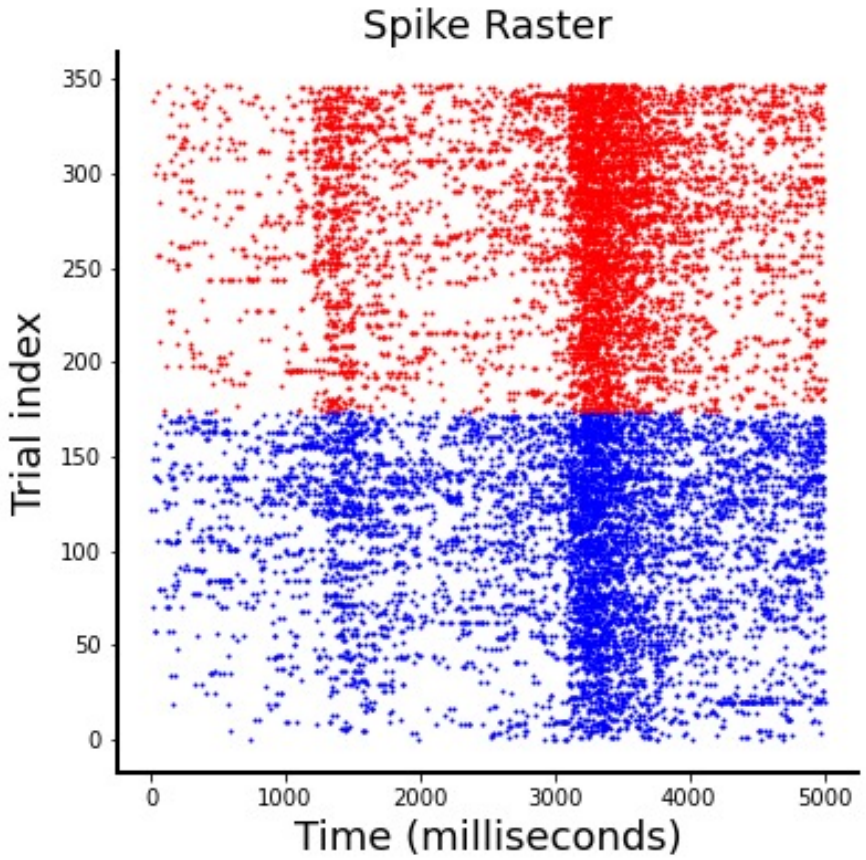
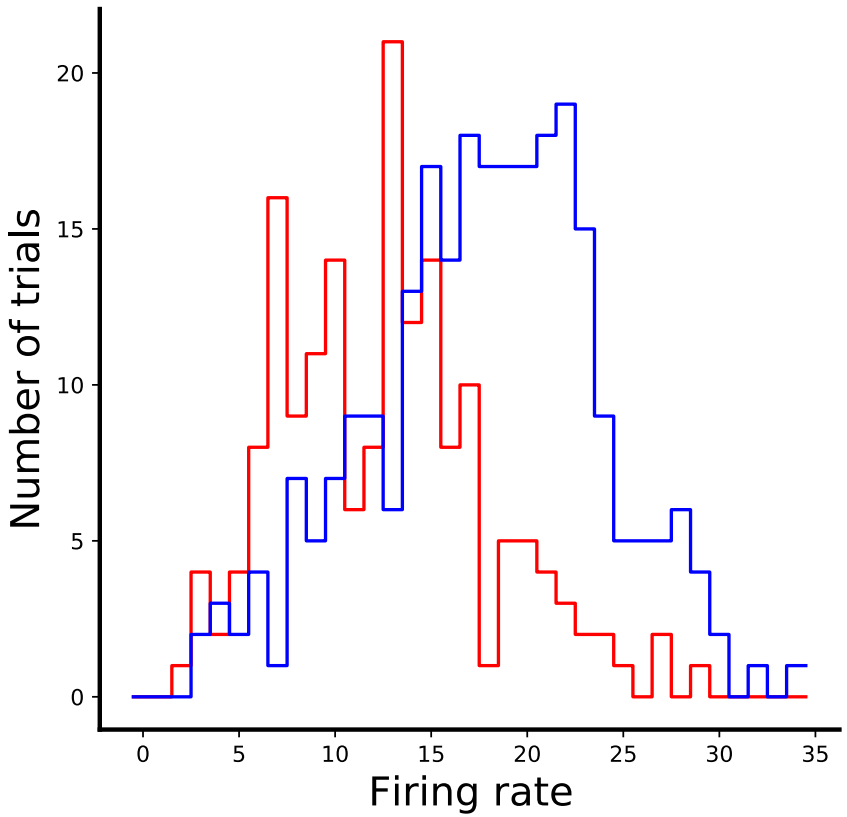
Introduction to decoding

At the single neuron level, decoding in trial-based designs is straightforward: asking whether the firing rate of a neuron is the same or different across conditions.

For example, is the firing rate the same or different for lick right and lick left trials in the delay period

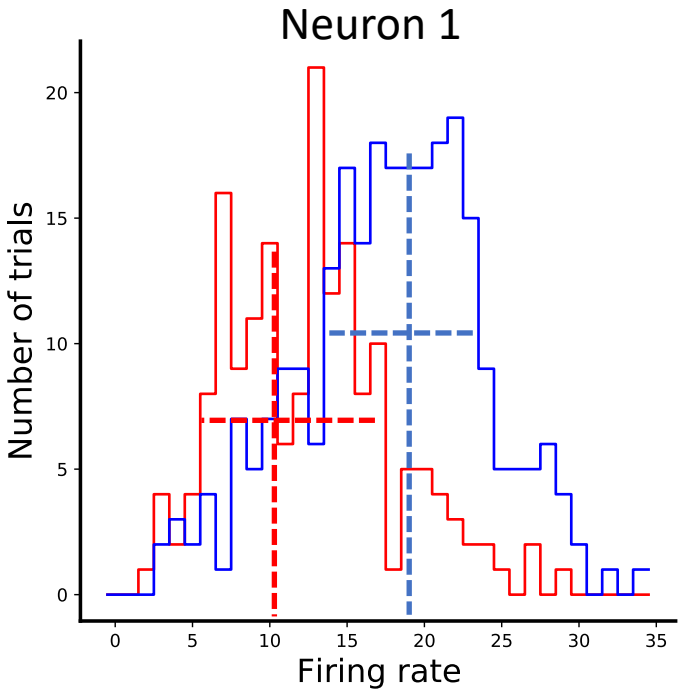




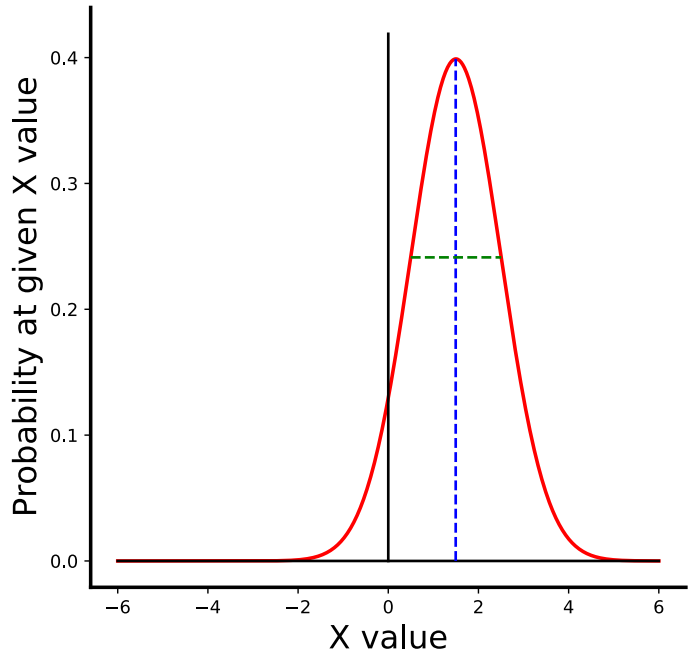
Quantifying decodability

D-prime: difference between the means of the two groups, normalized by the standard deviation within group

$$d' = \frac{\mu_1 - \mu_2}{\sigma_W}$$



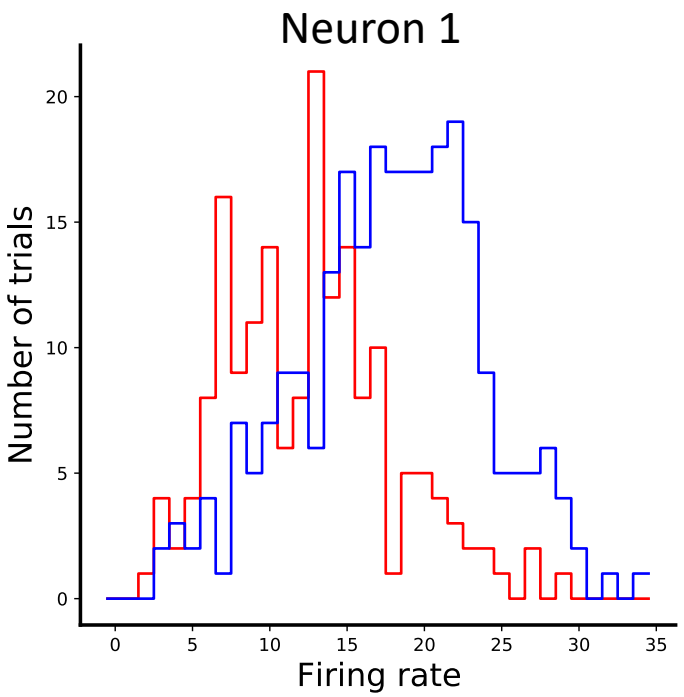
$$d'_{neuron\ 1} \approx 1$$

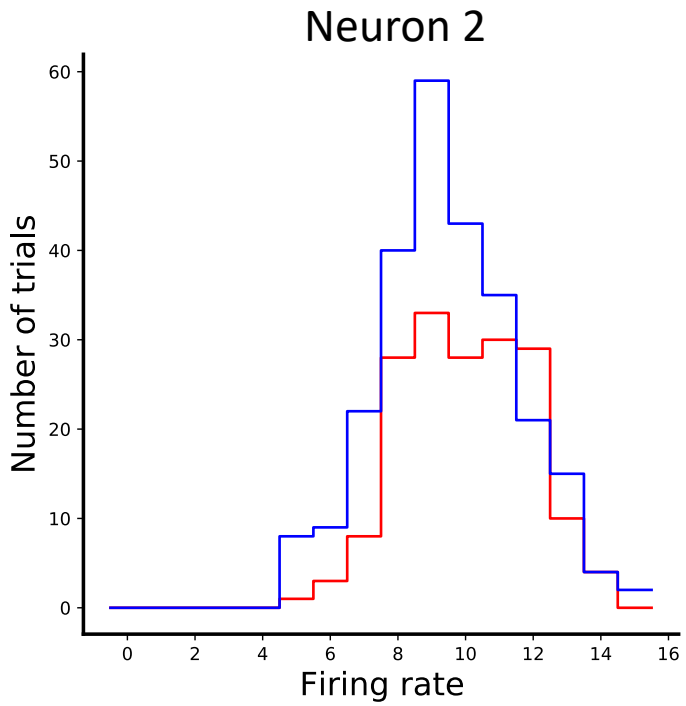


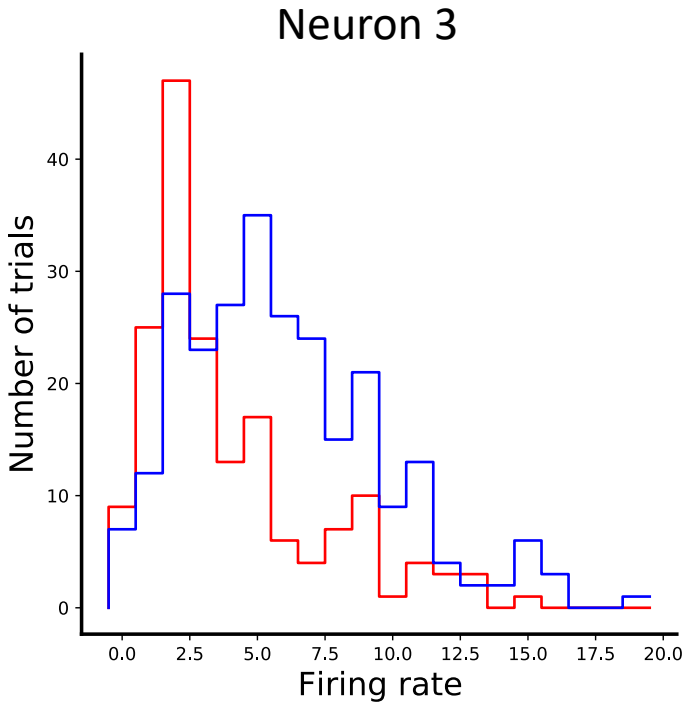
Quantifying decodability

D-prime: difference between the means of the two groups, normalized by the standard deviation within group

$$d' = \frac{\mu_1 - \mu_2}{\sigma_W}$$



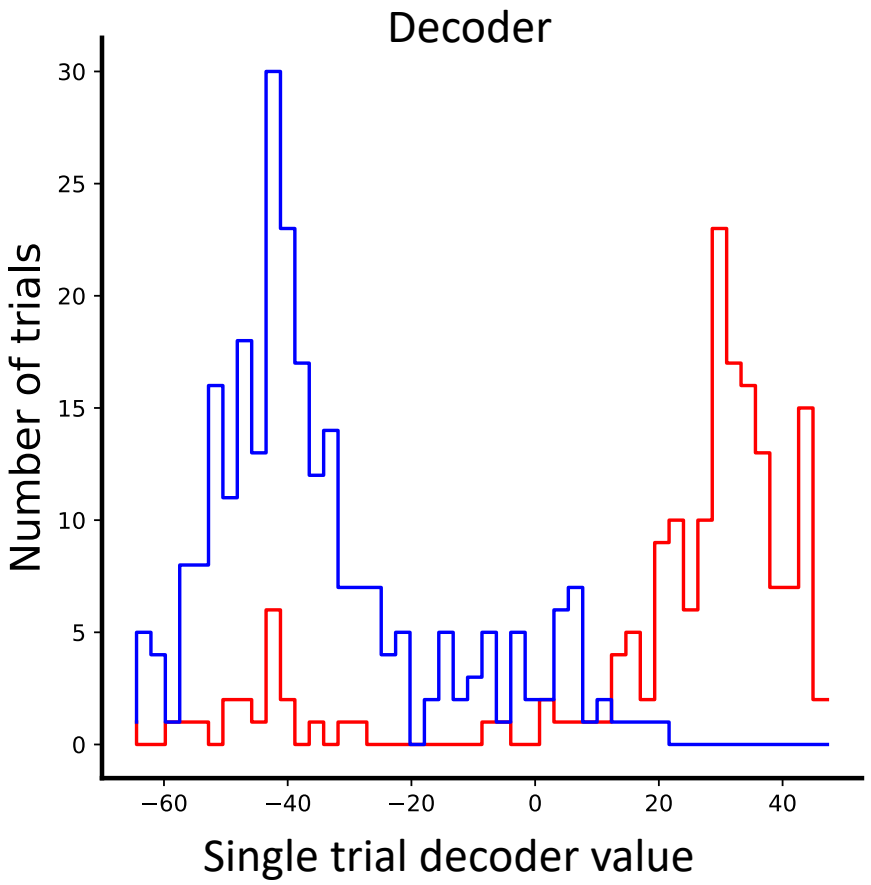


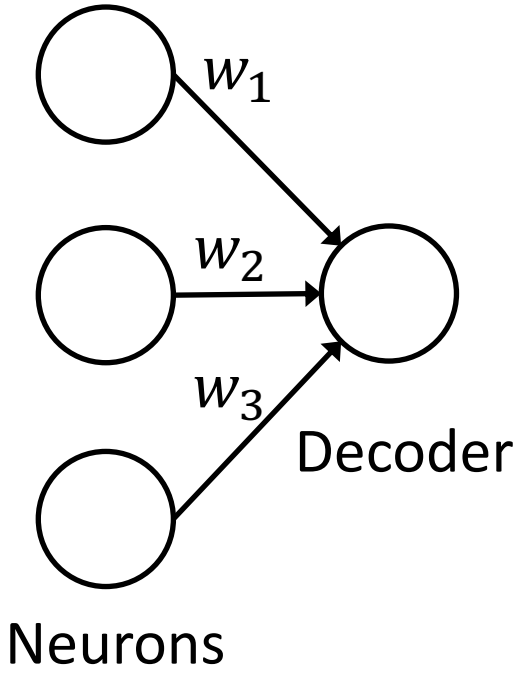


There are additional measures, like Receiver Operating Characteristic (ROC)

Population decoders

Can we combine firing rates across neurons to better tell apart which trial type it is?



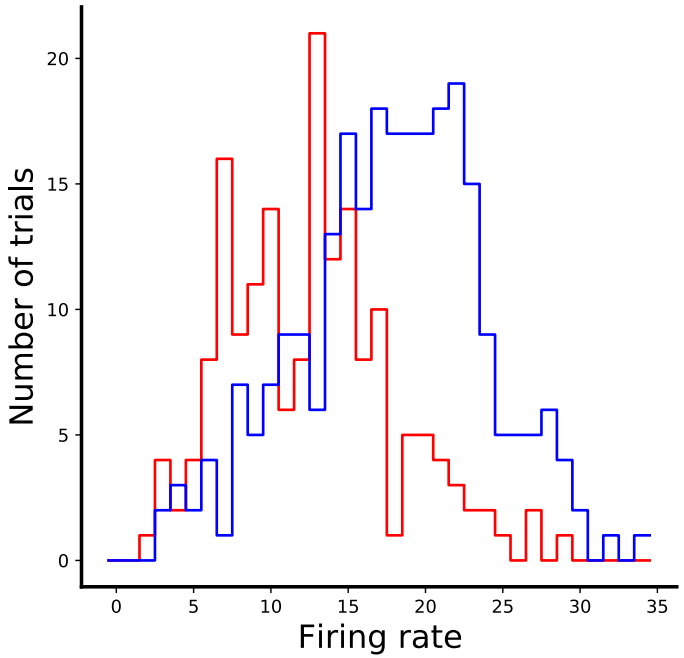


$$Single\ trial\ decoder\ value = \sum_{neurons} w_{neuron} * rate_{neuron}$$

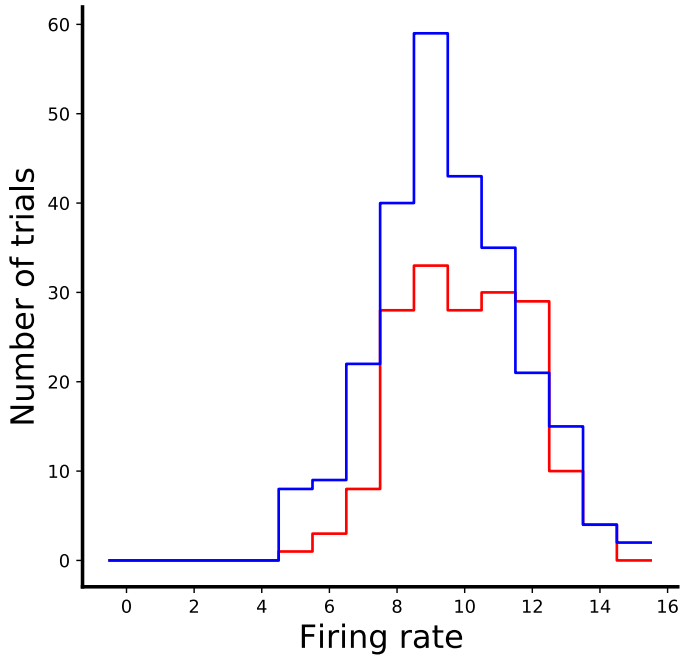
Linear Discriminant Analysis

What weights should we assign these neurons?

