Machine Learning Methods for Neural Data Analysis

Lecture 4: Spike Sorting

Scott Linderman

STATS 220/320 (NBIO220, CS339N). Winter 2023.

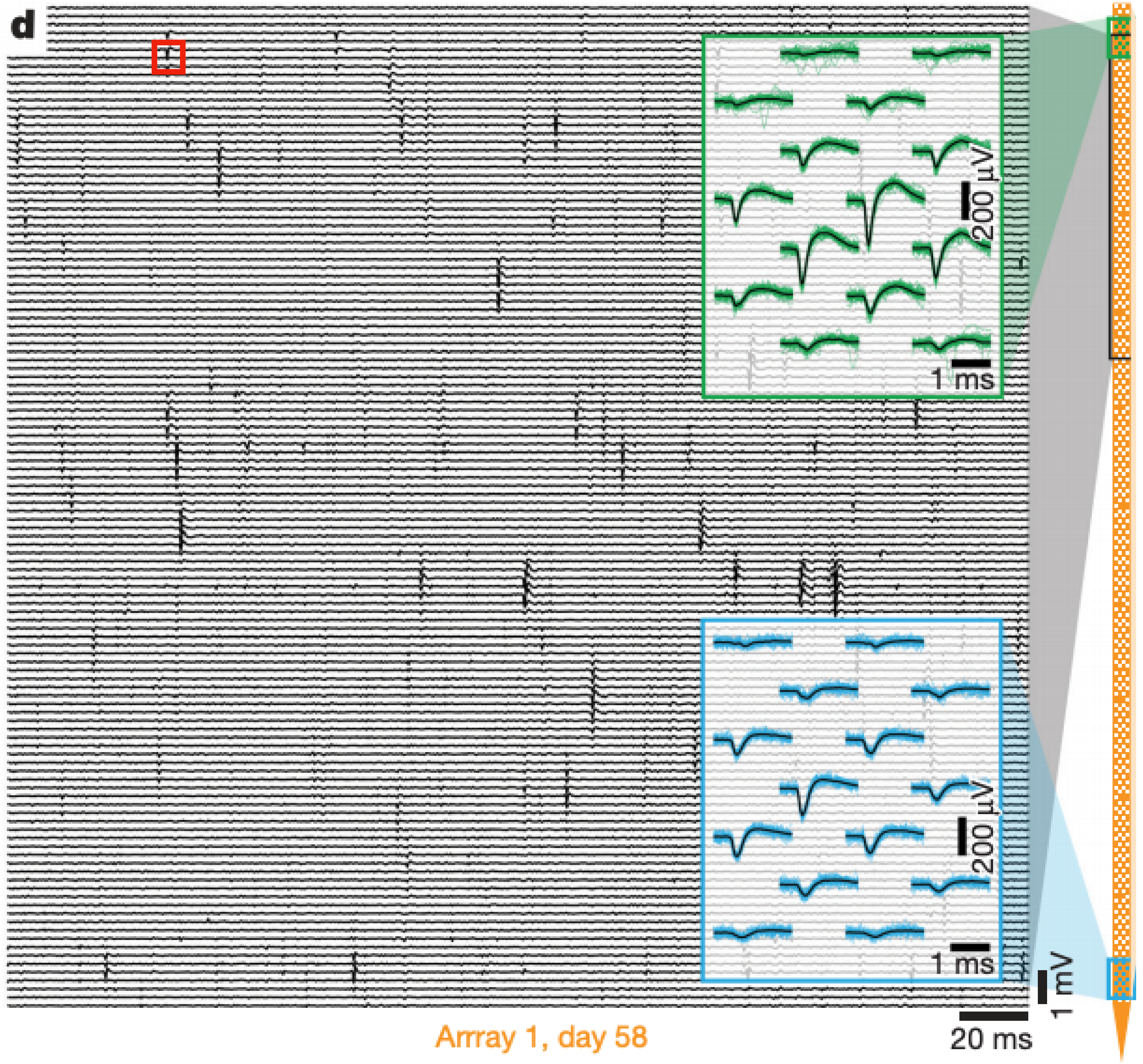
Announcements

- Course Website: https://slinderman.github.io/stats320
- Ed: I'll add auditors to Canvas and resync. If you're not on Ed yet, please let me know.
- Lab 0 will not graded, but it should be a good warm-up.
- Lab 1 is this Friday! We will implement the model in the **Spike Sorting by Deconvolution** notes.
  - Default plan is to come to this room, but stay tuned for announcements on Canvas/Ed!

