13; Wiener '58; DeBoer and Kuyper '68; ...]
- **Low-dimensional subspace** [Bialek '88; Brenner et al '00; Schwartz et al '01; Touryan and Dan '02; ...]
- **Recursive linear with exponential nonlinearity** [Truccolo et al '05; Pillow et al '05]

## Low-order polynomial model

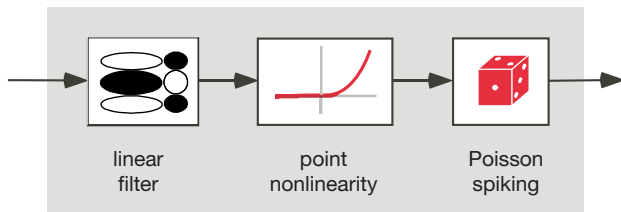


## Example: LN cascade model



- Threshold-like nonlinearity  $\Rightarrow$  linear classifier
- Classic model for Artificial Neural Networks
  - McCullough & Pitts (1943), Rosenblatt (1957), etc
- No spikes (output is firing rate)

## LNP cascade model



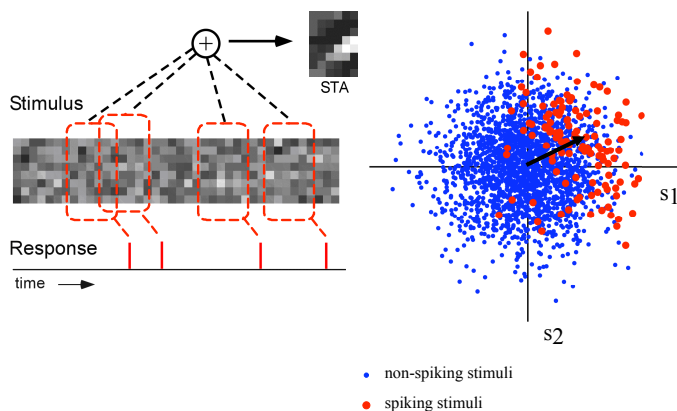
- Simplest descriptive spiking model
- Easily fit to (extracellular) data
- Descriptive, and interpretable (although *not* mechanistic)

## Simple LNP fitting

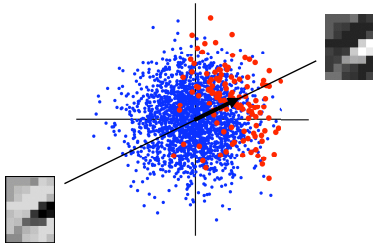
- Assuming:
  - stochastic stimuli, spherically distributed
  - average of spike-triggered ensemble (STA) is shifted from that of raw ensemble
- The STA (i.e., linear regression!) gives an **unbiased** estimate of  $w$  (for any  $f$ ). [on board]
- For exponential  $f$ , this is the ML estimate! [on board]

[Bussgang 52; de Boer & Kuyper 68]

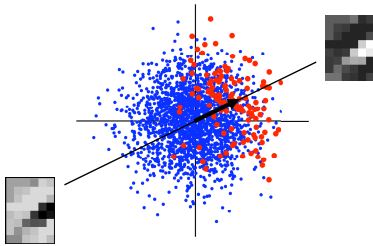
## Computing the STA



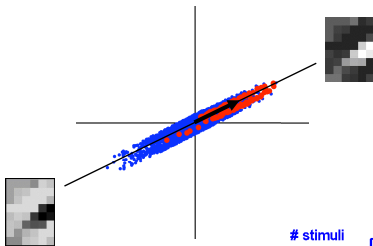
STA corresponds to a “direction” in stimulus space



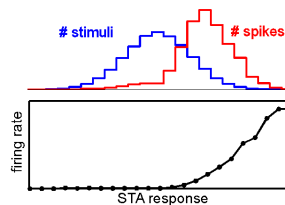
Projecting onto the STA



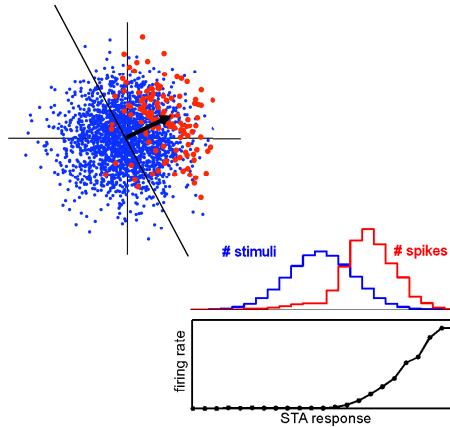
Solving for nonparametric nonlinearity



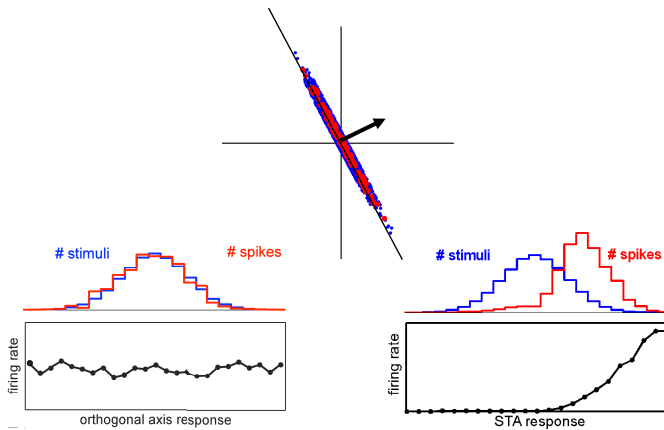
$$P(\text{spike}|\vec{s}) = \frac{P(\vec{s}|\text{spike})P(\text{spike})}{P(\vec{s})}$$



