

Generalized Linear Models (GLM)

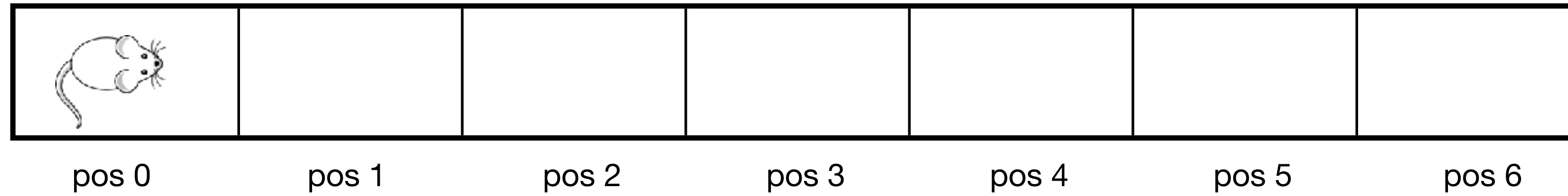
A conceptual introduction to GLM

Roadmap

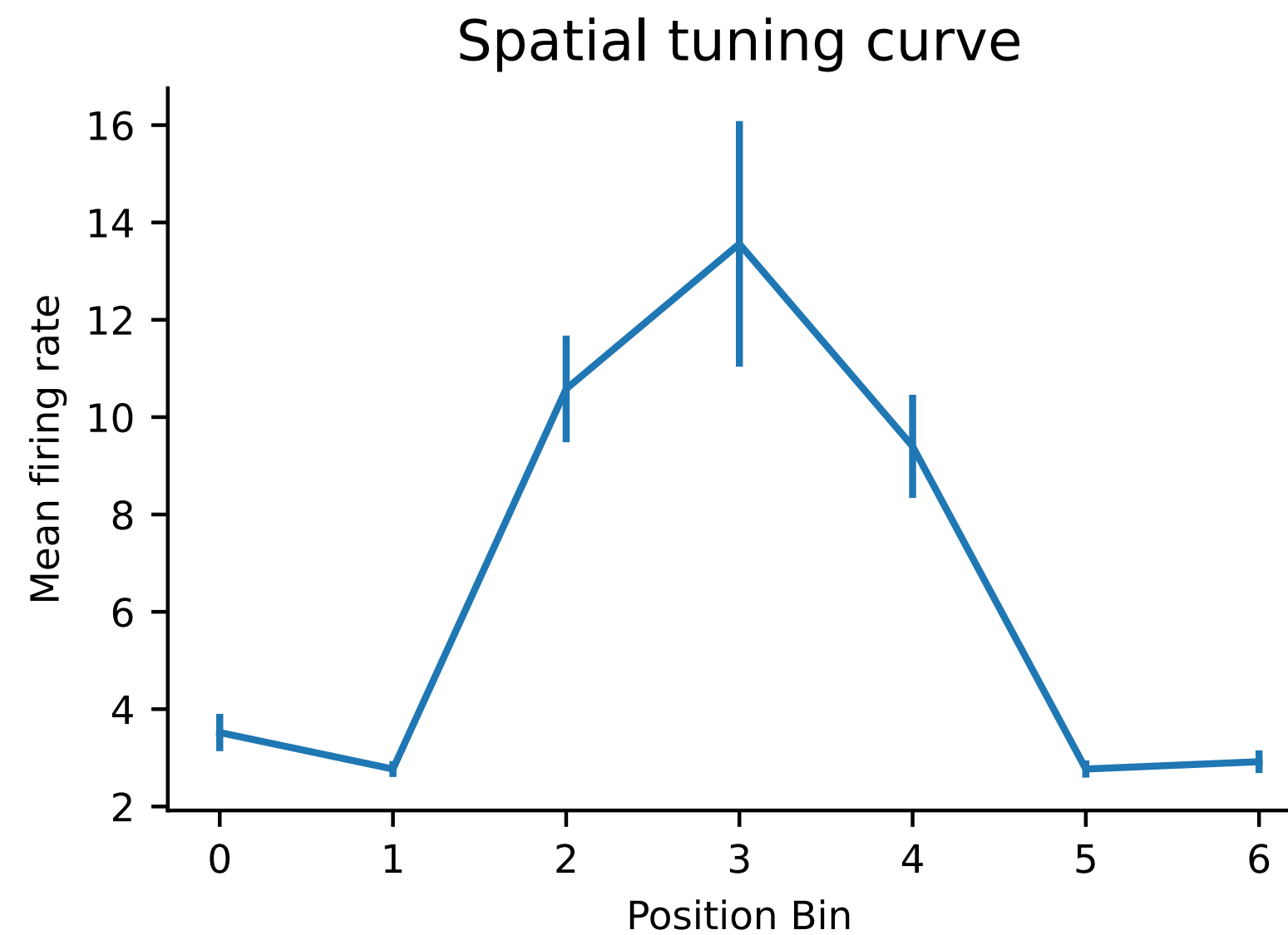
- Why models?
- What are GLMs?
- Why GLMs?
- What can I do with a GLM?
- GLMs In NeMoS
- What features can/should I use?
- Feature construction with Basis
- Summary
- Today's roadmap

Why models? A hook

linear maze



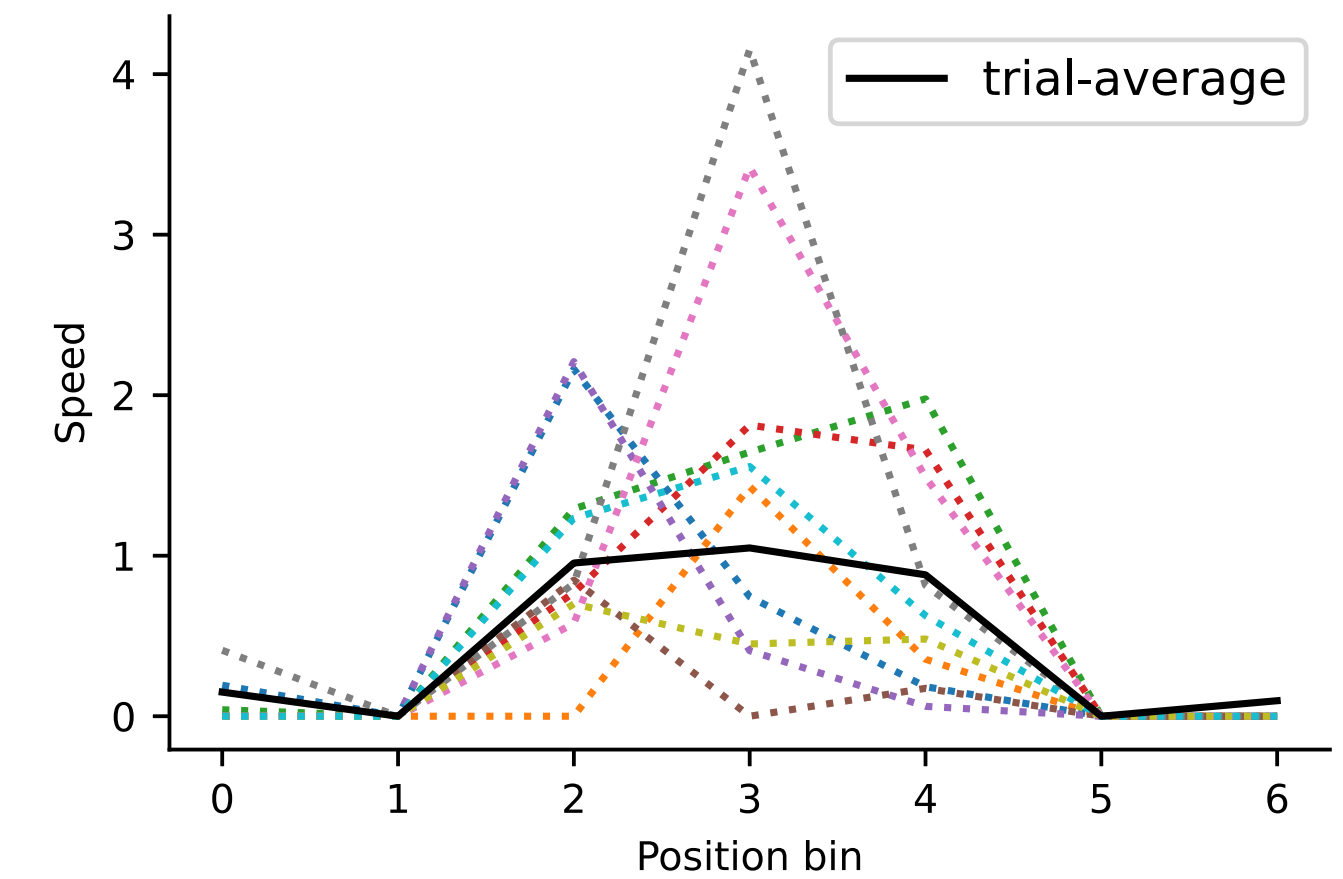
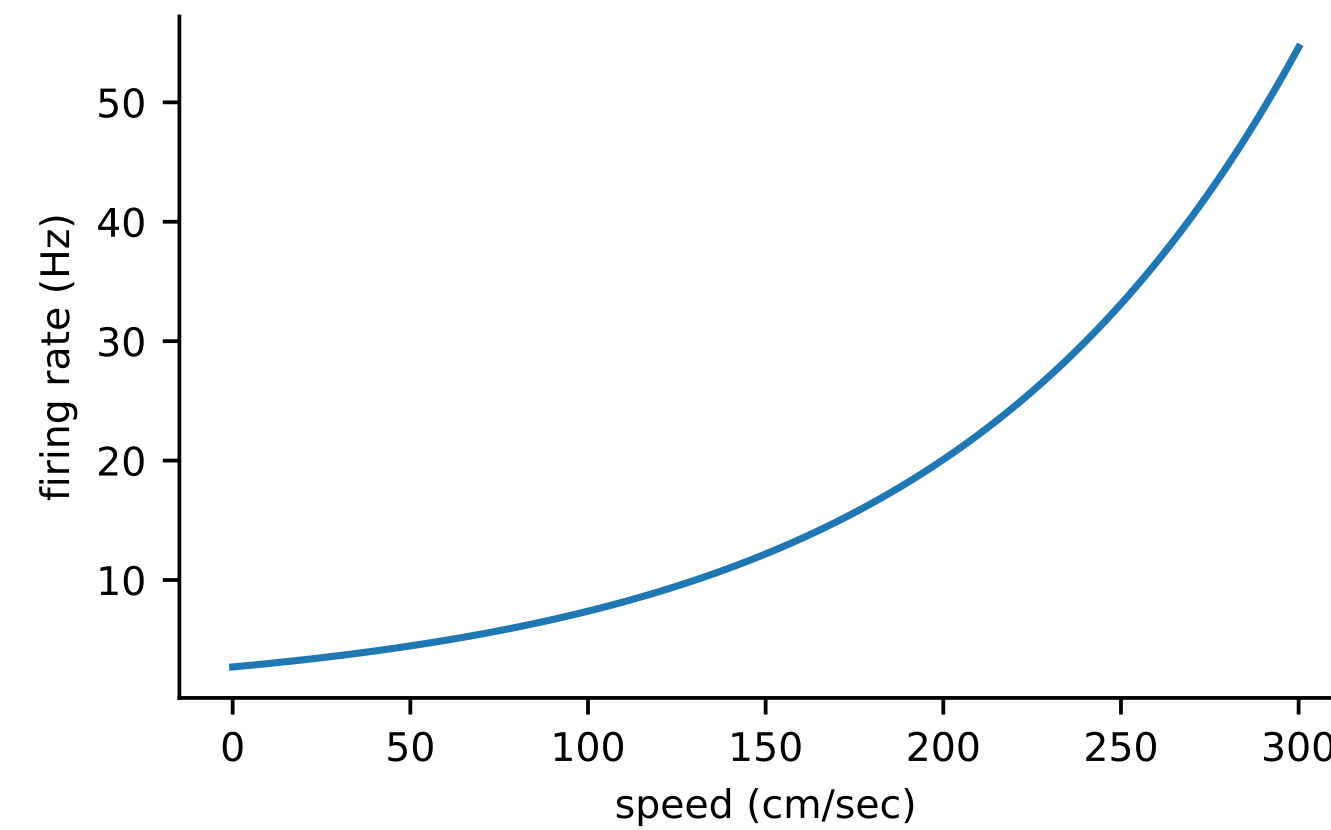
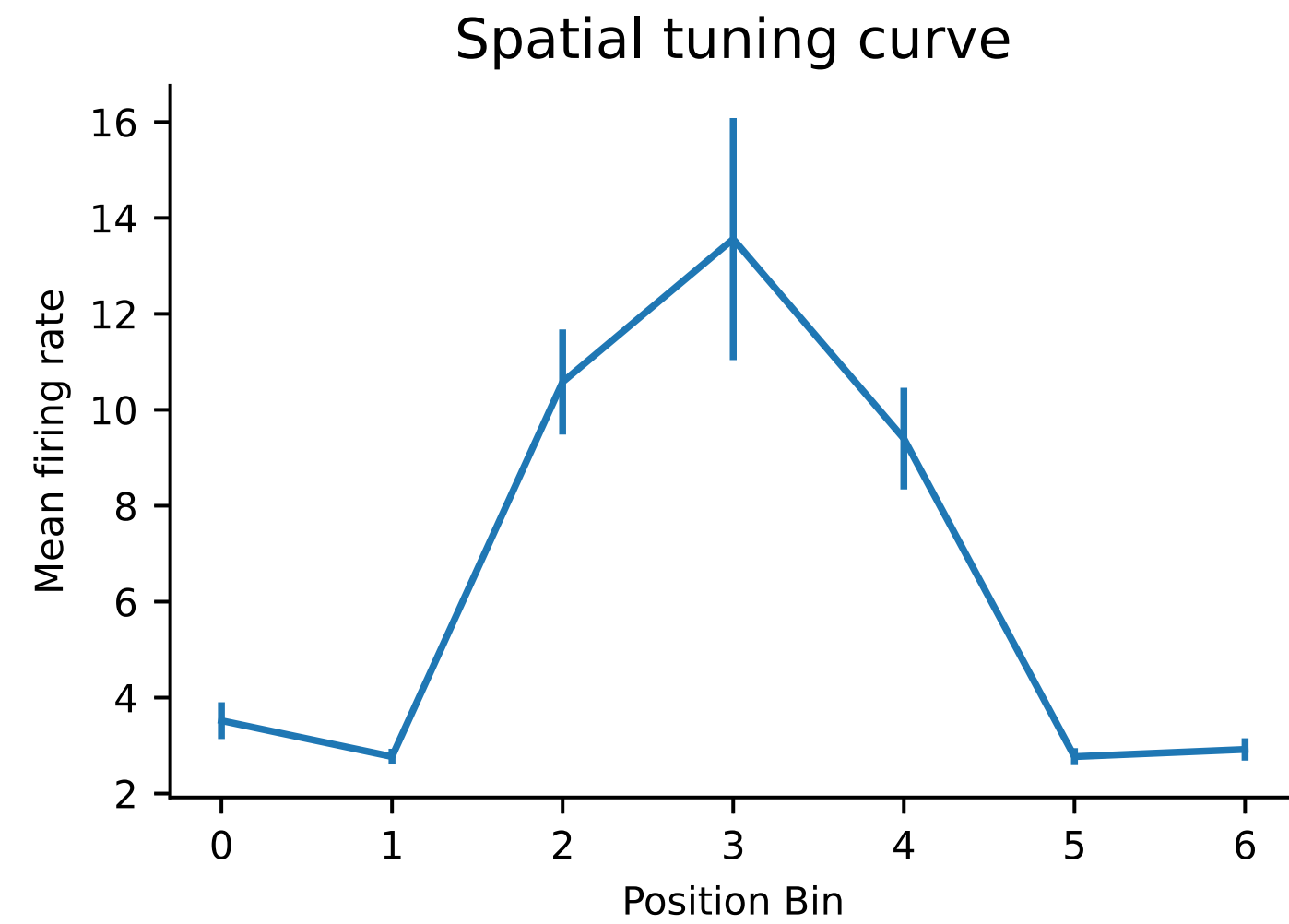
is this neuron encoding the mouse position?



Why models? A hook

..actually, not!

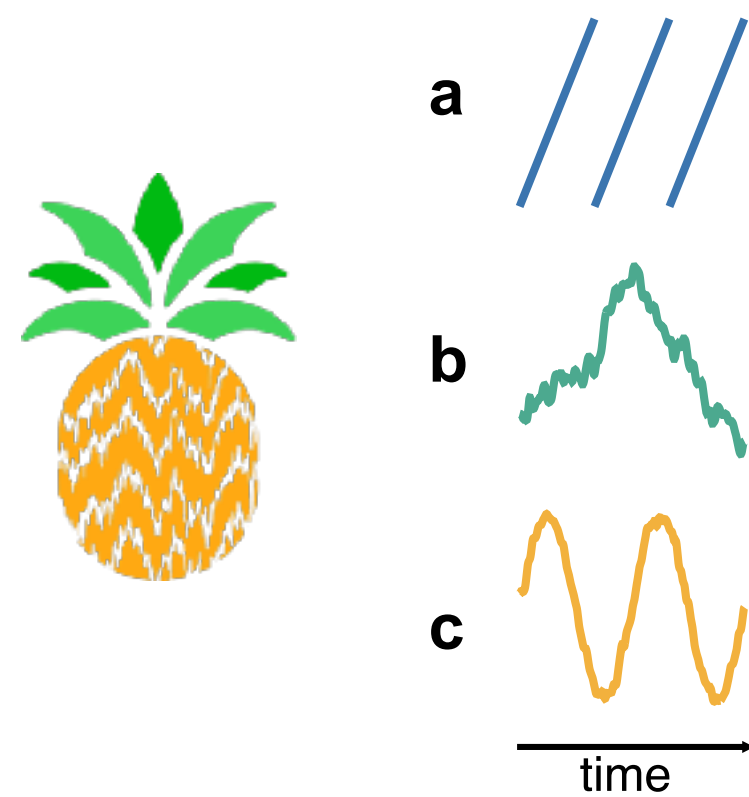
position and speed
are correlated



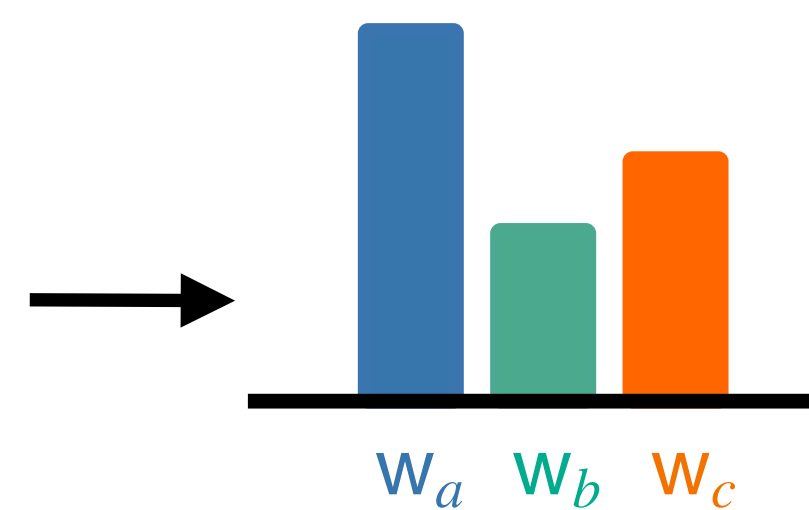
tuning functions don't tell you the whole story
need better models!

What are GLMs?

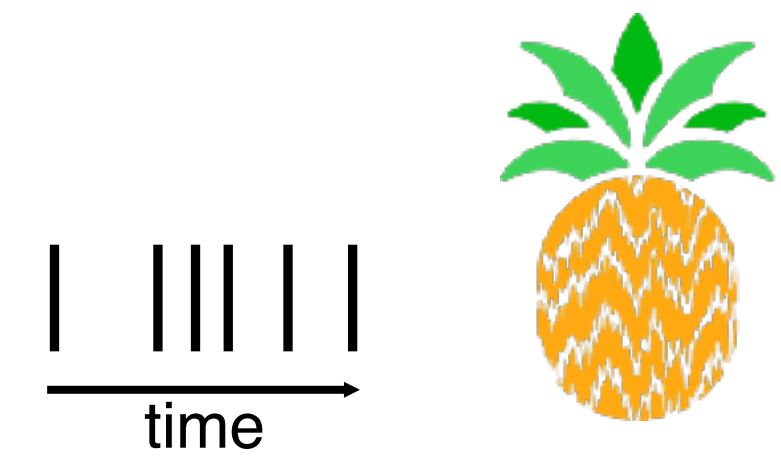
Pre-process



Weights



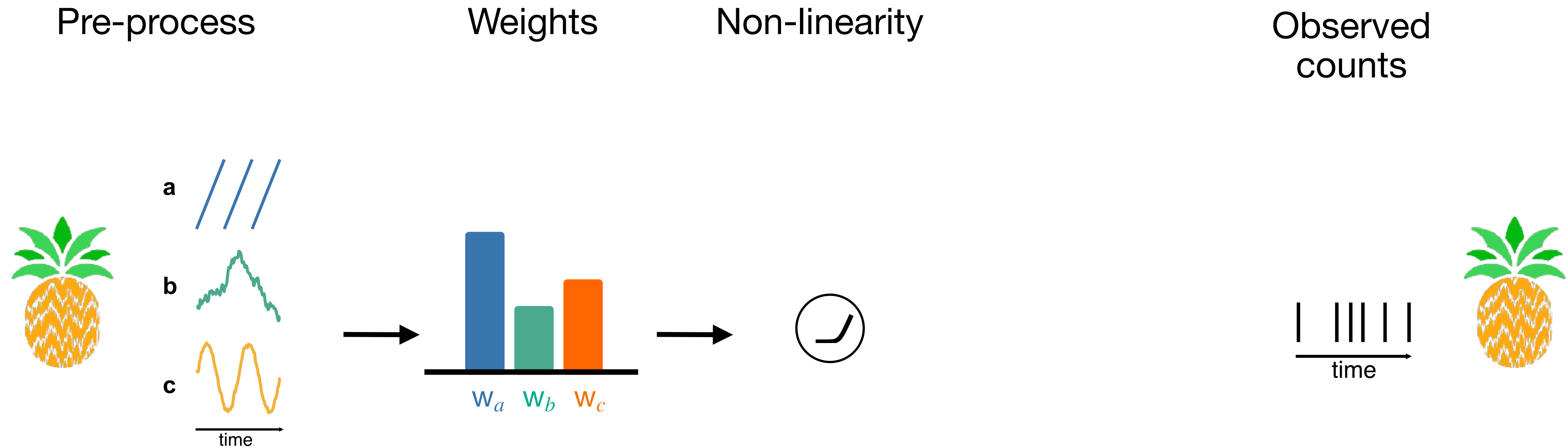
Observed counts



scale the inputs by some weights

$$\mathbf{a} \cdot w_a + \mathbf{b} \cdot w_b + \mathbf{c} \cdot w_c$$

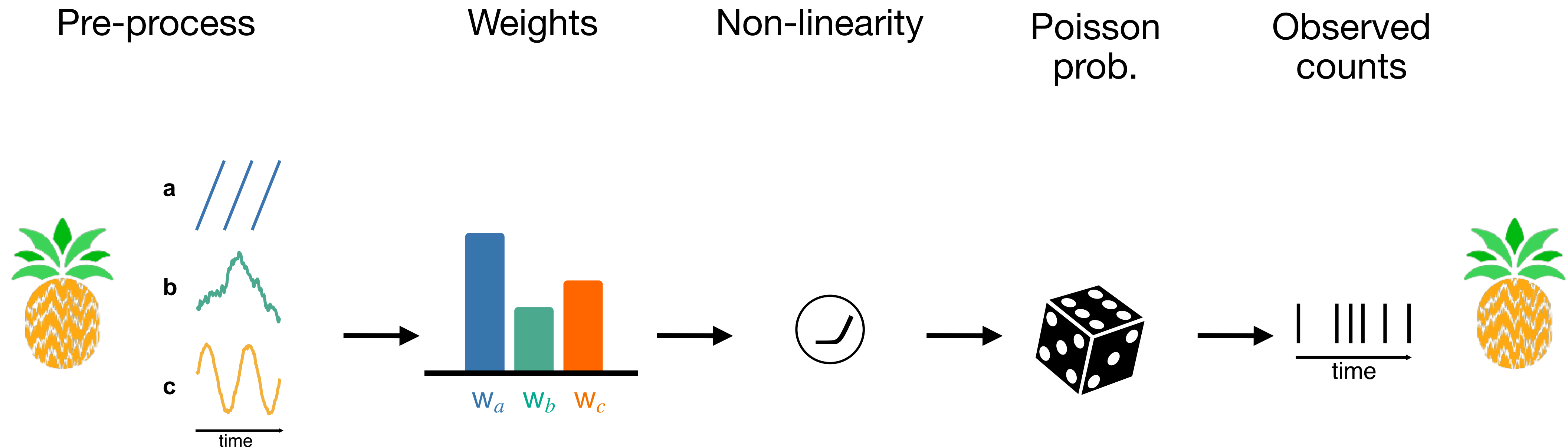
What are GLMs?



non-linearity to make the result positive

$$\text{firing rate} = \exp(\mathbf{a} \cdot \mathbf{w}_a + \mathbf{b} \cdot \mathbf{w}_b + \mathbf{c} \cdot \mathbf{w}_c)$$

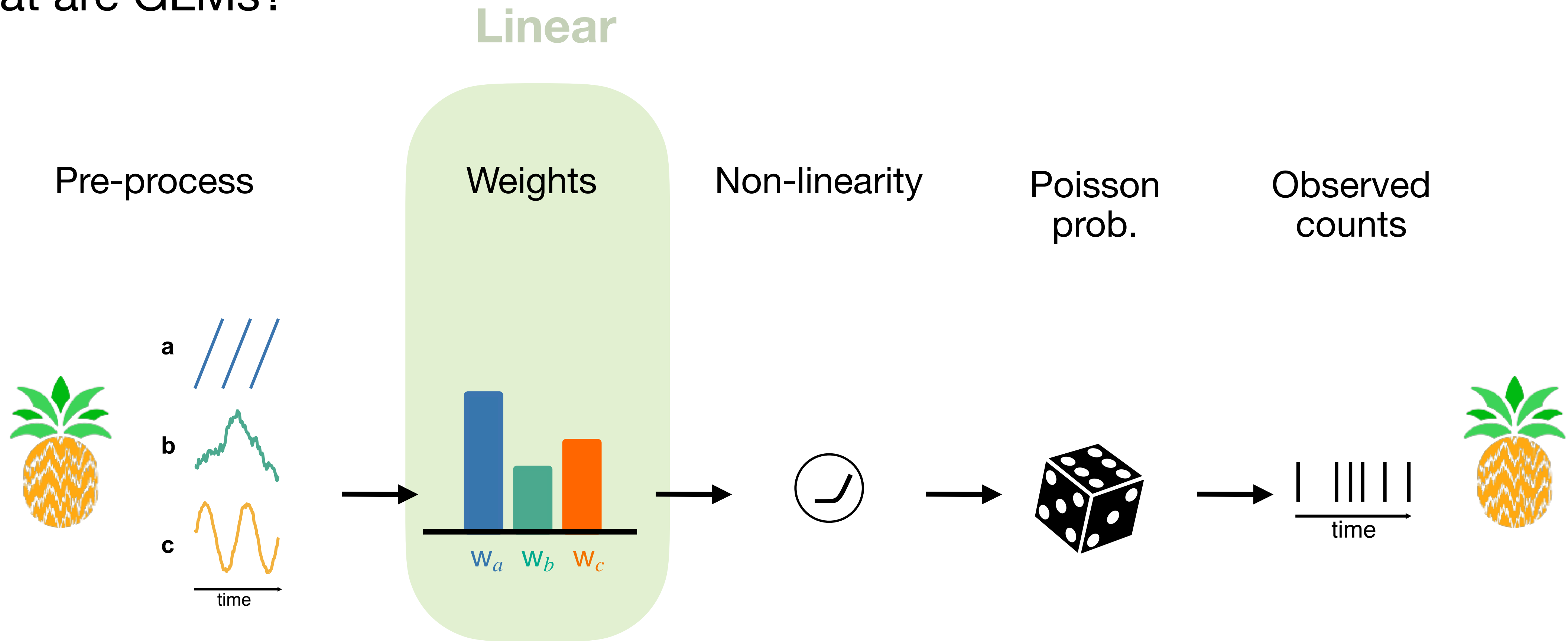
What are GLMs?



$$\text{firing rate} = \exp(\mathbf{a} \cdot \mathbf{w}_a + \mathbf{b} \cdot \mathbf{w}_b + \mathbf{c} \cdot \mathbf{w}_c)$$

$$\text{probability}(\text{spike count} = k) = \text{Poisson}(k \mid \text{firing rate})$$

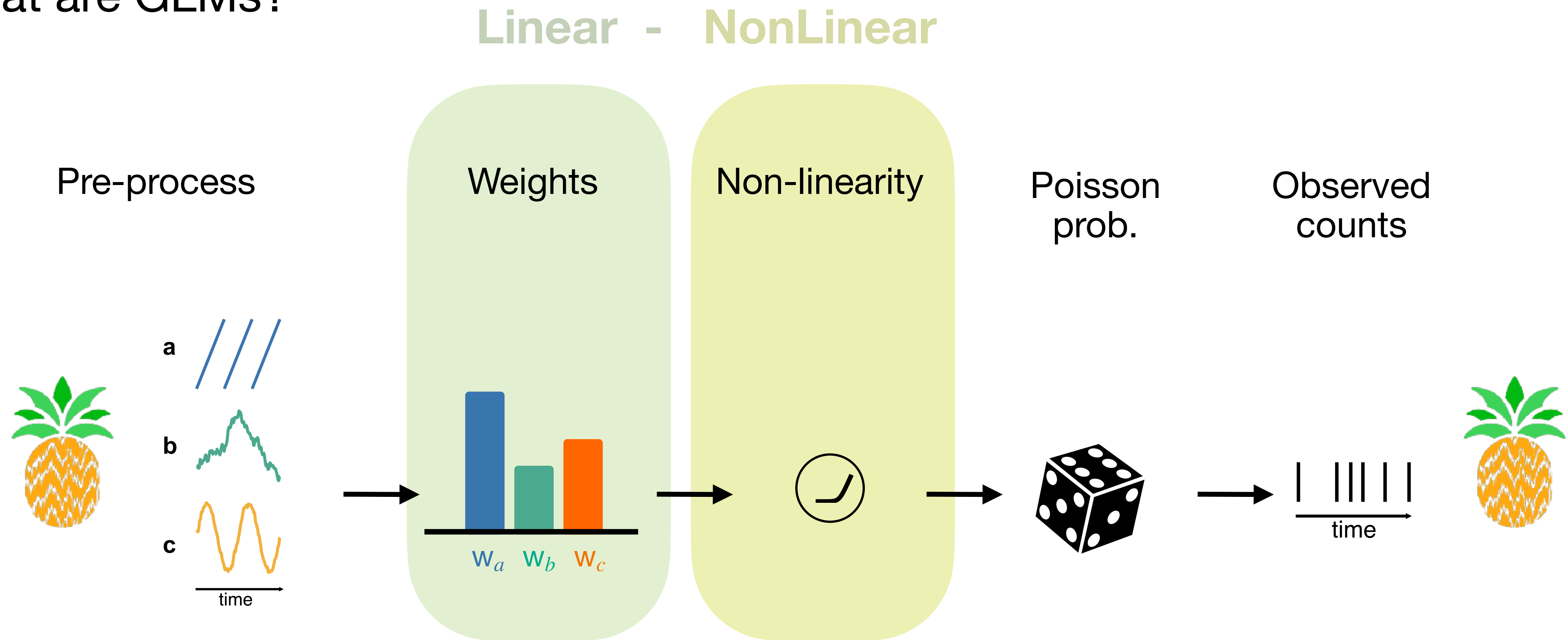
What are GLMs?



$$\text{firing rate} = \exp(\mathbf{a} \cdot \mathbf{w}_a + \mathbf{b} \cdot \mathbf{w}_b + \mathbf{c} \cdot \mathbf{w}_c)$$

$$\text{probability}(\text{spike count} = k) = \text{Poisson}(k \mid \text{firing rate})$$

What are GLMs?

