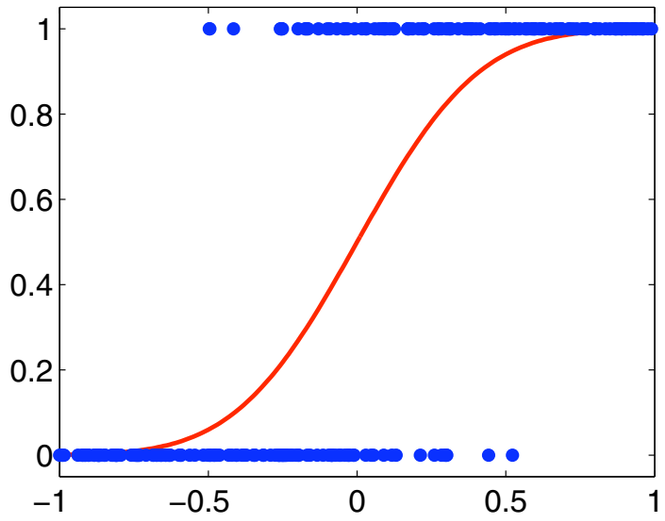
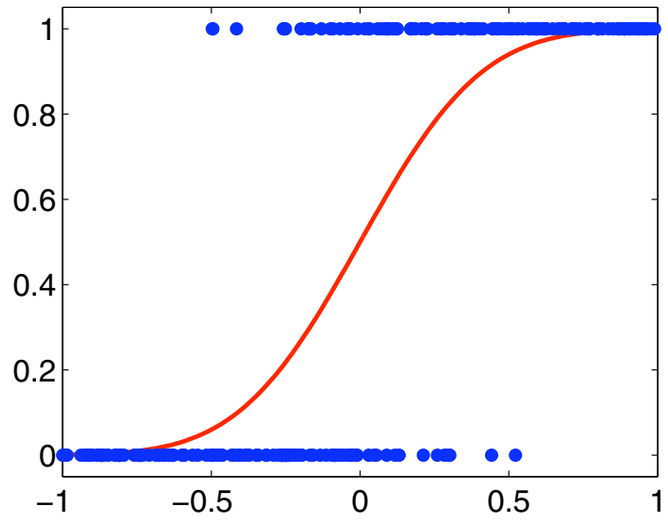
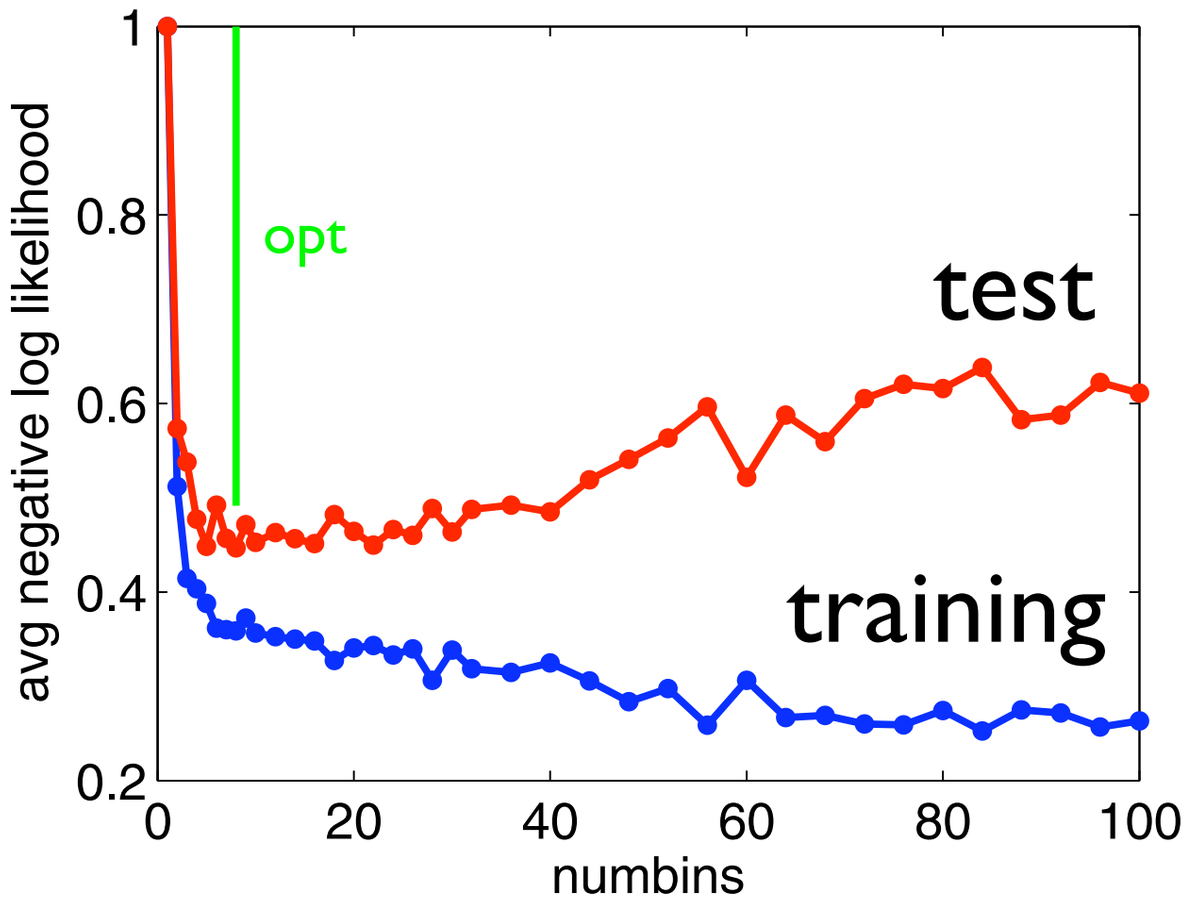
Raw data:

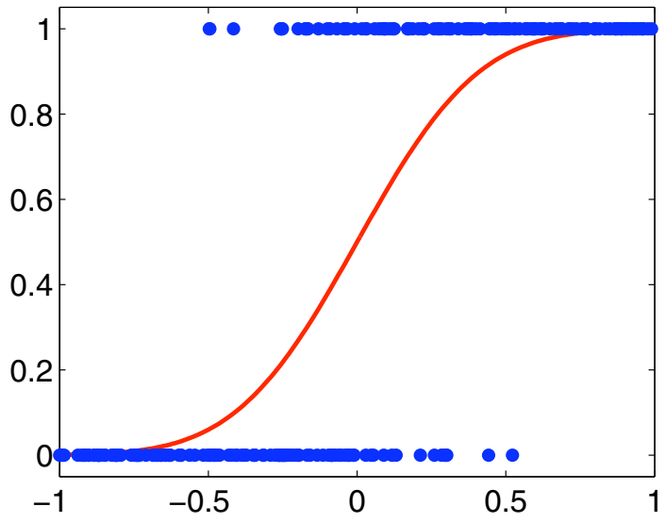


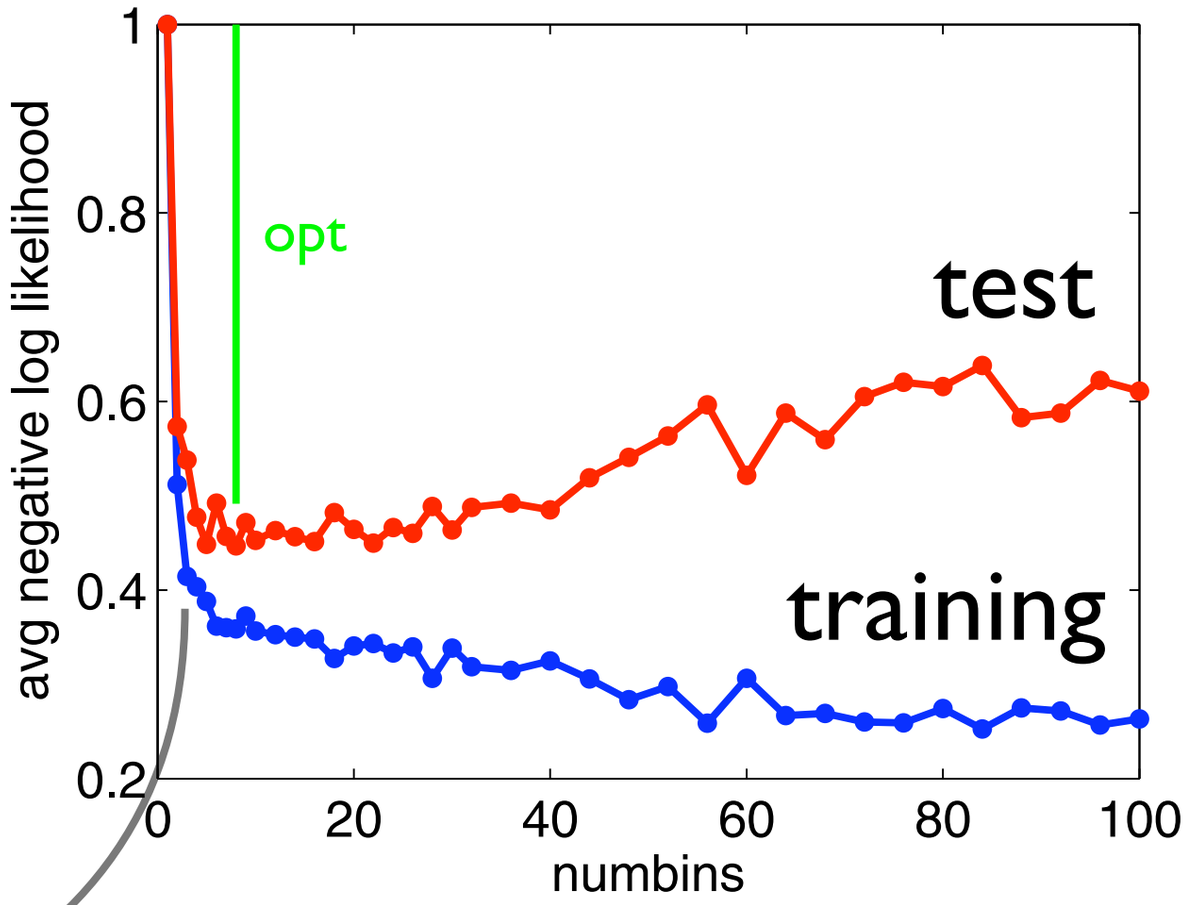
Raw data:


1
0.8
0.6
0.4
0.2
0
−1
−0.5
0
0.5
1

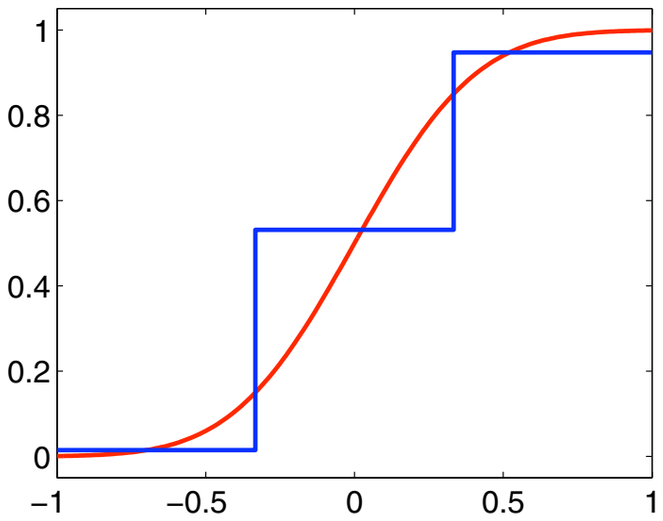

avg negative log likelihood
1
0.8
0.6
0.4
0.2
0
20
40
60
80
100
opt
test
training
numbins

Raw data:

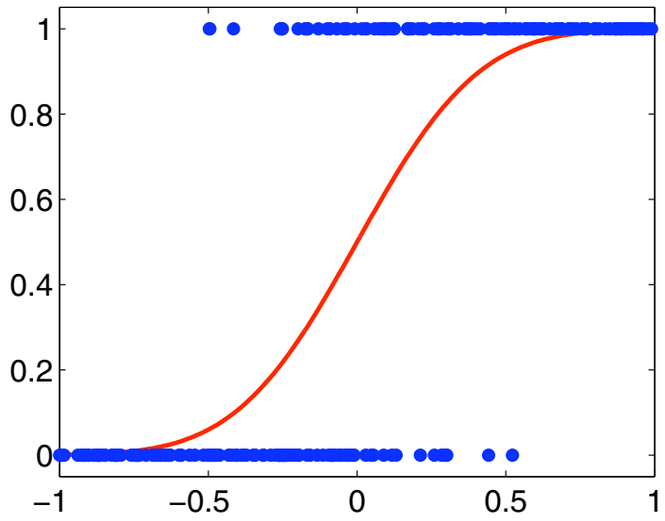


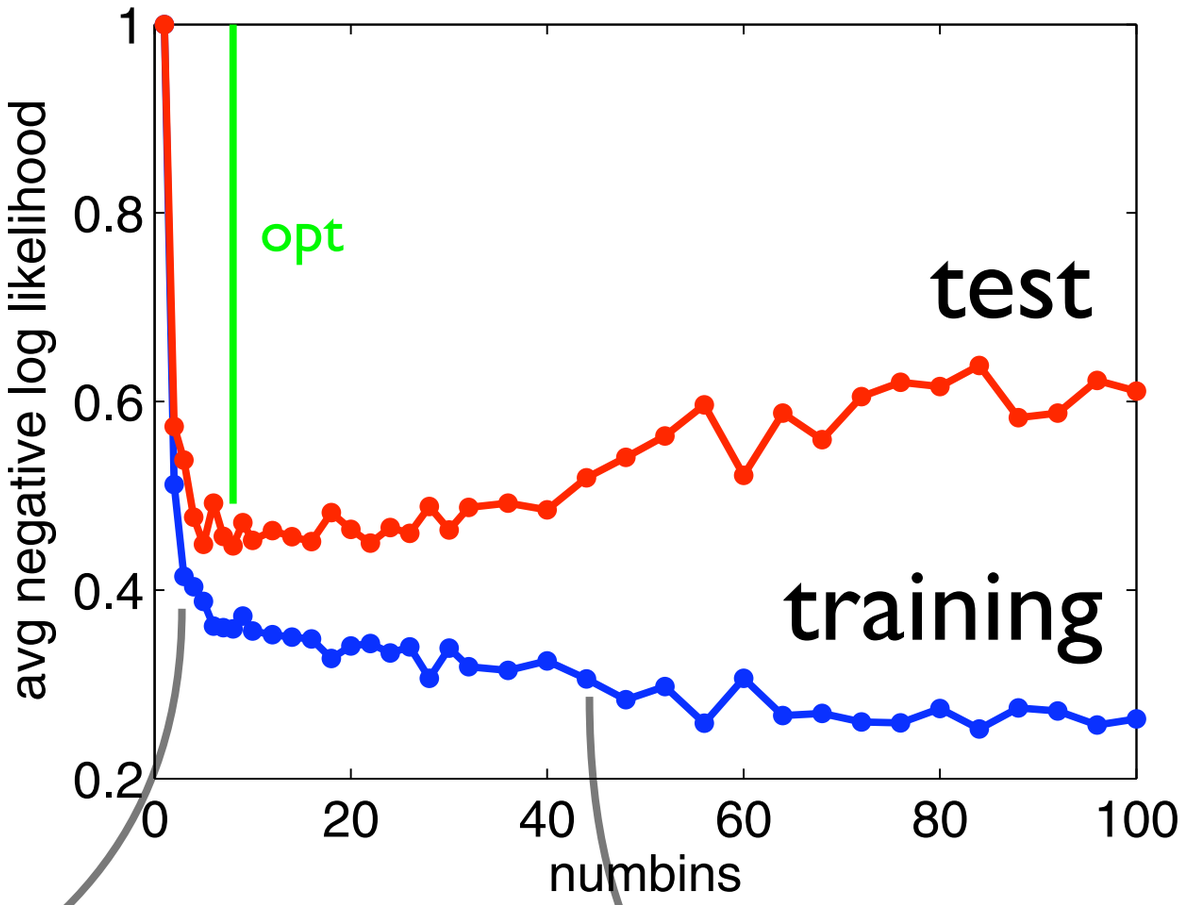


large bias

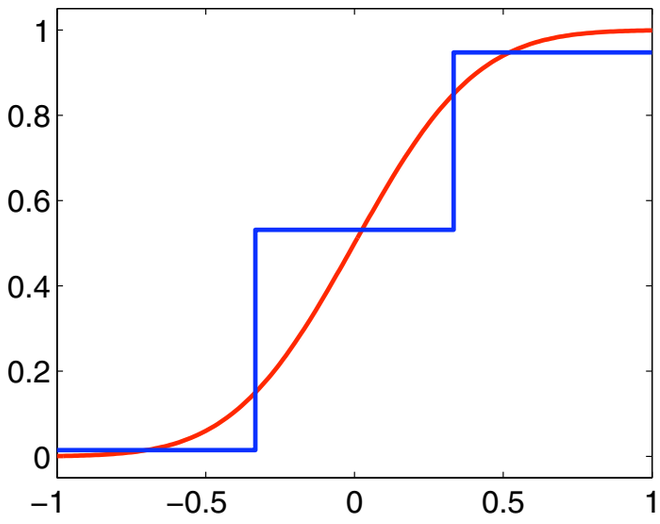


Raw data:

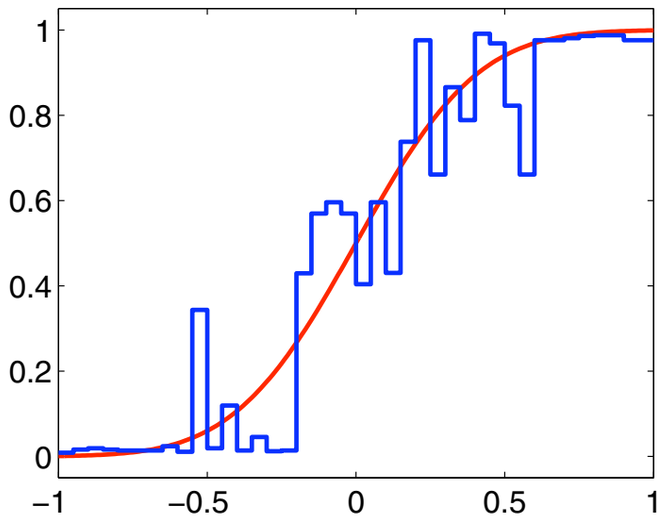




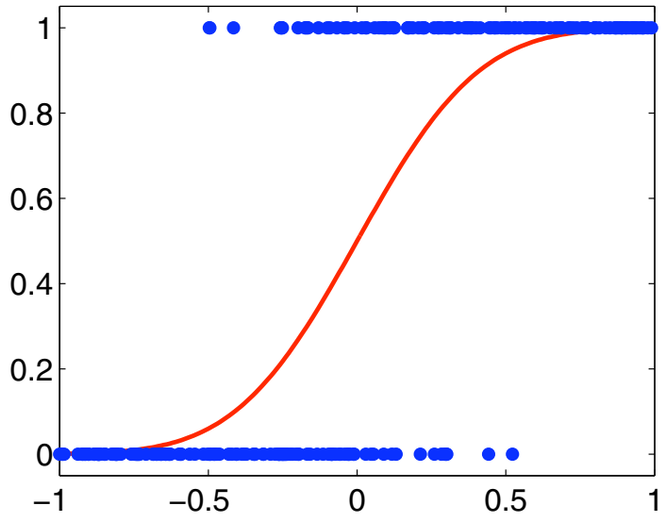
large bias

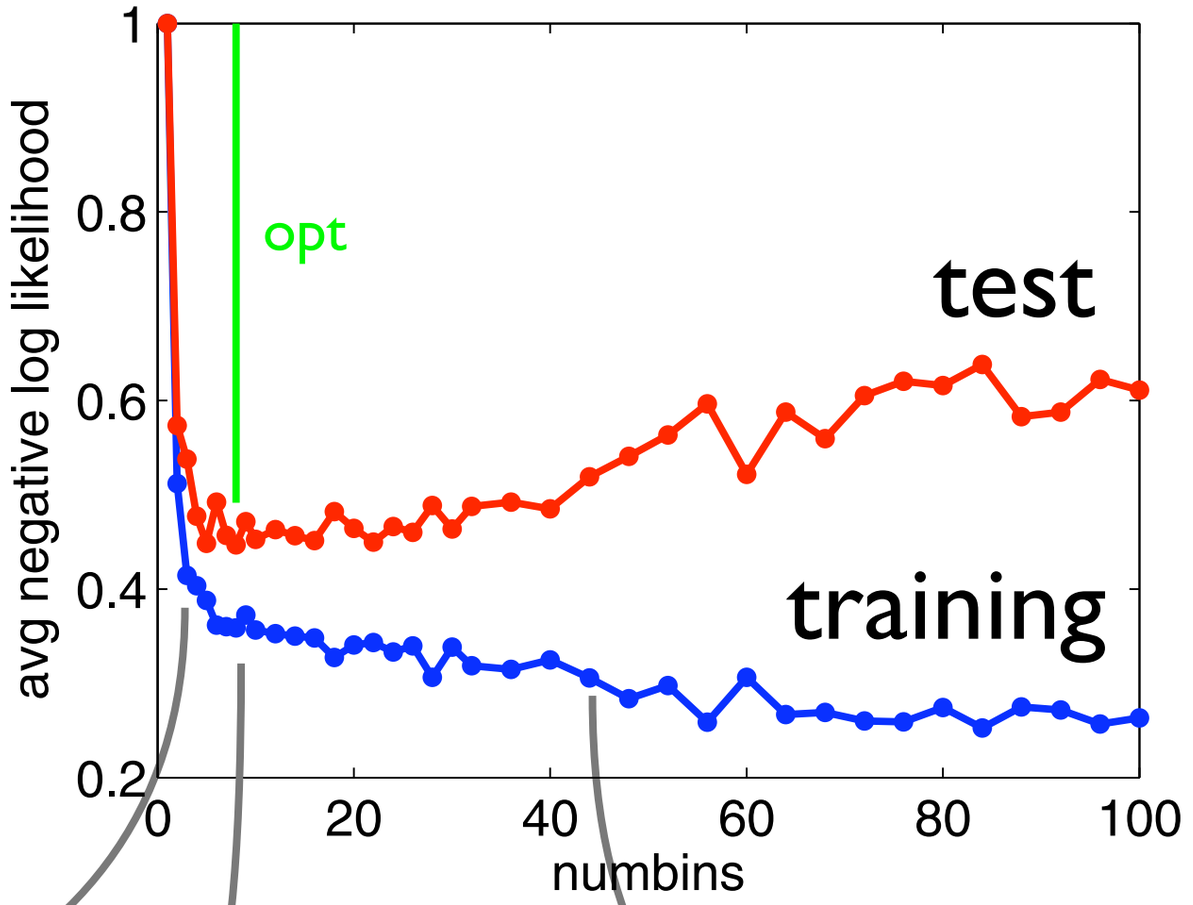


large variance

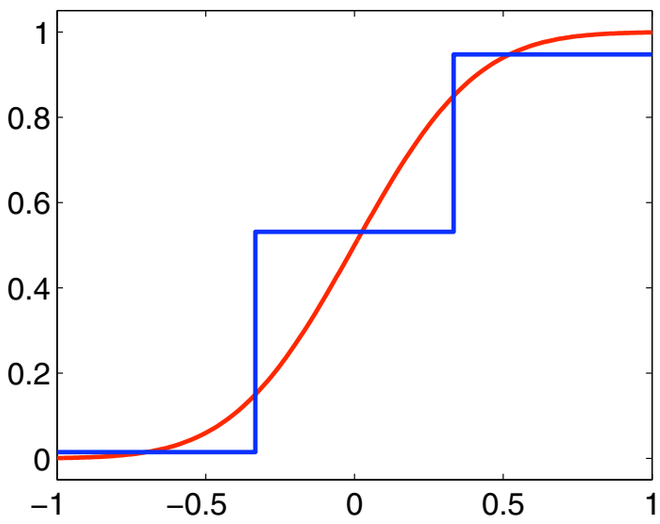


Raw data:

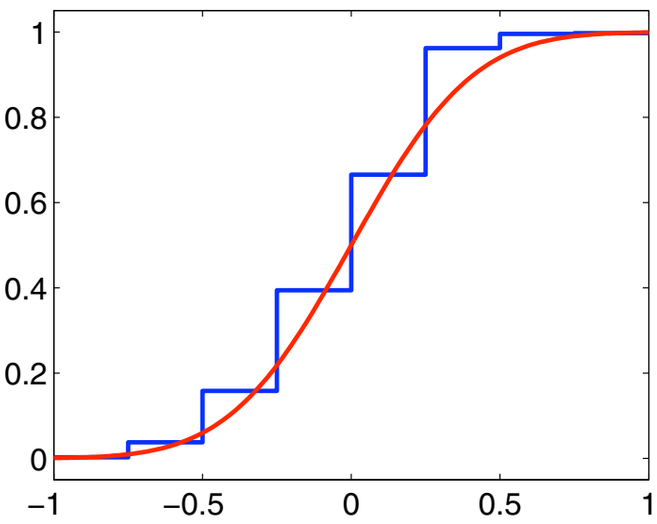




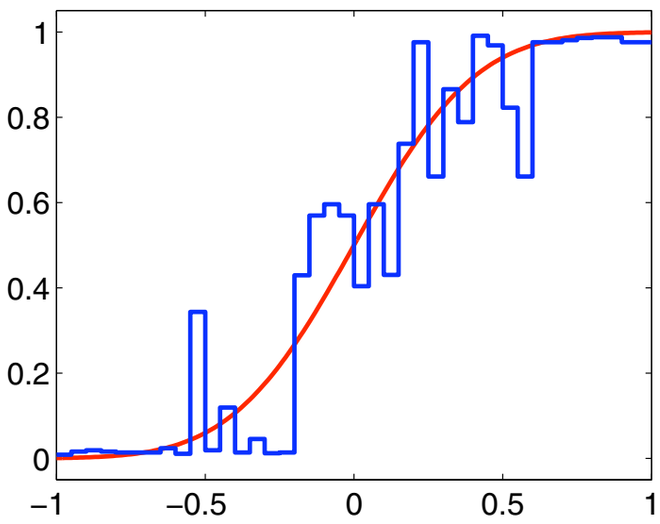
large bias



opt



large variance

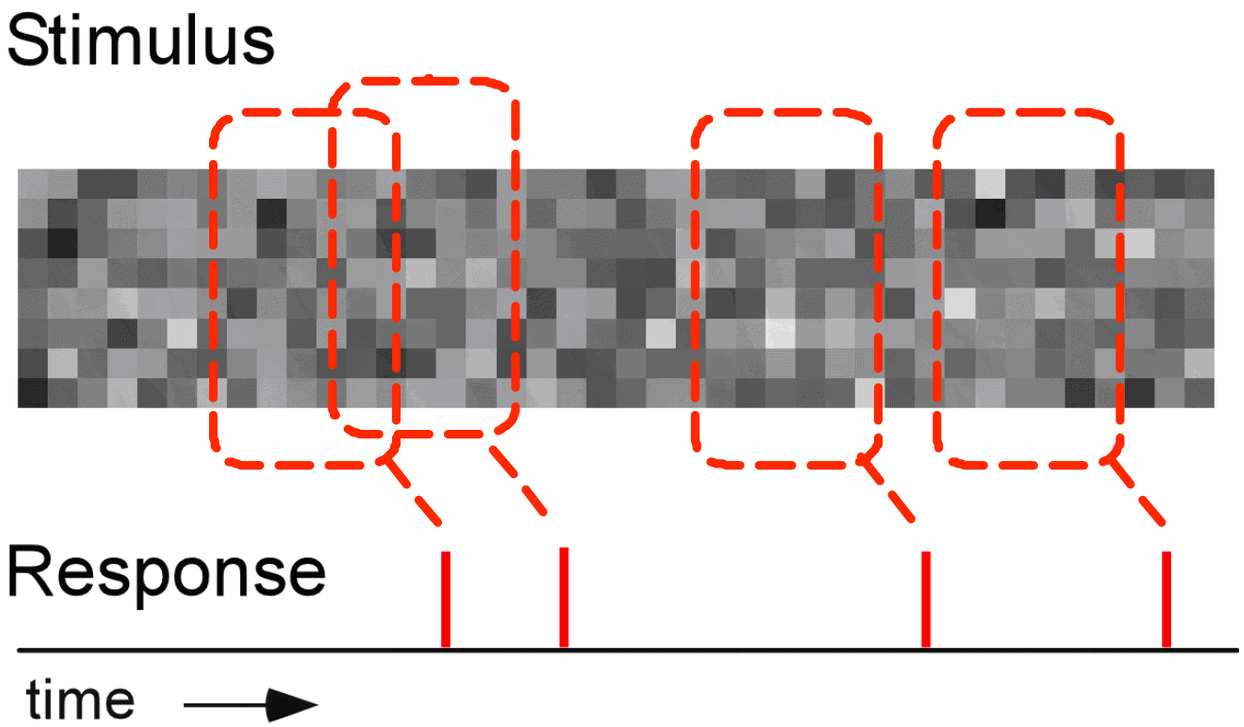


Leftovers

- MSE = bias^2 + variance
- Nonparametric (histogram) estimates (matlab example)

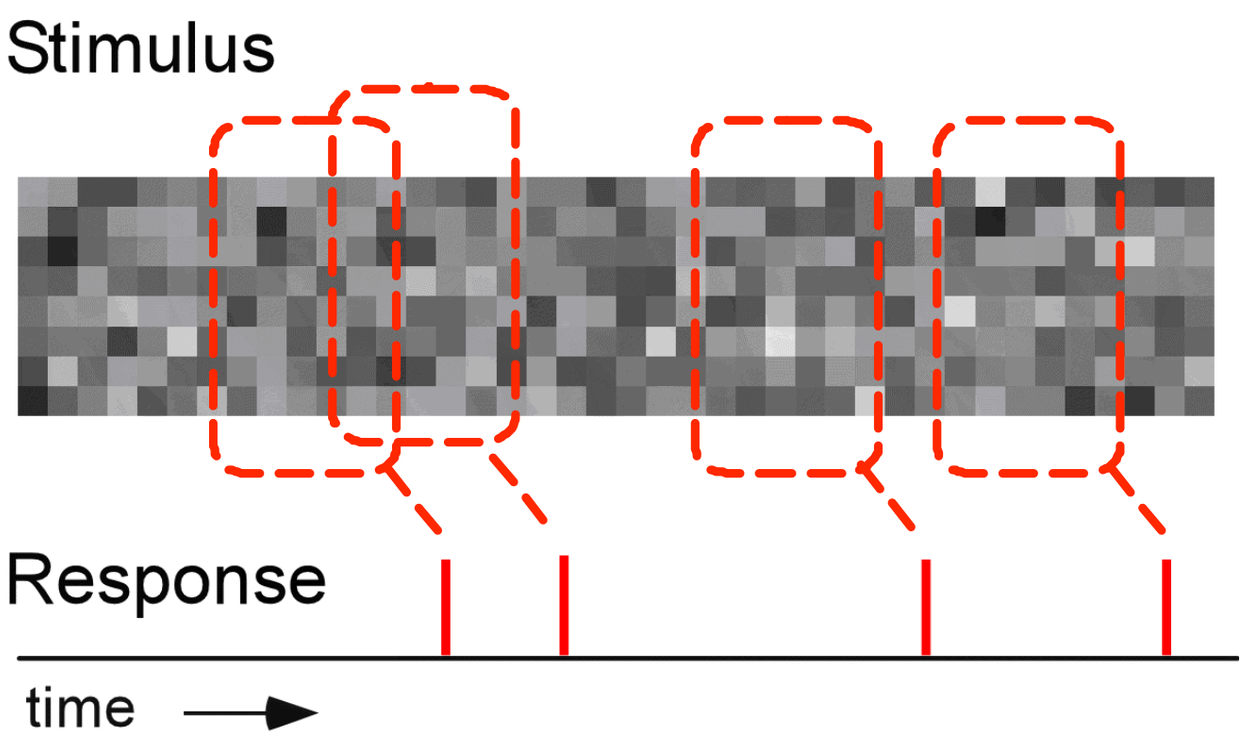
The multi-D problem

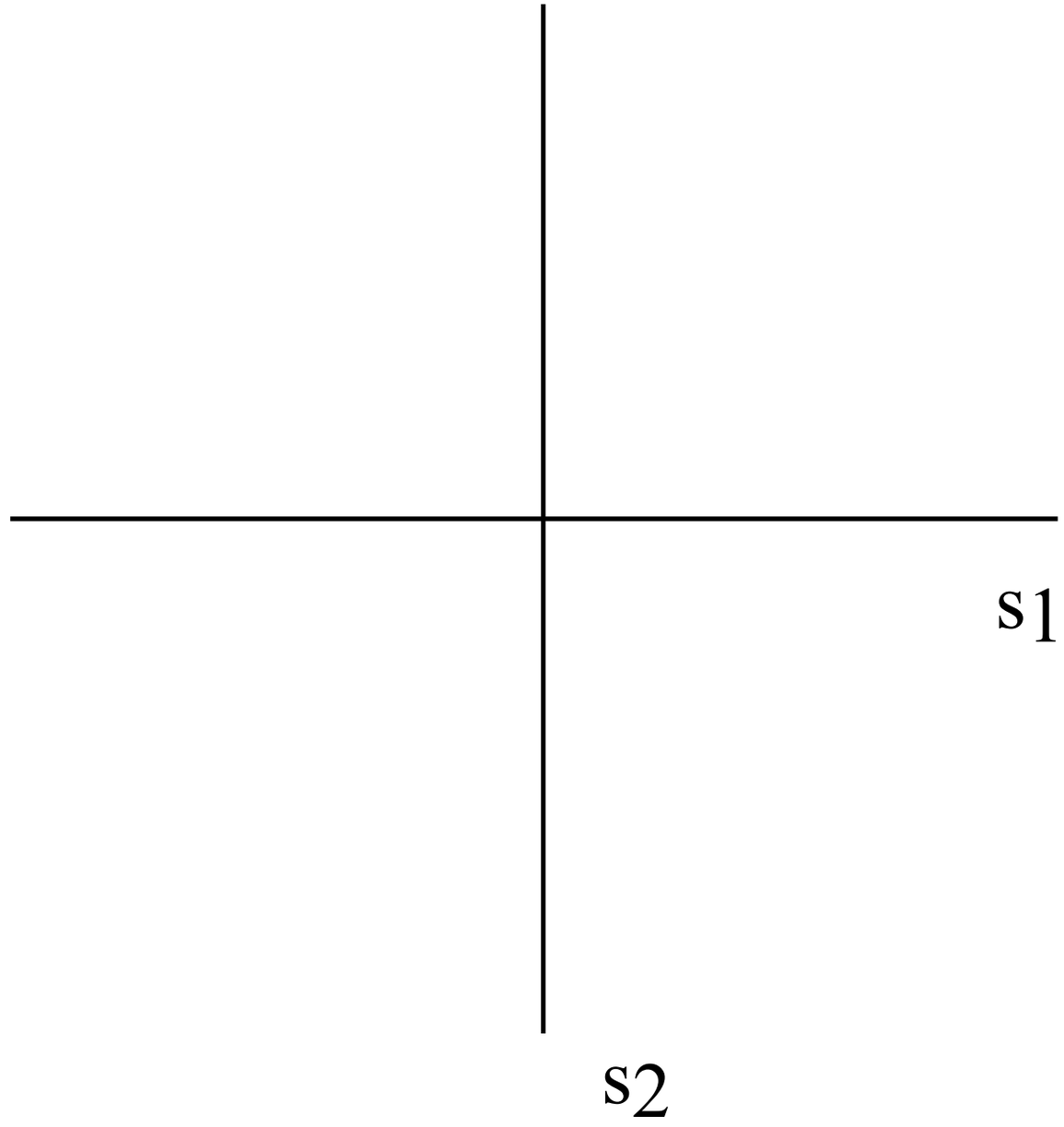
1D stimulus over time
(e.g., flickering bars)