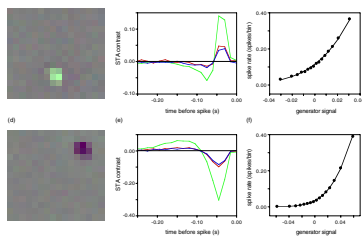
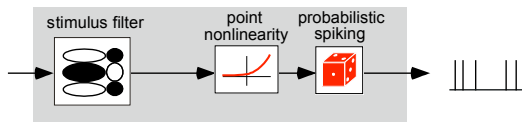
## Projecting onto an axis orthogonal to the STA



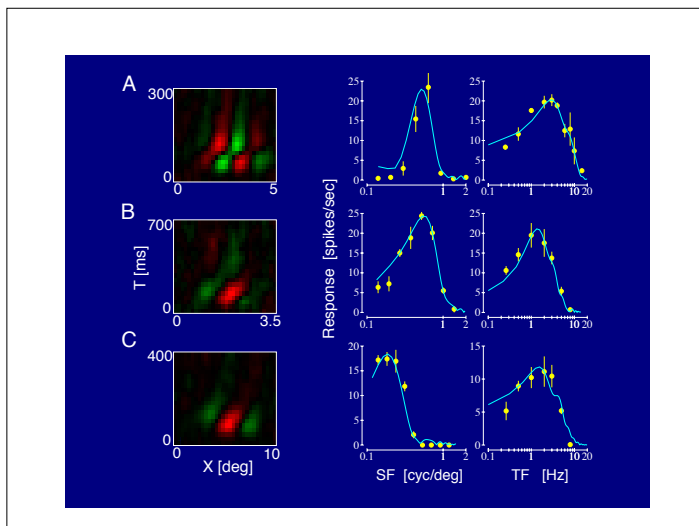
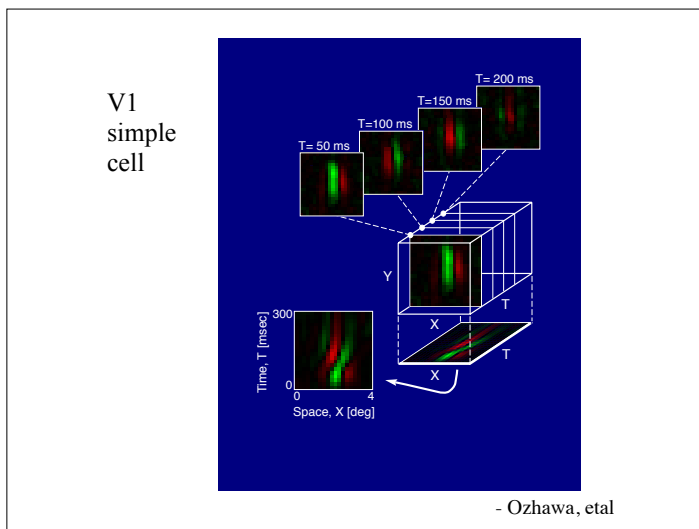
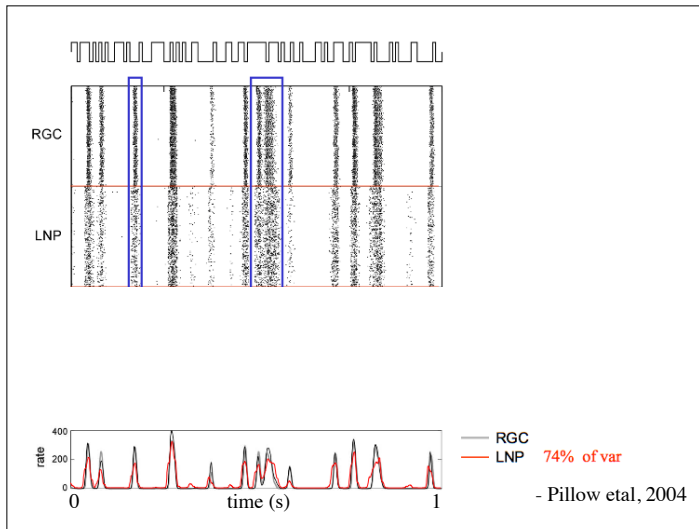
## Projecting onto an axis orthogonal to the STA



## LNP model, fit to retinal ganglion cells

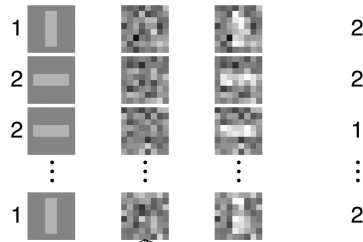


[Chichilnisky & Kalmer, 2002]



## Psychophysical “Classification Image”

(a) signal + noise = stimulus → response

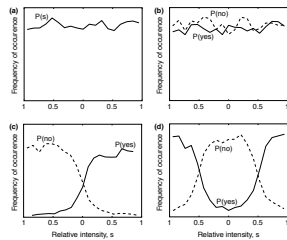
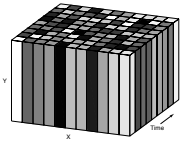


