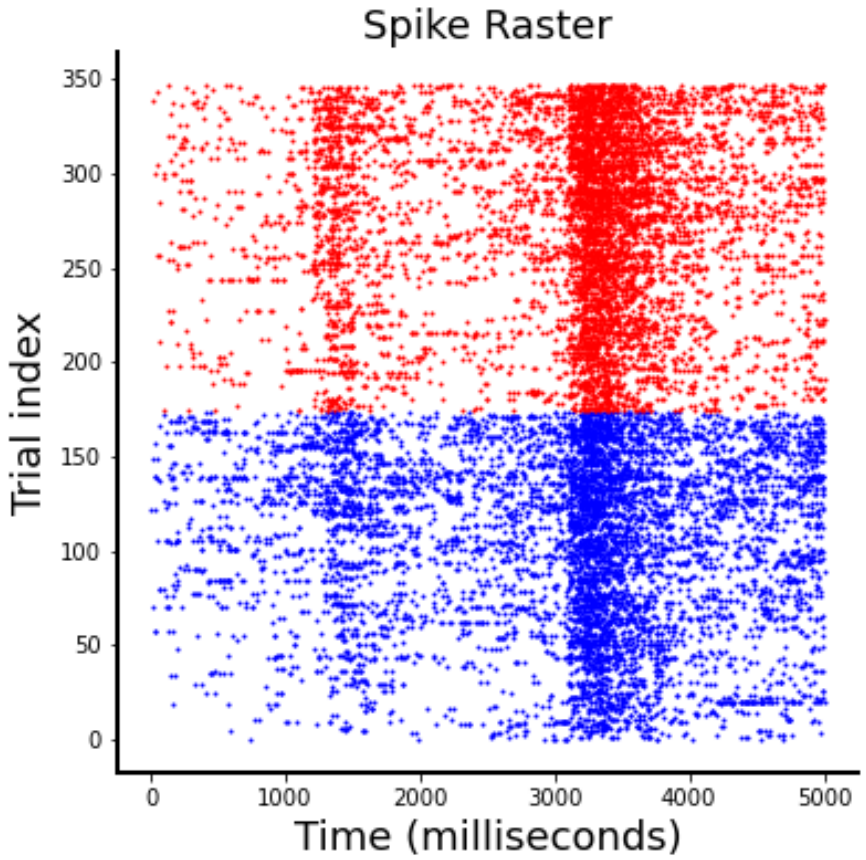
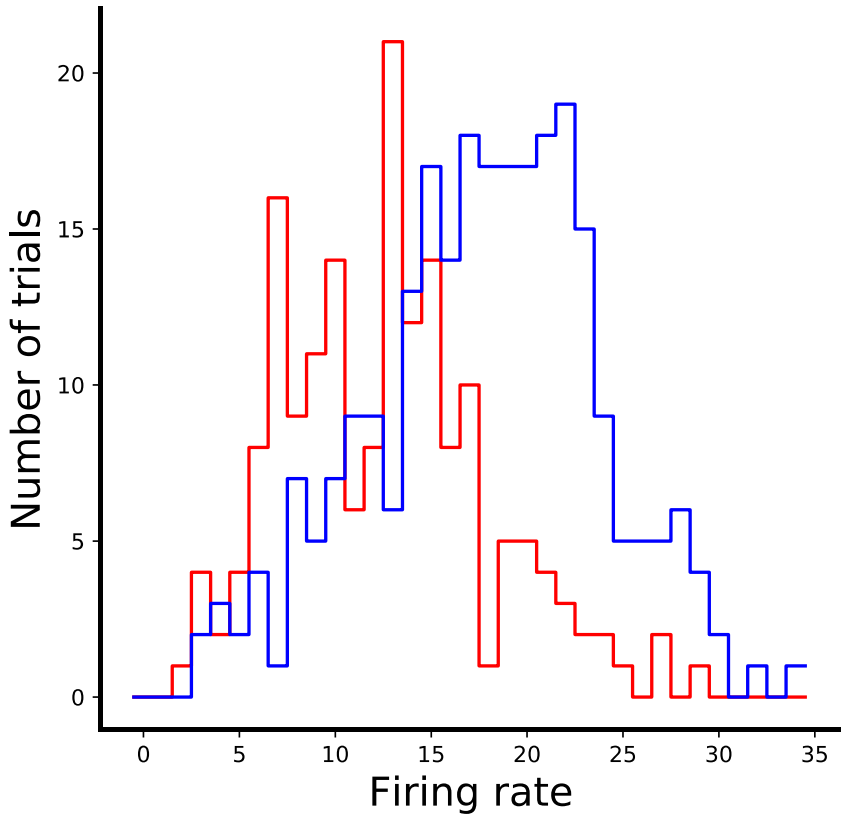
Population analysis lecture

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

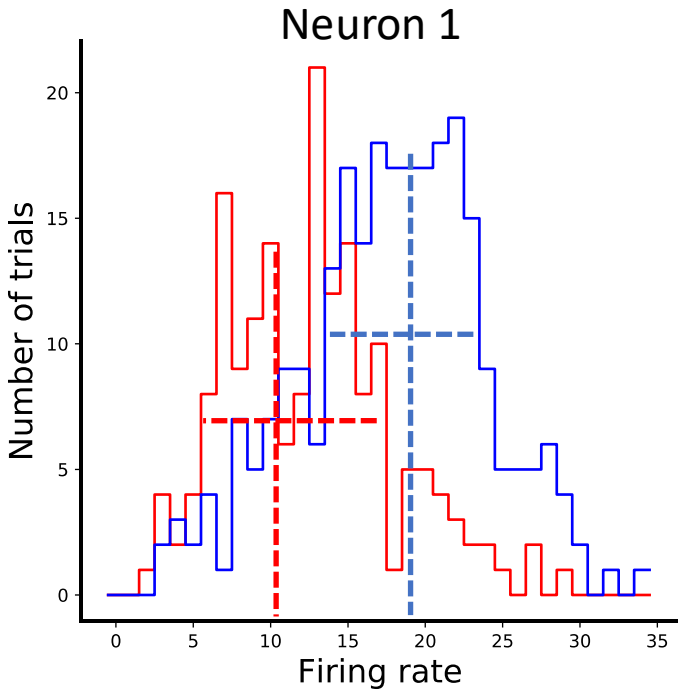




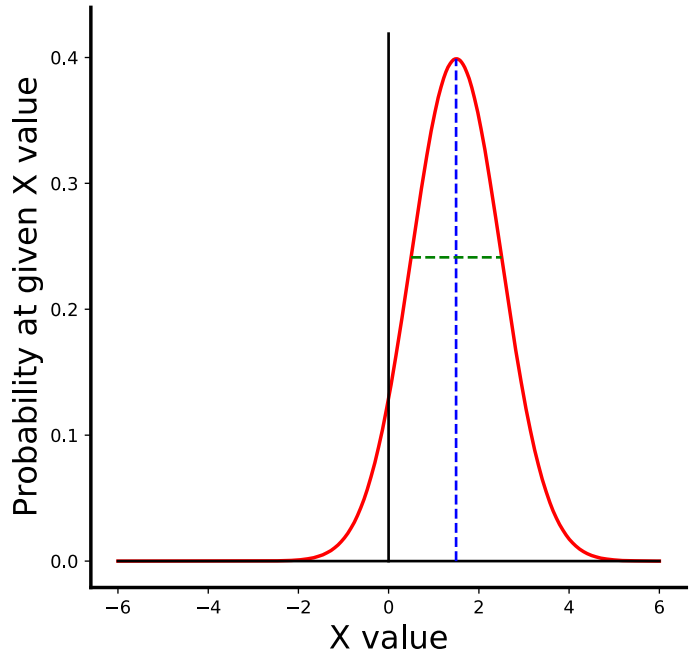
Quantifying decodability

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

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



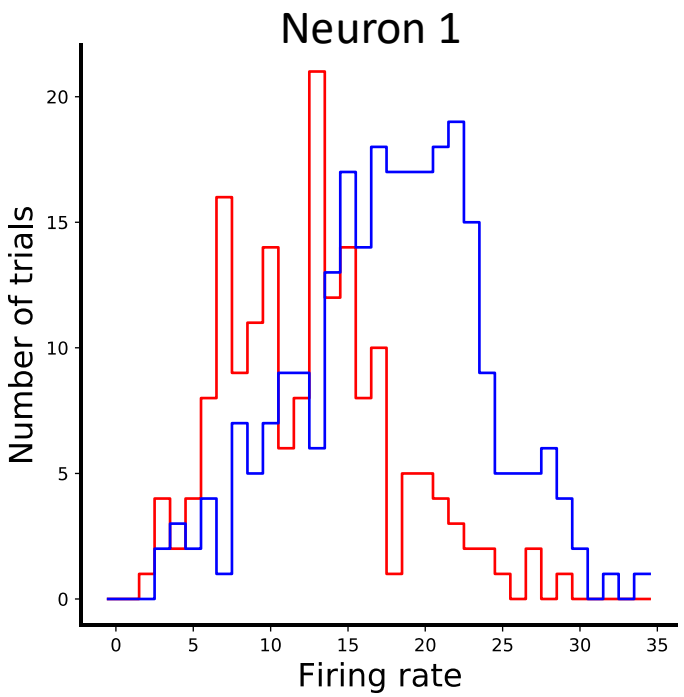
$$d'_{neuron\ 1} \approx 1$$

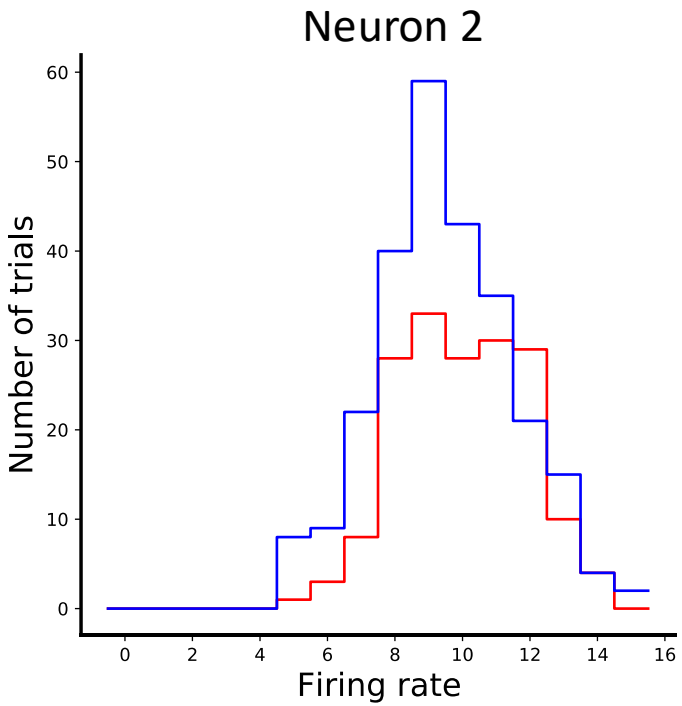


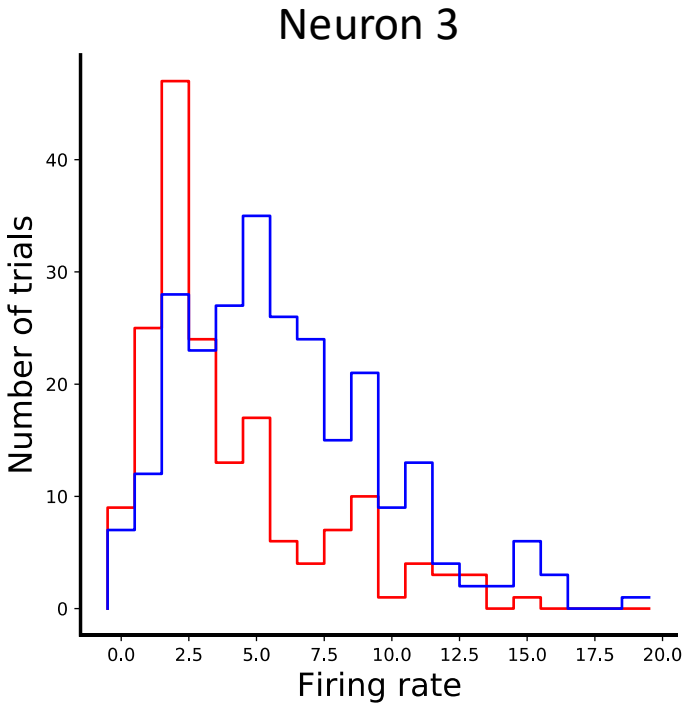
Quantifying decodability

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

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





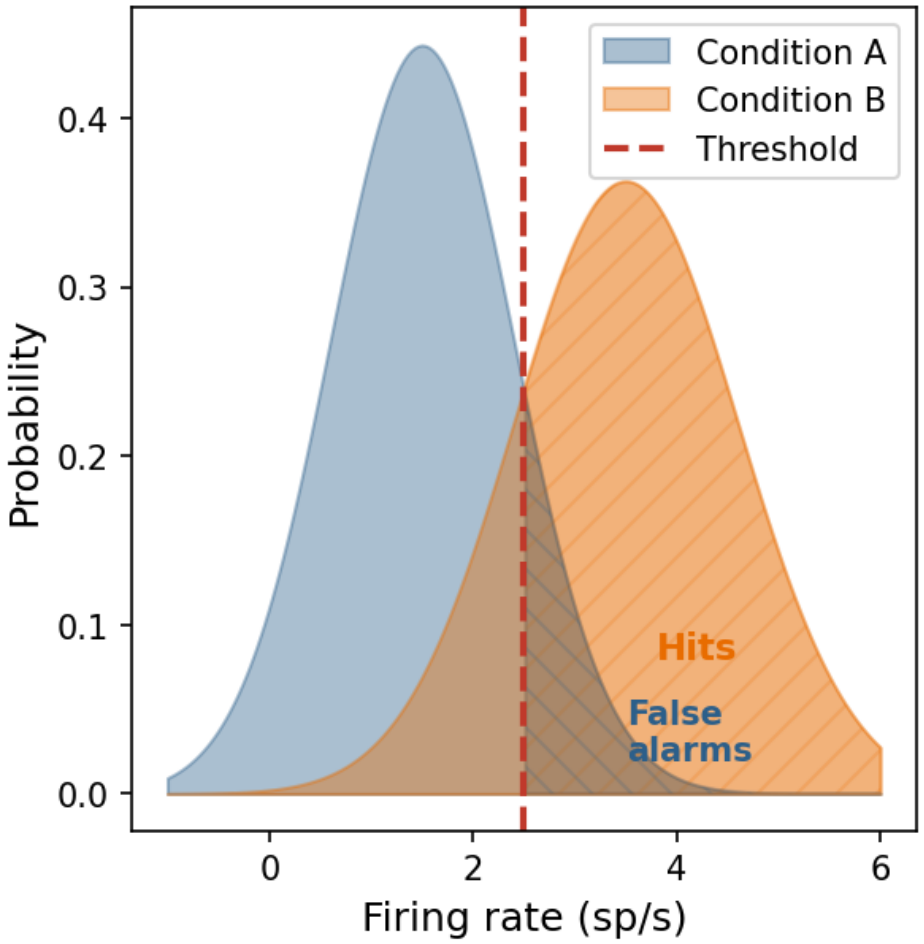


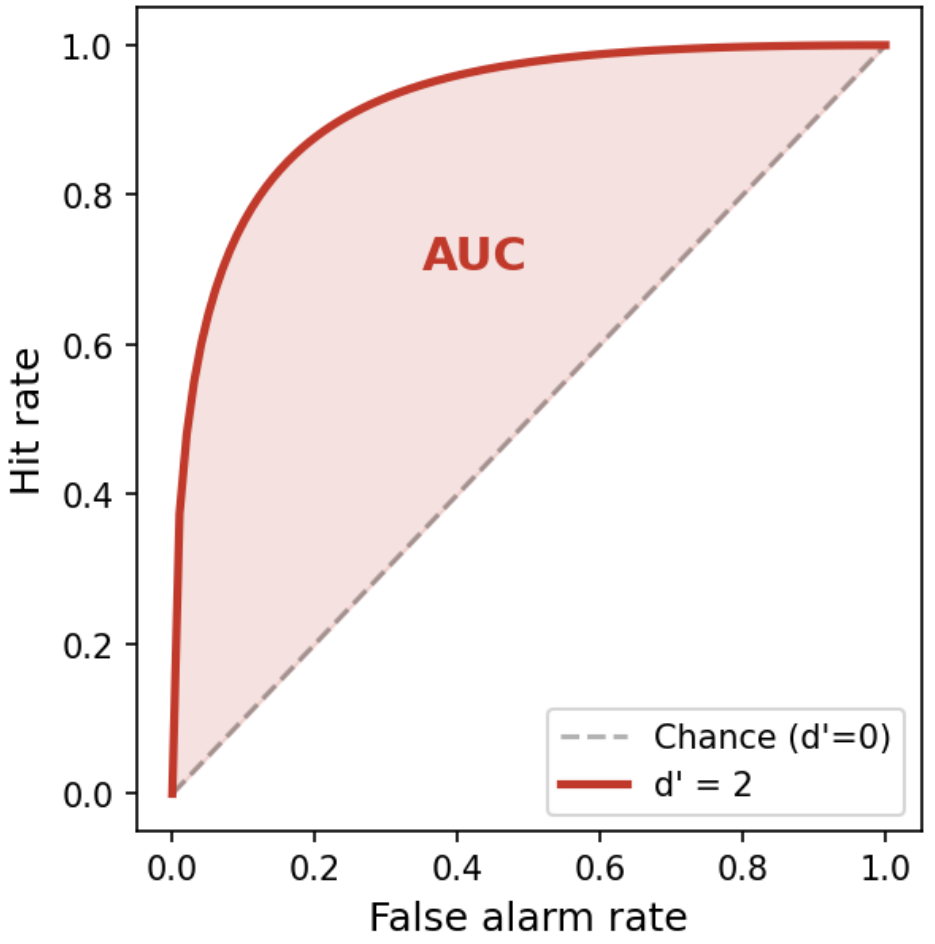
There are additional measures, like Receiver Operating Characteristic (ROC)

Receiver Operating Characteristic (ROC)

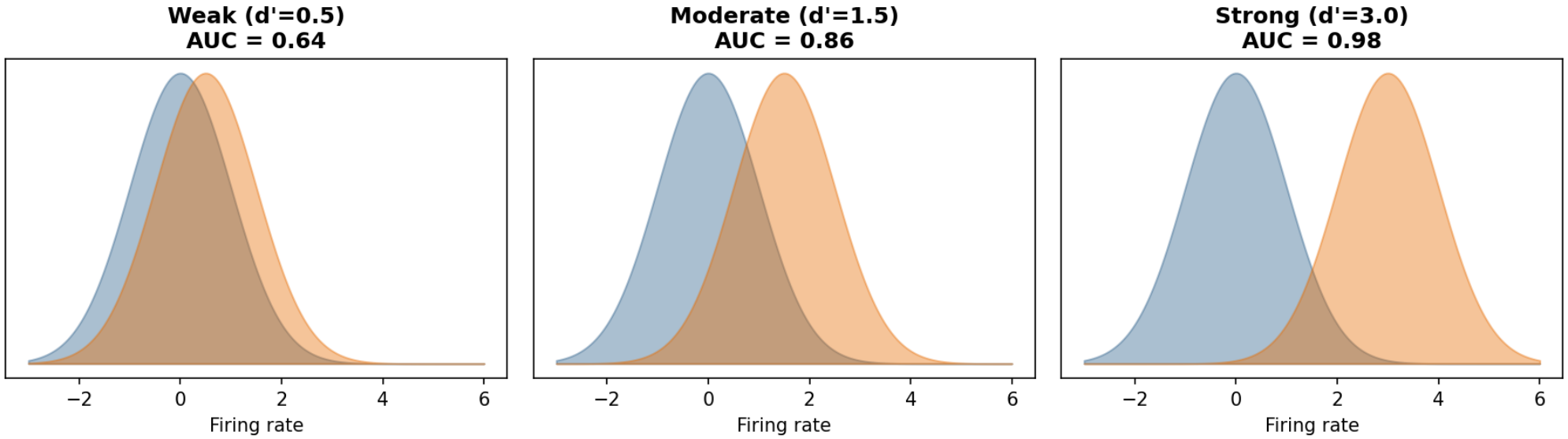
Link to viewer

Receiver Operating Characteristic (ROC)





AUC and d-prime are usually highly correlated

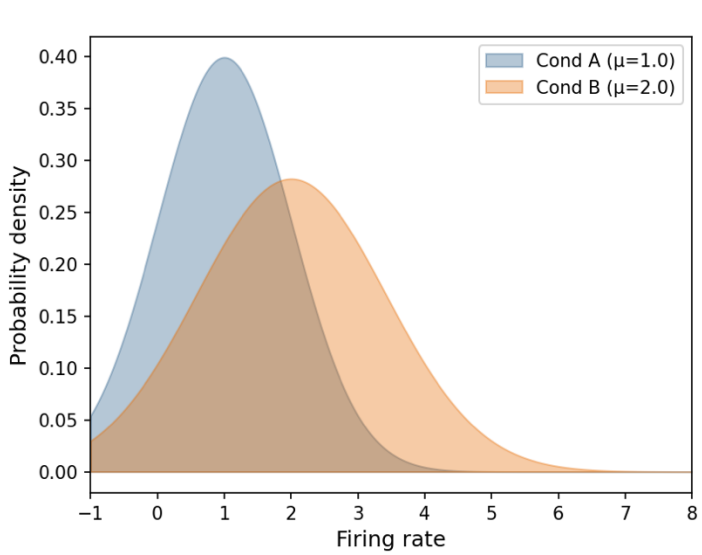


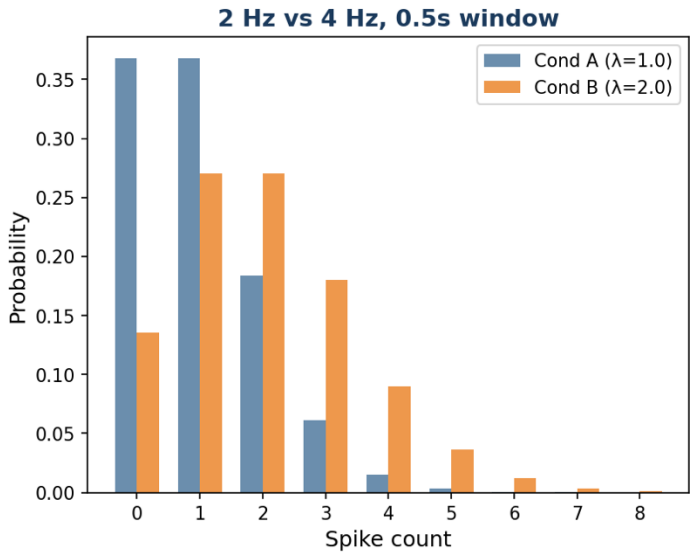
- d-prime measures separation in units of standard deviation
- AUC measures the probability of correctly ranking a random pair of trials
- For equal-variance Gaussians: AUC = Φ(d' / √2)
- Both are single-neuron summaries — they ignore relationships between neurons

Spike rates and time varying encoding challenges approaches

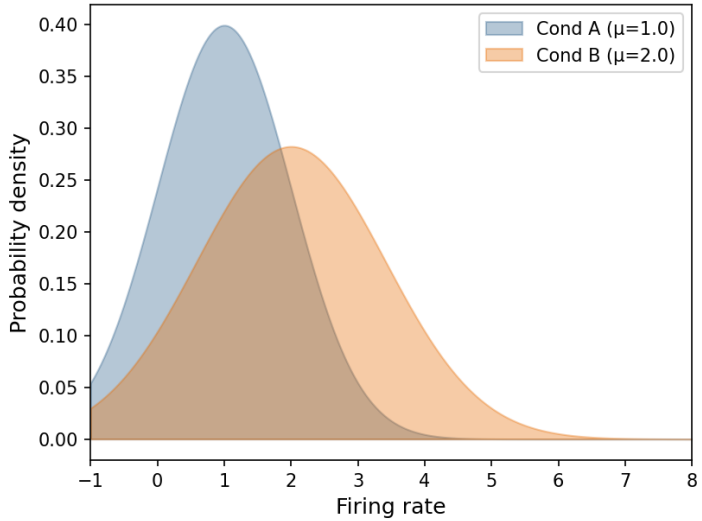
Link to viewer

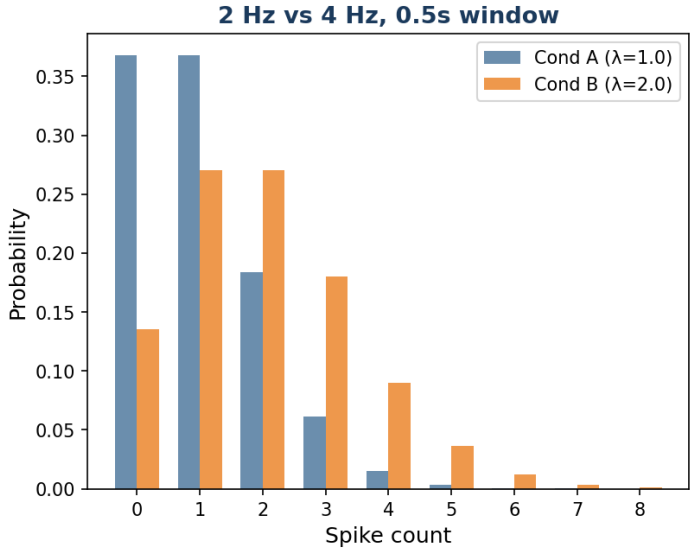
Spike rates and time varying encoding introduce discretization