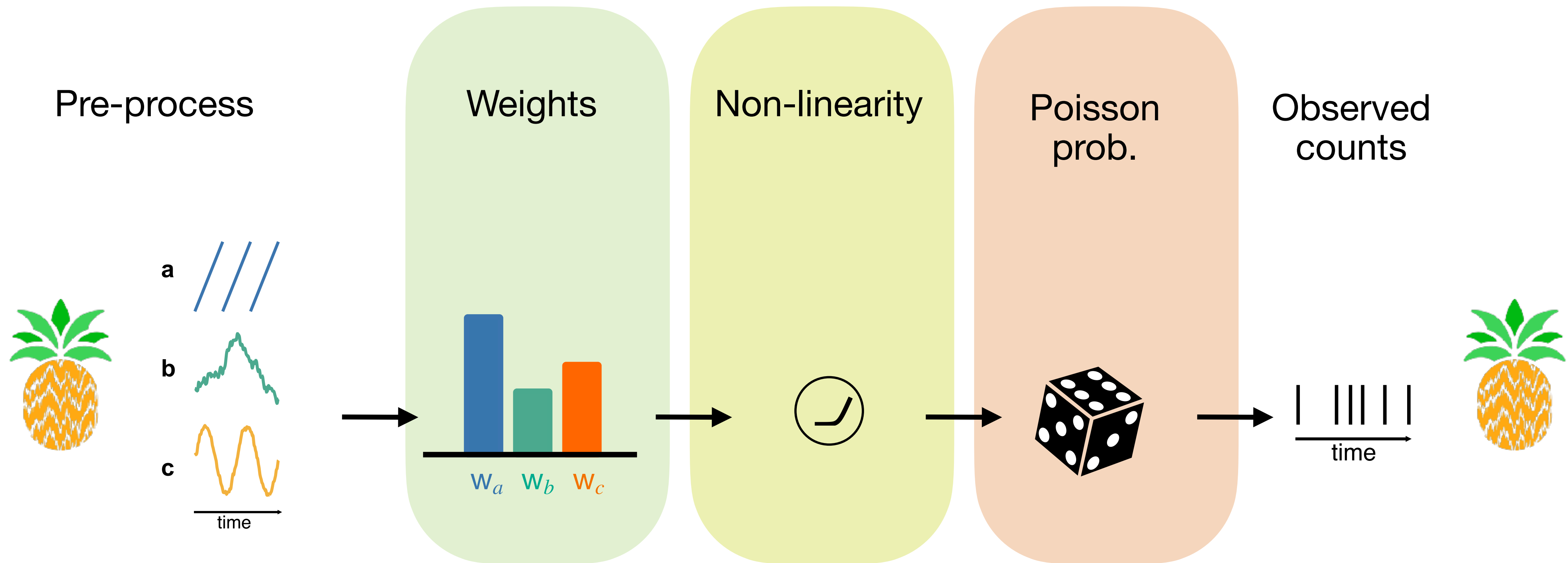


$$\text{firing rate} = \exp(\mathbf{a} \cdot \mathbf{w}_a + \mathbf{b} \cdot \mathbf{w}_b + \mathbf{c} \cdot \mathbf{w}_c)$$

$$\text{probability}(\text{spike count} = k) = \text{Poisson}(k \mid \text{firing rate})$$

What are GLMs?

Linear - NonLinear - Poisson (LNP)



$$\text{firing rate} = \exp(\mathbf{a} \cdot \mathbf{w}_a + \mathbf{b} \cdot \mathbf{w}_b + \mathbf{c} \cdot \mathbf{w}_c)$$

$$\text{probability}(\text{spike count} = k) = \text{Poisson}(k \mid \text{firing rate})$$

Terminology

$$\text{firing rate} = \exp(\mathbf{a} \cdot w_a + \mathbf{b} \cdot w_b + \mathbf{c} \cdot w_c)$$

- \mathbf{a} , \mathbf{b} , \mathbf{c} are called **features** or **predictors**

Terminology

$$\text{firing rate} = \exp(\mathbf{a} \cdot \mathbf{w}_a + \mathbf{b} \cdot \mathbf{w}_b + \mathbf{c} \cdot \mathbf{w}_c)$$

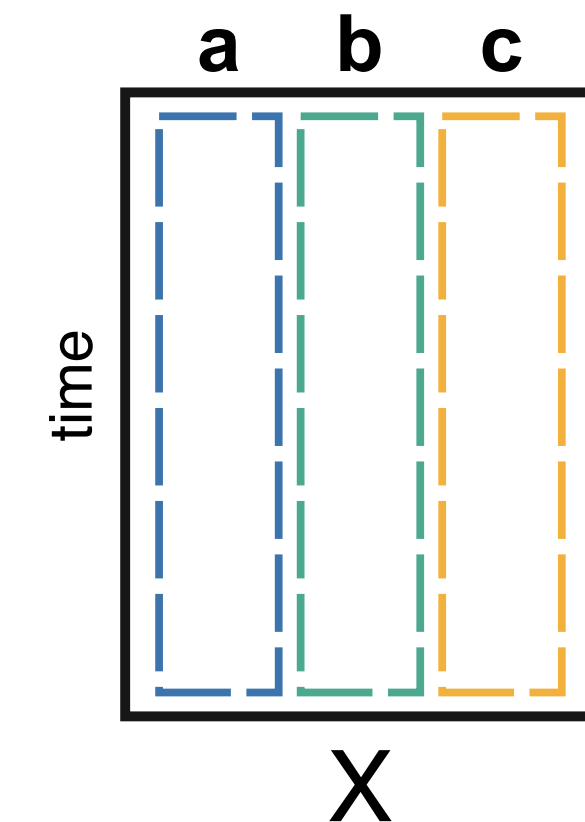
- \mathbf{a} , \mathbf{b} , \mathbf{c} are called **features** or **predictors**
- \mathbf{w}_a , \mathbf{w}_b , \mathbf{w}_c are called **weights** or **coefficients**

Terminology

$$\text{firing rate} = \exp(\mathbf{a} \cdot \mathbf{w}_a + \mathbf{b} \cdot \mathbf{w}_b + \mathbf{c} \cdot \mathbf{w}_c)$$

- \mathbf{a} , \mathbf{b} , \mathbf{c} are called **features** or **predictors**
- \mathbf{w}_a , \mathbf{w}_b , \mathbf{w}_c are called **weights** or **coefficients**
- Features are concatenated to form the **design** or **feature** matrix $\mathbf{X} = [\mathbf{a}, \mathbf{b}, \mathbf{c}]$

Feature matrix

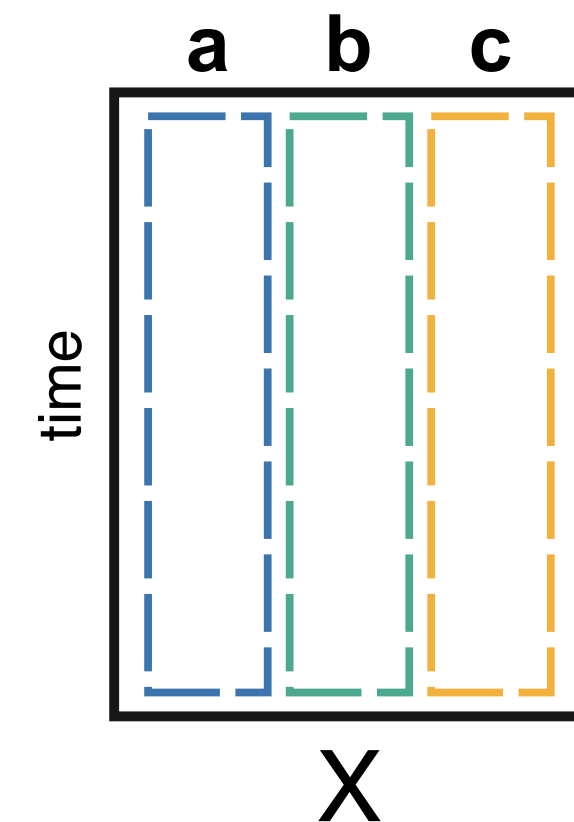


Terminology

$$\text{firing rate} = \exp(\mathbf{a} \cdot \mathbf{w}_a + \mathbf{b} \cdot \mathbf{w}_b + \mathbf{c} \cdot \mathbf{w}_c)$$

- \mathbf{a} , \mathbf{b} , \mathbf{c} are called **features** or **predictors**
- \mathbf{w}_a , \mathbf{w}_b , \mathbf{w}_c are called **weights** or **coefficients**
- Features are concatenated to form the **design** or **feature** matrix $\mathbf{X} = [\mathbf{a}, \mathbf{b}, \mathbf{c}]$
- The **likelihood** is the probability of observing spike counts given some features and weights.

Design matrix



Likelihood

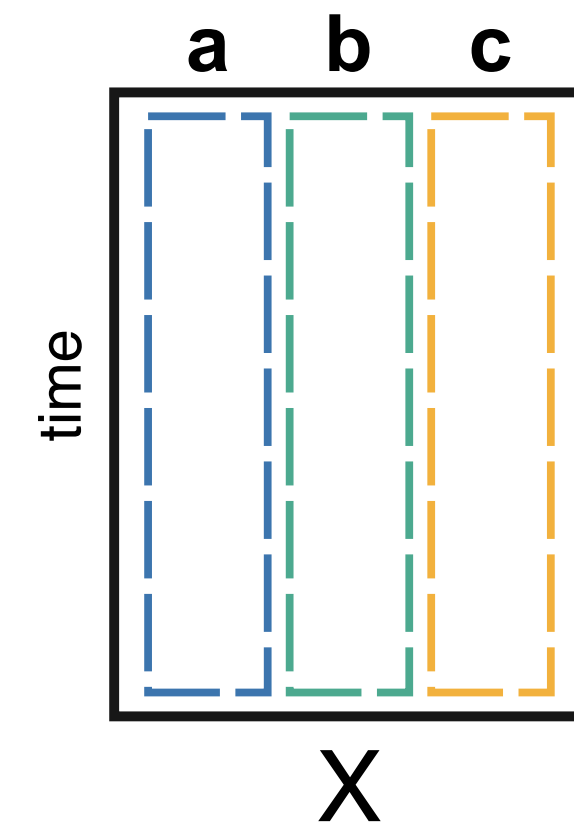
probability(spike count = k | \mathbf{X}, \mathbf{w})

Terminology

$$\text{firing rate} = \exp(\mathbf{a} \cdot \mathbf{w}_a + \mathbf{b} \cdot \mathbf{w}_b + \mathbf{c} \cdot \mathbf{w}_c)$$

- \mathbf{a} , \mathbf{b} , \mathbf{c} are called **features** or **predictors**
- \mathbf{w}_a , \mathbf{w}_b , \mathbf{w}_c are called **weights** or **coefficients**
- Features are concatenated to form the **design** or **feature** matrix $\mathbf{X} = [\mathbf{a}, \mathbf{b}, \mathbf{c}]$
- The **likelihood** is the probability of observing spike counts given some features and weights.
- The **likelihood is a function of the weights** because counts and features are fixed.

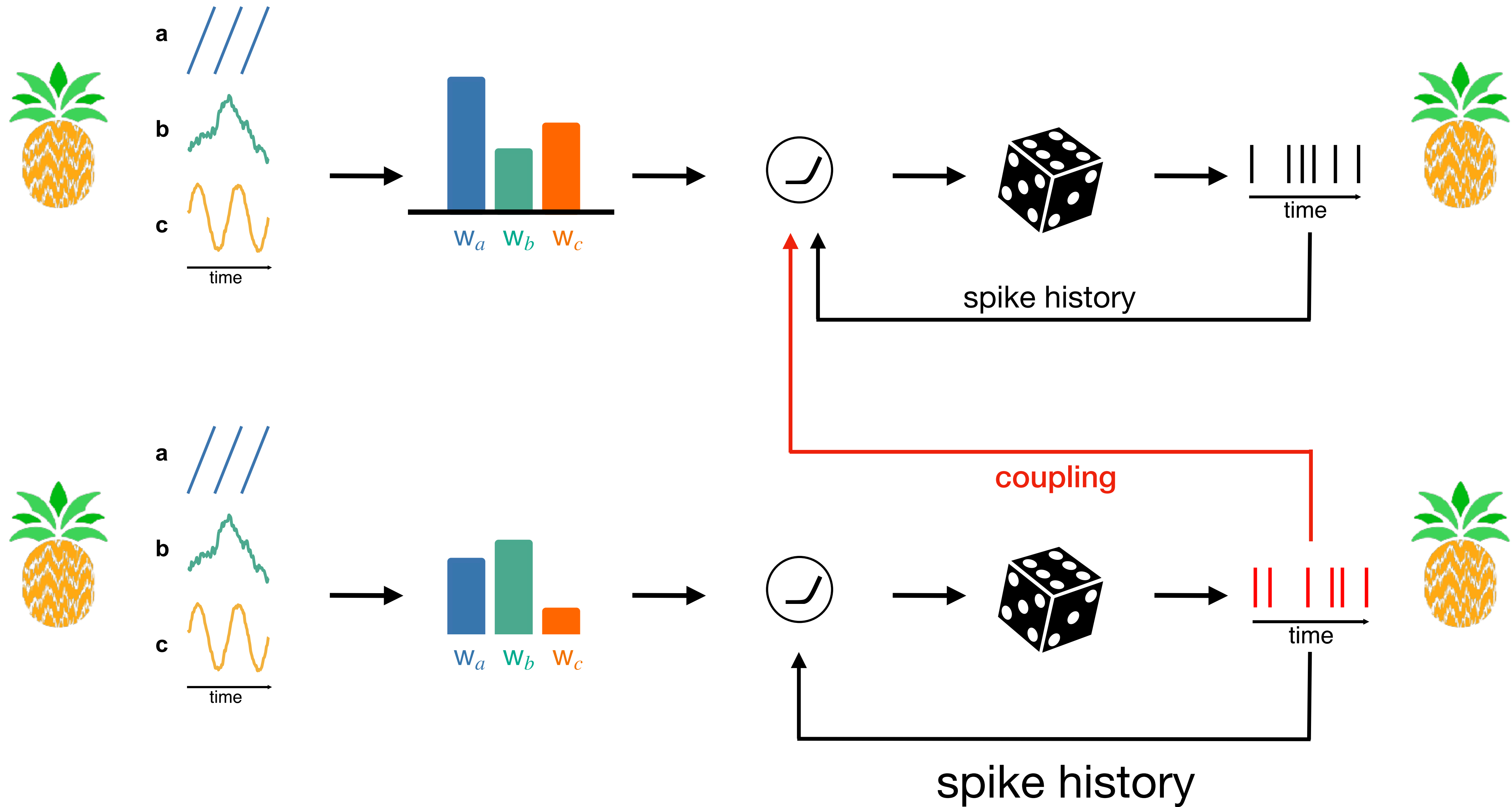
Design matrix



Likelihood

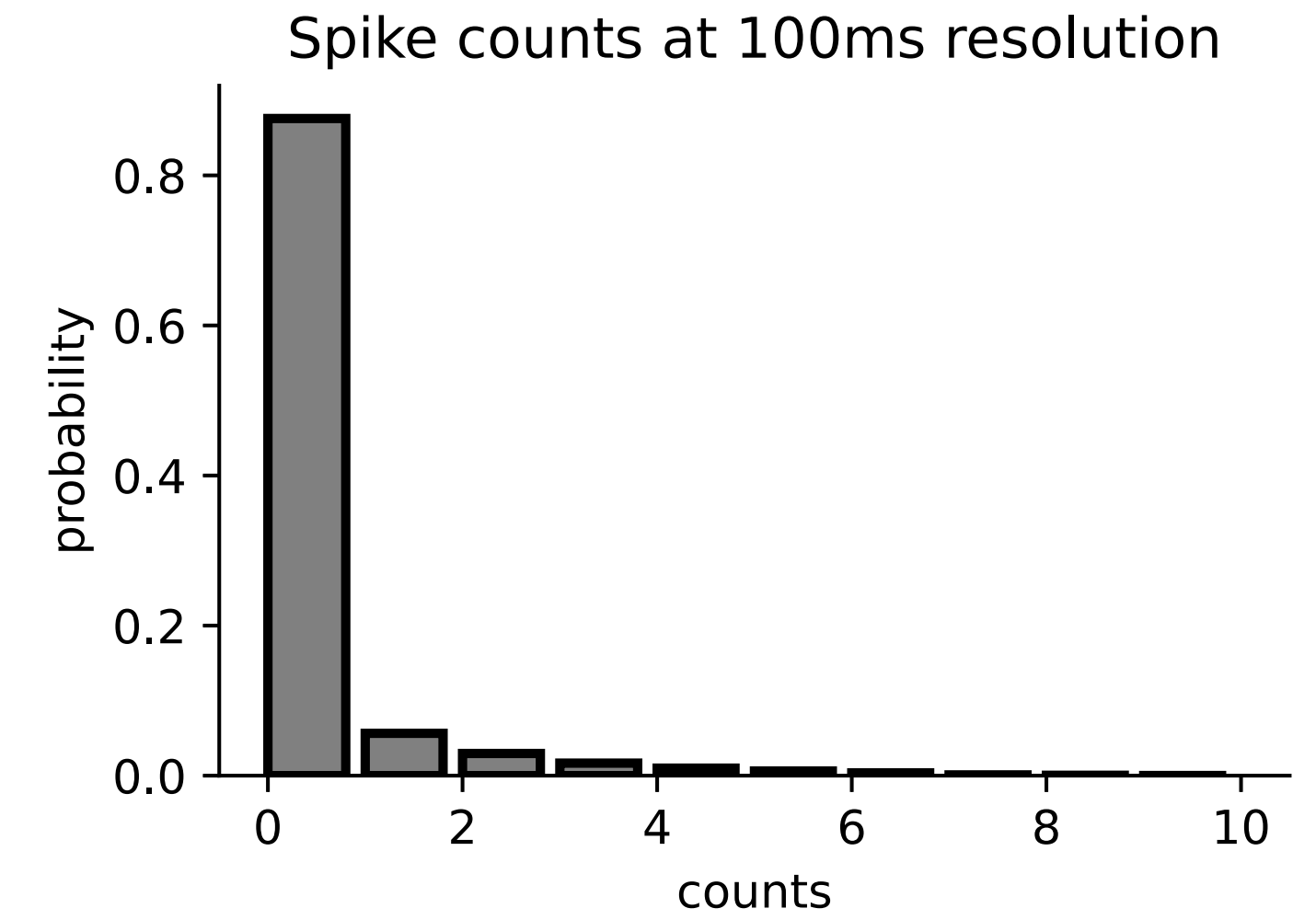
$$\text{probability}(\text{spike count} = k \mid \mathbf{X}, \mathbf{w})$$

What are GLMs?



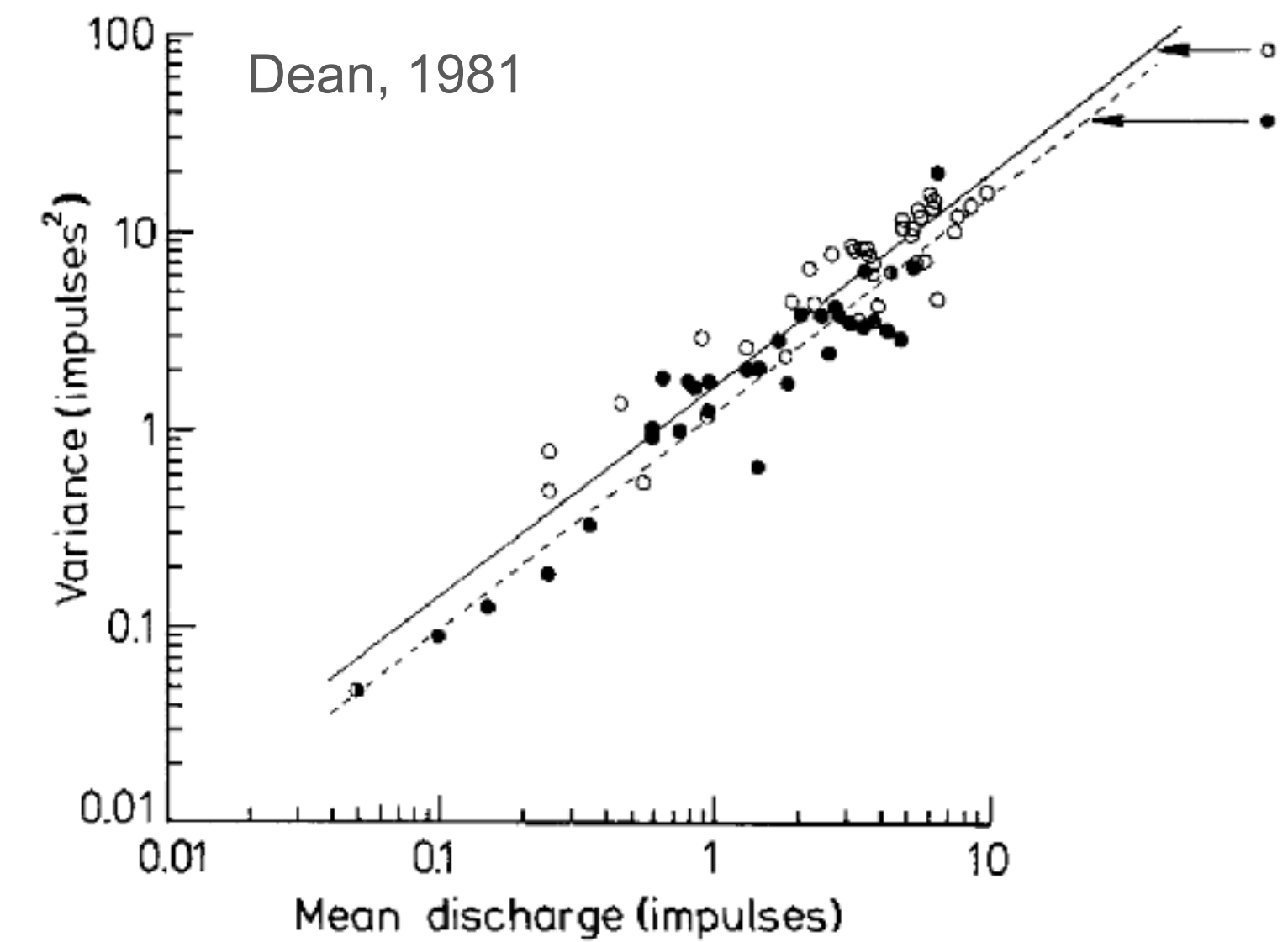
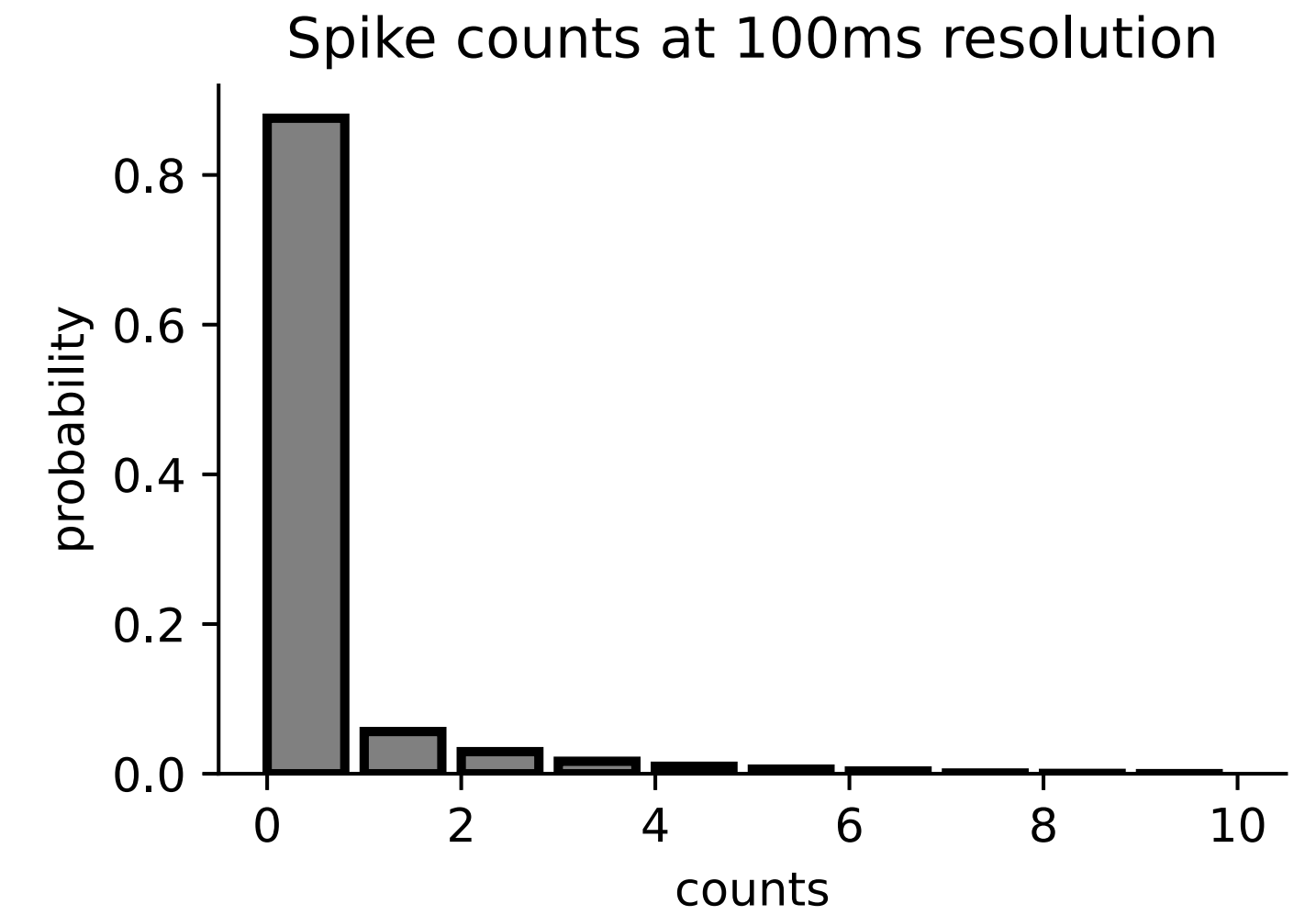
Why GLMs?

1. Why not linear regression?
which assumes normality
 - A. Spike counts are non-Gaussian



Why GLMs?

1. Why not linear regression?
which assumes normality
 - A. Spike counts are non-Gaussian
 - B. Neural activity variance is non-constant



Why GLMs?

1. Why not linear regression?

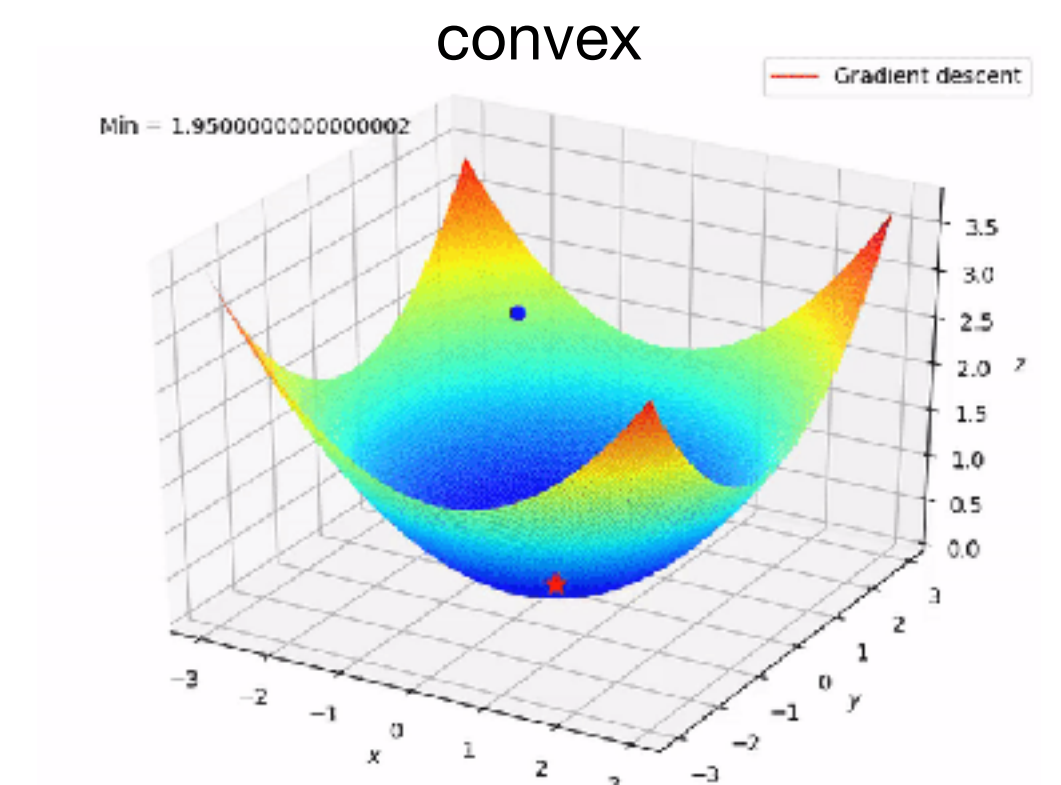
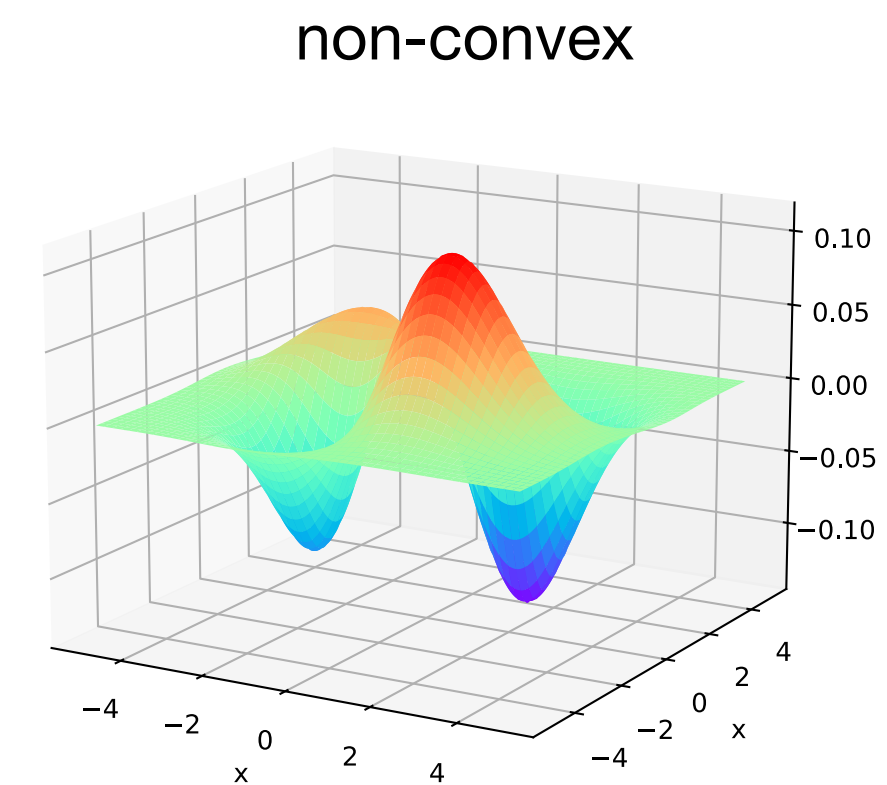
which assumes normality

A. Spike counts are non-Gaussian

B. Neural activity variance is non-constant

2. GLM are as **easy to fit** as linear regression

convex, unique optimal solution



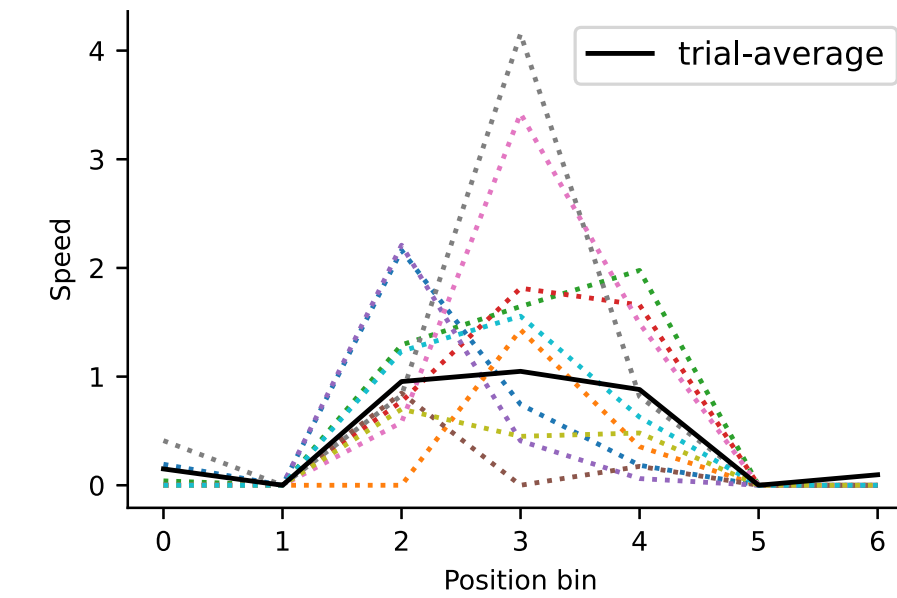
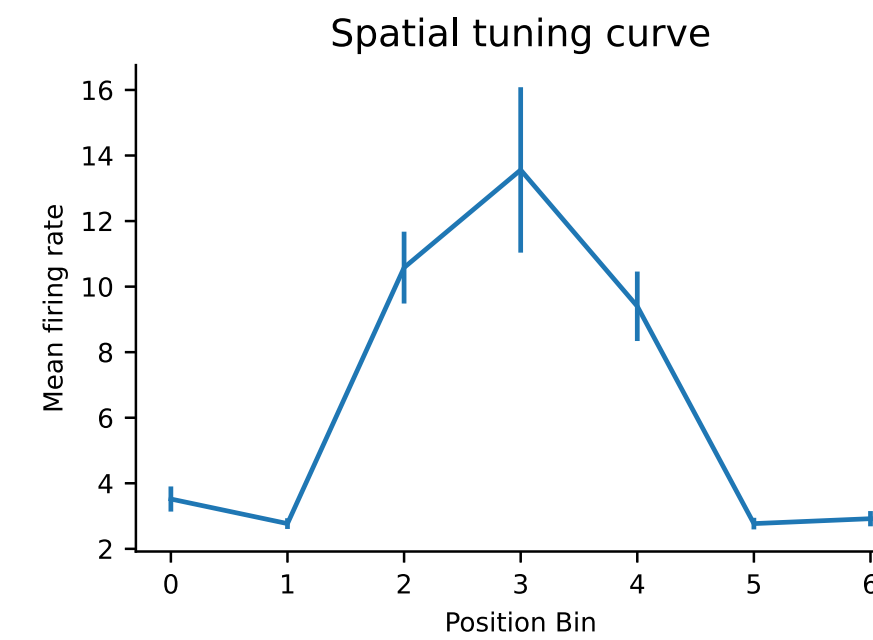
Why GLMs?