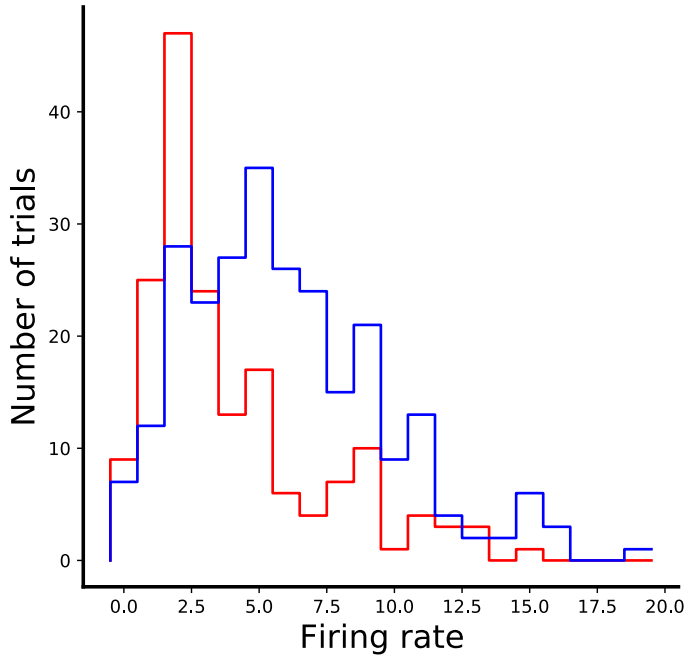
Neuron 1



Neuron 2



Neuron 3

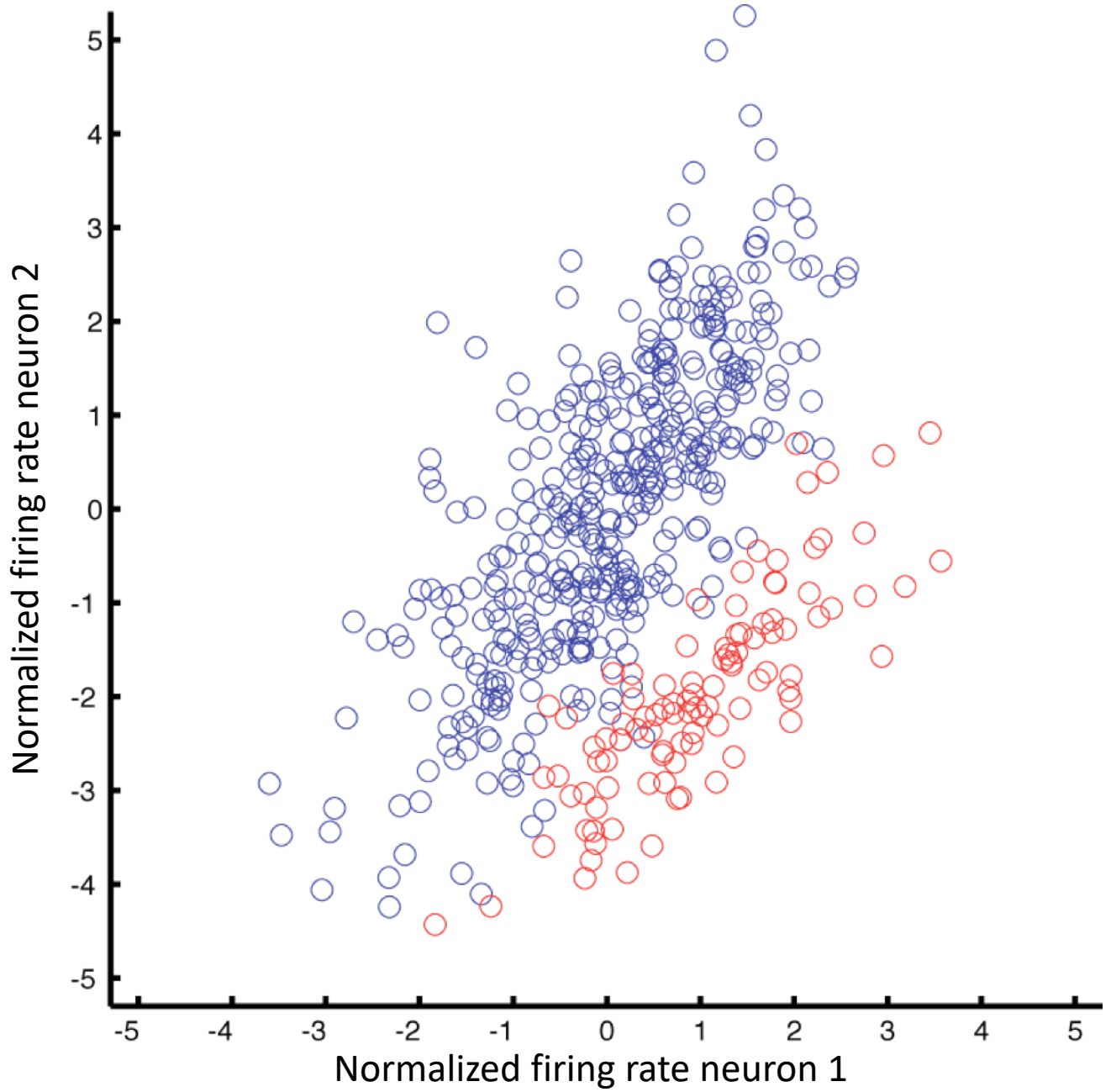


$$d' = \frac{\mu_1 - \mu_2}{\sigma_W}$$

Weights are given by the solution to an optimization problem: weights that best separate the two trial types (under some assumptions)

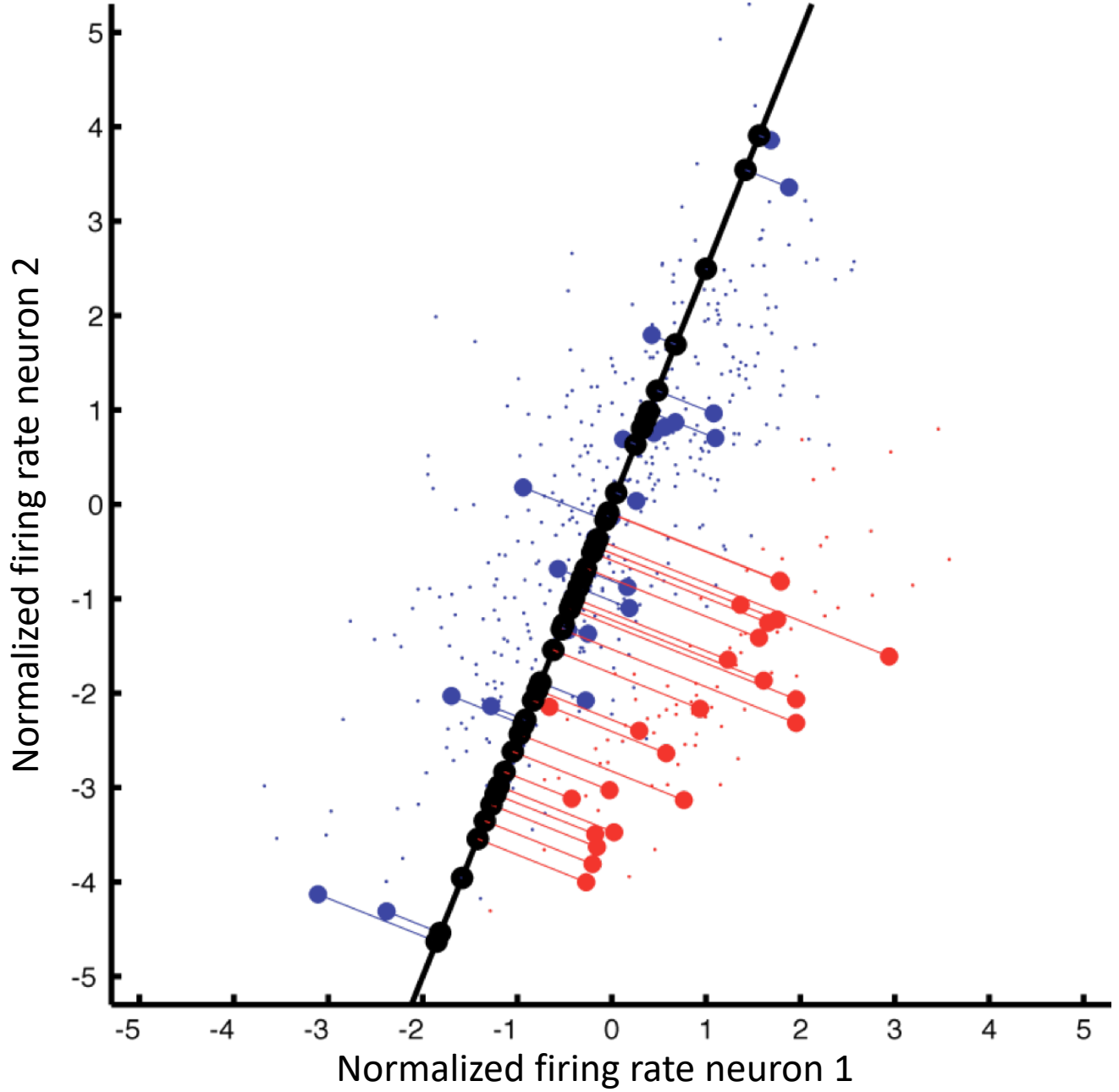
Example Linear Discriminant Analysis on synthetic data

We want a weighted sum that maximizes separability between the two classes once projected down to the decoder

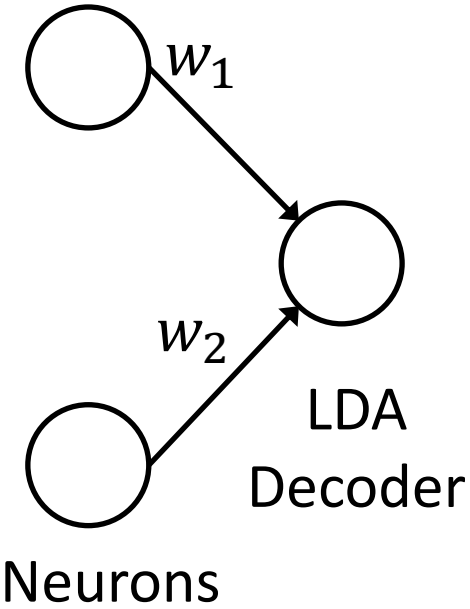


Example Linear Discriminant Analysis on synthetic data

We want a weighted sum that maximizes separability between the two classes once projected down to the decoder

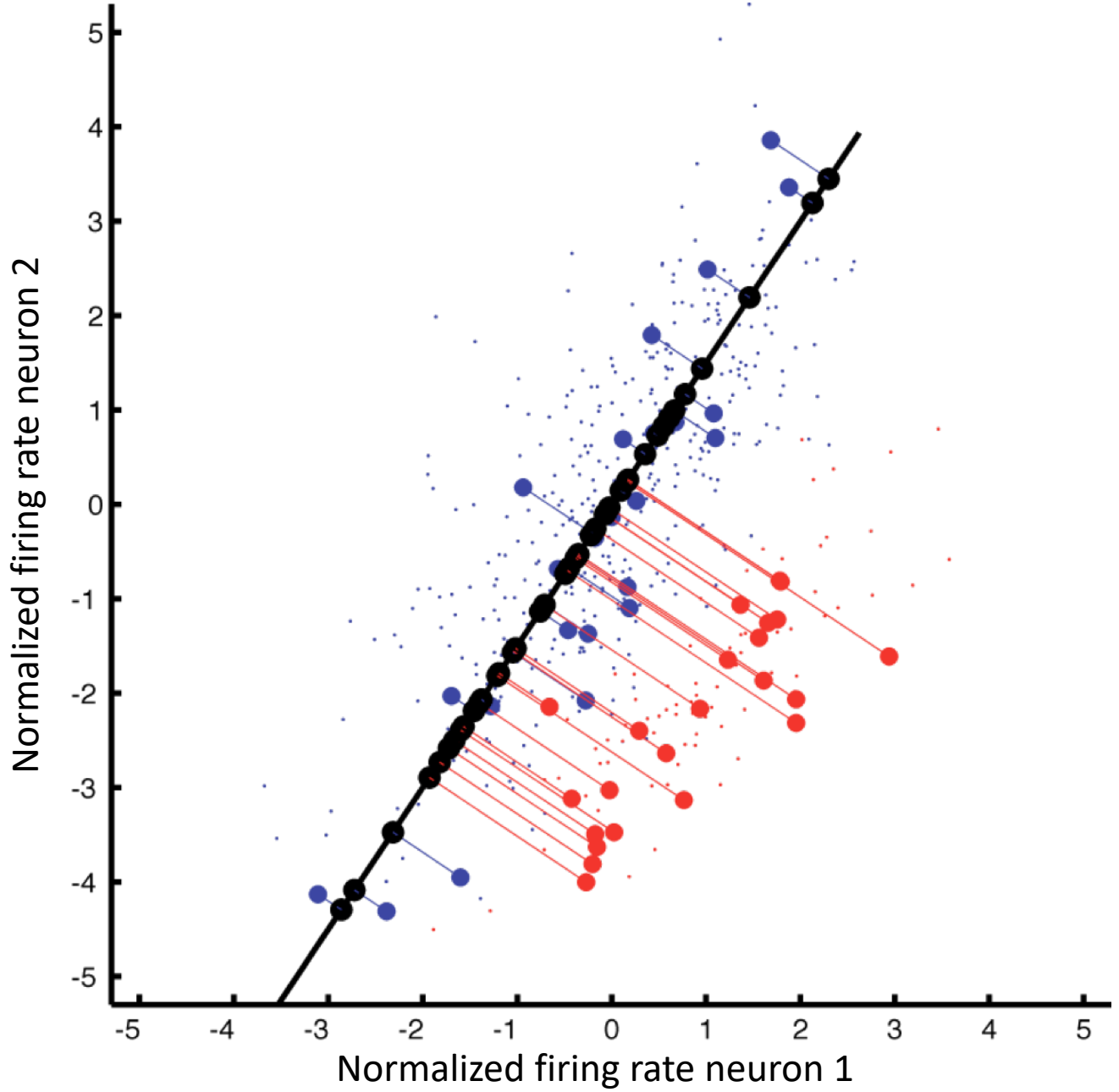


$$w_1 = 0.2$$
$$w_2 = 0.8$$

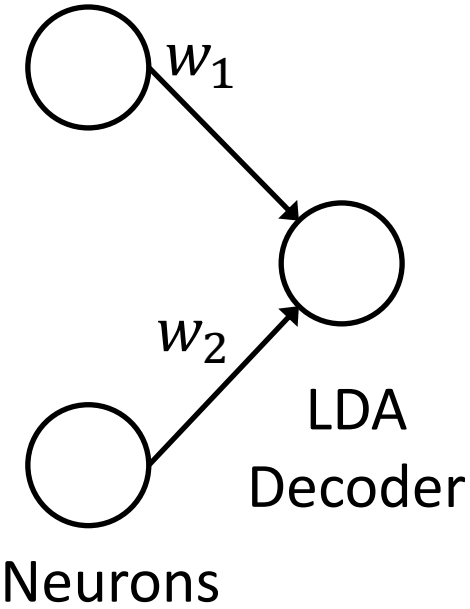


Example Linear Discriminant Analysis on synthetic data

We want a weighted sum that maximizes separability between the two classes once projected down to the decoder

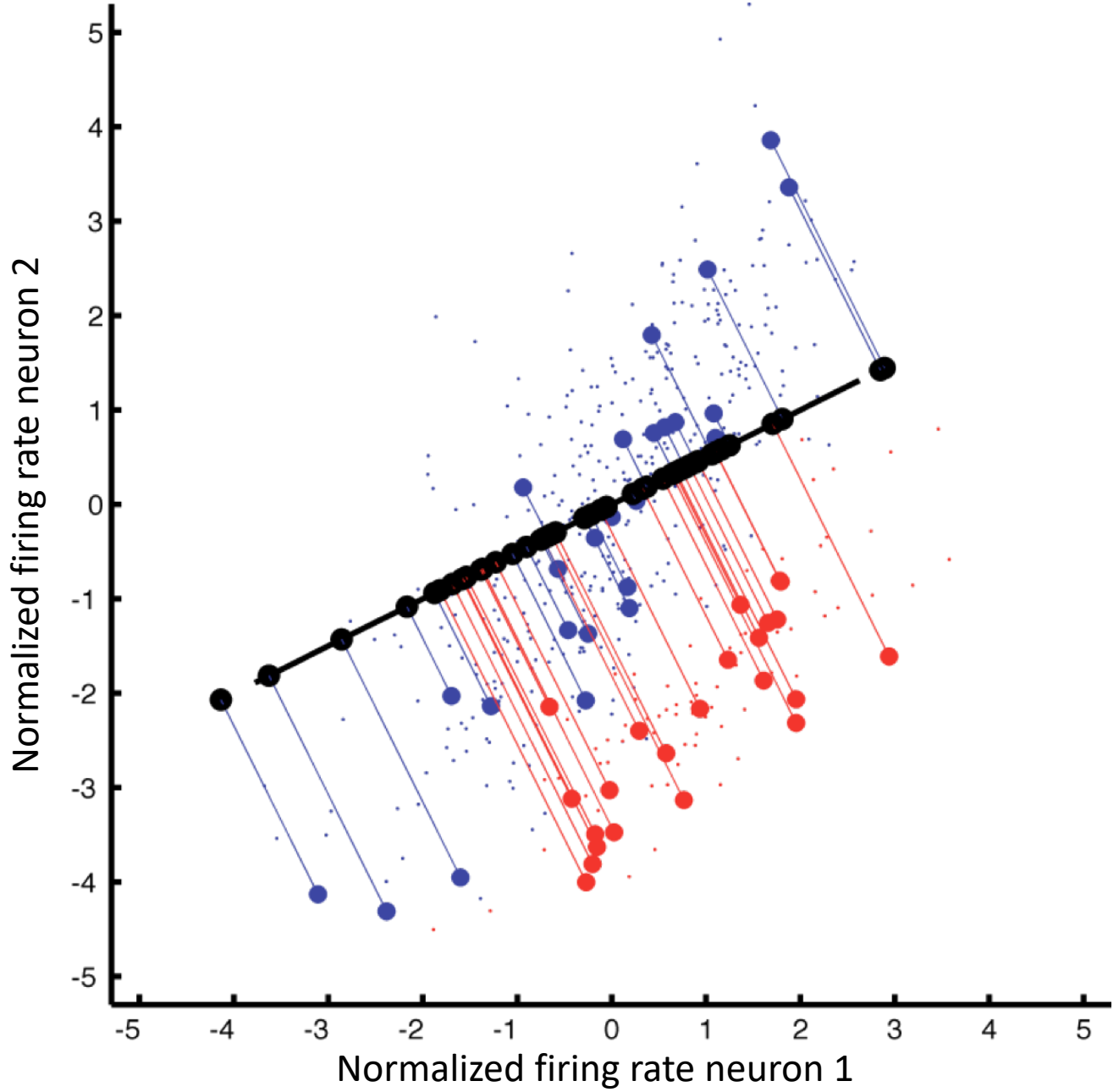


$$w_1 = 0.4$$
$$w_2 = 0.6$$

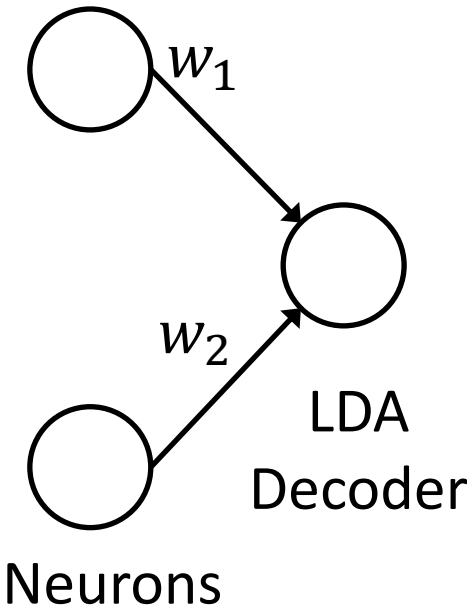


Example Linear Discriminant Analysis on synthetic data

We want a weighted sum that maximizes separability between the two classes once projected down to the decoder