(b)

$$(\bar{n}^{12} + \bar{n}^{22}) - (\bar{n}^{11} + \bar{n}^{21}) = c$$

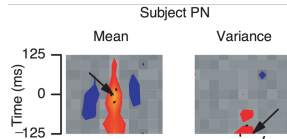
Murray R F J Vis 2011;11:2

Stimuli: 11x9  
movie of bars,  
uniform random  
intensities



Simulation:  
a) raw stimulus  
distribution  
b) cond. dist. for  
irrelevant bar  
c) cond. dist. for  
linear response  
model  
d) cond. dist. for  
quadratic (contrast)  
response

Task:  
Is center bar of  
middle frame  
brighter or darker  
than the mean?



[Neri & Heeger, 2002]

## ML estimation of LNP

If  $f_{\theta}(\vec{k} \cdot \vec{x})$  is convex (in argument and theta),  
and  $\log f_{\theta}(\vec{k} \cdot \vec{x})$  is concave,  
the likelihood of the LNP model is convex  
(for all data,  $\{n(t), \vec{x}(t)\}$ )

Examples:  $e^{\vec{k} \cdot \vec{x}(t)}$

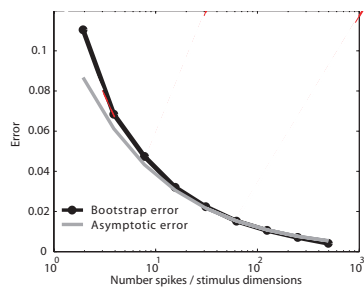
$$(\vec{k} \cdot \vec{x}(t))^{\alpha}, \quad 1 < \alpha < 2$$

[Paninski, '04]

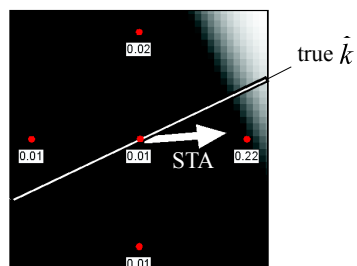
## Sources of STA estimation error

- Finite data (convergence goes as  $1/N$ )
- Non-spherical stimuli (estimator can be biased)
- Model failures

## Variance behavior of STA

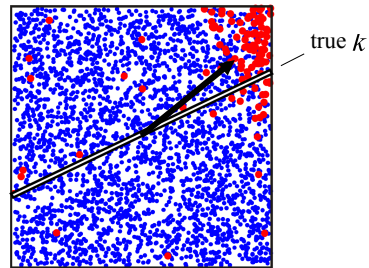


Example 1:  
“sparse” noise





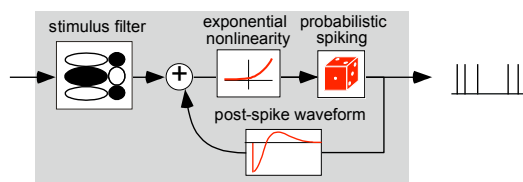
Example 2:  
uniform noise



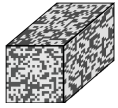
## LNP model failures (& solutions)

- Neural response depends on spike history  
=> introduce spike history feedback
- Symmetric nonlinearities and/or multi-dimensional front-end (e.g., V1 complex cells)  
=> spike-triggered covariance, subspace analyses
- White noise doesn't drive mid- to late-stage neurons well  
=> cascade LNP on top of an "afferent" model

## Recursive LNP

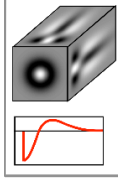


stimulus & spike train

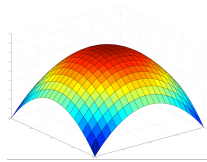


maximize  
likelihood

model  
parameters

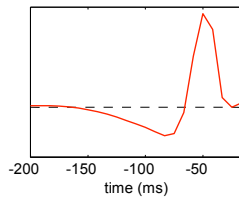


Critical: Likelihood function has no  
local maxima [Paninski 04]