1. Why not linear regression?
which assumes normality

- A. Spike counts are non-Gaussian
- B. Neural activity variance is non-constant

2. GLM are as **easy to fit** as linear regression
convex, unique optimal solution

3. GLM are **flexible**
model multiple inputs jointly

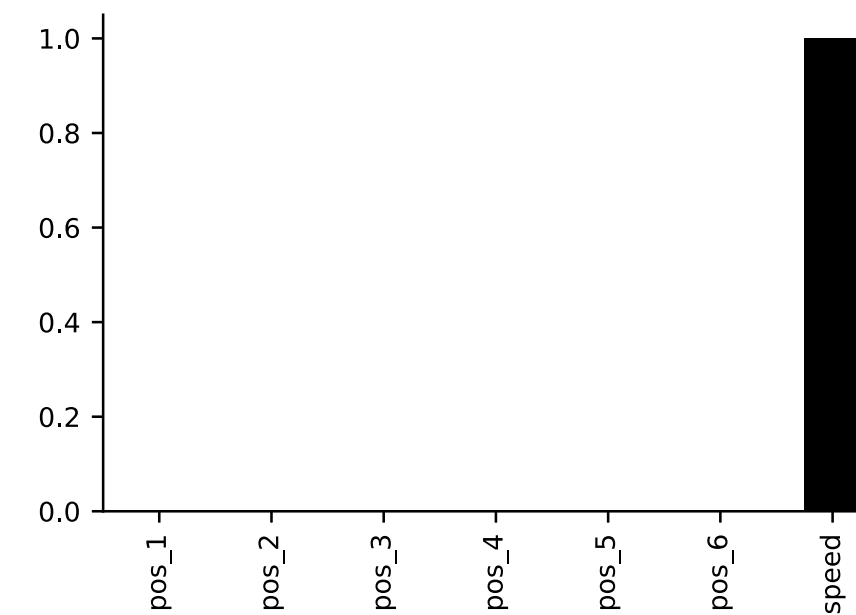


Firing rate model:

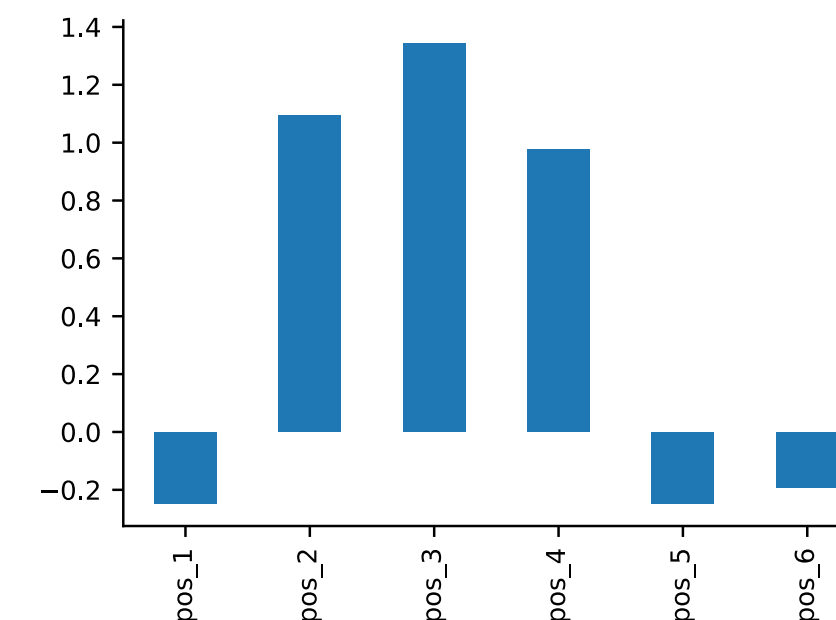
$$\text{firing rate} = \exp(w_0 \cdot \text{pos}_0(t) + \dots + w_6 \cdot \text{pos}_6(t))$$

$$\text{pos}_i(t) = \begin{cases} 1 & \text{if mouse is in position } i \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

true weights



GLM position



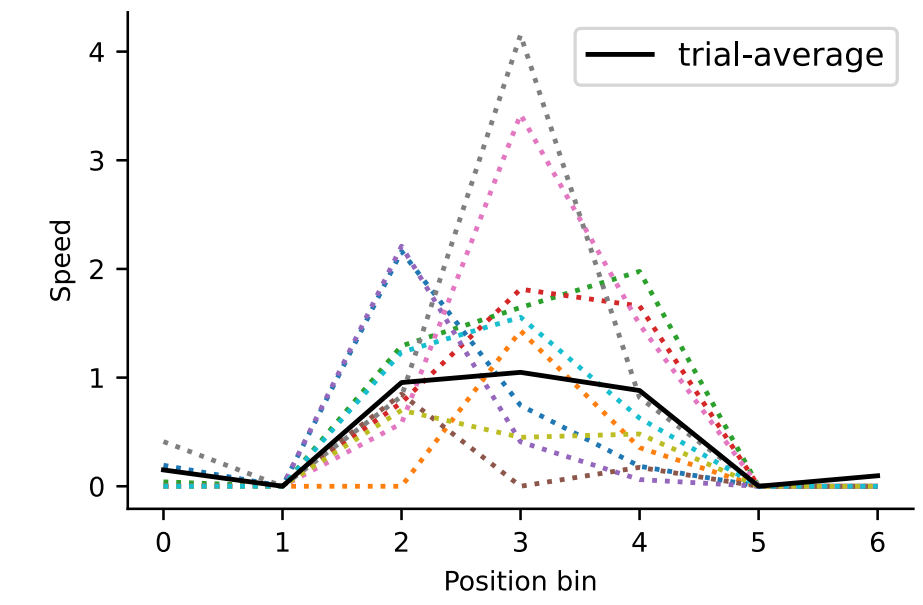
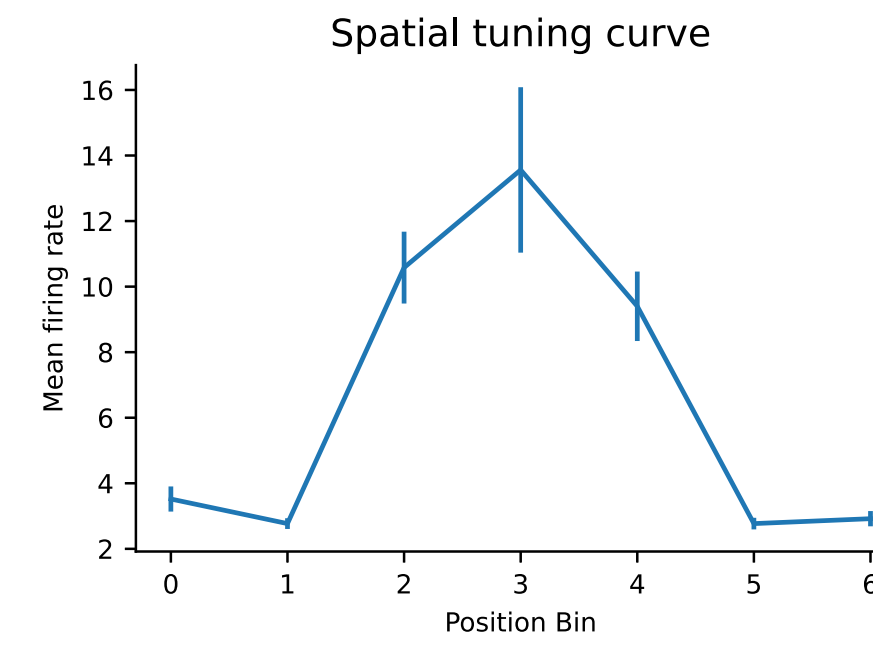
Why GLMs?

1. Why not linear regression?
which assumes normality

- A. Spike counts are non-Gaussian
- B. Neural activity variance is non-constant

2. GLM are as **easy to fit** as linear regression
convex, unique optimal solution

3. GLM are **flexible**
model multiple inputs jointly

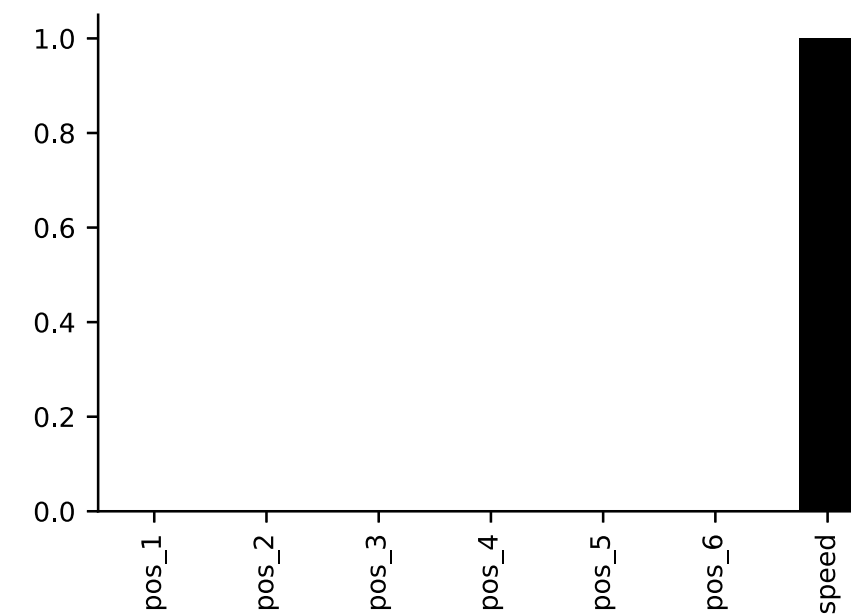


Firing rate model:

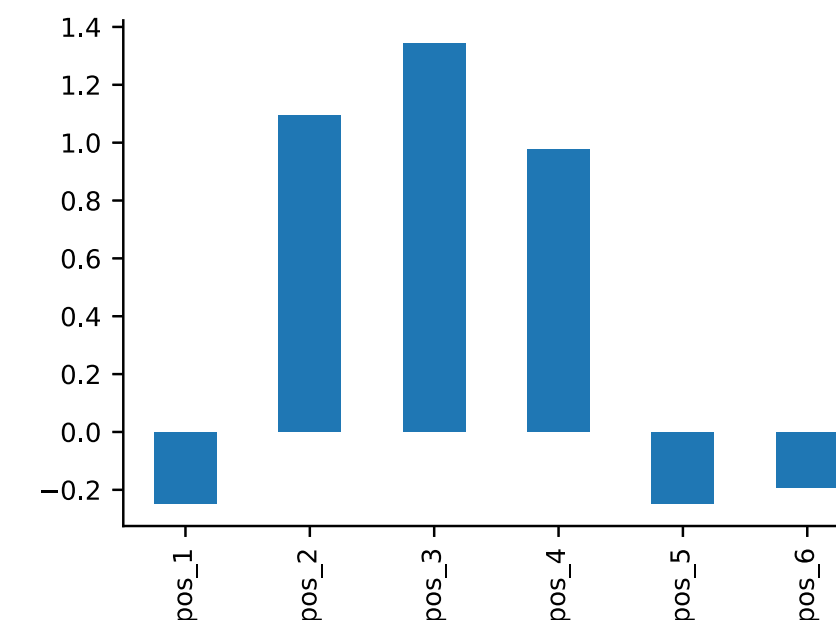
$$\text{firing rate} = \exp(w_0 \cdot \text{pos}_0(t) + \dots + w_6 \cdot \text{pos}_6(t) + w_s \cdot \text{speed}(t))$$

$$\text{pos}_i(t) = \begin{cases} 1 & \text{if mouse is in position } i \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

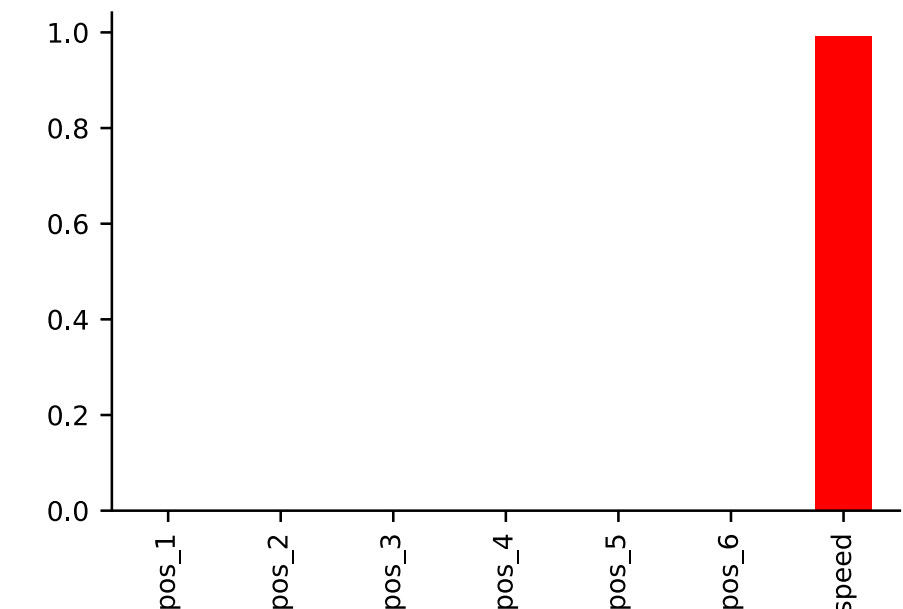
true weights



GLM position



GLM position + speed



What can I do with a GLM?

1. Model responses to high dimensional inputs
images, videos, 2D/3D positions...

Pillow et al., 2008
Retina Macaques

Hardcastle et al., 2018
MEC mice

Gardner et al. 2019
MEC rats

Park et al. 2019
LIP Macaques

Weber & Pillow 2017
simulations

Peyrache et al., 2018
ADN mice

What can I do with a GLM?

1. Model responses to high dimensional inputs

images, videos, 2D/3D positions...

2. Non-linear responses

place cells, head-direction, grid cells

Pillow et al., 2008

Retina Macaques

Hardcastle et al., 2018

MEC mice

Gardner et al. 2019

MEC rats

Park et al. 2019

LIP Macaques

Weber & Pillow 2017

simulations

Peyrache et al., 2018

ADN mice

What can I do with a GLM?

1. Model responses to high dimensional inputs

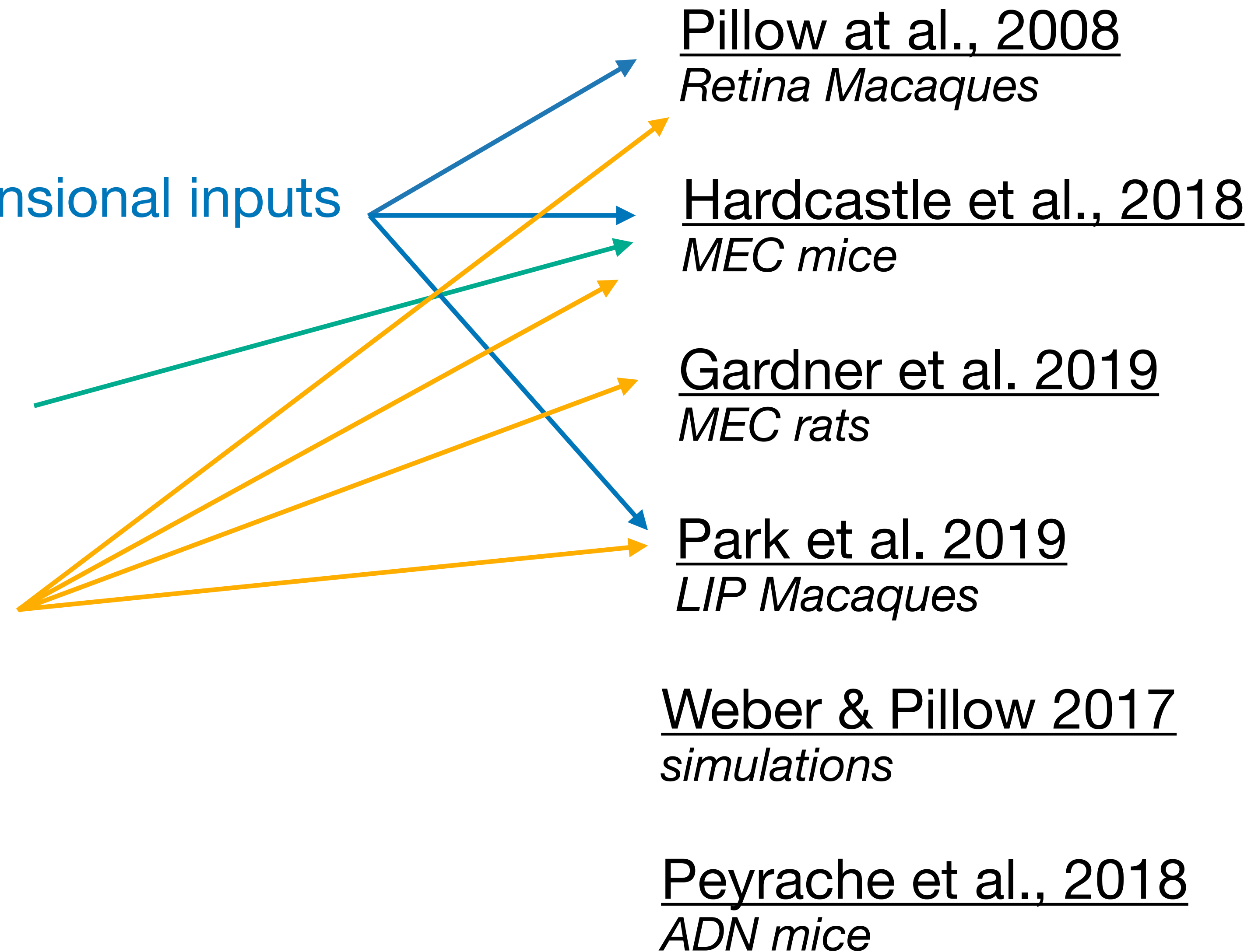
images, videos, 2D/3D positions...

2. Non-linear responses

place cells, head-direction, grid cells

3. Functional connectivity

and other time-dependent effects



What can I do with a GLM?

1. Model responses to high dimensional inputs

images, videos, 2D/3D positions...

2. Non-linear responses

place cells, head-direction, grid cells

3. Functional connectivity

and other time-dependent effects

4. Generate surrogate dataset

Pillow et al., 2008

Retina Macaques

Hardcastle et al., 2018

MEC mice

Gardner et al. 2019

MEC rats

Park et al. 2019

LIP Macaques

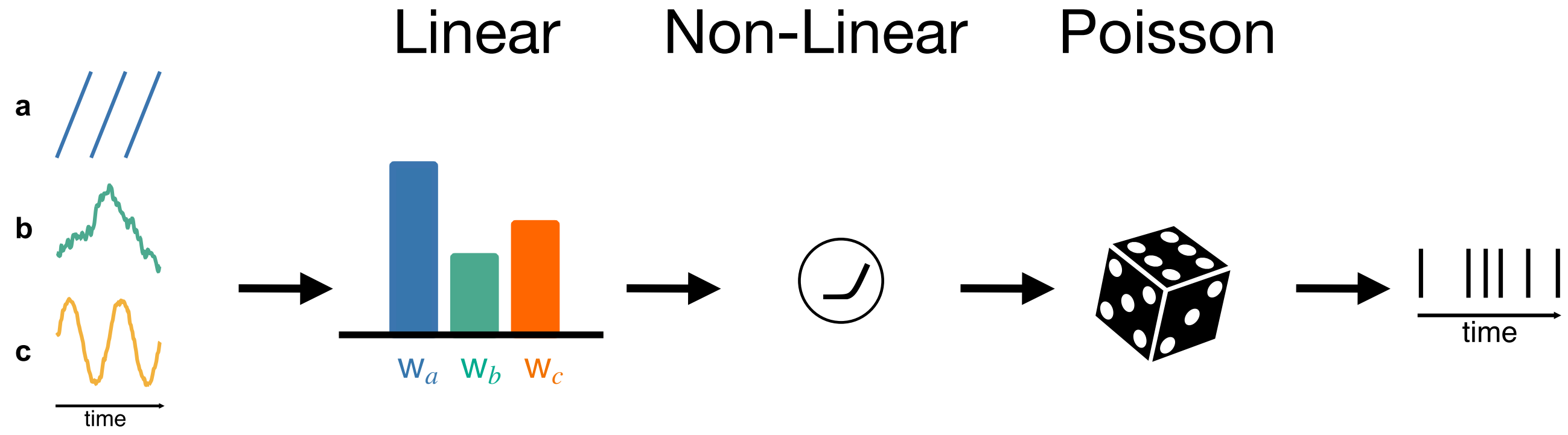
Weber & Pillow 2017

simulations

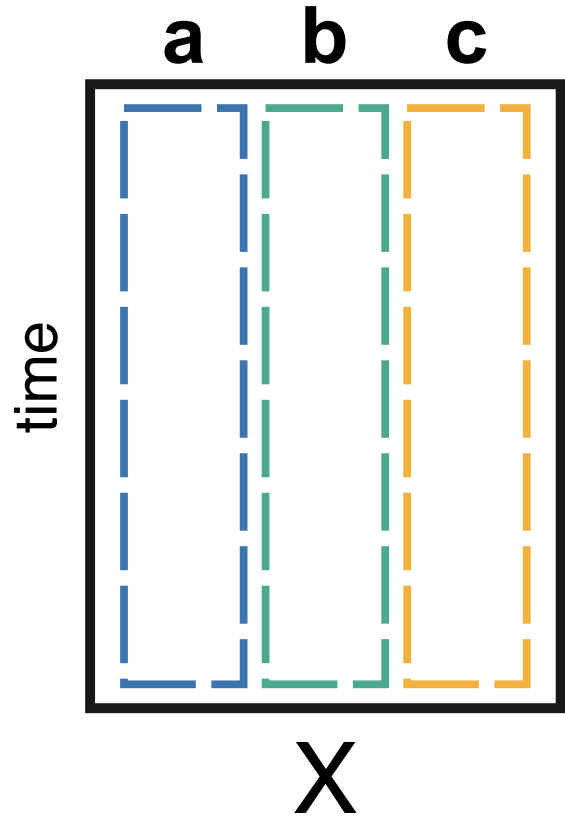
Peyrache et al., 2018

ADN mice

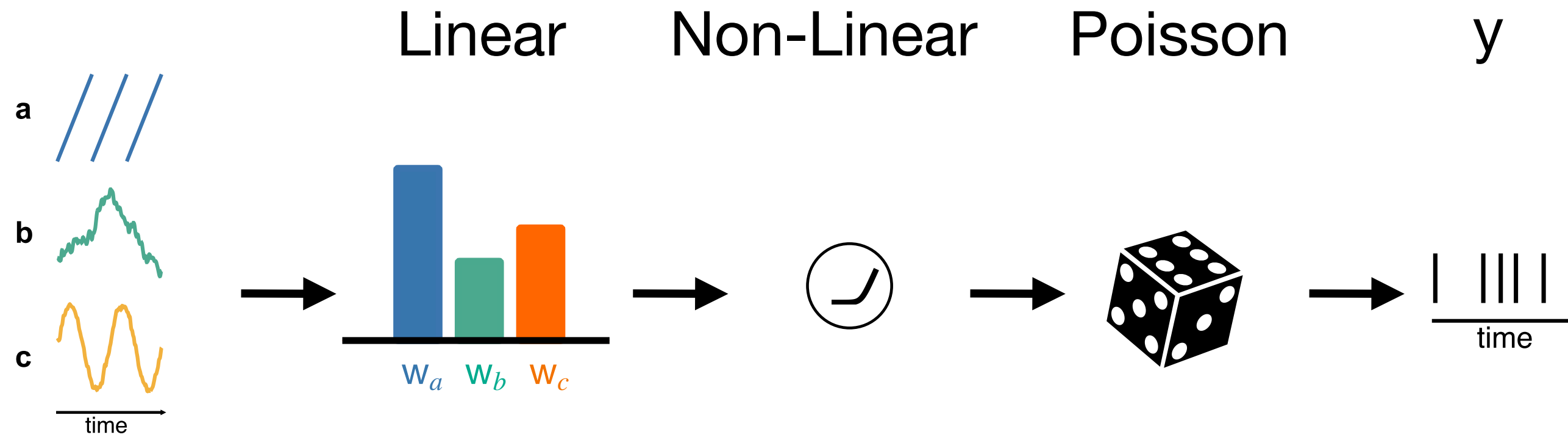
GLM in NeMoS



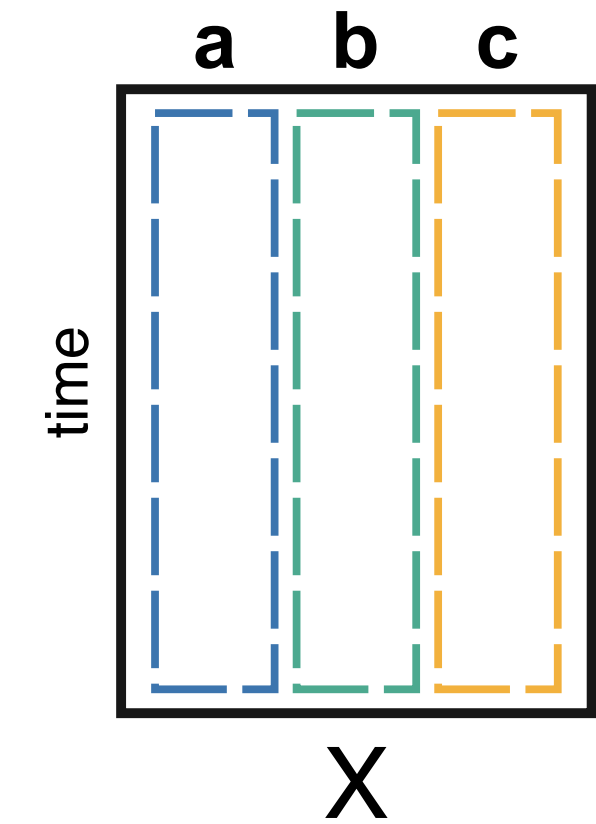
Feature matrix



GLM in NeMoS



Feature matrix

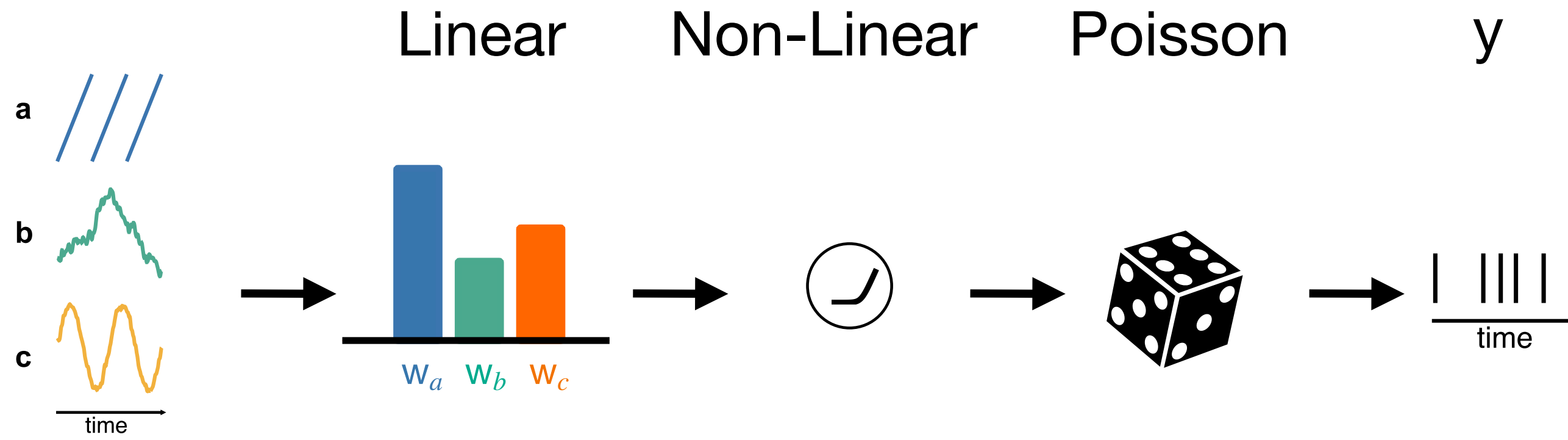


```
import nemos as nmo
```

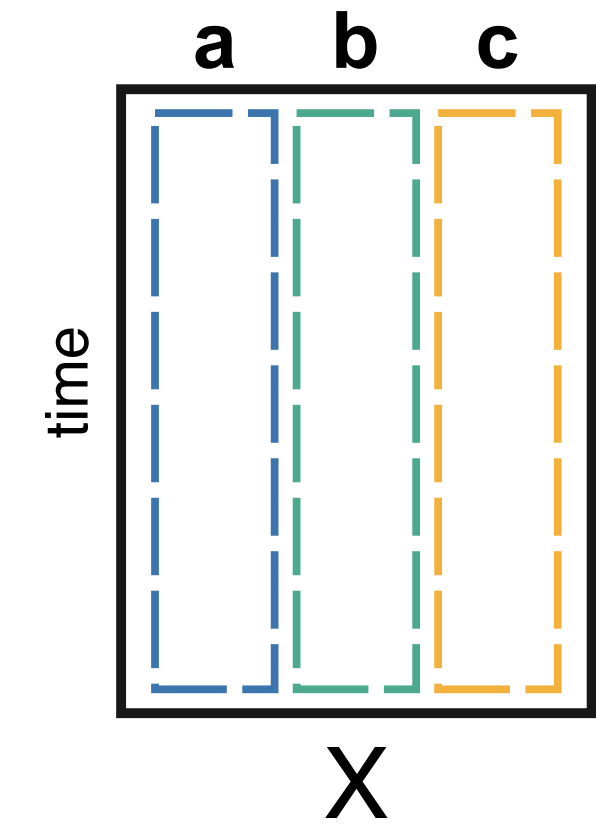
```
model = nmo.glm.GLM()
```

Define the model

GLM in NeMoS



Feature matrix



```
import nemos as nmo
```

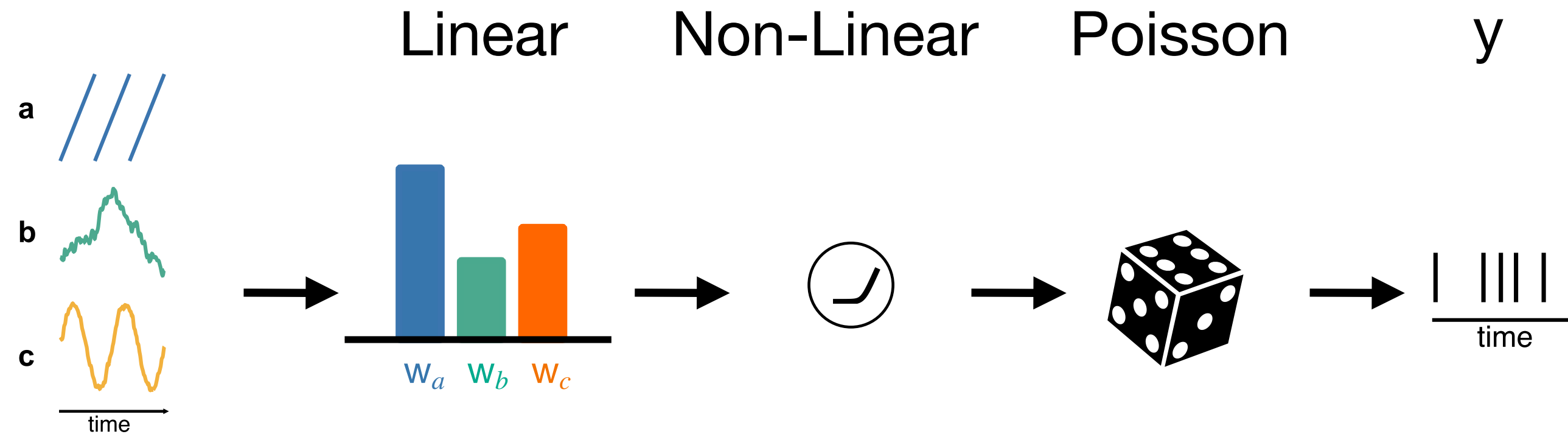
```
model = nmo.glm.GLM()
```

```
model.fit(X, y)
```