- 8 x 6 stimulus block
  = 48-dimensional vector

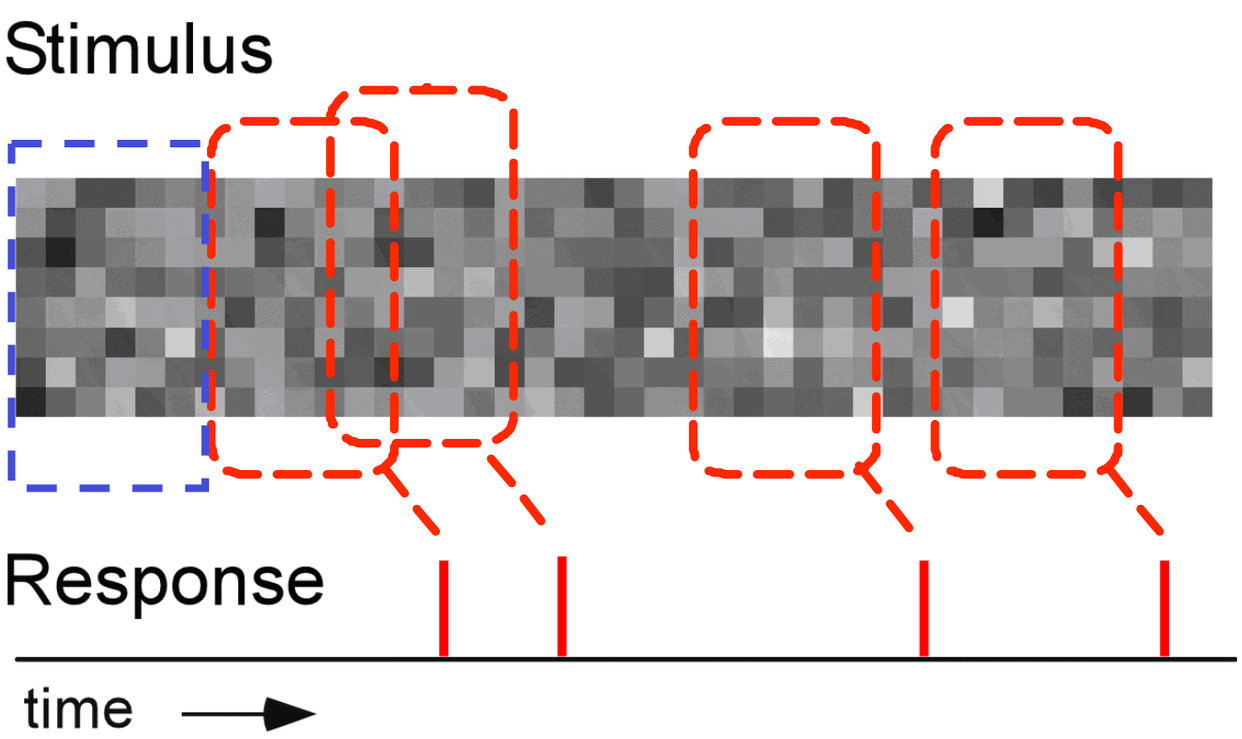
Geometric picture

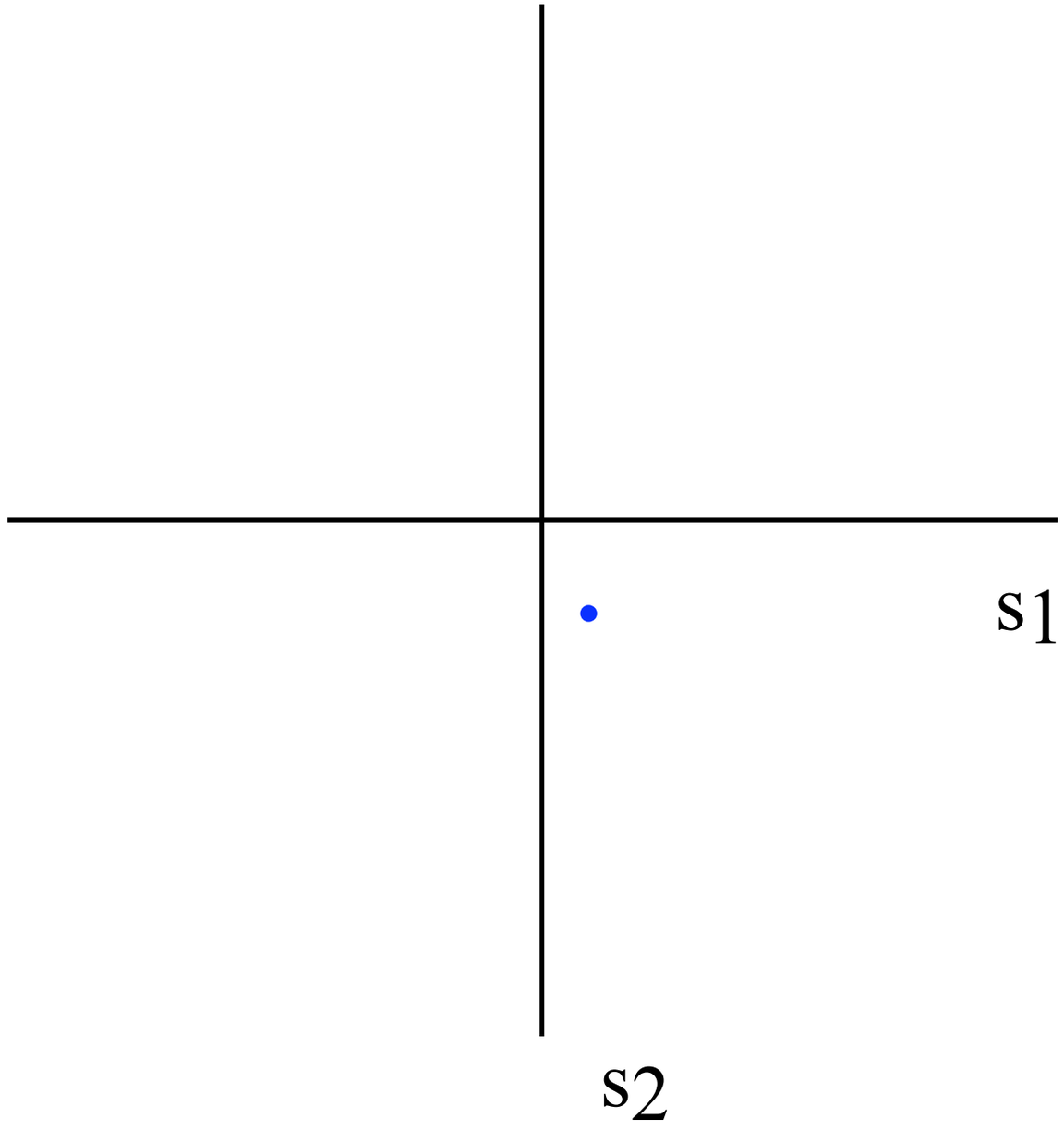




- raw stimuli
- spiking stimuli

Geometric picture

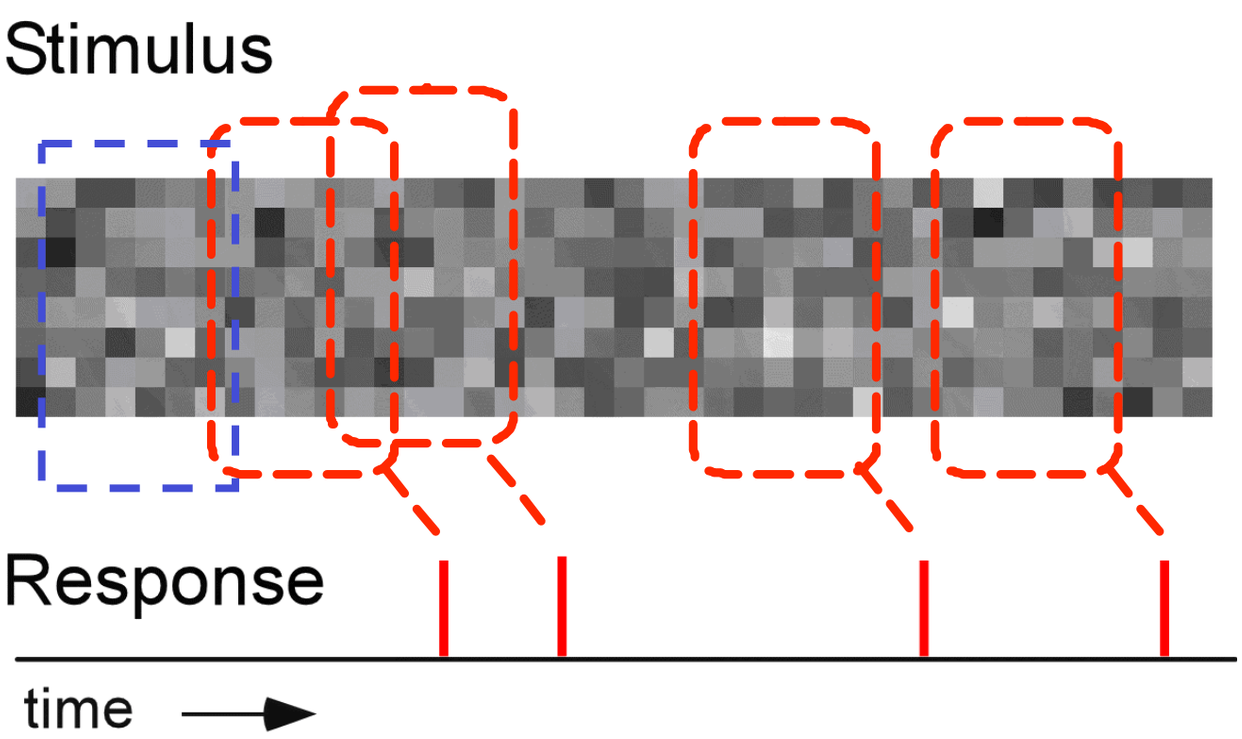


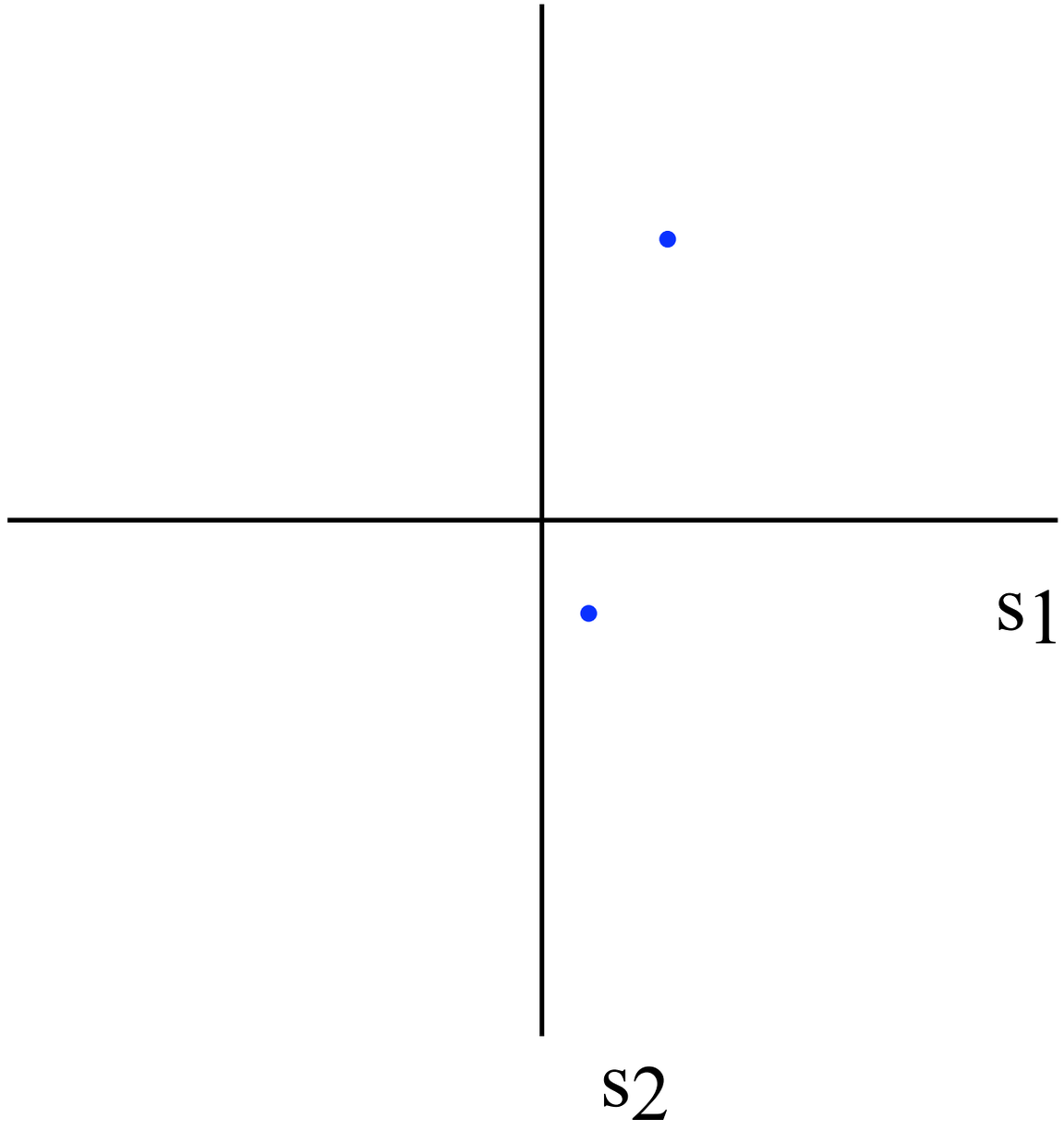


raw stimuli

spiking stimuli

Geometric picture

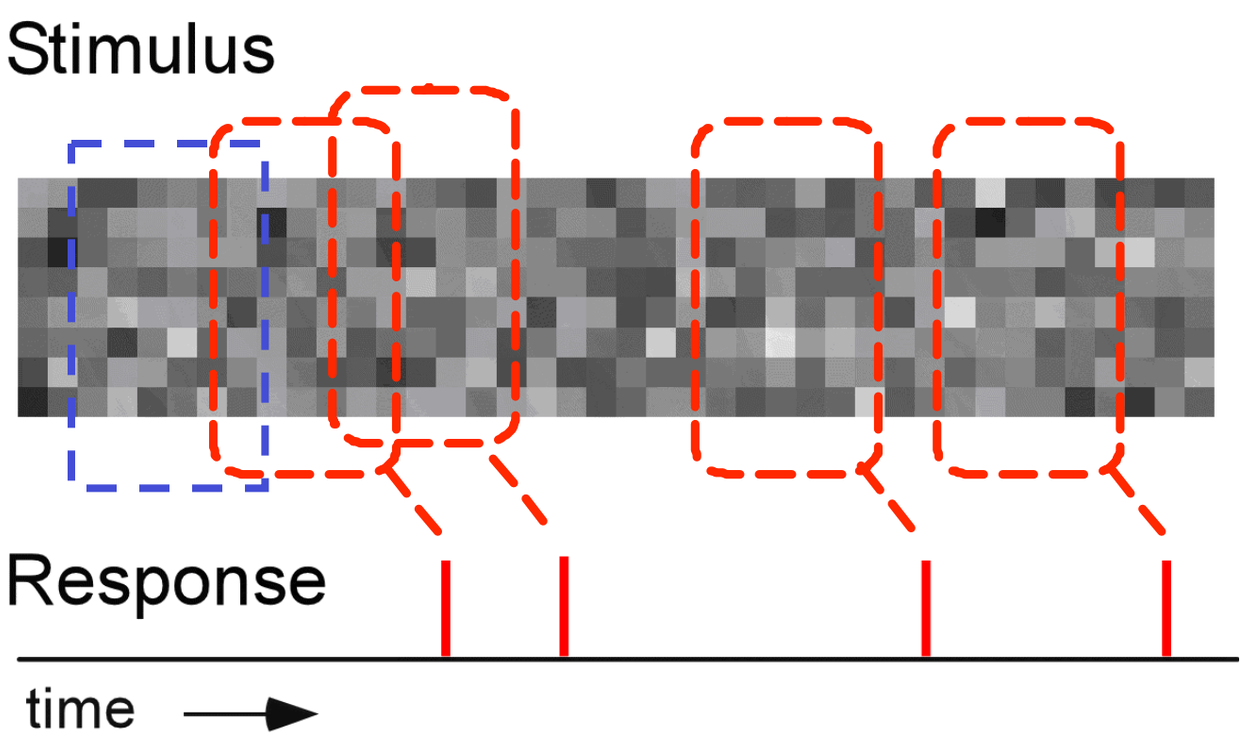


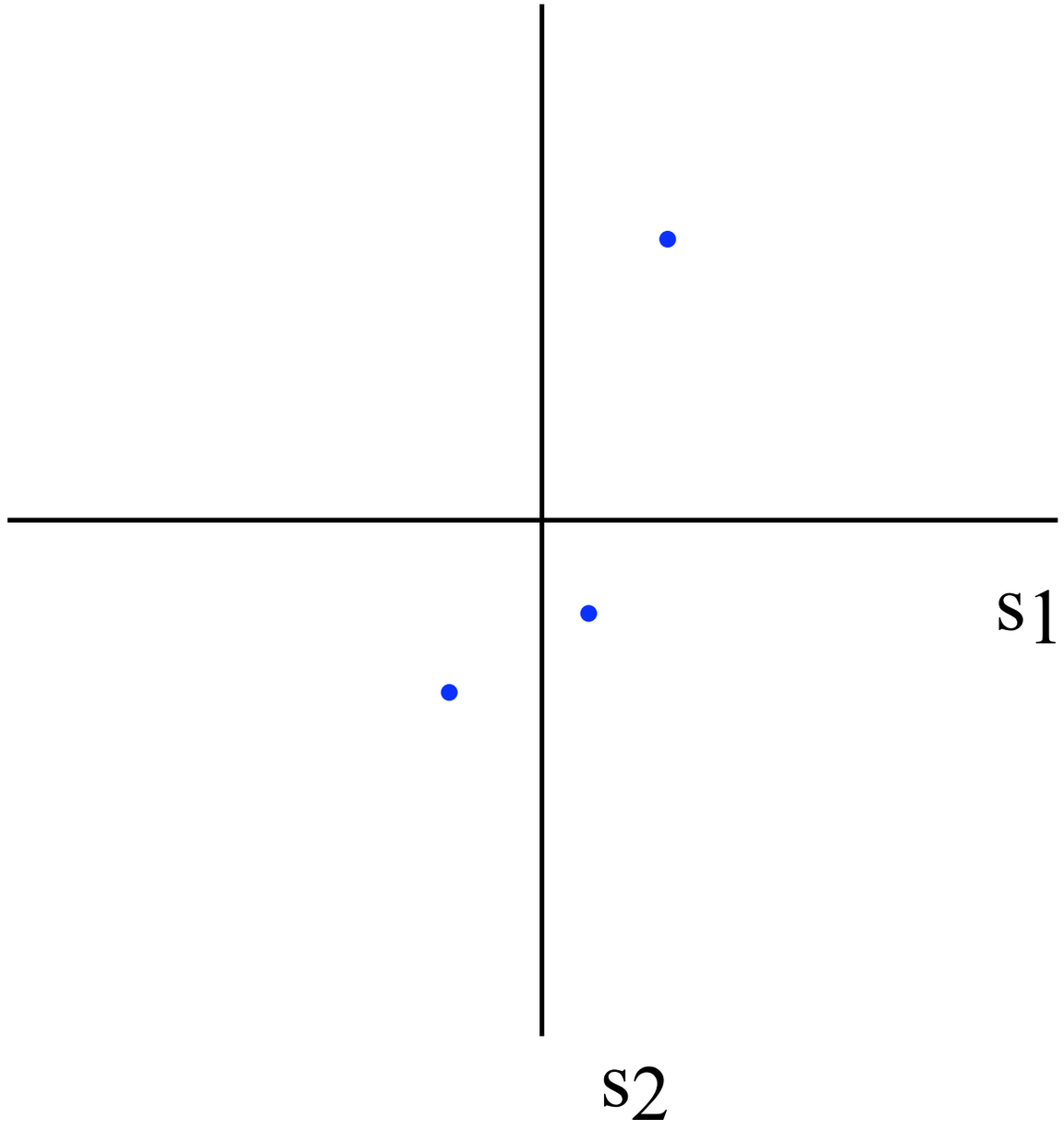


raw stimuli

spiking stimuli

Geometric picture

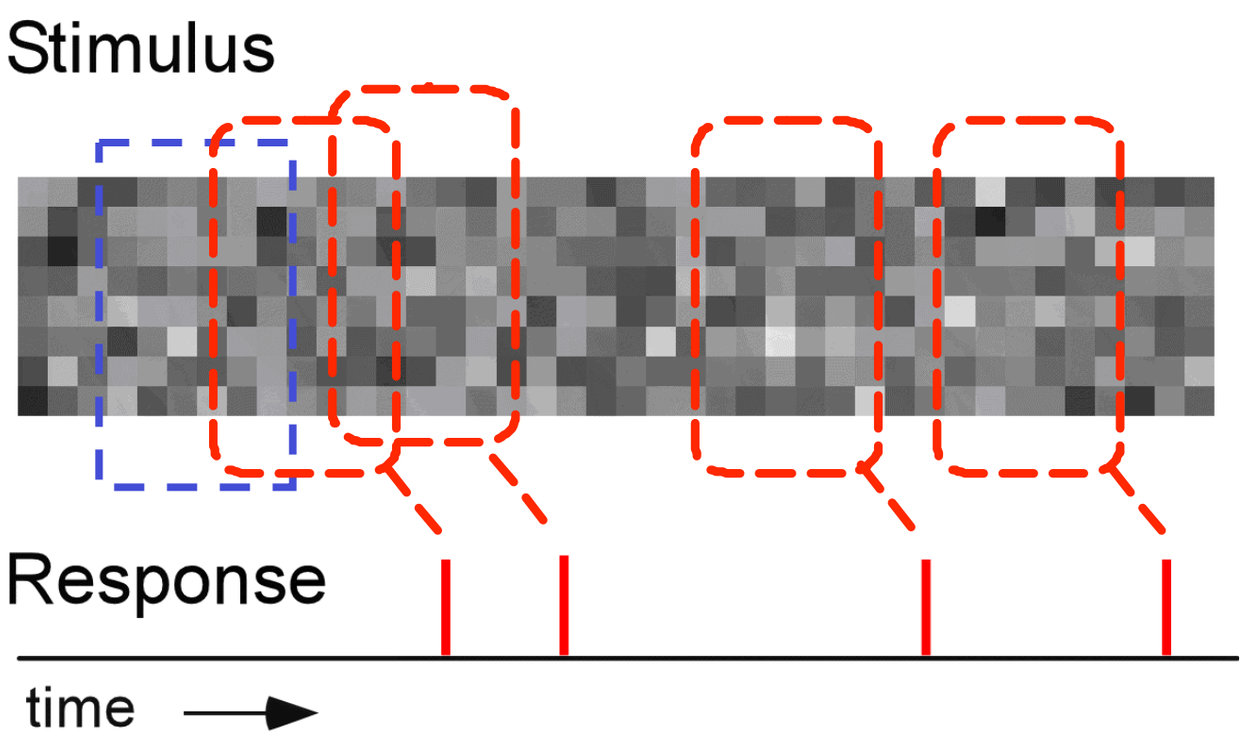