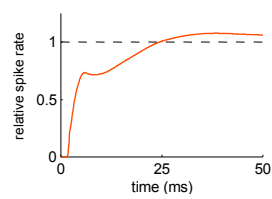


## Example ON cell

temporal filter



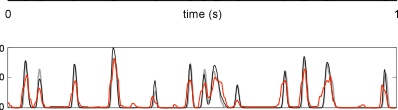
post-spike waveform



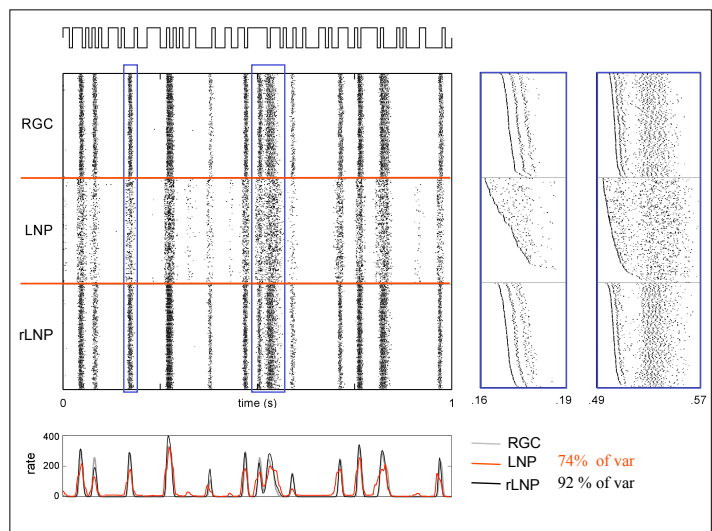
RGC

LNP

rLNP



— RGC  
— LNP 74% of var  
— rLNP 92 % of var




---

---

---

---

---

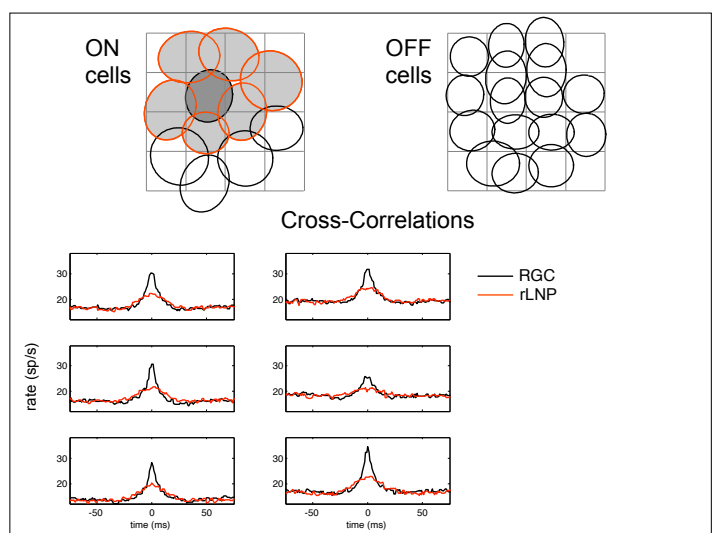
---

---

---

---

---




---

---

---

---

---

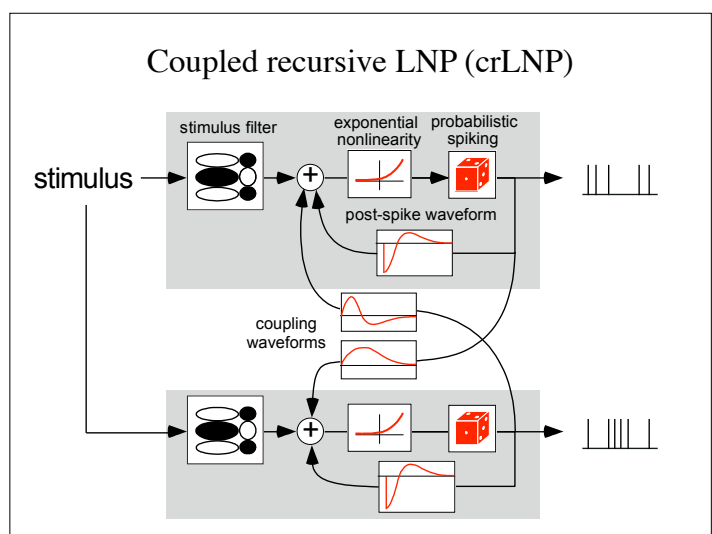
---

---

---

---

---




---

---

---

---

---

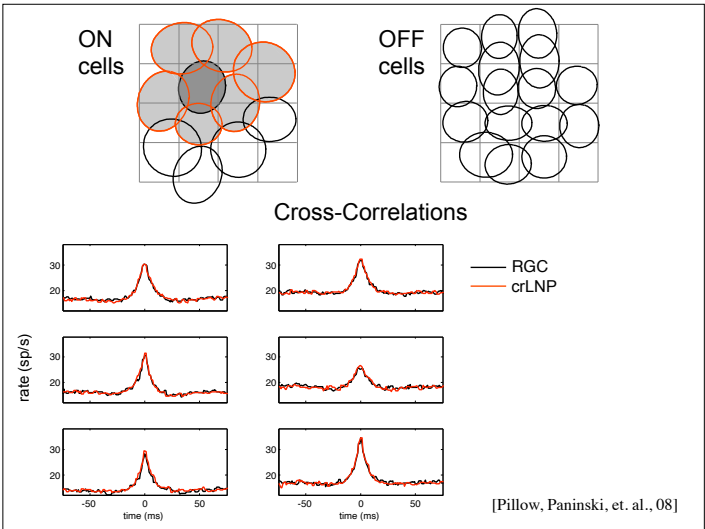
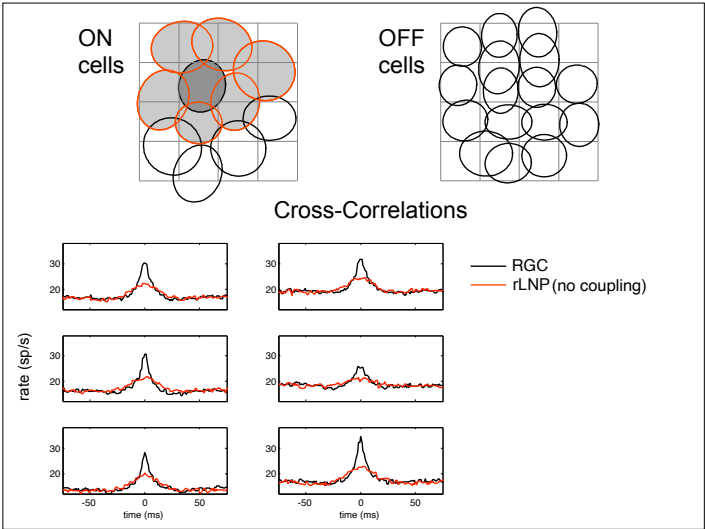
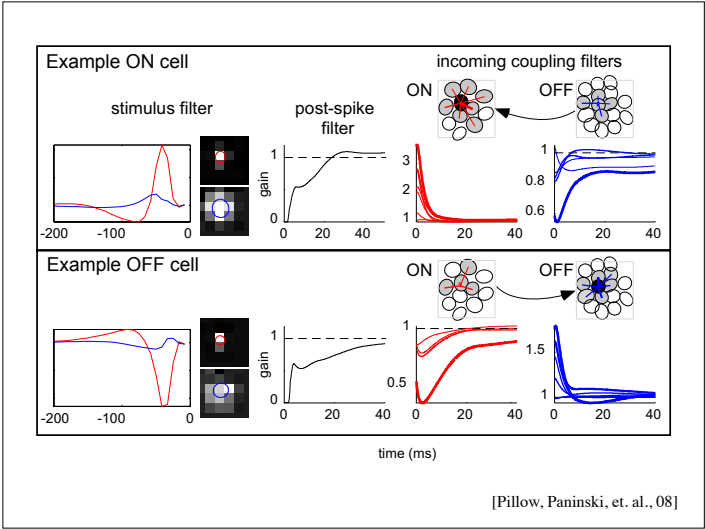
---