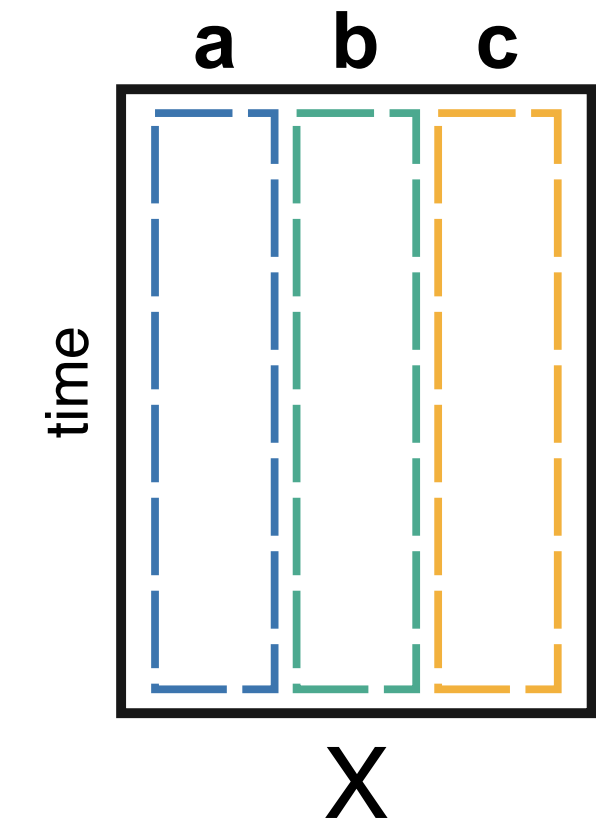
Define the model

Fit the GLM (learn w_a , w_b , w_c)

GLM in NeMoS



Feature matrix



```
import nemos as nmo
```

```
model = nmo.glm.GLM()
```

```
model.fit(X, y)
```

```
firing_rate = model.predict(X)
```

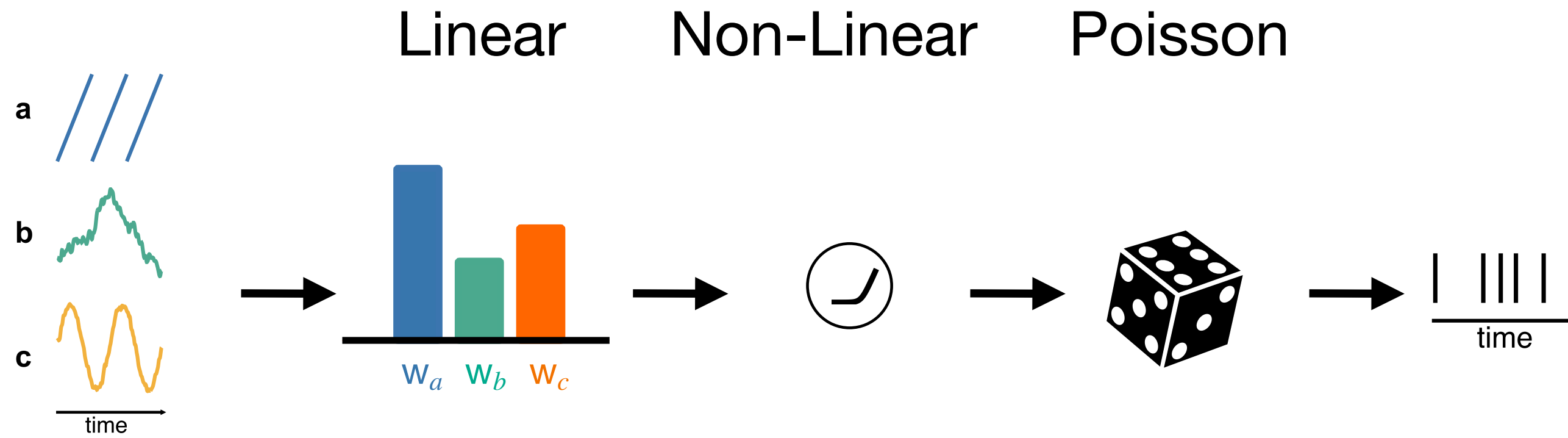
Define the model

Fit the GLM (learn w_a , w_b , w_c)

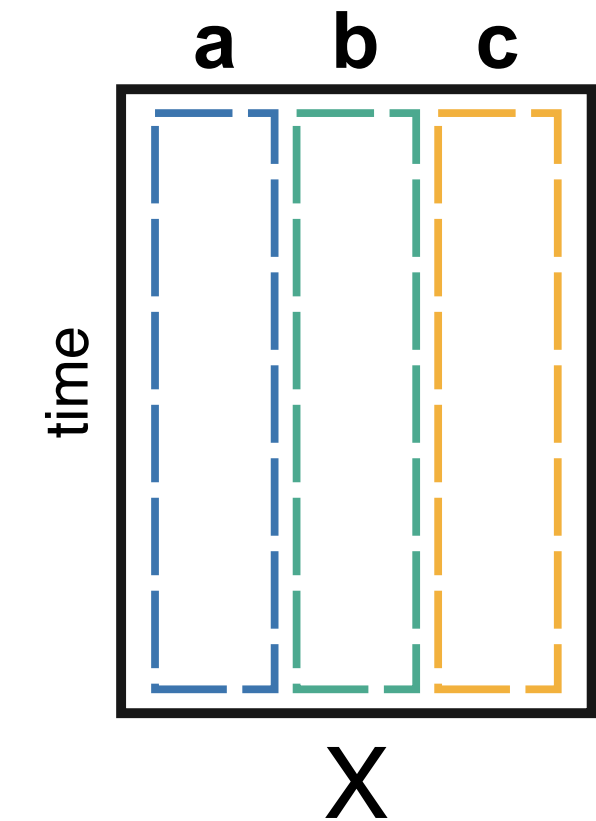
Predict the firing rate

$\exp(a \cdot w_a + b \cdot w_b + c \cdot w_c)$

GLM in NeMoS



Feature matrix



```
import nemos as nmo
```

```
model = nmo.glm.GLM()
```

```
model.fit(X, y)
```

```
firing_rate = model.predict(X)
```

```
log_likelihood = model.score(X, y)
```

Define the model

Fit the GLM (learn w_a , w_b , w_c)

Predict the firing rate
 $\exp(a \cdot w_a + b \cdot w_b + c \cdot w_c)$

Compute the log-likelihood

Summary GLMs

- **GLMs**: weighted sum of features \rightarrow exponential \rightarrow Poisson spike generation

What features can/should I use?

- It's up to the scientist!
- Choosing features is a way to formulate hypothesis about the neural encoding.
- Any fixed (not learned) transformation of your data is valid* (counting, binning, projecting into Principal Components, filtering, squaring ...)

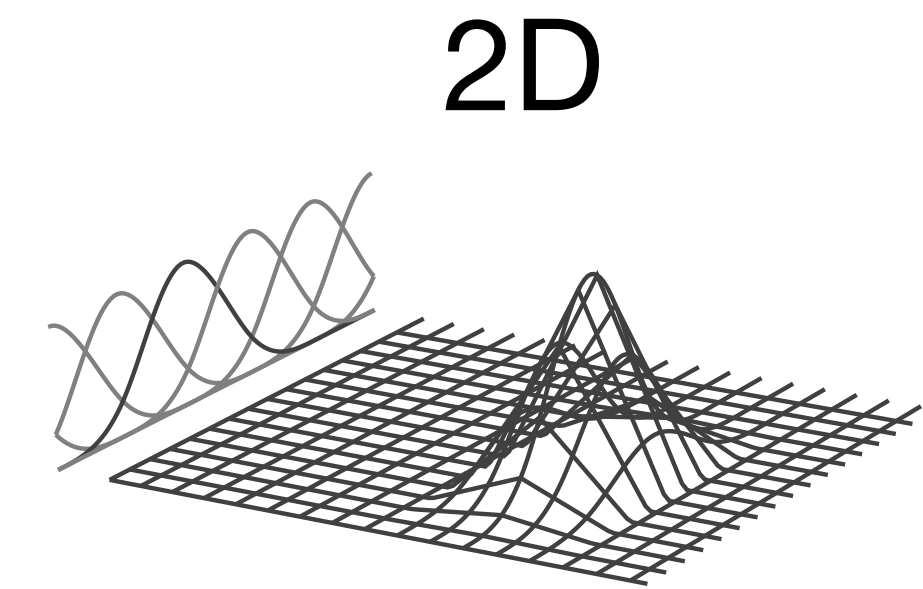
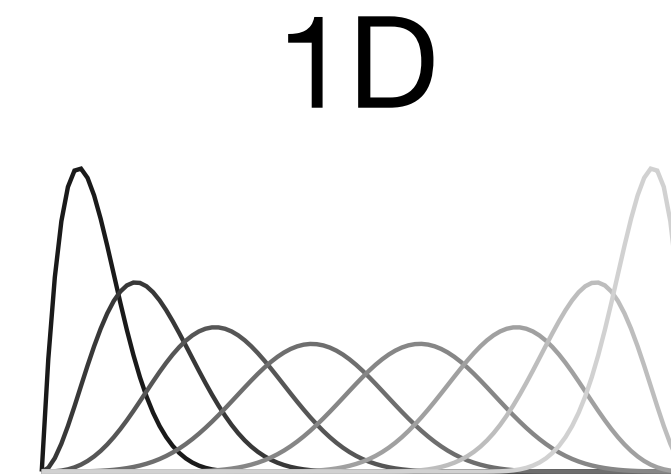
**as long as the resulting time axis matches that of the spike counts*

Constructing Features in NeMoS

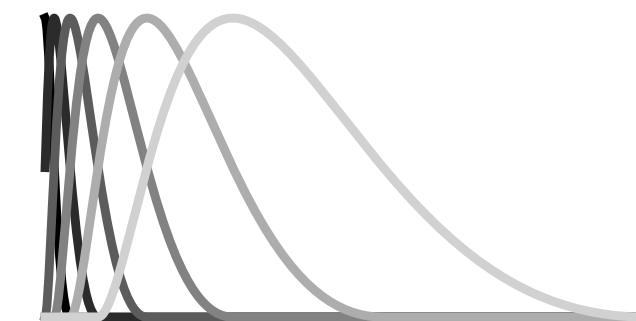
- NeMoS provides the **basis** module for feature construction

Constructing Features in NeMoS

- NeMoS provides the **basis** module for feature construction
- Basis are **fixed non-linearities**

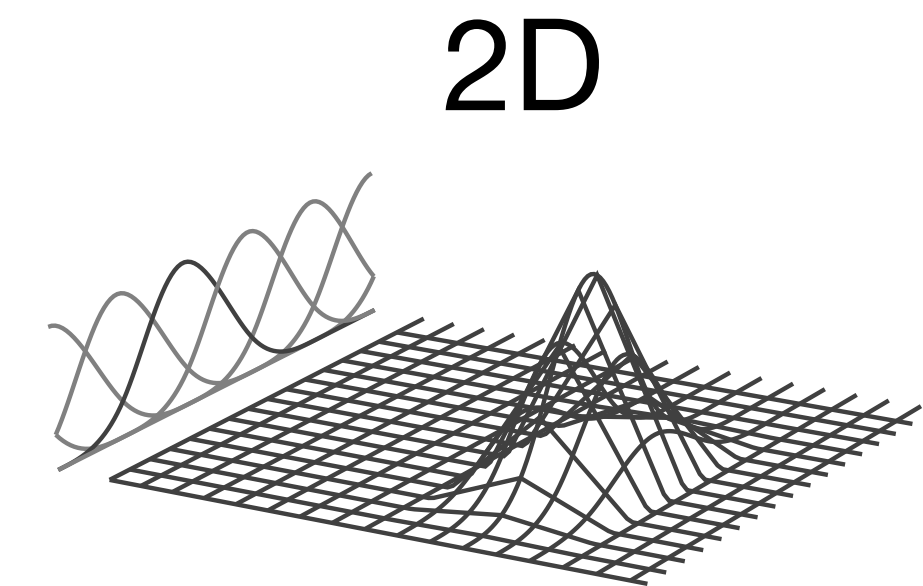
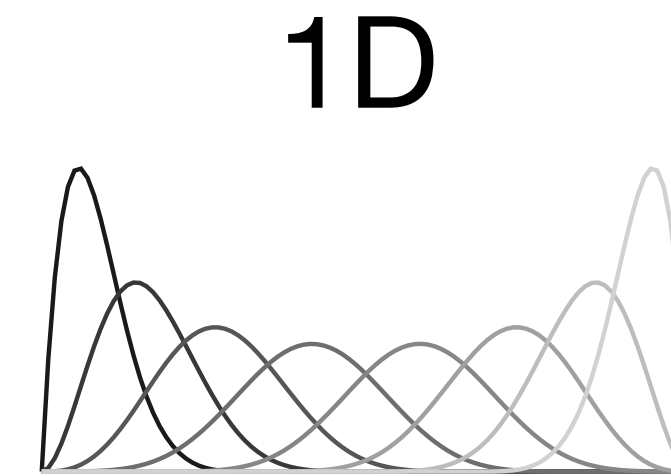


log-stretched

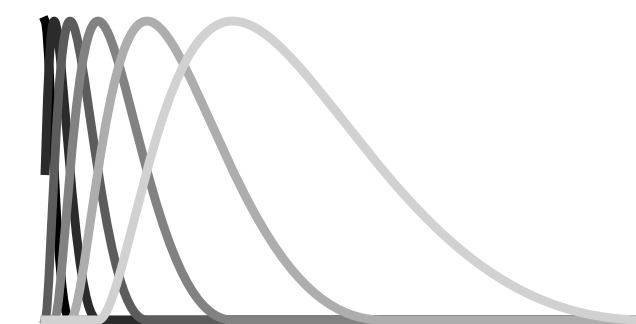


Constructing Features in NeMoS

- NeMoS provides the **basis** module for feature construction
- Basis are **fixed non-linearities**
- Assume that **firing rate varies smoothly/gradually**

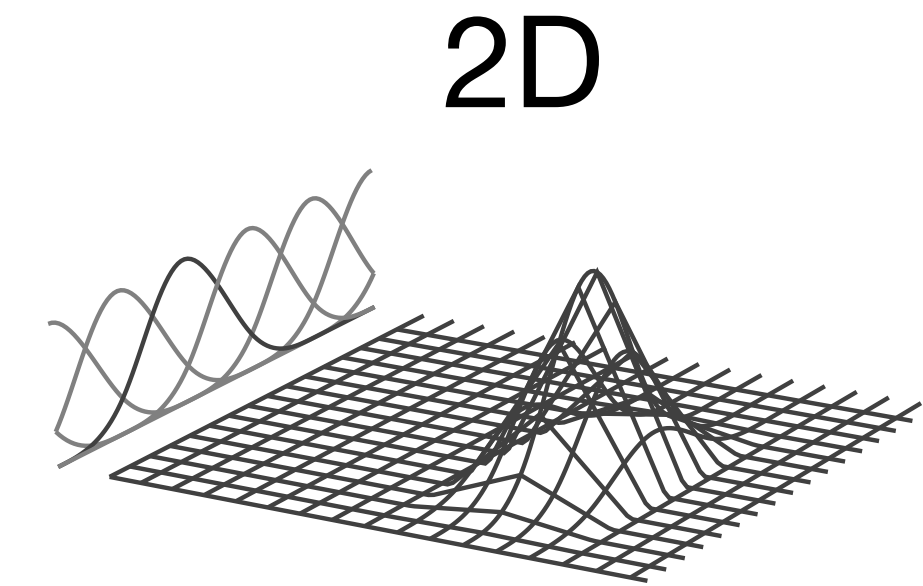
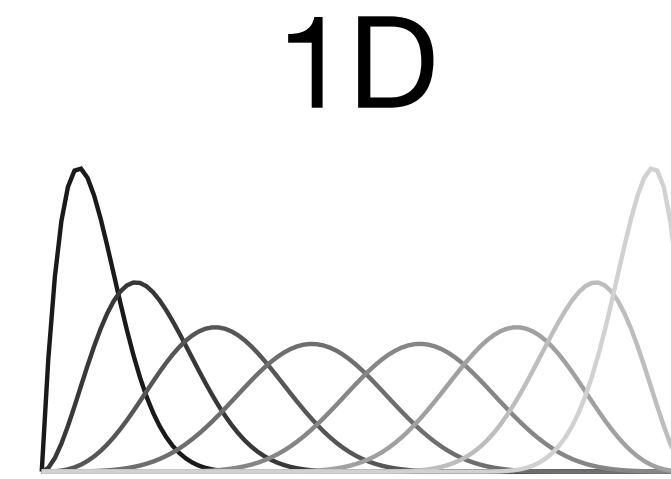


log-stretched

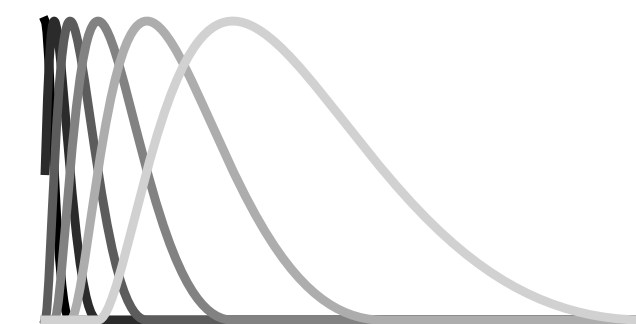


Constructing Features in NeMoS

- NeMoS provides the **basis** module for feature construction
- Basis are **fixed non-linearities**
- Assume that **firing rate varies smoothly/gradually**
- Used for:
 1. Reducing dimensionality
 2. Non-linear firing rate modulation
 3. Time dependent effects

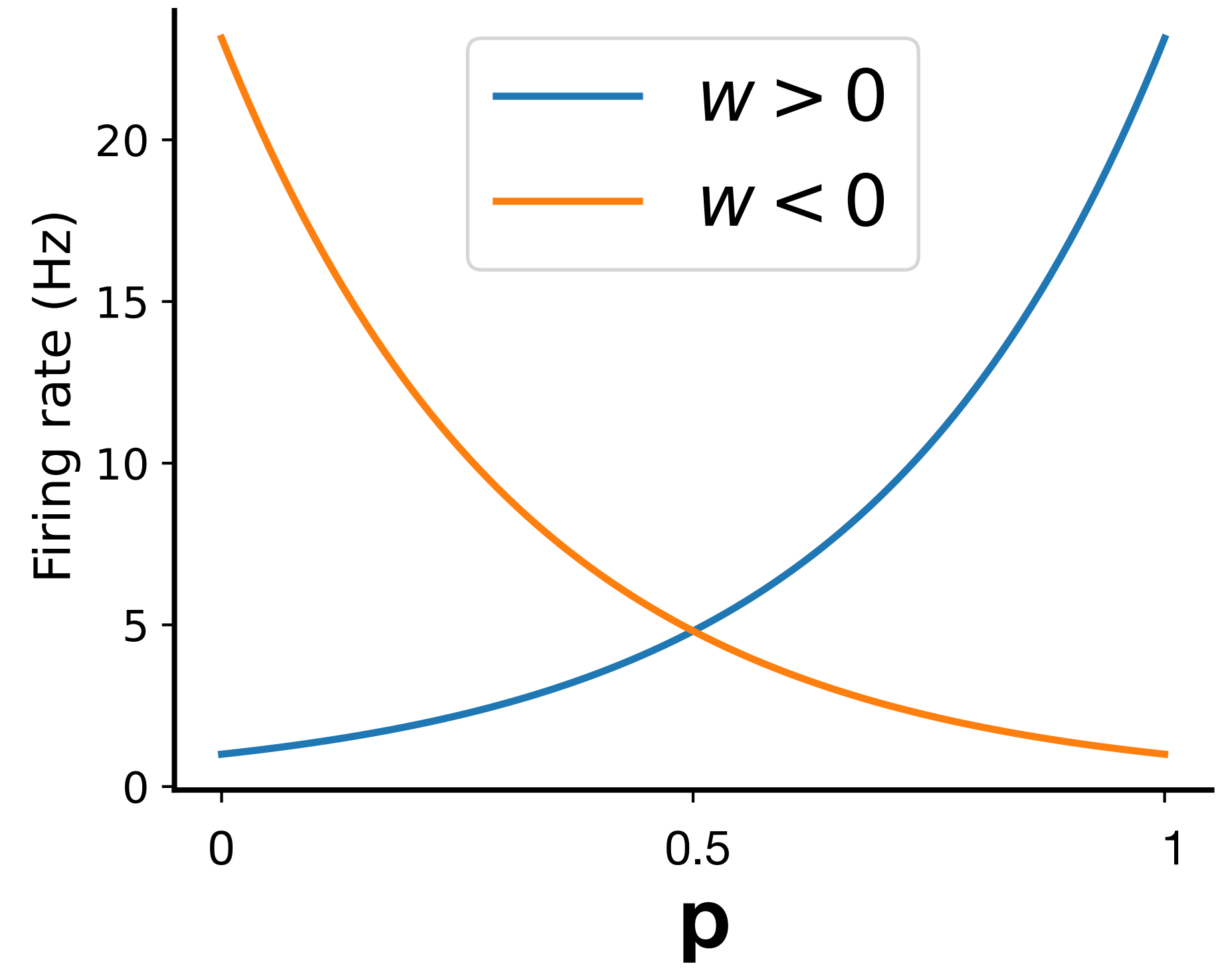
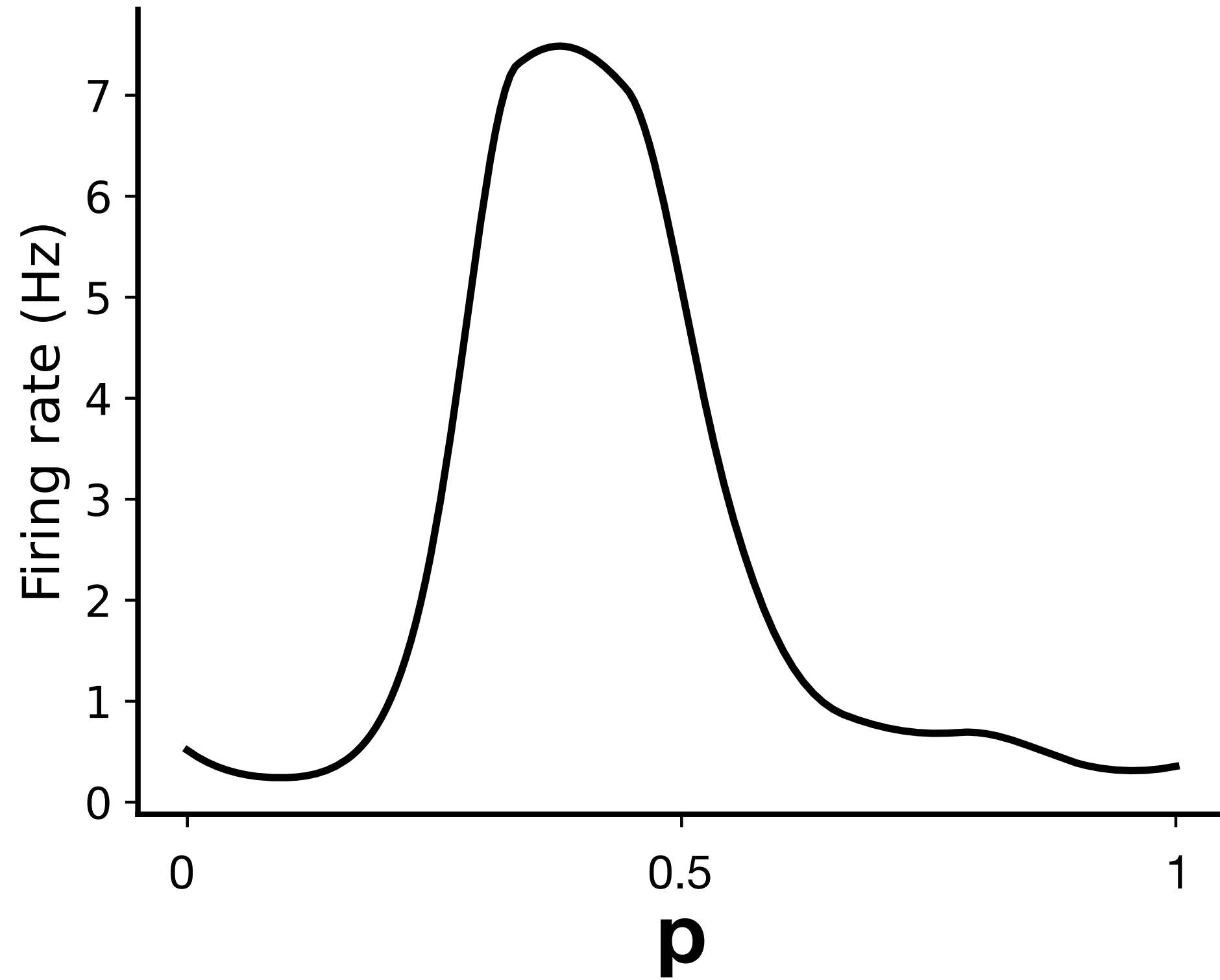


log-stretched



Example: 1D Non-Linear Rate Map

Position tuning



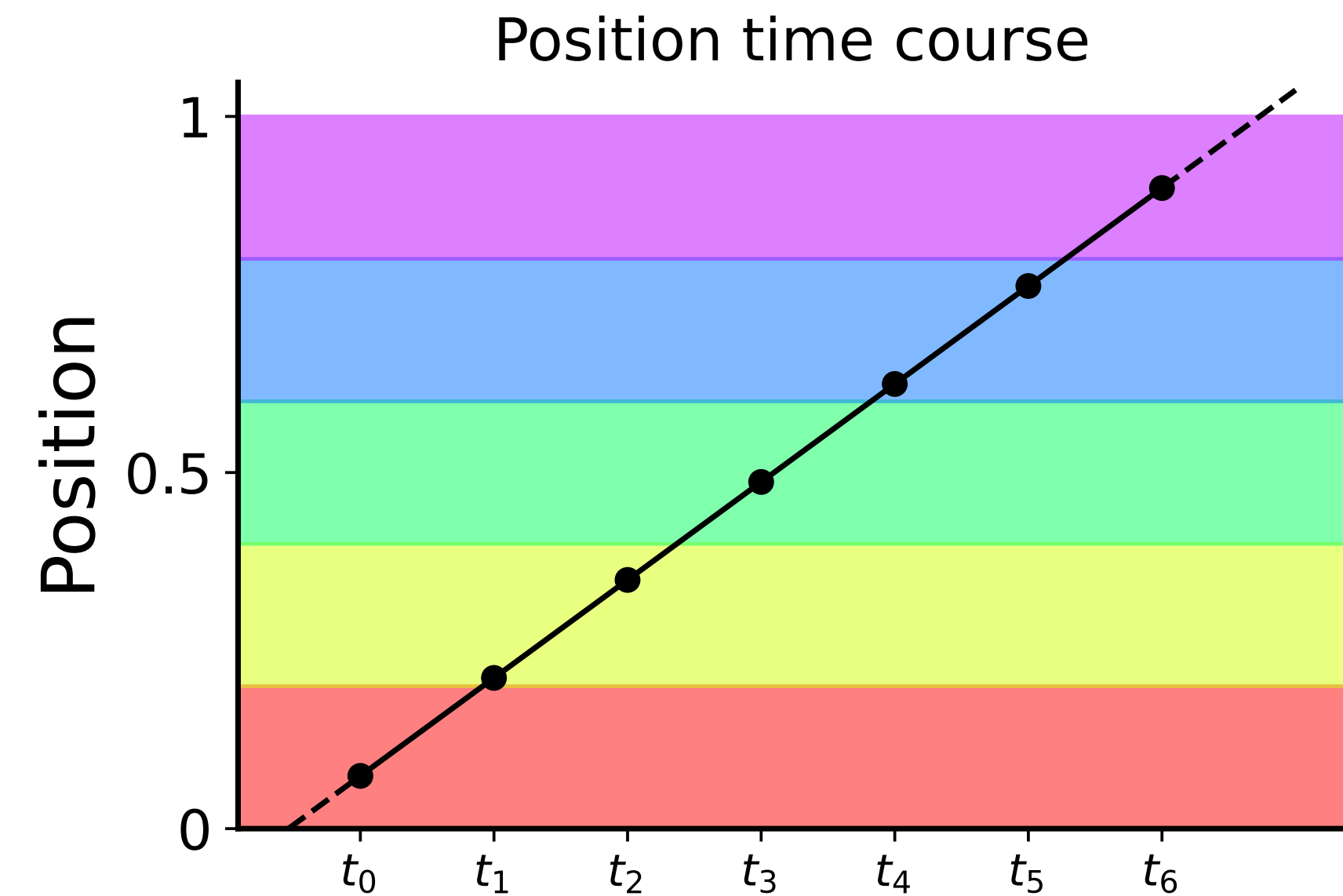
$$\text{firing rate}[t] = \exp(\mathbf{p}[t] \cdot w + c)$$