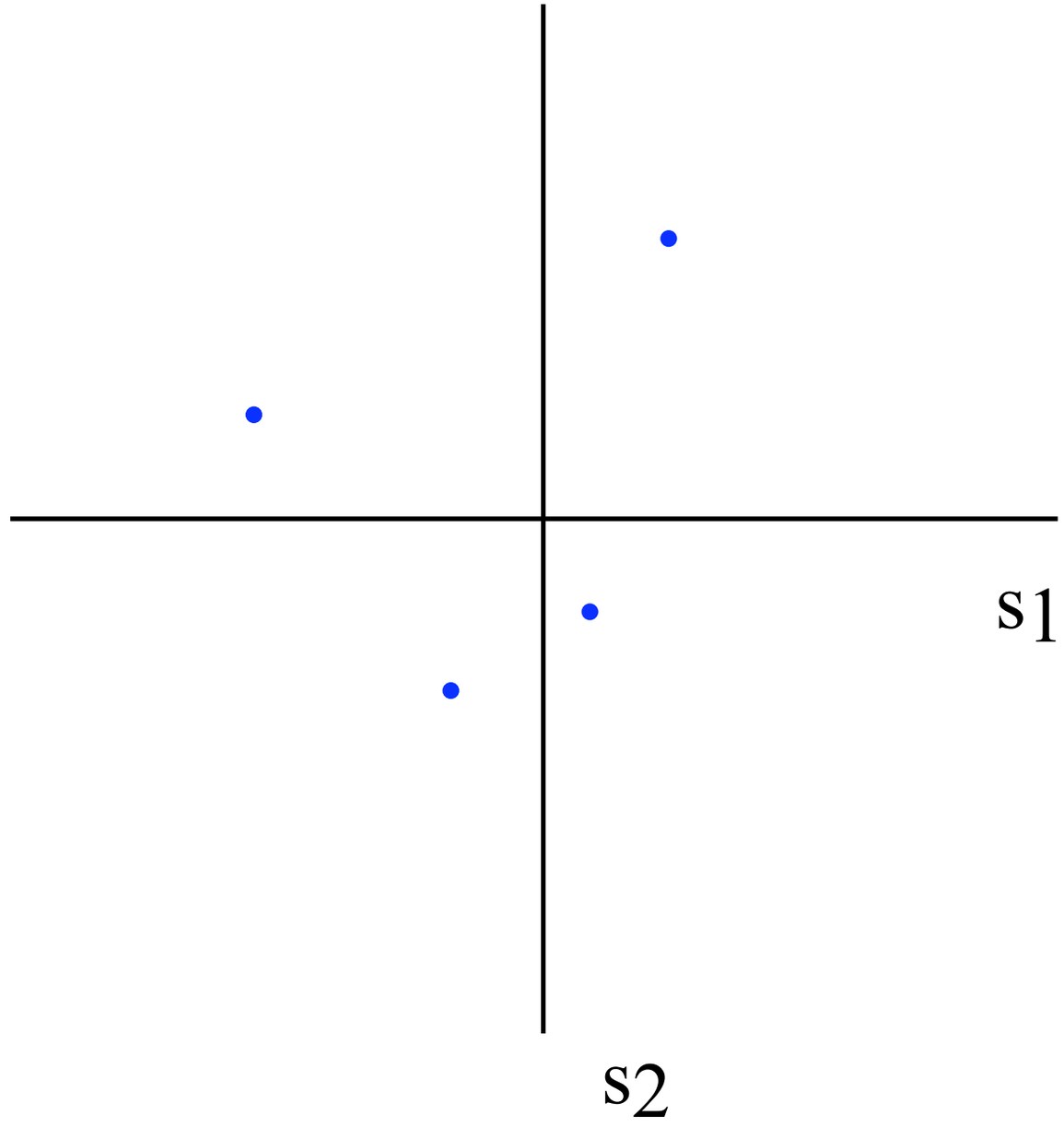


raw stimuli

spiking stimuli

Geometric picture

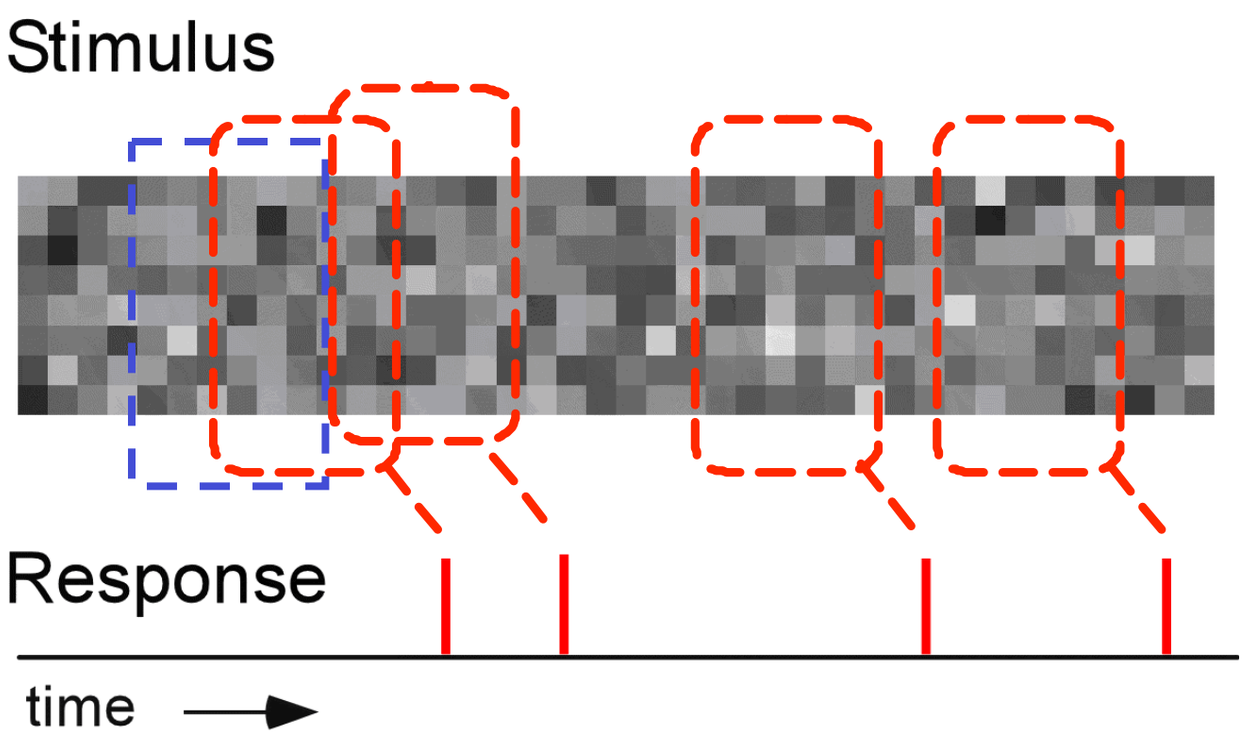


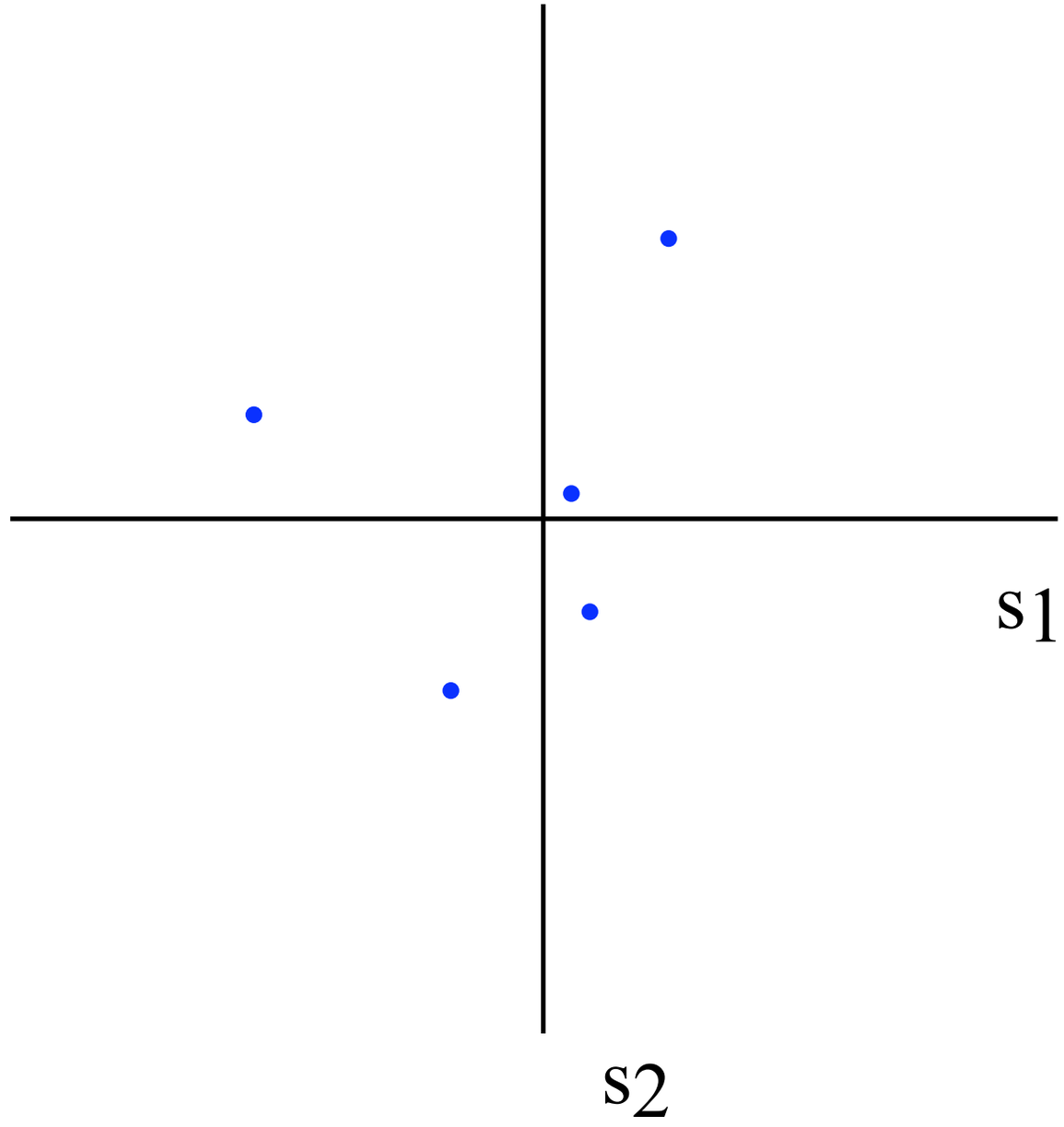


raw stimuli

spiking stimuli

Geometric picture

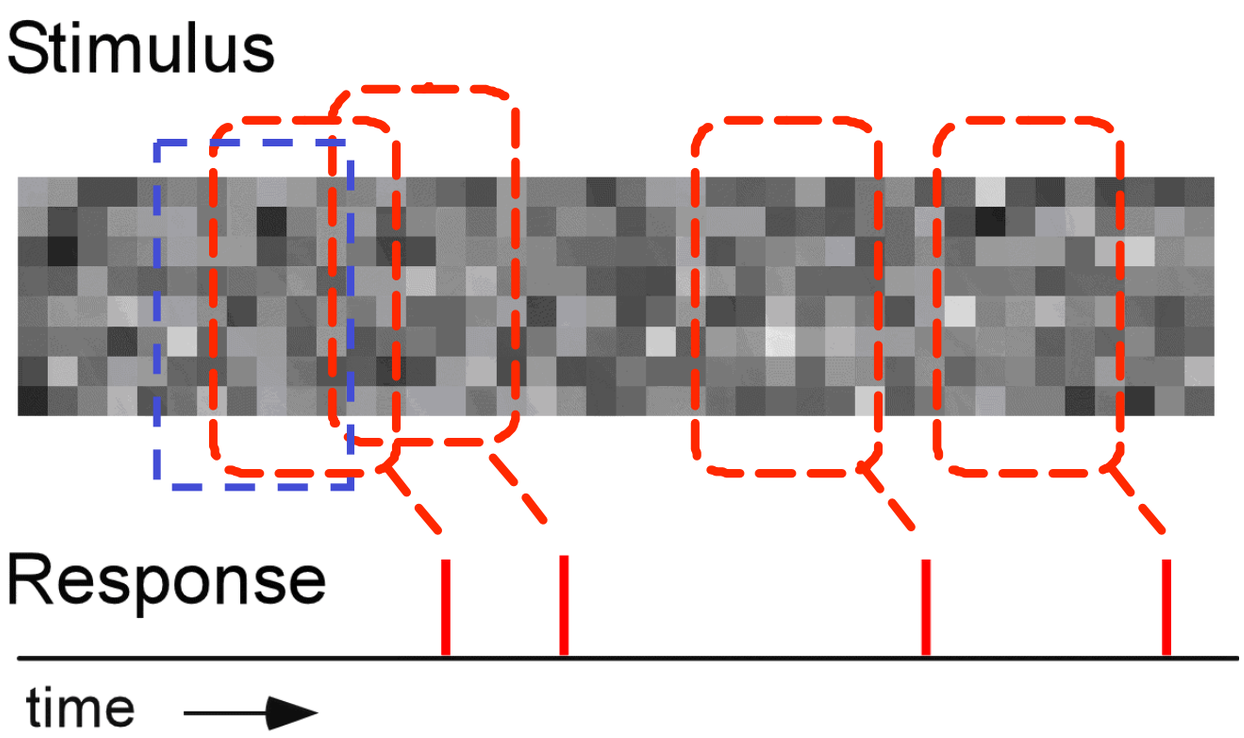


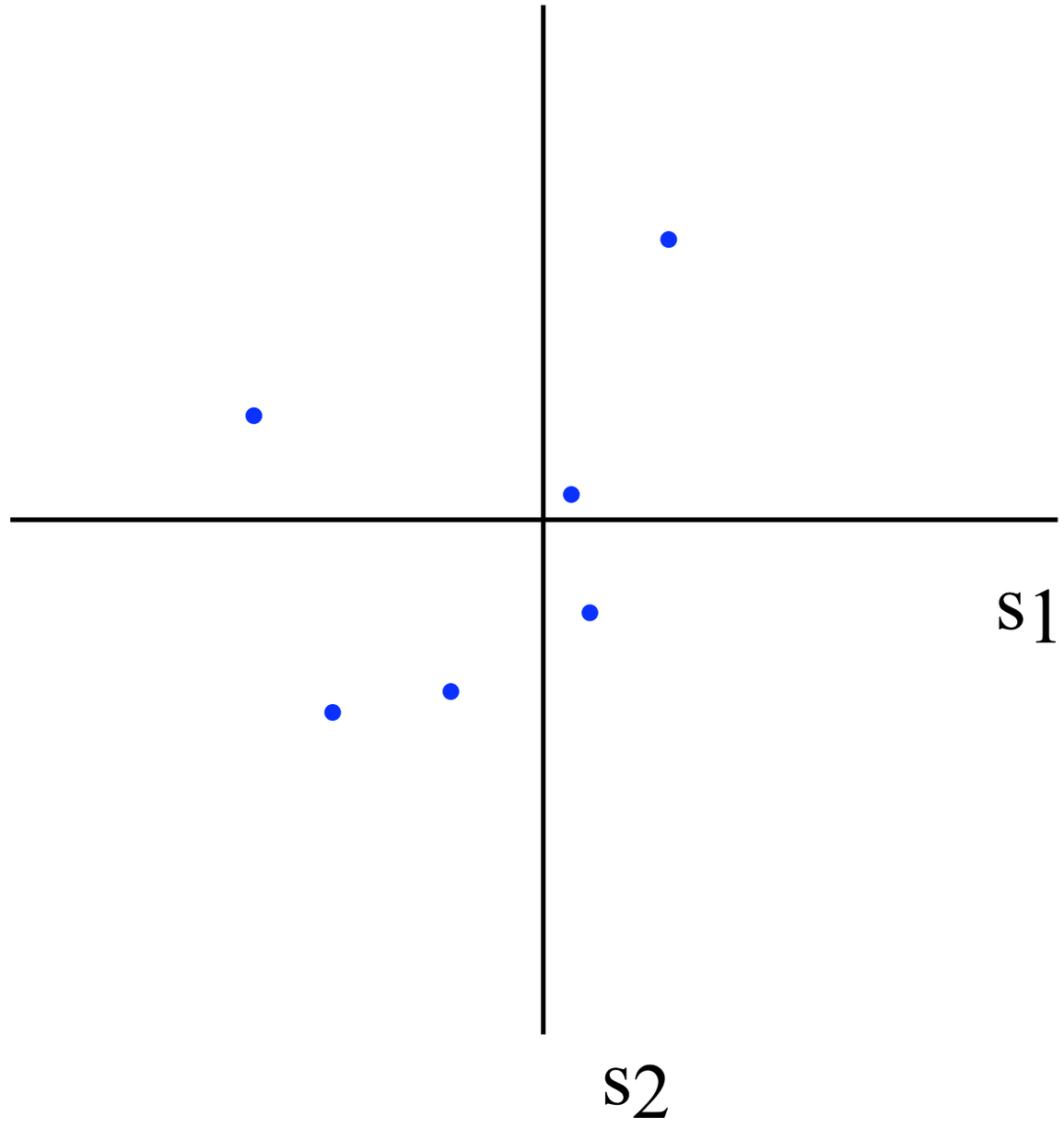


raw stimuli

spiking stimuli

Geometric picture

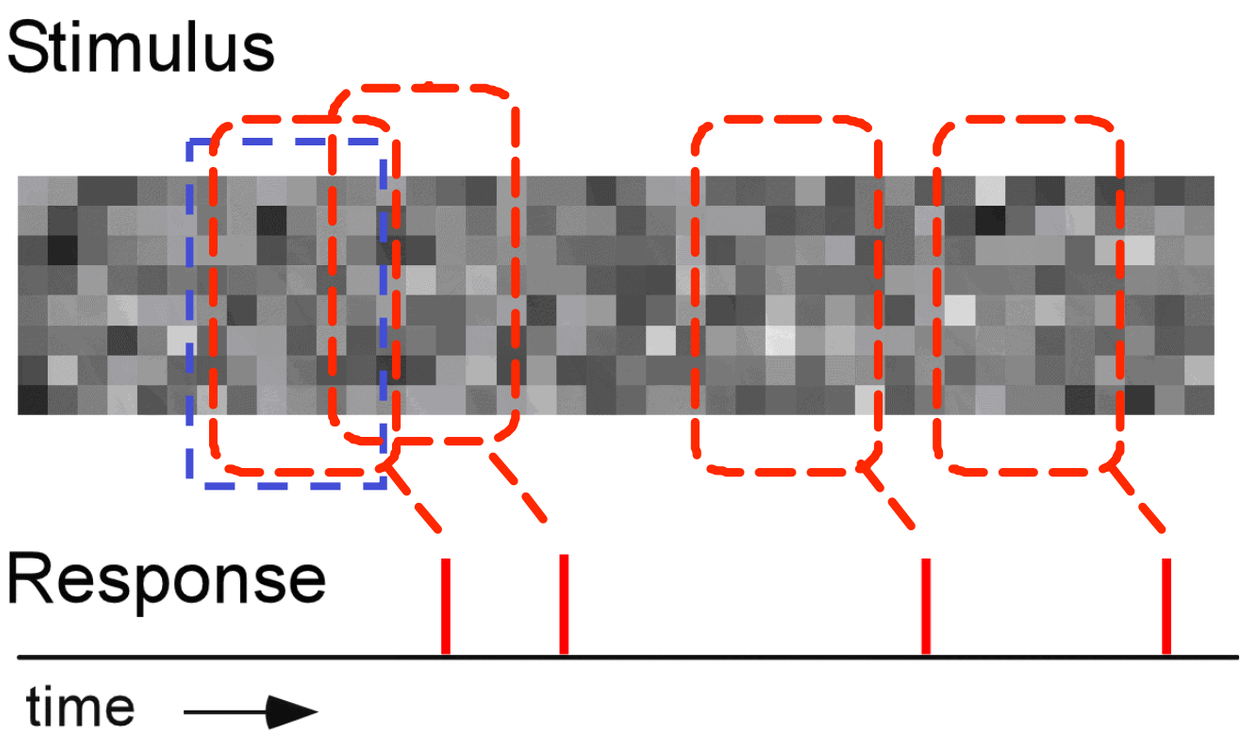


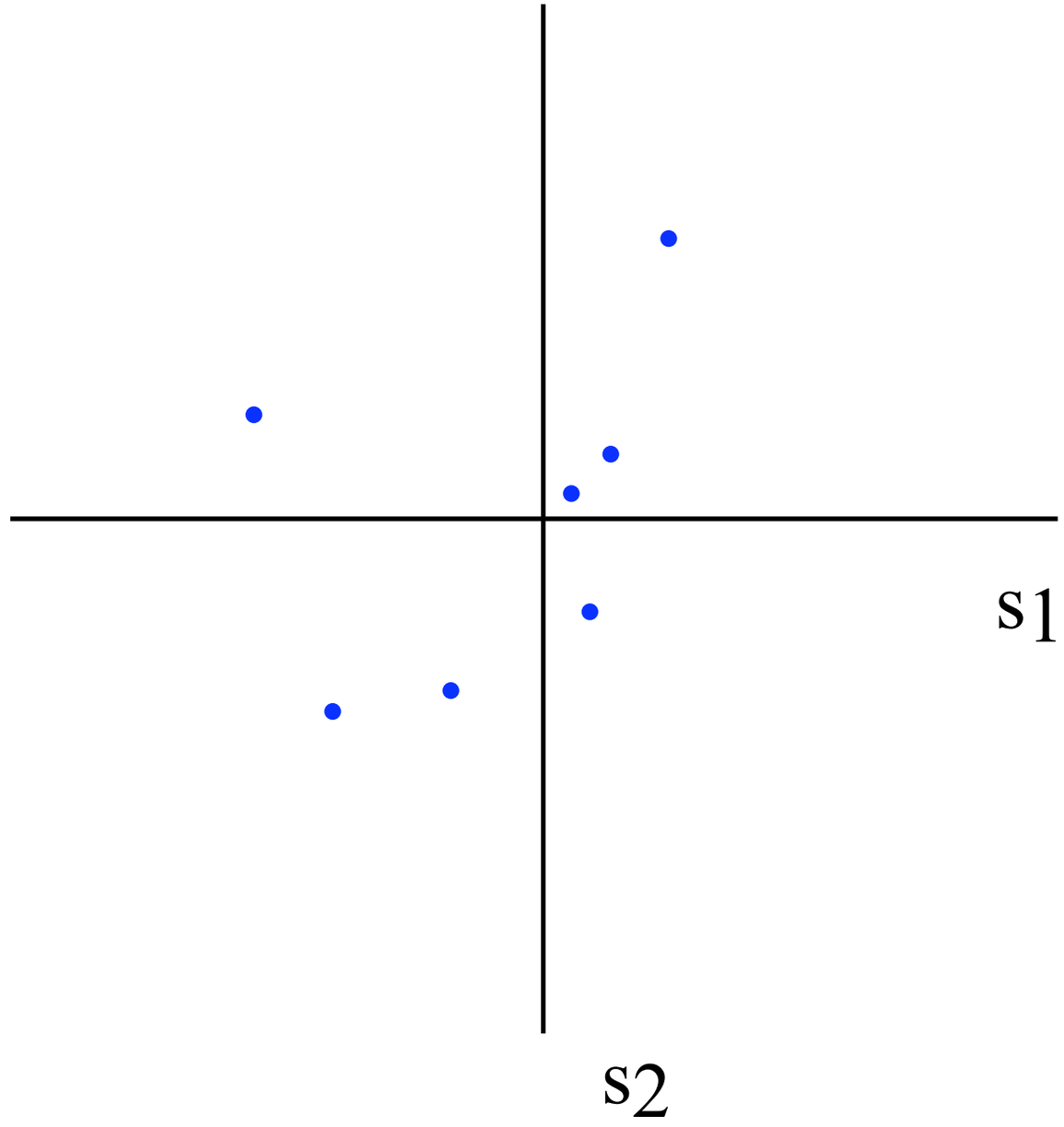


raw stimuli

spiking stimuli