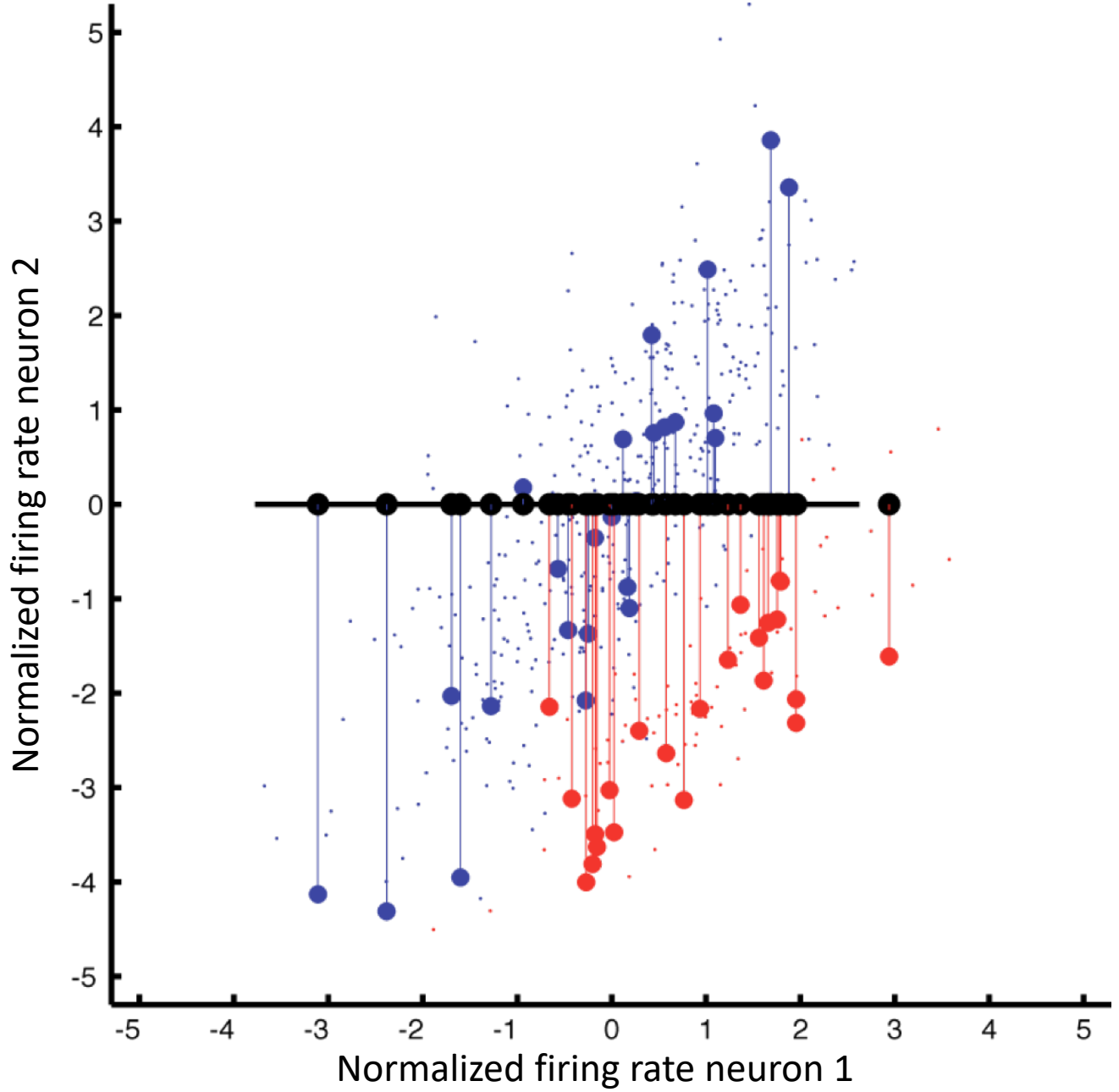


$w_1 = 0.6$
$w_2 = 0.4$

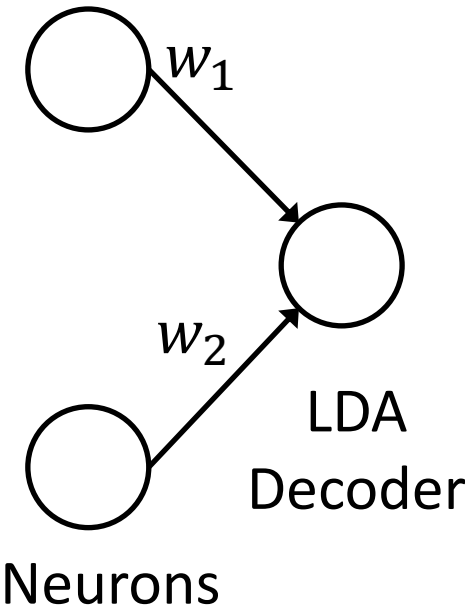


Example Linear Discriminant Analysis on synthetic data

We want a weighted sum that maximizes separability between the two classes once projected down to the decoder

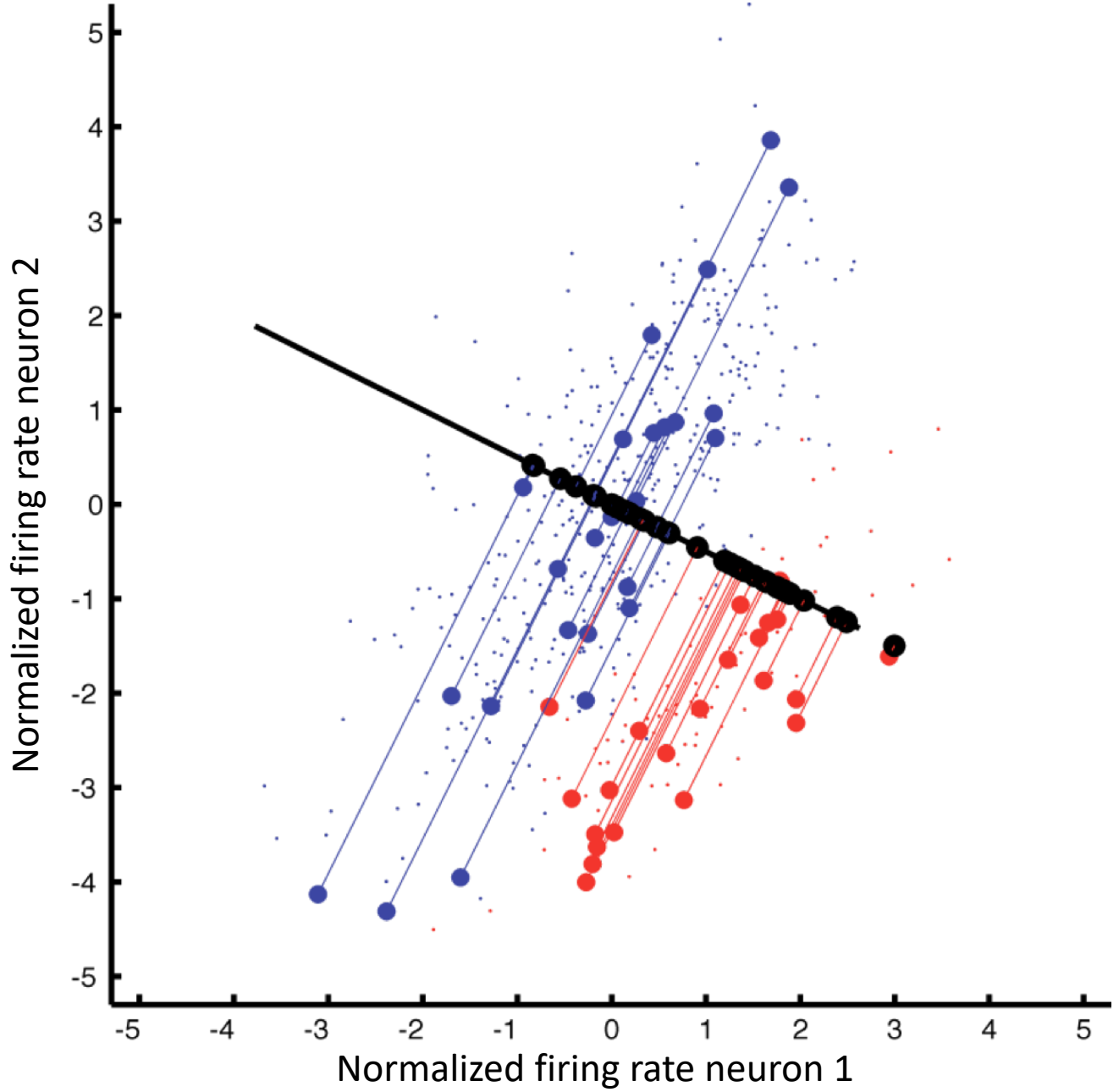


$$w_1 = 1$$
$$w_2 = 0$$

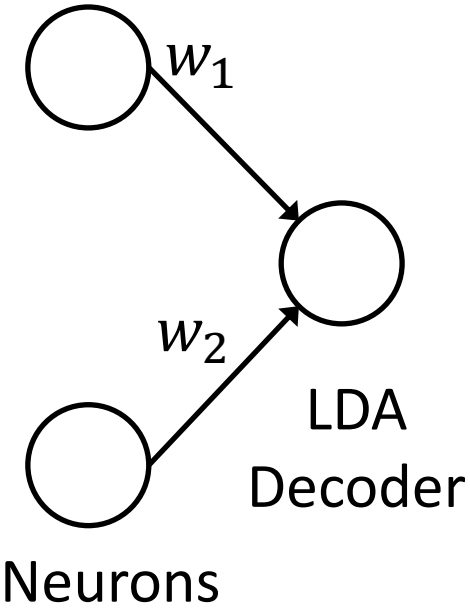


Example Linear Discriminant Analysis on synthetic data

We want a weighted sum that maximizes separability between the two classes once projected down to the decoder



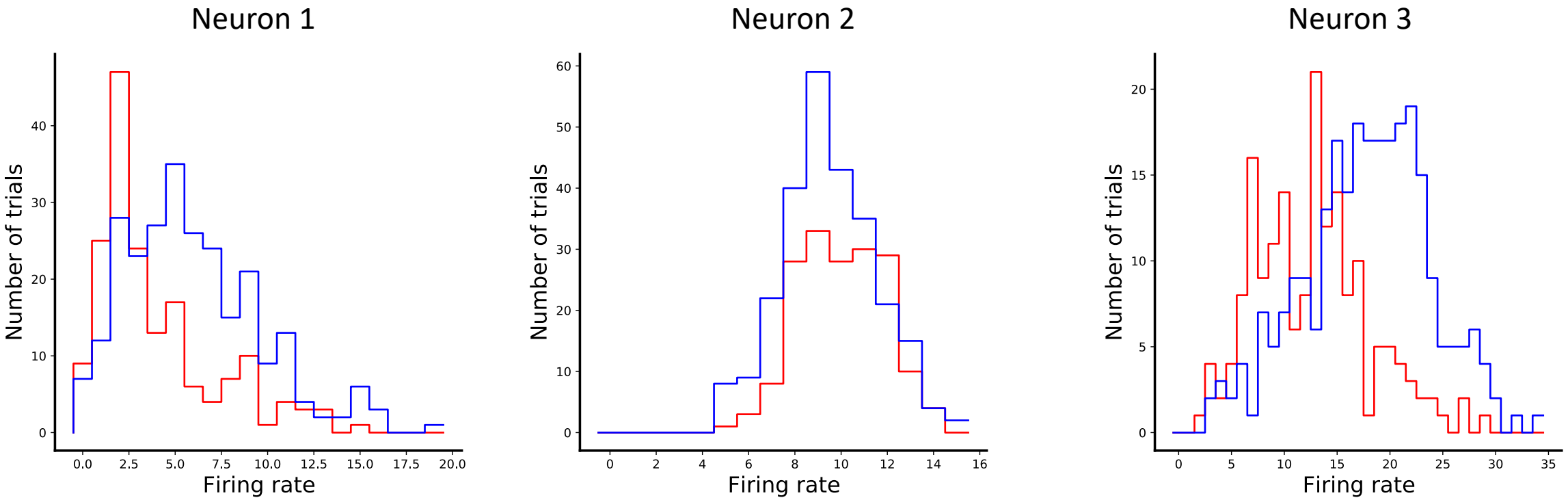
$$w_1 = 0.8$$
$$w_2 = -0.2$$



Skip derivation

Linear Discriminant Analysis

Weighted linear sum. What weights should we assign these neurons?

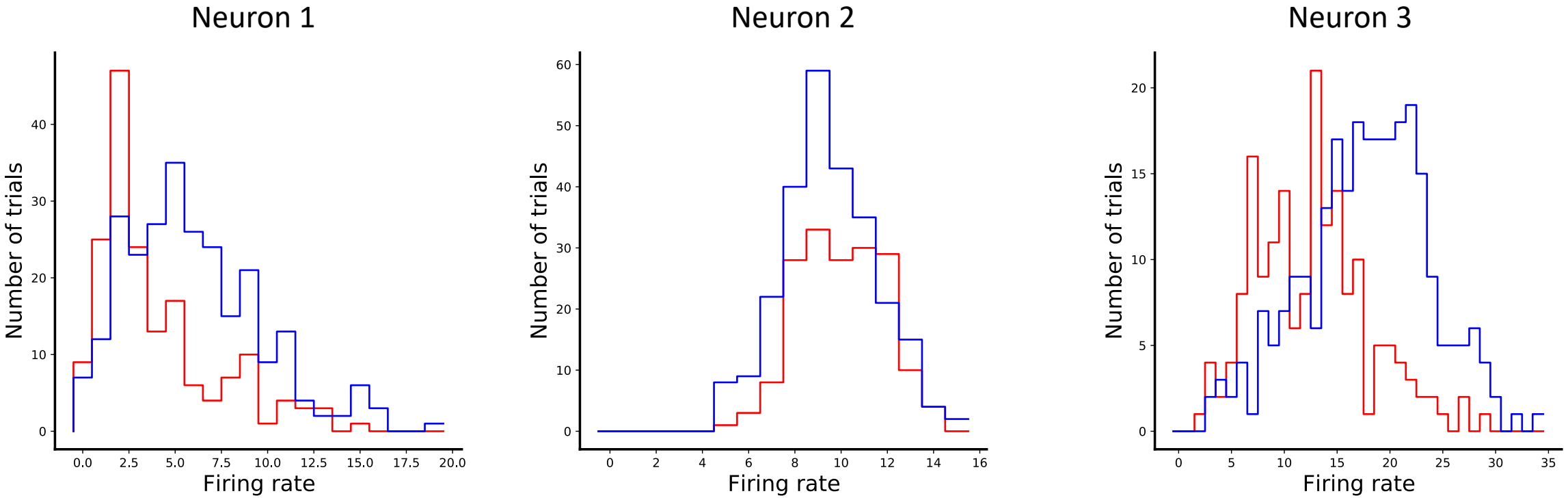


Solution isn't exactly taking the d-prime for each neuron

$$d' = \frac{\mu_1 - \mu_2}{\sigma_W}$$

Linear Discriminant Analysis

Weighted linear sum. What weights should we assign these neurons?