$\mathbf{p}[t]$ = position at time t

Example: 1D Non-Linear Rate Map

Simple approach:

- Bin the position



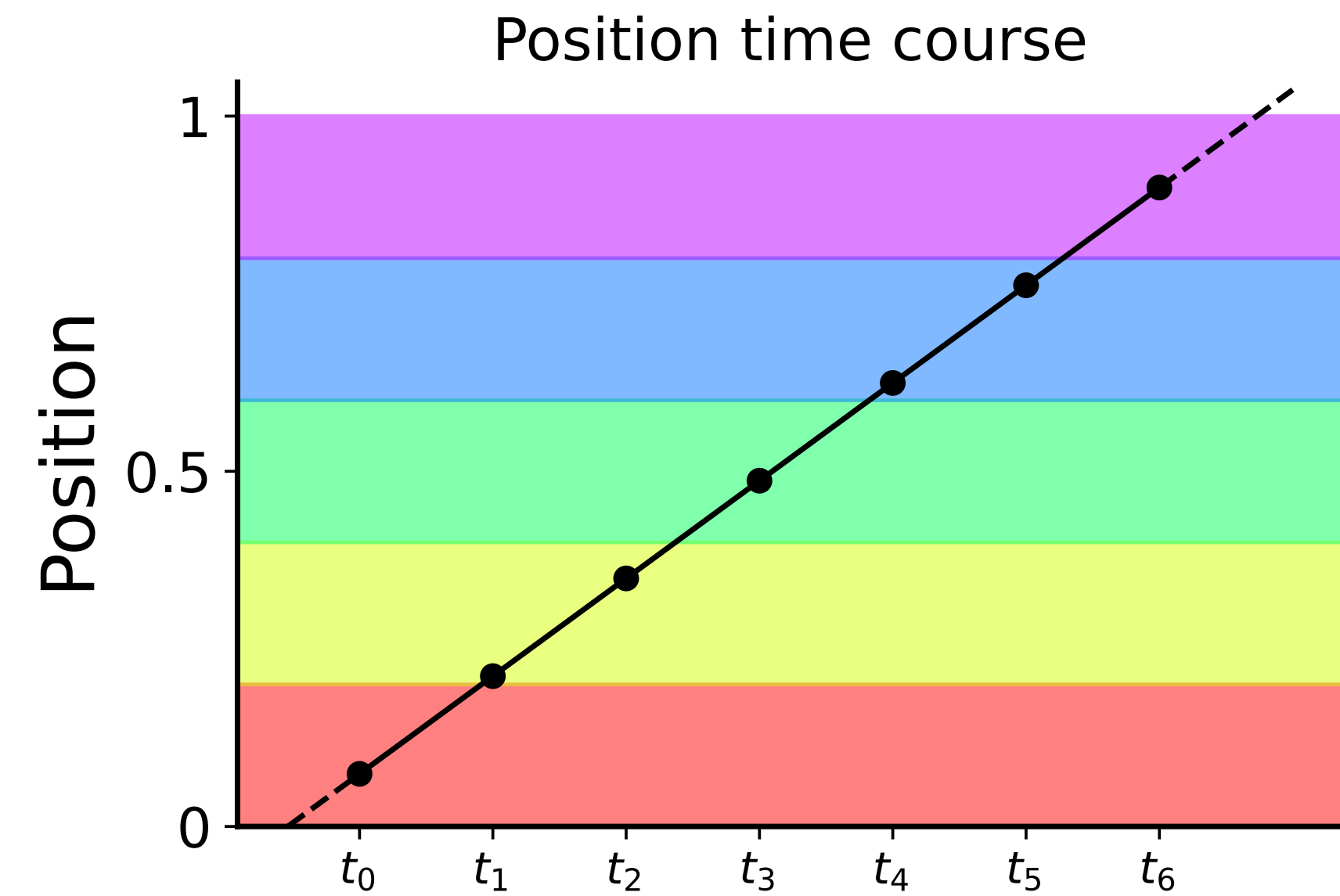
$$\text{firing rate}[t] = \exp(\delta_1[t] \cdot w_1 + \dots + \delta_5[t] \cdot w_5)$$

$$\delta_i(t) = \begin{cases} 1 & \text{if position is in bin } i \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

Example: 1D Non-Linear Rate Map

Simple approach:

- Bin the position
- Learn one weight per bin

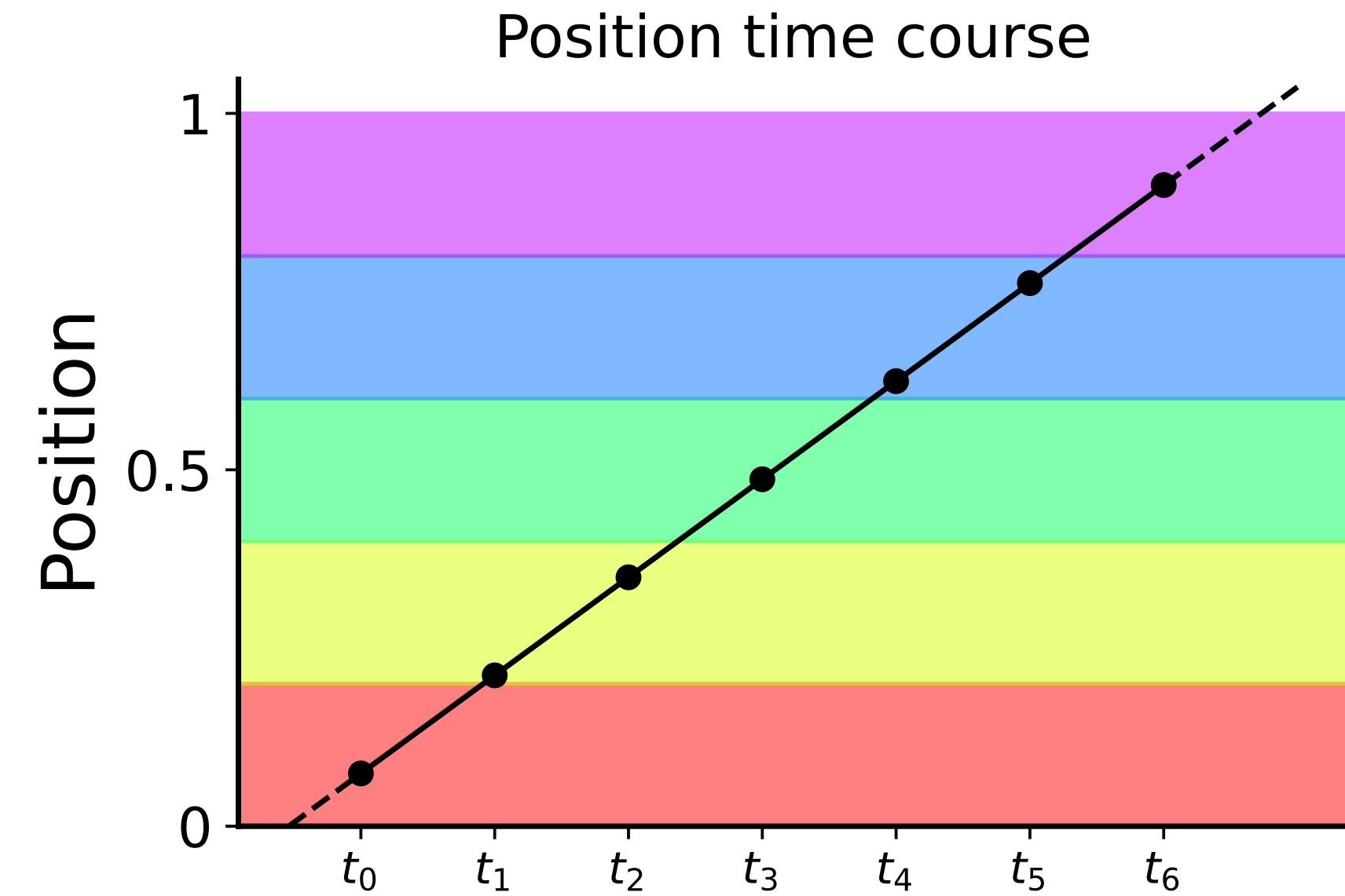


$$\text{firing rate}[t] = \exp\left(X \cdot \begin{bmatrix} w_1 \\ \vdots \\ w_5 \end{bmatrix}\right)$$

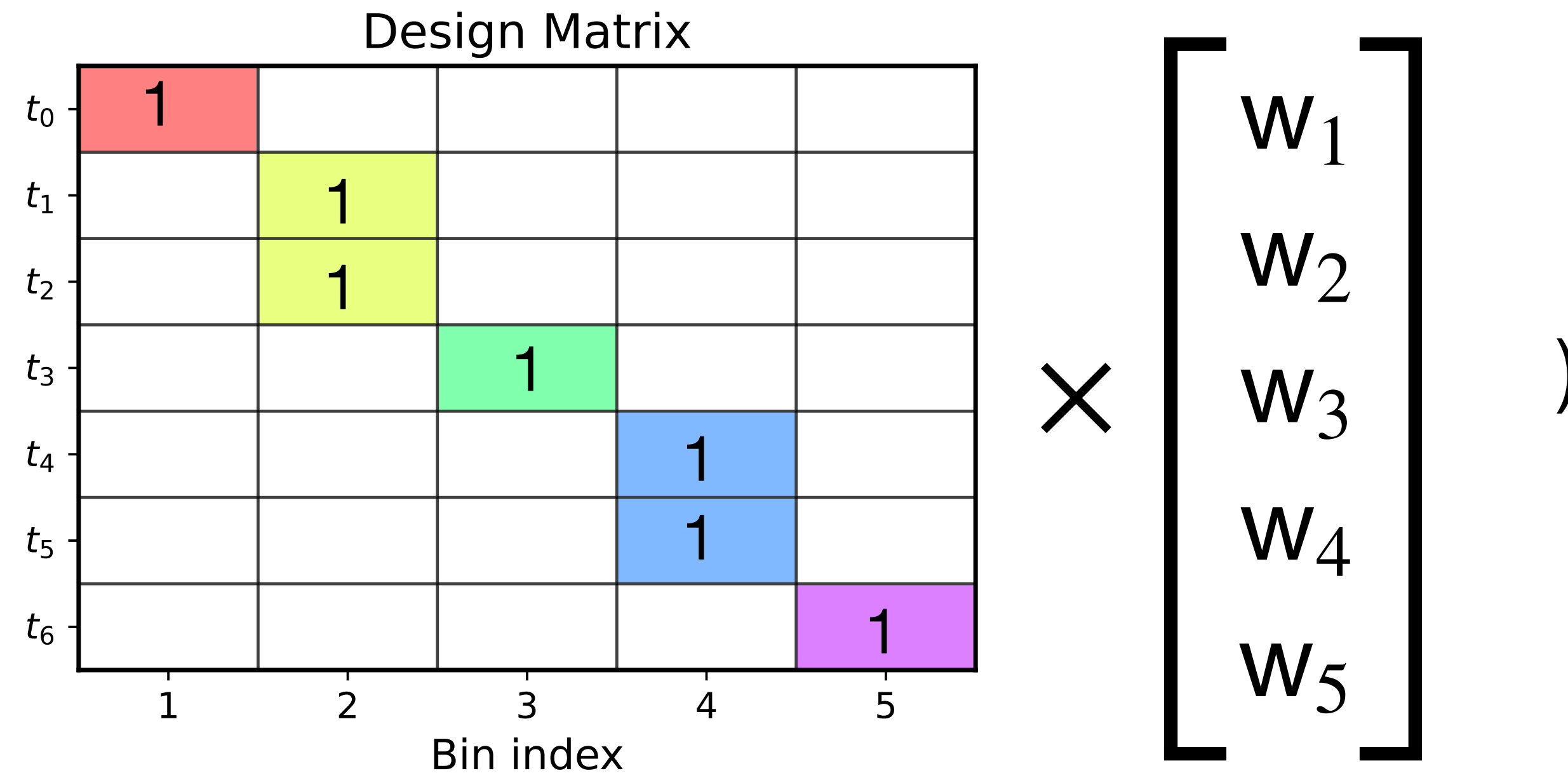
Example: 1D Non-Linear Rate Map

Simple approach:

- Bin the position
- Learn one weight per bin
- One-hot design matrix

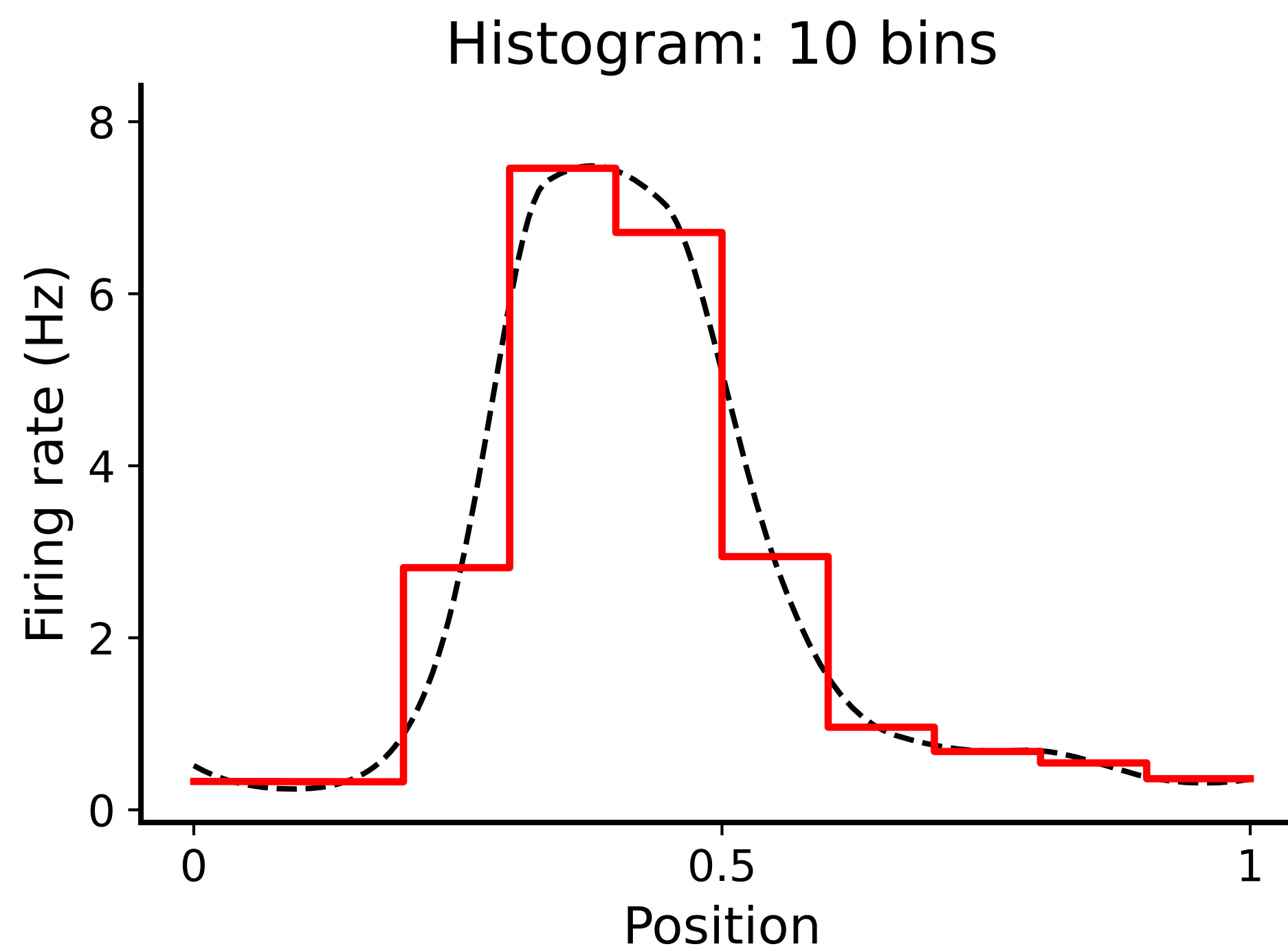


firing rate = exp(



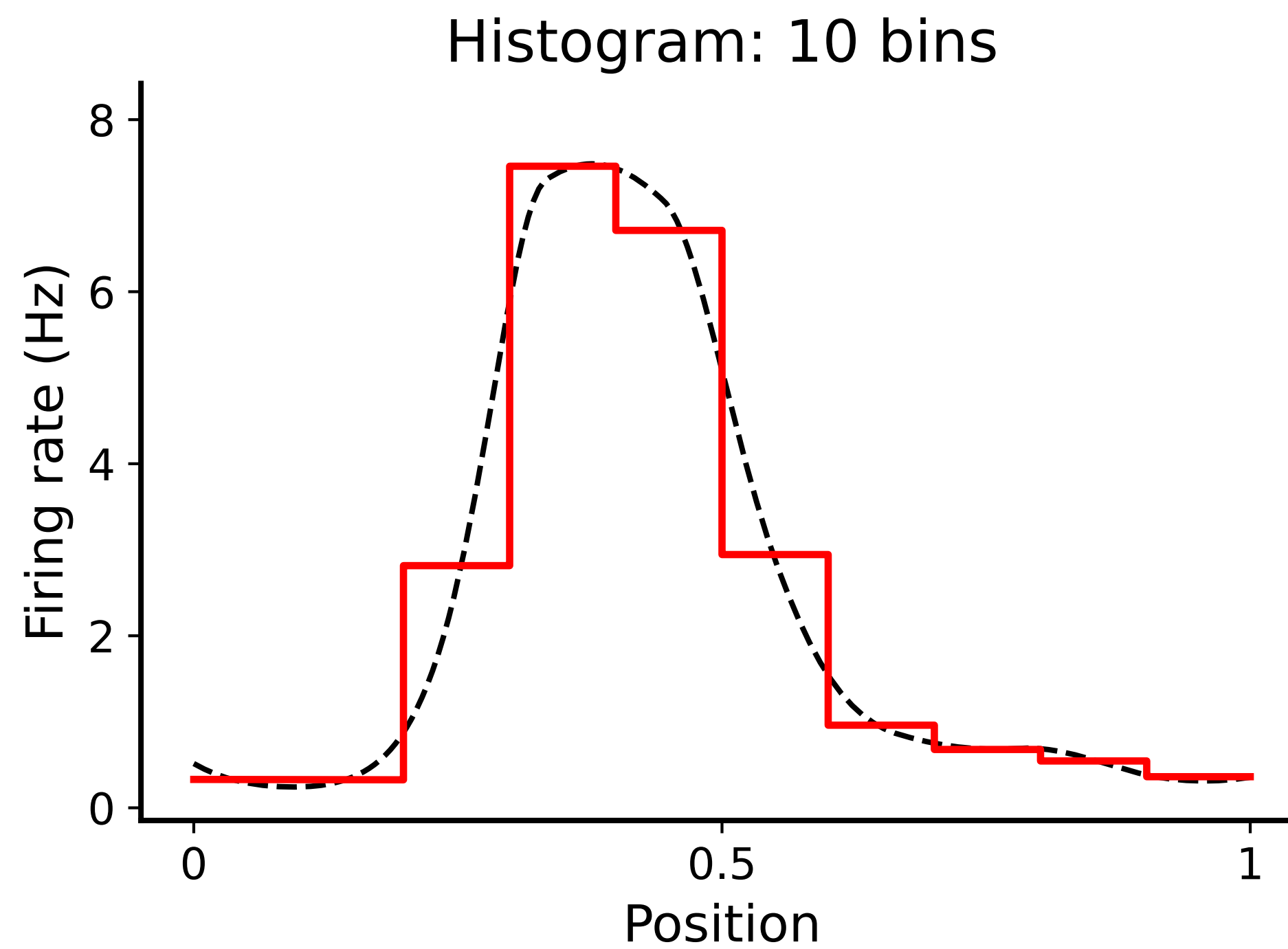
Limitations of binning

- Fit is **piecewise constant**



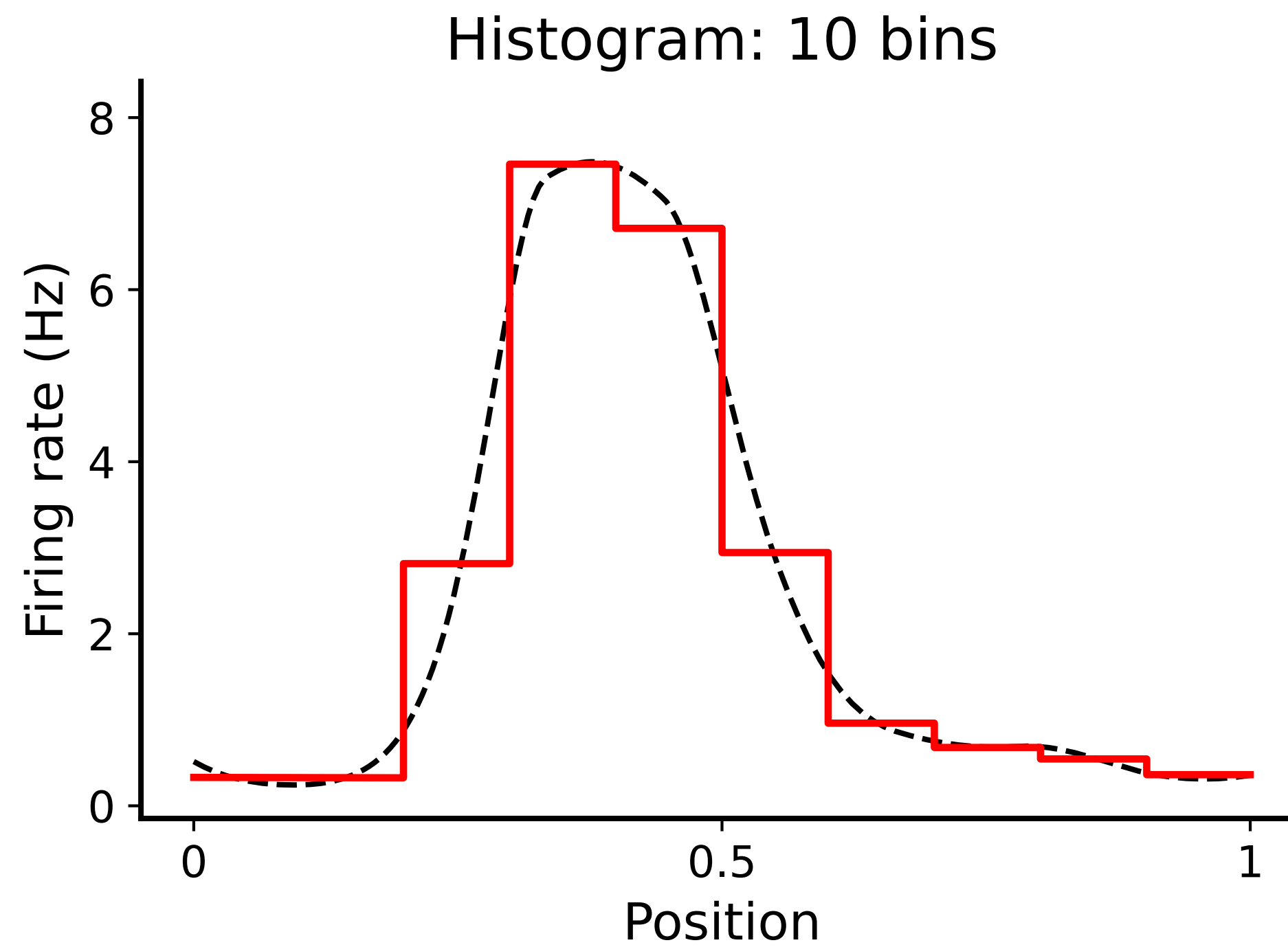
Limitations of binning

- Fit is **piecewise constant**
- More bins = finer resolution but more bins = more parameters



Limitations of binning

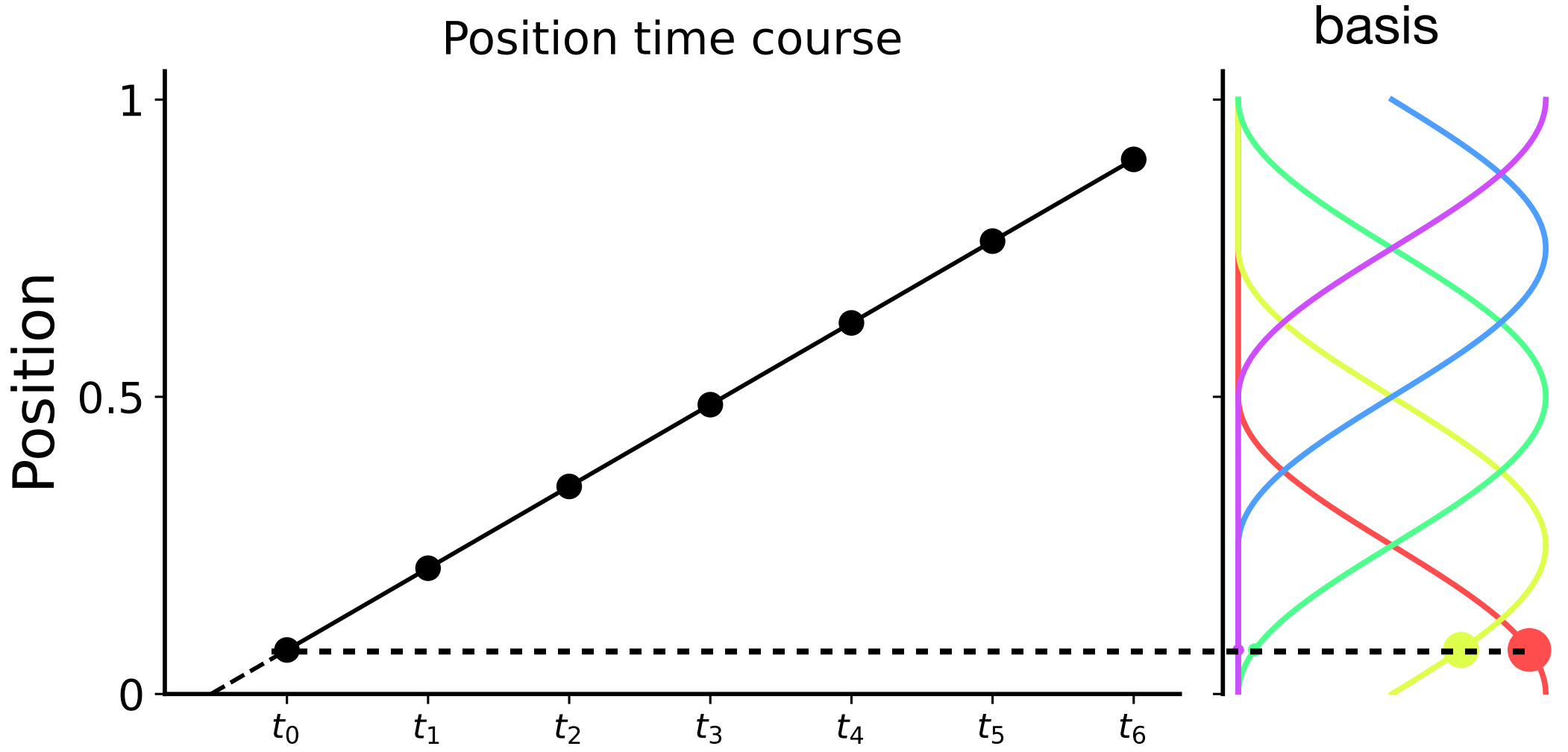
- Fit is **piecewise constant**
- More bins = finer resolution but more bins = more parameters
- Can we do better by exploiting the fact that tuning curves are smooth?