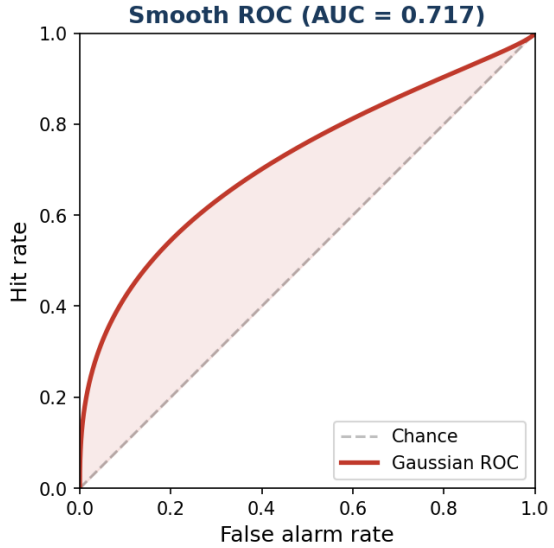


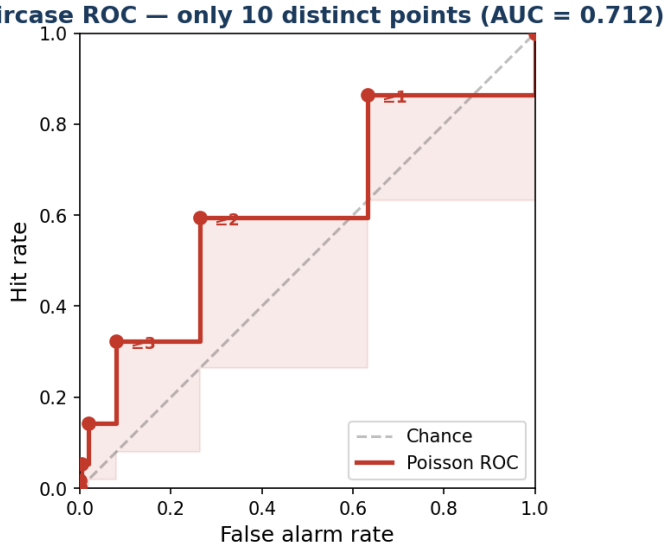


Spike rates and time varying encoding introduce discretization



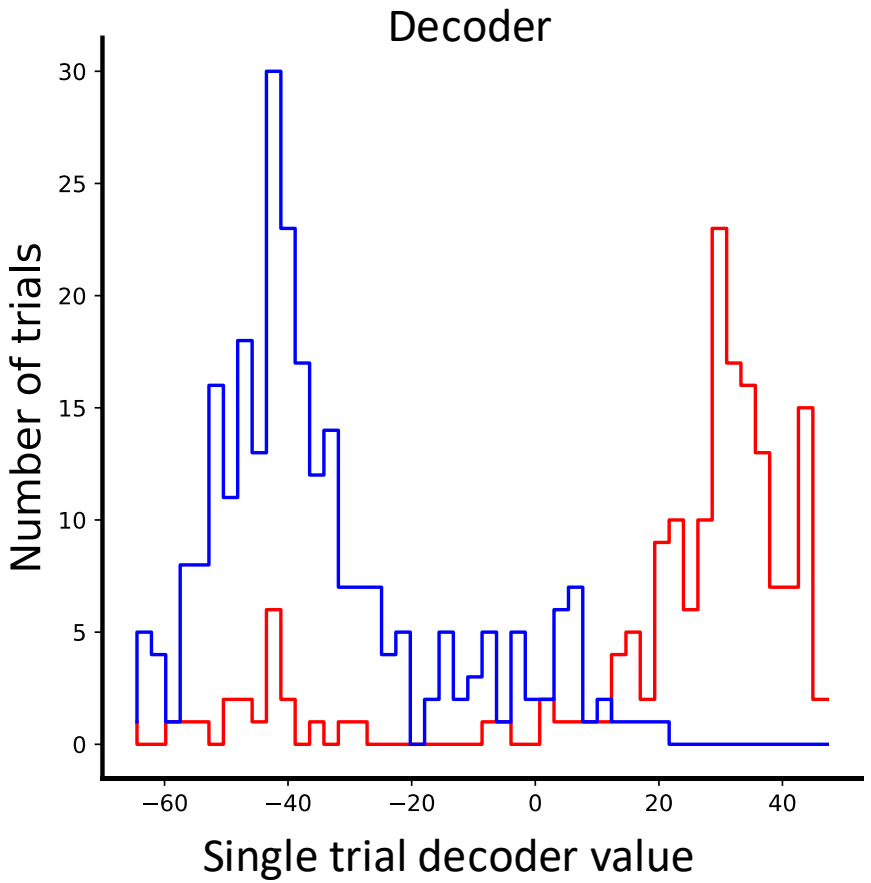


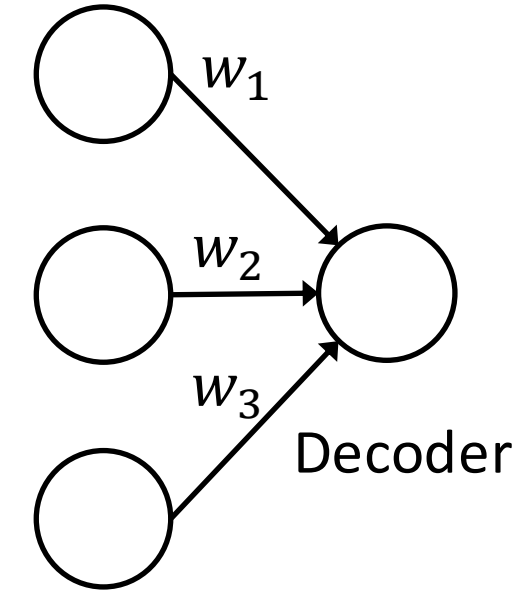




Population decoders

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

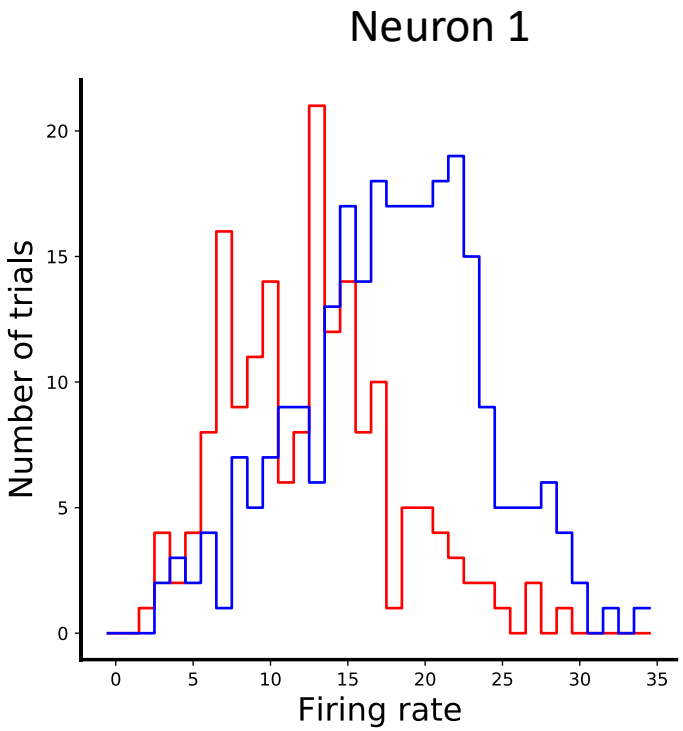


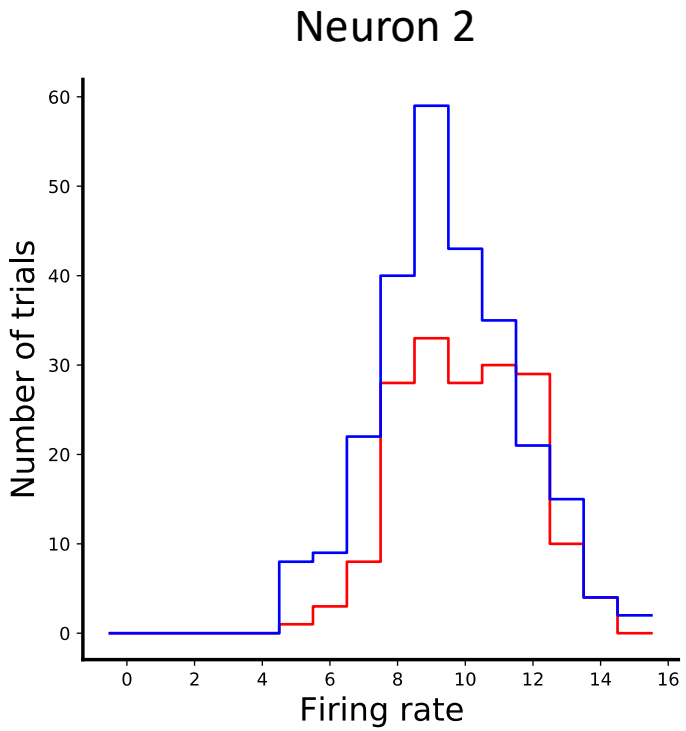


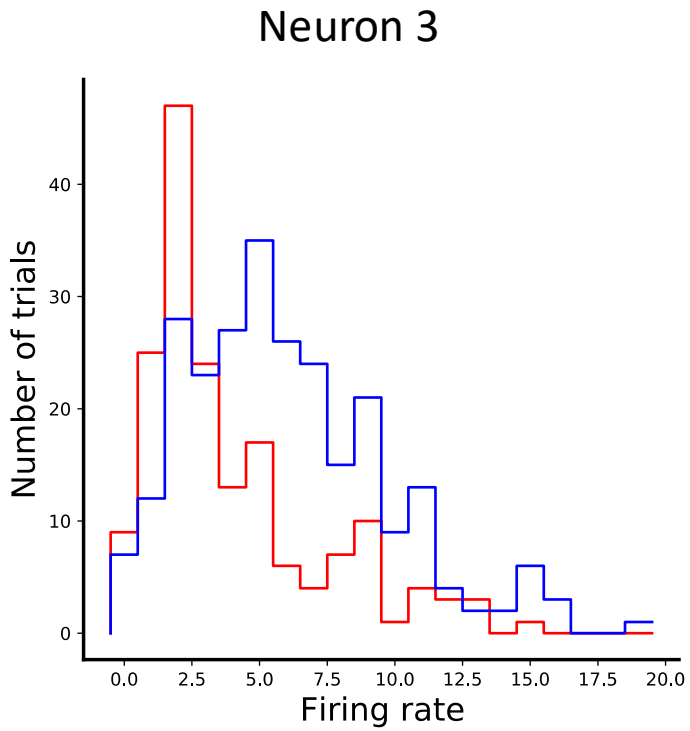
$$Single\ trial\ decoder\ value = \sum_{neurons} w_{neuron} * rate_{neuron}$$

Population decoders

What weights should we assign these neurons?



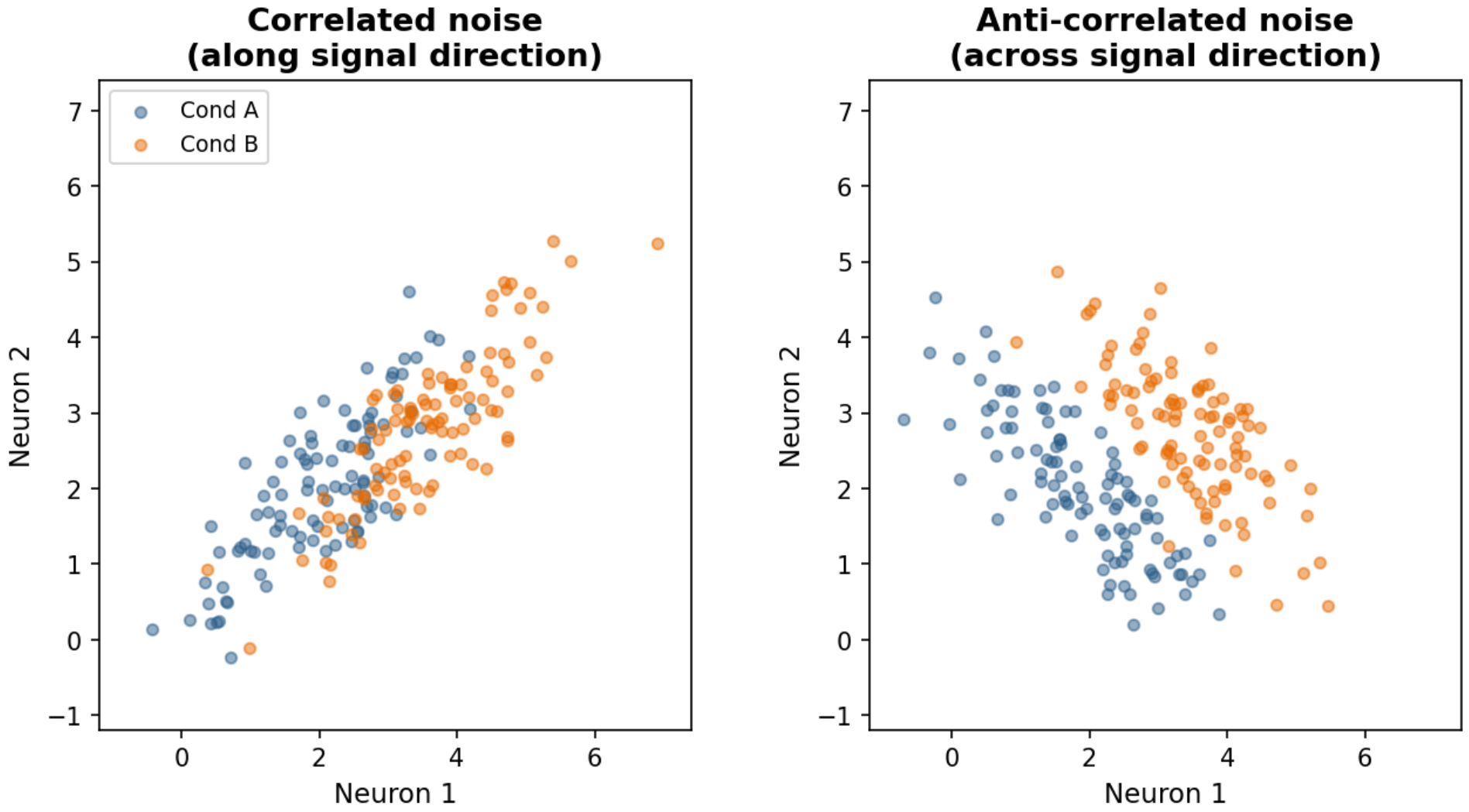




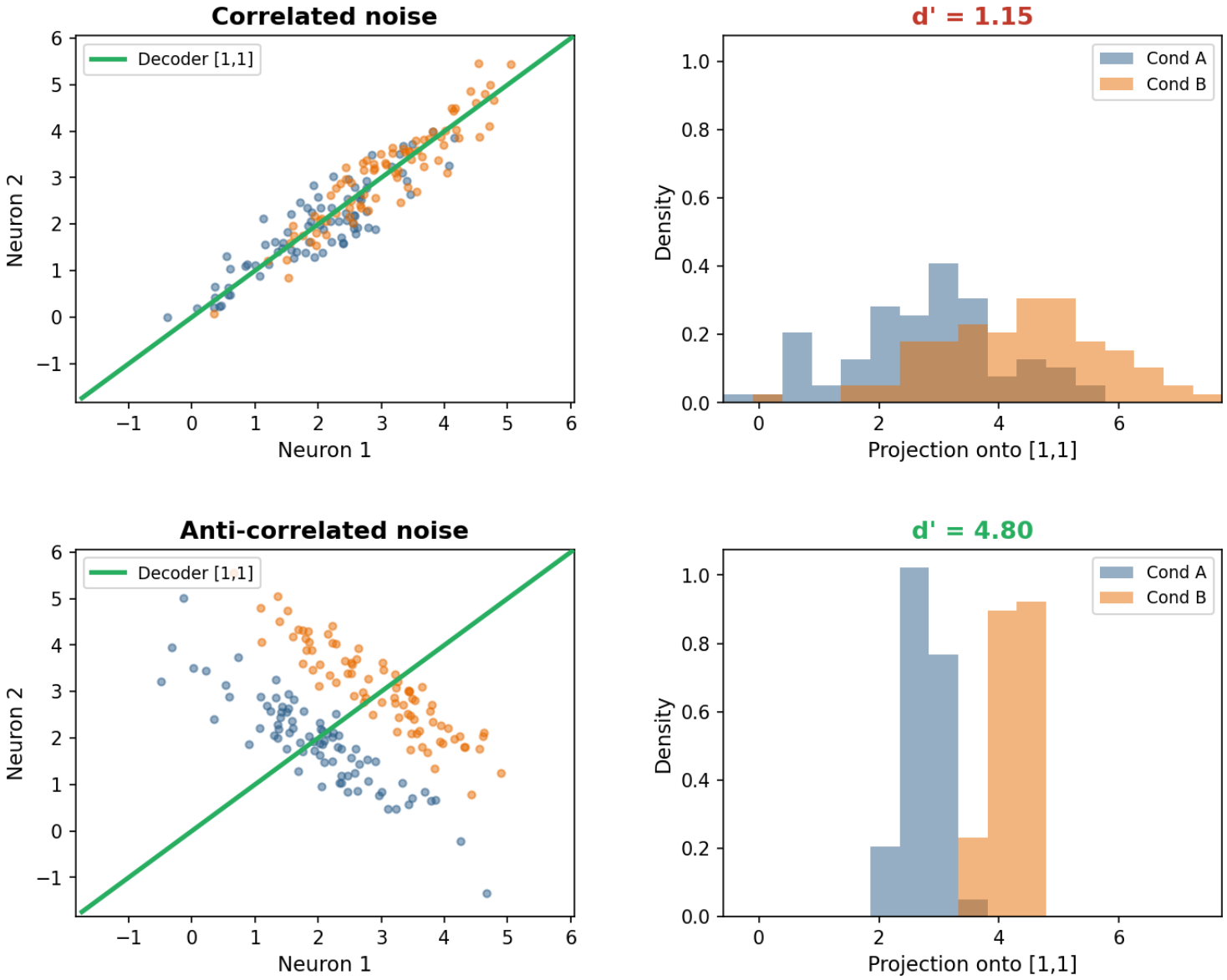
$$d' = \frac{\mu_1 - \mu_2}{\sigma_W}$$

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

Decoder weights should depend not just on encoding strength

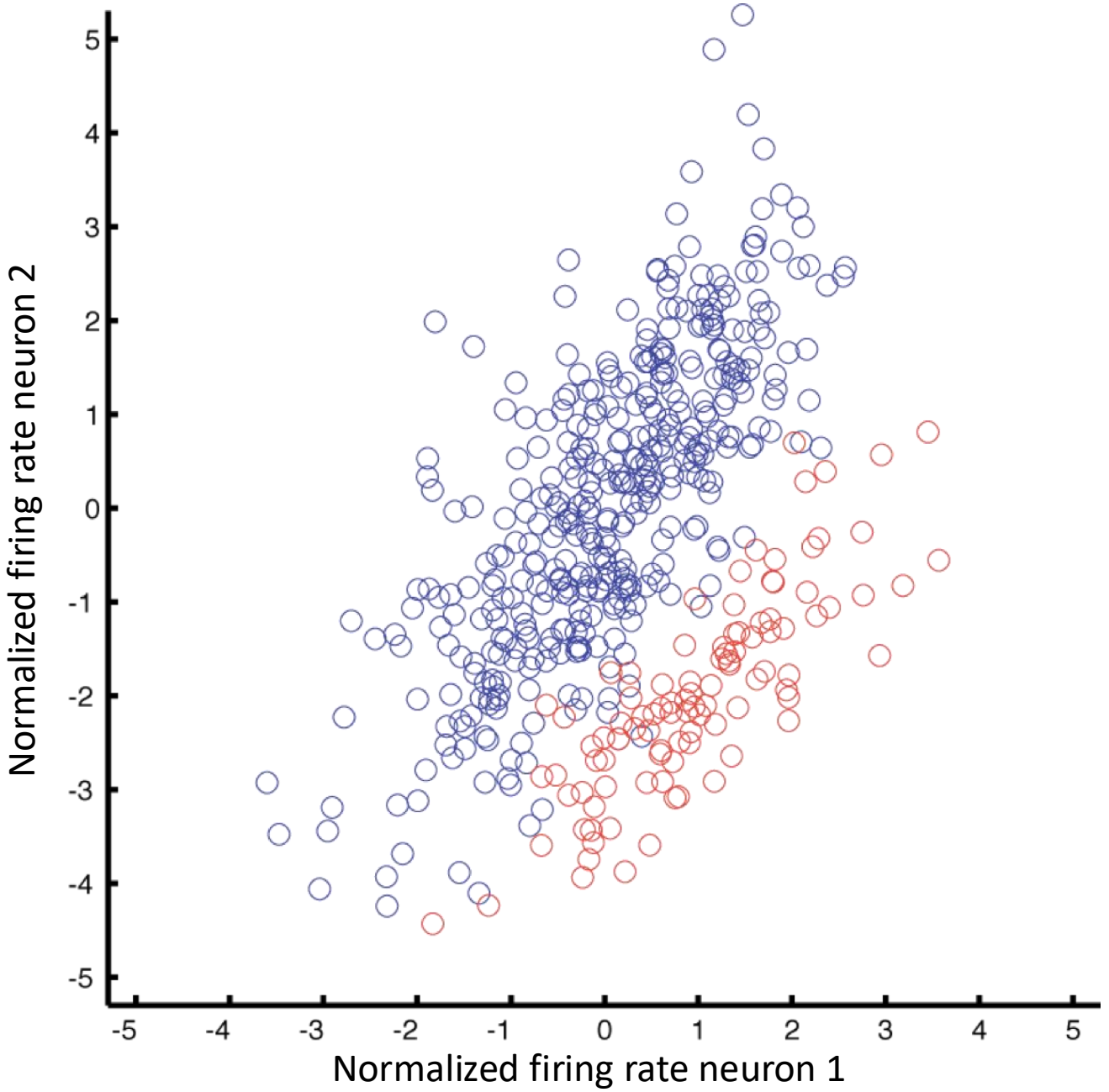


Decoder weights should depend not just on encoding strength



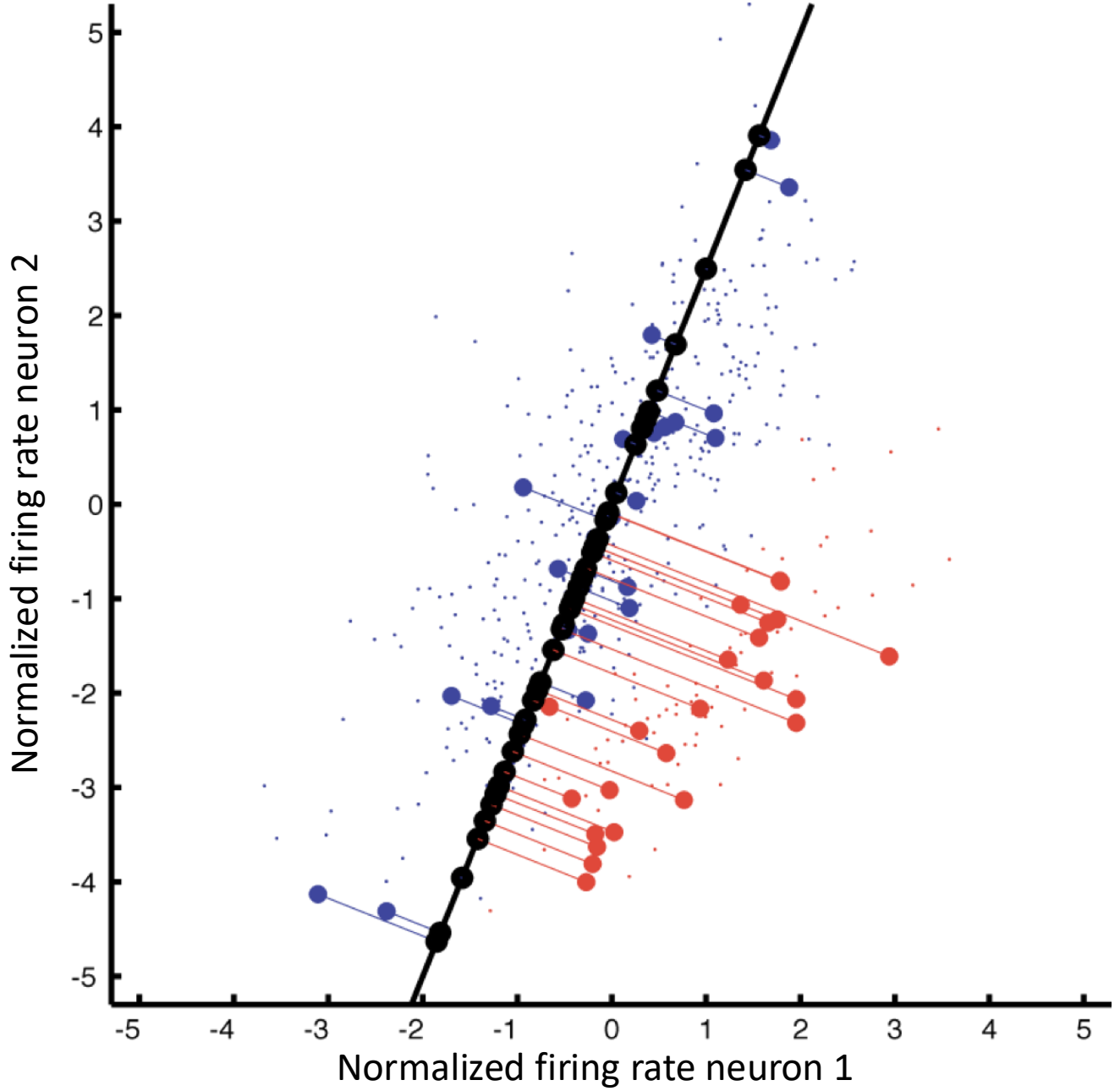
Example Linear Discriminant Analysis on synthetic data