---

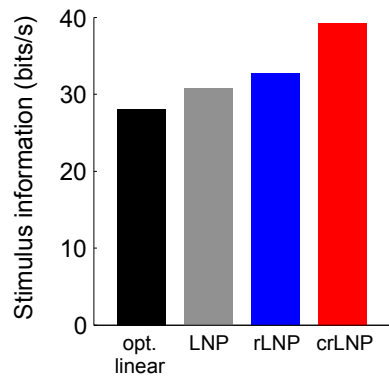
---

---

---

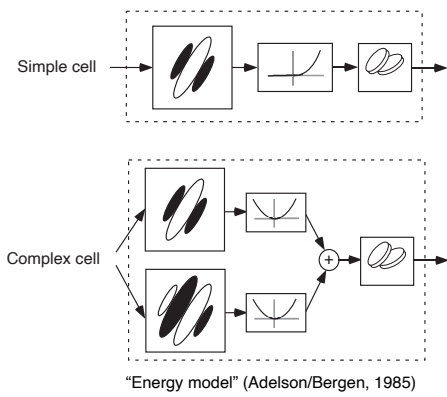


## Decoding

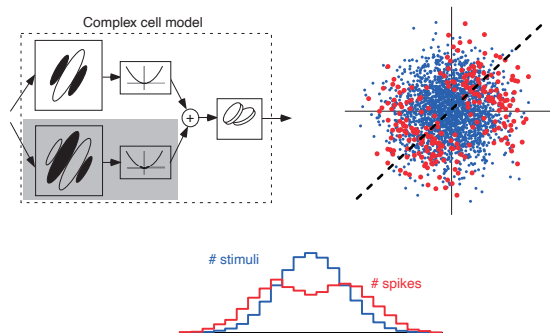


[Pillow, Paninski, et. al., 08]

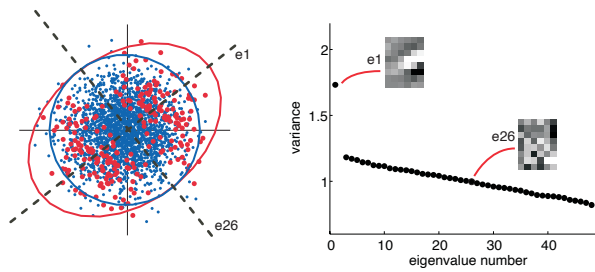
## Classic V1 models



## STA failure on complex cell

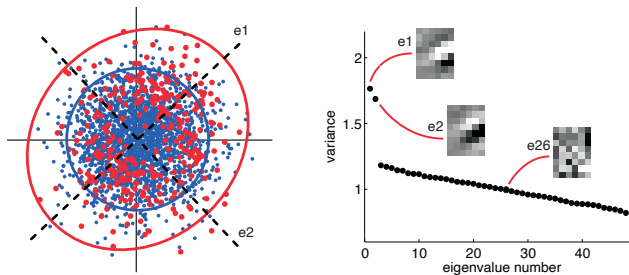


## Spike-triggered covariance (STC)



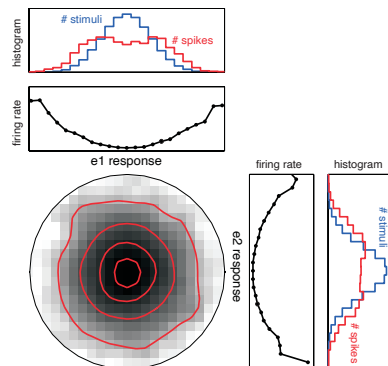
[Bialek '88; Brenner et al '00; Schwartz et al '01; Touryan and Dan '02; ...]