Example: 1D Non-Linear Rate Map



Example: 1D Non-Linear Rate Map



firing rate = exp(

Design Matrix

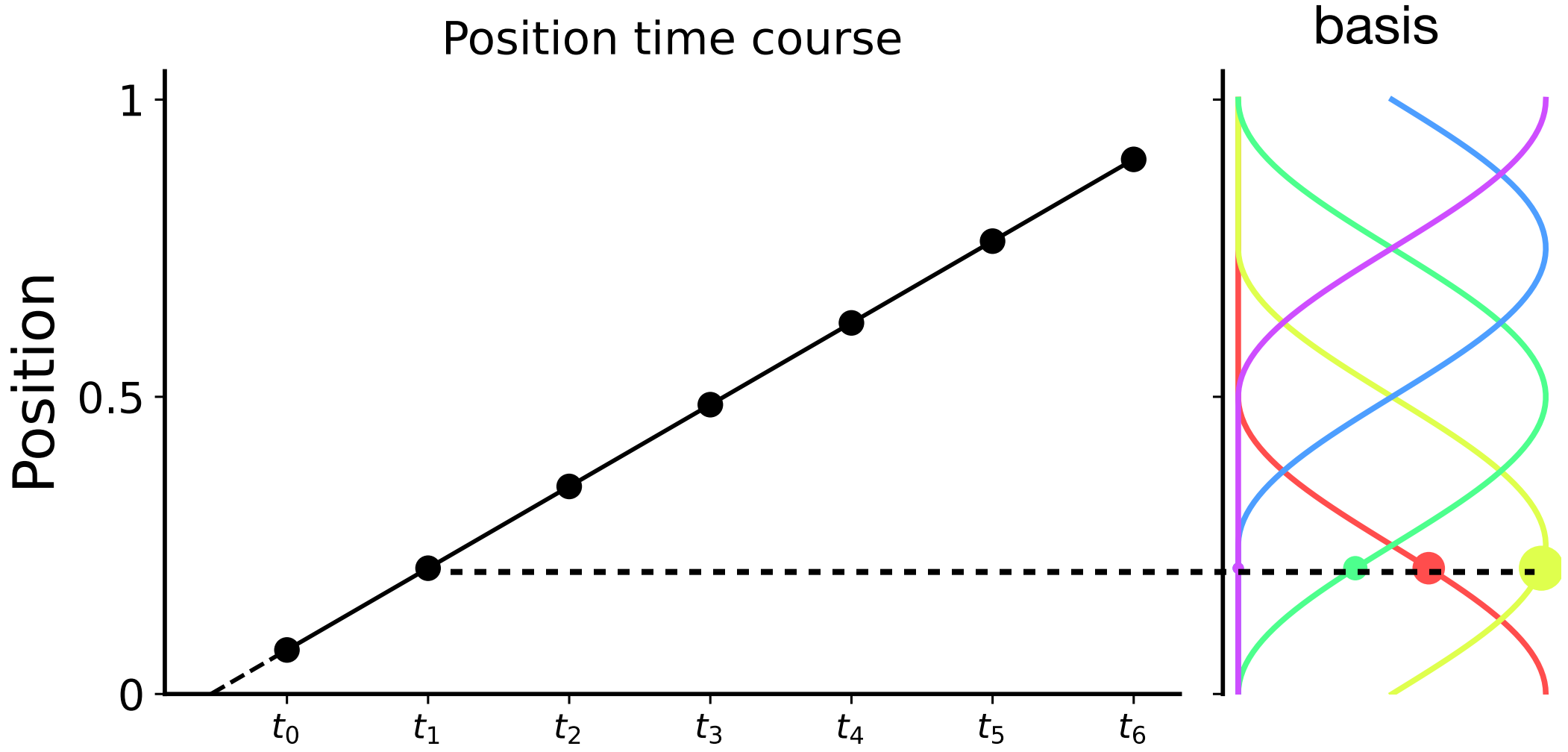
t_0					
t_1					
t_2					
t_3					
t_4					
t_5					
t_6					
	1	2	3	4	5

Bin index

\times

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix}$$
 $)$

Example: 1D Non-Linear Rate Map



firing rate = $\exp($

Design Matrix

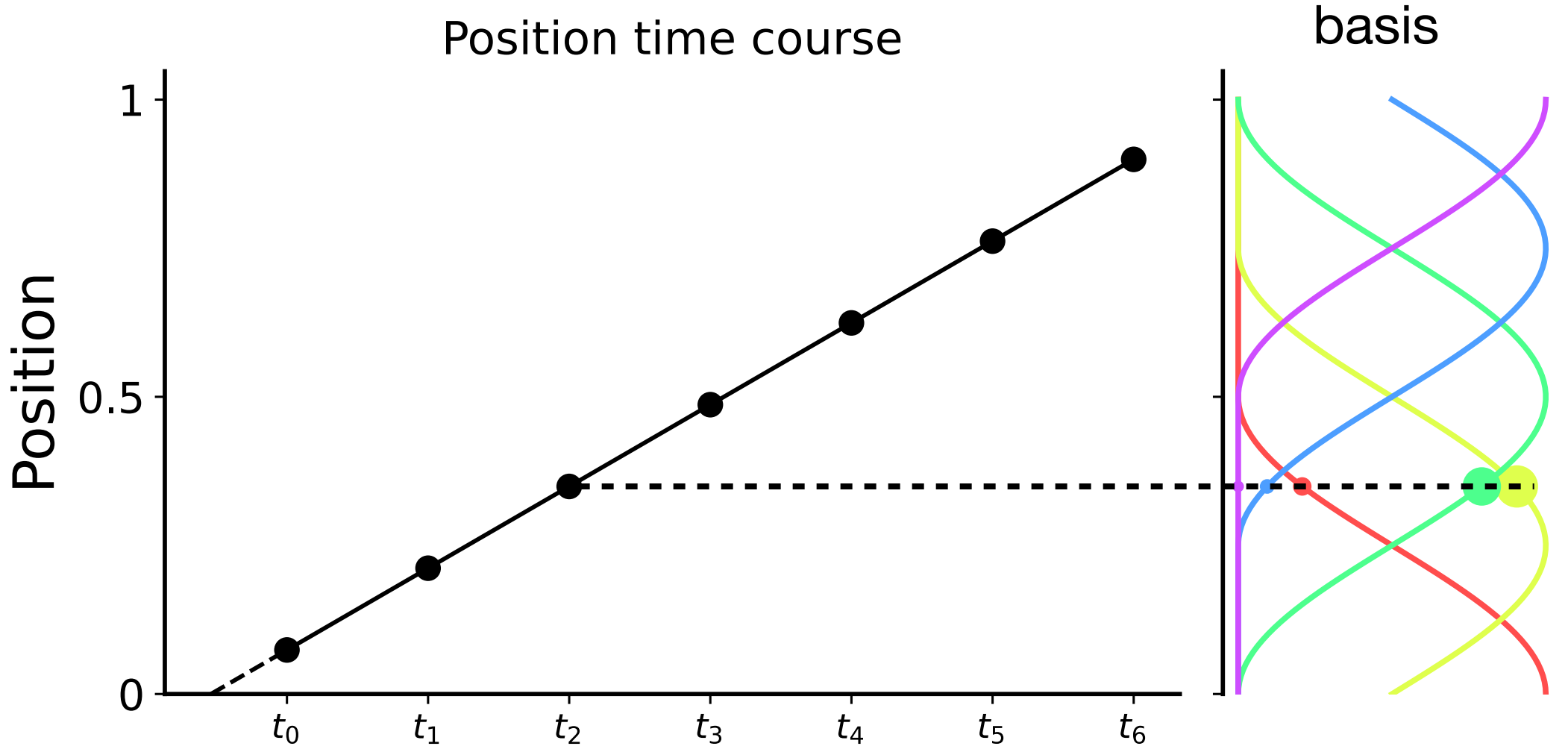
t_0					
t_1					
t_2					
t_3					
t_4					
t_5					
t_6					
	1	2	3	4	5

Bin index

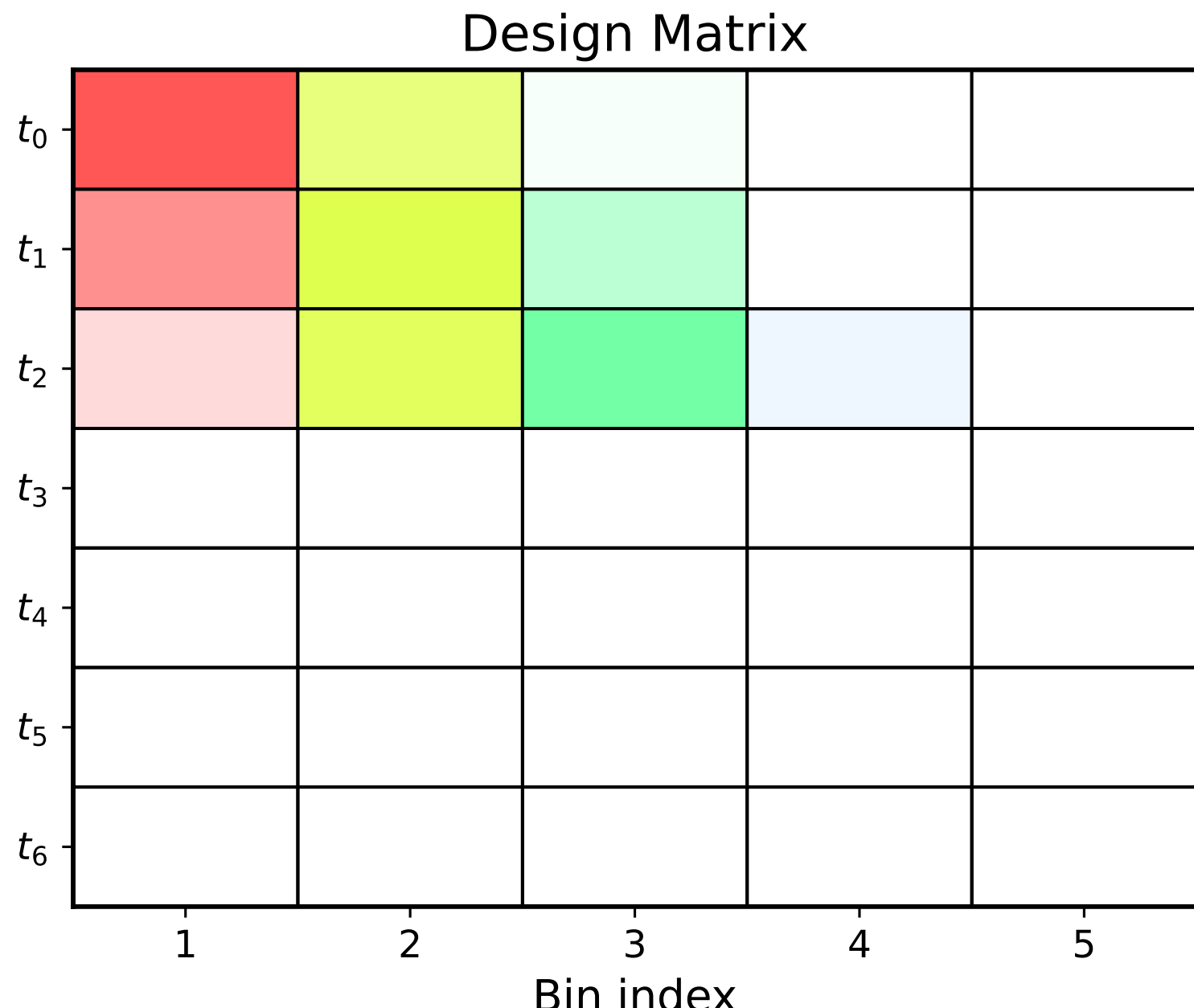
\times

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix}$$
 $)$

Example: 1D Non-Linear Rate Map

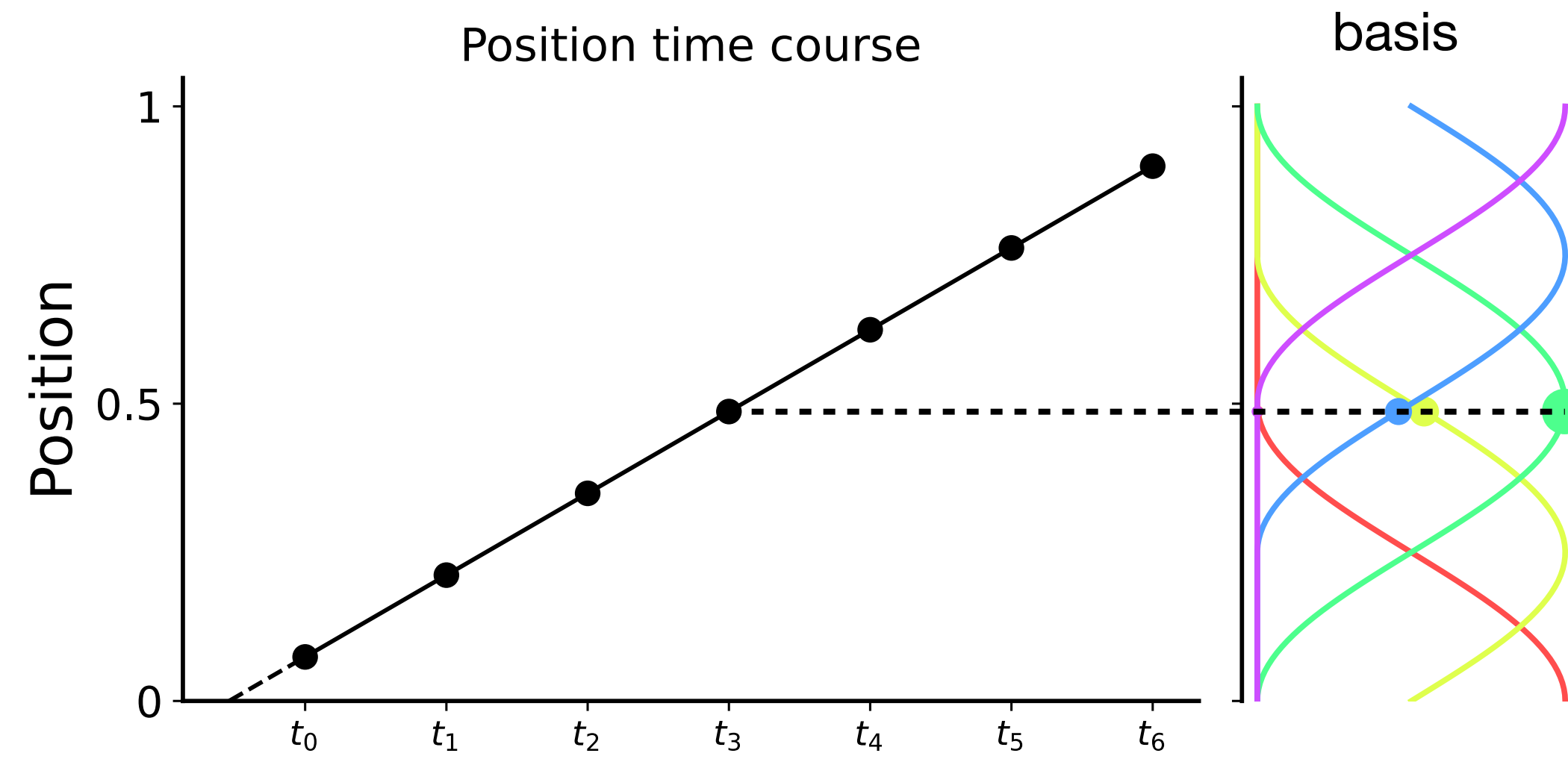


firing rate = exp(

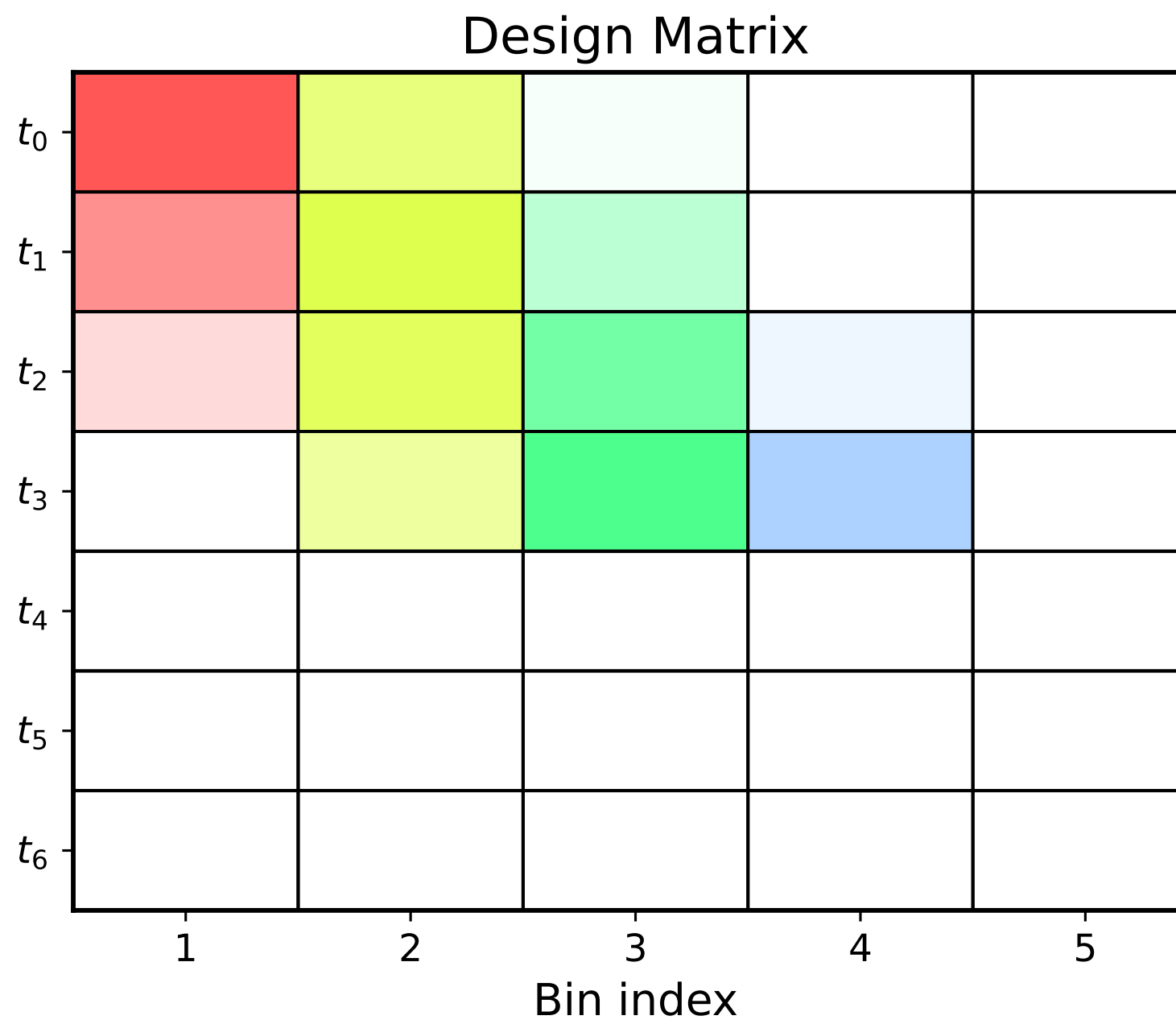


$$\times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix})$$

Example: 1D Non-Linear Rate Map



firing rate = $\exp($



\times

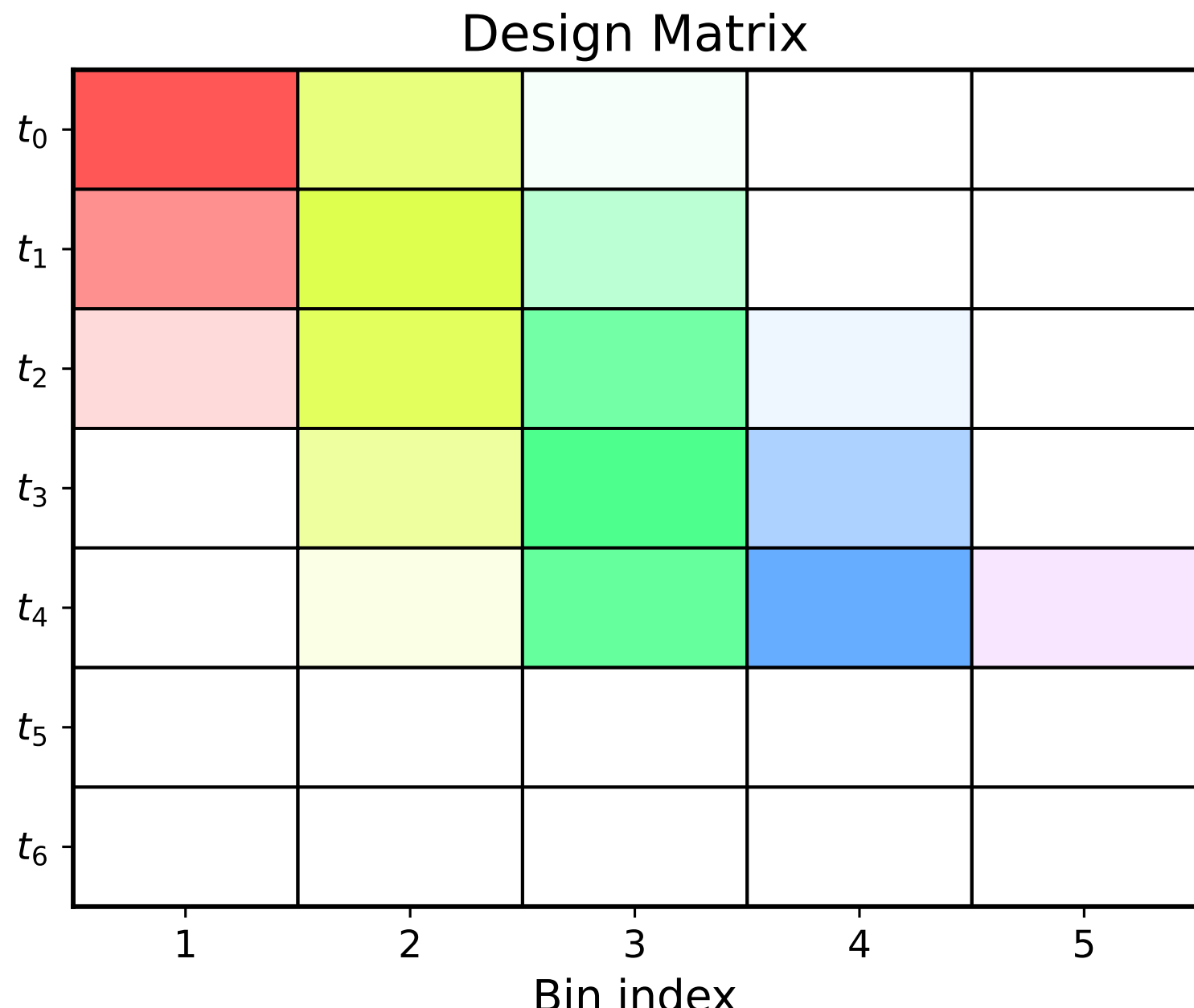
$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix}$$

)