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

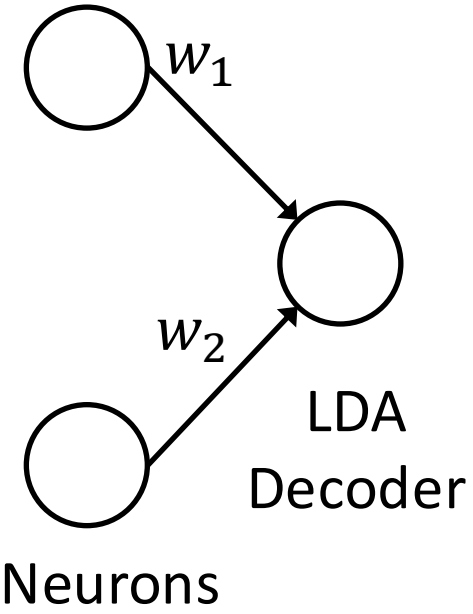


Example Linear Discriminant Analysis on synthetic data

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

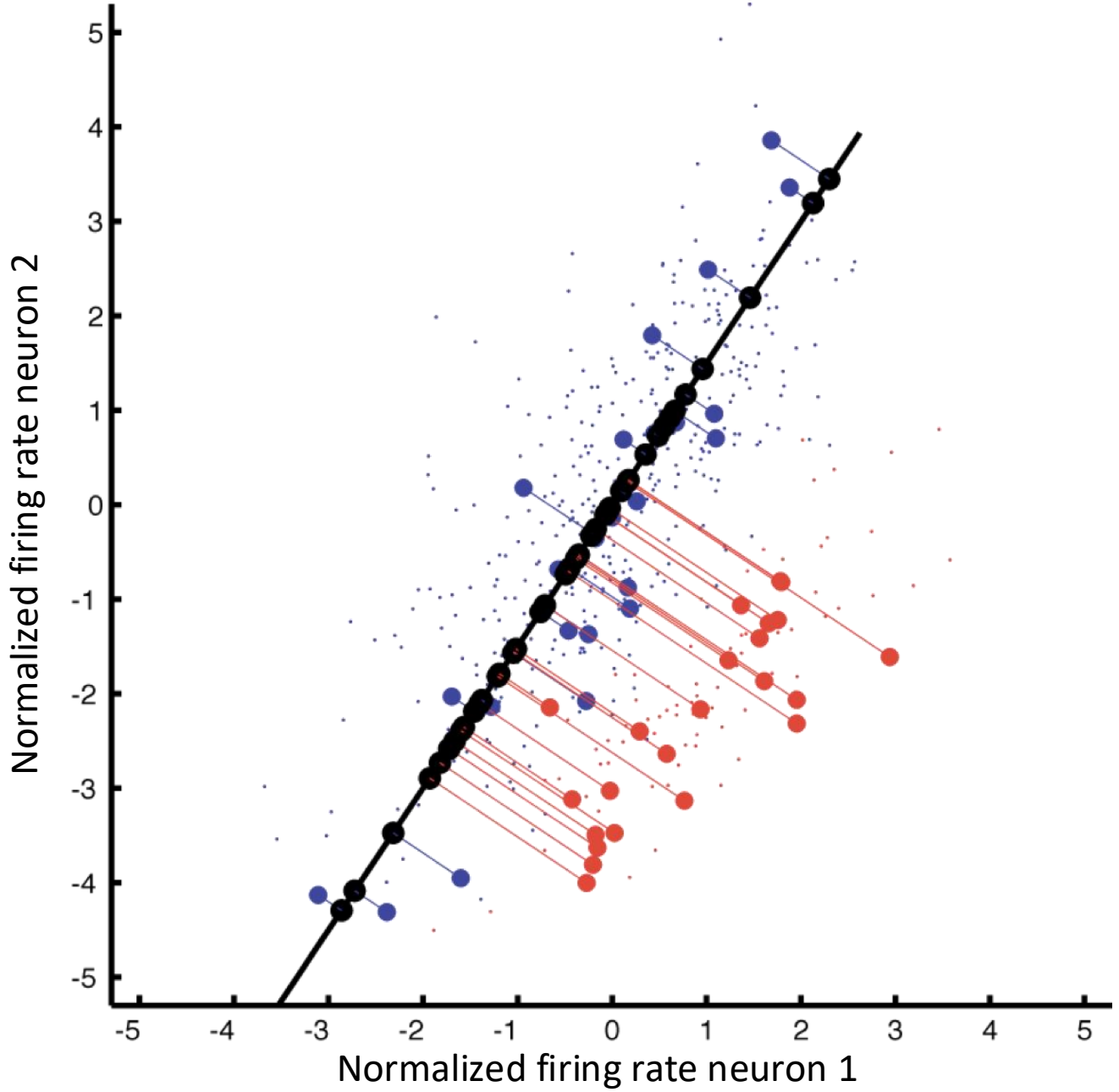


$$w_1 = 0.2$$
$$w_2 = 0.8$$

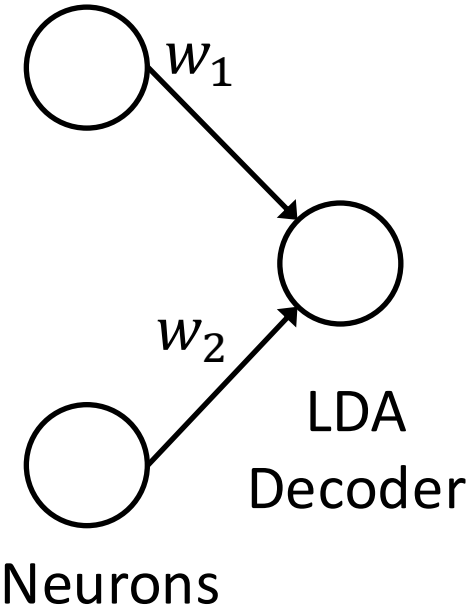


Example Linear Discriminant Analysis on synthetic data

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

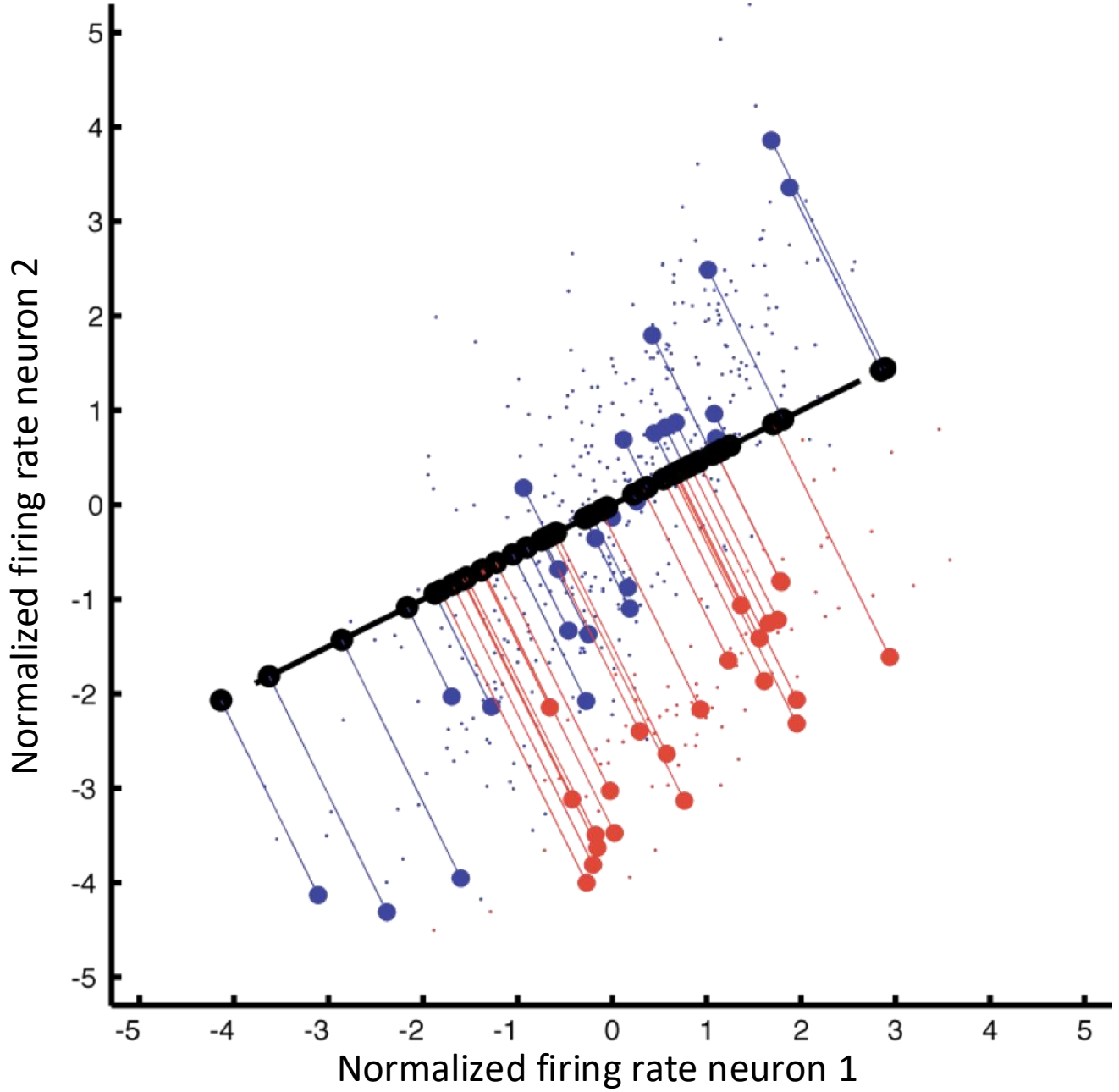


$$w_1 = 0.4$$
$$w_2 = 0.6$$

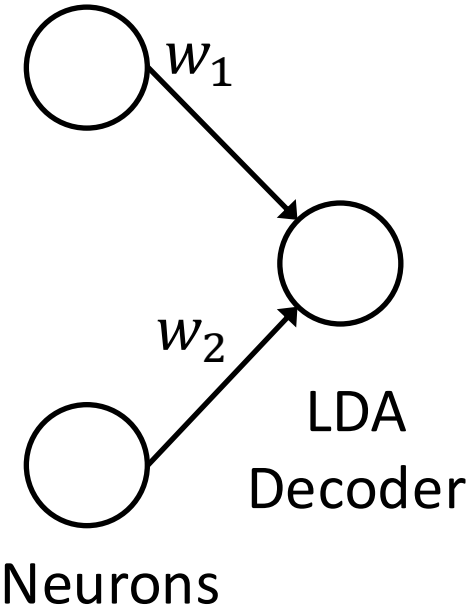


Example Linear Discriminant Analysis on synthetic data

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

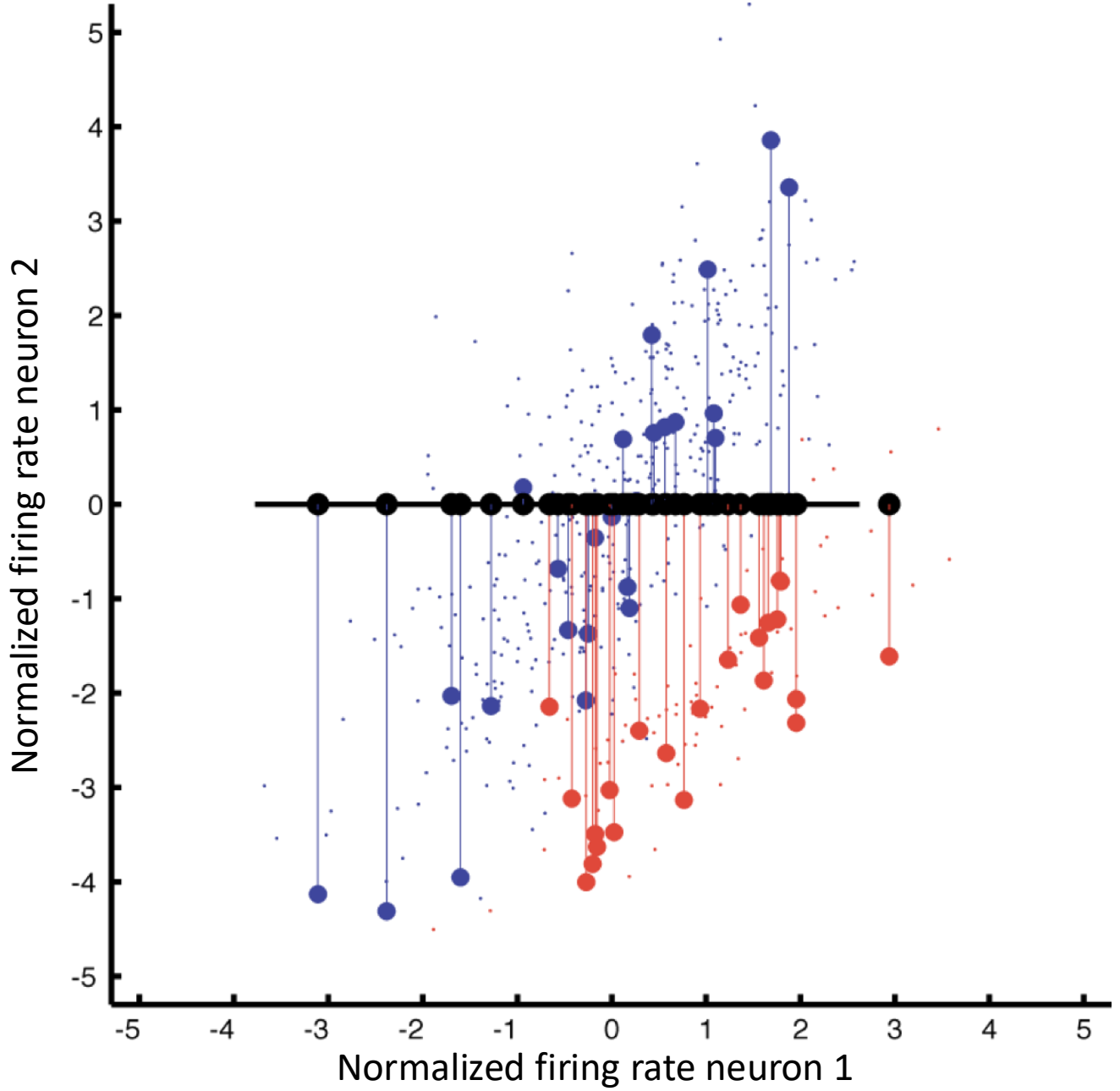


$$w_1 = 0.6$$
$$w_2 = 0.4$$

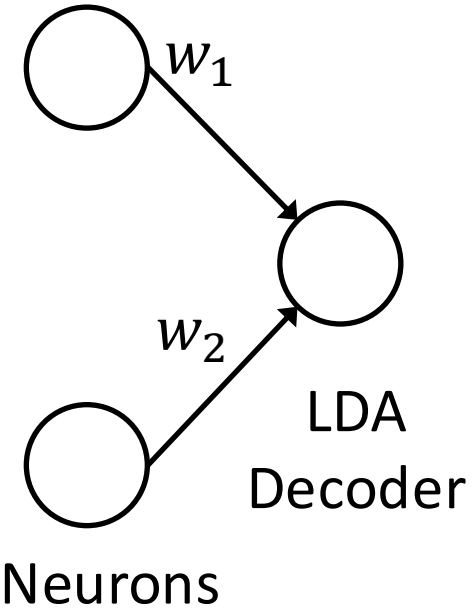


Example Linear Discriminant Analysis on synthetic data

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

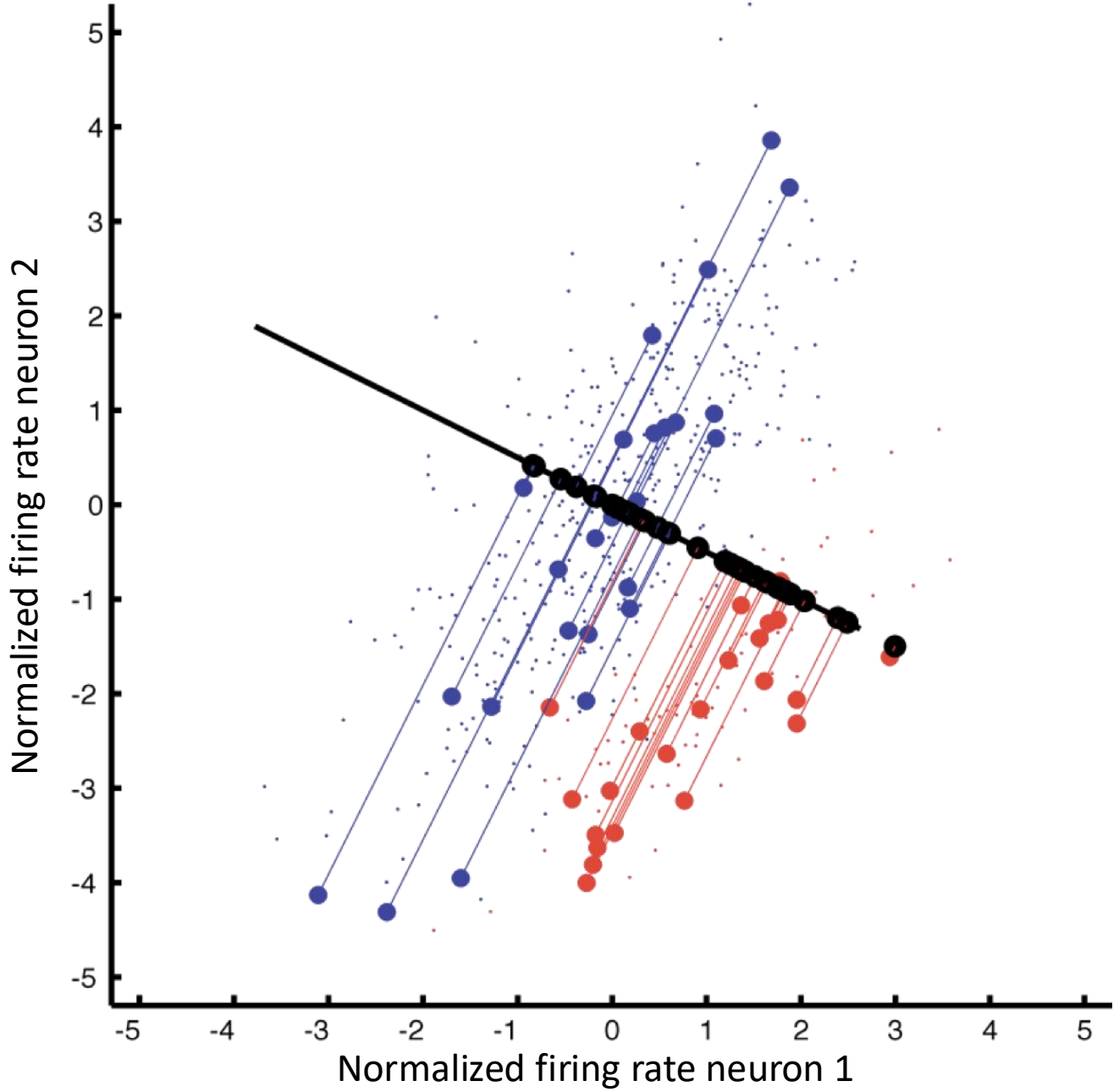


$$w_1 = 1$$
$$w_2 = 0$$

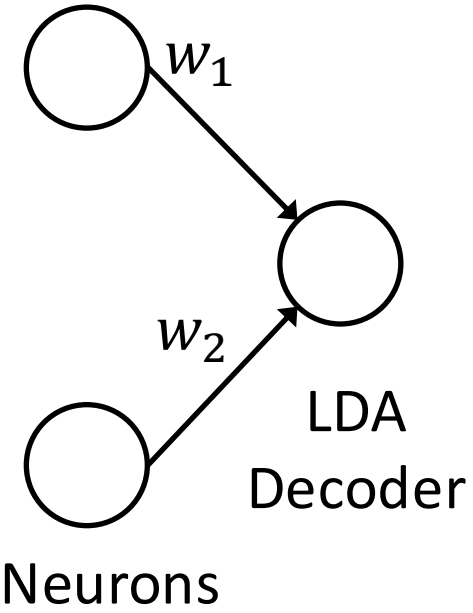


Example Linear Discriminant Analysis on synthetic data

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



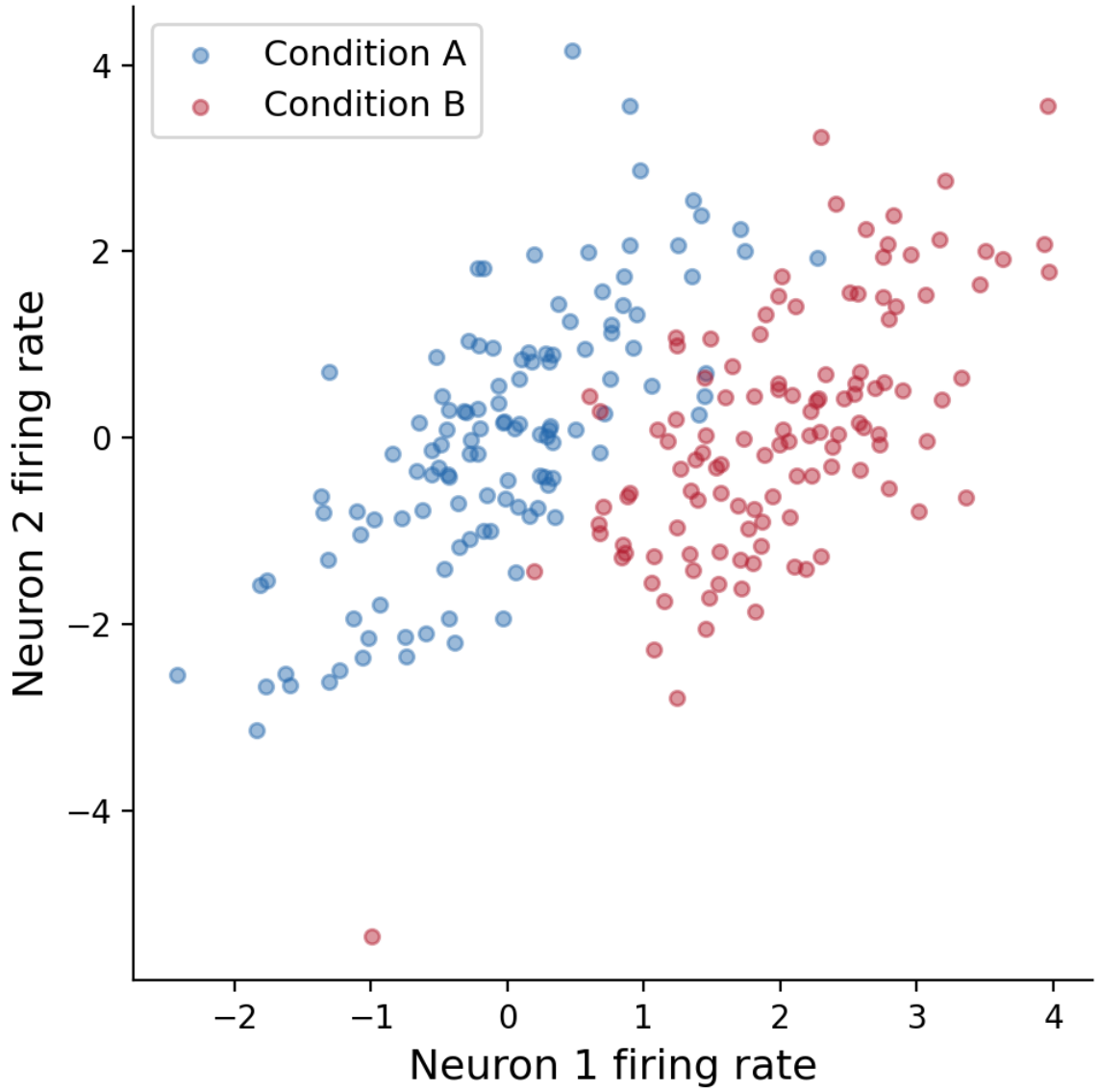
$$w_1 = 0.8$$
$$w_2 = -0.2$$



LDA: which direction to project onto?

Goal: find the set of weights that best separates the two conditions when the data is projected on it.

projection = $w_1 \cdot r_1 + w_2 \cdot r_2 + \ldots + w_n \cdot r_n$



LDA: which direction to project onto?

Goal: find a weighted combination of firing rates

$$\text{projection} = w_1 \cdot r_1 + w_2 \cdot r_2 + \ldots + w_n \cdot r_n$$

that best separates the two conditions.

Two things matter:

1. Large separation between the projected means
   → the classes are far apart on the projection axis

2. Small spread of each class after projection
   → the classes don't overlap much

Maximizing (1) alone isn't enough — you might project onto a direction where the classes are far apart on average, but each class is so spread out that they still overlap.

We want to maximize separation relative to spread.

This is like maximizing d' of the projected values.

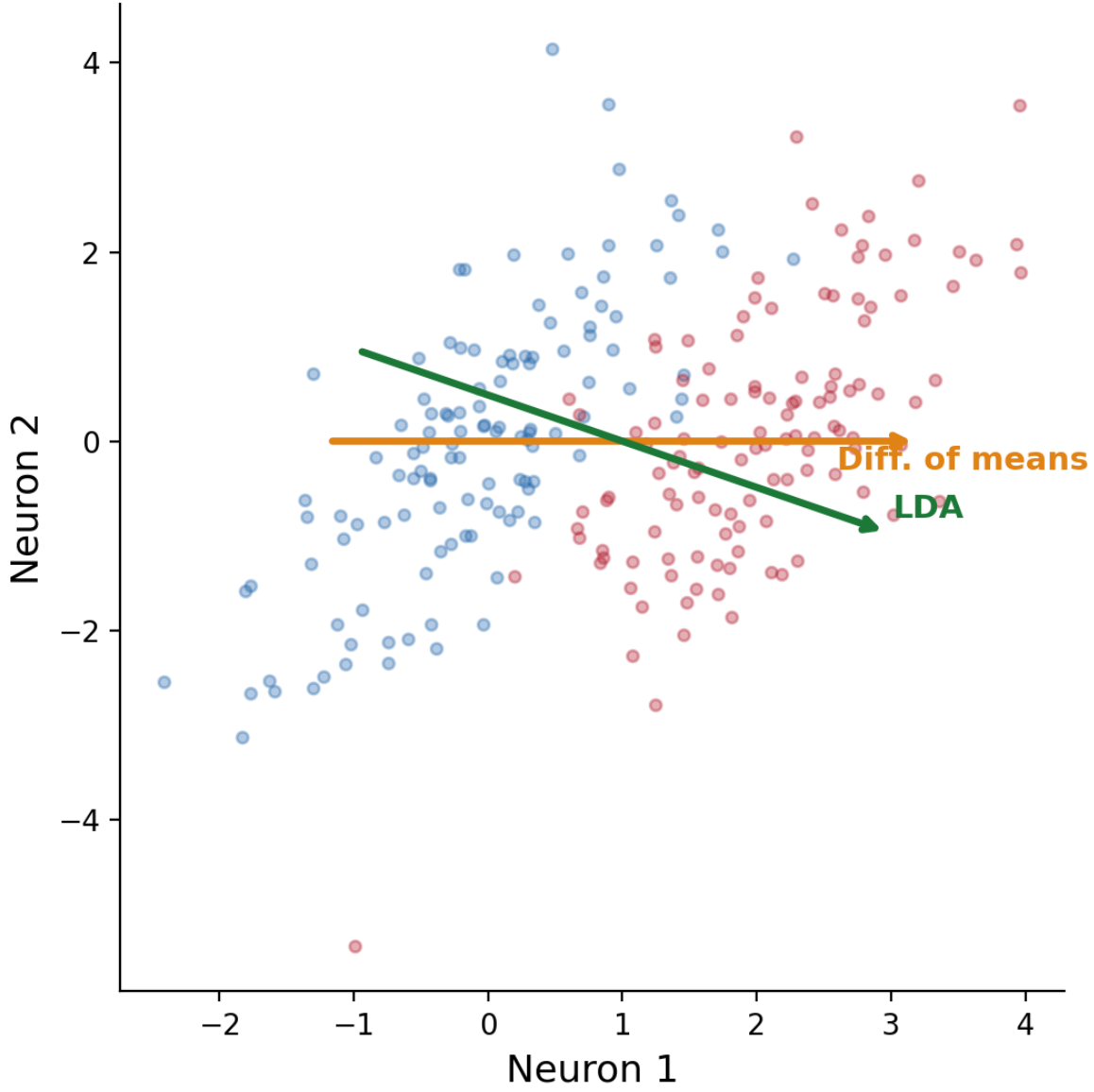
LDA: not just the direction that connects both means

Intuitive guess: project along the line connecting the two class means.

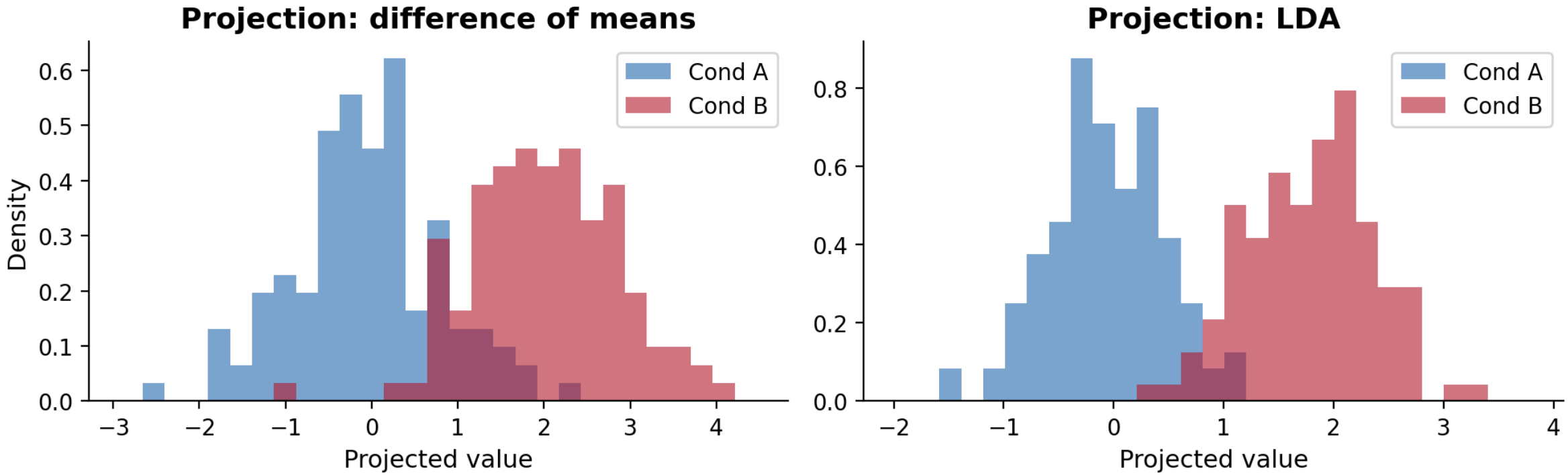
This maximizes the distance between the projected means (the numerator).

But it ignores how spread out each class is after projection (the denominator).

If neurons are correlated, this direction can have a lot of within-class overlap.



LDA: not just the direction that connects both means



LDA finds the direction that maximizes separation relative to spread — even when it's not the line connecting the means.

Fisher criterion

$$J(\mathbf{w}) = \frac{(\mathbf{w}^T\mu_1 - \mathbf{w}^T\mu_2)^2}{\mathbf{w}^T\Sigma_{\text{pooled}}\,\mathbf{w}}$$

= (separation of projected means)² / (projected within-class variance)

This is the ratio we've been building toward:

- Numerator: squared distance between projected class means
- Denominator: pooled within-class variance after projection

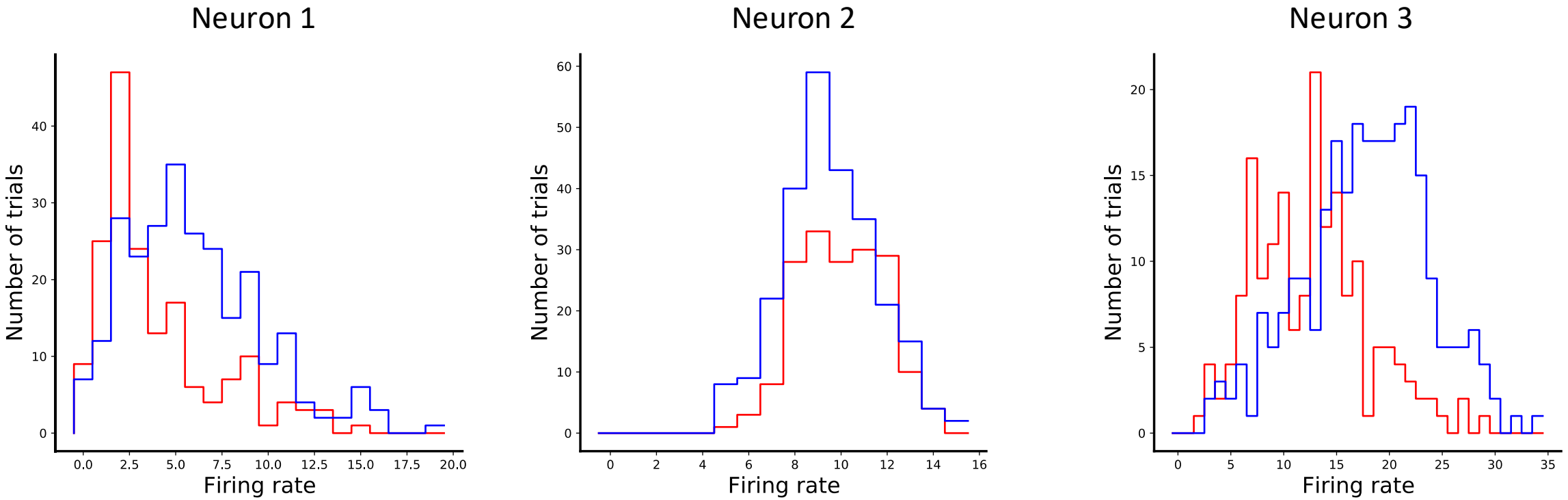
Maximizing J(w) is equivalent to maximizing d' of the projection.