Simple Spike Sorting

A simple probabilistic model

- Start with a zoomed-out view of average voltage in relatively large time bins (e.g. 2ms).
- Let N be the number of channels.
- Let T be the number of 2ms time bins.
- Let $x_{n,t}$ be the average voltage on channel n in time bin t.
- At this resolution, spikes can be contained to a single bin.



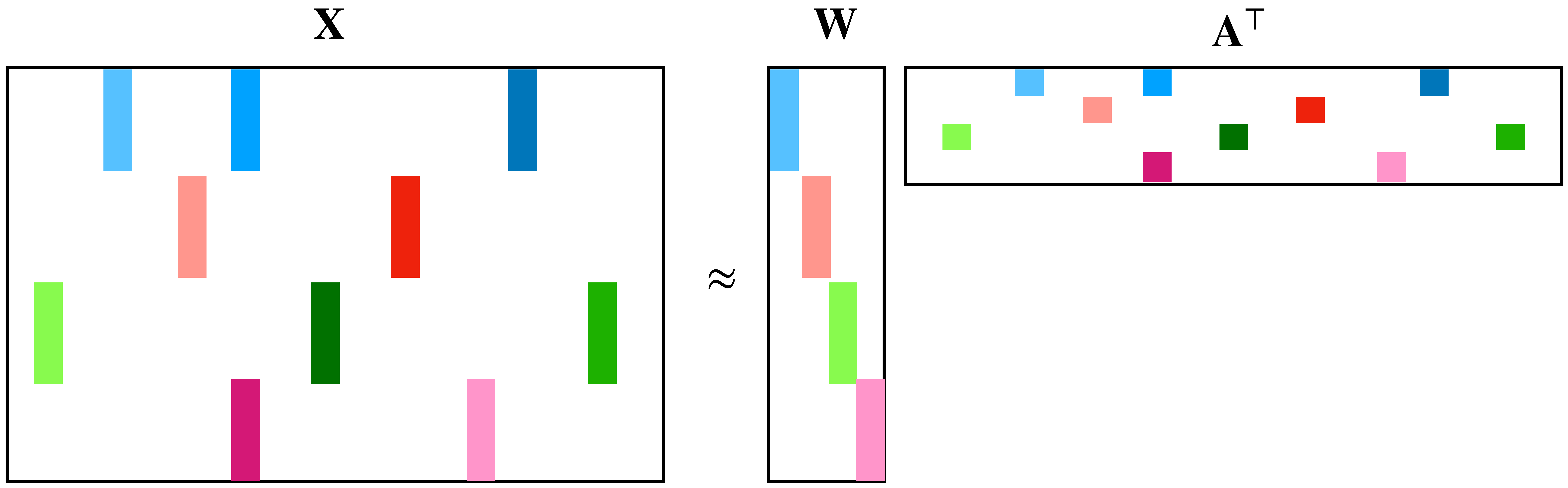
A simple probabilistic model

Assumptions

- There are K neurons. When neuron k spikes it produces a **waveform** $\mathbf{w}_k = (w_{k,1}, \ldots, w_{k,N}) \in \mathbb{R}^N$
- Let $\mathbf{a}_k = (a_{k,1}, \ldots, a_{k,T}) \in \mathbb{R}_+^T$ denote the time series of spike **amplitudes** for neuron k.
    - Since neurons spike only a few times a second, amplitudes are mostly zero.
    - Amplitudes are non-negative.
- If two neurons spike at the same, waveforms add.
- Voltage recordings have additive noise.

A simple probabilistic model

Matrix factorization perspective



A simple probabilistic model

Accounting for scale invariance

- Notice that the model is **invariant to rescaling.**
  - Multiple $\mathbf{a}_k$ by constant $c > 0$ and scale $\mathbf{w}_k$ by c^{-1}.
- We can remove this degree of freedom by forcing $\|\mathbf{w}_k\|_2 = 1$; e.g., with a **uniform prior** on the unit hypersphere,

$$\mathbf{w}_k \sim \mathrm{Unif}(\mathbb{S}_{N-1})$$

- where $\mathbb{S}_{N-1} = \left\{\mathbf{u} : \mathbf{u} \in \mathbb{R}^N \text{ and } \|\mathbf{u}\|_2 = 1\right\}$