Example: 1D Non-Linear Rate Map

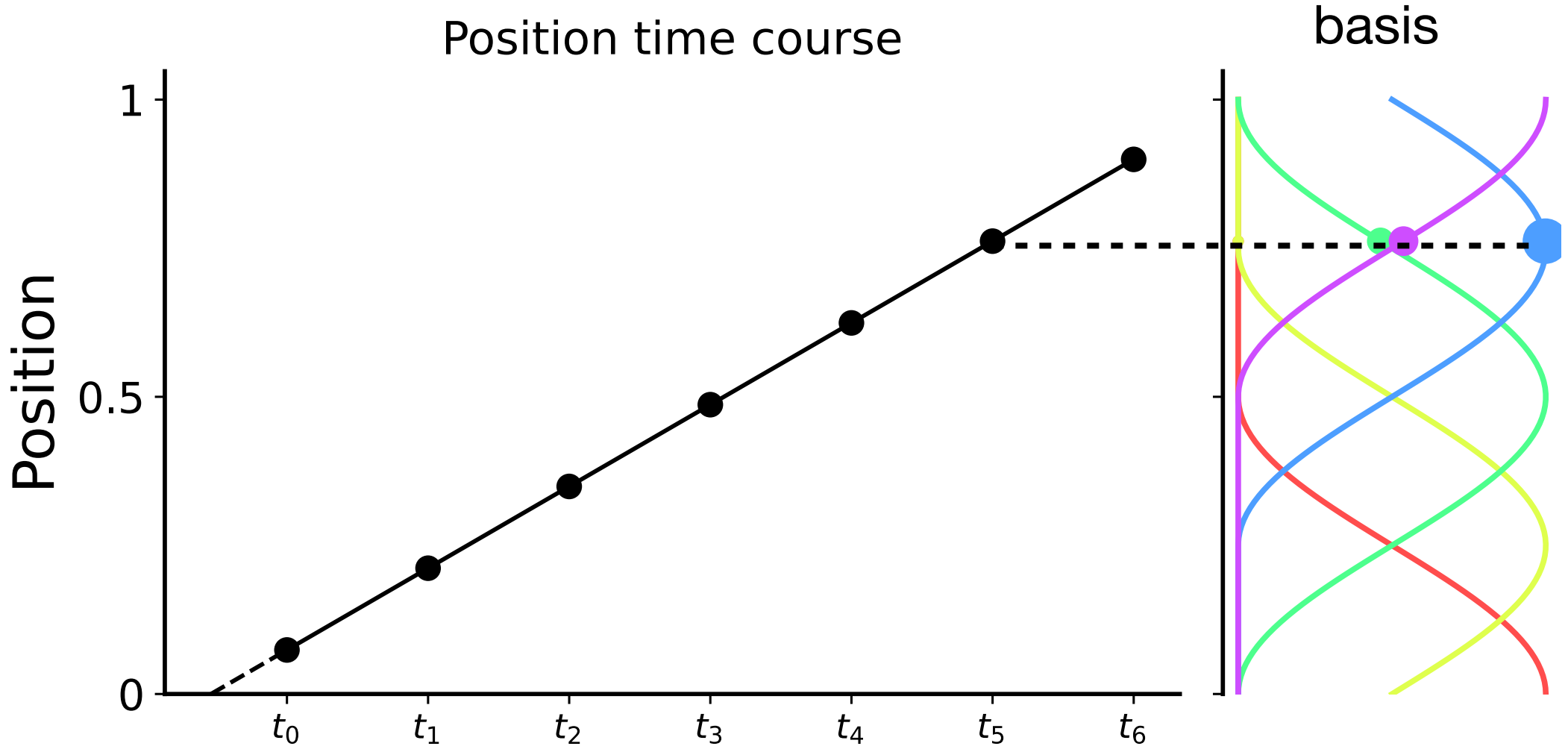


firing rate = exp(

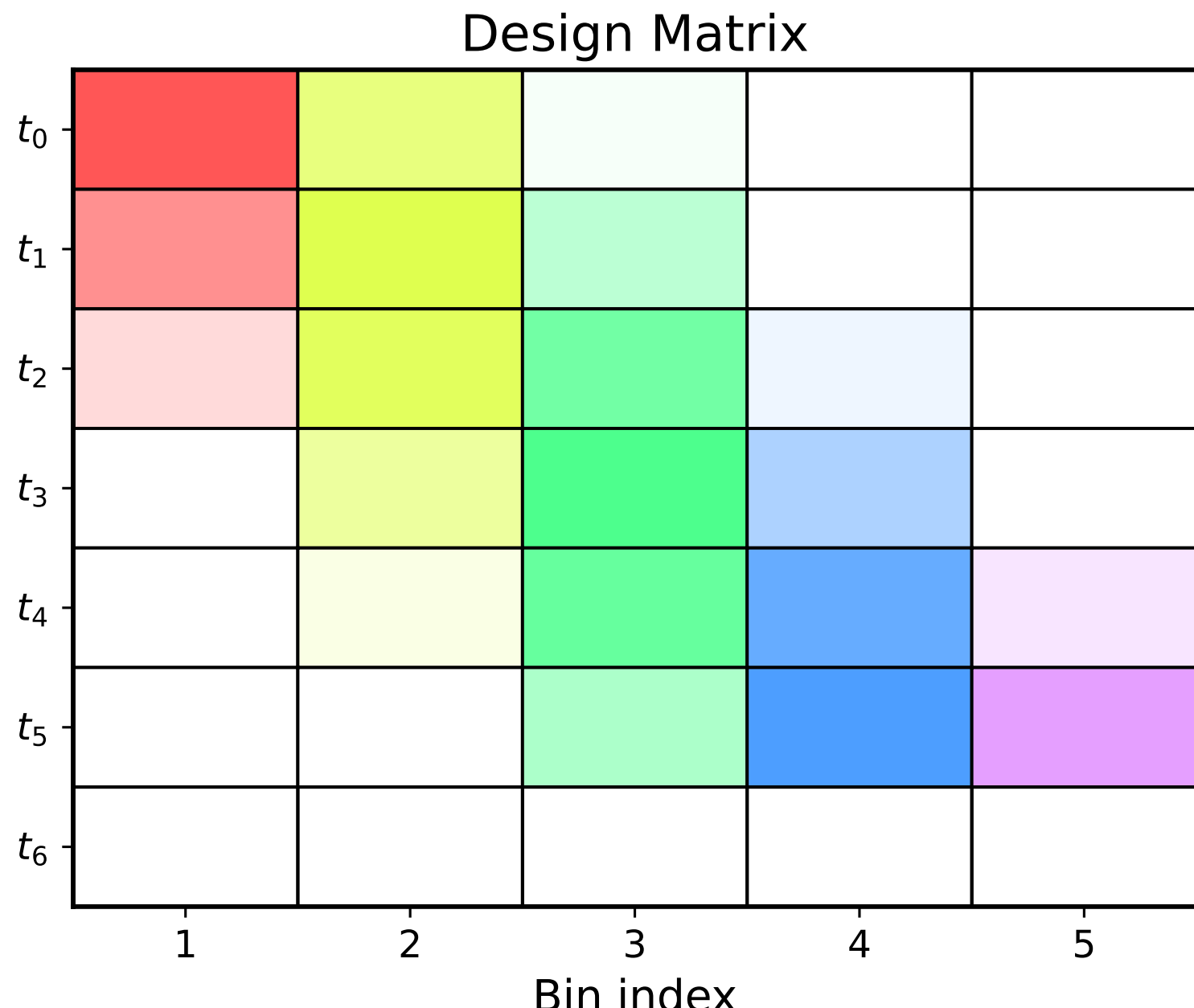


$$\times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix})$$

Example: 1D Non-Linear Rate Map

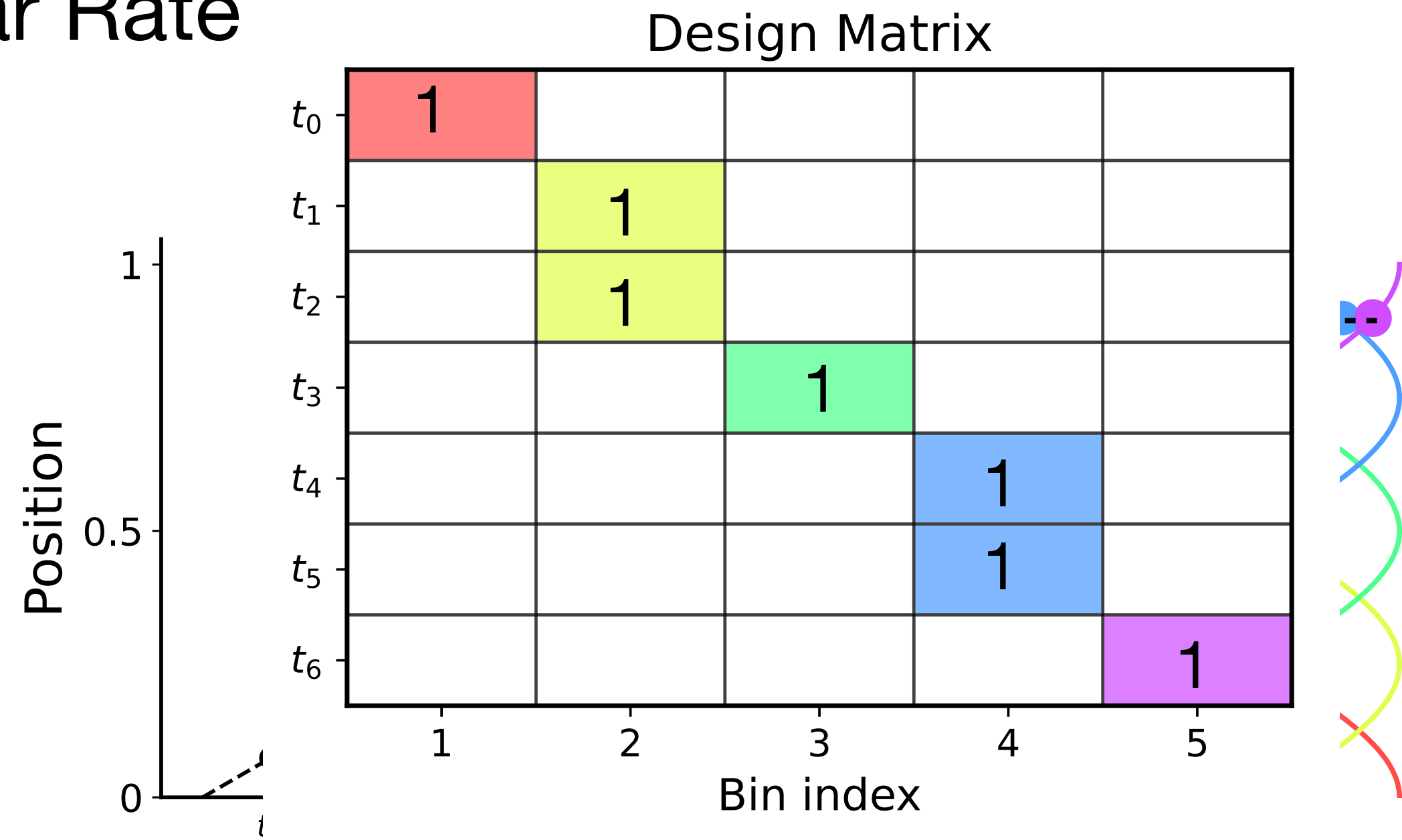


firing rate = exp(

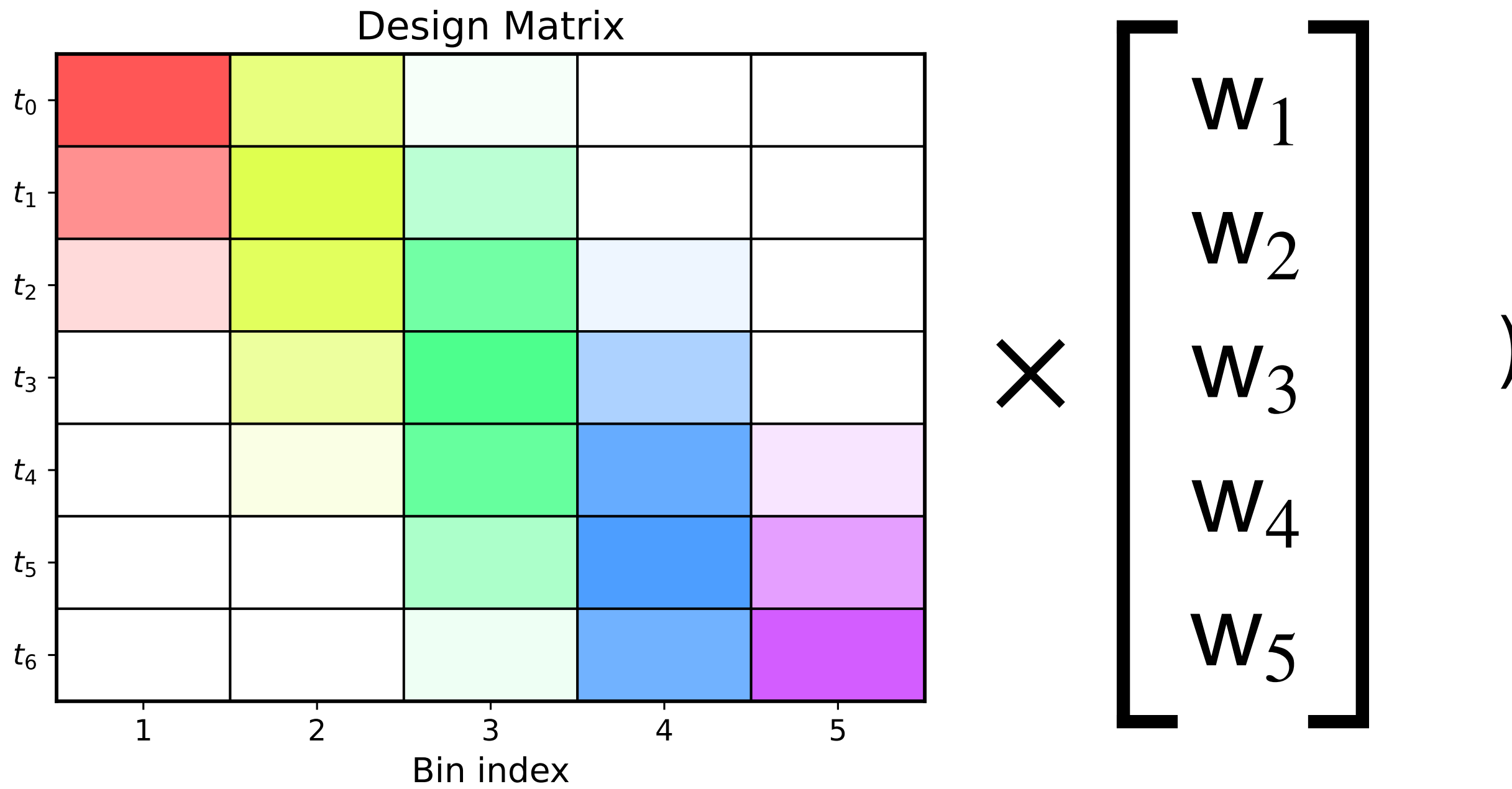


$$\times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix})$$

Example: 1D Non-Linear Rate



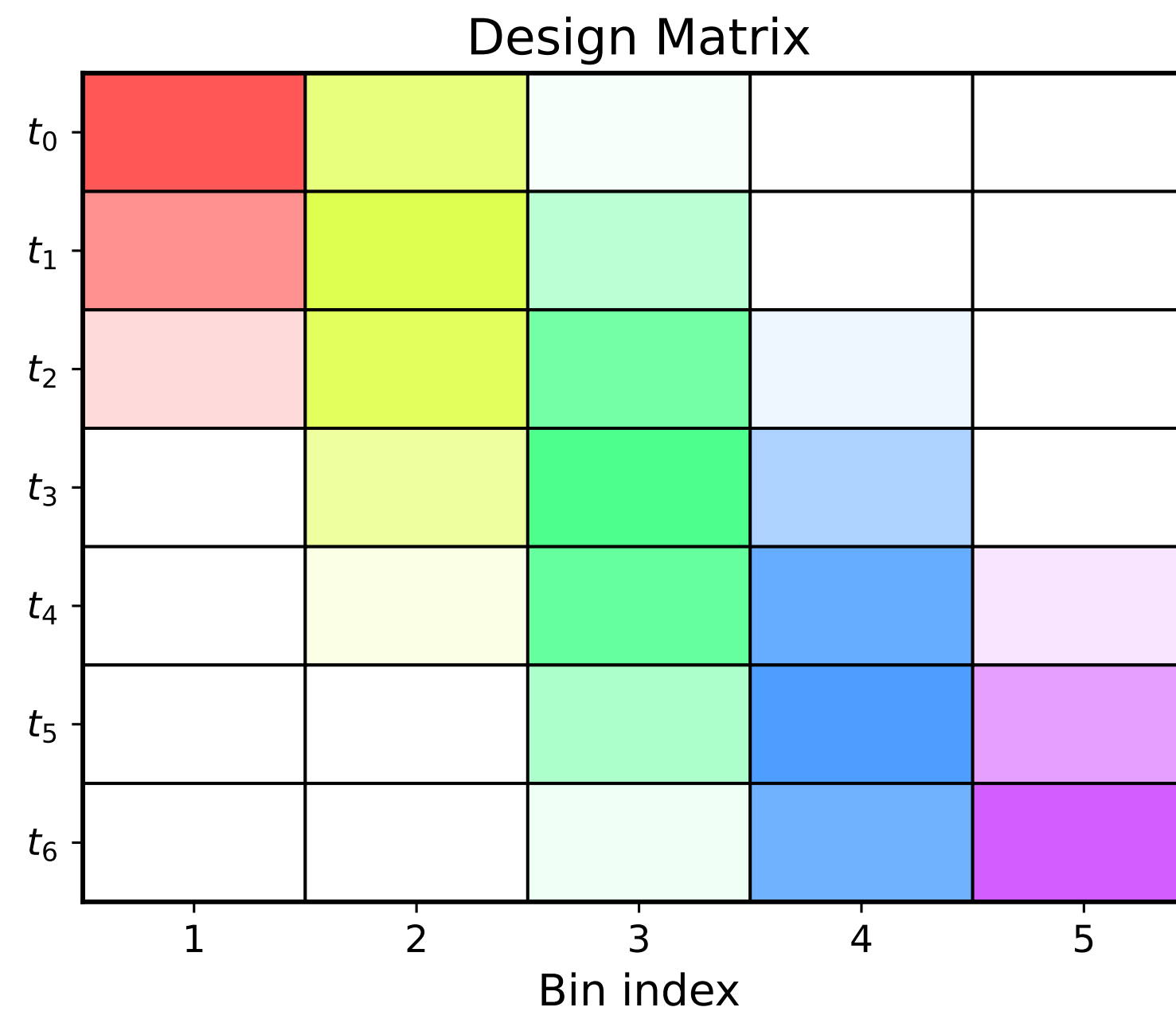
firing rate = $\exp($



1D Non-Linear Rate Map in

How to define the design r

- Define an `Eval` basis obje



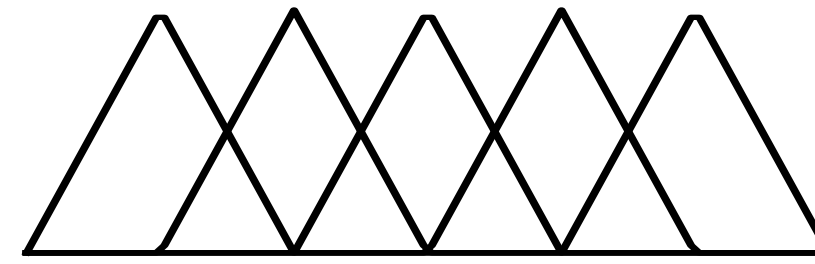
```
In [22]: basis = nmo.basis.RaisedCosineLinearEval(5)
```

Extending to 2D

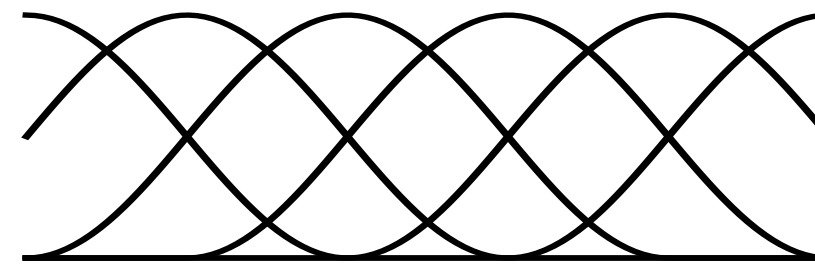
- 2D basis = 1D basis \times 1D basis

Extending to 2D

- 2D basis = 1D basis \times 1D basis



x



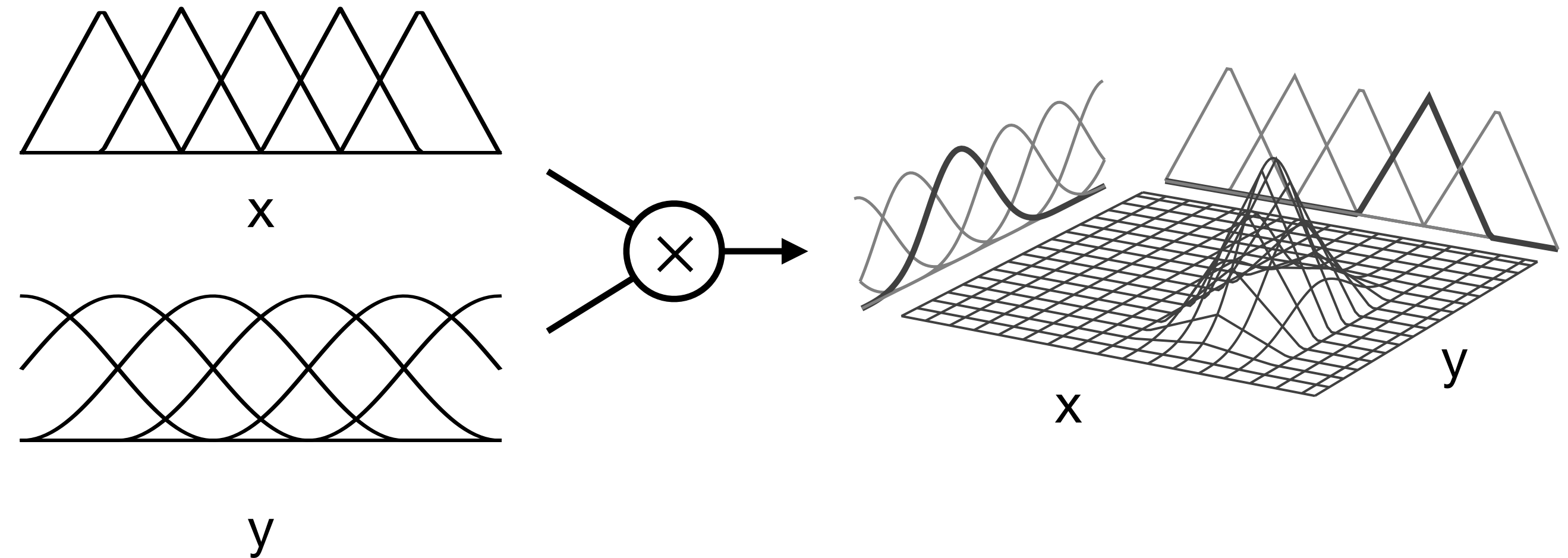
y

```
In [16]: b_x = nmo.basis.BSplineEval(5, order=2)
```

```
In [17]: b_y = nmo.basis.RaisedCosineLinearEval(6)
```

Extending to 2D

- 2D basis = 1D basis \times 1D basis
- Each 2D feature: $b_x(x) \cdot b_y(y)$



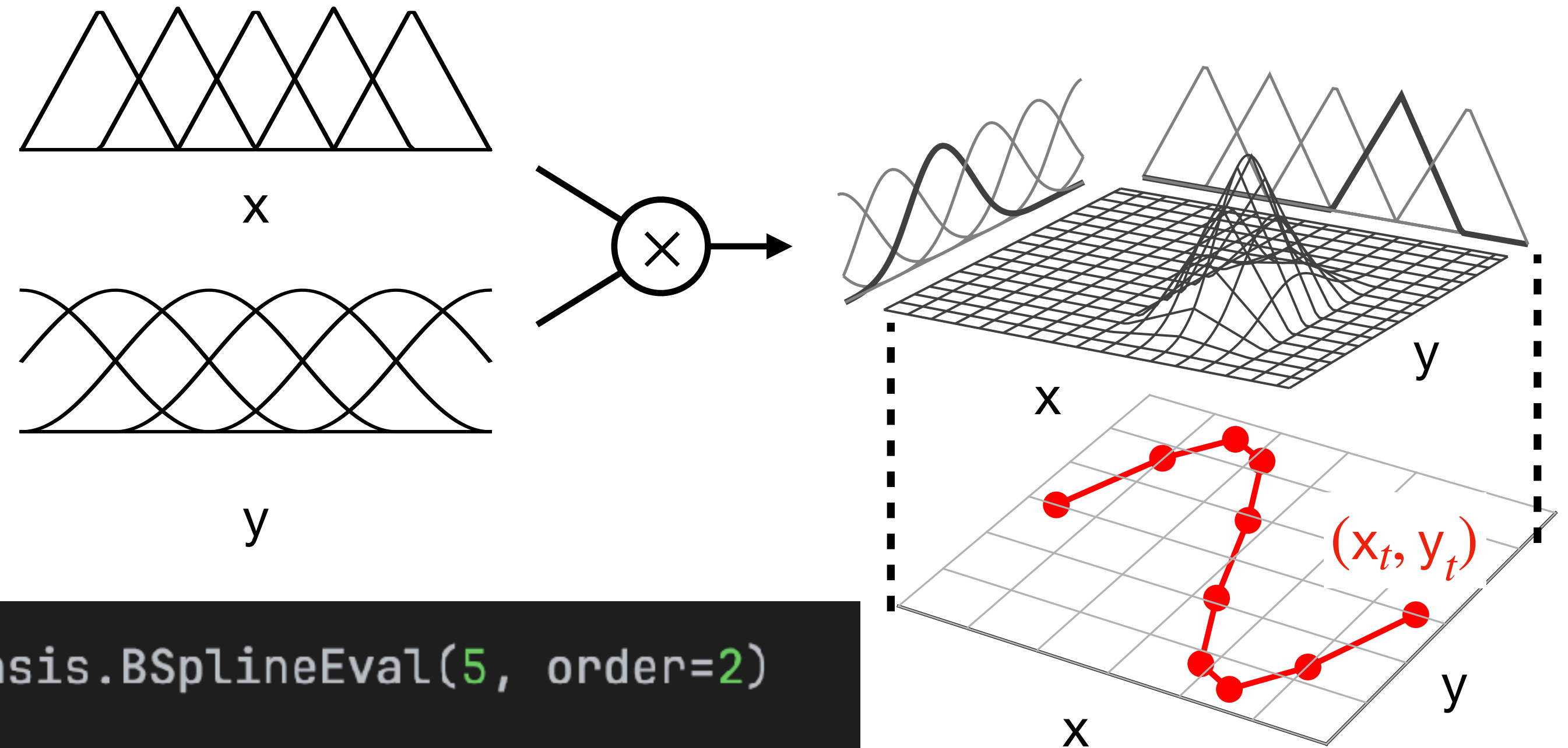
```
In [16]: b_x = nmo.basis.BSplineEval(5, order=2)
```

```
In [17]: b_y = nmo.basis.RaisedCosineLinearEval(6)
```

```
In [18]: bas_2d = b_x * b_y
```

Extending to 2D

- 2D basis = 1D basis \times 1D basis
- Each 2D feature: $b_x(x) \cdot b_y(y)$
- $5 \times 6 = 30$ features



```
In [16]: b_x = nmo.basis.BSplineEval(5, order=2)
```

```
In [17]: b_y = nmo.basis.RaisedCosineLinearEval(6)
```

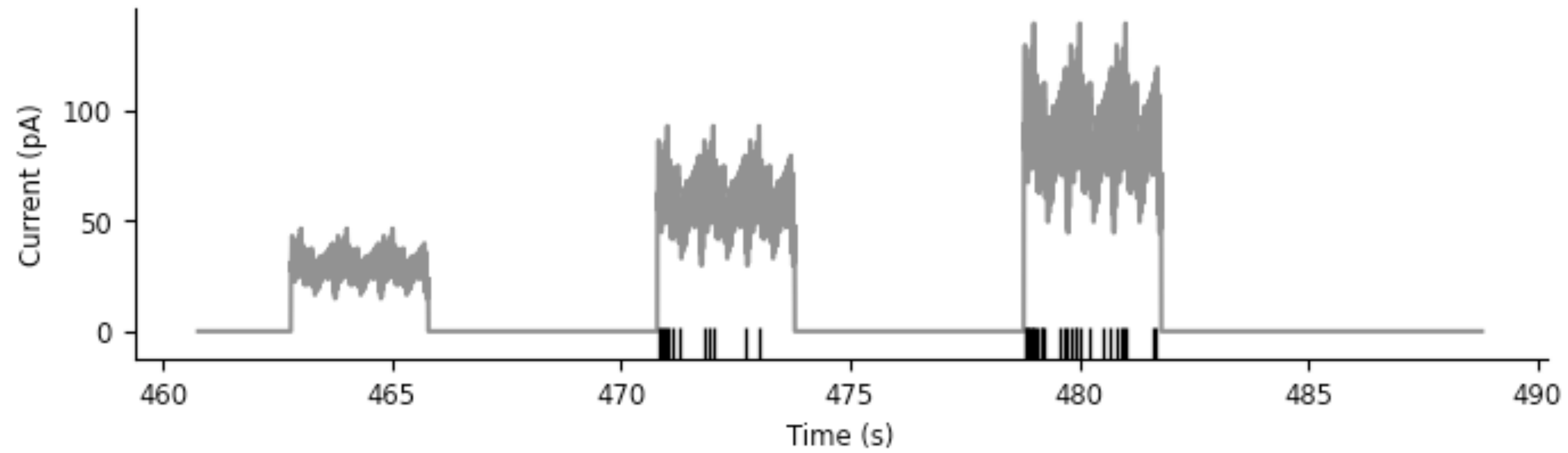
```
In [18]: bas_2d = b_x * b_y
```

```
In [19]: X = bas_2d.compute_features(x, y)
```

```
In [20]: X.shape
```

```
Out[20]: (10, 30)
```

Example: History Effects



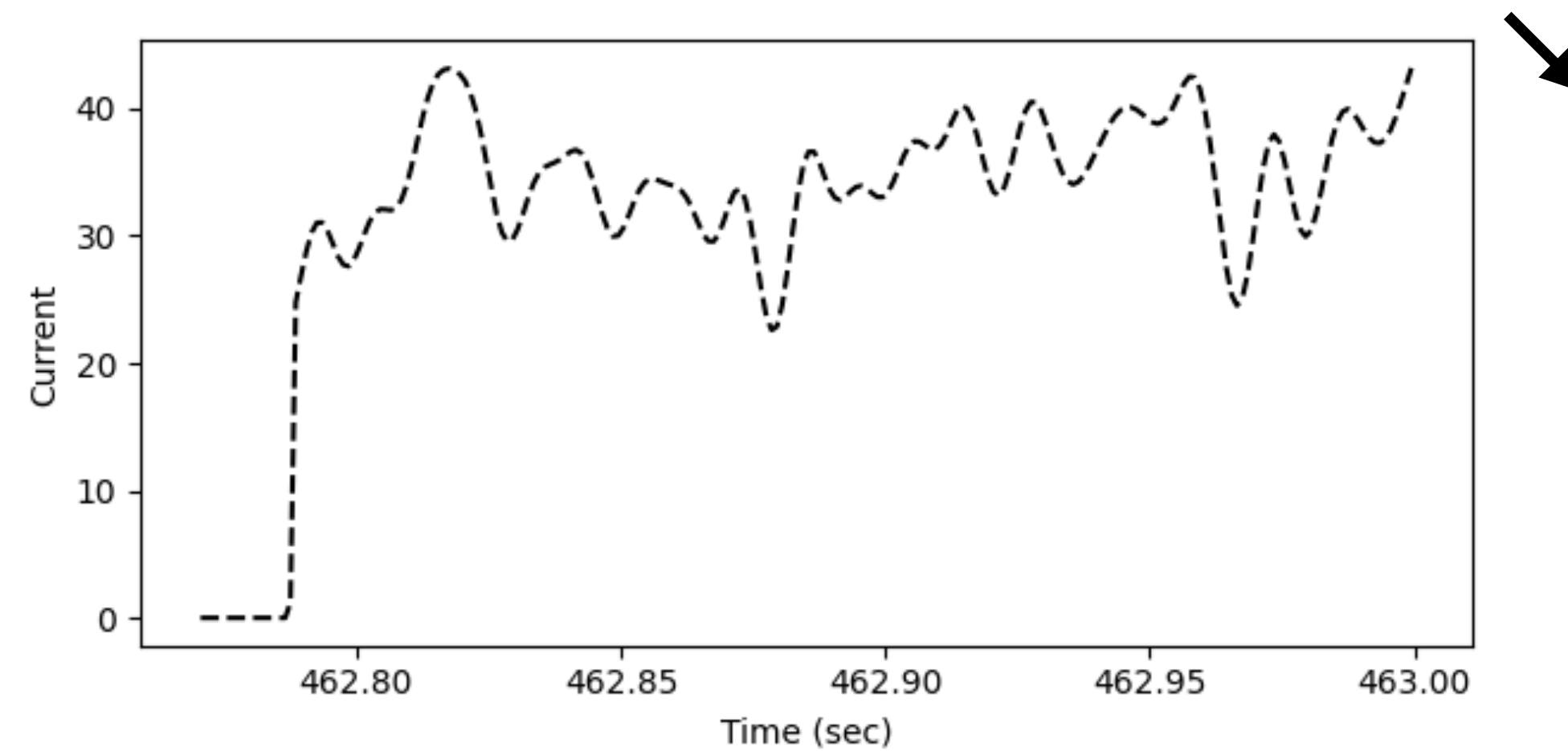
— Injected current

||| Evoked spikes

Example: History Effects

Simple design matrix construction:

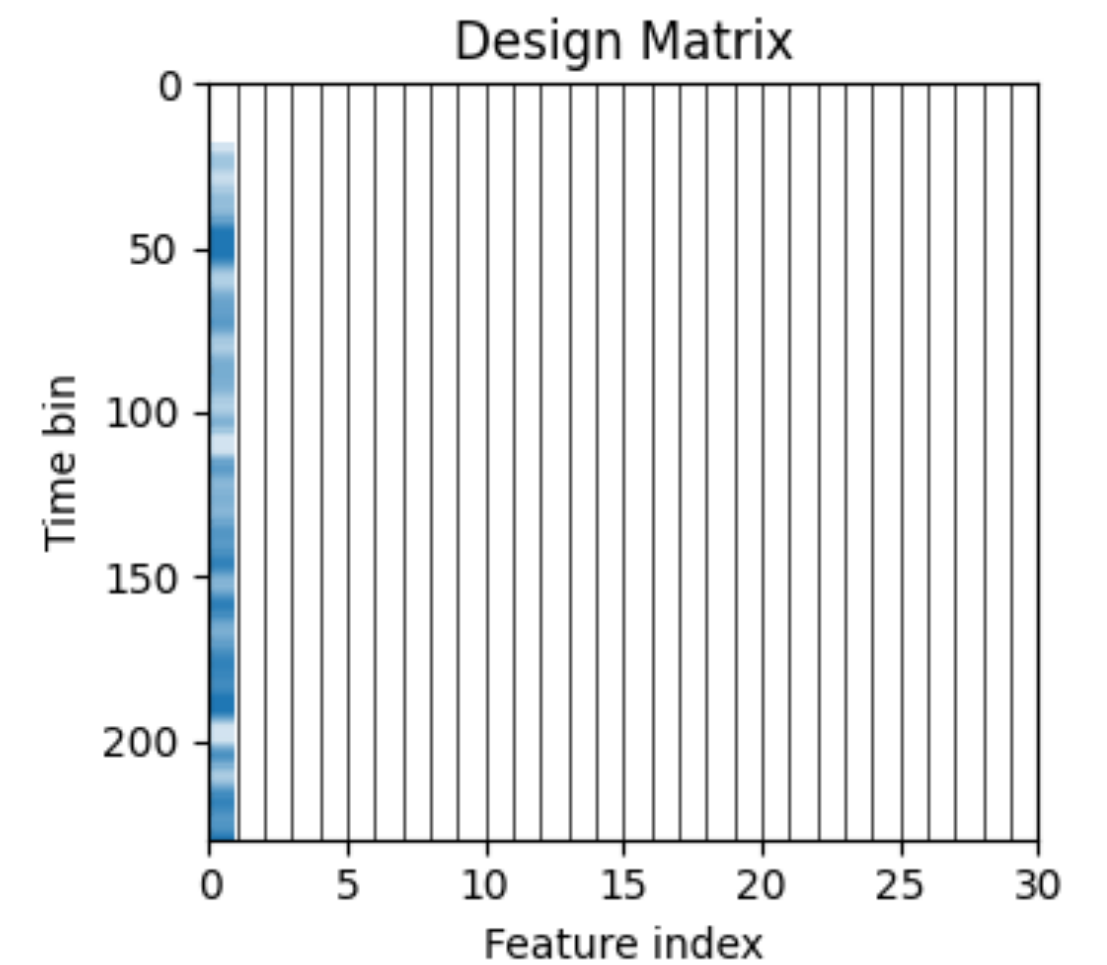
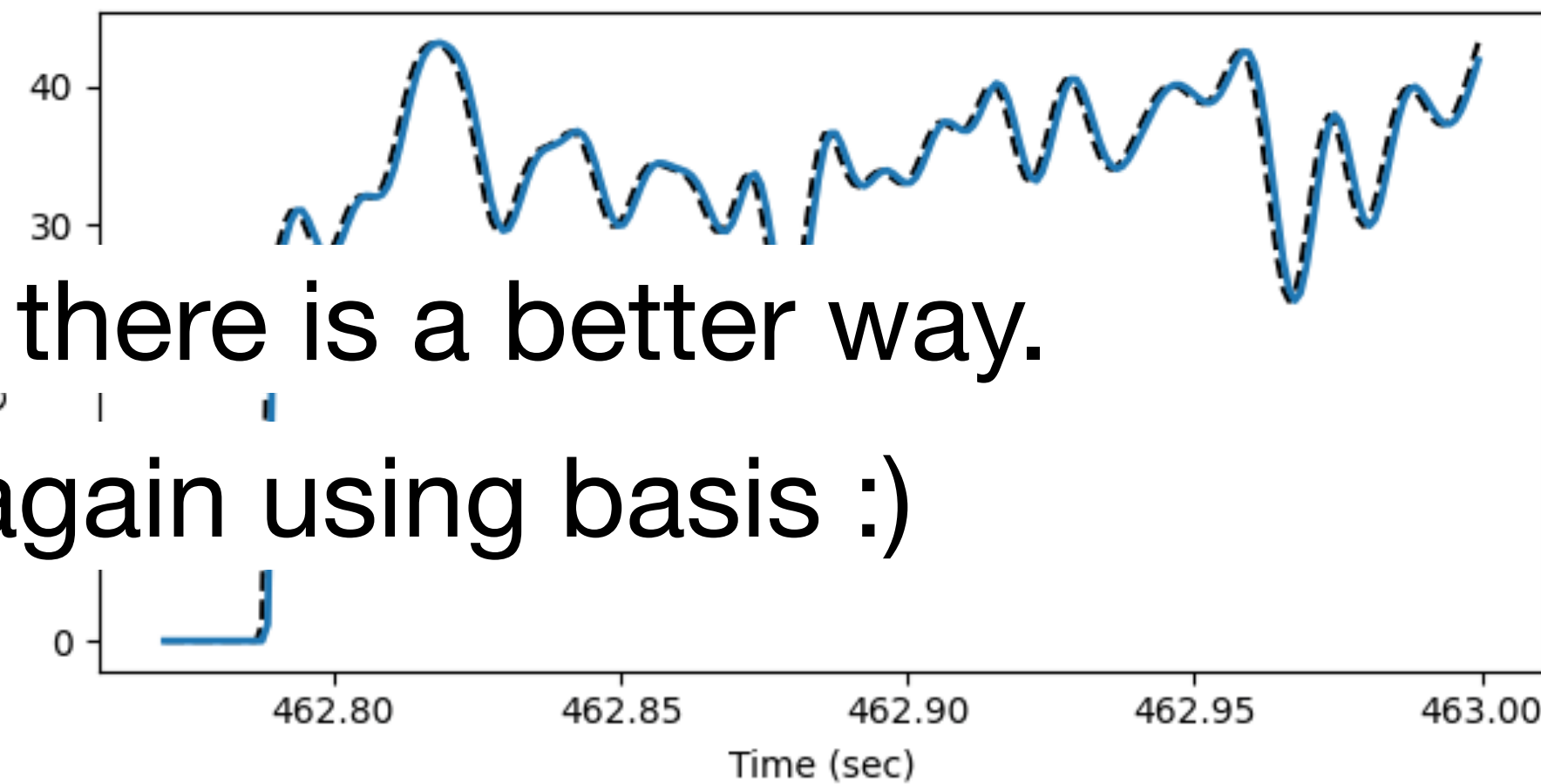
- Select a history window (e.g. 30 samples window)