The direction w that maximizes J(w) is the LDA solution.

Skip derivation

Linear Discriminant Analysis

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

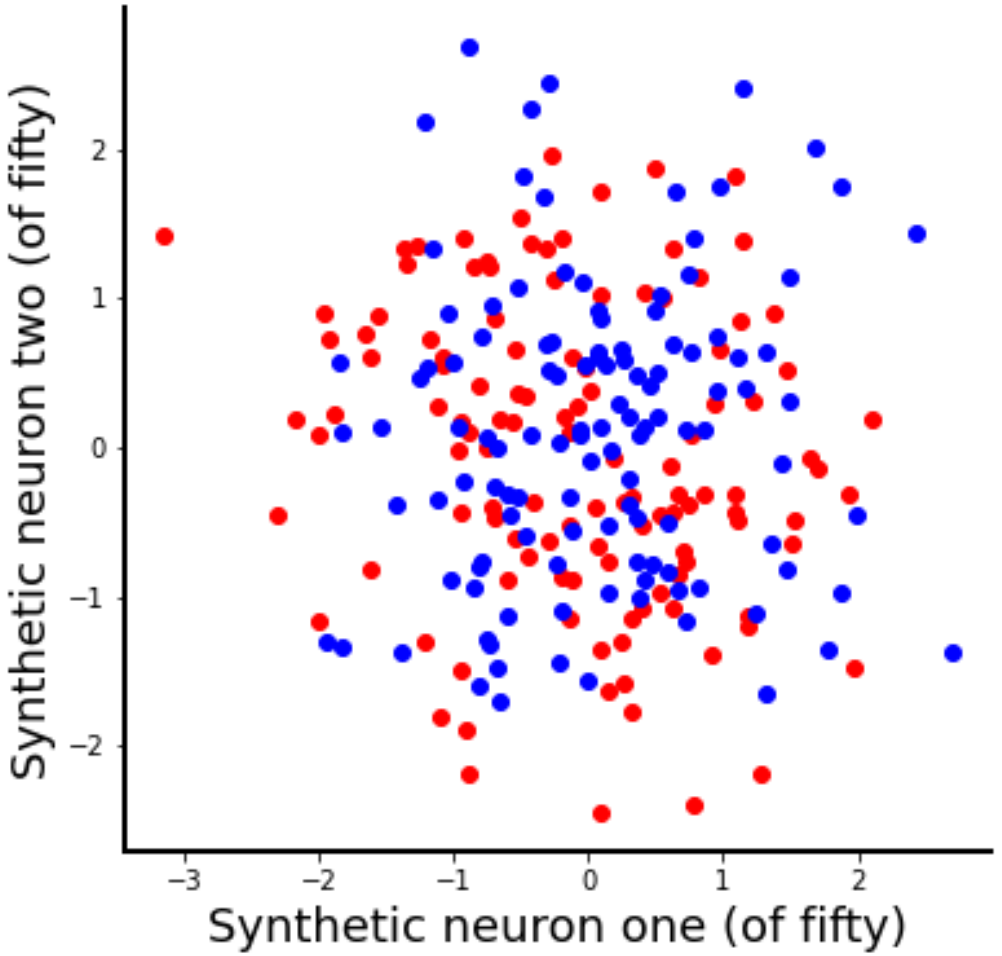


Solution to optimization given by top eigenvectors of matrix: $S_W^{-1} S_B$

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

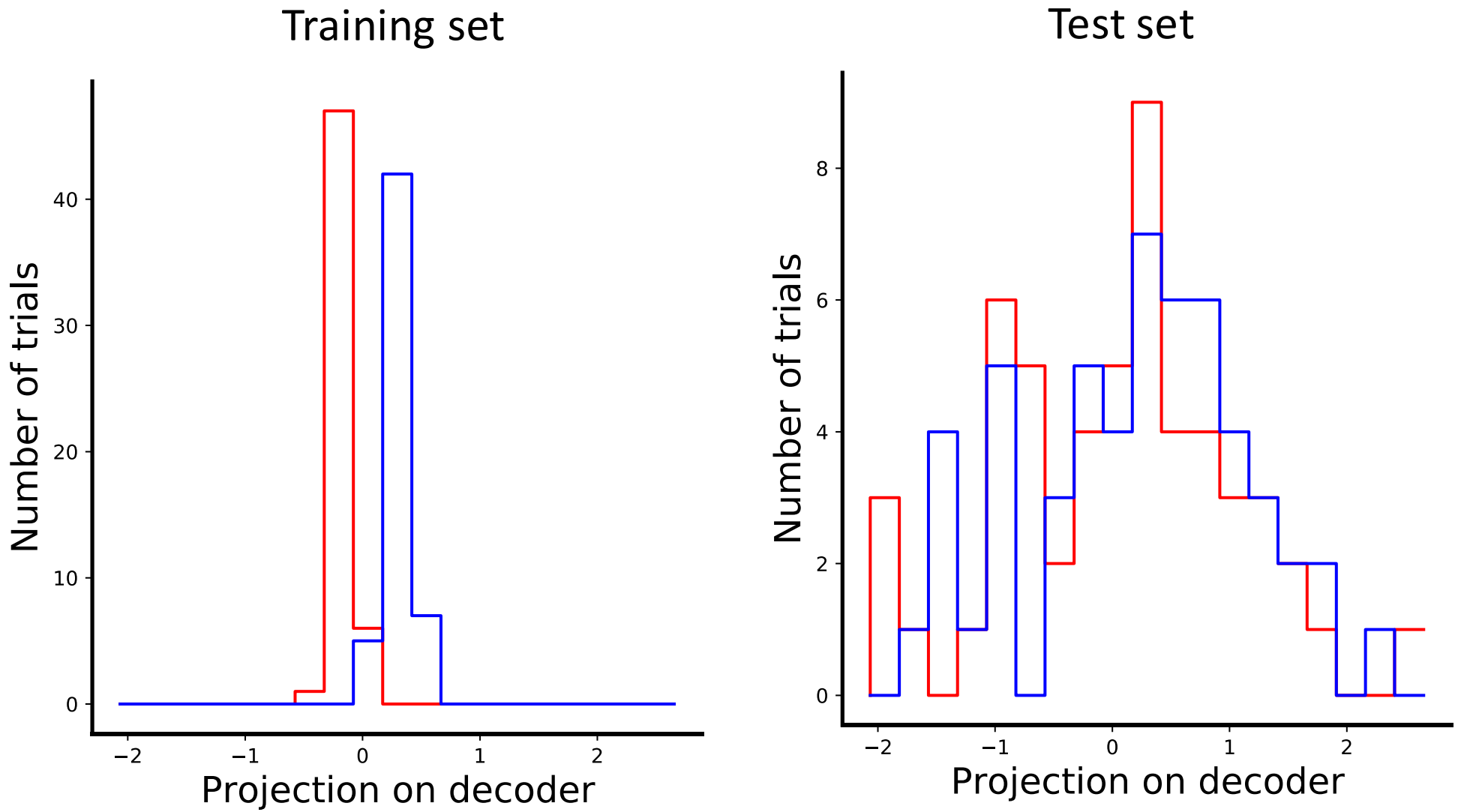
S_W is the within group variance matrix, S_B is the between group variance matrix

Overfitting and separation of training and test datasets



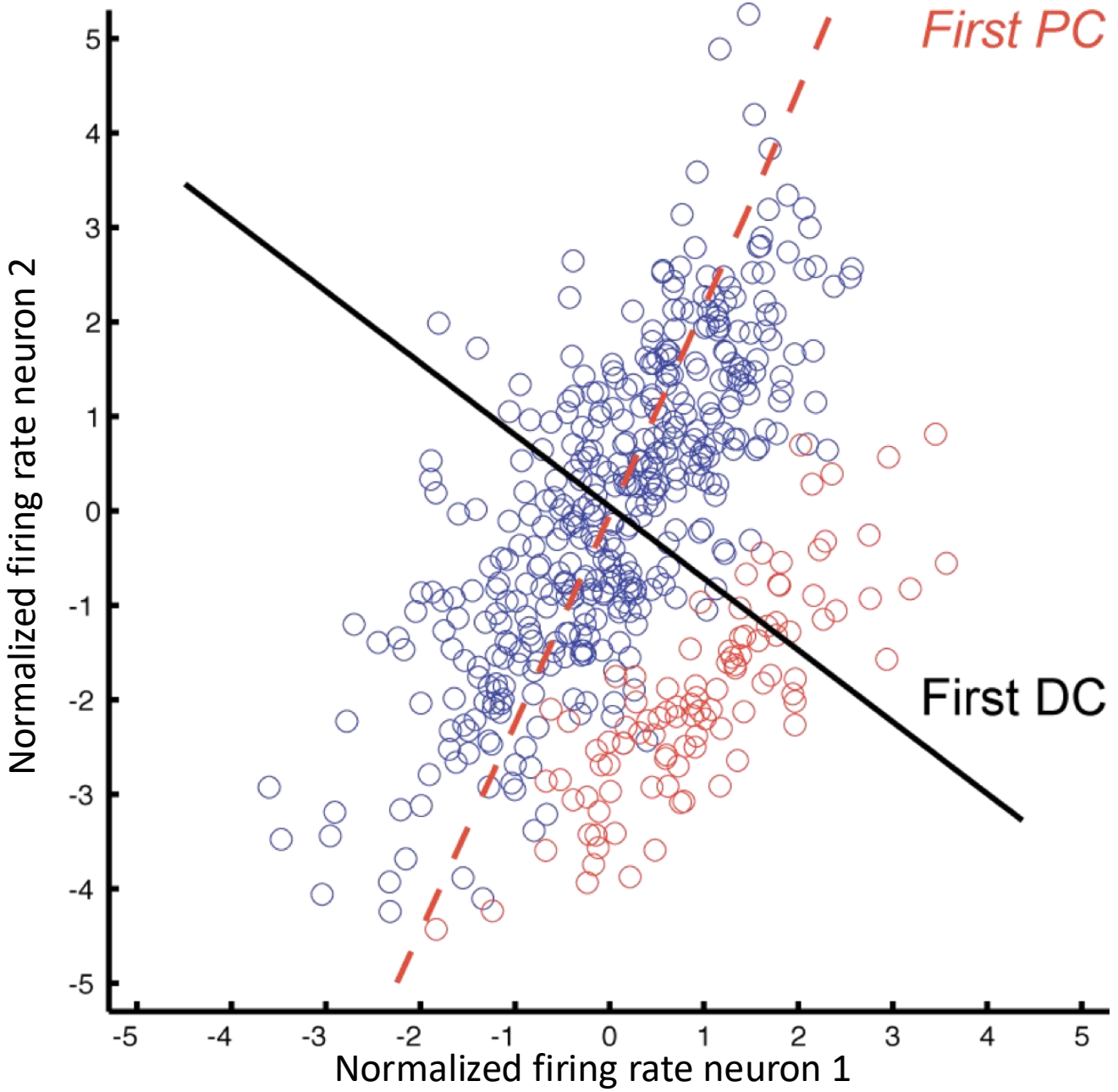
Samples from synthetic dataset generated randomly with the same mean

Overfitting and separation of training and test datasets

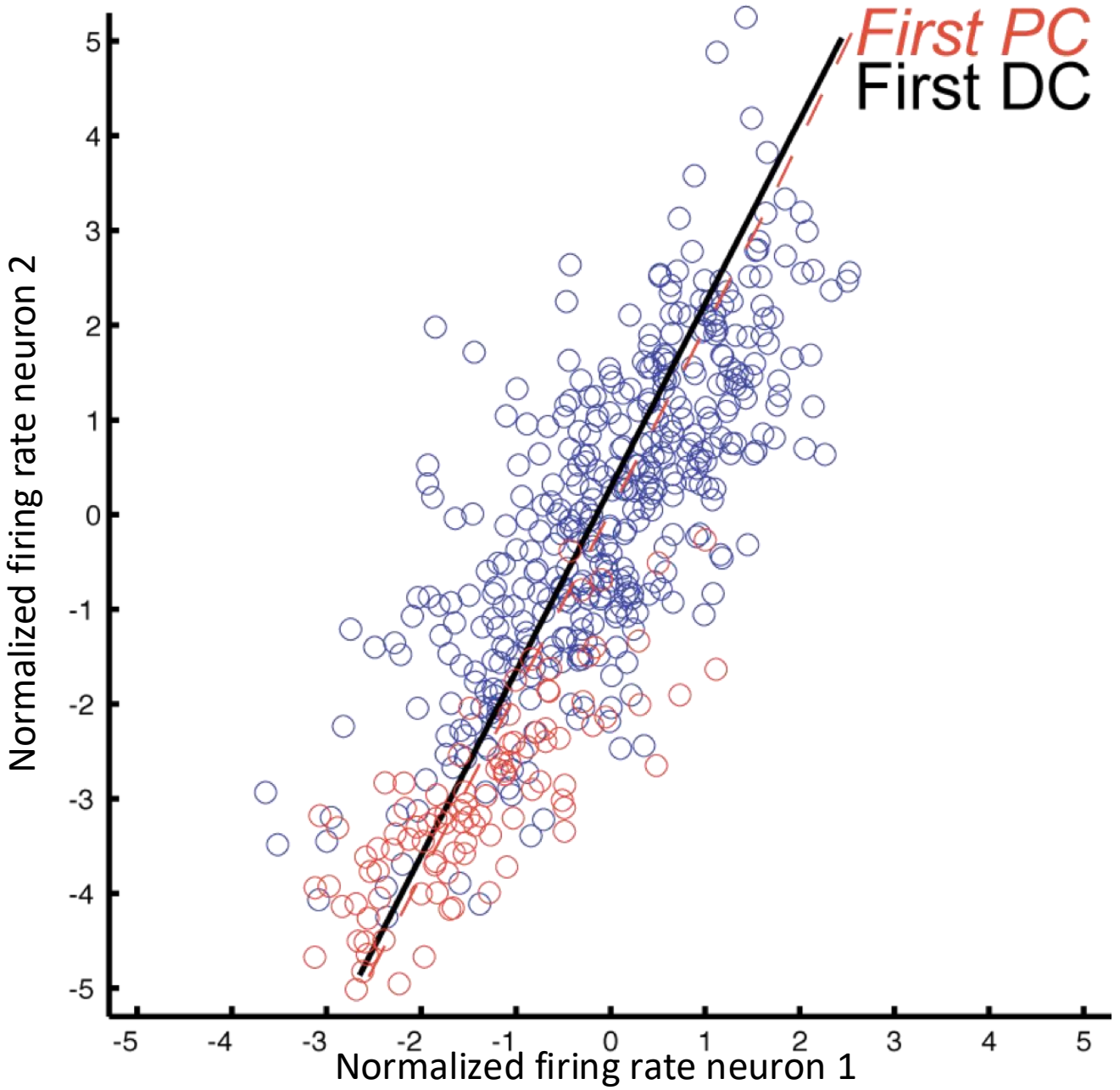


Over fitting!

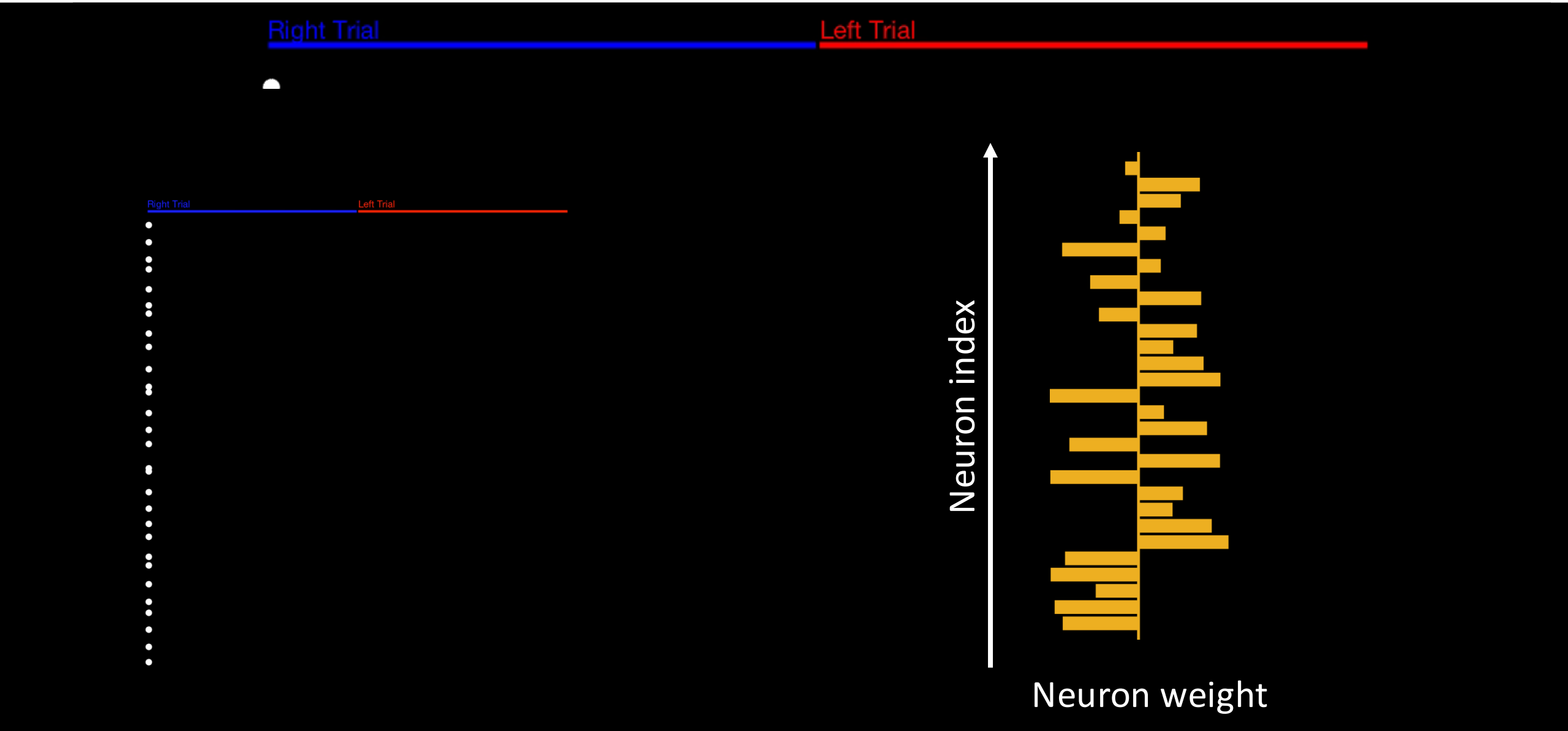
LDA


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

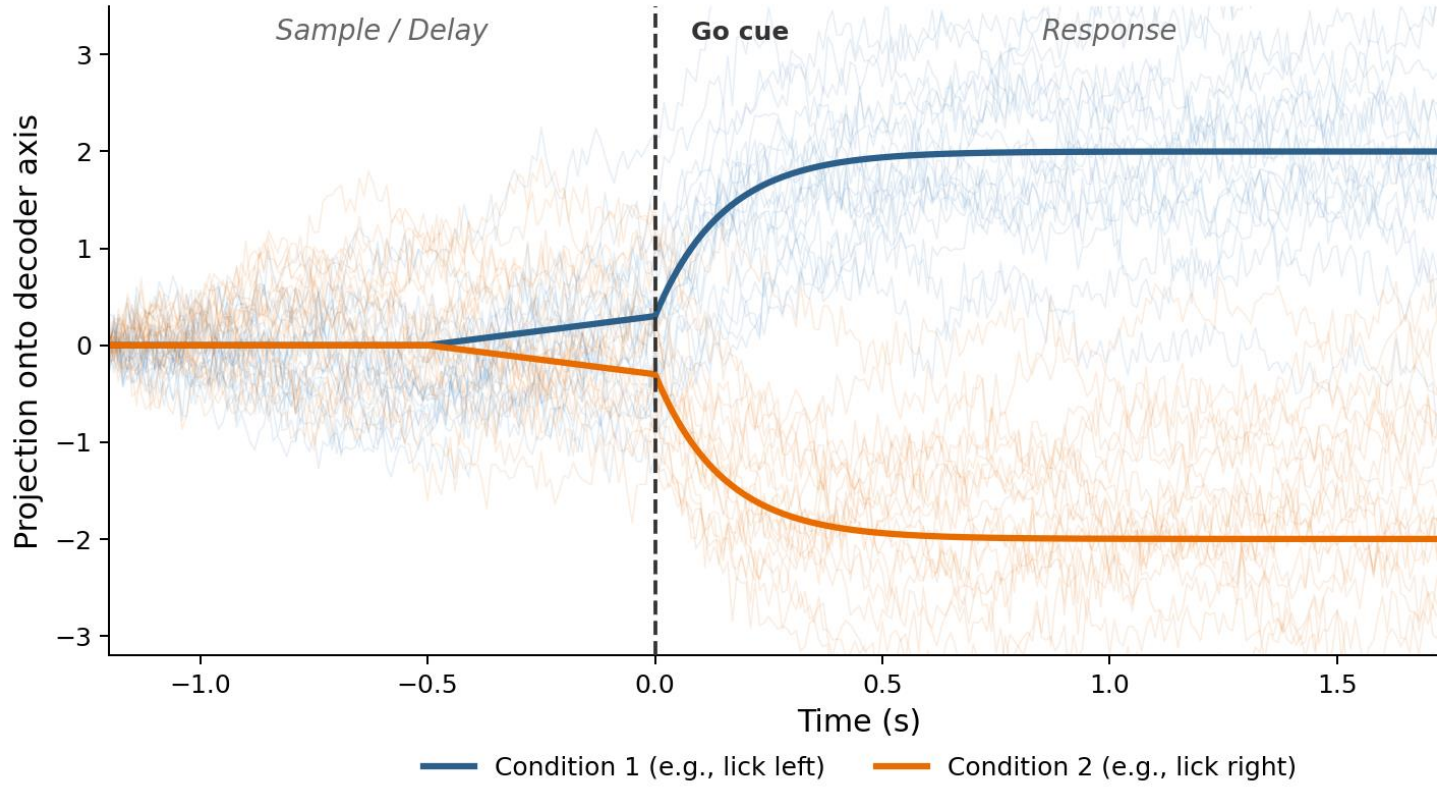
LDA


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



Decoder is static, but one can analyze projection over time



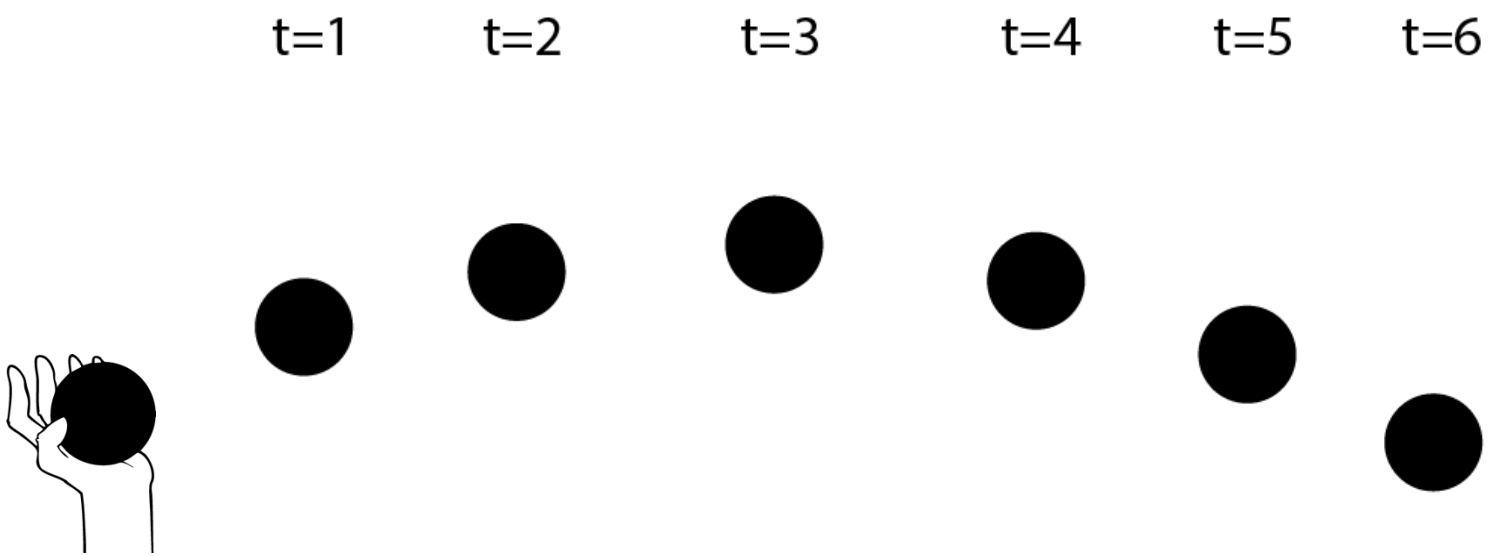
The decoder weights are fit at one time point (e.g., during the delay period).

But we can project the neural activity at every time point onto this same decoder axis.

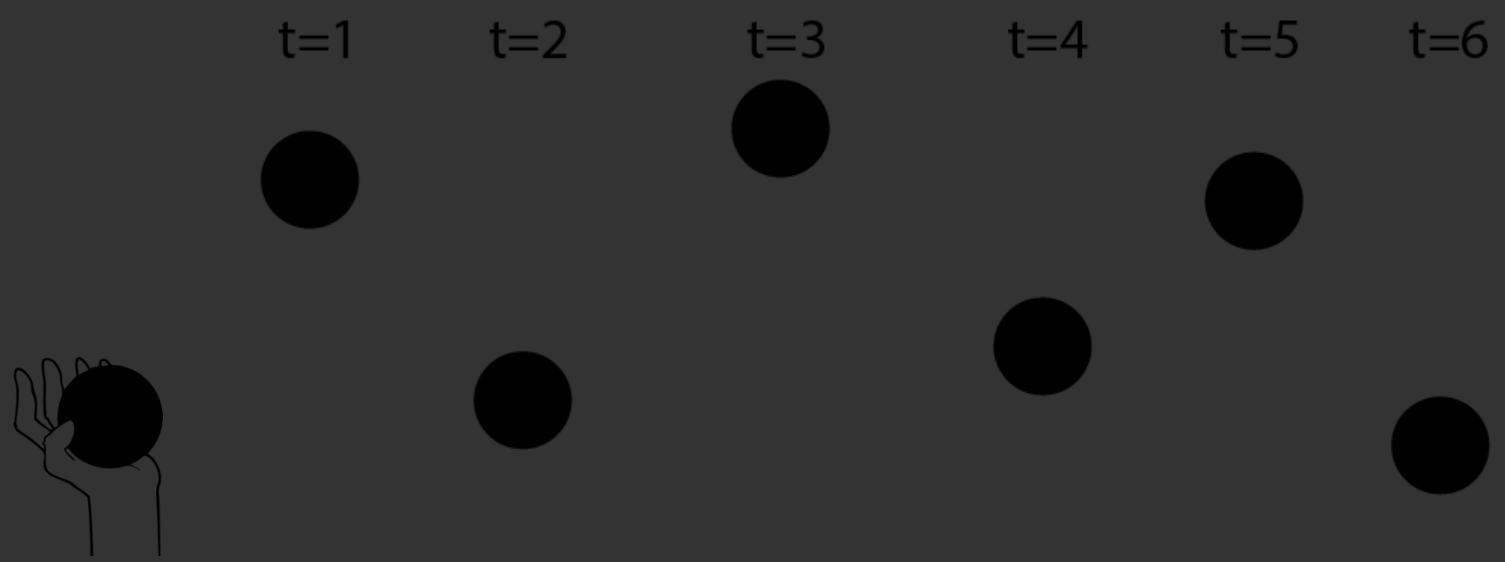
This reveals: when does the population start representing the upcoming choice?