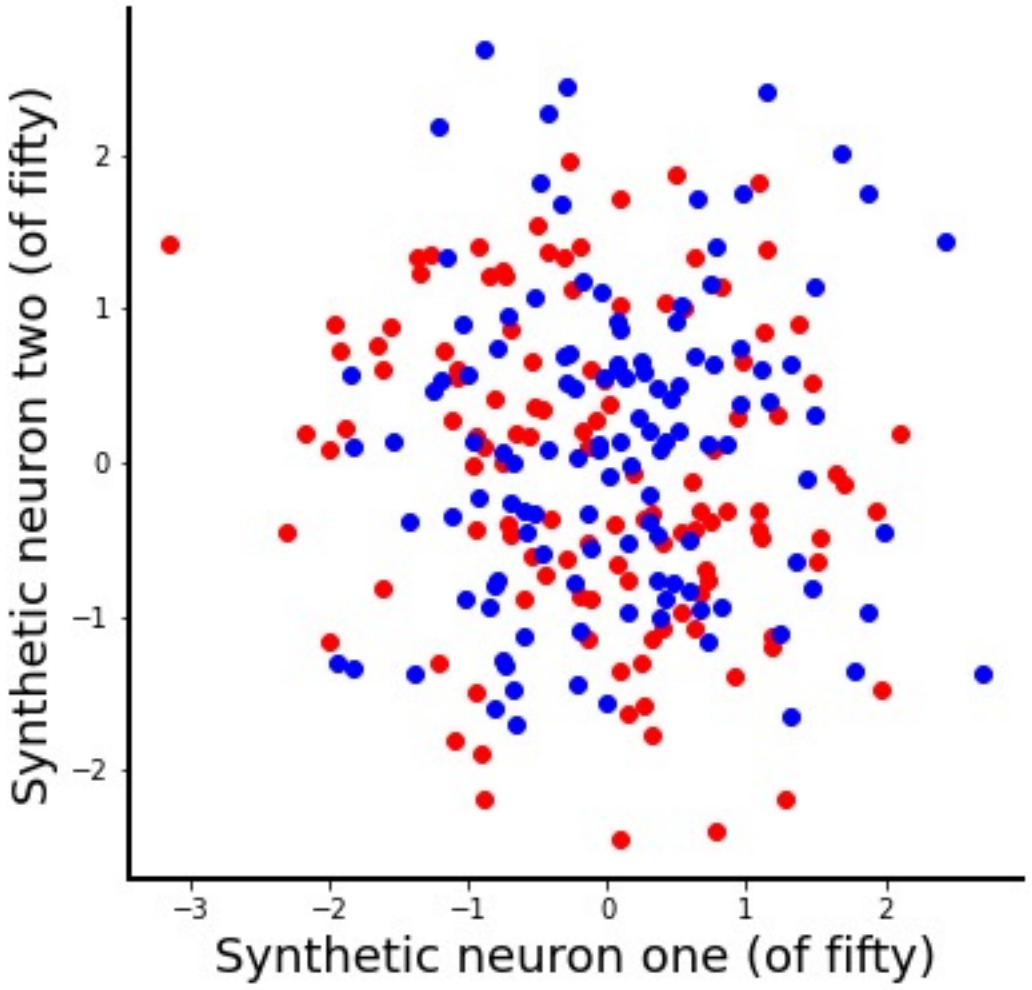


Solution to optimization given by top eigenvectors of matrix: $S_W^{-1} S_B$

$$d' = \frac{\mu_1 - \mu_2}{\sigma_W}$$

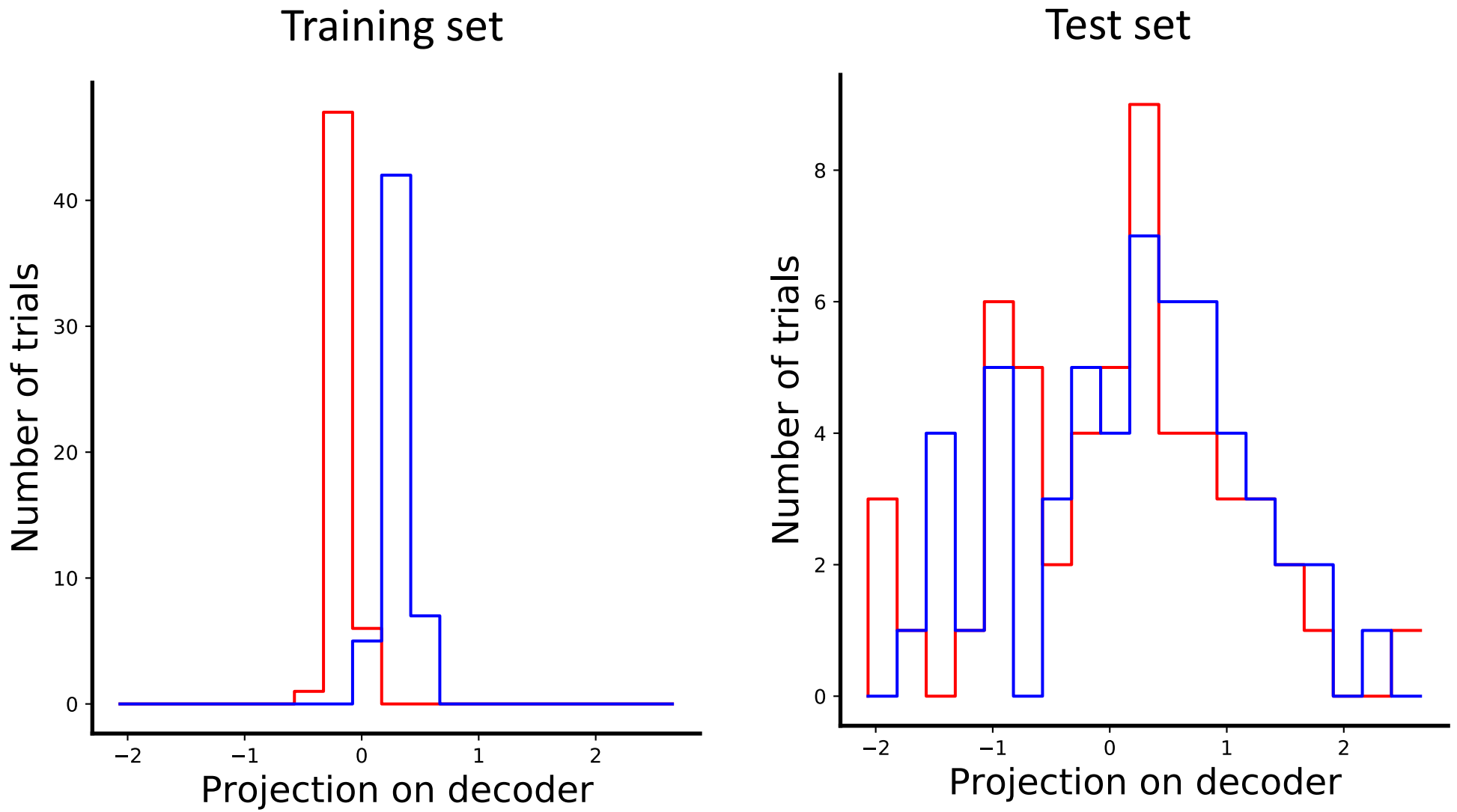
S_W is the within group variance matrix, S_B is the between group variance matrix

Overfitting and separation of training and test datasets



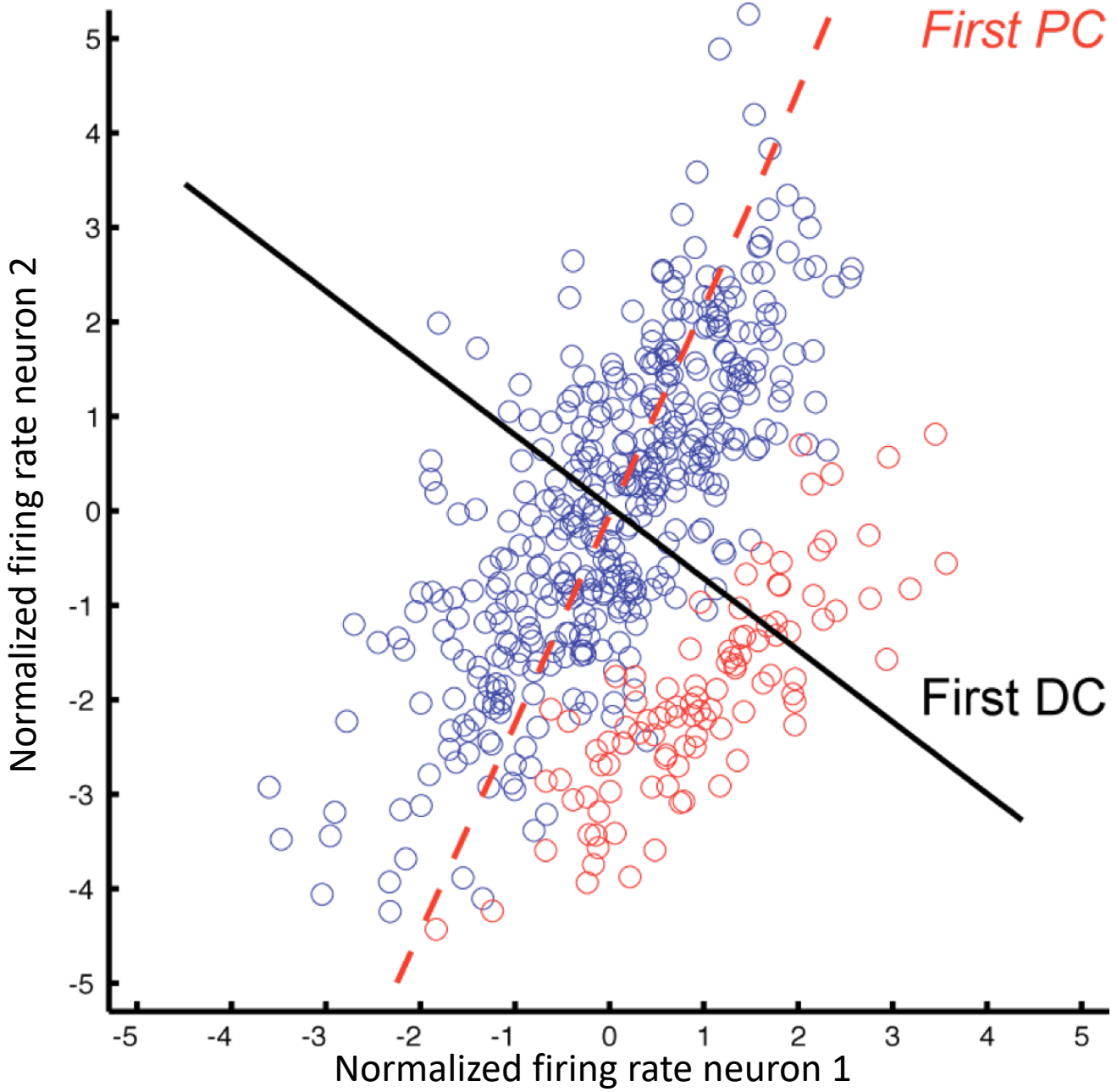
Samples from synthetic dataset generated randomly with the same mean

Overfitting and separation of training and test datasets

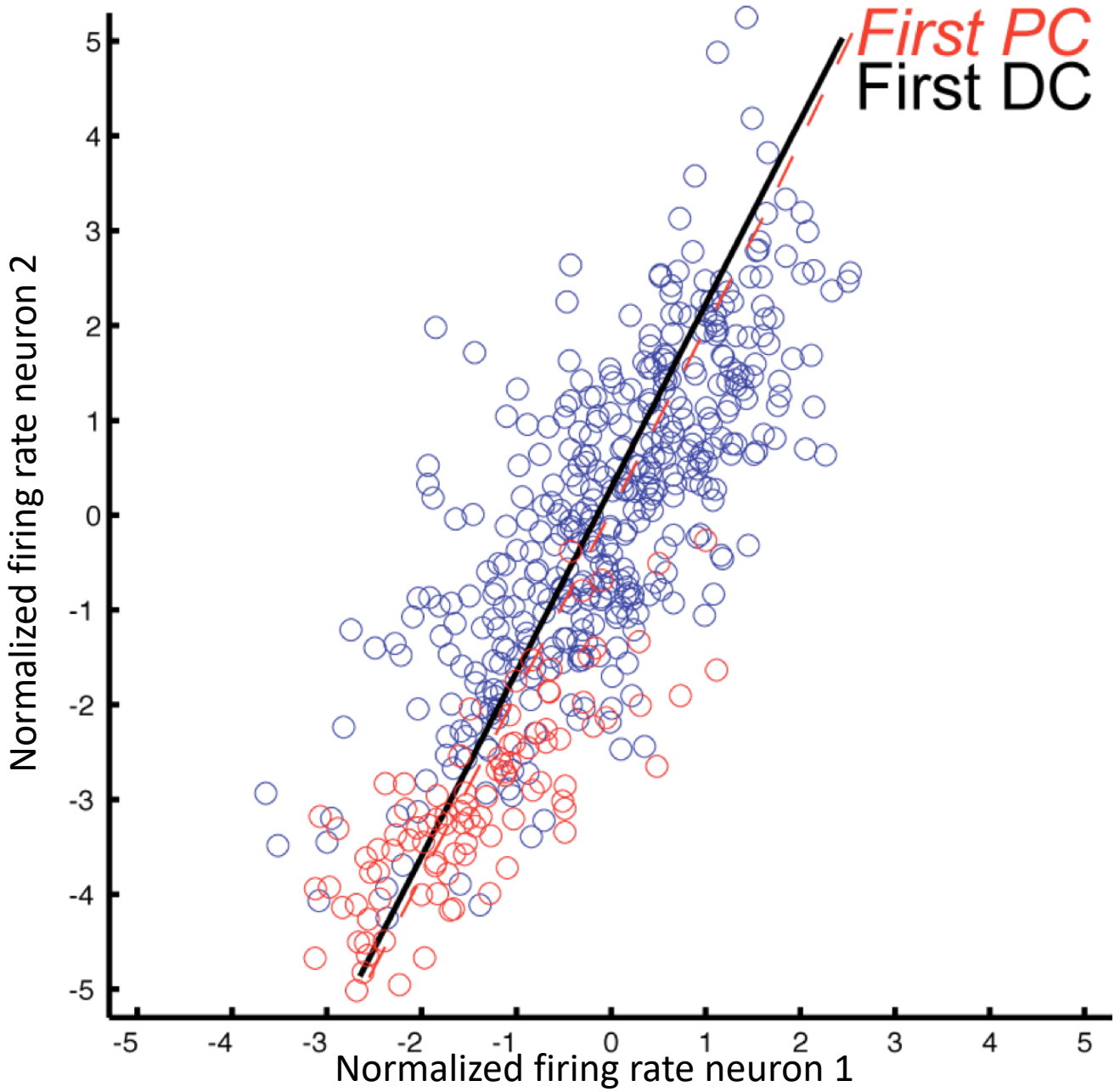


Over fitting!

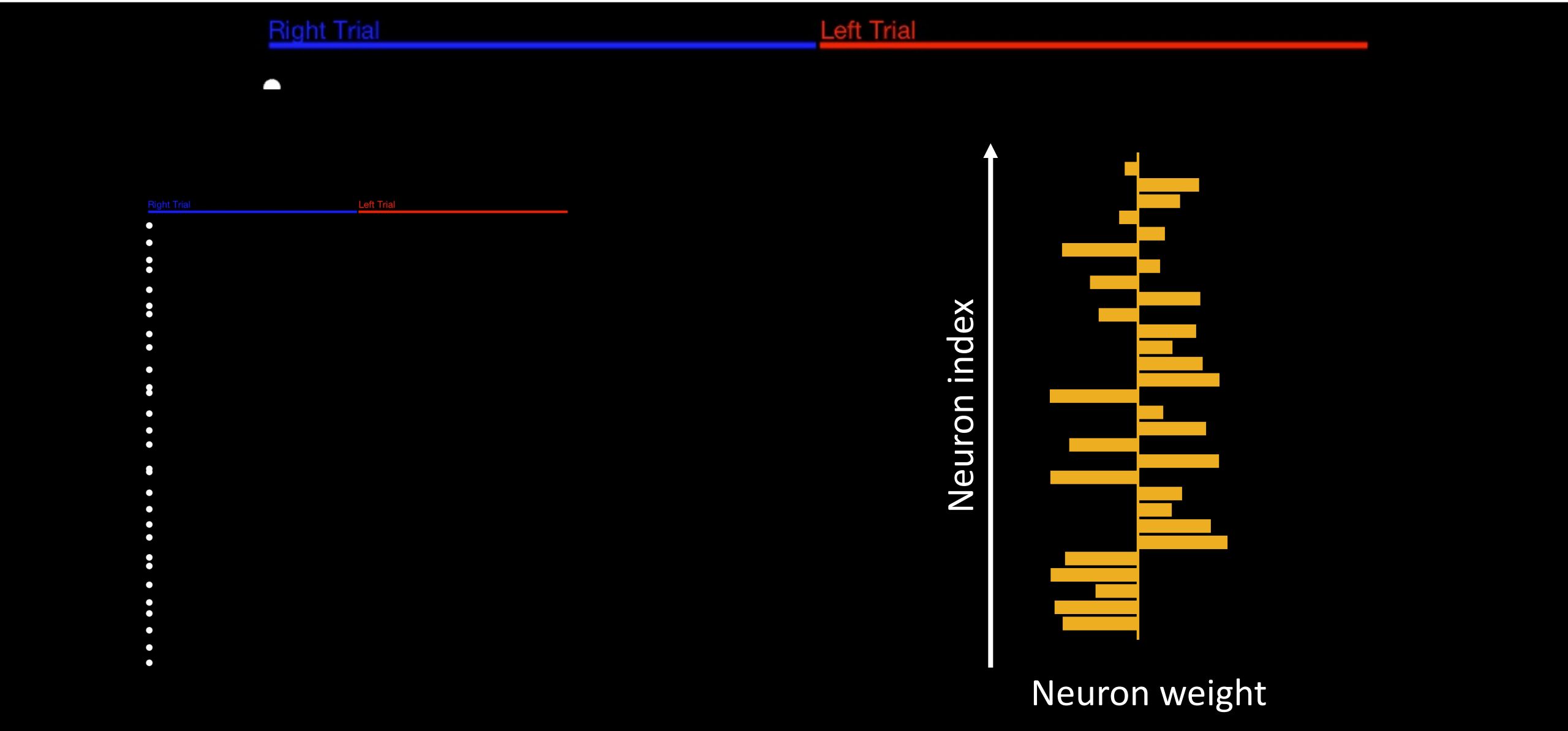
LDA


First PC
First DC
Normalized firing rate neuron 2
Normalized firing rate neuron 1
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

LDA

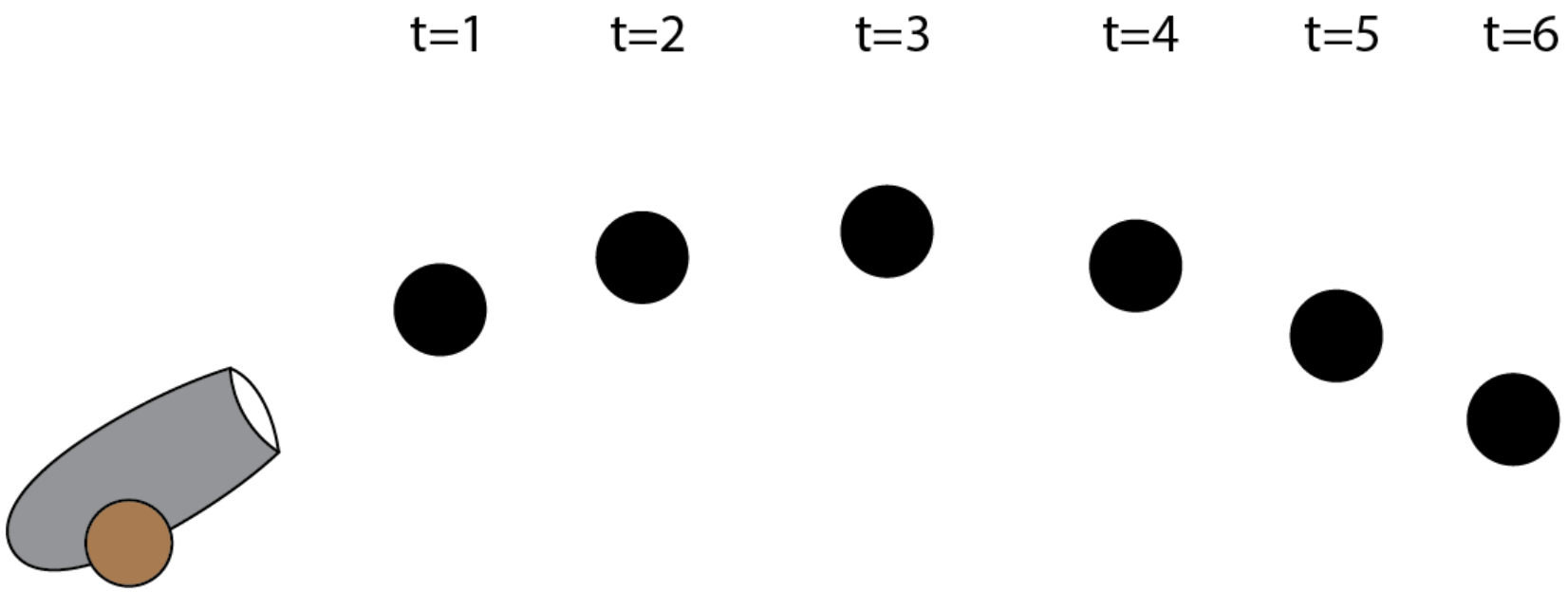

First PC
First DC
Normalized firing rate neuron 2
Normalized firing rate neuron 1
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