## STC on complex cell (simulation)

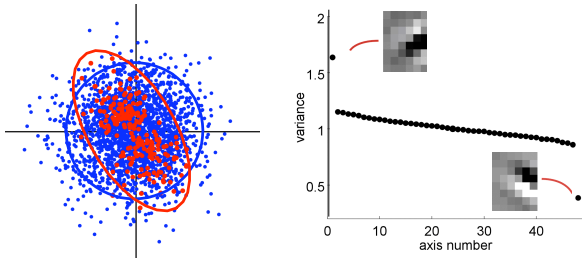


Use STC to find subspace that modulates response

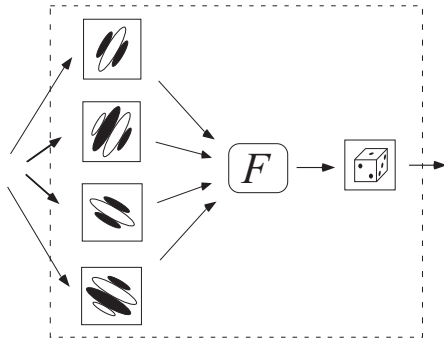
## STC on complex cell (simulation)



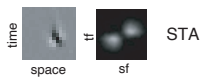
## STC: suppressive filters



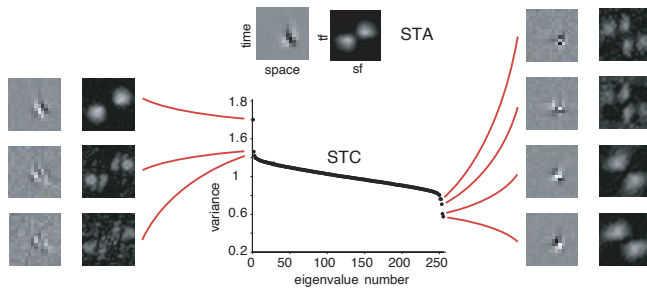
## Subspace LNP model



## V1 simple cell

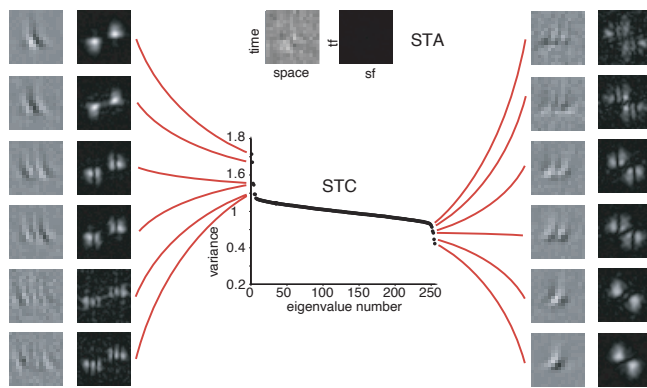


## V1 simple cell

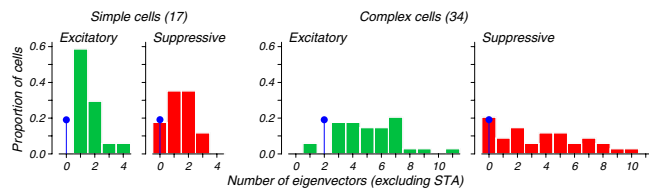


[Rust, Schwartz, et. al., 2005]

## V1 complex cell

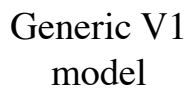
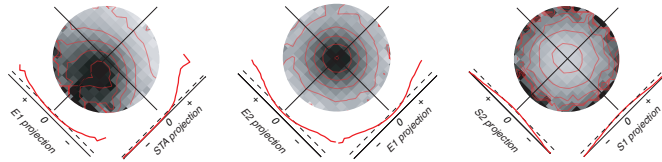
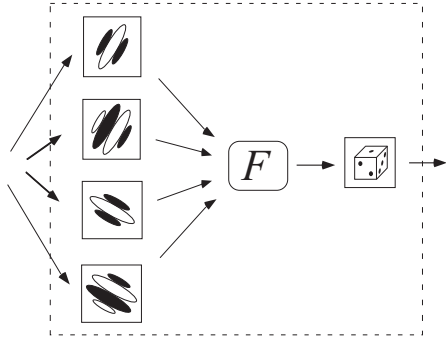


[Rust, Schwartz, et. al., 2005]



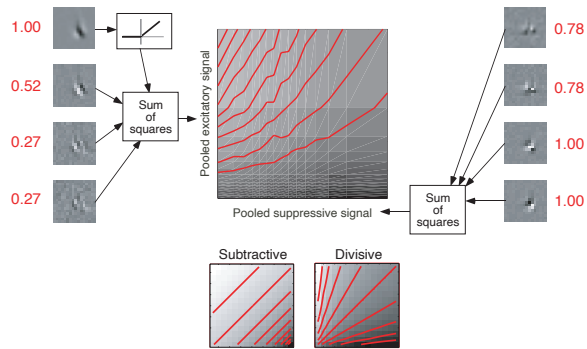
[Rust, Schwartz, et. al., 2005]