Switching gears: analyzing structure in population recordings

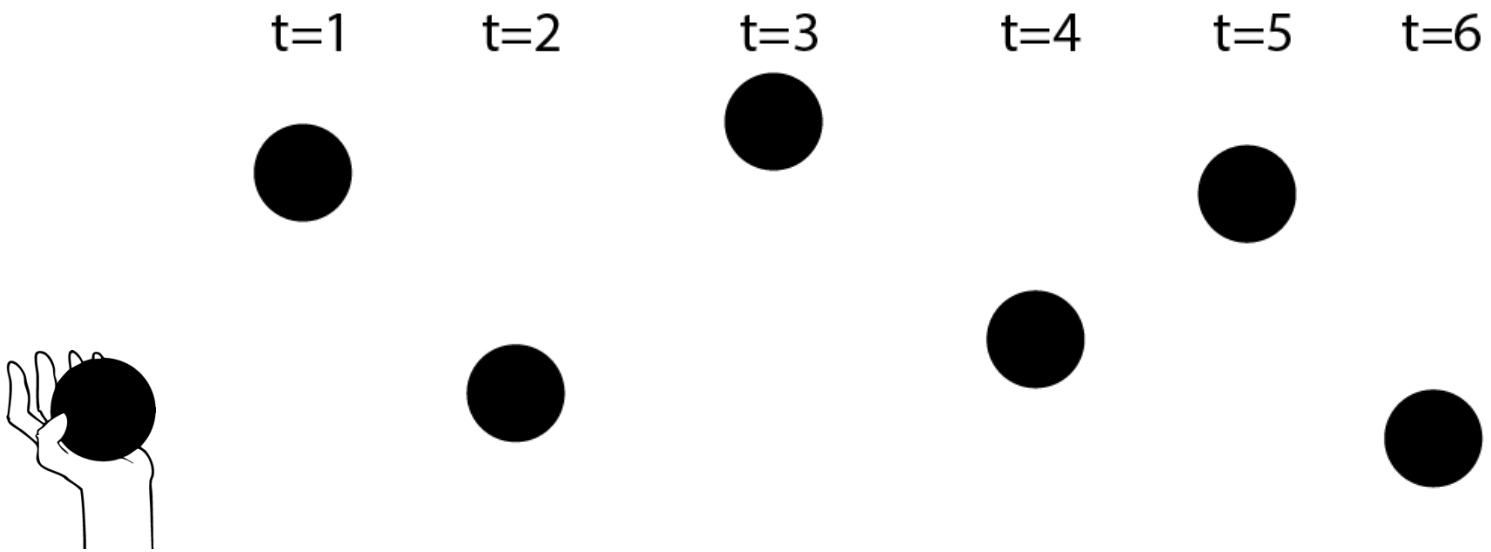
Modeling structure in population recordings: why do it?



Modeling structure in population recordings: why do it?

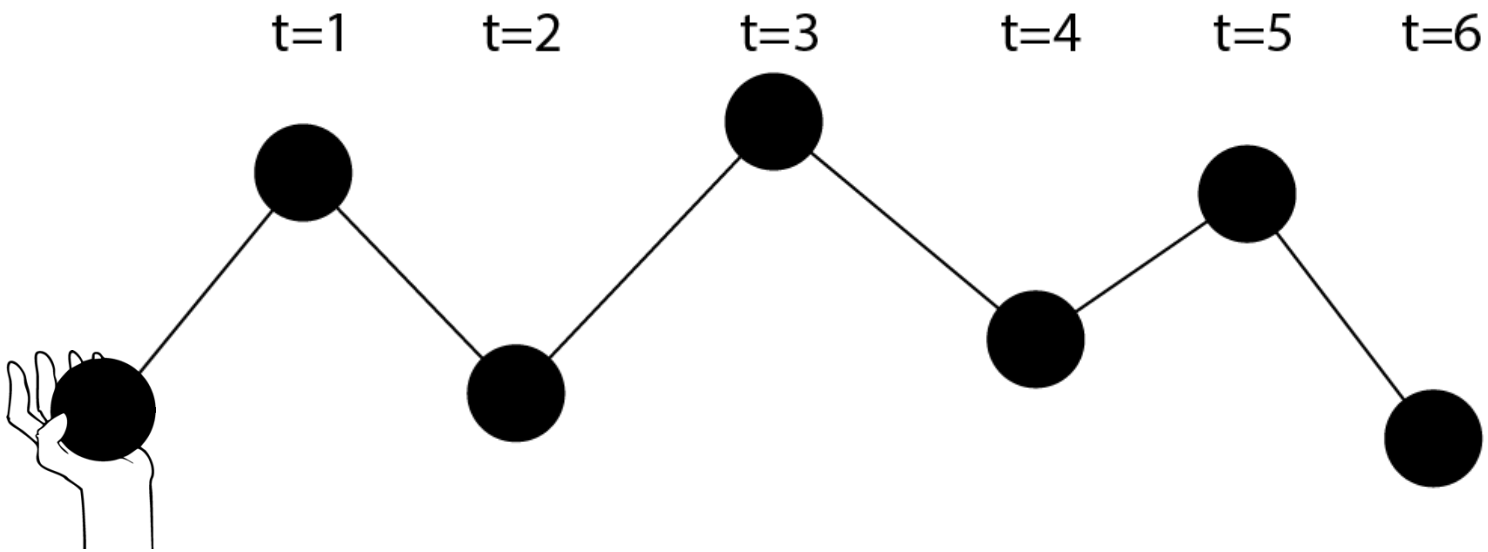


Modeling structure in population recordings: why do it?



Modeling structure in population recordings: why do it?

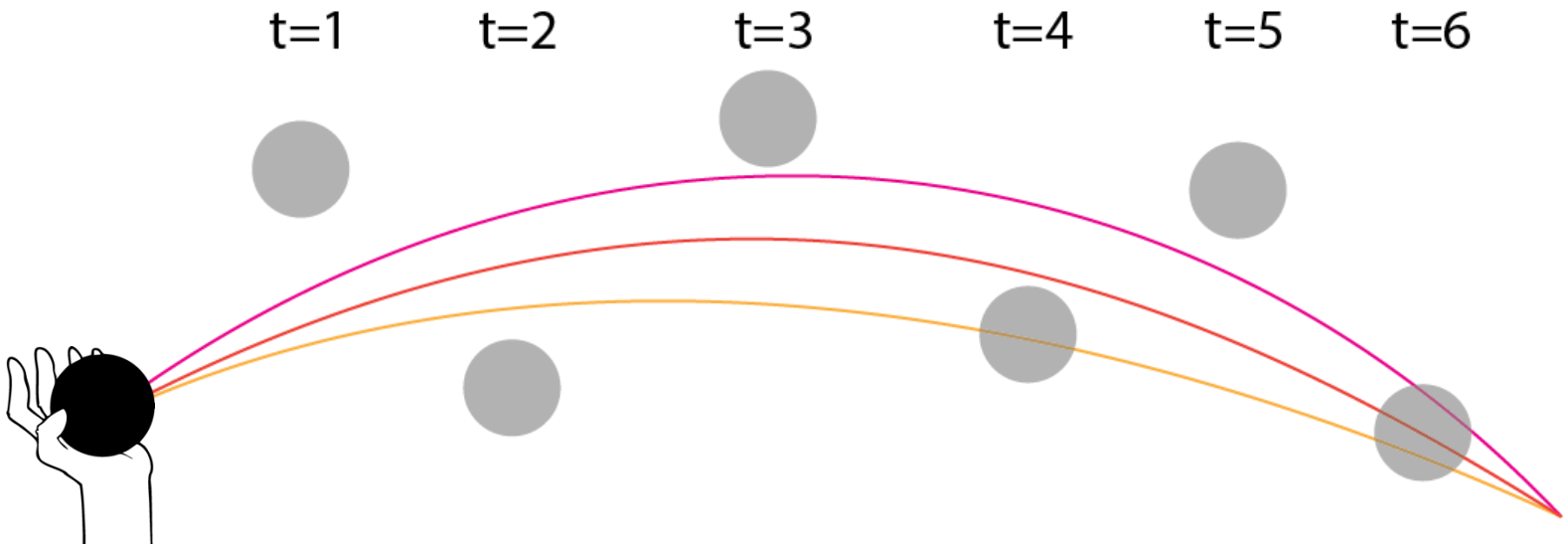
Trajectory very unlikely given our understanding of the system!



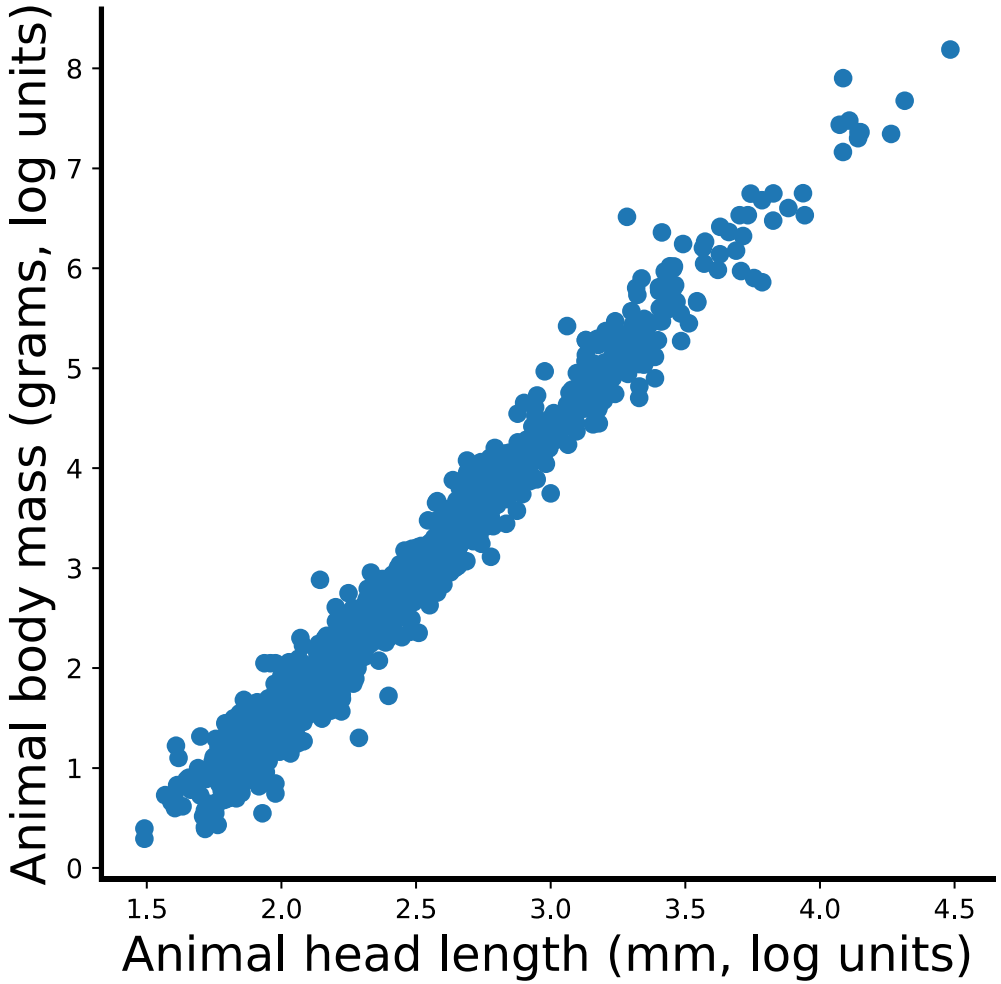
Modeling structure in population recordings: why do it?

Trajectory very unlikely given our understanding of the system!

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

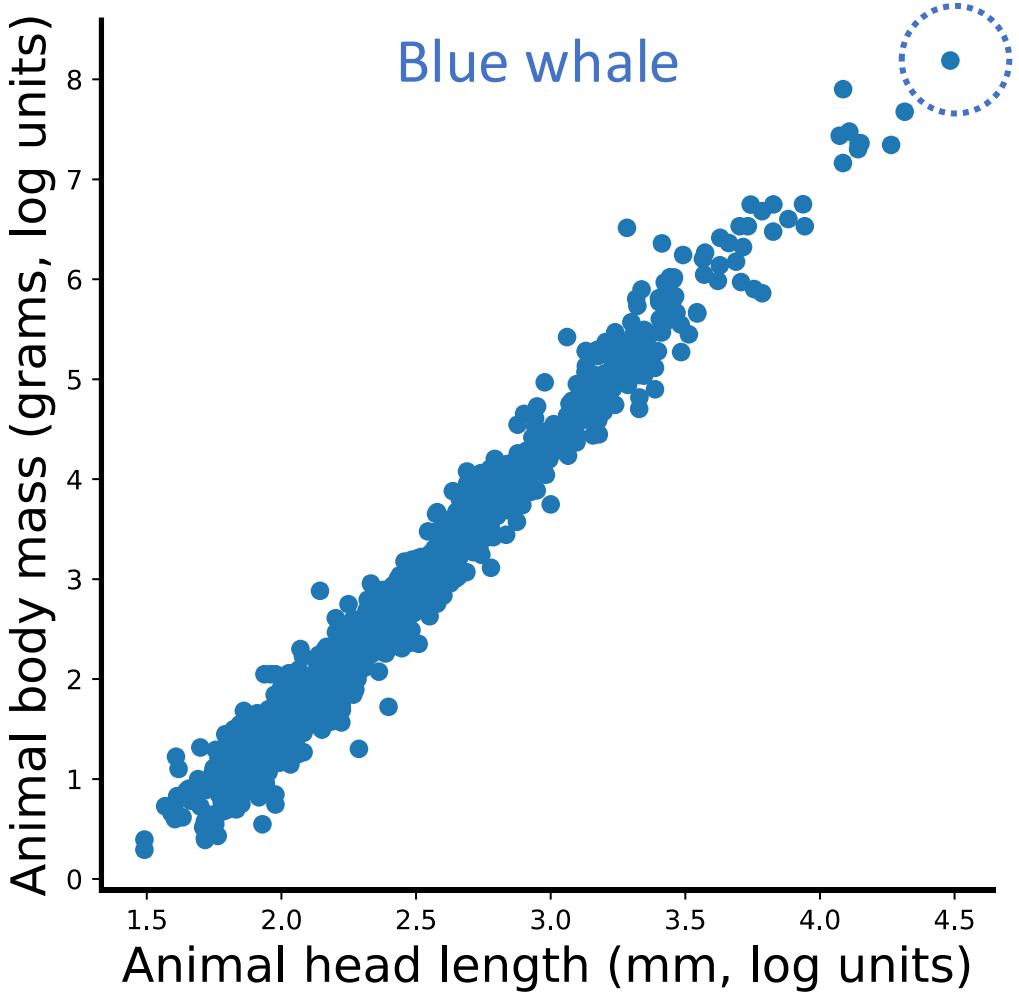


Example: animal properties



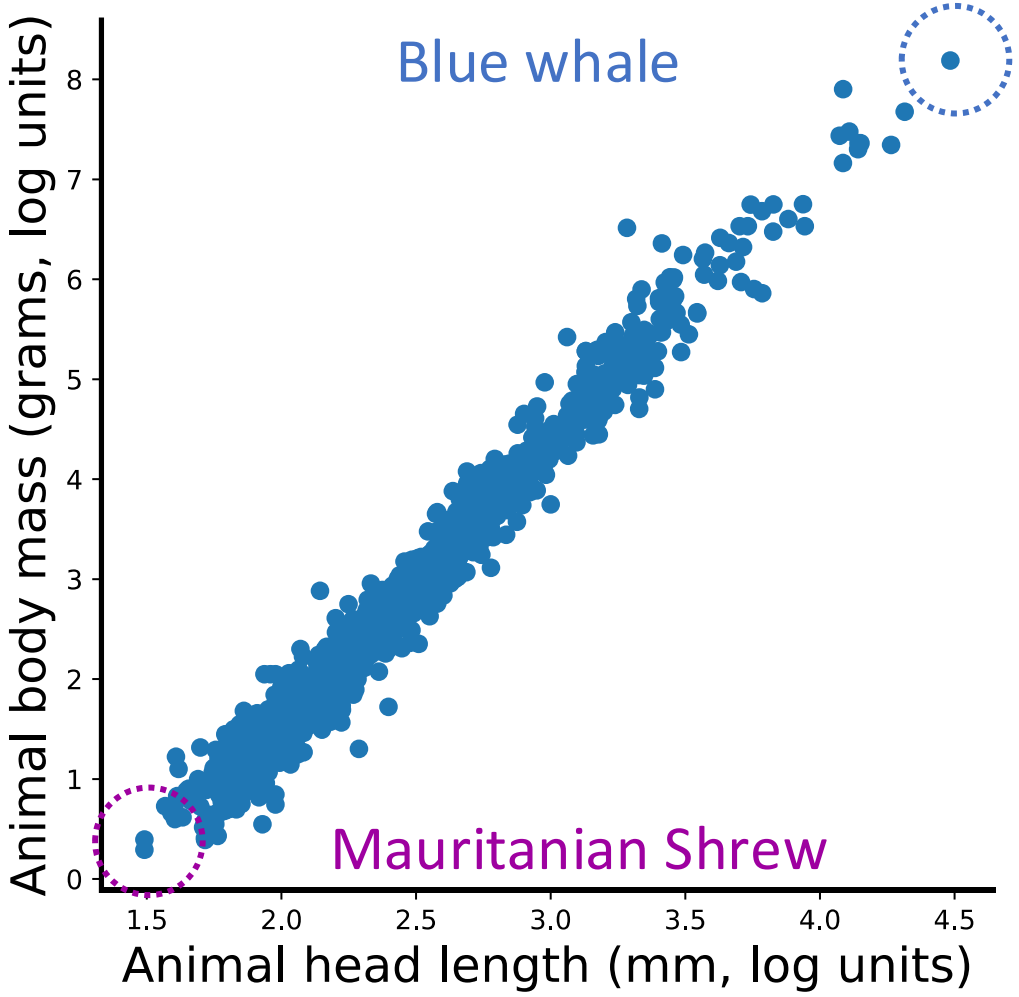
PanTHERIA dataset

Example: animal properties



PanTHERIA dataset

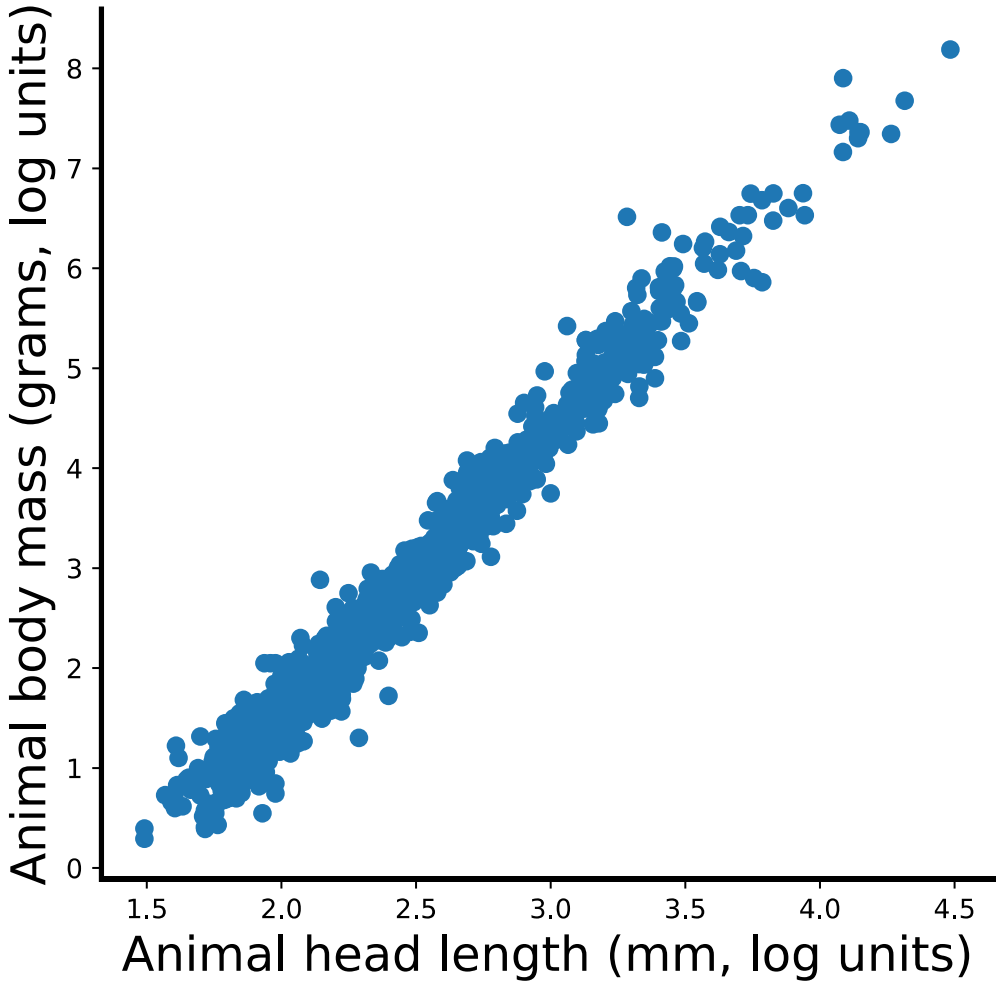
Example: animal properties



PanTHERIA dataset

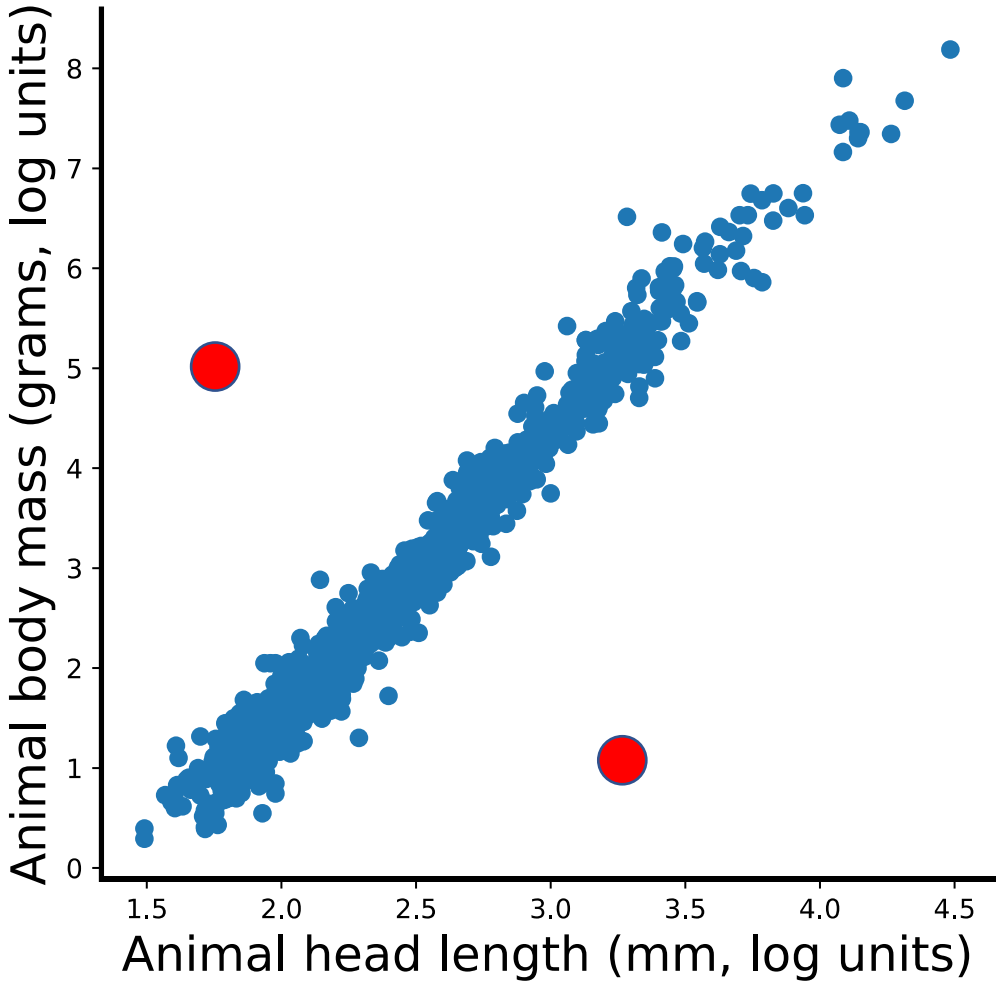
Factor analysis

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



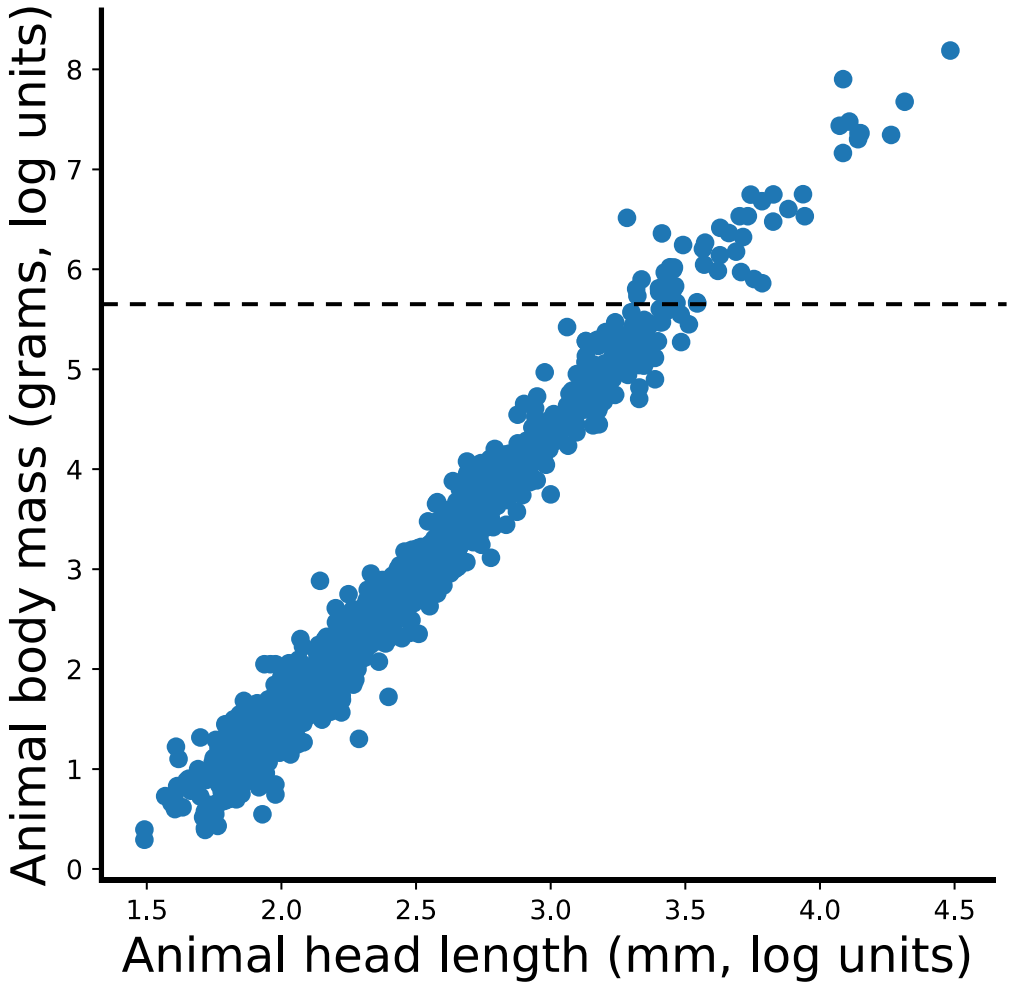
Factor analysis

This will allow us to explain why the two points in red are unlikely



Factor analysis

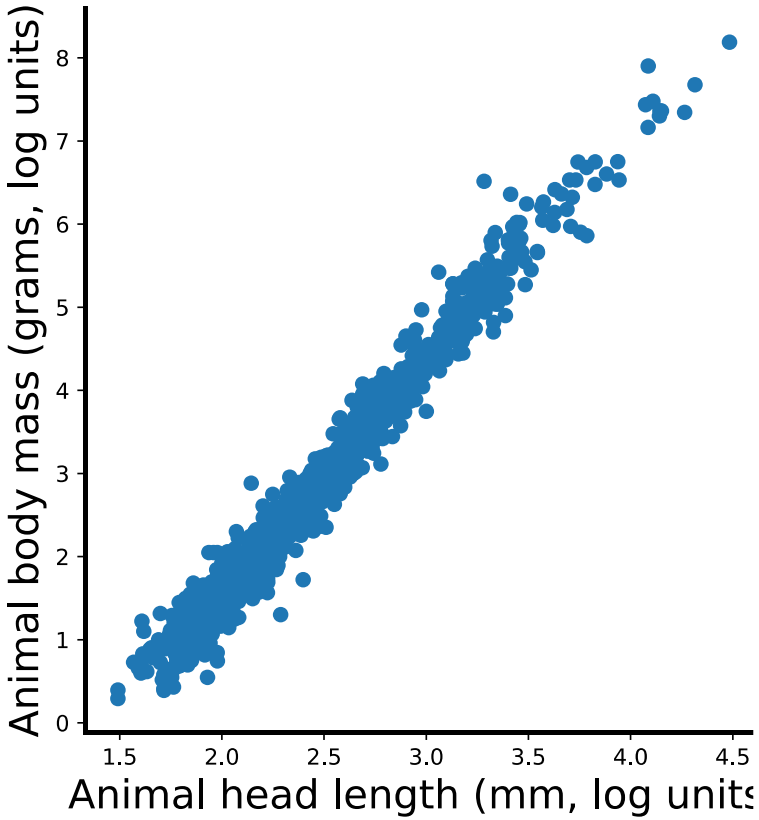
Allows us to guess missing values (impute)