Decoder is static, but one can analyze projection over time

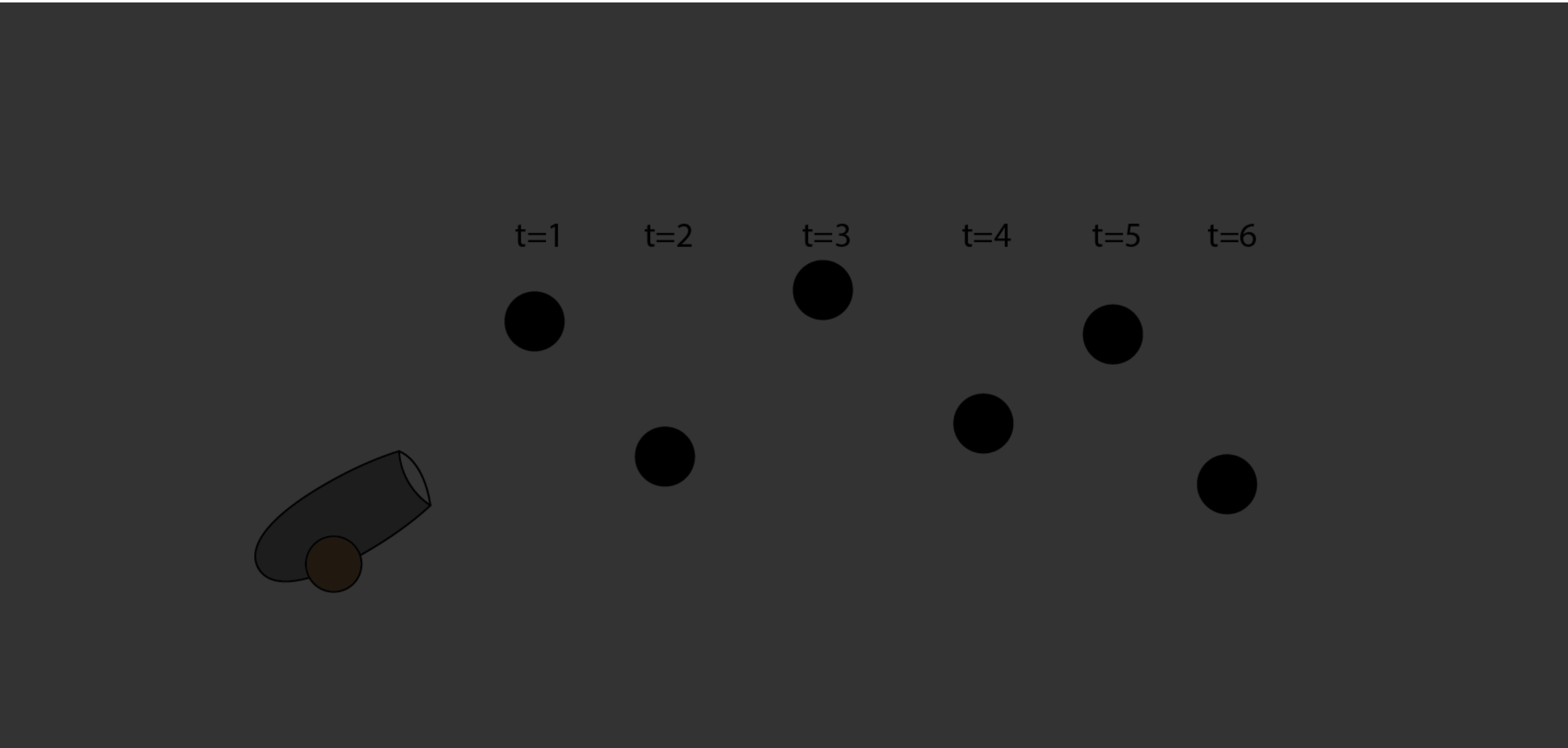


Switching gears: analyzing structure in population recordings

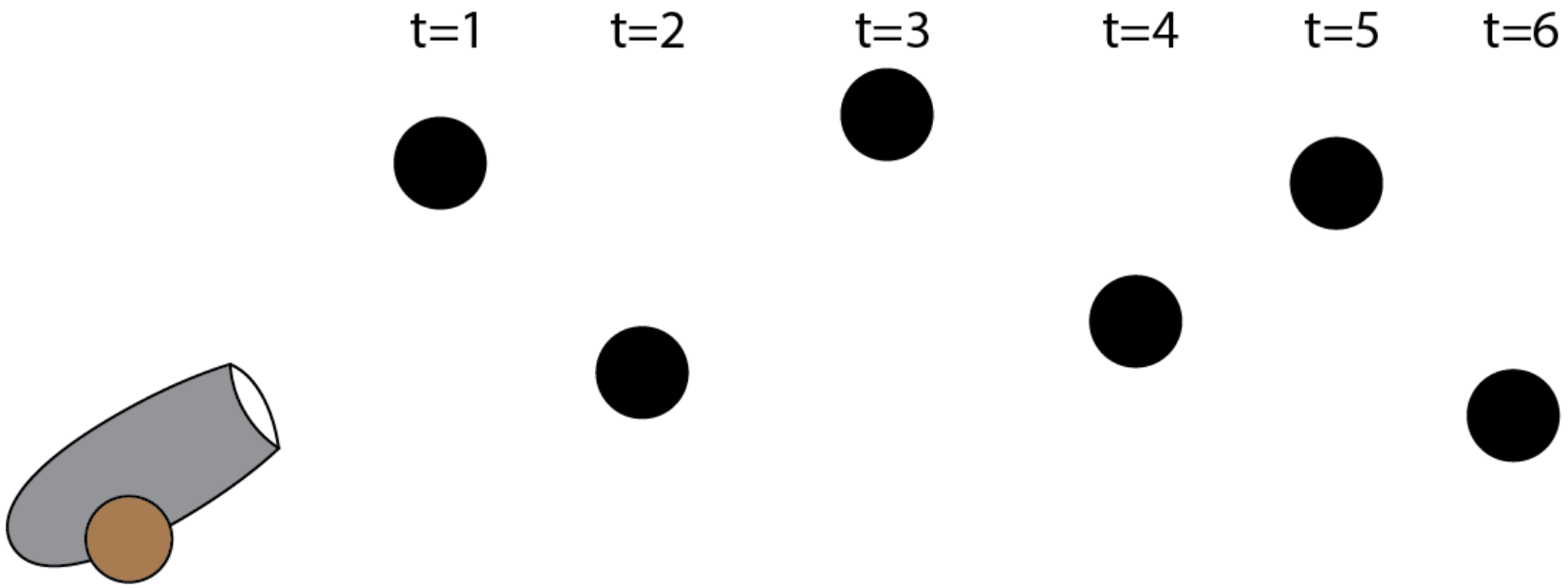
Modeling structure in population recordings: why do it?



Modeling structure in population recordings: why do it?

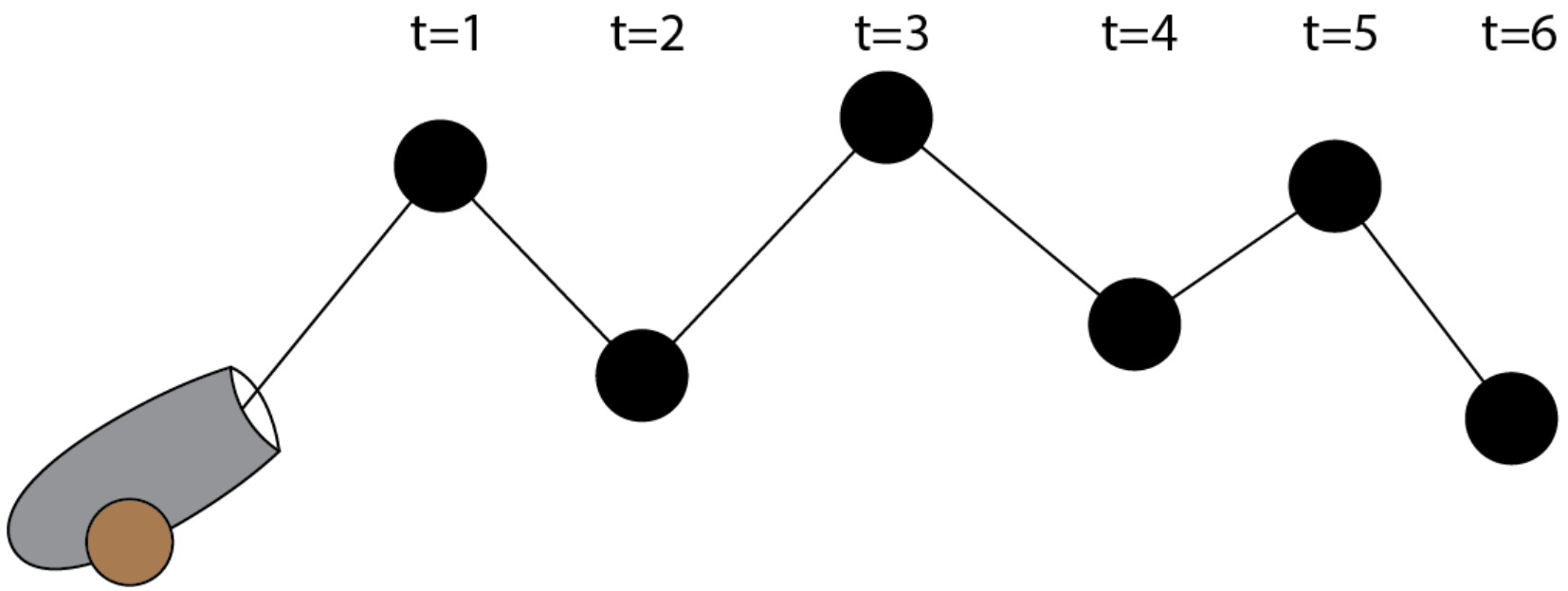


Modeling structure in population recordings: why do it?



Modeling structure in population recordings: why do it?

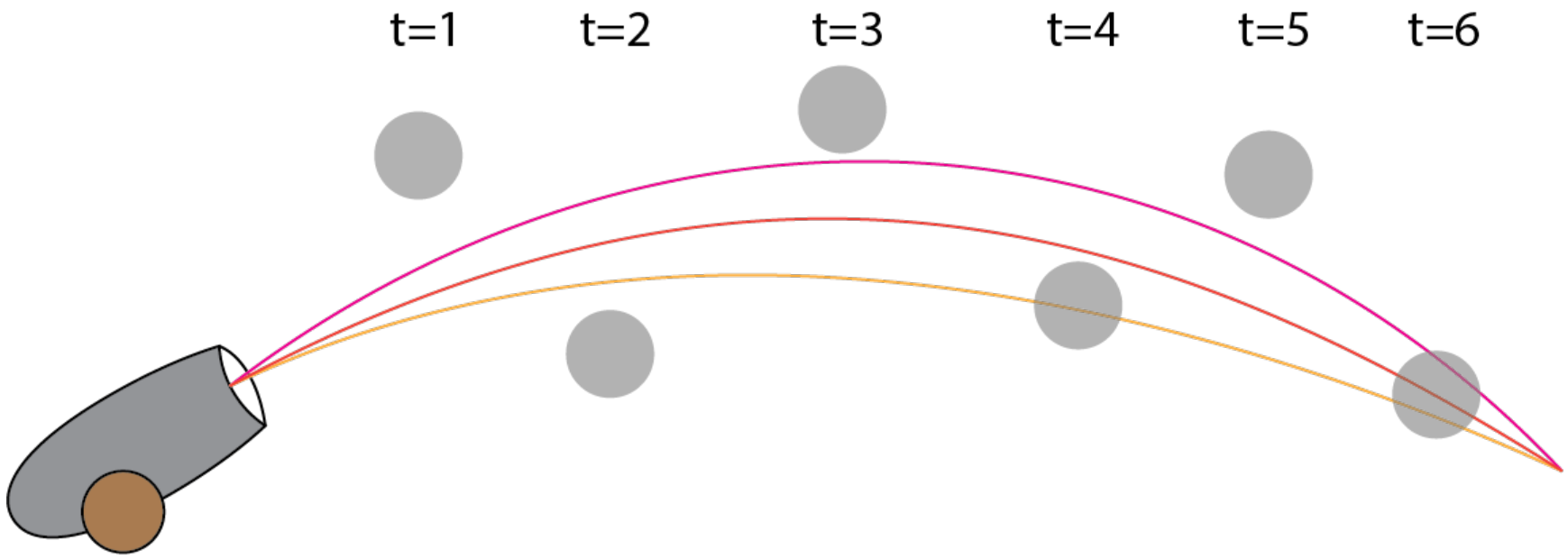
Trajectory very unlikely given our understanding of the system!



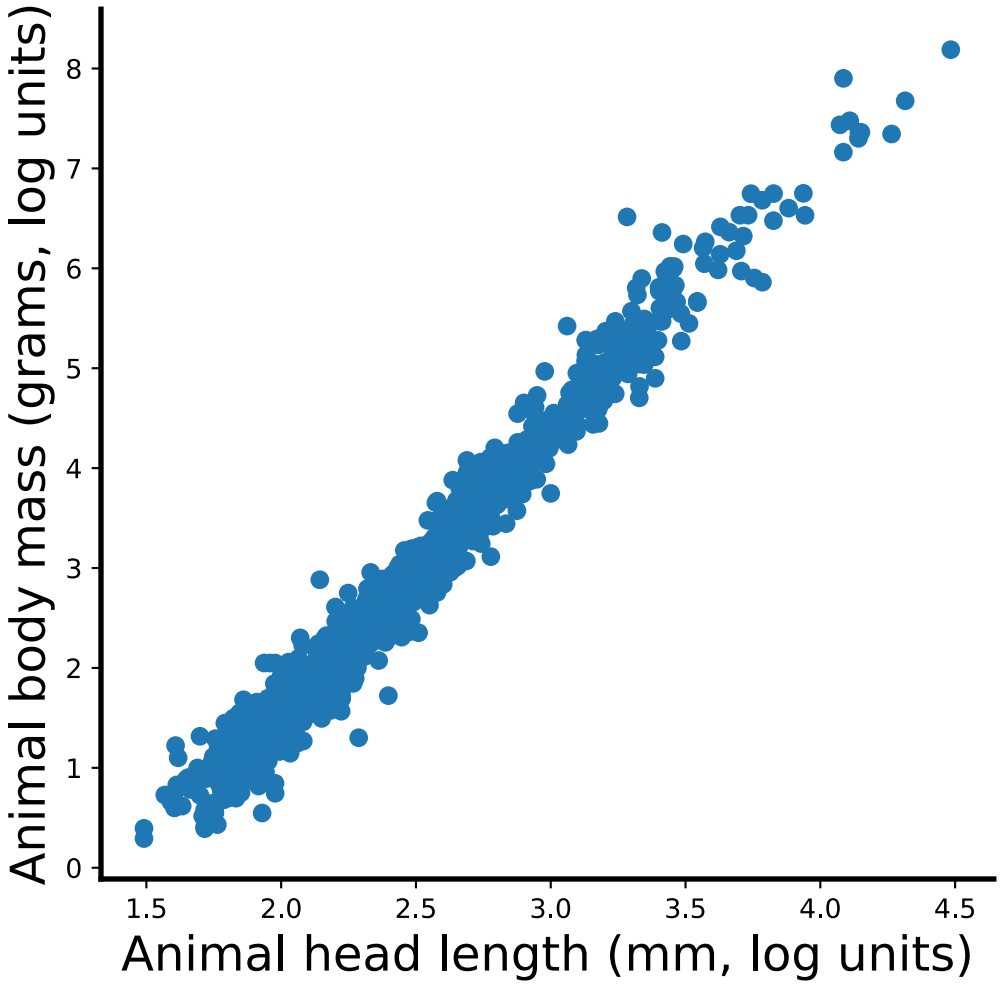
Modeling structure in population recordings: why do it?

Trajectory very unlikely given our understanding of the system!

Assume measurements were noisy and use them to pick trajectory consistent with understanding of system and the measurements