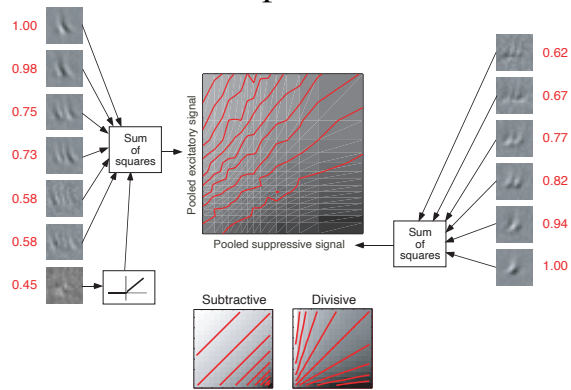
[Rust, Schwartz, et. al., 2005]

## Simple cell

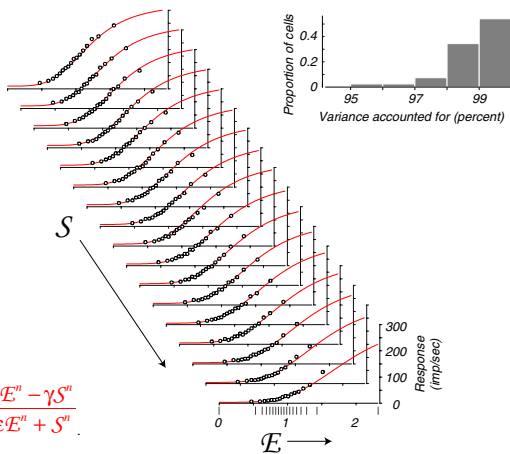


[Rust, Schwartz, et. al., 2005]

## Complex cell



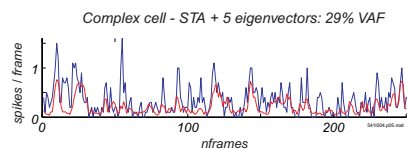
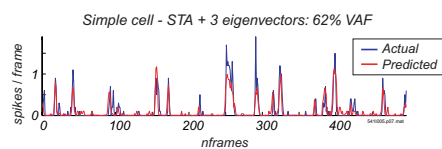
[Rust, Schwartz, et. al., 2005]



$$\frac{\alpha + \beta E^n - \gamma S^n}{\delta^n + \epsilon E^n + S^n}$$

[Rust, Schwartz, et. al., 2005]

## PSTH validation

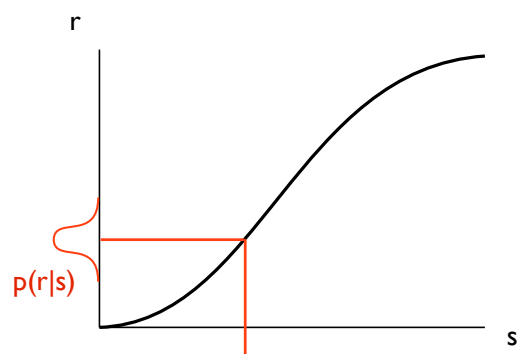


Failure of Poisson spiking assumption

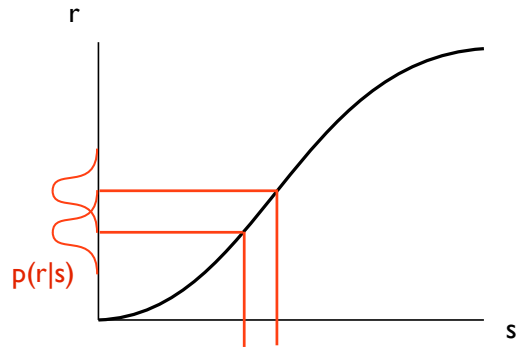
## “Decoding” neural populations?

- Connecting neural response to behavior
- Engineering: Brain-Computer Interfaces
- Test/compare encoding models

## Encoding determines discriminability

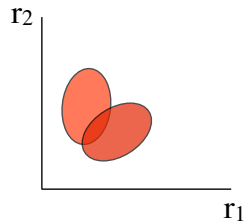


## Probabilistic encoding model determines discriminability



Discriminability ( $d'$ ) is (approximately) slope/stdev

## Two neurons



Same fundamental issues as 1D case:

- Probabilistic encoding determines discriminability
- Intuitively, overlap is distance/spread
- For linear decoding: project onto discrimination axis

## I. Simple/intuitive population decoding

• Linear?  $\hat{s}(\vec{r}) = \sum_n r_n s_n$

(simple, but usually doesn't work well)

• Winner-take-all  $\hat{s}(\vec{r}) = s_m, \quad m = \arg \max_n \{r_n\}$

(simple, but discontinuous and noise-susceptible)

• Population vector [Georgopoulos et.al., 1986]  $\hat{s}(\vec{r}) = \frac{\sum_n r_n s_n}{\sum_n r_n}$

(also simple, more robust)

## II. Statistically optimal decoding

- Maximum likelihood (ML)  $\hat{s}(\vec{r}) = \arg \max_s p(\vec{r}|s)$
- Maximum a posteriori (MAP)  $\hat{s}(\vec{r}) = \arg \max_s p(\vec{r}|s) \cdot p(s)$
- Minimum Mean Squared Error (MMSE),  
a.k.a. Bayes Least Squares (BLS)  $\hat{s}(\vec{r}) = \mathbf{E}(s|\vec{r})$