Latent variable models as data recipes

Data recipe:

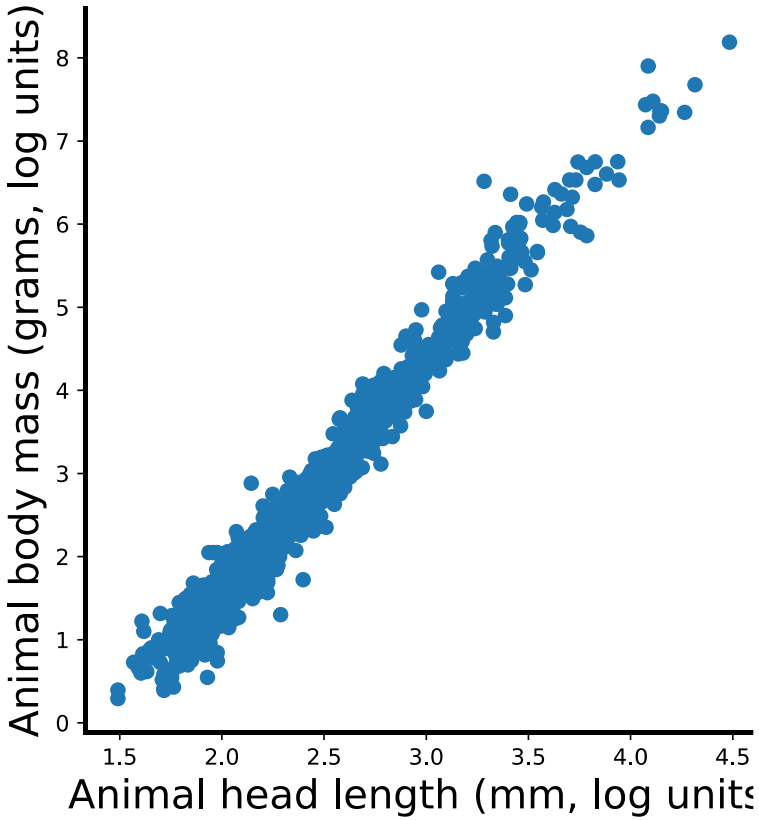
Generate random 1D Gaussian

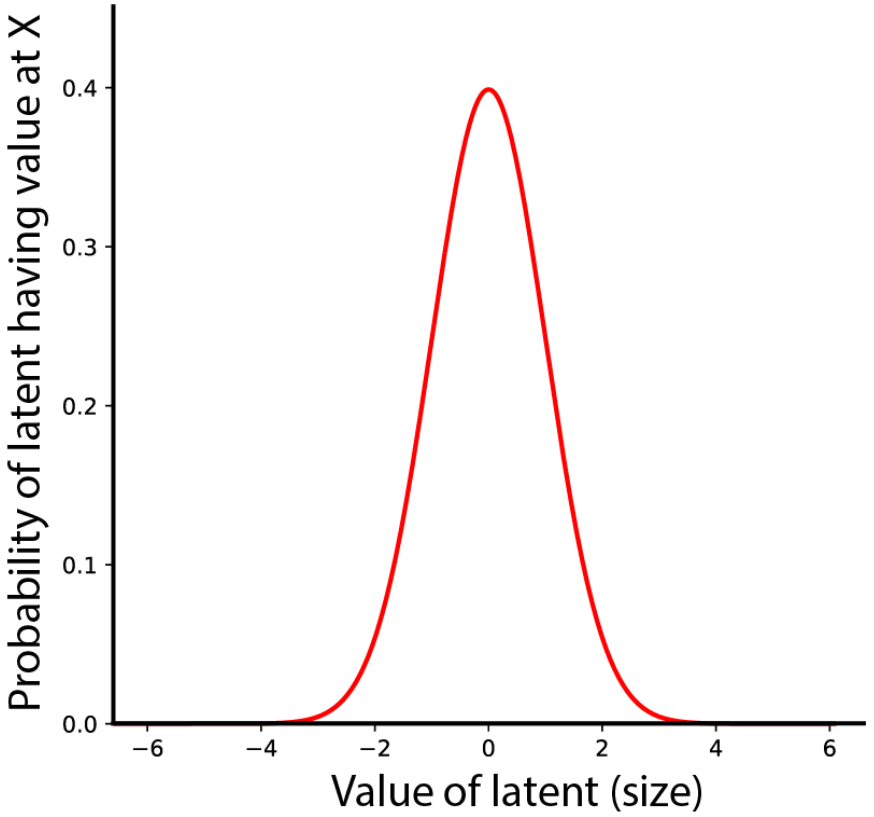


Latent variable models as data recipes

Data recipe:

Generate random 1D Gaussian for latent



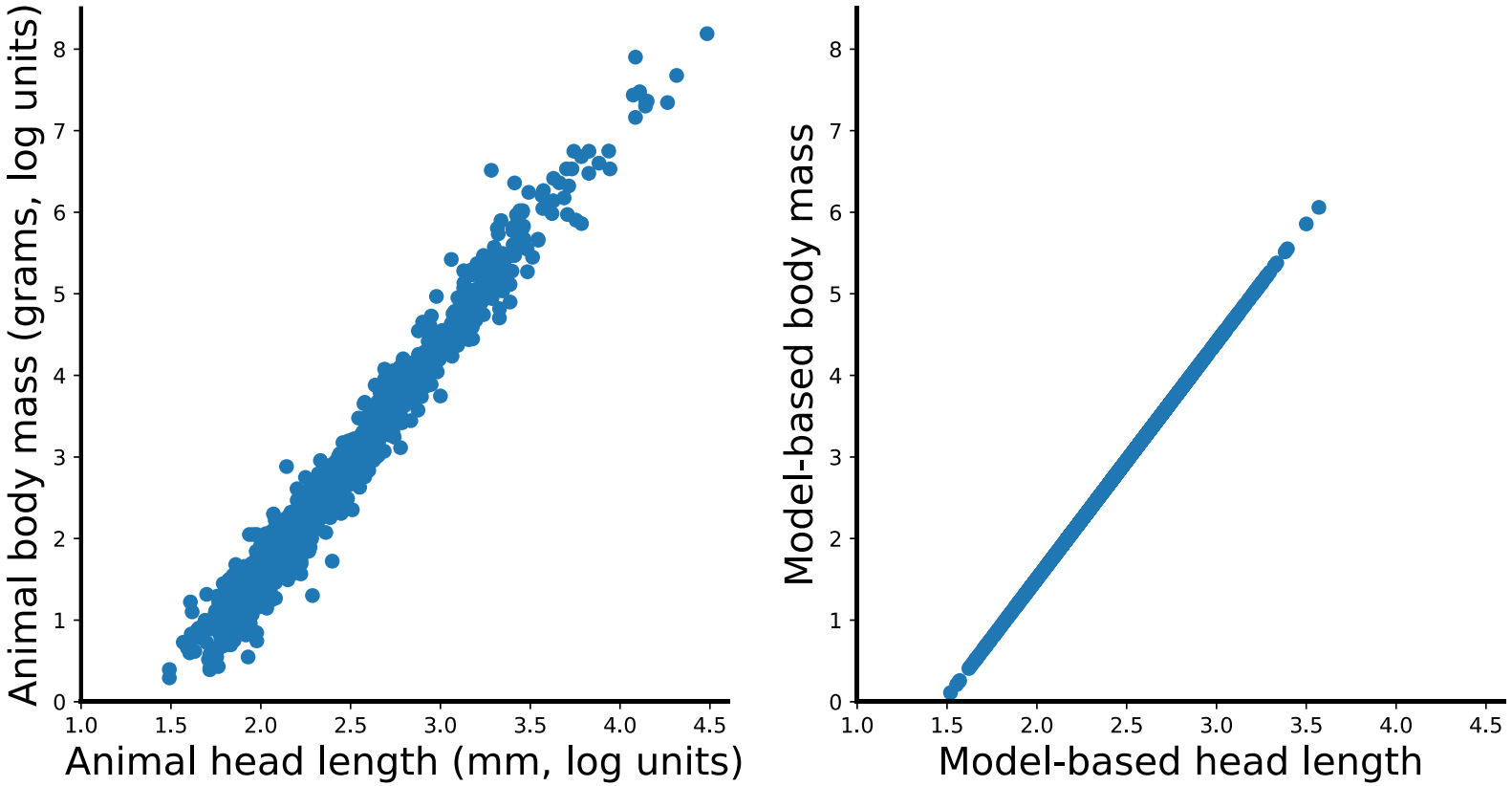


Latent variable models as data recipes

Data recipe:

Generate random 1D Gaussian for latent

Multiply by “Size” factor



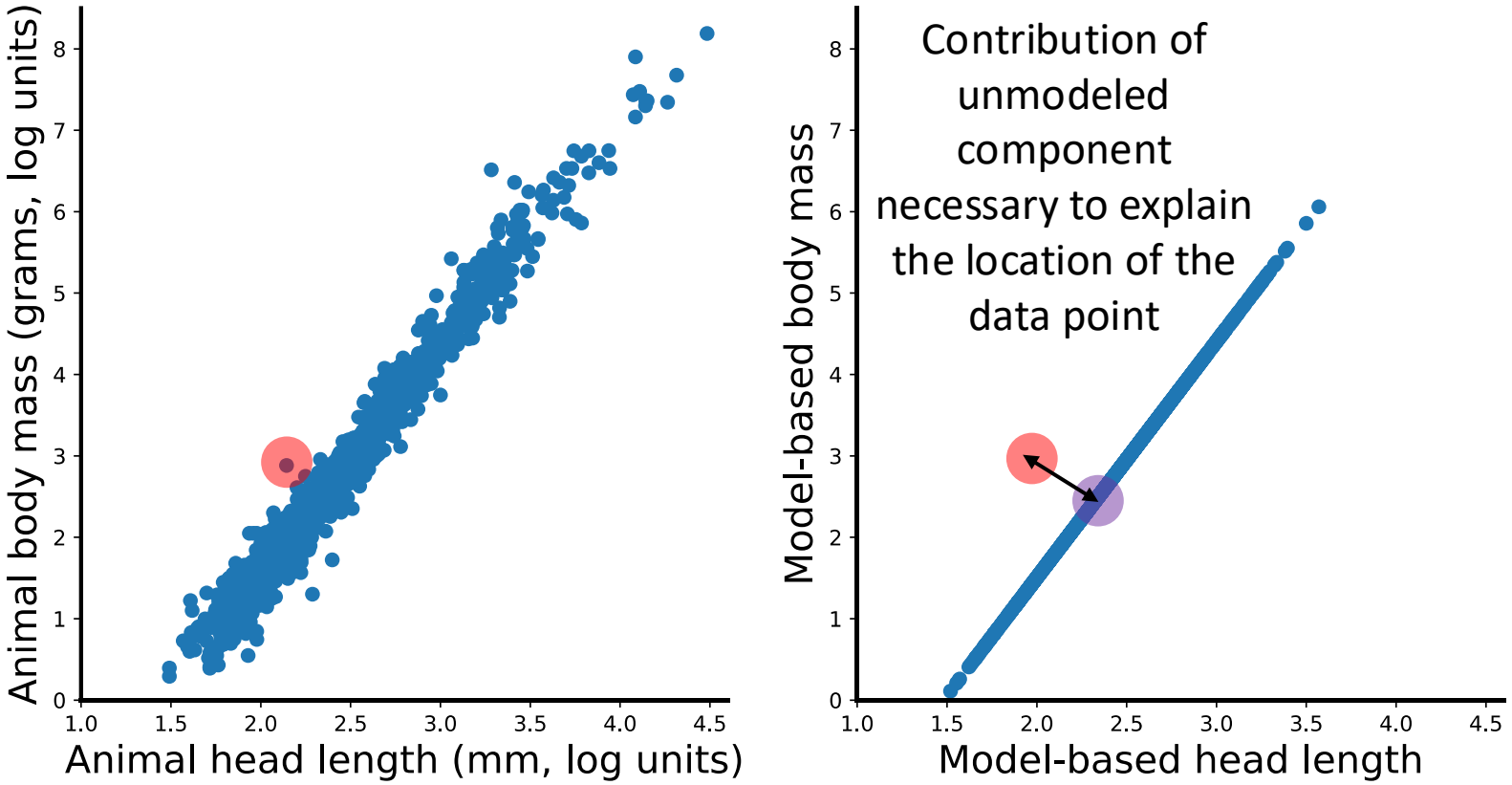
Unmodeled component contribution explains deviation of data from factor values

Data recipe:

Start with random 1D Gaussian

Multiply by “Size” factor

Add independent unmodeled component to each point



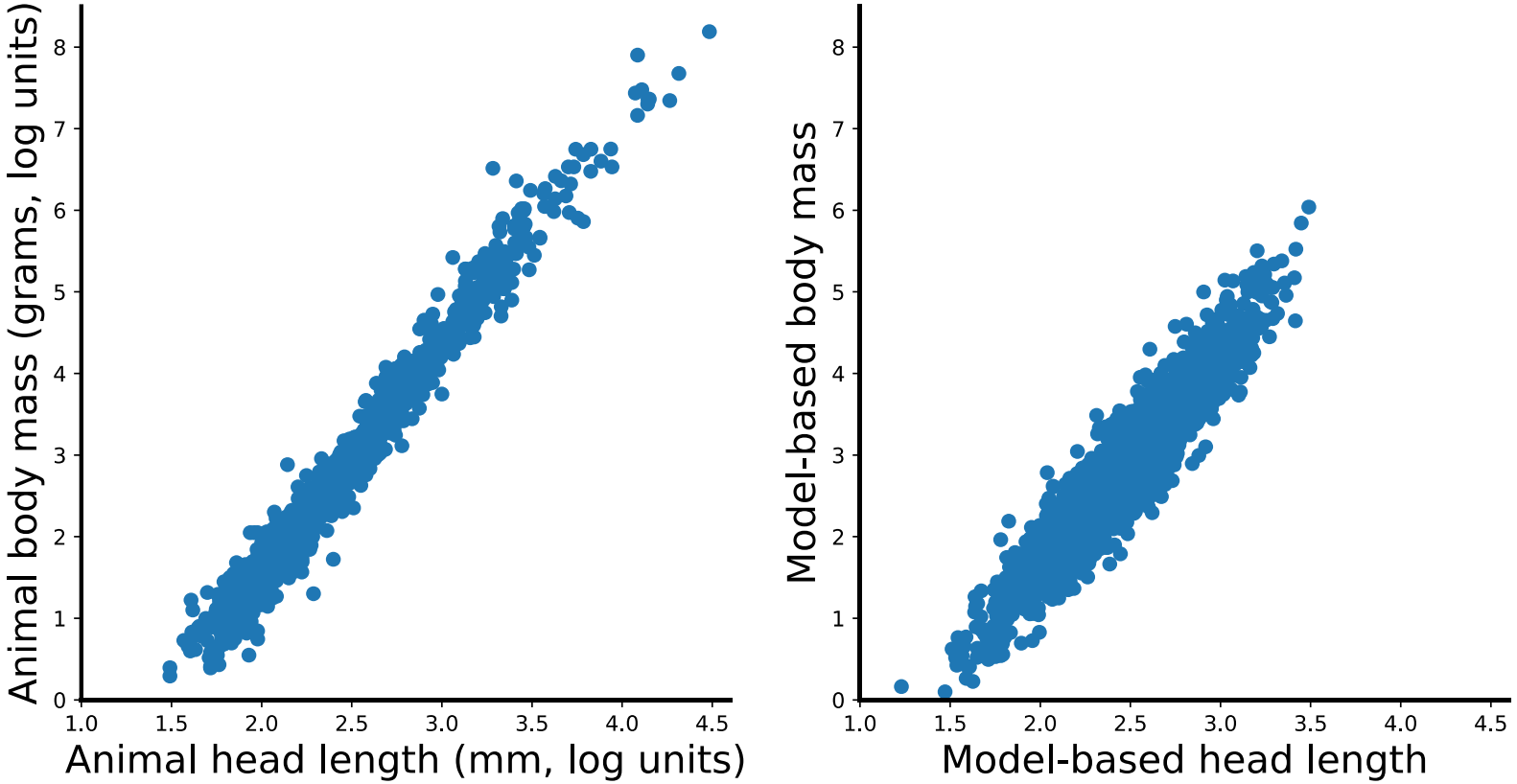
Latent variable models as data recipes

Data recipe:

Start with random 1D Gaussian

Multiply by “Size” factor

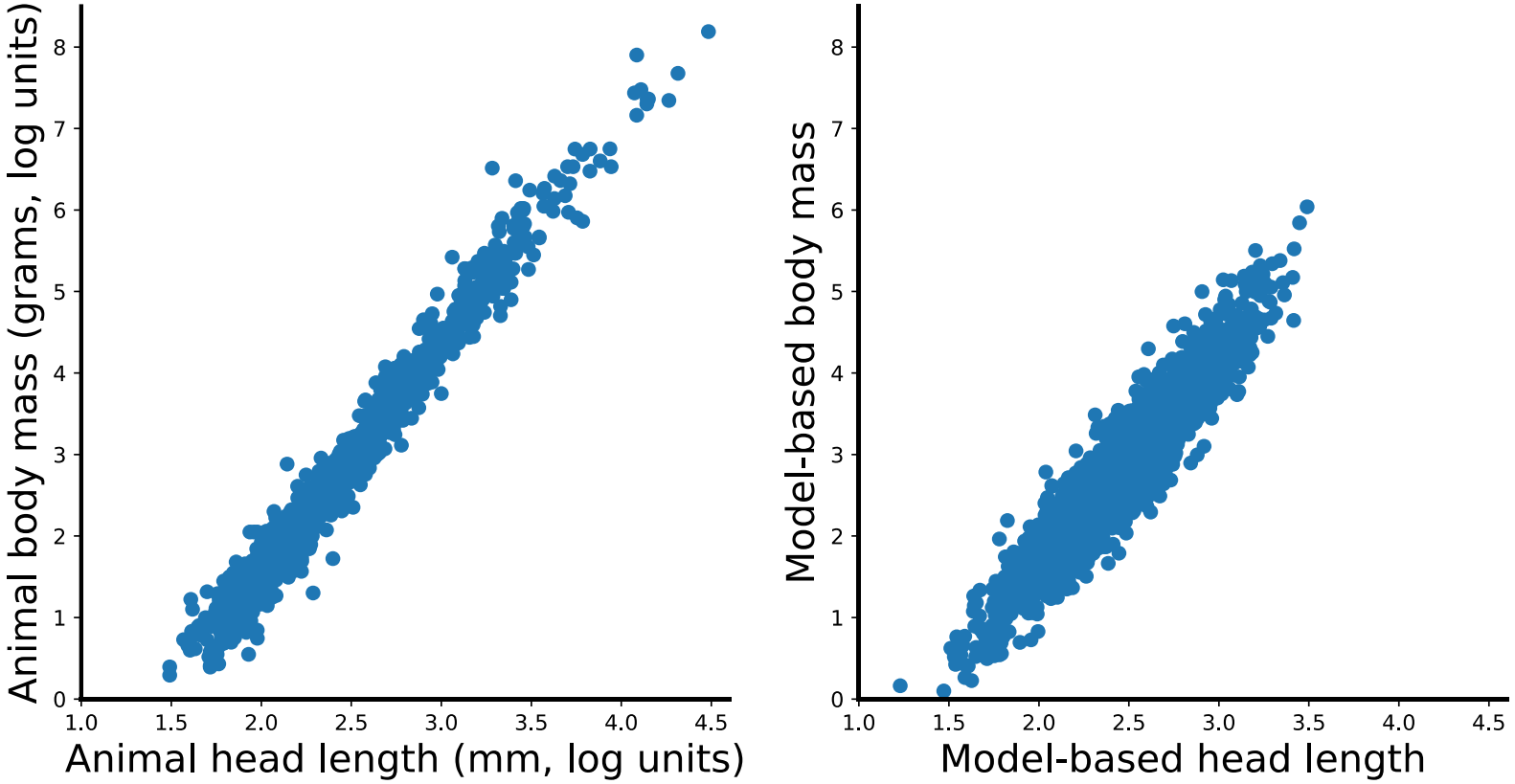
Add independent noise to each point



Latent variable models as data recipes

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

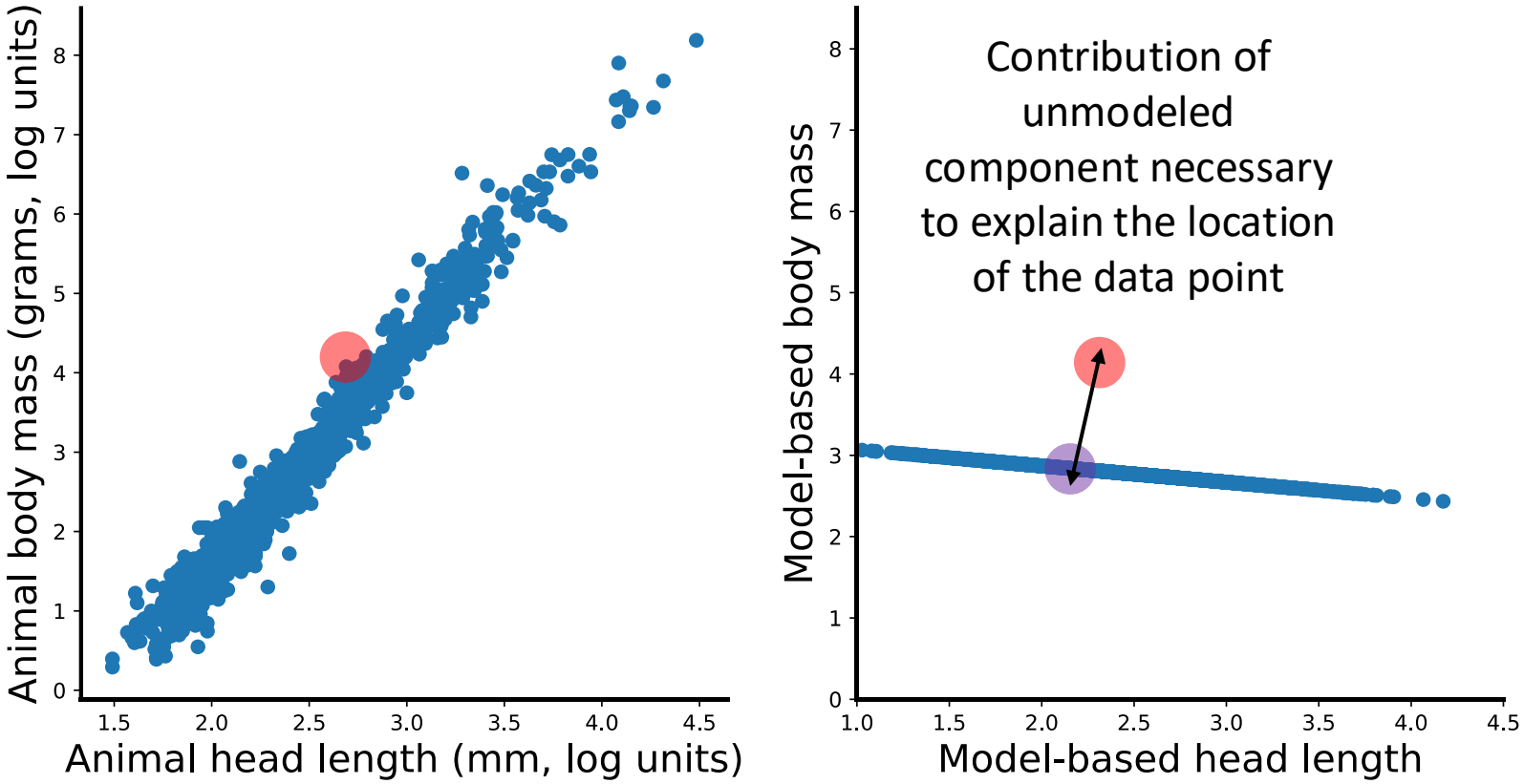
Why a recipe?
It allows us to say just how likely is it to have observed an animal with a certain head length and body mass given the model



Bad models need increasingly unlikely unmodeled contribution

Why a recipe?