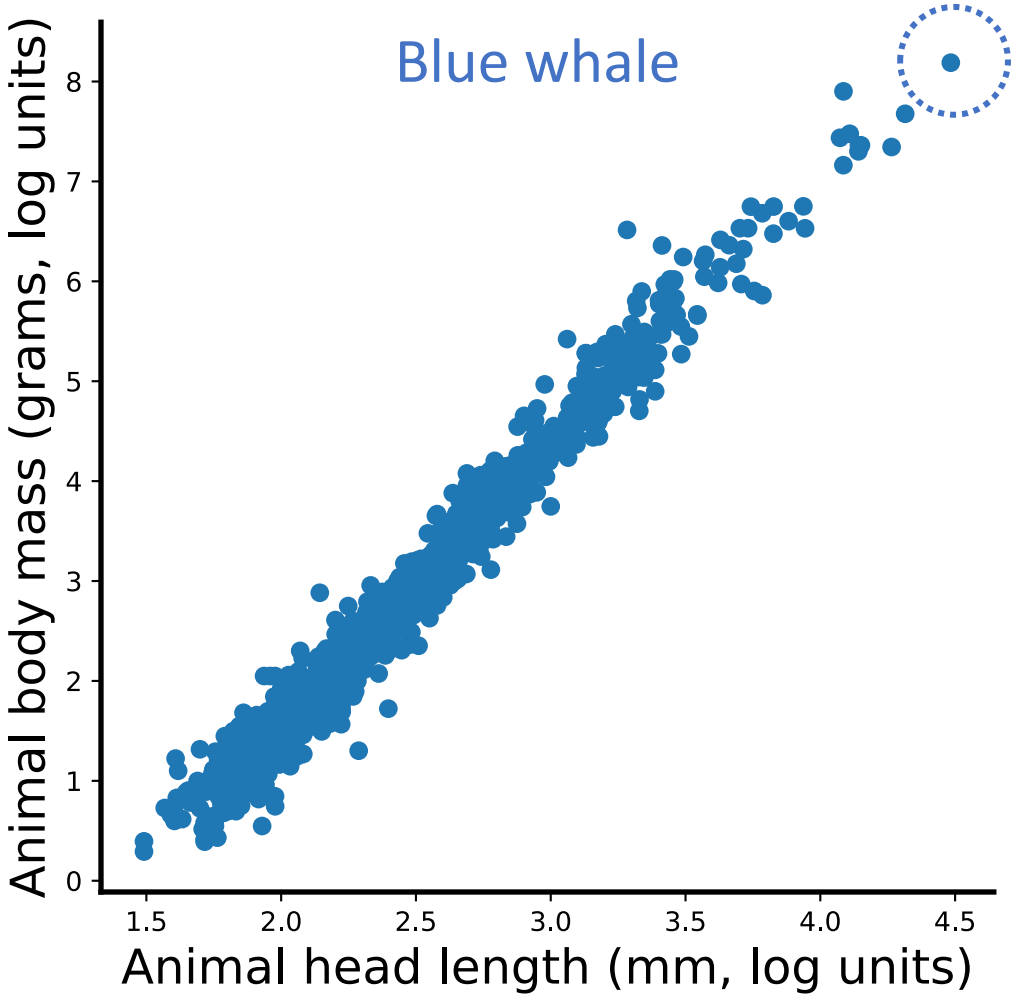


Example: animal properties



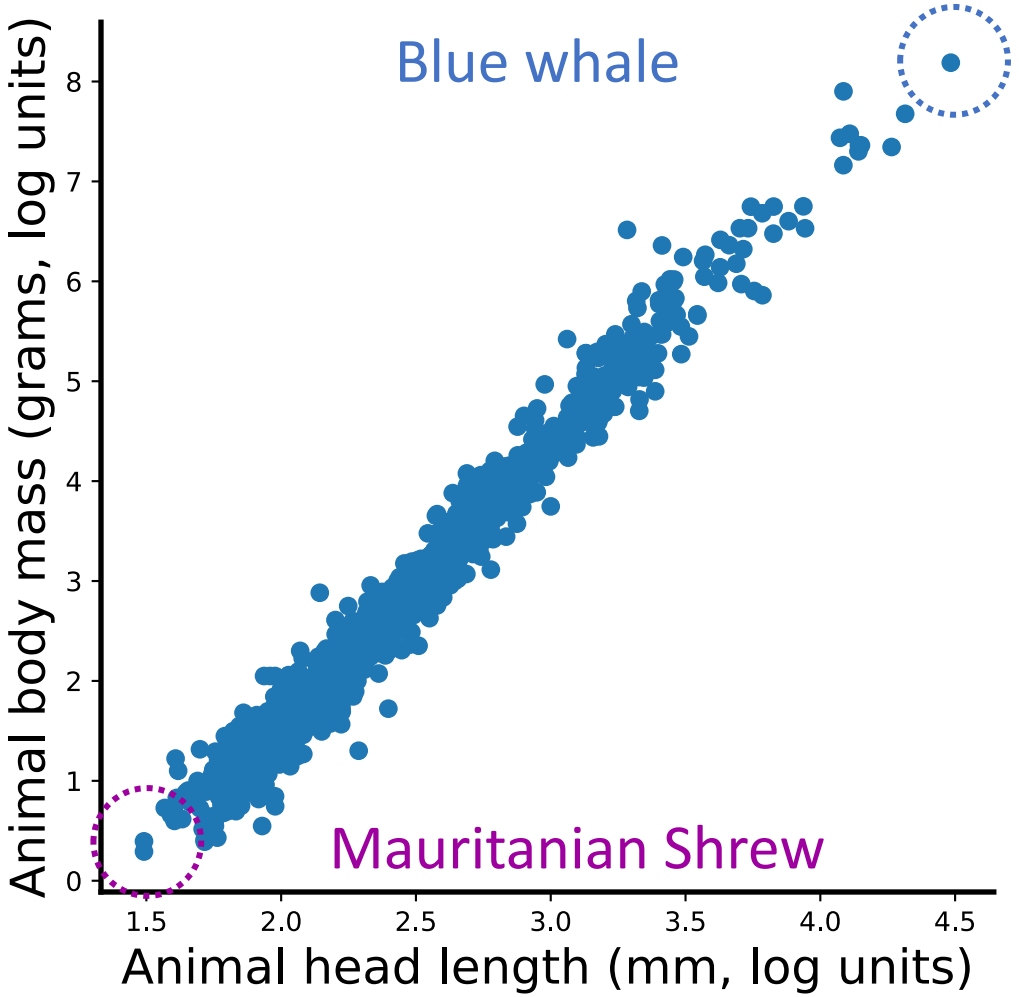
PanTHERIA dataset

Example: animal properties



PanTHERIA dataset

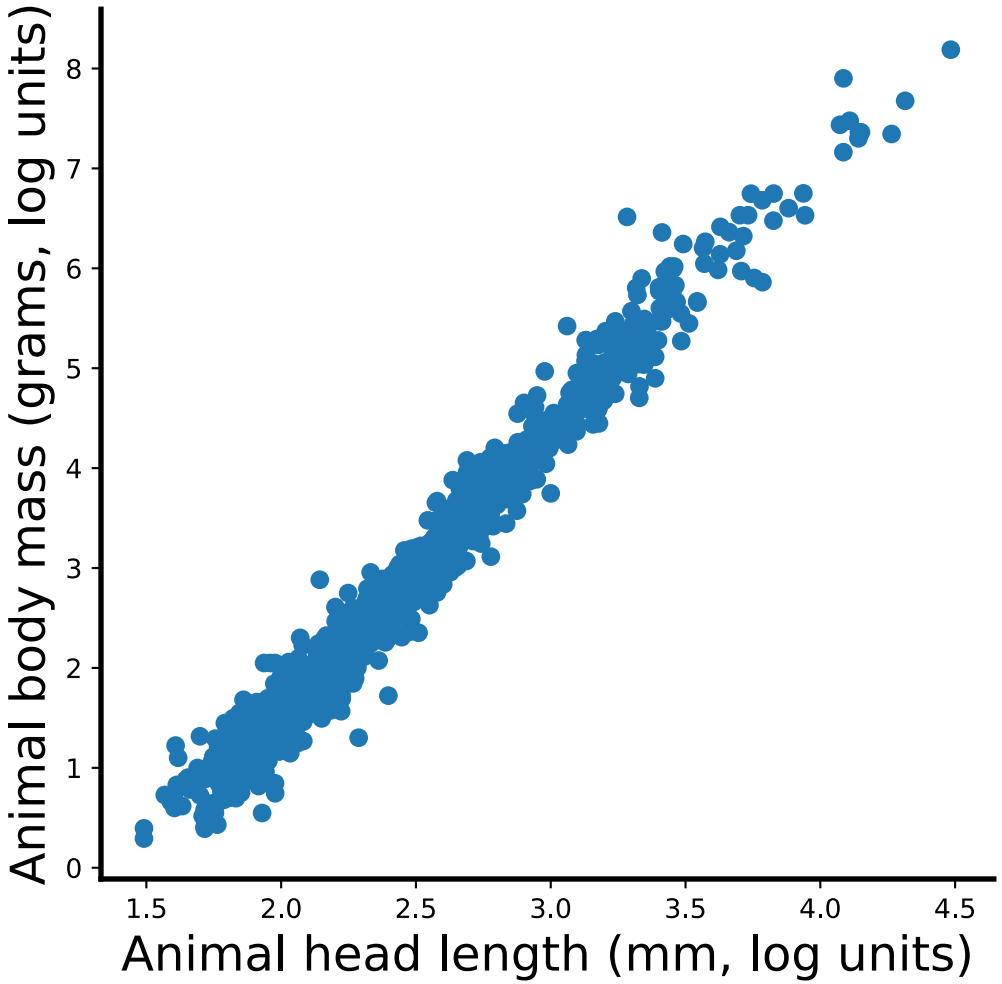
Example: animal properties



PanTHERIA dataset

Factor analysis

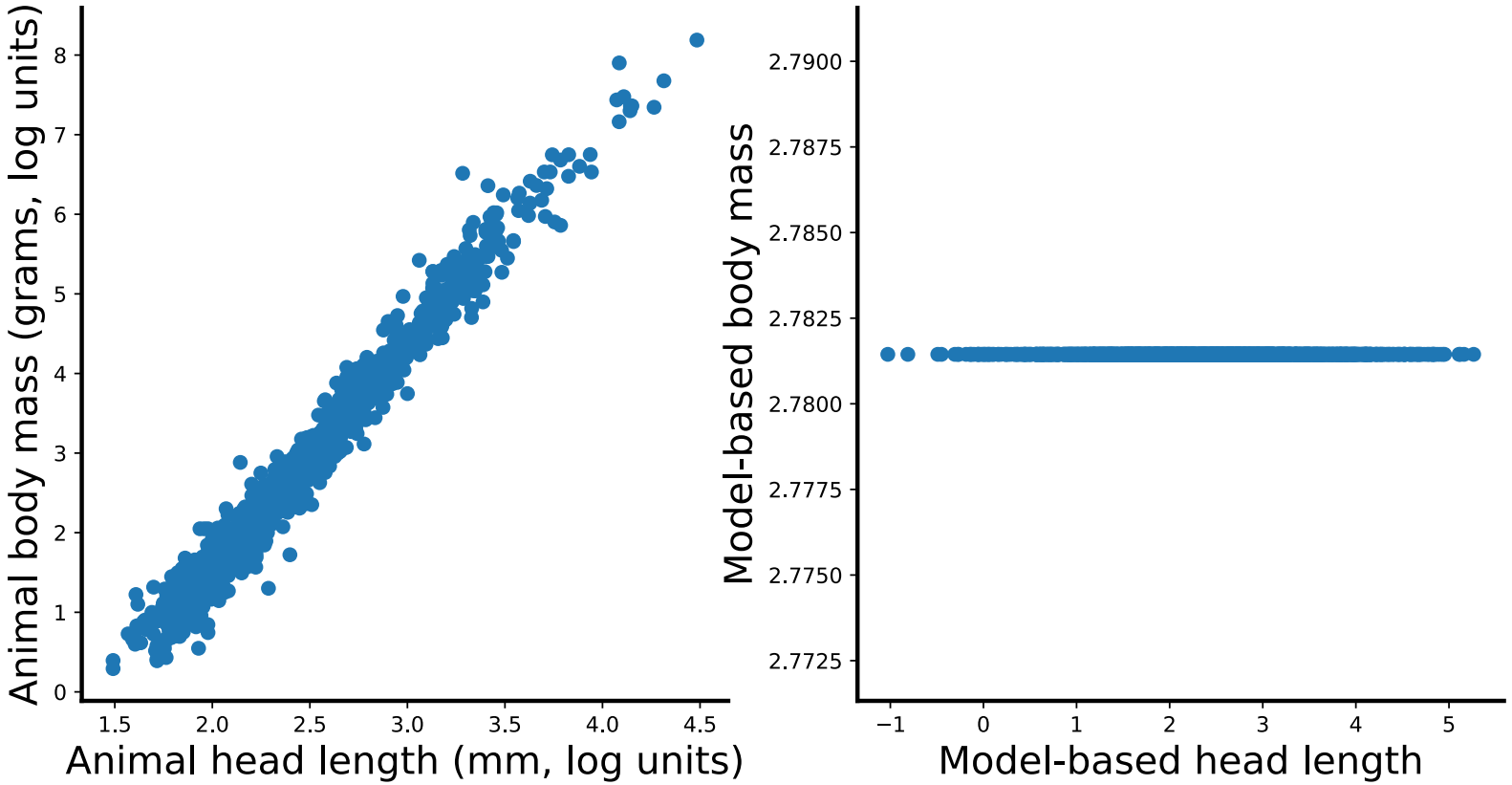
These two properties can be thought of as connected through a common factor which we could call “Size”



Latent variable models as data recipes

Data recipe:

Generate random 1D Gaussian

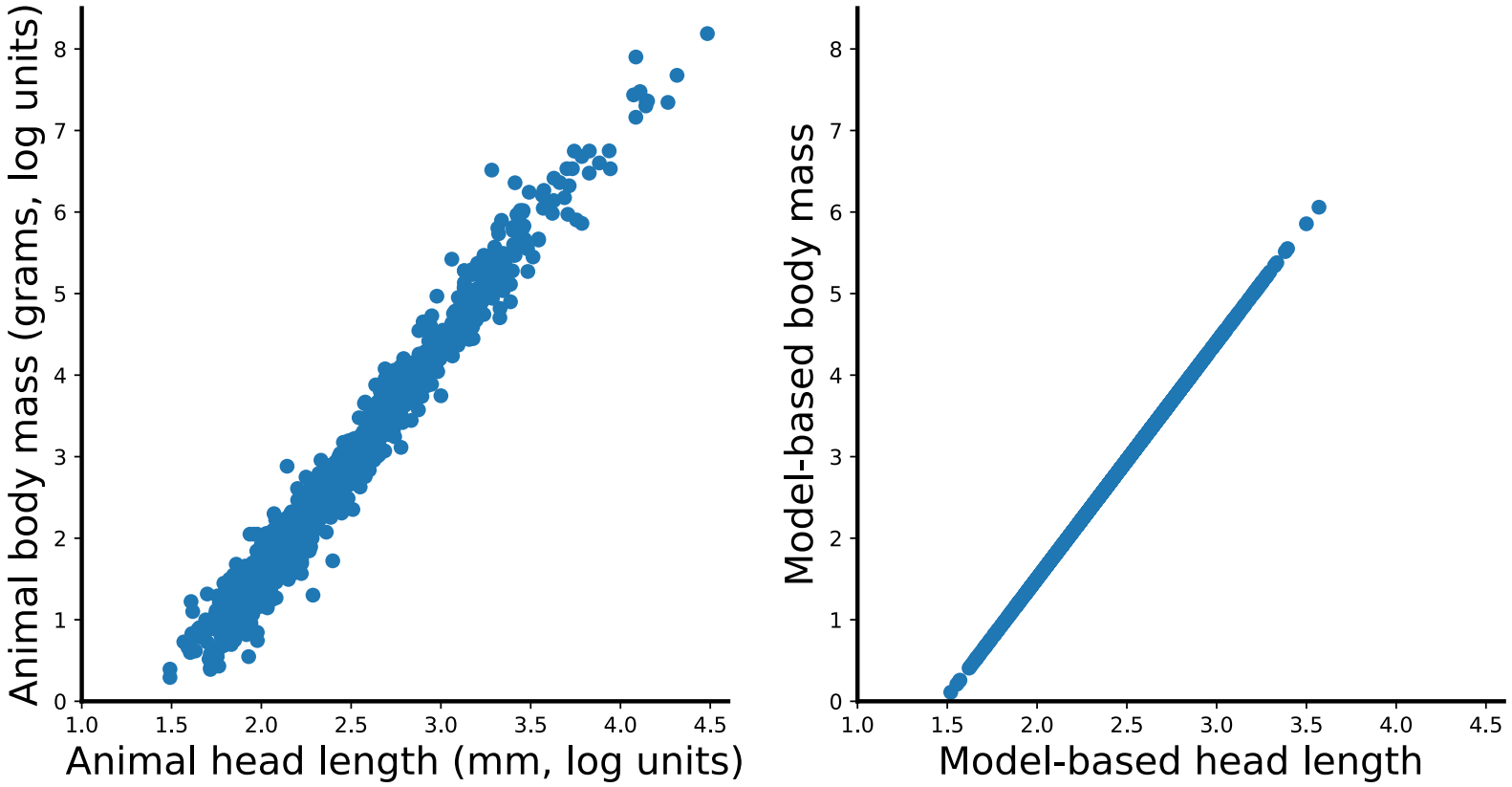


Latent variable models as data recipes

Date recipe:

Start with random 1D Gaussian

Multiply by “Size” factor

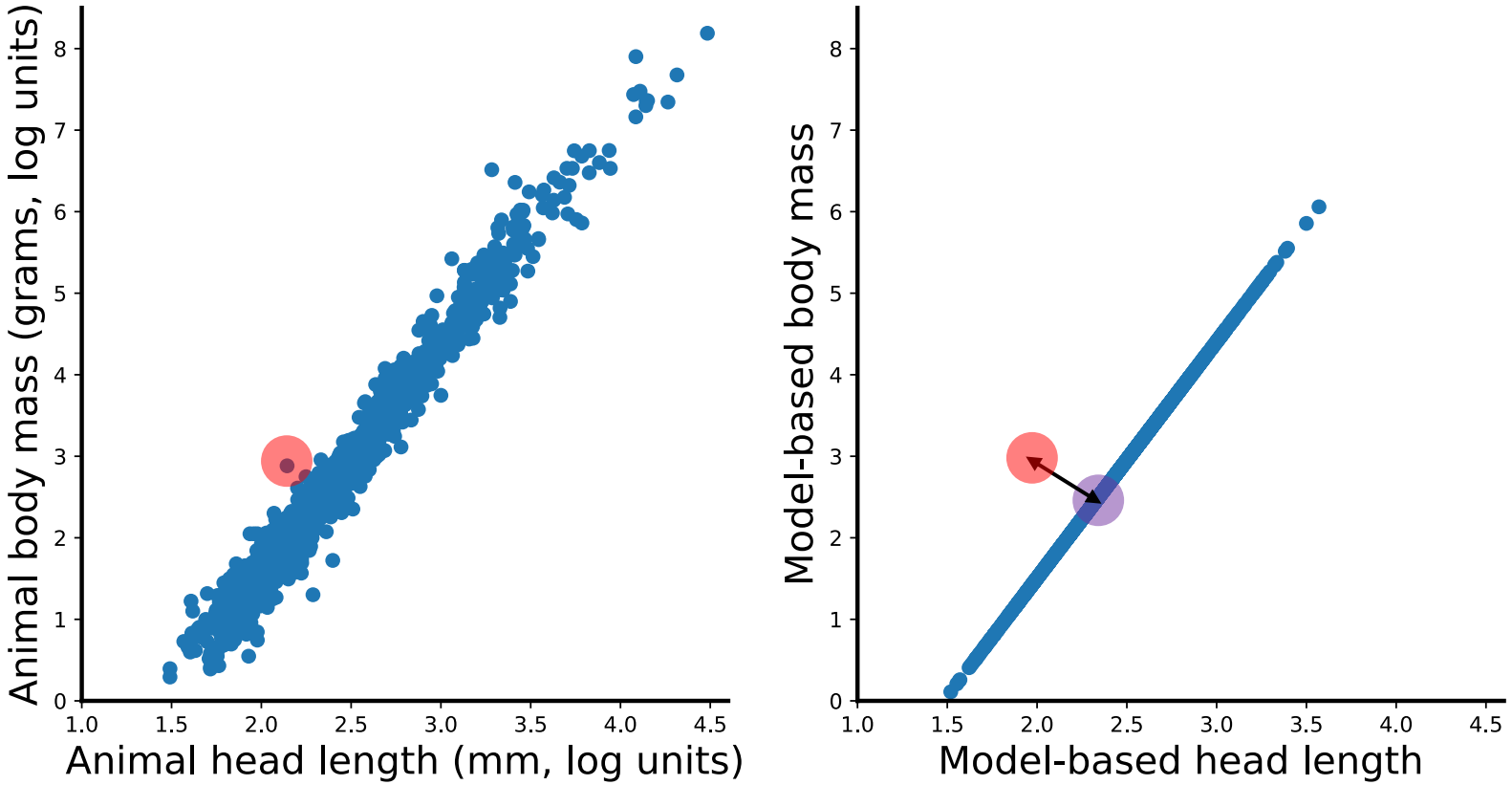


Noise contribution explains deviation of data from factor values

Date recipe:

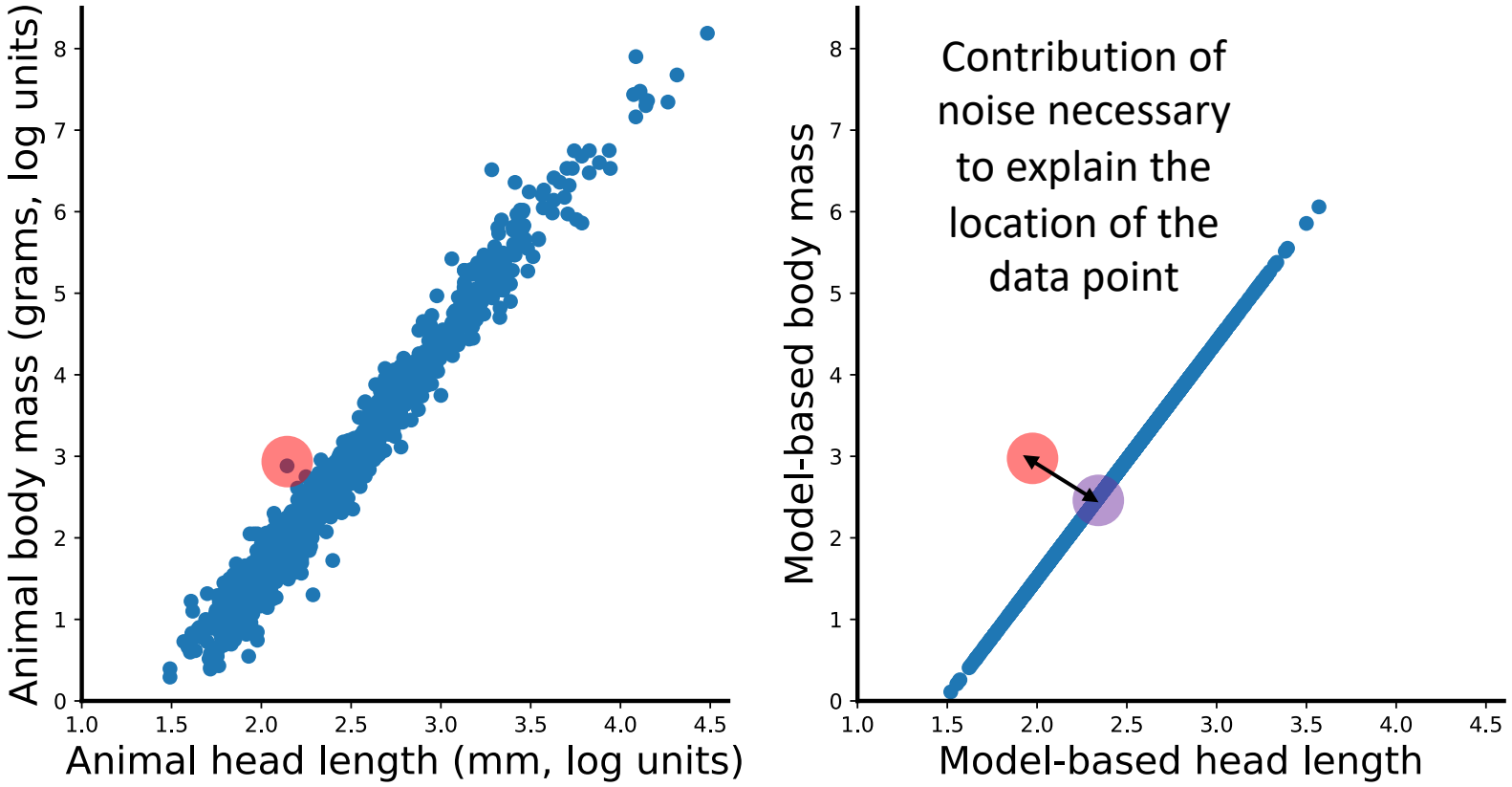
Start with random 1D Gaussian

Multiply by “Size” factor



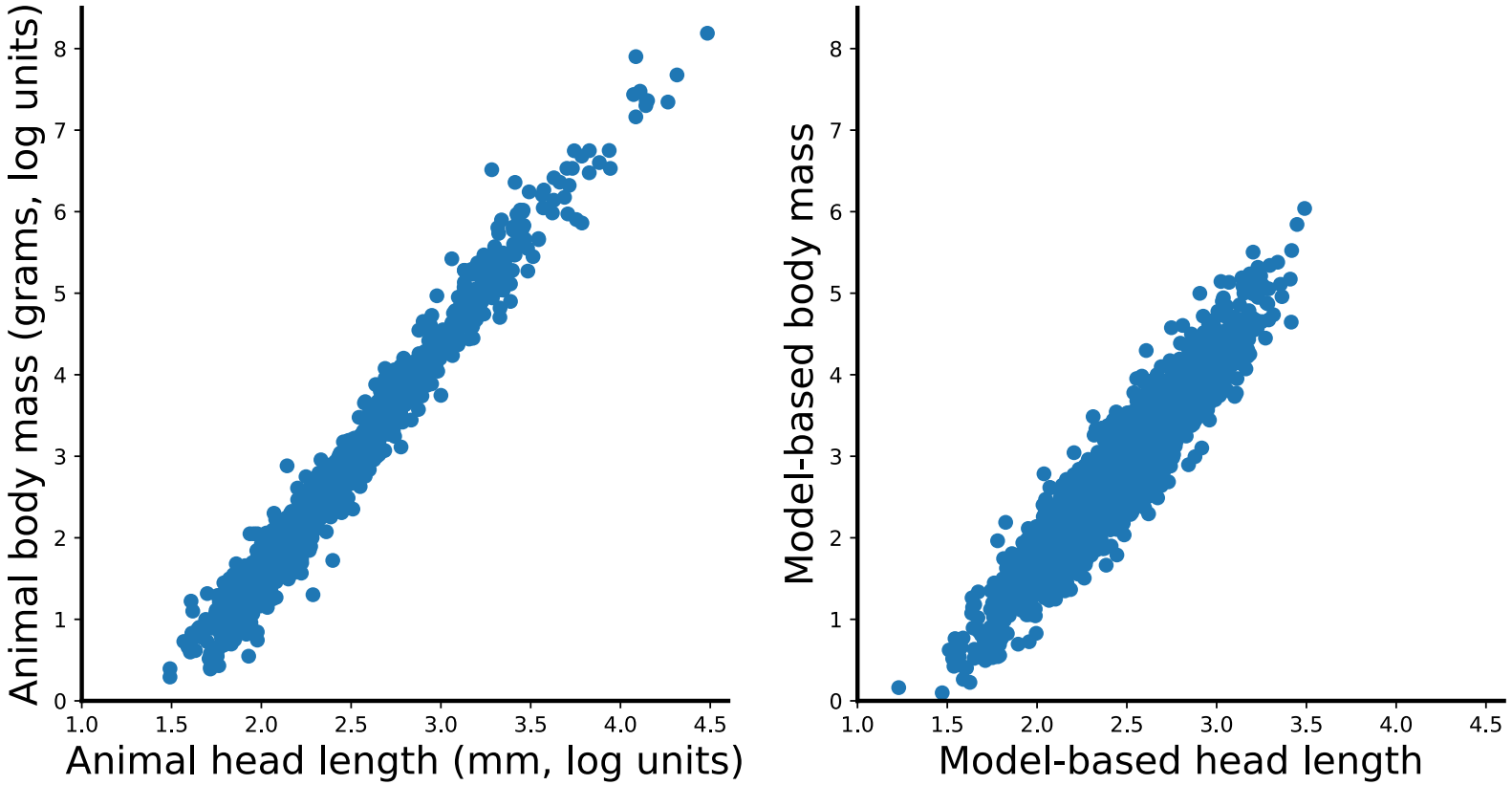
Noise contribution explains deviation of data from factor values

Date recipe:
Start with random 1D Gaussian
Multiply by “Size” factor
Add independent noise to each point



Latent variable models as data recipes

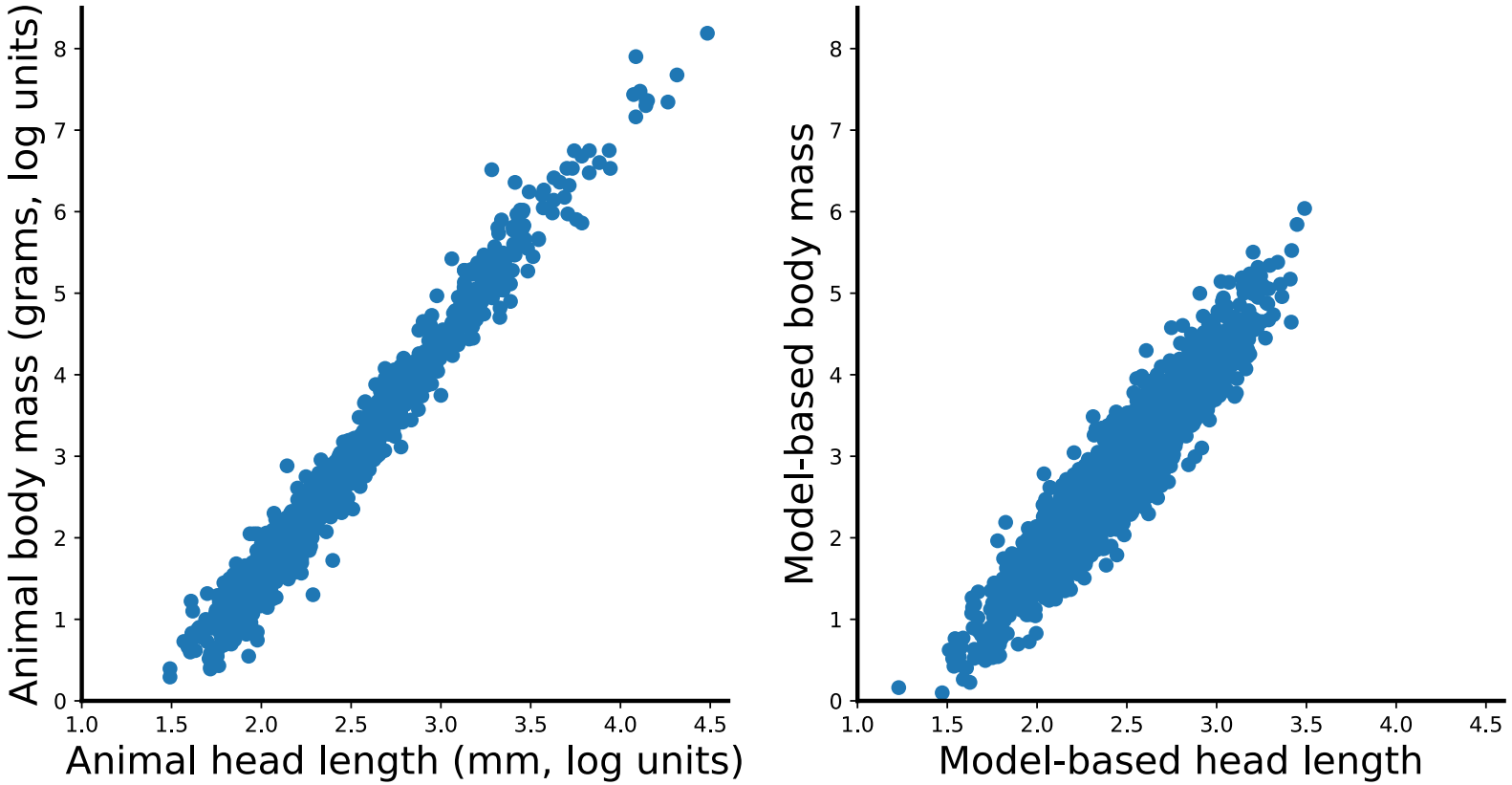
Data recipe:
Start with random 1D Gaussian
Multiply by “Size” factor
Add independent noise to each point



Latent variable models as data recipes

Data recipe:
Start with random 1D Gaussian
Multiply by “Size” factor
Add independent noise to each point

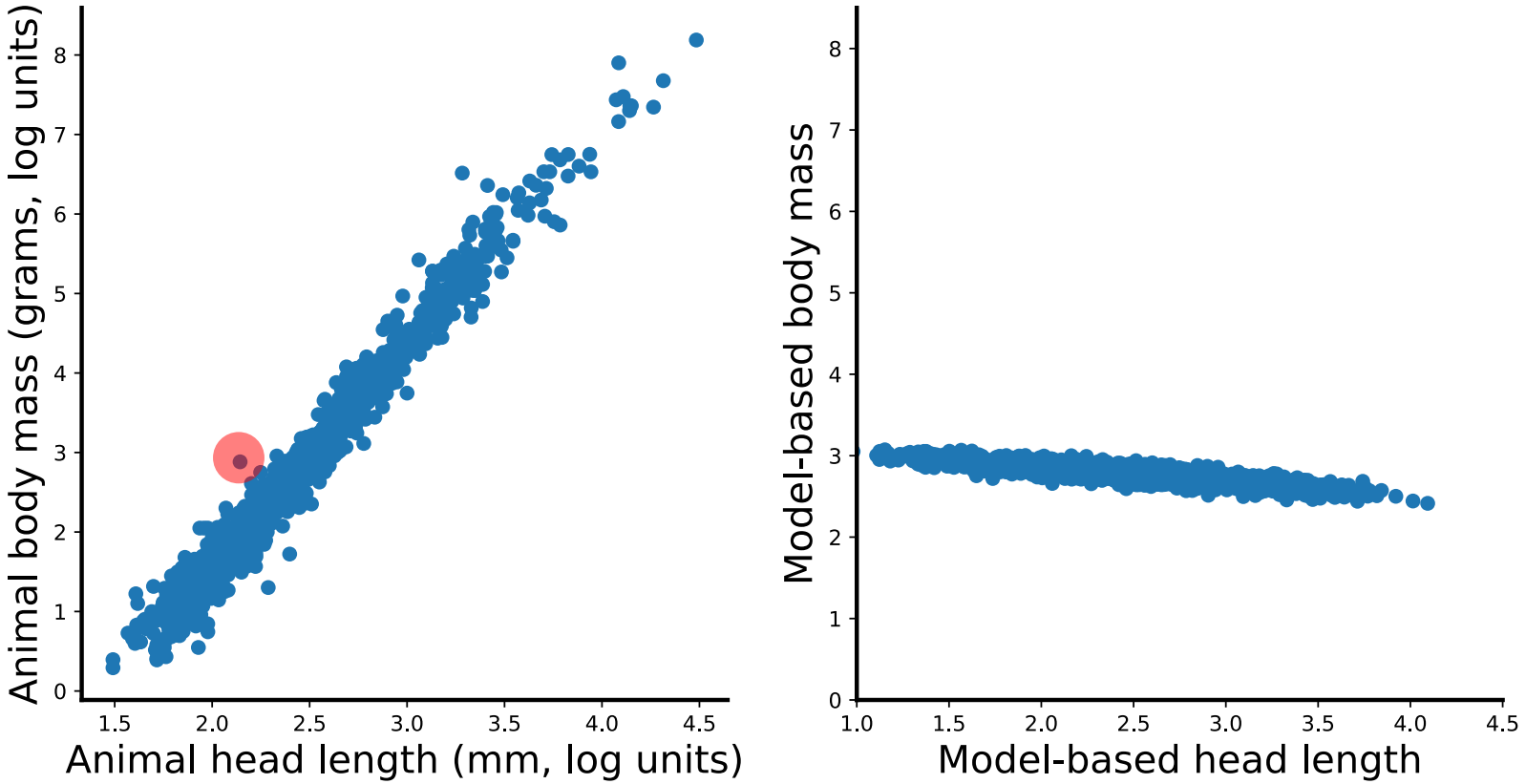
Why a recipe?
It allows us to say just how likely is it to have observed an animal with a certain head length and body mass given the model



Example of a bad factor model

Why a recipe?

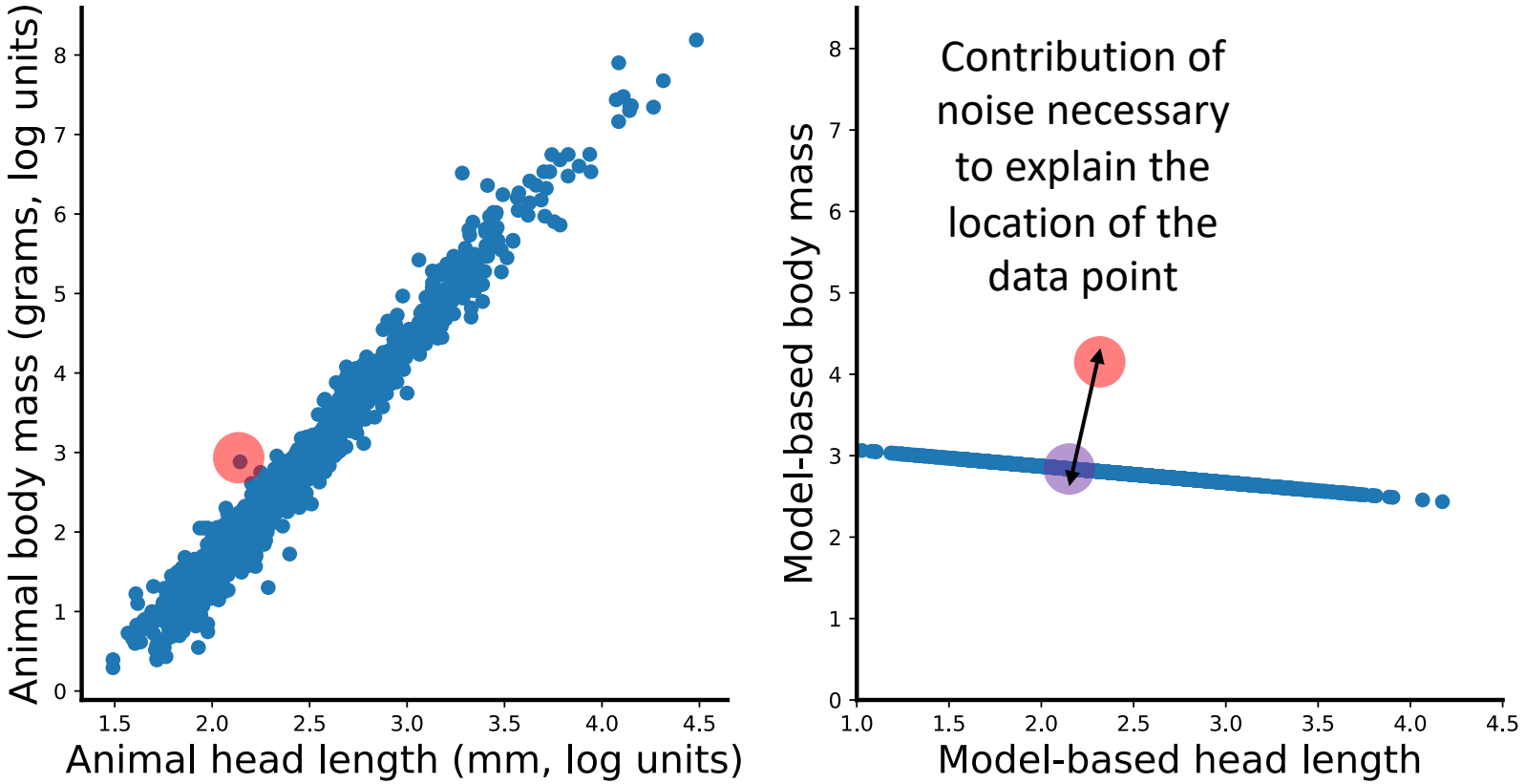
It allows us to say just how likely is it to have observed an animal with a certain head length and body mass given the model



Bad models need increasingly unlikely noise contribution

Why a recipe?