### ML decoding for a Poisson-spiking neural population

[Ma, Beck, Latham, Pouget, 2006; Jazayeri & Movshon, 2006]

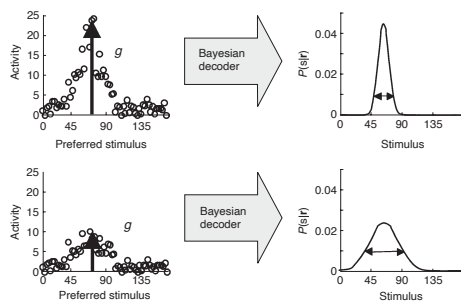
$$p(\vec{r}|s) = \prod_{n=1}^N \frac{h_n(s)^{r_n} e^{-h_n(s)}}{r_n!}$$

$$\log(p(\vec{r}|s)) = \sum_{n=1}^N r_n \log(h_n(s)) - h_n(s) - \log(r_n!)$$

If  $\sum_{n=1}^N h_n(s)$  is constant (i.e., tuning curves “tile”), just minimize the response-weighted sum of log tuning curves.

Special cases allow closed-form solutions:

- Gaussian tuning curves  $h_n(s) = \exp(-(s - s_n)^2 / 2\sigma^2)$
- von Mises tuning curves  $h_n(s) = \exp(\kappa \cos(s - s_n))$

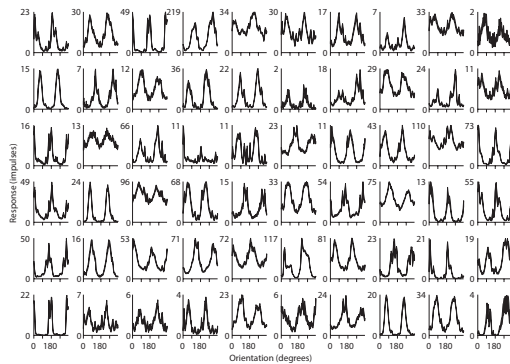


$$\hat{s}(\vec{r}) = \frac{\sum_n r_n s_n}{\sum_n r_n} \quad \hat{\sigma}(\vec{r}) = \frac{\sigma_{TC}}{\sqrt{\sum_n r_n}}$$

[Ma, Beck, Latham & Pouget, 06]

## Population Decoding

The data: tuning curves  $f_i$



[Graf, Kohn, Jazayeri & Movshon, 11]

## Comparing population decoders

1) The ML decoder, assuming independent Poisson responses (the PID):

$$\begin{aligned} \log L(\theta) &= \log \left( \prod_{i=1}^N p(r_i | \theta) \right) = \sum_{i=1}^N \log \left( \frac{f_i(\theta)^{r_i}}{r_i!} \exp(-f_i(\theta)) \right) \\ &= \sum_{i=1}^N \log(f_i(\theta)) r_i - \sum_{i=1}^N f_i(\theta) - \sum_{i=1}^N \log(r_i!) = \sum_{i=1}^N W_i(\theta) r_i + B(\theta) \end{aligned}$$

For discrimination between two values, likelihood ratio is *linear* function of responses:

$$\begin{aligned} \log LR(\theta_1, \theta_2) &= \log \left( \frac{L(\theta_1)}{L(\theta_2)} \right) = \log L(\theta_1) - \log L(\theta_2) \\ &= \sum_{i=1}^N [W_i(\theta_1) - W_i(\theta_2)] r_i + [B(\theta_1) - B(\theta_2)] \\ &= \sum_{i=1}^N w_i(\theta_1, \theta_2) r_i + b(\theta_1, \theta_2) \end{aligned}$$

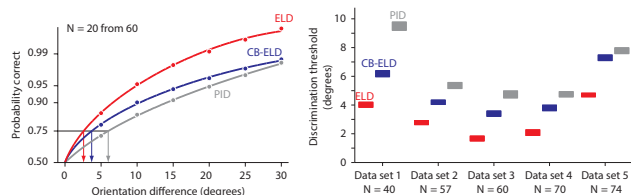
[Graf, Kohn, Jazayeri & Movshon, 11]

## Comparing population decoders

2) Alternatively, compute an SVM on the measured response vectors for each orientation, the empirical linear decoder (ELD):

$$y(\theta_1, \theta_2) = \sum_{i=1}^N w_i(\theta_1, \theta_2) r_i + b(\theta_1, \theta_2)$$

3) For each neuron and orientation, shuffle the responses across trials and train a new SVM, the correlation-blind empirical linear decoder (CB-ELD).



[Graf, Kohn, Jazayeri & Movshon, 11]

## Where we've been...

- Linear algebra / linear systems
  - Ex: Trichromacy
- Least squares
  - regression / PCA
- Linear shift-invariant systems
  - convolution / Fourier transforms
  - Ex: Auditory filtering
- Summary statistics - dispersion, central tendency
- Statistical inference
  - estimation, bias, variance, convergence
  - maximum likelihood estimator (MLE), MAP, Bayes
  - Ex: signal detection theory
  - Classification, clustering
- Model fitting
  - model comparison, overfitting, regularization, cross-validation
  - Ex: fitting an LNP model
  - Ex: population decoding