Geometric picture

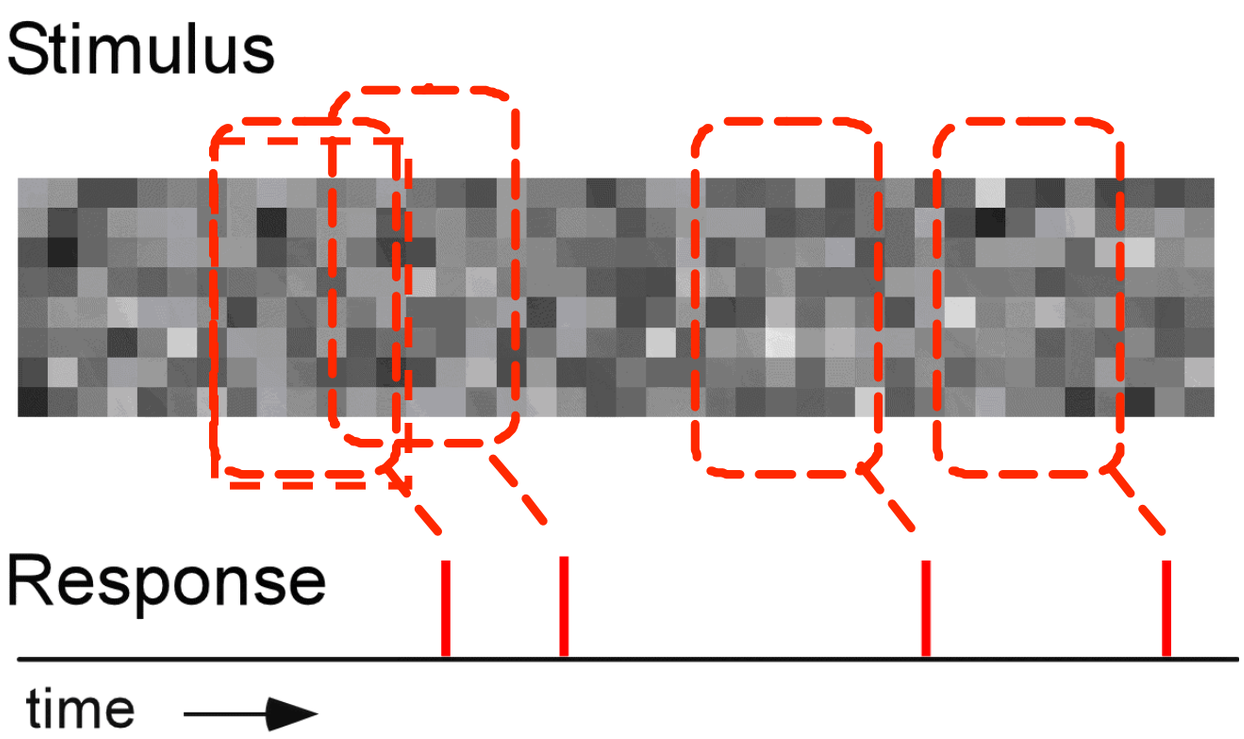


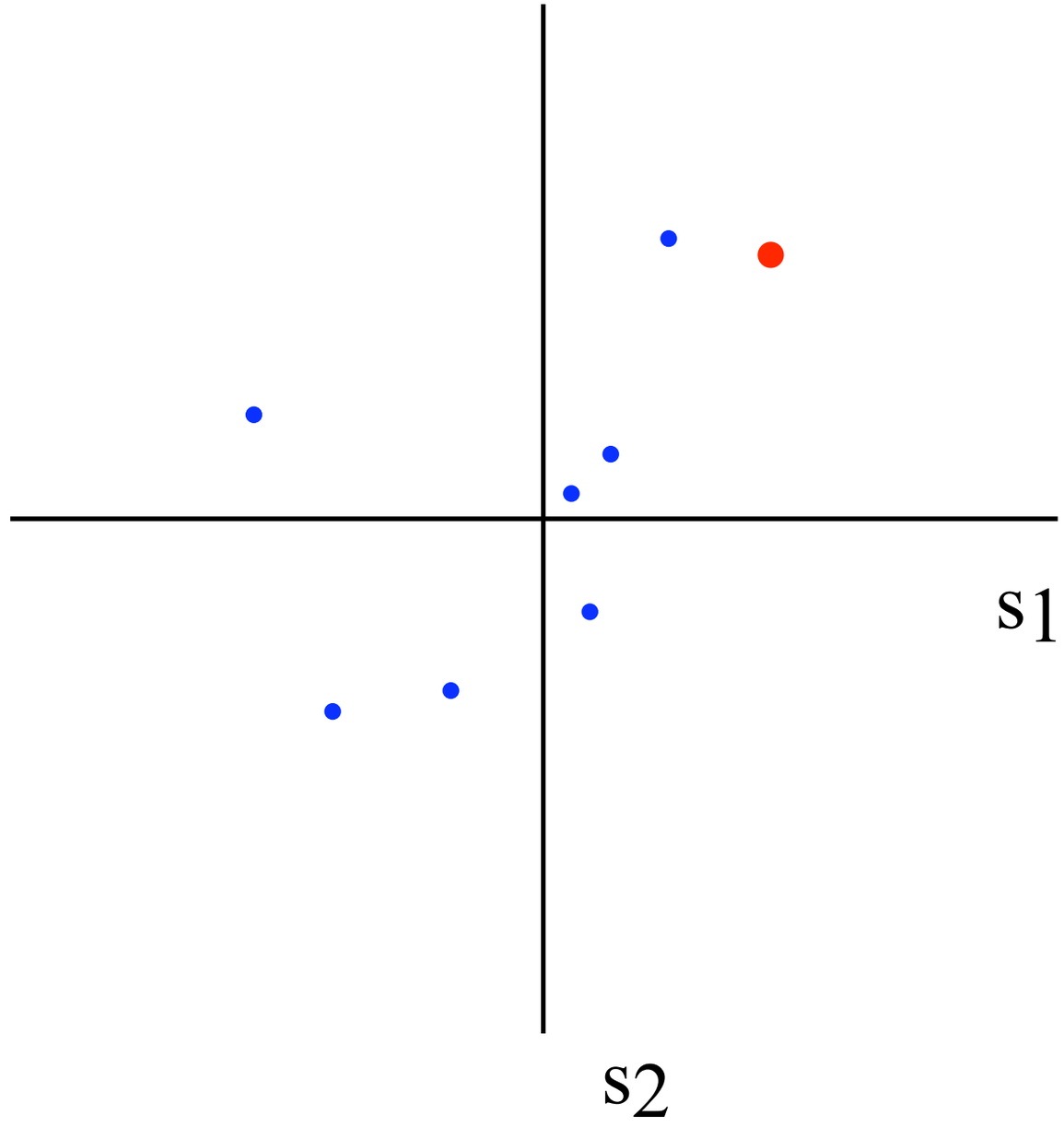


raw stimuli

spiking stimuli

Geometric picture

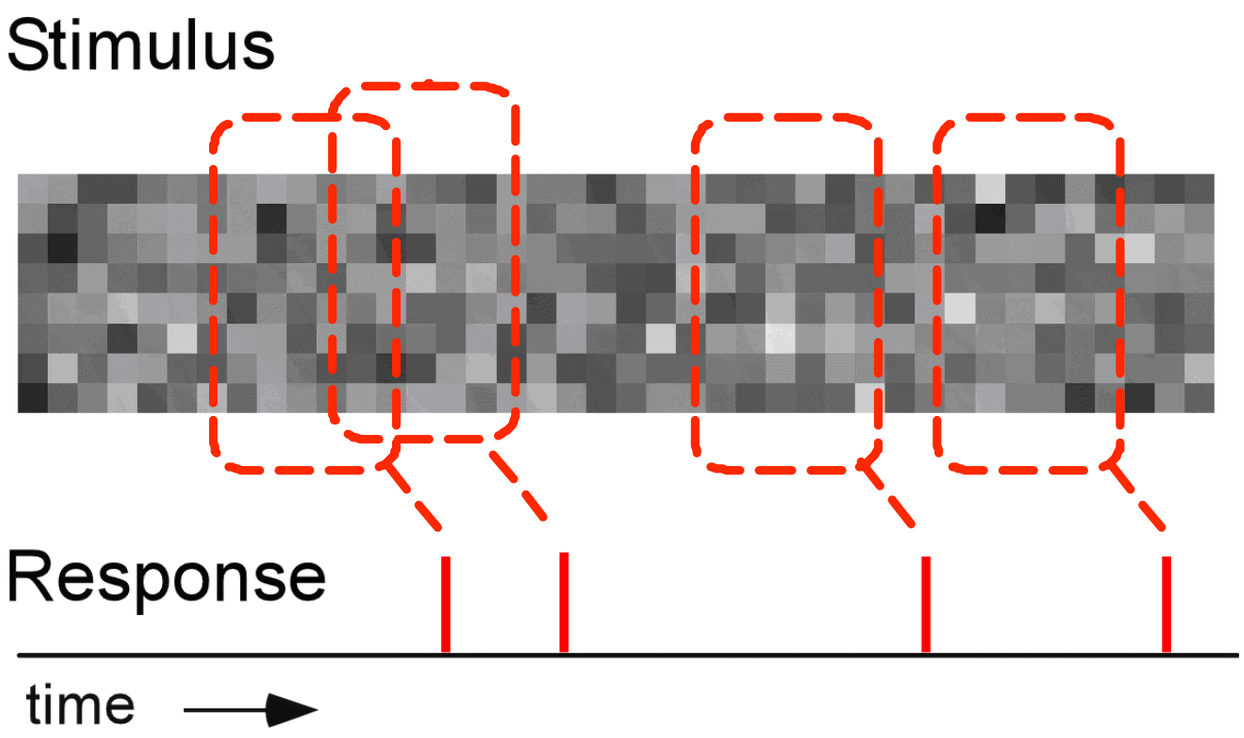


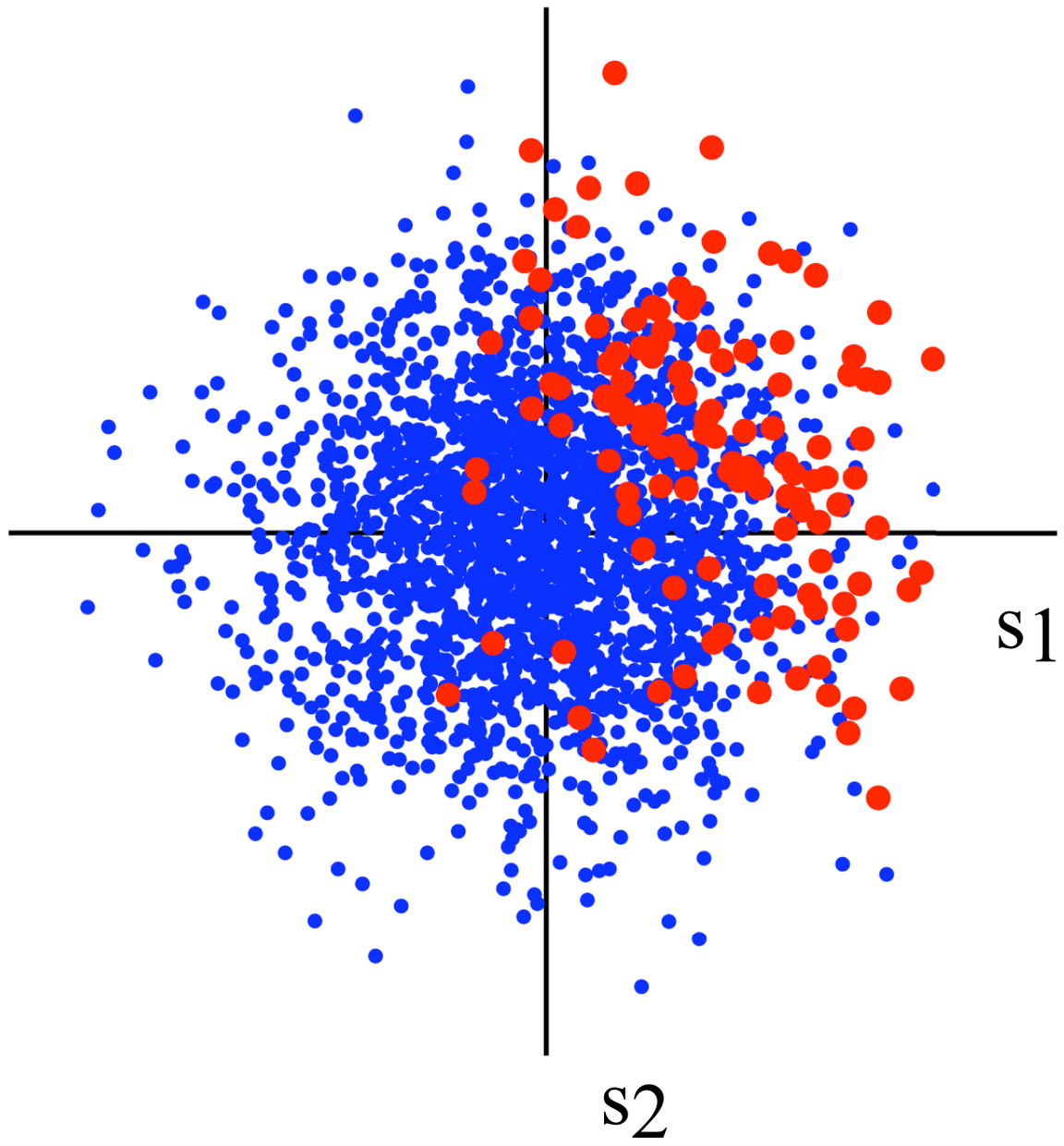


raw stimuli

spiking stimuli

Geometric picture

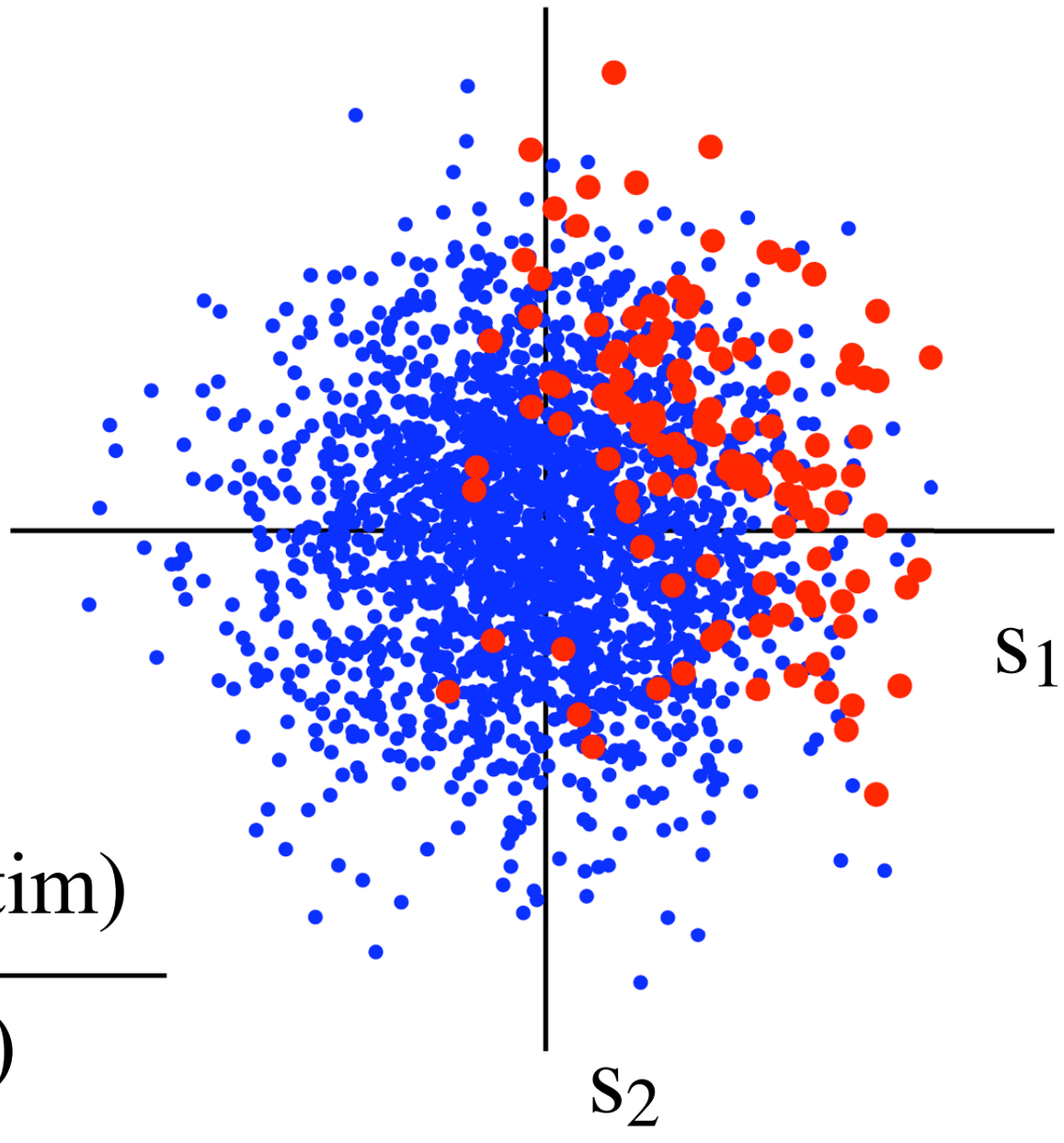




raw stimuli

spiking stimuli

Neural response is captured by relationship between the distribution of red points (spiking stim) and blue points (raw stim)