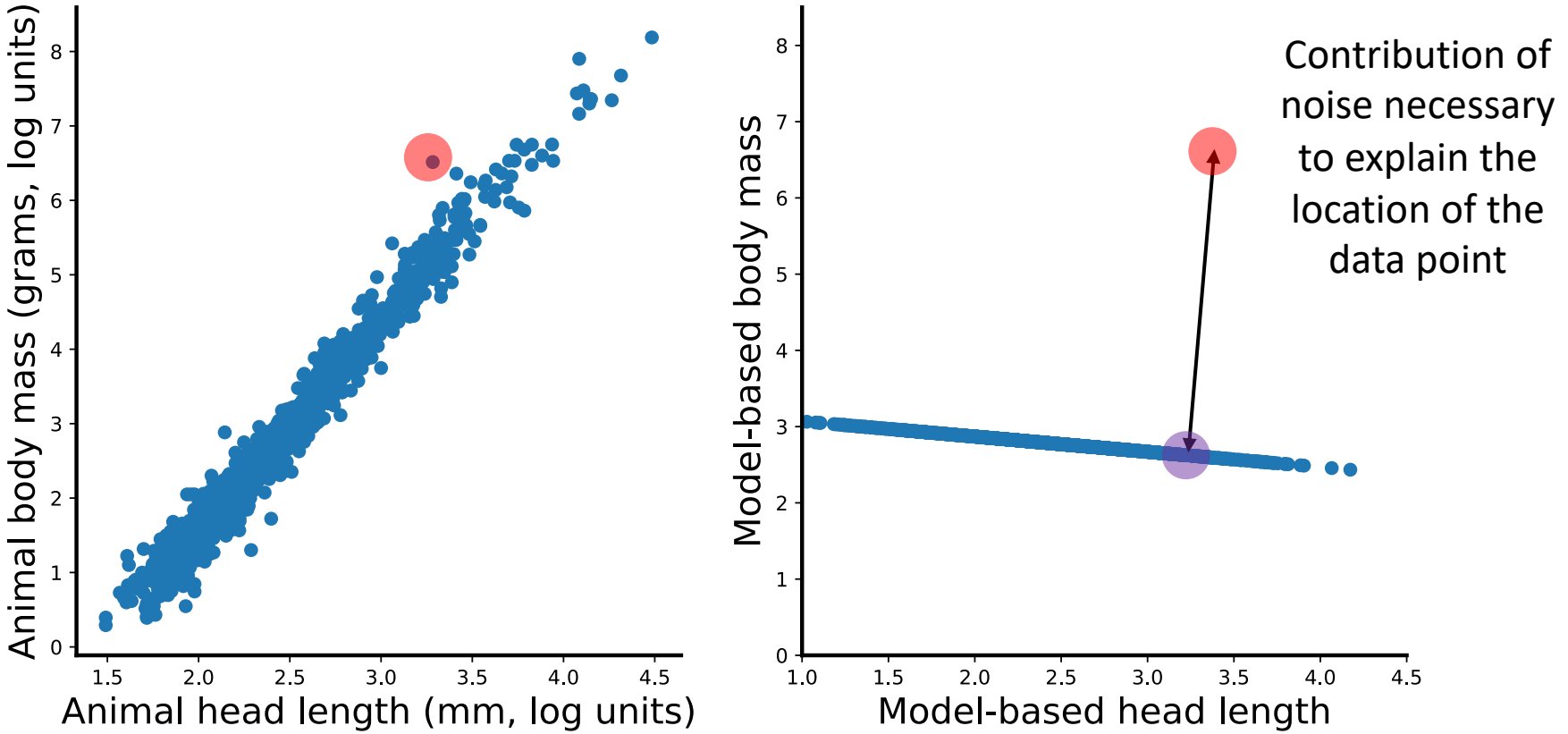
It allows us to say just how likely is it to have observed an animal with a certain head length and body mass given the model



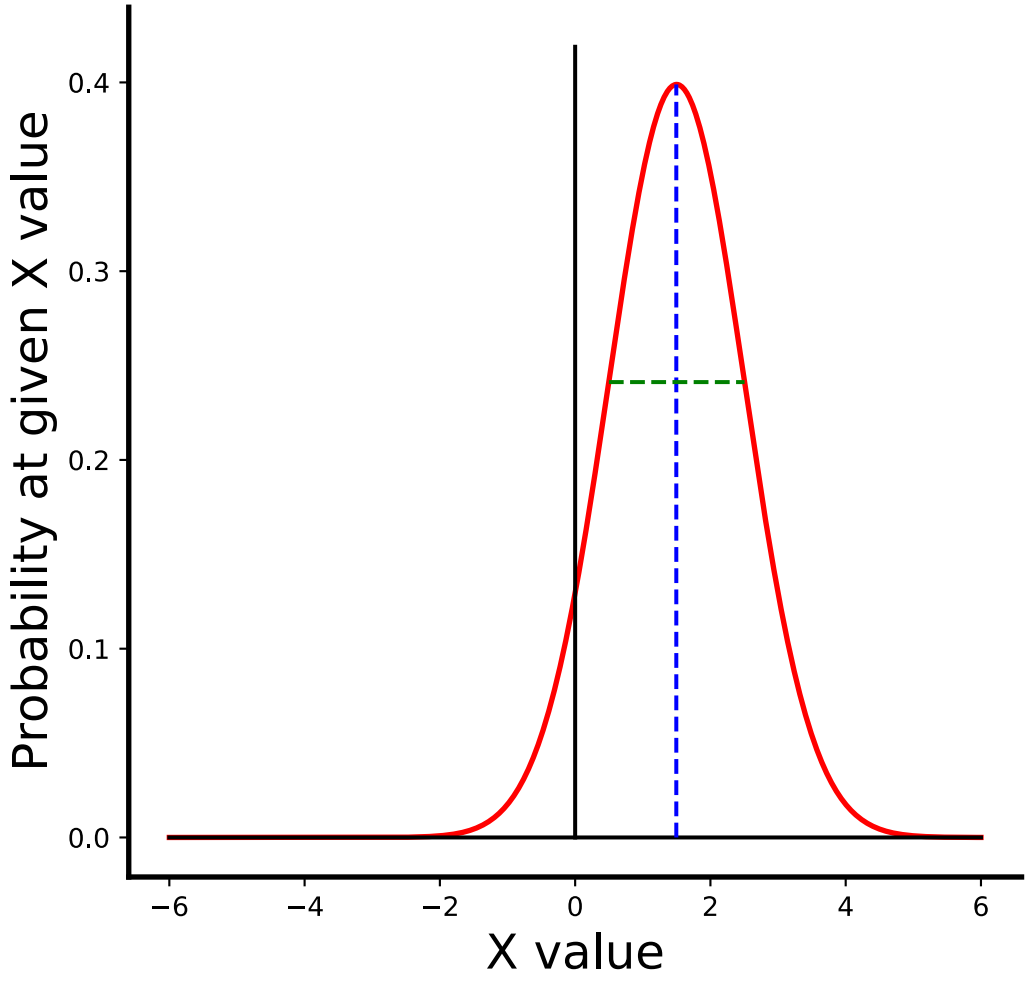
Bad models need increasingly unlikely noise contribution

Why a recipe?

It allows us to say just how likely is it to have observed an animal with a certain head length and body mass given the model



Probability of a data point: Gaussian distributions



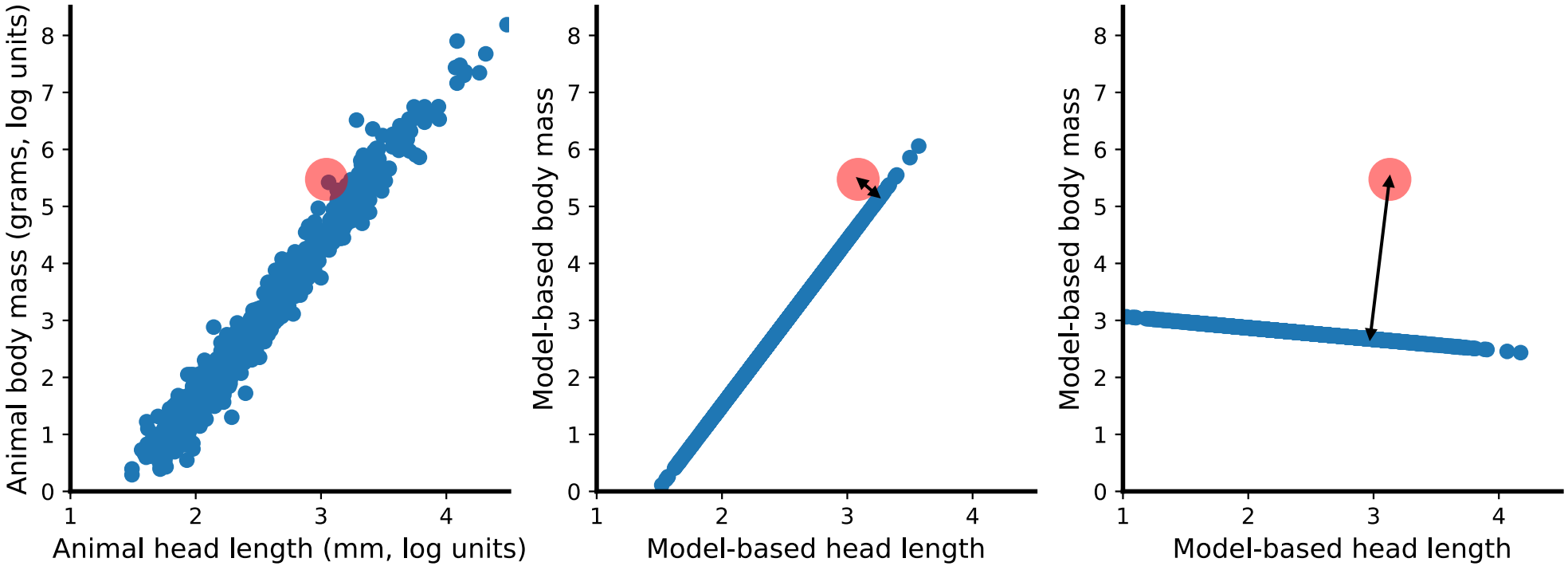
$$P\left(y = y^i_{data}\right) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{\left(y^i_{data} - \mu\right)^2}{2\sigma^2}\right)$$

One-dimensional Gaussian

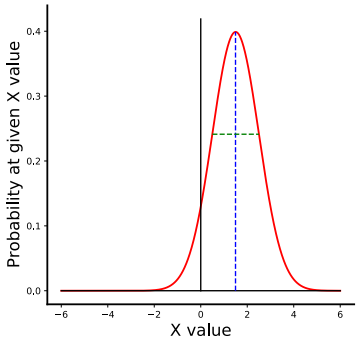
$$P\left(y = y^i_{data}\right) \sim -\frac{\left(y^i_{data} - \mu\right)^2}{2\sigma^2}$$

Probability is proportional to distance from mean in units of standard deviation

Data is more likely (less noise needed to explain) with good models

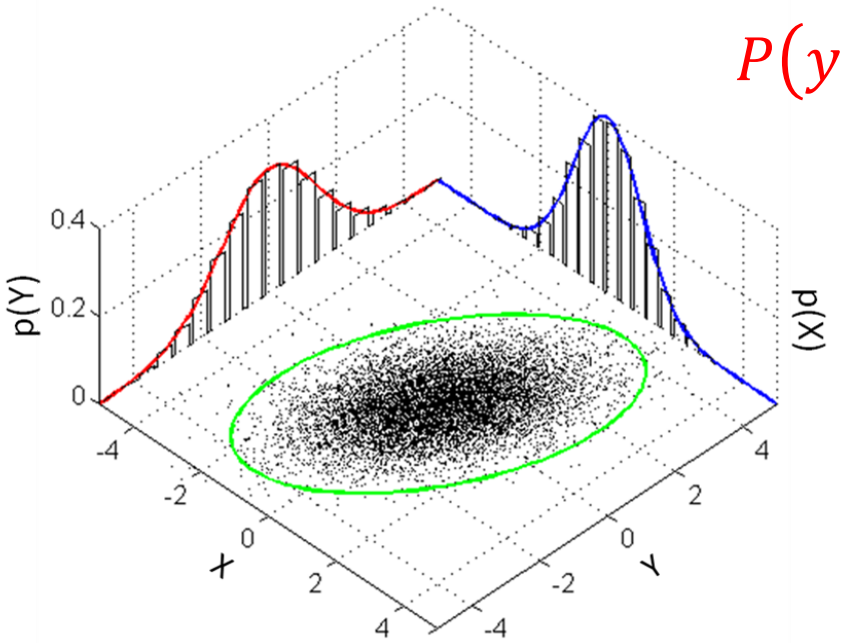


Probability of a data point: Gaussian distributions



$$P\left(y = y^i_{data}\right) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{\left(y^i_{data} - \mu\right)^2}{2\sigma^2}\right)$$

One dimensional Gaussian



$$P\left(y = y^i_{data}\right) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} exp\left(\left(y^i_{data} - \mu\right)^T \Sigma^{-1}\left(y^i_{data} - \mu\right)\right)$$

Multivariate Gaussian

$$P\left(y = y^i_{data}\right) \sim \left(y^i_{data} - \mu\right)^T \Sigma^{-1}\left(y^i_{data} - \mu\right)$$

Probability is proportional to distance from mean in units of standard deviation

And now with equations: probability of a data point

Data recipe:

Start with random 1D Gaussian $P(z) = \mathcal{N}(0,1)$

Multiply by size factor $x = z * f + \mu$

Assume independent noise for each point $P(y|x) \sim \mathcal{N}(0, \Psi)$

We can now write the probability of a data point:

$$P(y) \sim \mathcal{N}(\mu, ff^T + \Psi)$$

And the full expression:

$$P\left(y = y^i_{data}\right) = \frac{1}{(2\pi)^{n/2}|ff^T + \Psi|^{1/2}} exp\left(\left(y^i_{data} - \mu\right)^T (ff^T + \Psi)^{-1}\left(y^i_{data} - \mu\right)\right)$$

And now with equations: probability of dataset

$$P(y) \sim \mathcal{N}(\mu, ff^T + \Psi)$$

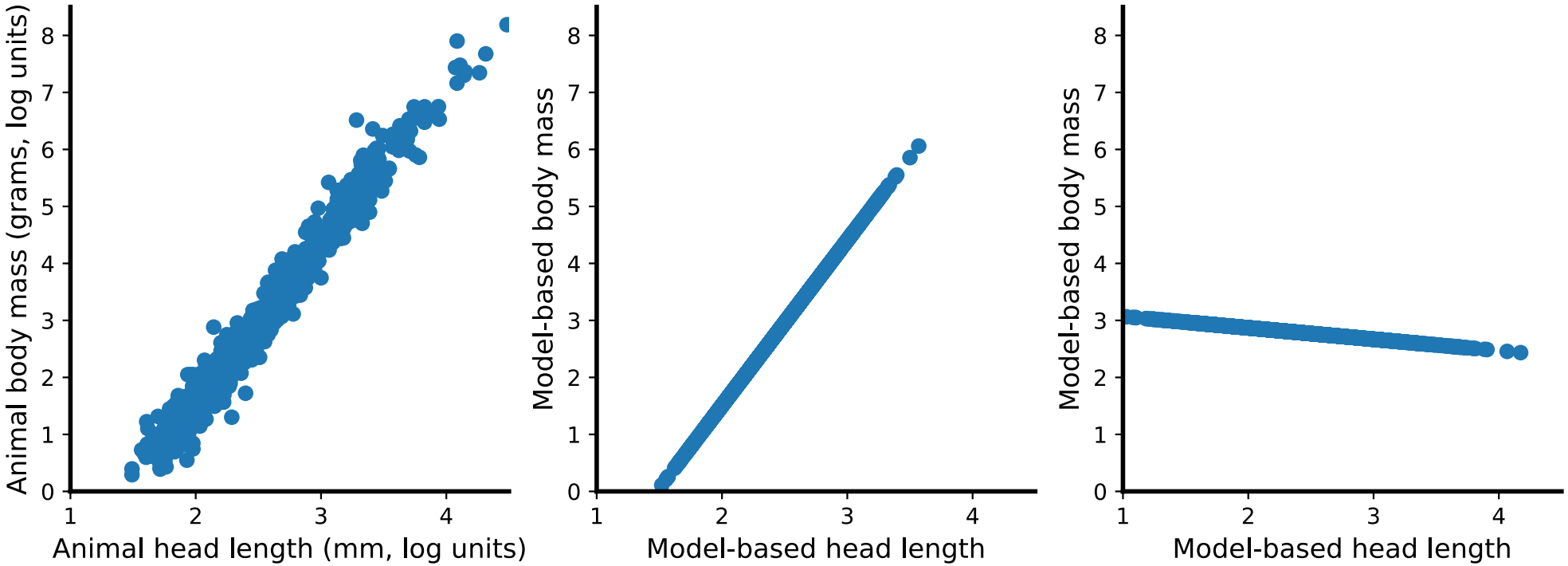
If we assume each data point is independent, then probability of multiple independent events is just multiplying the probability of each event:

$$P(Data) = \prod_{data\ points\ i} P(y = y^i_{data})$$

$$\log(P(Data)) = \sum_{data\ points\ i} \log(P(y = y^i_{data})$$

We find the latent structure by assuming a given structure and finding the parameters (relation between size factor and head length / body mass) that maximize the likelihood of the data given the model

Data is more likely (less noise needed to explain) with good models



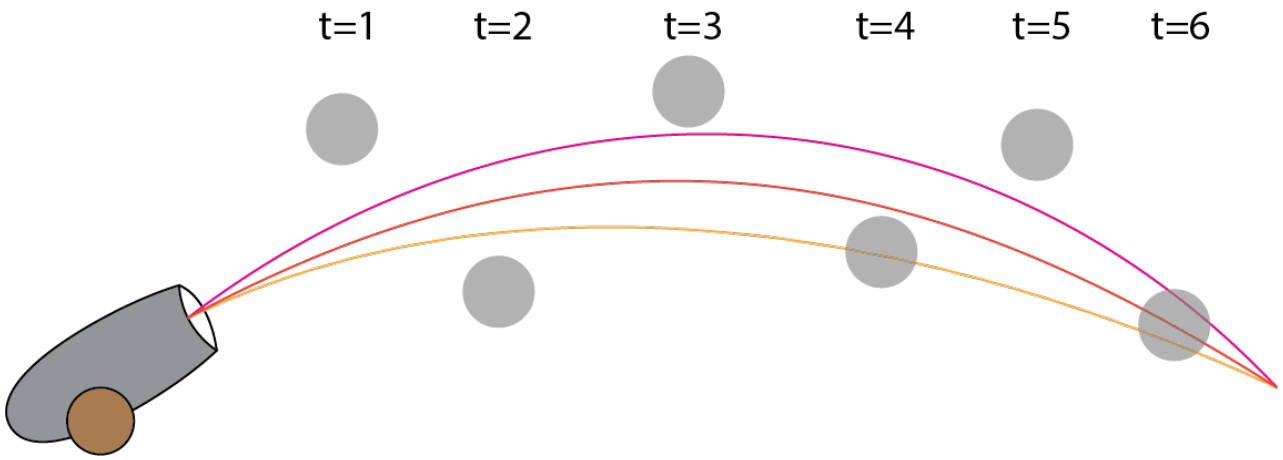
We find the latent structure by assuming a given structure and finding the parameters (relation between size factor and head length / body mass) that maximize the likelihood of the data given the model

Why all this trouble for something that looks like PCA?

Data recipes can capture much more interesting structure!

PCA is a completely static model, it completely ignores dynamics in the data

Latent dynamics models:



$$x = z * f + \mu$$

Factor Analysis model

$$x(t) = Ax(t-1) + \epsilon$$

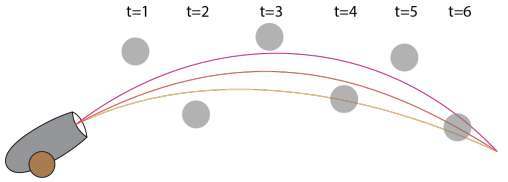
Latent Linear Dynamical System model

Summary

All biological data has structure! Just finding structure is not a result!

Finding structure can be useful if we can directly interpret it

Finding structure can be useful to deal with noisy measurements



This is not an exhaustive list. The point is that it needs to be thought through