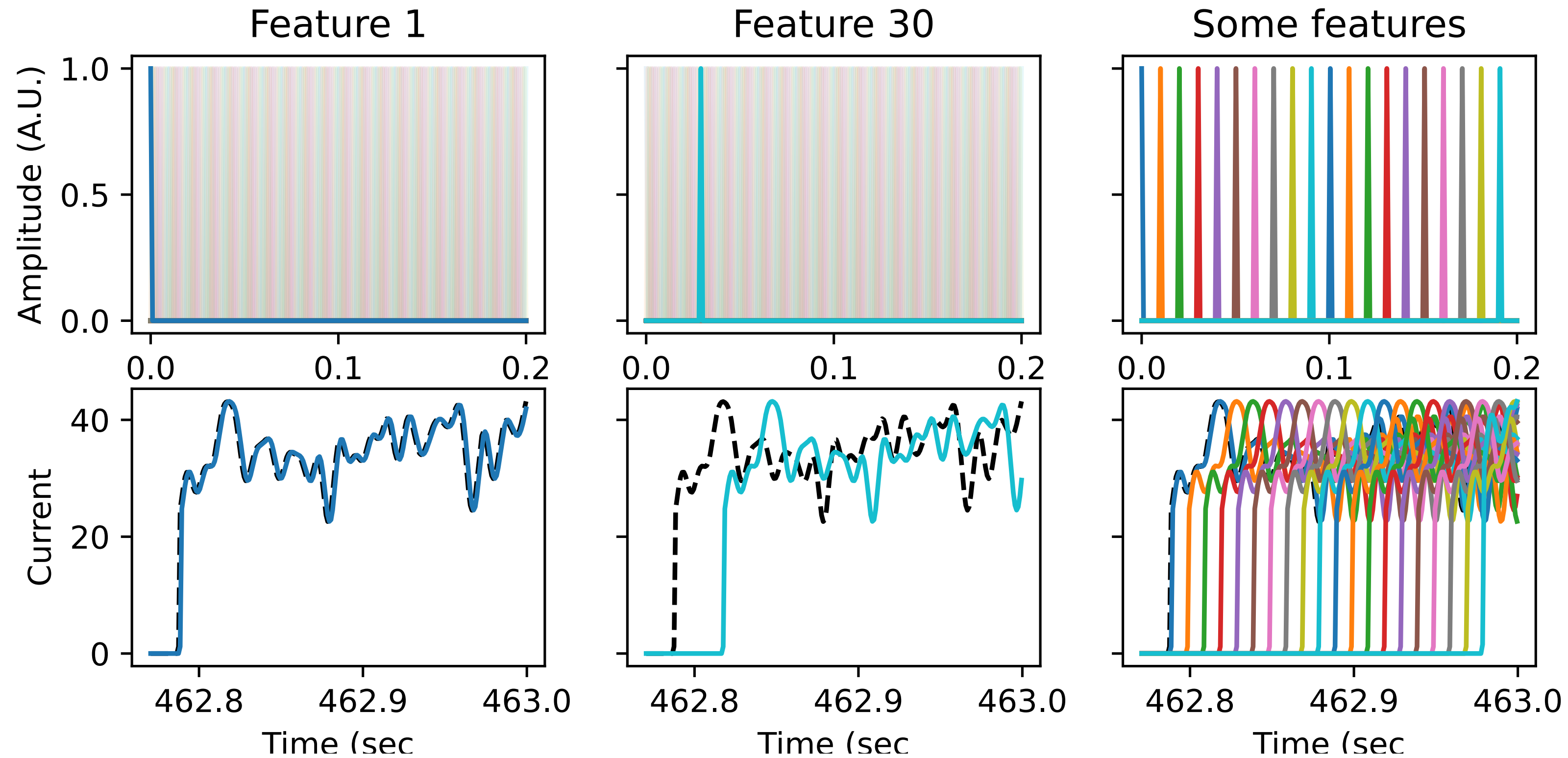
Example: History Effects

Simple design matrix construction:

- Repeat for whole length cOften there is a better way.
the history window
(e.g. 30 samples window)
...again using basis :)



Example: History Effects



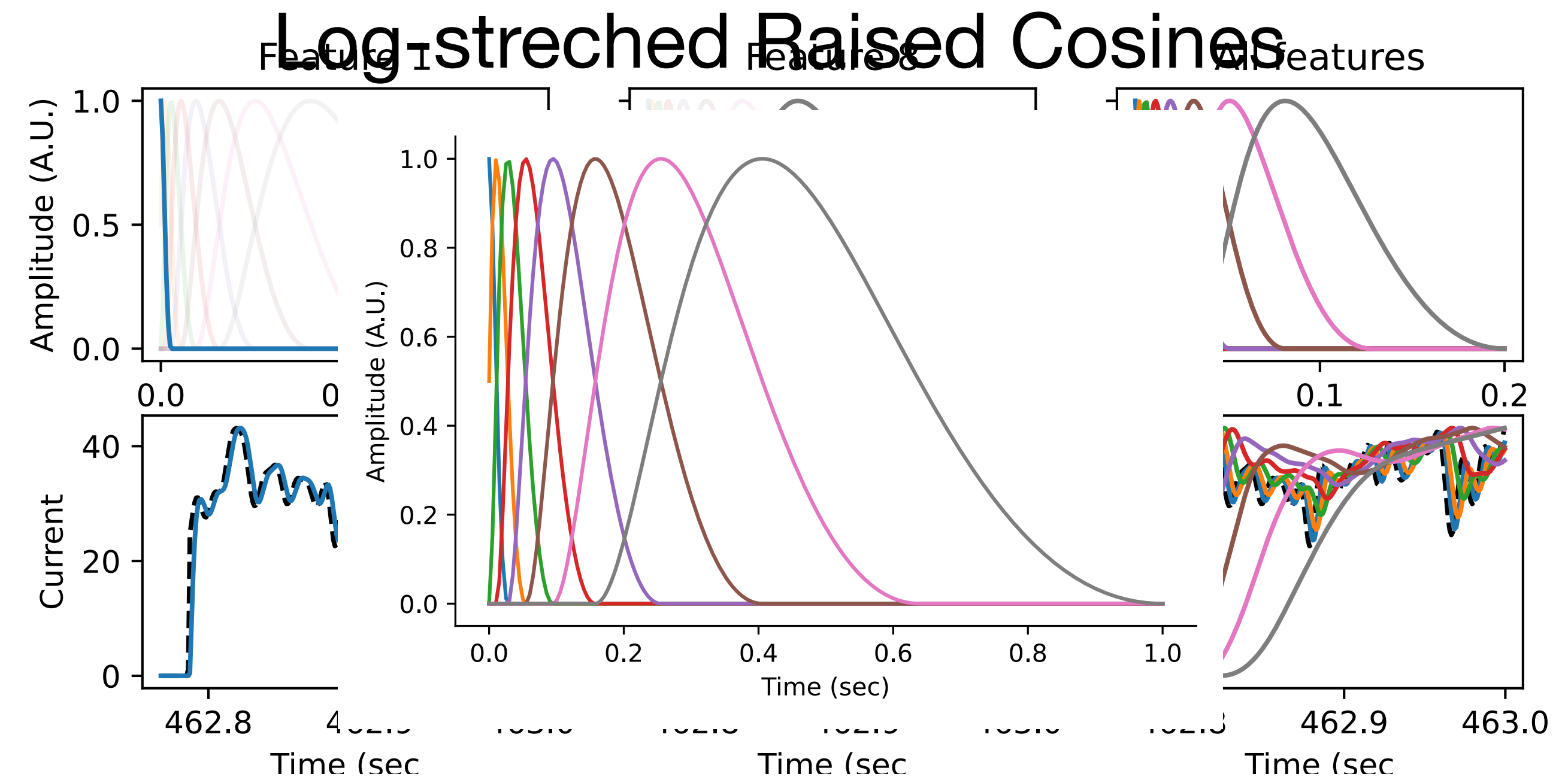
extracting shifted copies of the signal = convolution with delta functions

convolutional equivalent of 1-hot encoding

Example: History Effects

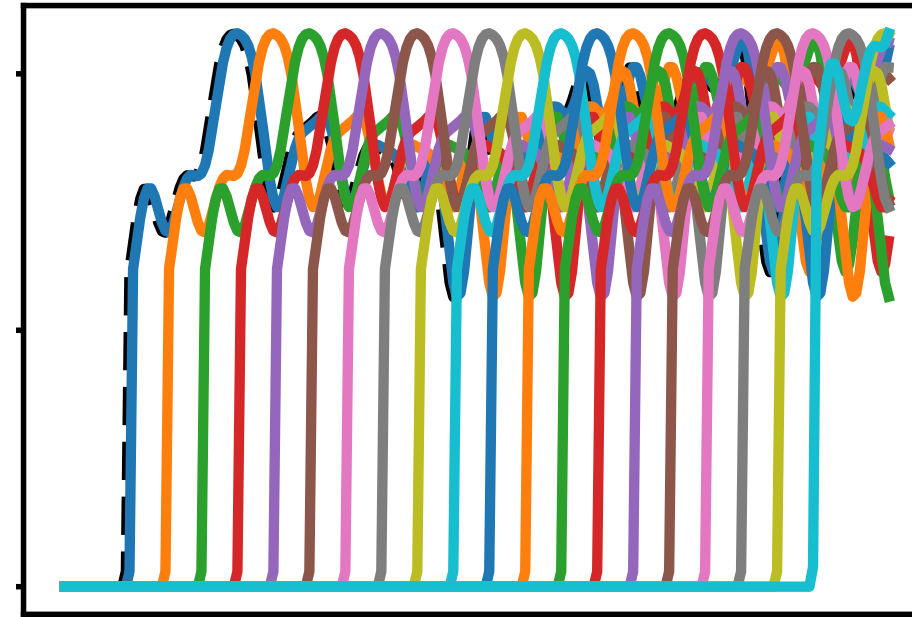
Reduce dimensions:

- Replace the deltas with basis functions
- Convolve the current with the basis

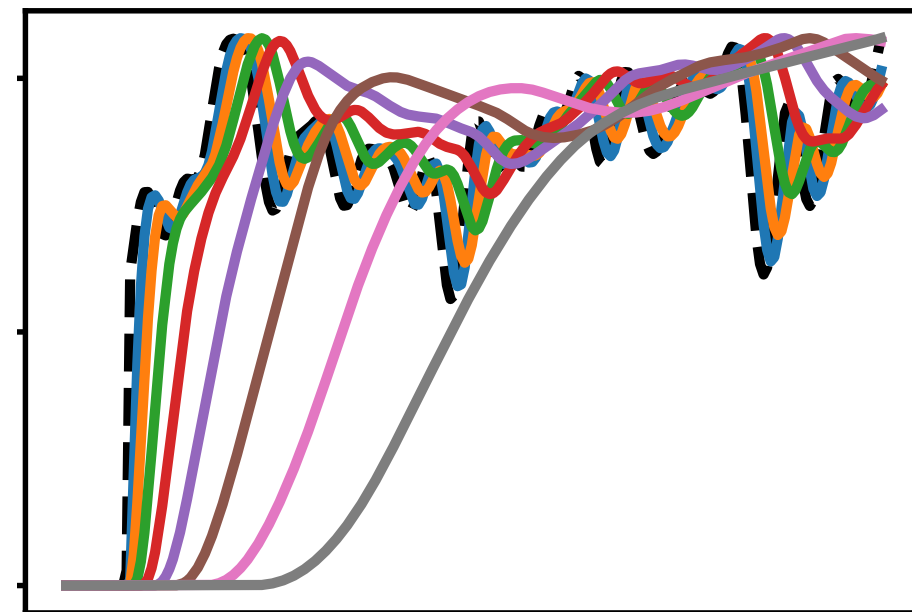


Example: History Effects

Shifted current:



Convolved with basis:



```
In [68]: window_size=100
```

```
In [69]: raw_history_basis = nmo.basis.HistoryConv(window_size)
```

```
In [70]: X = raw_history_basis.compute_features(current)
```

```
In [71]: X.shape
```

```
Out[71]: (1000, 100)
```

```
In [78]: window_size=100
```

```
In [79]: basis_history = nmo.basis.RaisedCosineLogConv(  
...:     window_size=window_size,  
...:     n_basis_funcs=8  
...: )
```

```
In [80]: X = basis_history.compute_features(current)
```

```
In [81]: X.shape
```

```
Out[81]: (1000, 8)
```

Multiple Predictors

- Multiple predictors:
speed, position, head direction...

Multiple Predictors

- Multiple predictors:
speed, position, head direction...
- A basis per predictor

```
In [43]: b_sp = nmo.basis.BSplineEval(4)
In [44]: b_pos = nmo.basis.MSplineEval(5)
In [45]: b_hd = nmo.basis.CyclicBSplineEval(6)
```

b_sp

b_pos

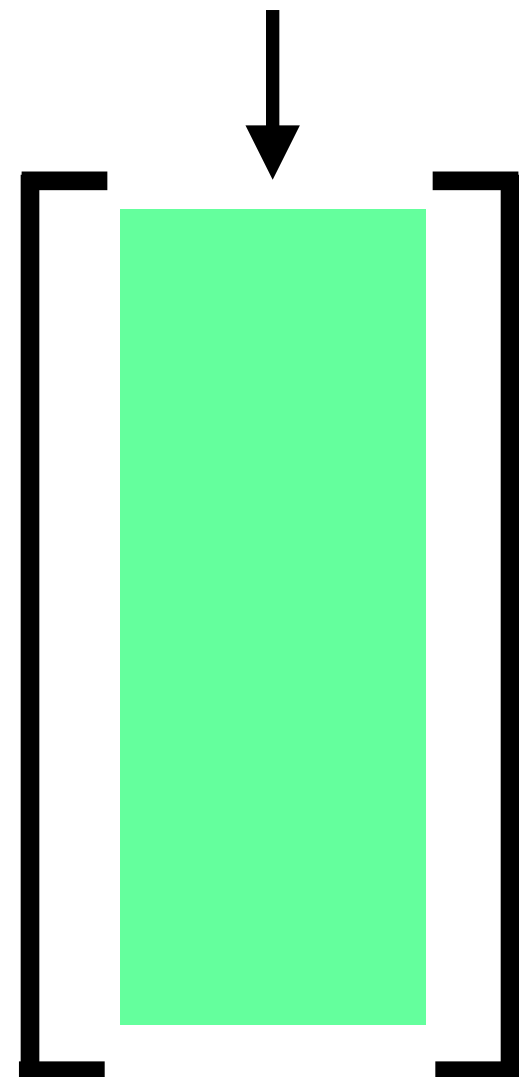
b_hd

Multiple Predictors

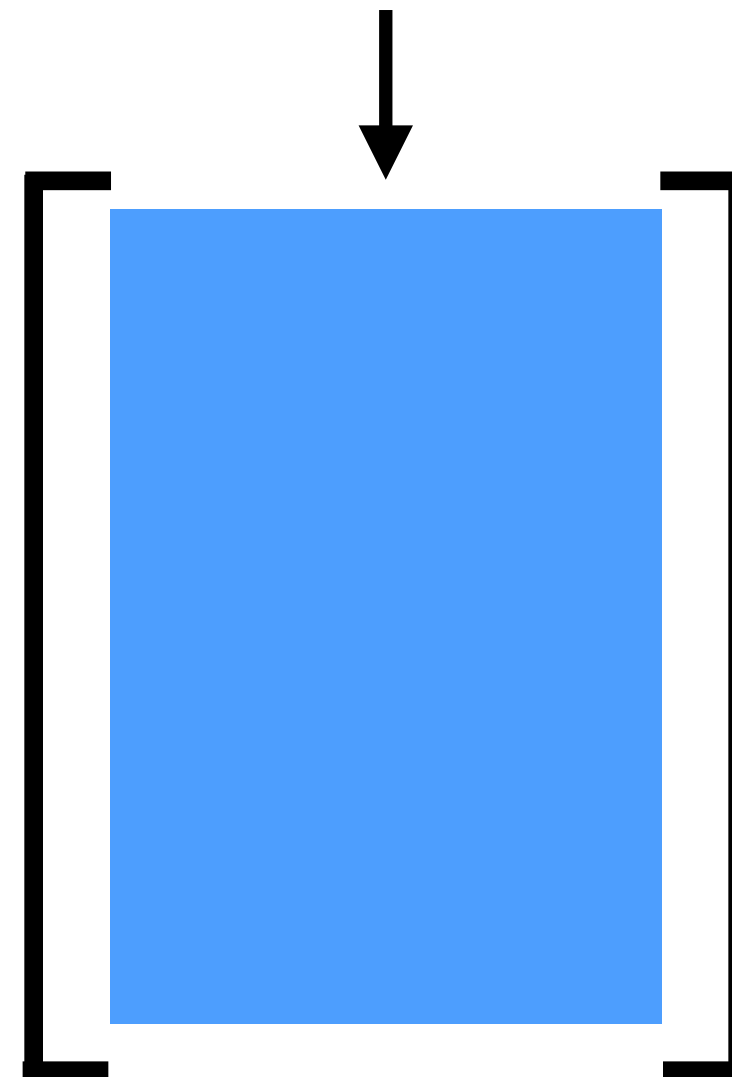
- Multiple predictors:
speed, position, head direction...
- A basis per predictor

```
In [43]: b_sp = nmo.basis.BSplineEval(4)
In [44]: b_pos = nmo.basis.MSplineEval(5)
In [45]: b_hd = nmo.basis.CyclicBSplineEval(6)
```

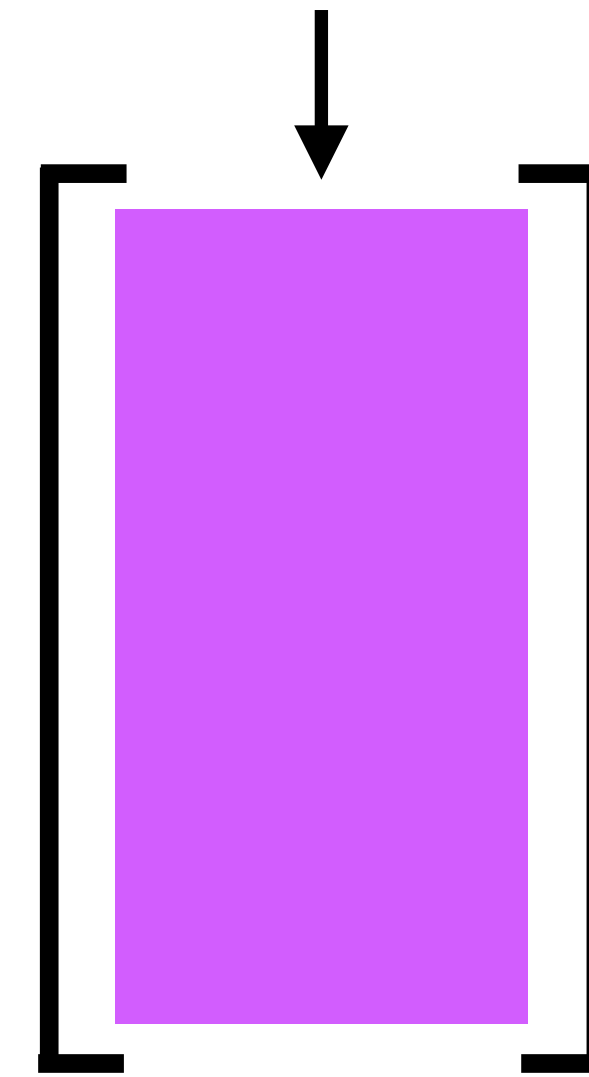
b_sp.compute_features(speed)



b_pos.compute_features(position)



b_hd.compute_features(head_dir)



Multiple Predictors

- Multiple predictors: speed, position, head direction...
- A basis per predictor
- Add the basis

```
In [43]: b_sp = nmo.basis.BSplineEval(4)
In [44]: b_pos = nmo.basis.MSplineEval(5)
In [45]: b_hd = nmo.basis.CyclicBSplineEval(6)
In [46]: add_basis = b_sp + b_pos + b_hd
```

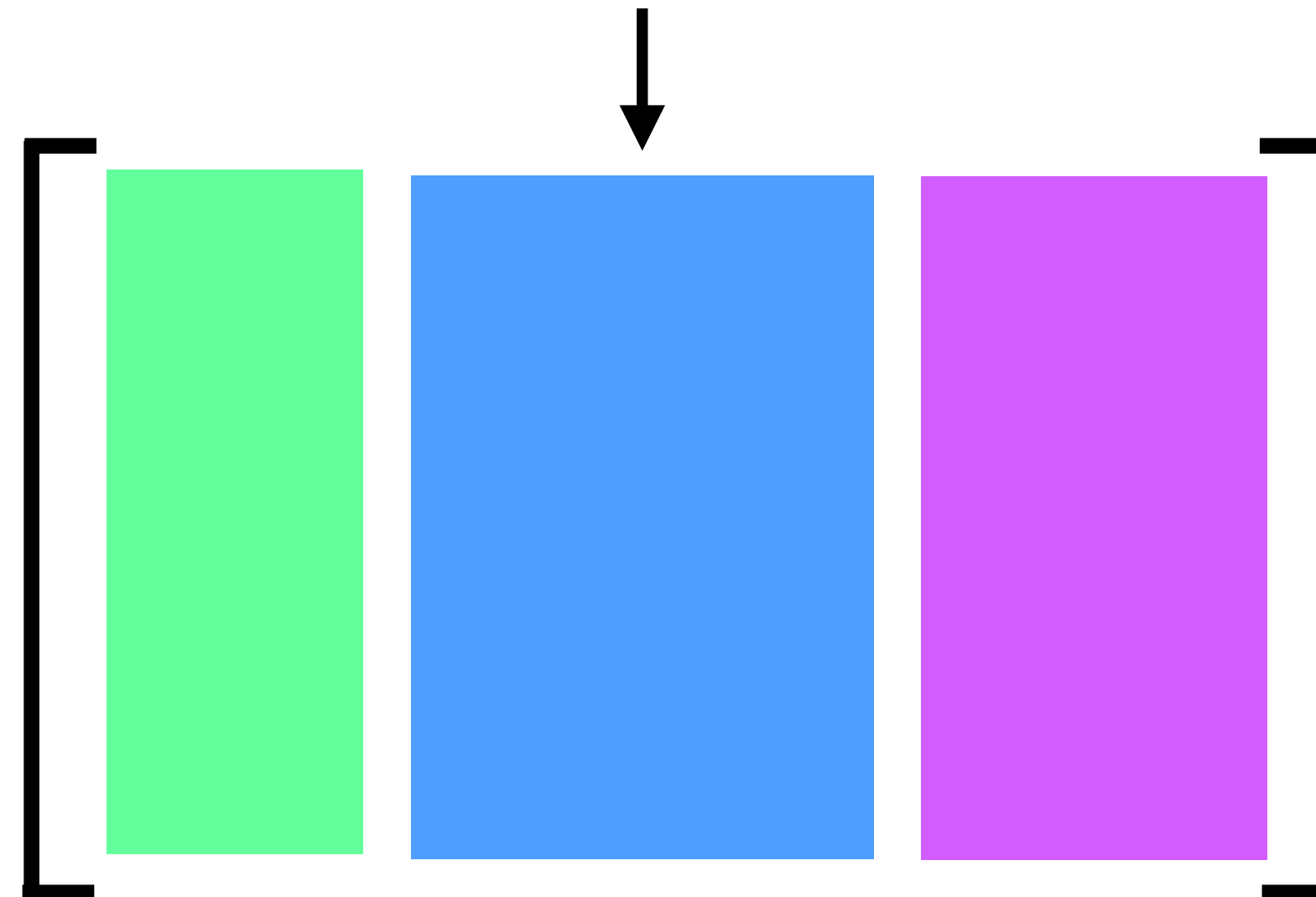
```
add_basis = b_sp + b_pos + b_hd
```

Multiple Predictors

- Multiple predictors: speed, position, head direction...
- A basis per predictor
- Add the basis
- Total features: $4 + 5 + 6 = 15$

```
In [43]: b_sp = nmo.basis.BSplineEval(4)
In [44]: b_pos = nmo.basis.MSplineEval(5)
In [45]: b_hd = nmo.basis.CyclicBSplineEval(6)
In [46]: add_basis = b_sp + b_pos + b_hd
In [47]: X = add_basis.compute_features(speed, position, head_dir)
In [48]: X.shape
Out[48]: (6, 15)
```

`add_basis.compute_features(speed, position, head_dir)`



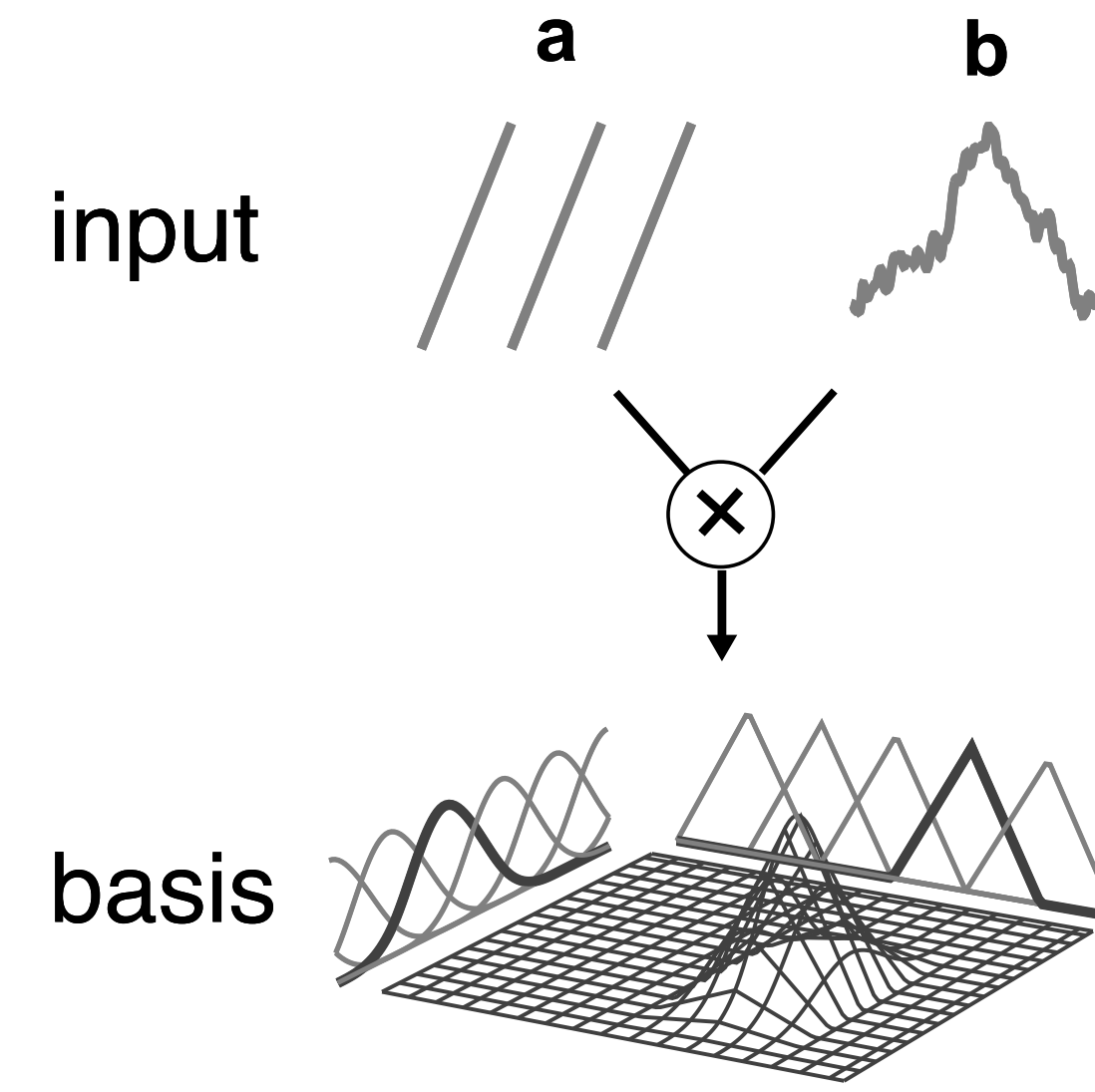
Summarizing

- **Fixed non-linearities** for feature construction
- Exploit smoothness, less weights
- **Type of bases:**
 - *Eval*: evaluation (non-linear tuning)
 - *Conv*: convolution (temporal effects)
 - *Multiply*: interactions (e.g., 2D)
 - *Add*: multiple predictors

Summarizing

```
In [54]: b1 = nmo.basis.MSplineEval(5)
```

```
In [55]: b2 = nmo.basis.BSplineEval(5, order=2)
```



Summary

- Tuning functions don't tell the whole story, need better models

Why NeMoS?

1. Pynapple support:

- Carry over time information (and metadata)
- Handle disjoint epochs avoiding border artifacts

Why NeMoS?

1. Pynapple support:

- Carry over time information (and metadata)
- Handle disjoint epochs avoiding border artifacts

2. Neuroscience specificity:

- Raised Cosine basis, population GLMs, simplified model design

Why NeMoS?

1. Pynapple support:

- Carry over time information (and metadata)
- Handle disjoint epochs avoiding border artifacts

2. Neuroscience specificity:

- Raised Cosine basis, population GLMs, simplified model design

3. Scikit-learn compatibility:

- Well known, simple API

Why NeMoS?

1. Pynapple support:

- Carry over time information (and metadata)
- Handle disjoint epochs avoiding border artifacts

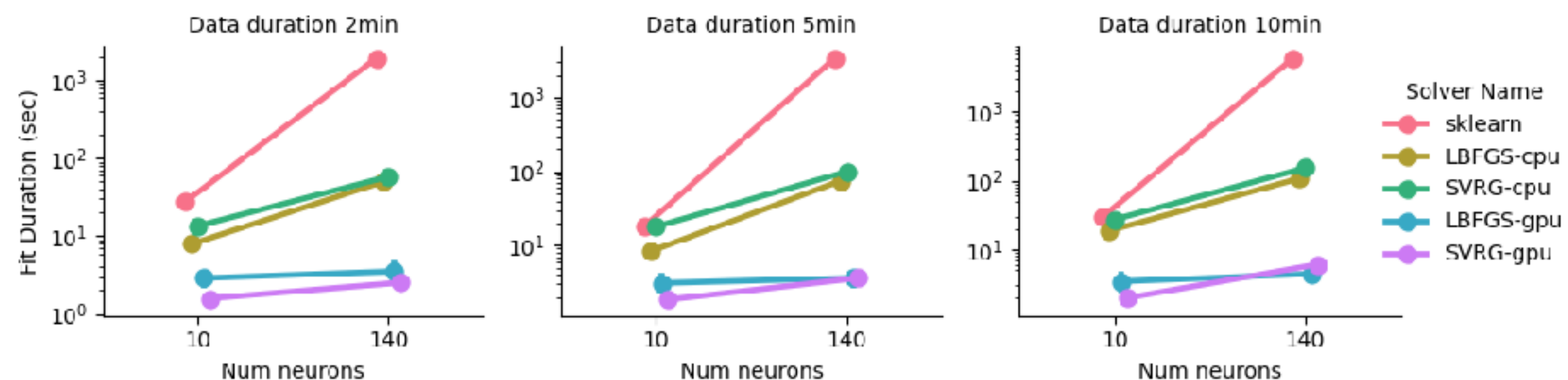
2. Neuroscience specificity:

- Raised Cosine basis, population GLMs, simplified model design

3. Scikit-learn compatibility:

- Well known, simple API

4. Performance (GPU)



Today's roadmap

(11:30 AM – 1:00 PM)

- **Current injection live coding:**

- Load and explore a intracellular recordings from the Allen Brain Map with pynapple.
- Fit an LNP model to a single input.
- Capture temporal effects using NeMoS' basis.

(2:00 PM – 3:30 PM)

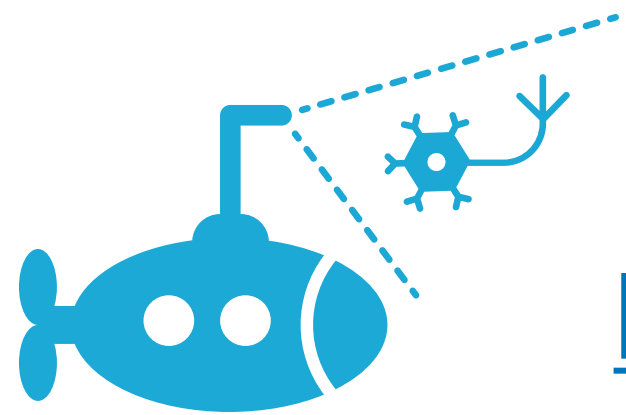
- **Group Projects: Analyze Head direction cells**

(4:30 PM – 6:00 PM)

- **Group Projects: Analyzing Calcium Imaging data with Pynapple and NeMoS**



Documentation Website



<https://nemos.readthedocs.io/en/stable/>



<https://pynapple.org/>



@nemos_neuro

@thepynapple