Expressed in terms of Bayes' rule:

$$P(\text{spike}|\text{stim}) = \frac{P(\text{spike, stim})}{P(\text{stim})}$$

Cannot be estimated nonparametrically...

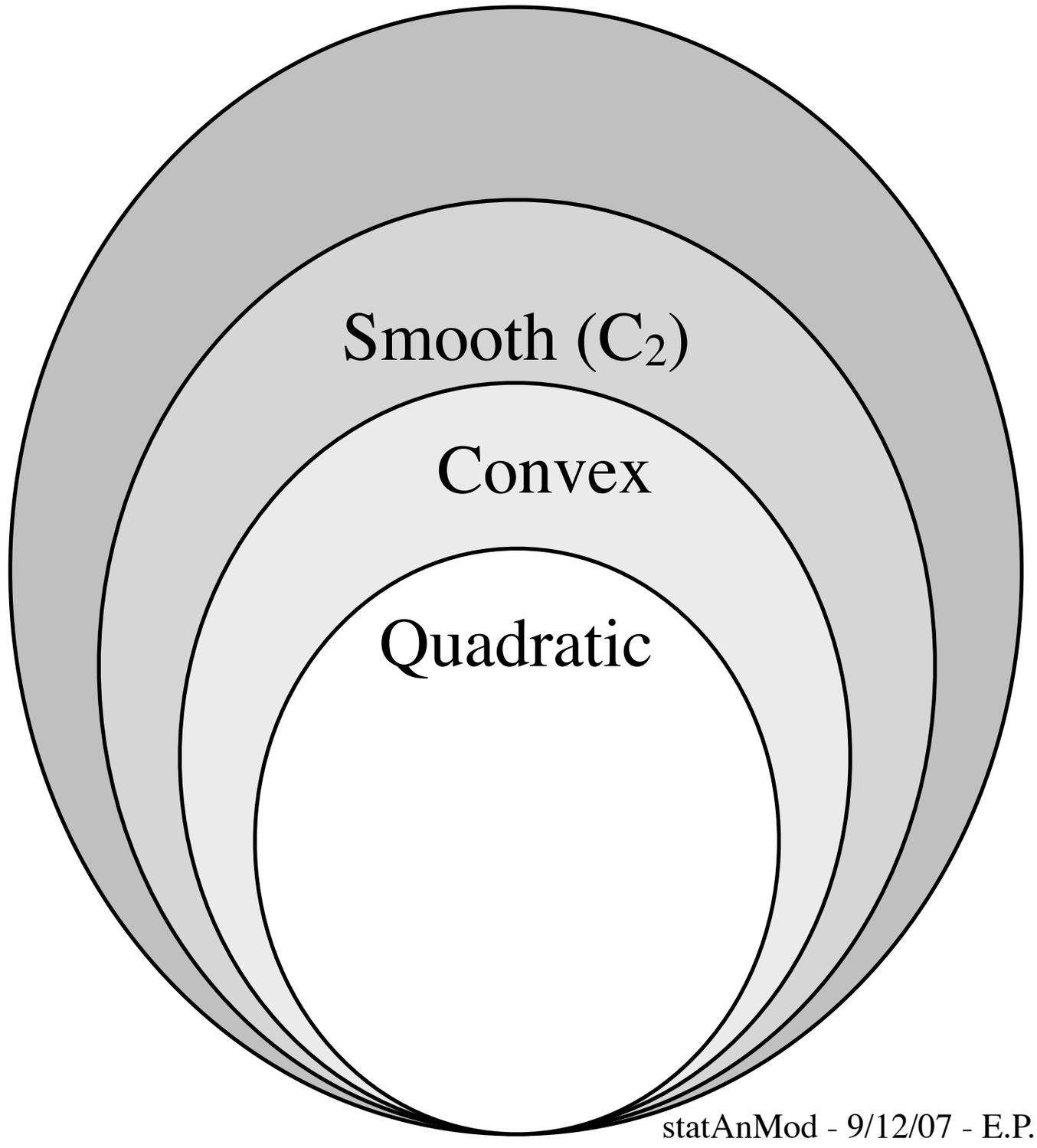
Parametric model & ML

- ML is *consistent* - converges to correct value)
- ML errors are *asymptotically Normal*
- ML is *asymptotically optimal* (minimum variance)
- ML is *equivariant* - for monotonic *f*,

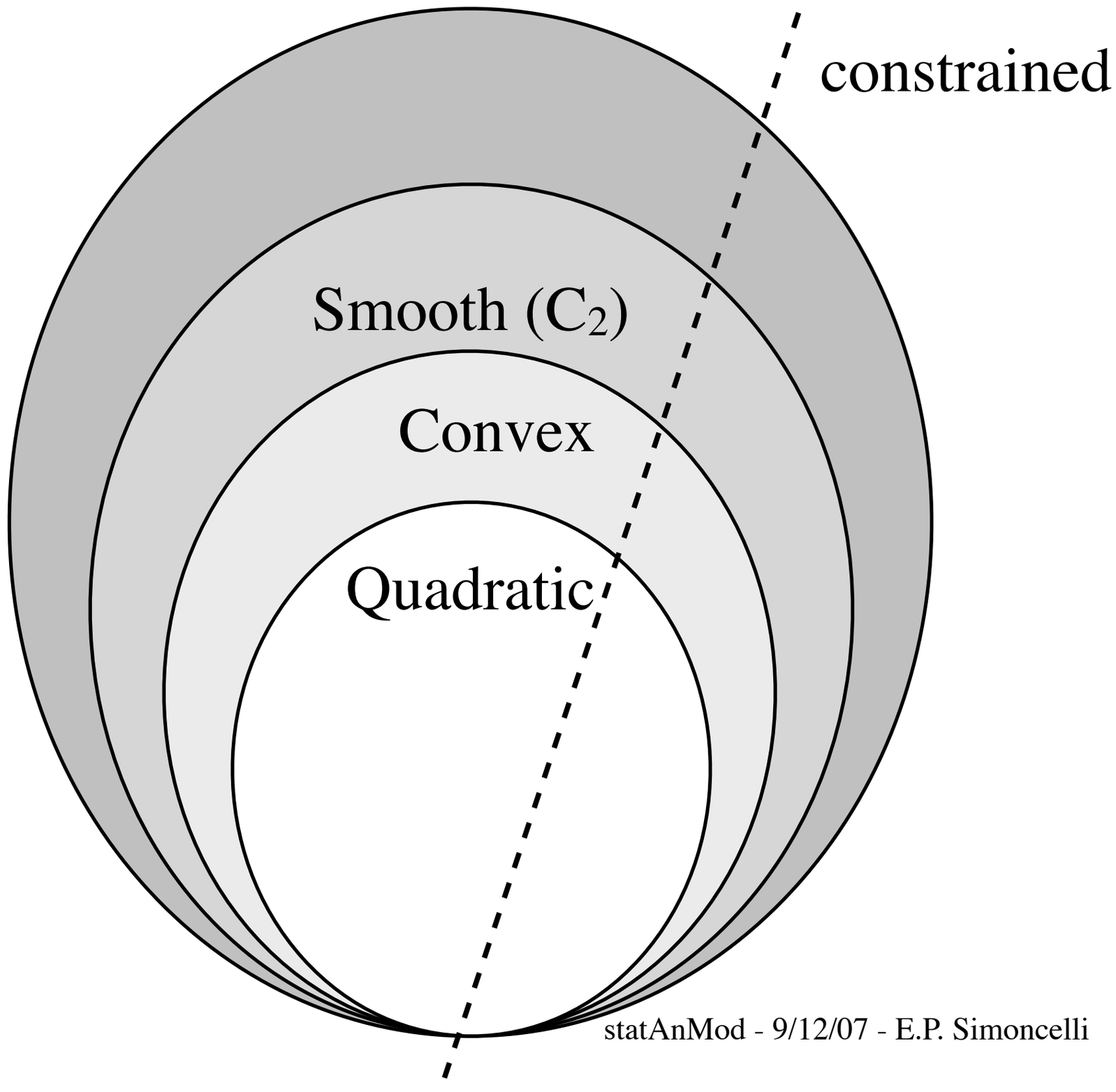
$$\arg\max_{\theta} E(x;\theta) = \arg\max_{\theta} f(E(x;\theta))$$