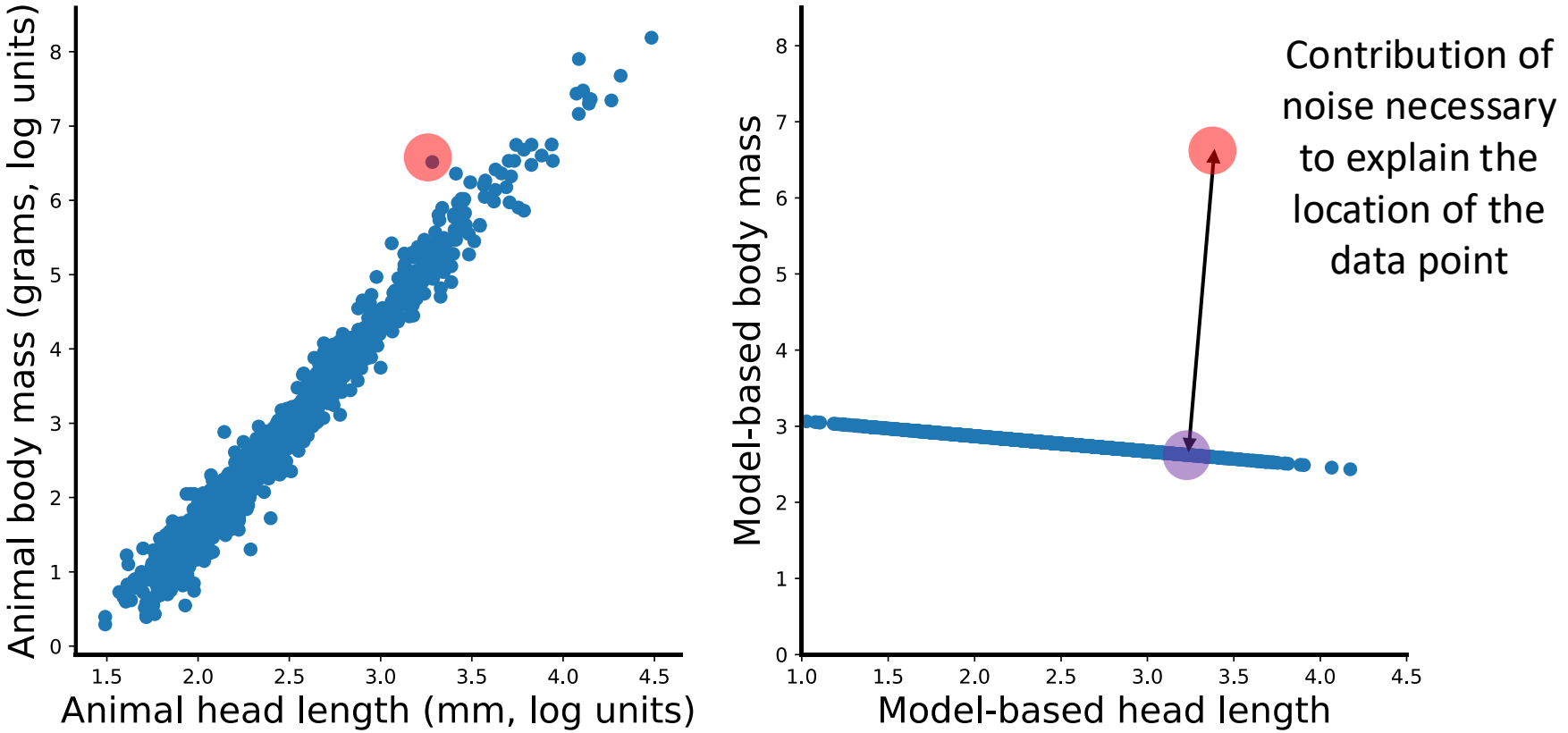
It allows us to say just how likely is it to have observed an animal with a certain head length and body mass given the model



Bad models need increasingly unlikely noise contribution

Why a recipe?

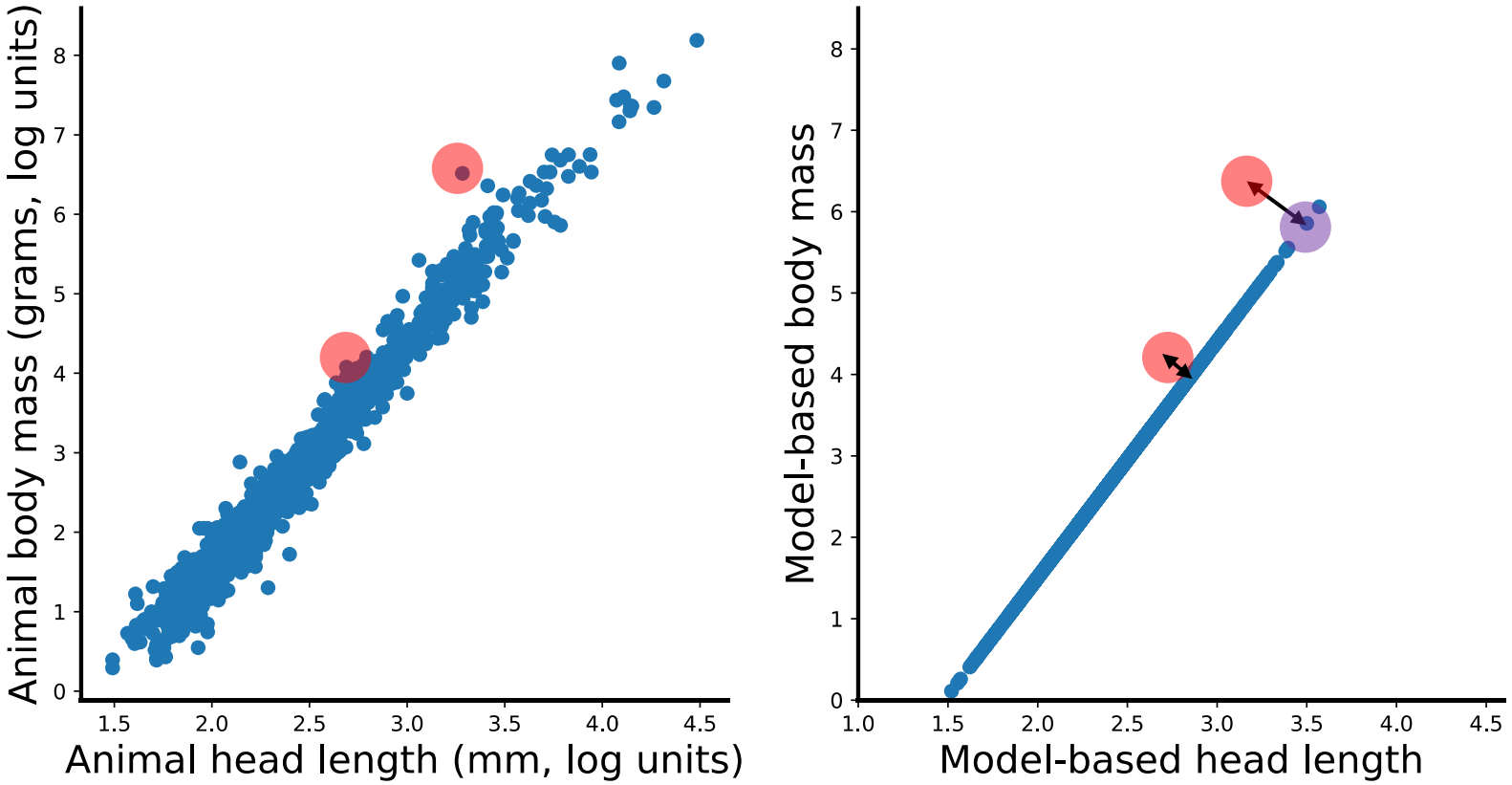
It allows us to say just how likely is it to have observed an animal with a certain head length and body mass given the model



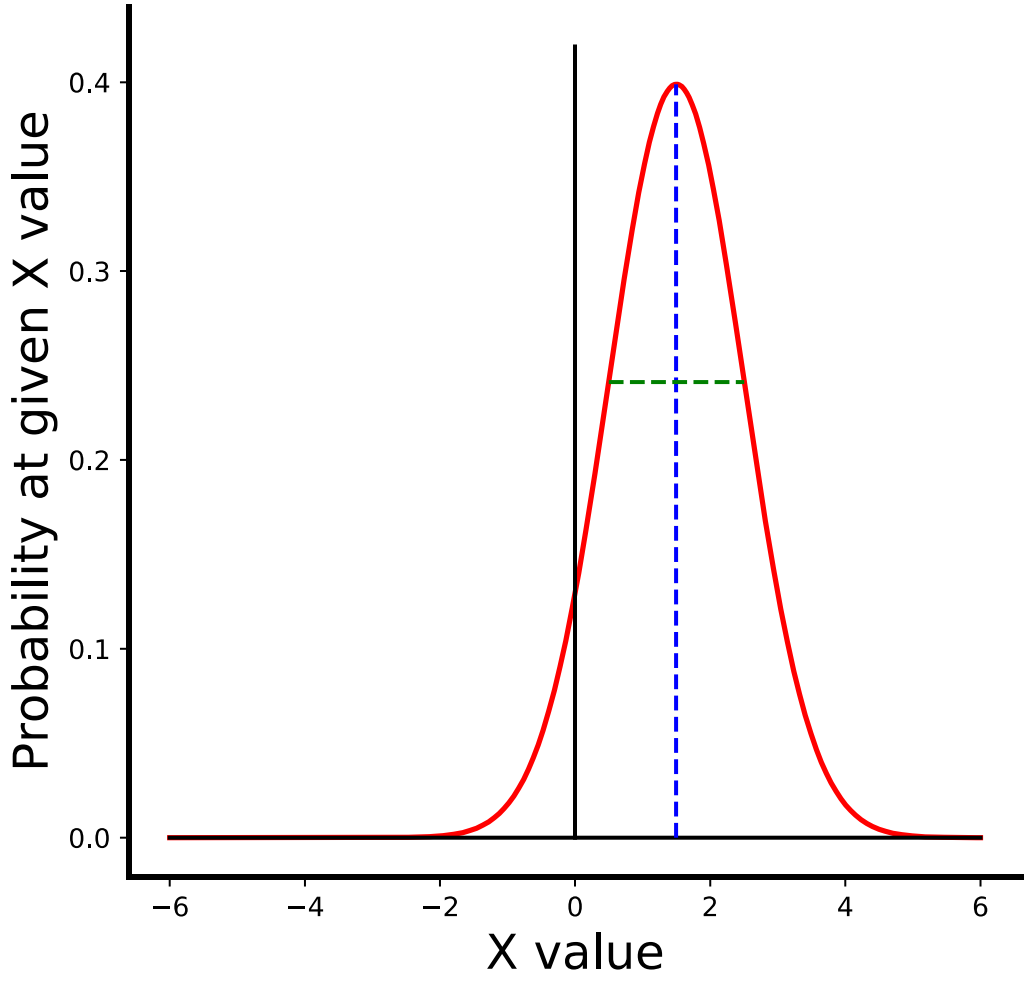
Bad models need increasingly unlikely noise contribution

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



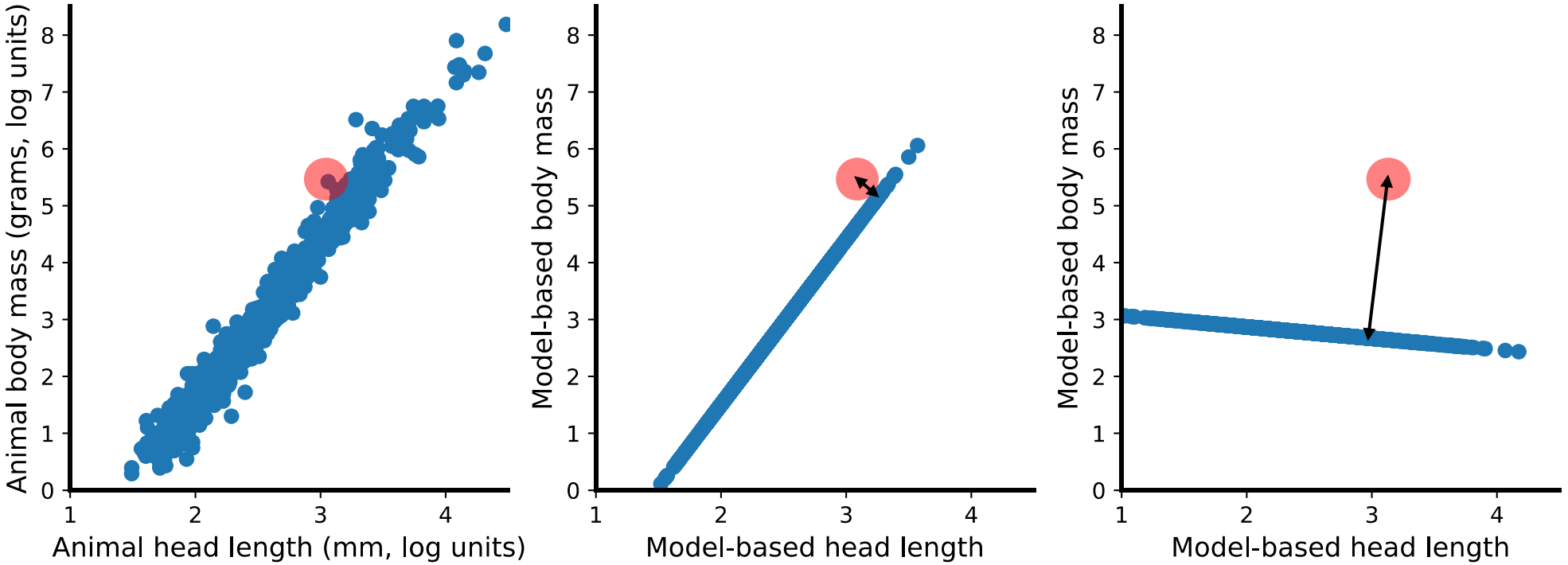
$$P\left(y = y^i_{data}\right) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{\left(y^i_{data} - \mu\right)^2}{2\sigma^2}\right)$$

One-dimensional Gaussian

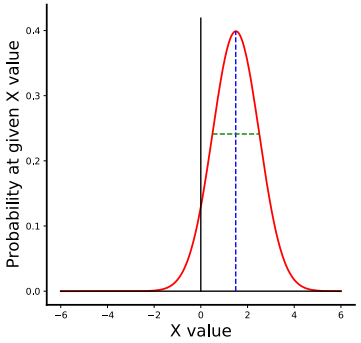
$$P\left(y = y^i_{data}\right) \sim -\frac{\left(y^i_{data} - \mu\right)^2}{2\sigma^2}$$

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

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

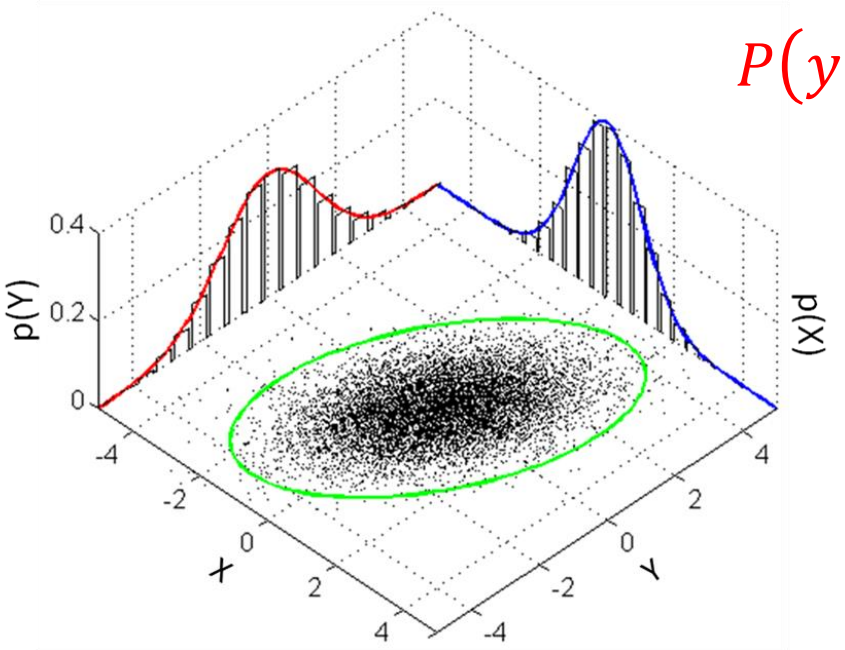


Probability of a data point: Gaussian distributions



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

One dimensional Gaussian



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

Multivariate Gaussian

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

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

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

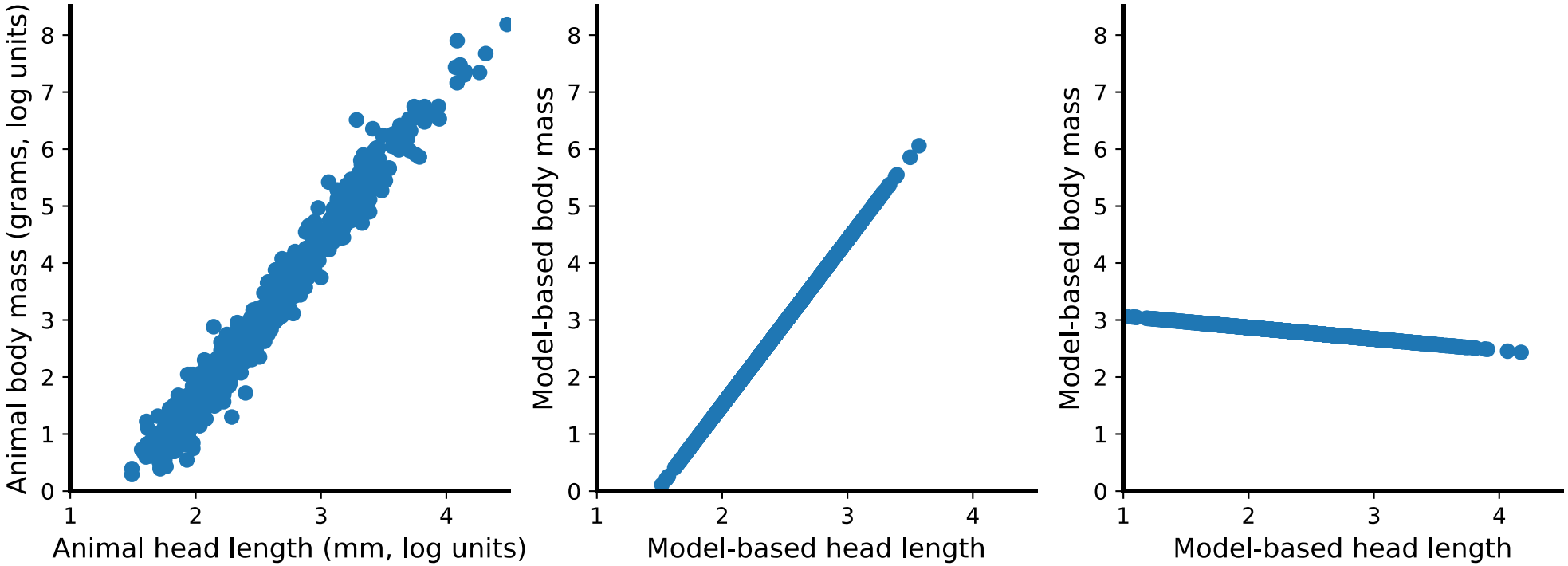
If we assume each data point is independent, then probability of multiple independent events is just multiplying the probability of each event:

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

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

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



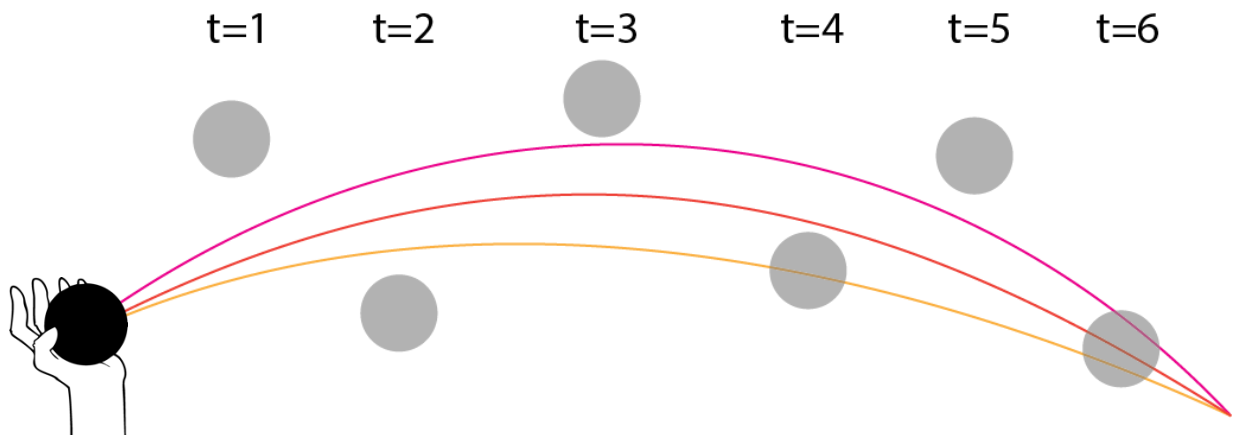
We find the latent structure by assuming a given structure and finding the parameters (relation between size factor and head length / body mass) that maximize the likelihood of the data given the model

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:



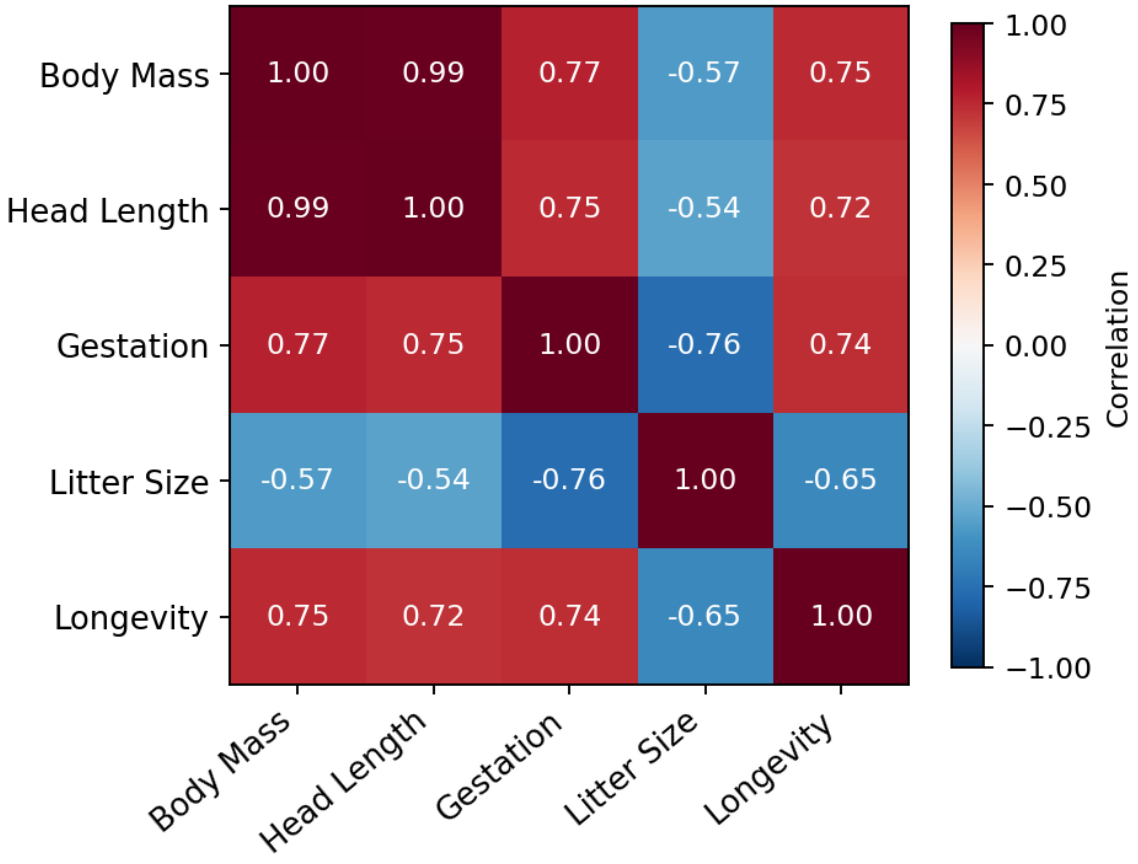
$$x = z * f + \mu$$

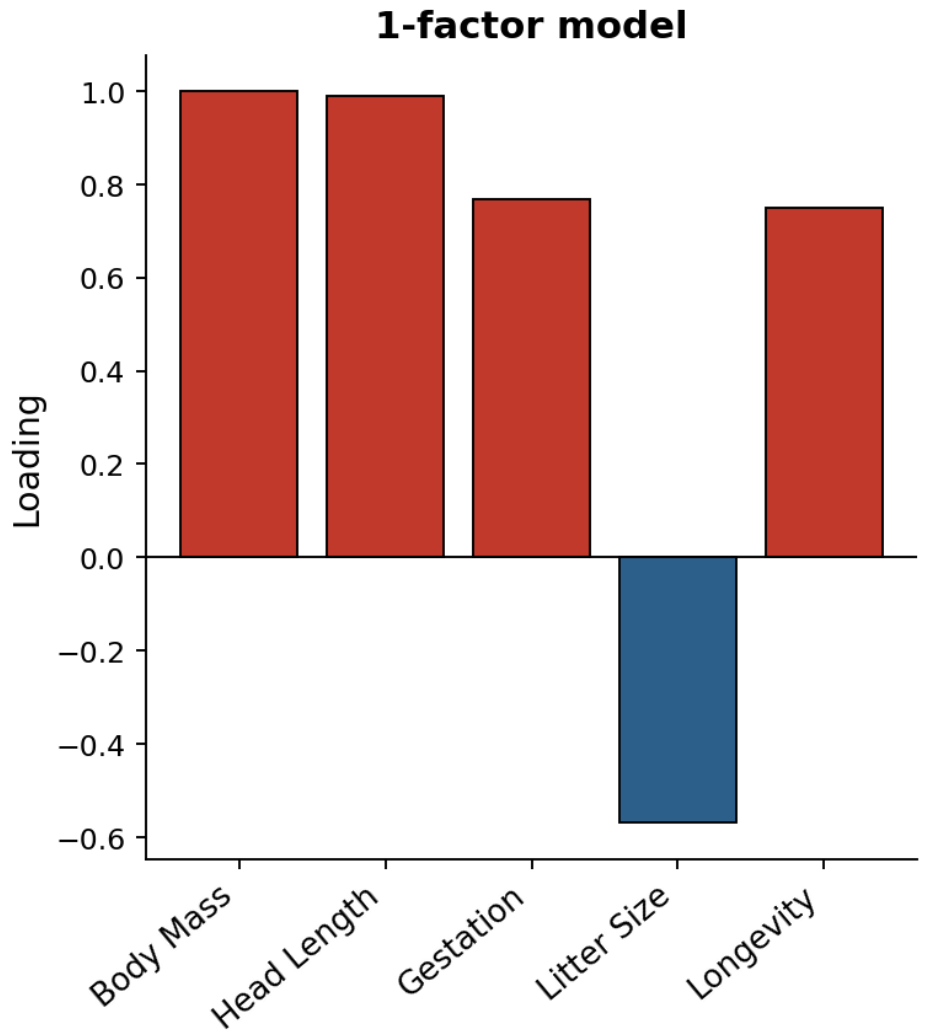
Factor Analysis model

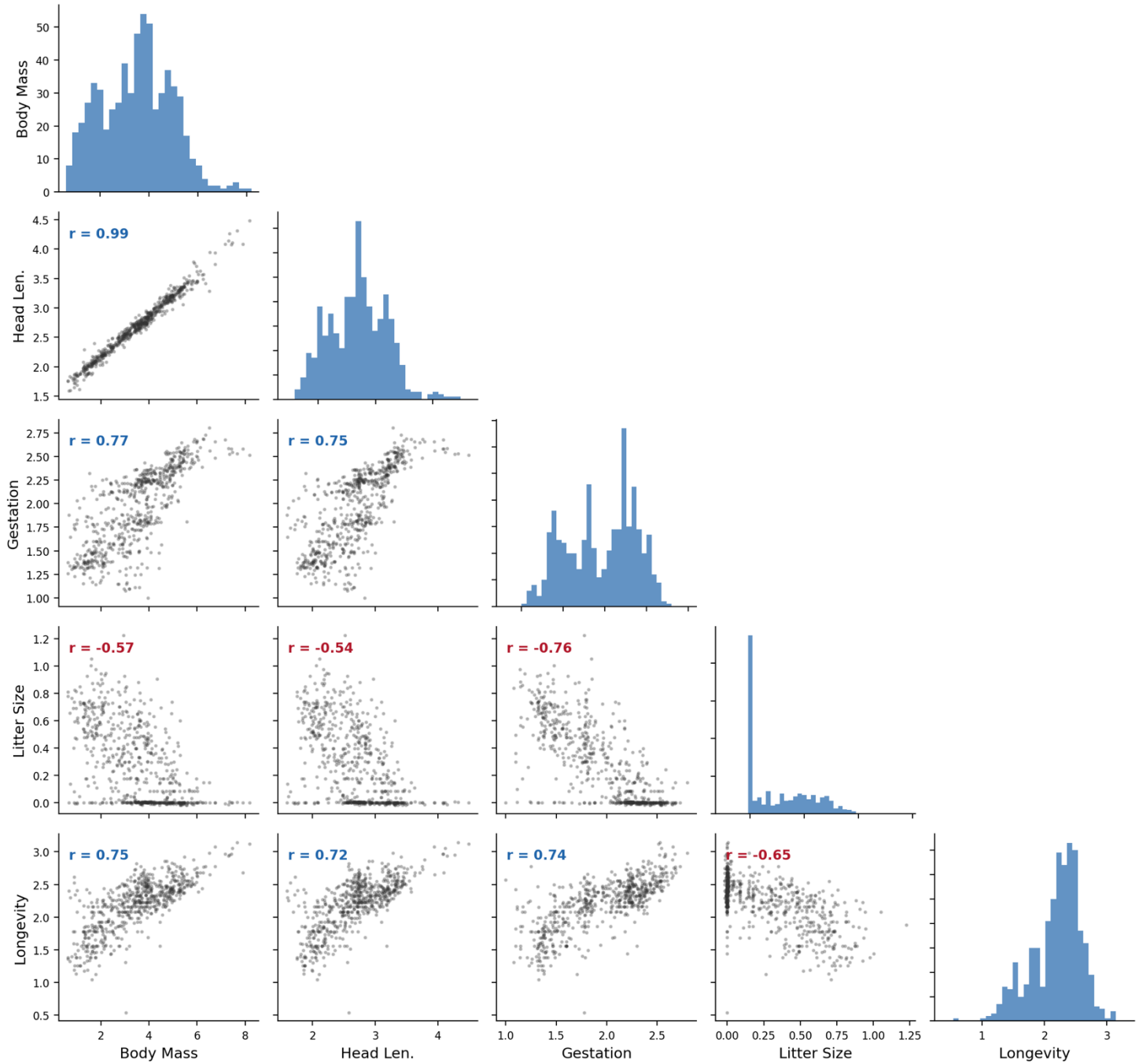
$$x(t) = Ax(t-1) + \epsilon$$

Latent Linear Dynamical System model

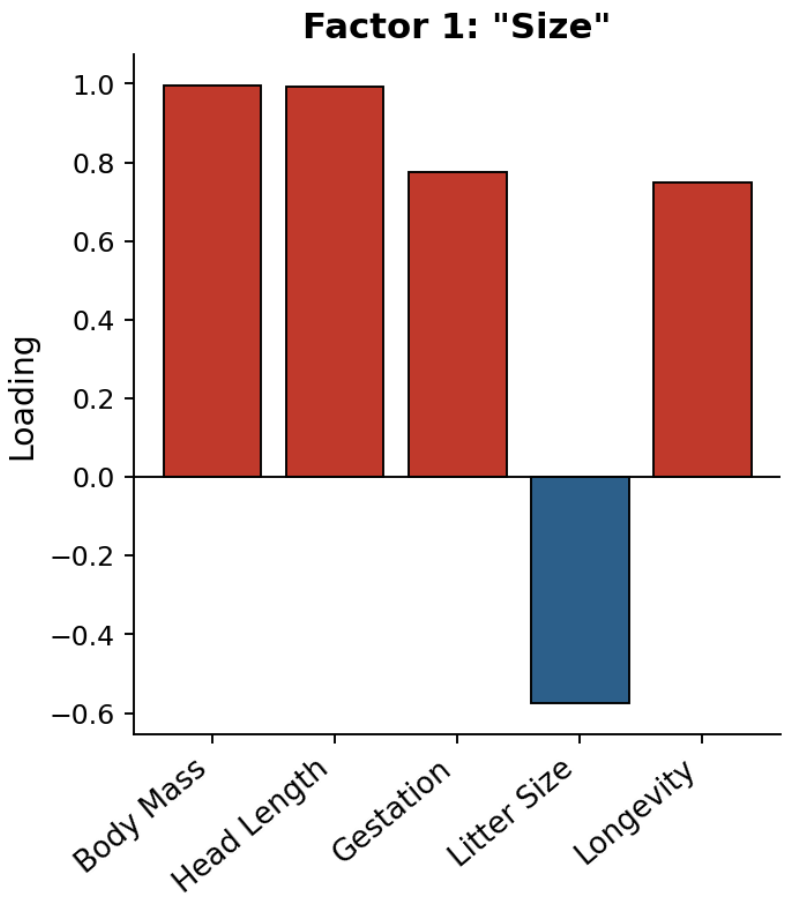
Factor Analysis beyond a single factor

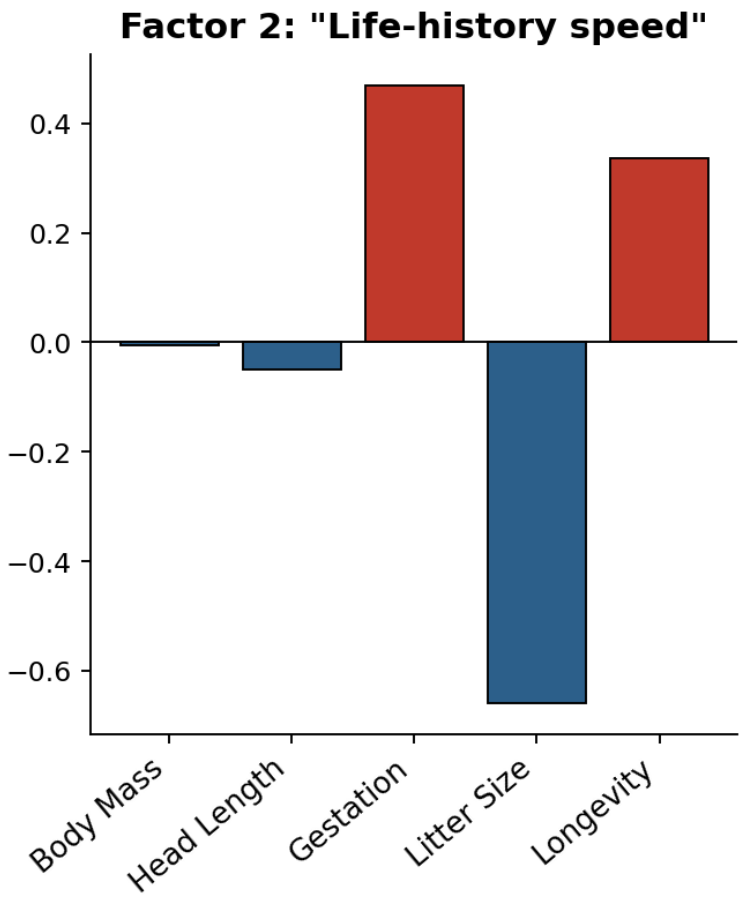


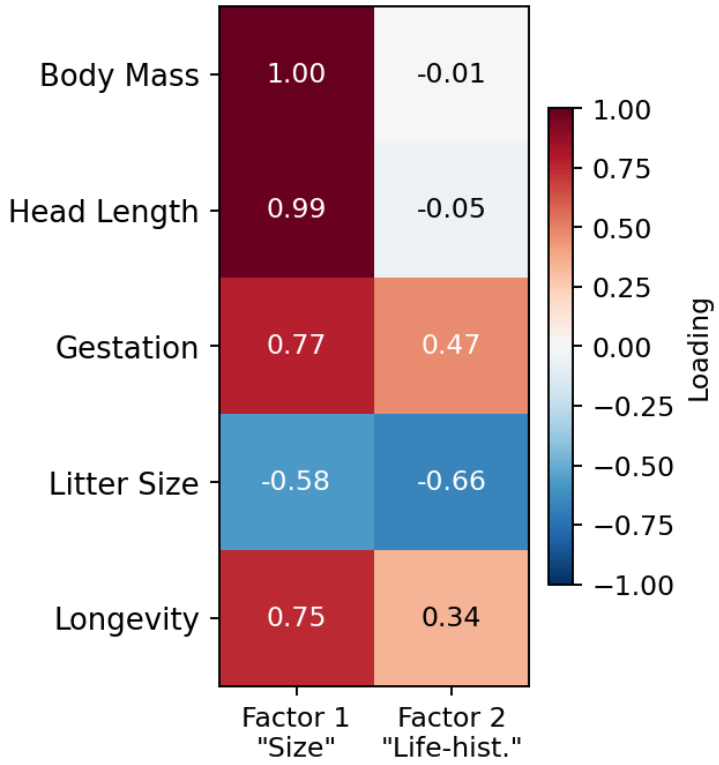



Body Mass
Head Len.
Gestation
Litter Size
Longevity
r = 0.99
r = 0.77
r = 0.75
r = -0.57
r = -0.54
r = -0.76
r = 0.75
r = 0.72
r = 0.74
r = -0.65
Body Mass
Head Len.
Gestation
Litter Size
Longevity

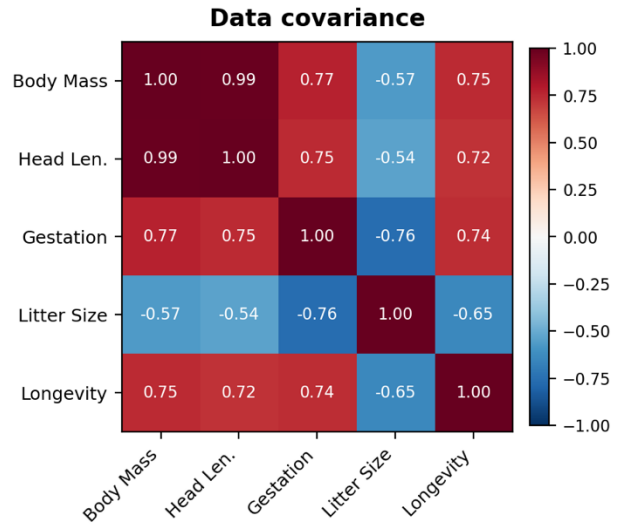
What does the second factor look like?



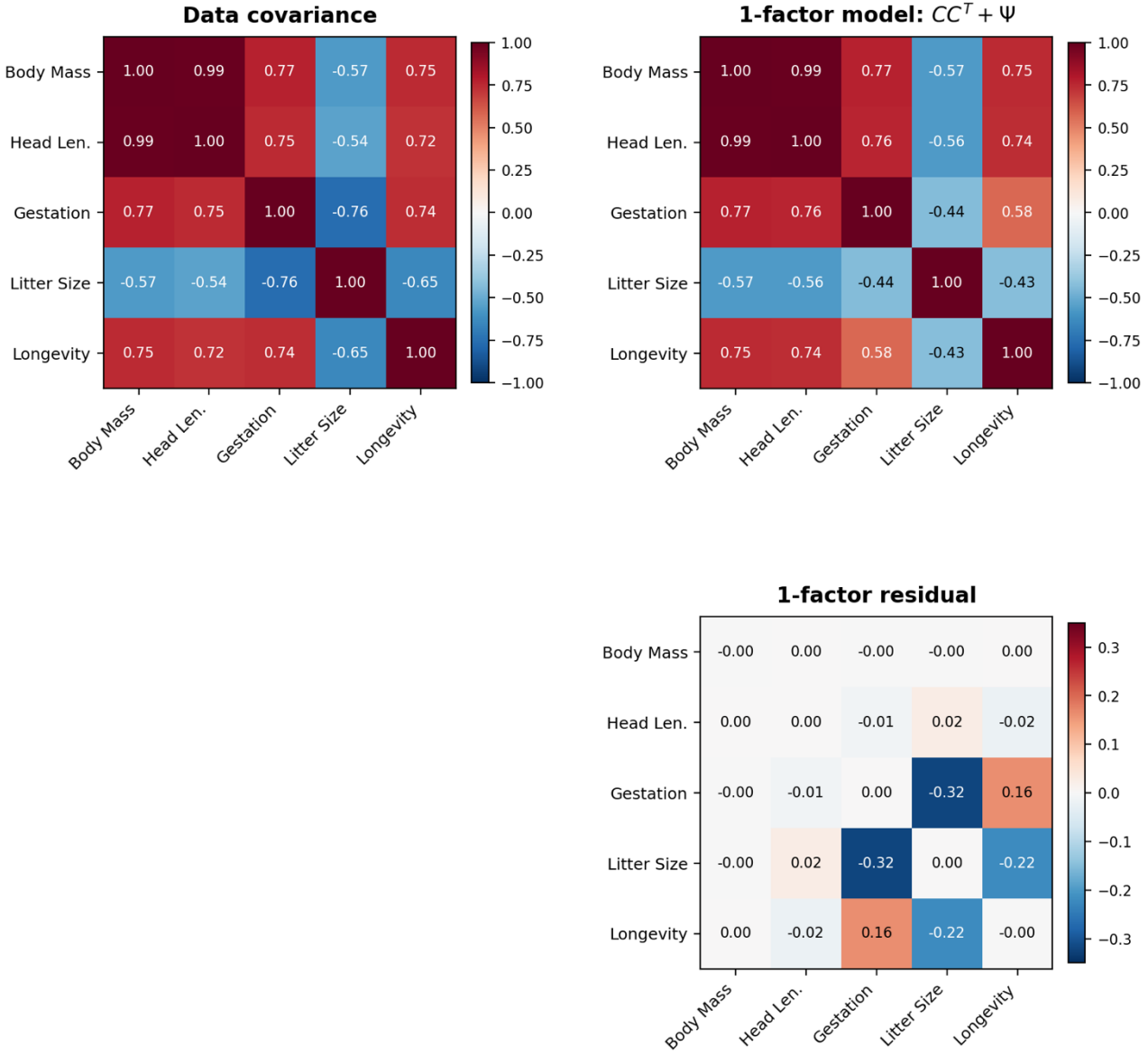




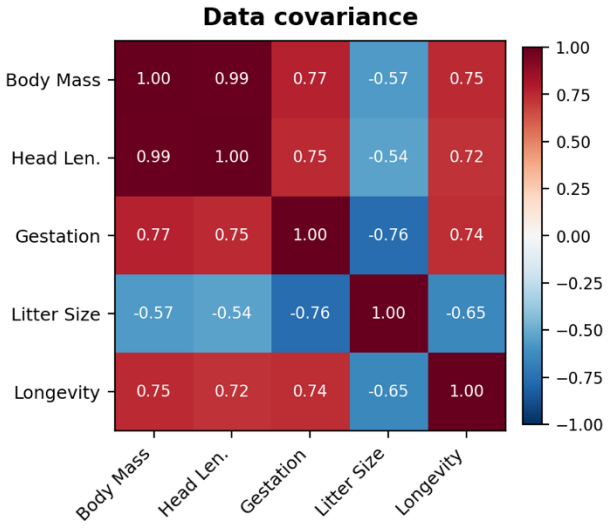
What does the second factor contribute to the covariance?

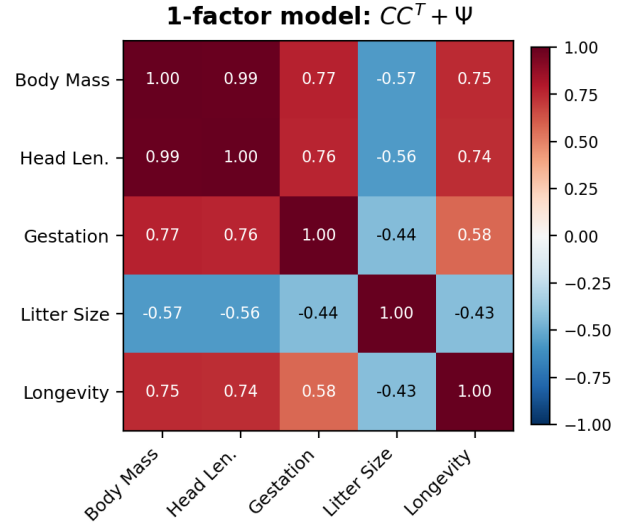


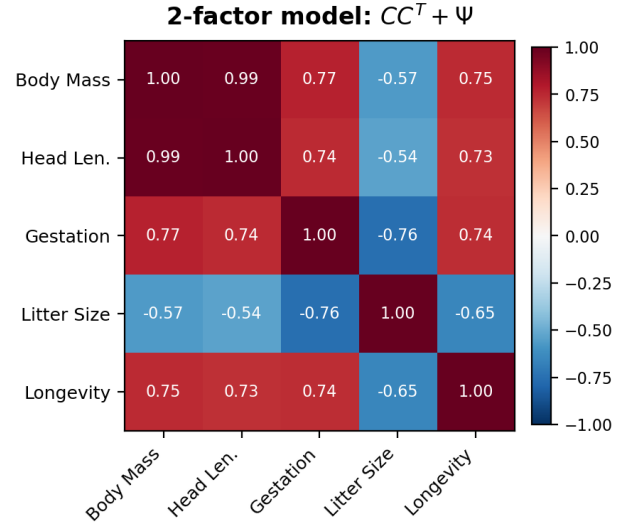
What does the second factor contribute to the covariance?

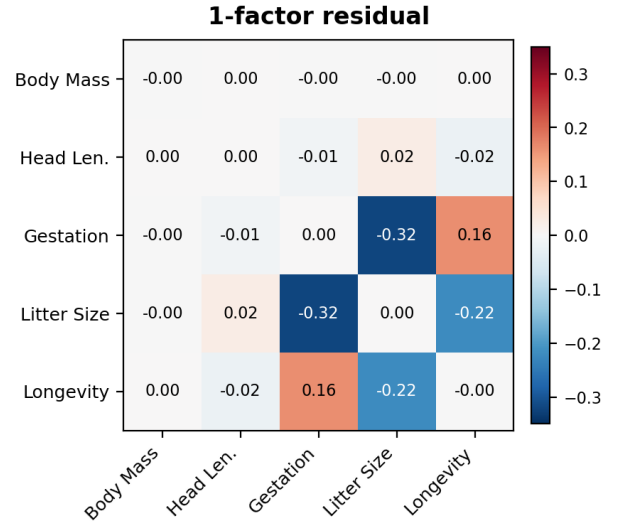


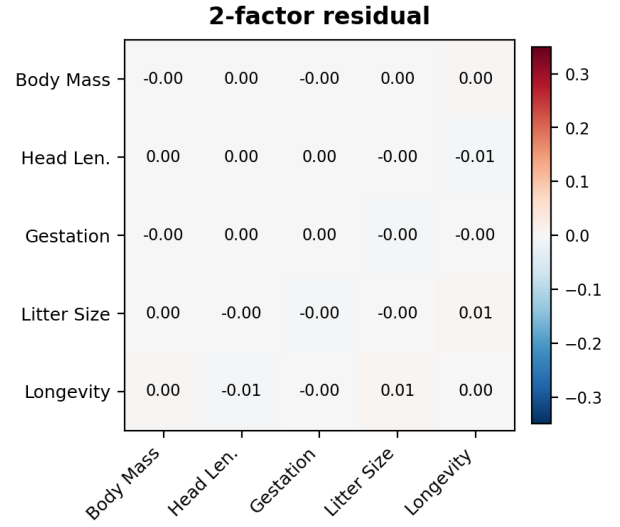
What does the second factor contribute to the covariance?



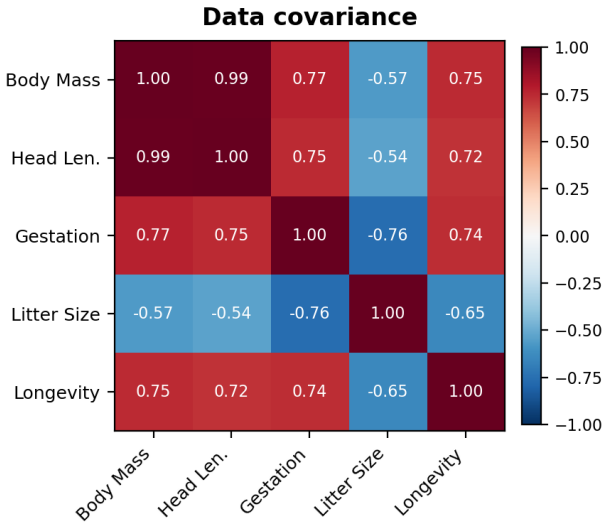


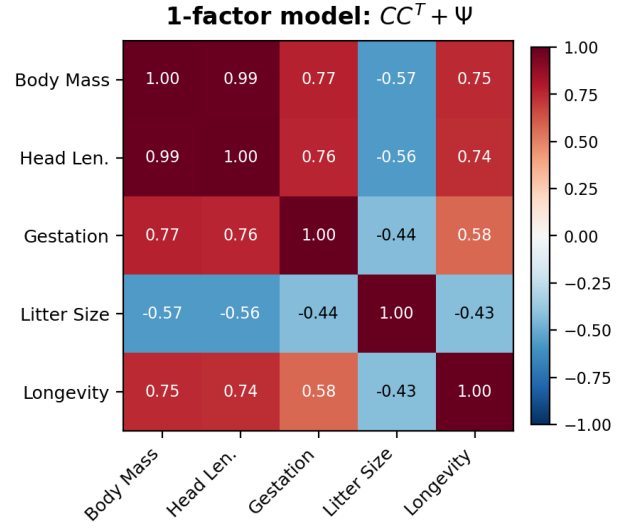


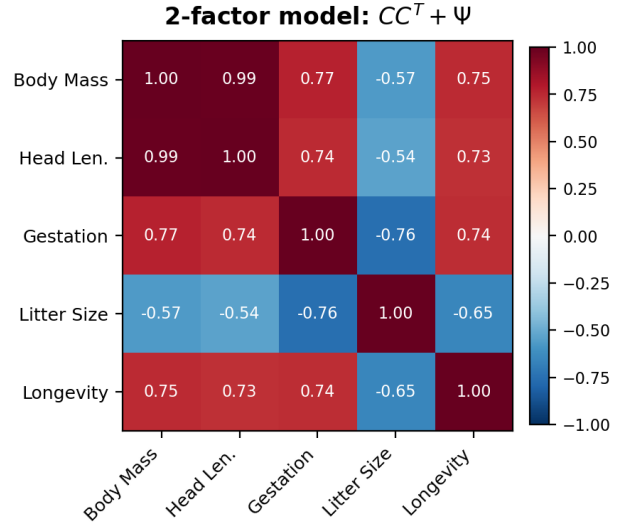




What does the second factor contribute to the covariance?





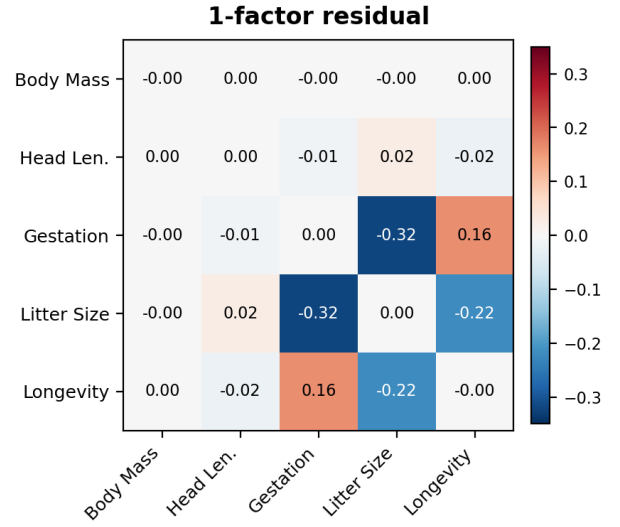


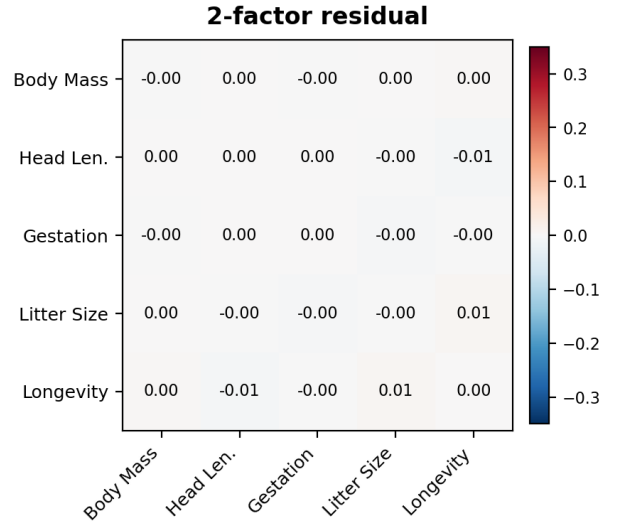
Residual Frobenius norm:

1-factor: 0.60

2-factor: 0.02

(97% reduction)





Animals in the 2D latent (Factor Analysis) space

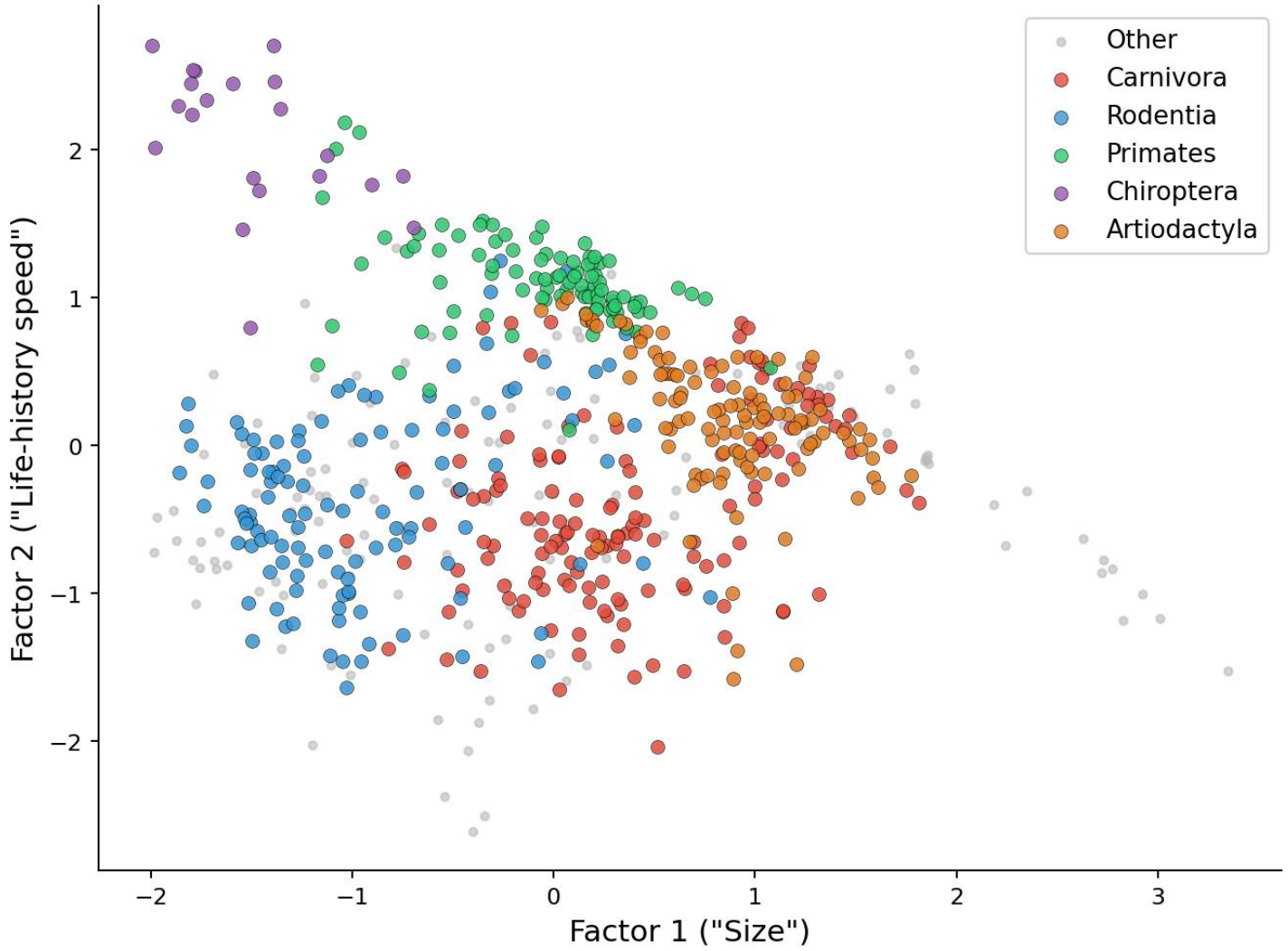
Rodentia: Small, fast life-history

Primates: Larger, slow life-history

Chiroptera (bats): Small but long-lived

Artiodactyla (pigs, hippos): Large, moderate pace

Carnivora: Spread across both axes

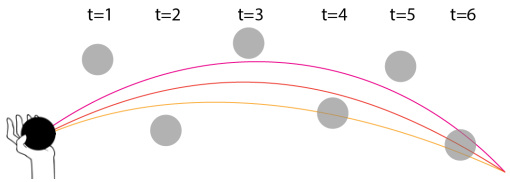


Summary

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