eg., *f = log, exp, tan, ...*

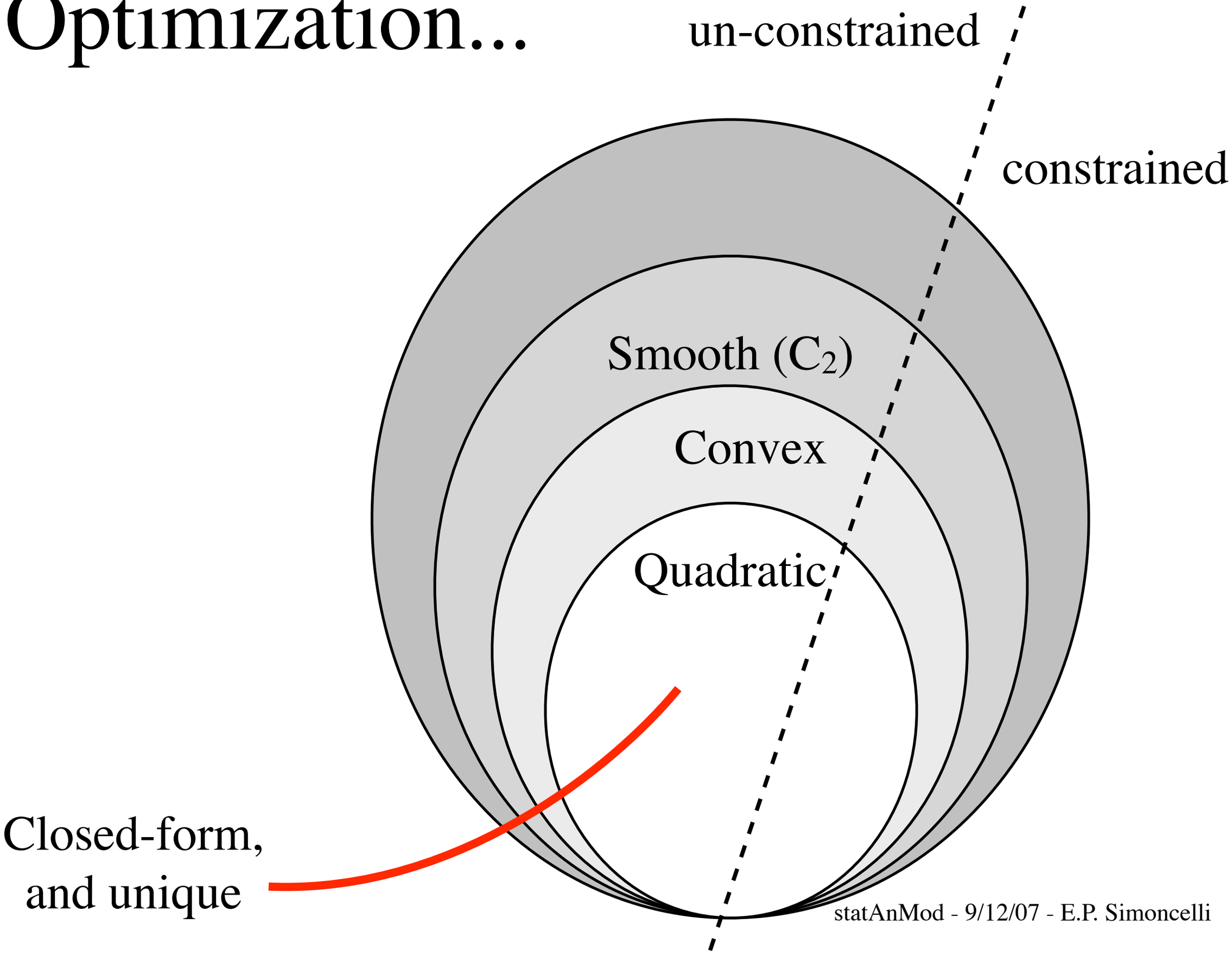
Optimization...



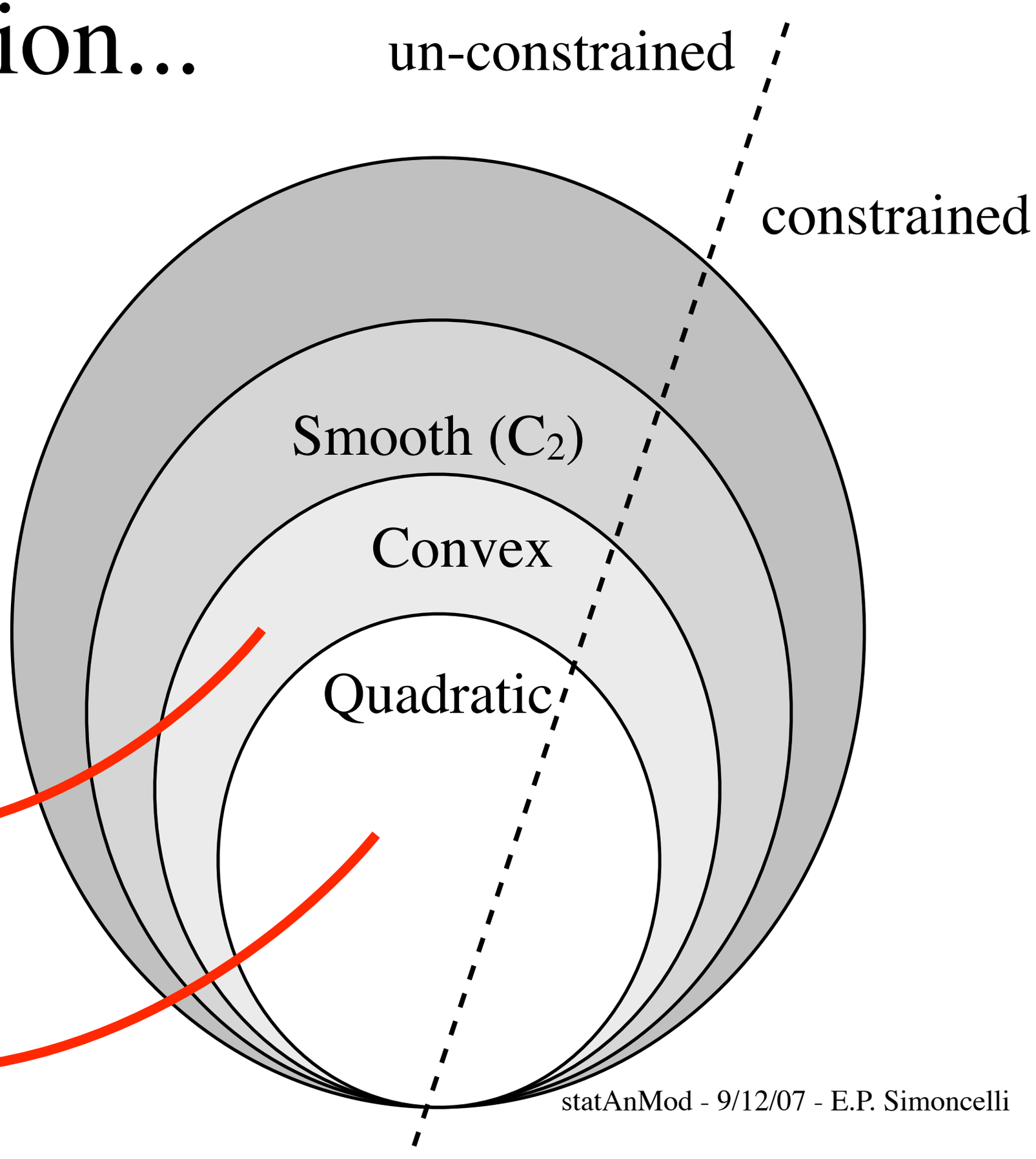
Optimization...



Optimization...



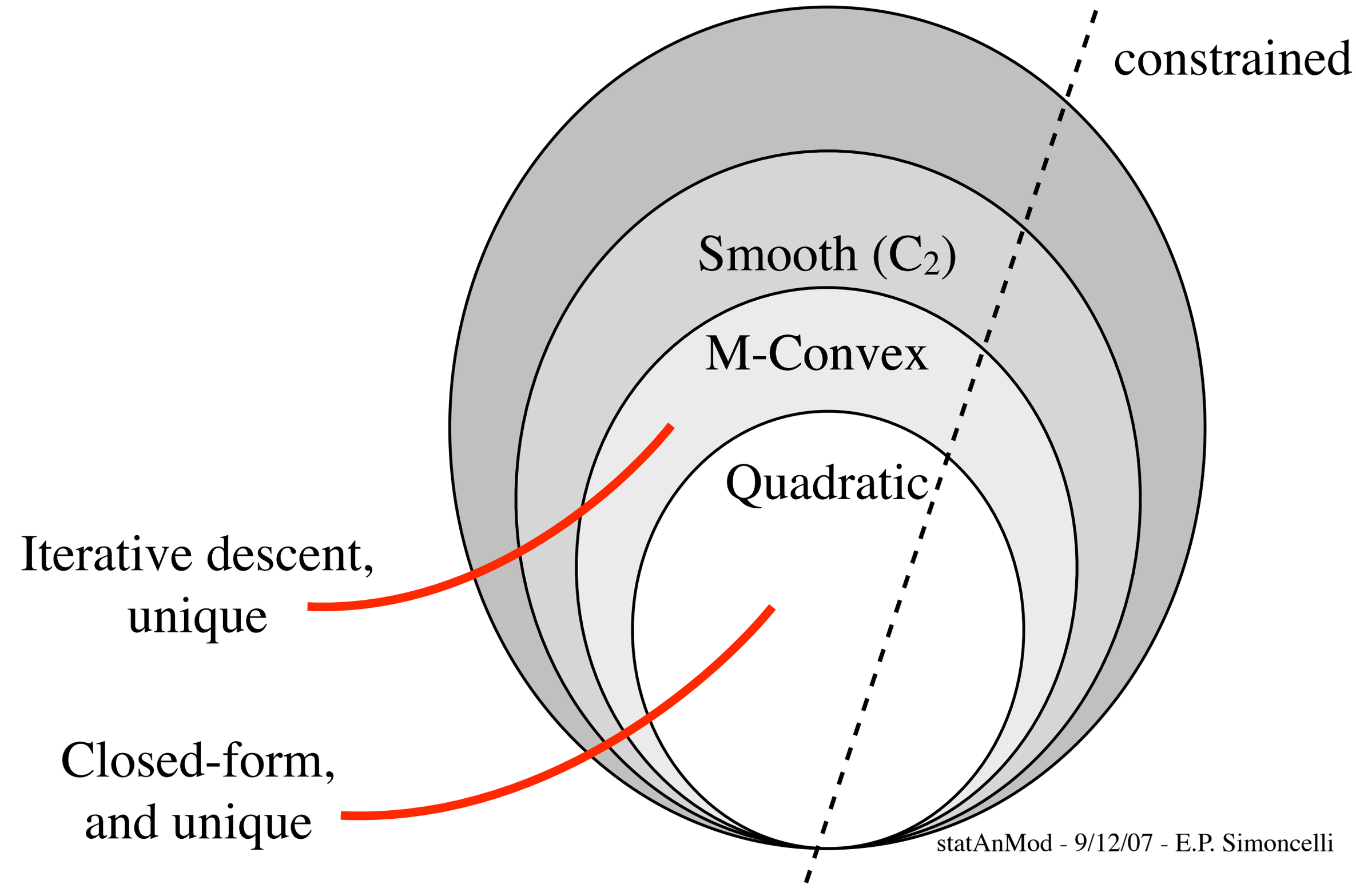
Optimization...



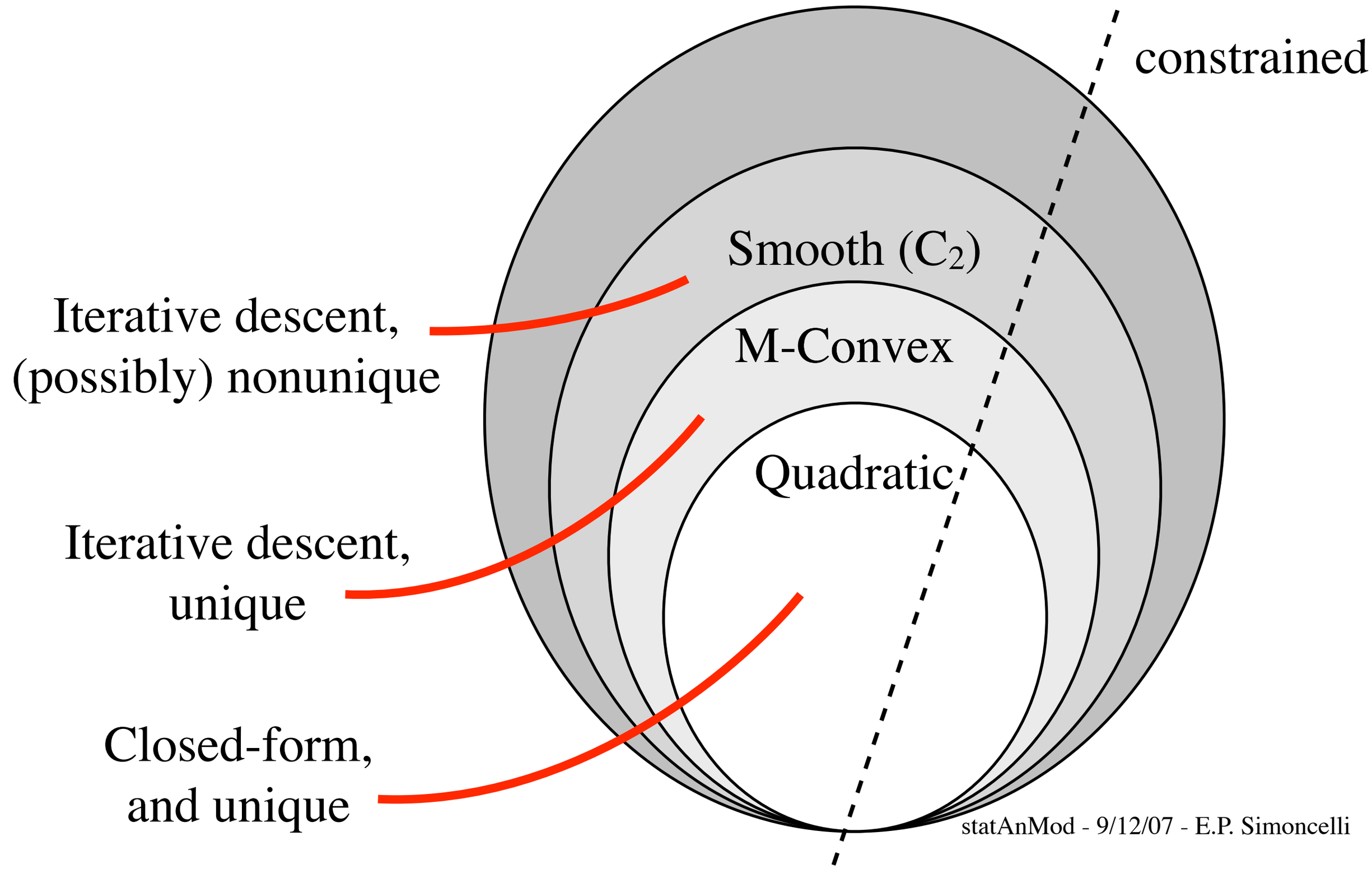
Iterative descent, unique

Closed-form, and unique

Optimization...



Optimization...



Optimization...

un-constrained

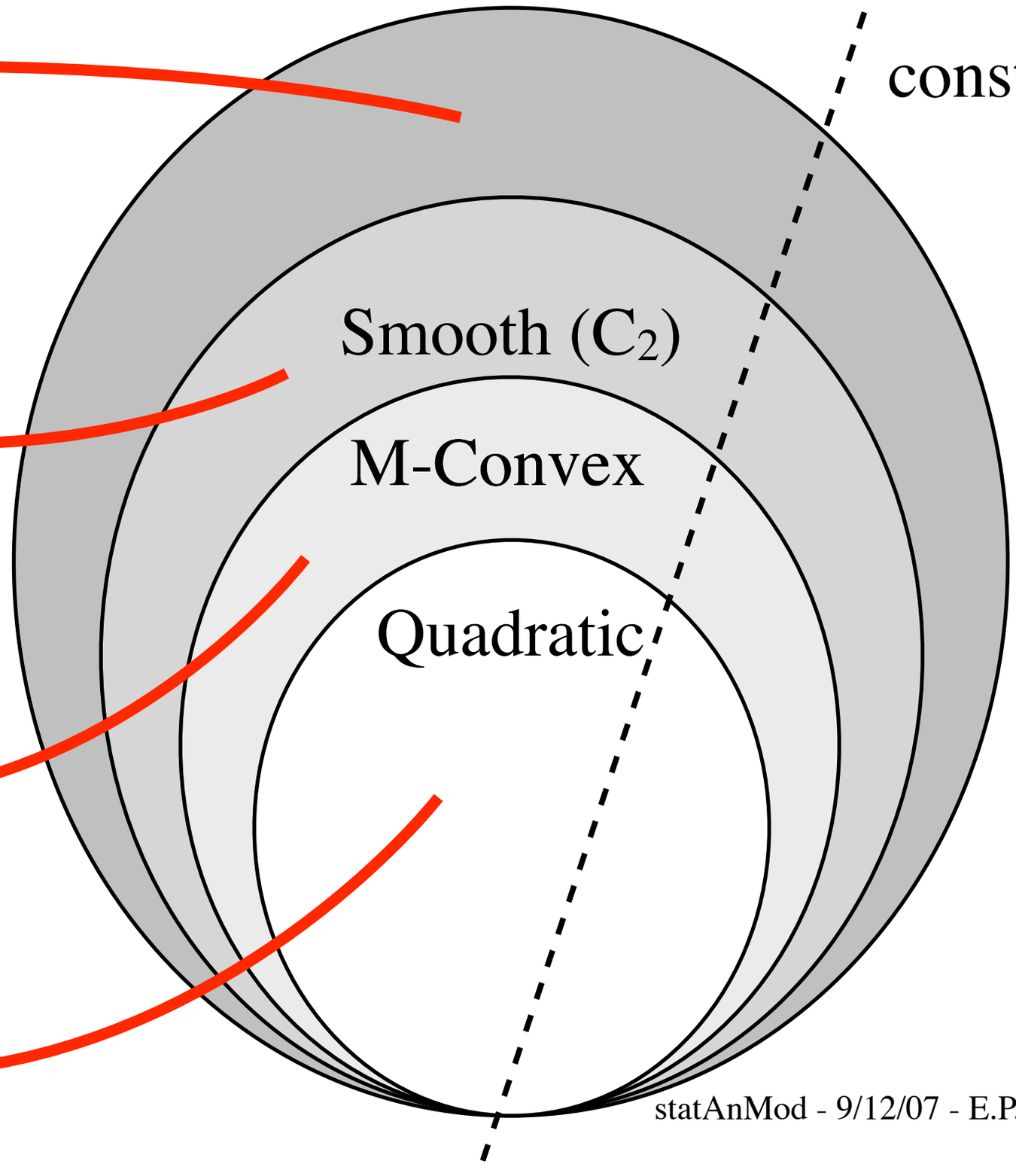
constrained

Heuristics,
exhaustive search,
(pain & suffering)

Iterative descent,
(possibly) nonunique

Iterative descent,
unique

Closed-form,
and unique



Linear/Gaussian model

$$n(t) = \vec{k} \cdot \vec{x}(t) + w(t)$$

Quadratic error => unique closed-form solution:

$$\hat{k} = (X^T X)^{-1} X^T \vec{N}$$

where matrix X contains all $\vec{x}(t)$

vector $n(t)$ contains all $\vec{N}$

Linear/Gaussian model

$$n(t) = \vec{k} \cdot \vec{x}(t) + w(t)$$

Quadratic error => unique closed-form solution:

$$\hat{k} = (X^T X)^{-1} X^T \vec{N}$$

where matrix X contains all $\vec{x}(t)$

vector $n(t)$ contains all $\vec{N}$

Nice... but neurons are not linear

and noise is not Gaussian

Polynomial Model

(Volterra/Weiner Kernels)

$$r(\vec{x}) = k_0 + \vec{k_1} \cdot \vec{x} + \vec{x}^T K_2 \vec{x} + \ldots$$

Polynomial Model

(Volterra/Weiner Kernels)

$$r(\vec{x}) = k_0 + \vec{k}_1 \cdot \vec{x} + \vec{x}^T K_2 \vec{x} + \ldots$$

const

□

Polynomial Model

(Volterra/Weiner Kernels)

$$r(\vec{x}) = k_0 + \vec{k}_1 \cdot \vec{x} + \vec{x}^T K_2 \vec{x} + \ldots$$

const

vector

Polynomial Model

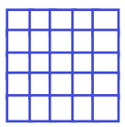
(Volterra/Weiner Kernels)

$$r(\vec{x}) = k_0 + \vec{k}_1 \cdot \vec{x} + \vec{x}^T K_2 \vec{x} + \ldots$$

const

vector

matrix



Polynomial Model

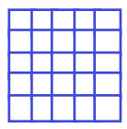
(Volterra/Weiner Kernels)

$$r(\vec{x}) = k_0 + \vec{k}_1 \cdot \vec{x} + \vec{x}^T K_2 \vec{x} + \ldots$$

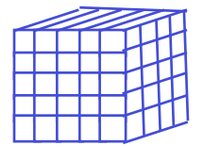
const

vector

matrix



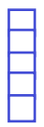
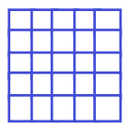
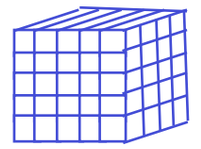
3-tensor



Polynomial Model

(Volterra/Weiner Kernels)

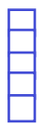
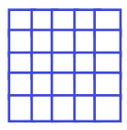
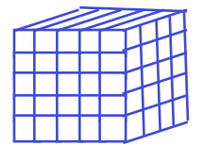
$$r(\vec{x}) = k_0 + \vec{k}_1 \cdot \vec{x} + \vec{x}^T K_2 \vec{x} + \ldots$$

	const	vector	matrix	3-tensor
				
# pars:	1	n (20)	n^2 (400)	n^3 (8000)

Polynomial Model

(Volterra/Weiner Kernels)

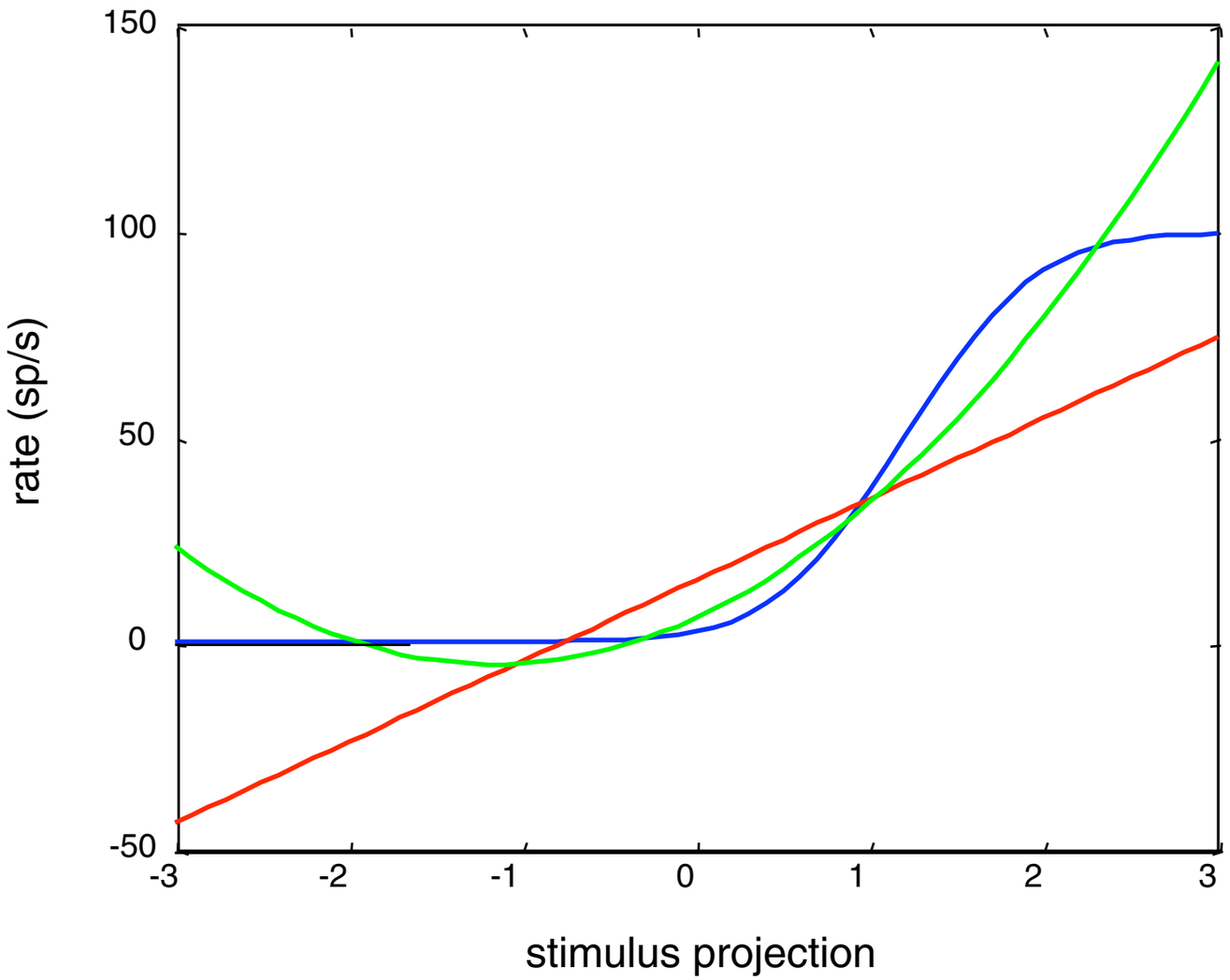
$$r(\vec{x}) = k_0 + \vec{k}_1 \cdot \vec{x} + \vec{x}^T K_2 \vec{x} + \ldots$$

	const	vector	matrix	3-tensor
				
# pars:	1	n (20)	n^2 (400)	n^3 (8000)

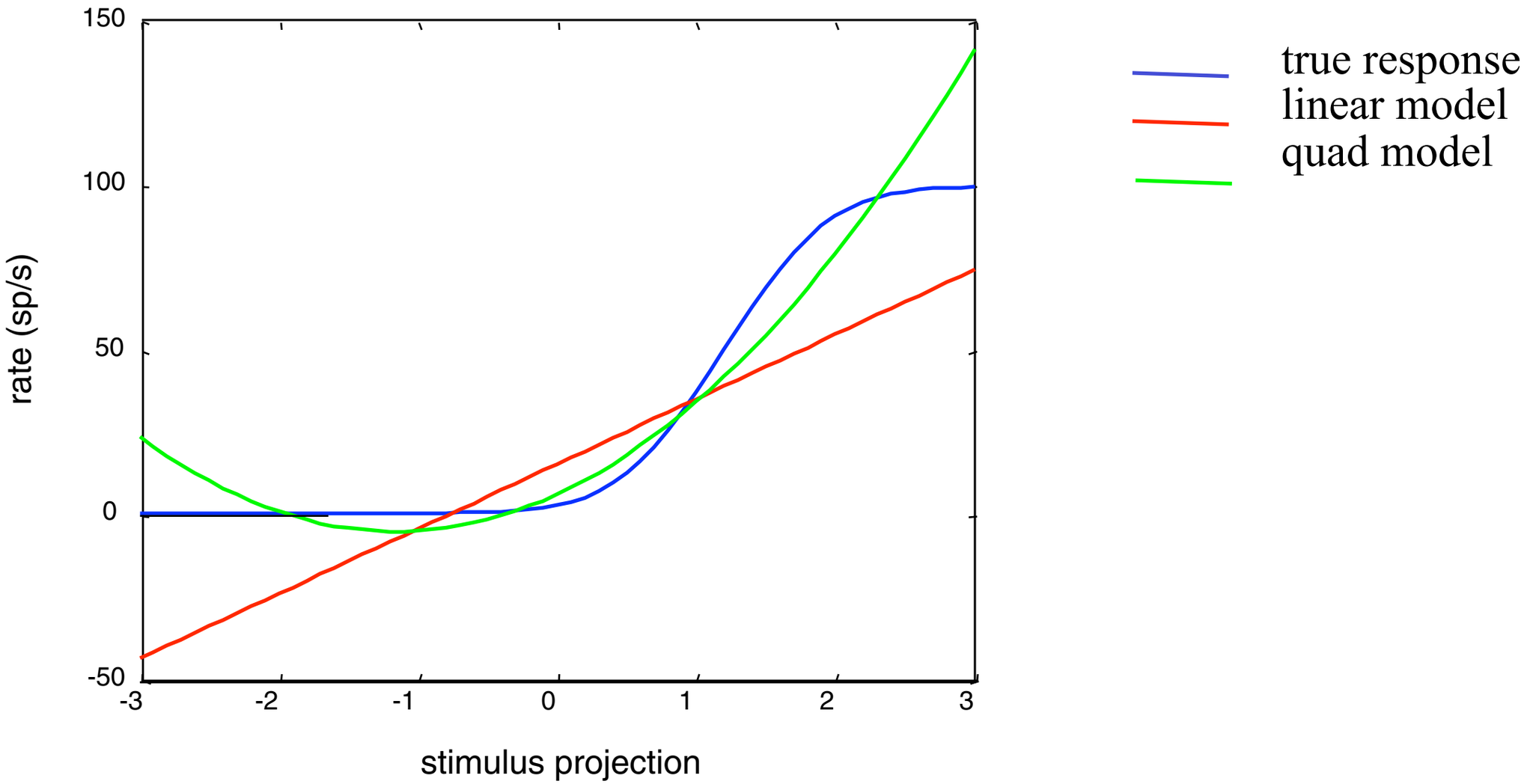
In practice, insufficient data to go beyond 2nd order

Low-order polynomials do a poor job of representing the nonlinearities found in neurons

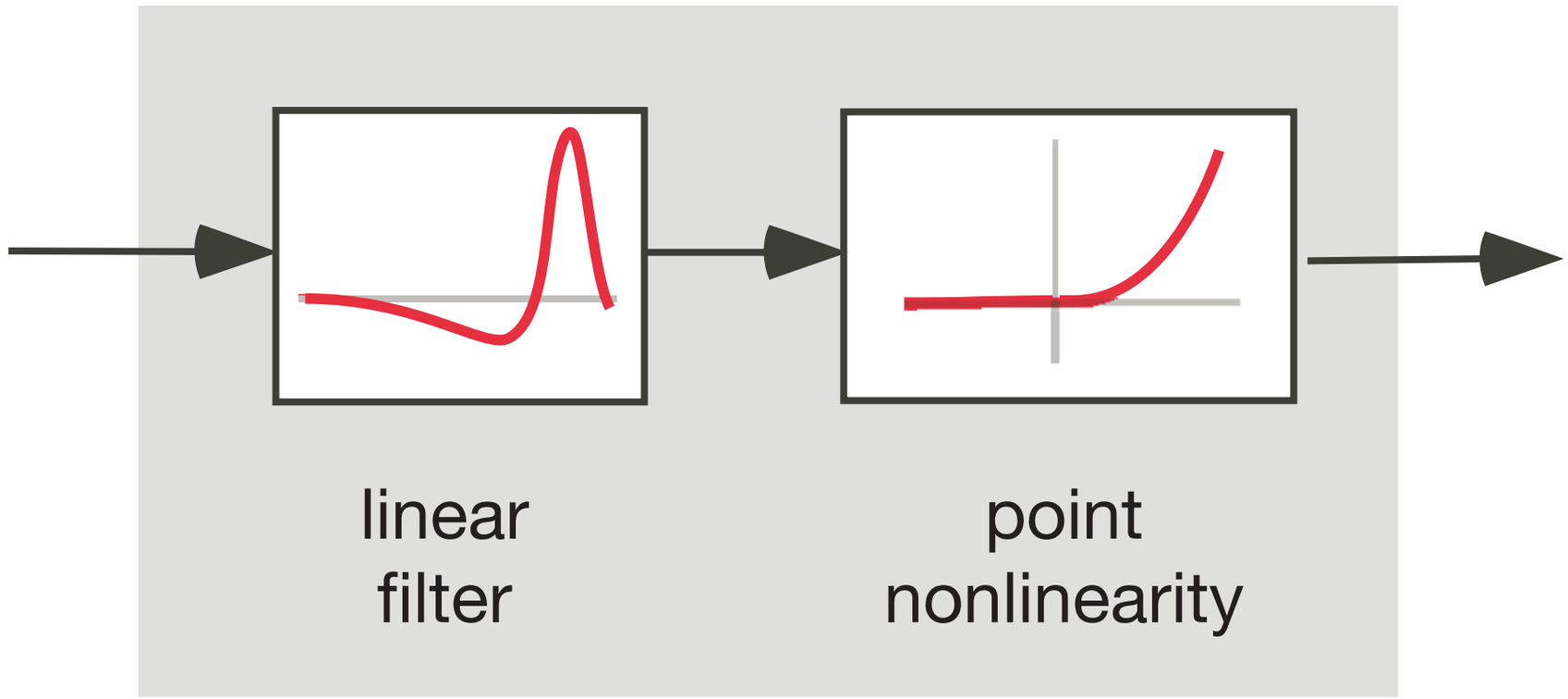
Low-order polynomials do a poor job of representing the nonlinearities found in neurons



Low-order polynomials do a poor job of representing the nonlinearities found in neurons

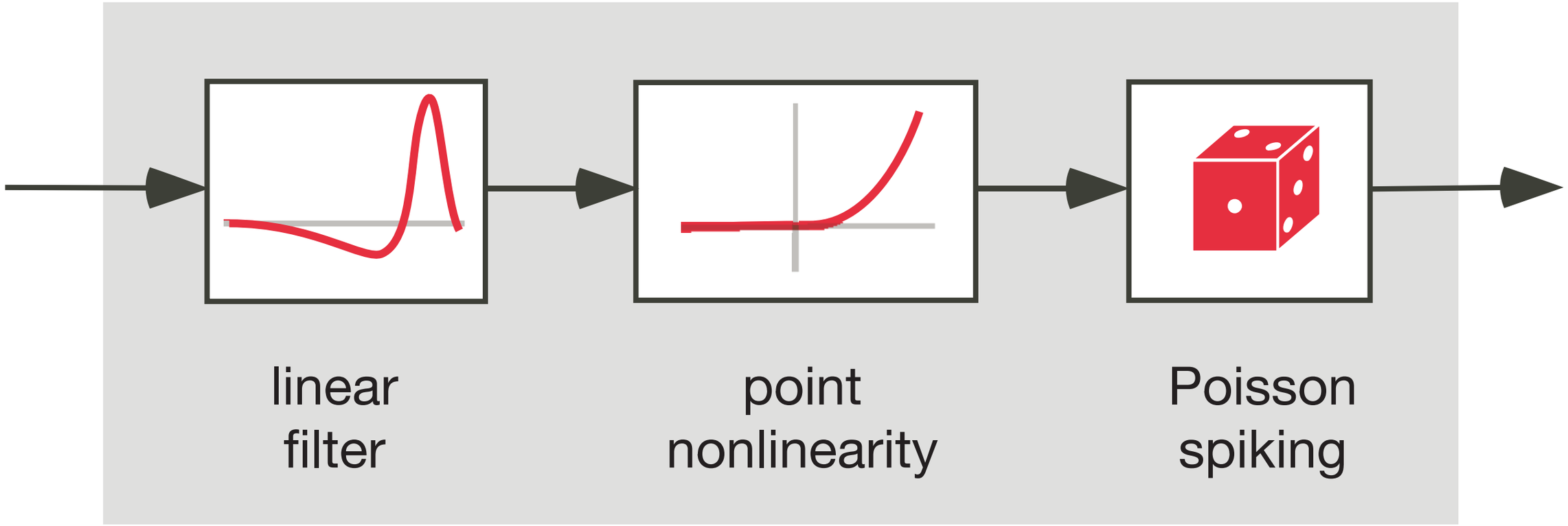


LN cascade model



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

LNP cascade model



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

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 observed data, $\{n(t), \vec{x}(t)\}$)

[Paninski, '04]

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 observed 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]

ML estimation of LNP

[on board]

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 observed data, $\{n(t), \vec{x}(t)\}$)

[Paninski, '04]

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 observed 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]