A simple probabilistic model

Prior on amplitudes

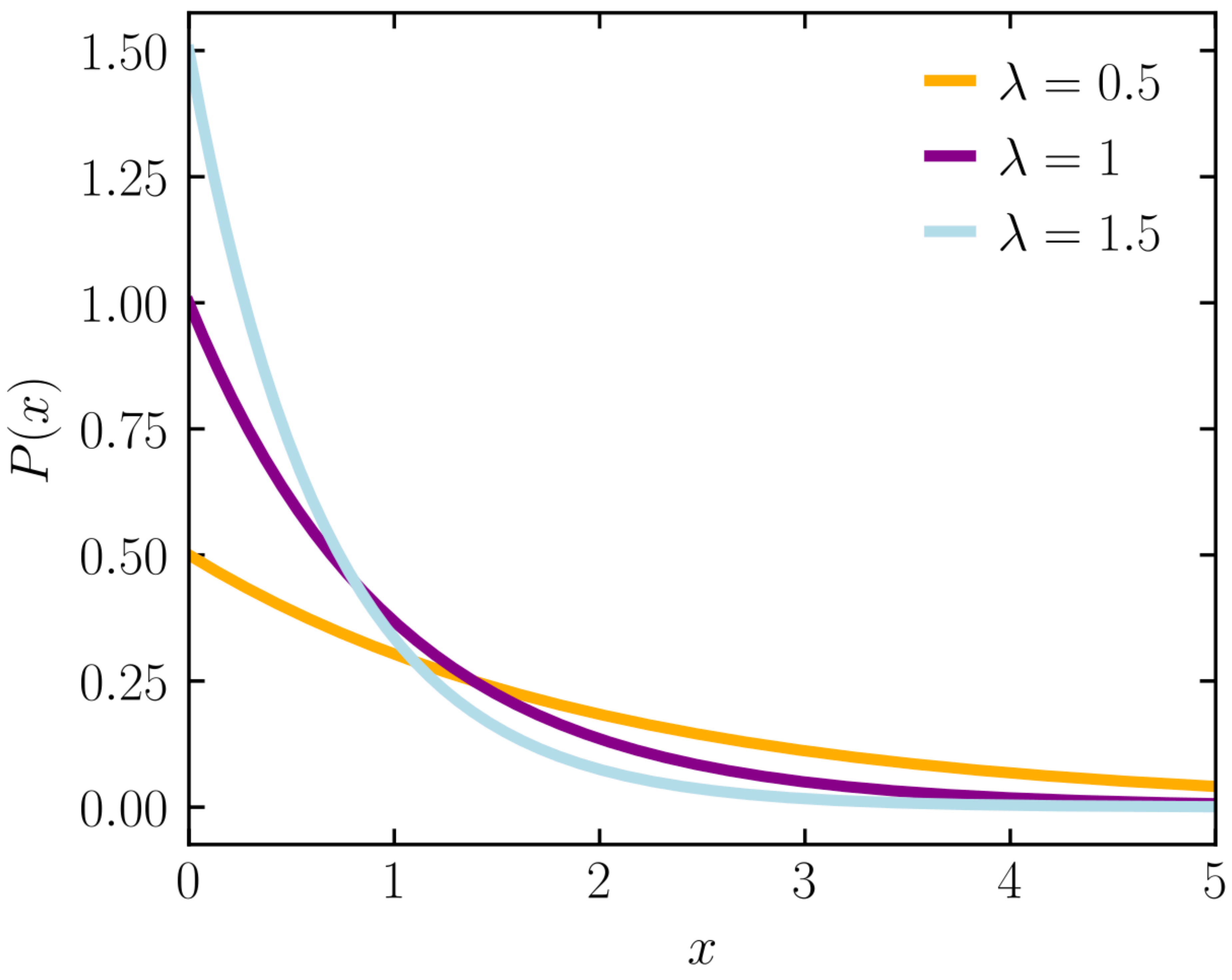
- To complete the model, we place an **exponential** prior on amplitudes,

$$a_{k,t} \sim \mathrm{Exp}(\lambda)$$

where λ is the inverse-scale (aka rate) parameter.

- It's pdf is $\mathrm{Exp}(x; \lambda) = \lambda e^{-\lambda x}$.

- As we will see, this prior will lead to **sparse** estimates.



https://en.wikipedia.org/wiki/Exponential_distribution

A simple probabilistic model

Noise model

- So far, $\mathbf{X} = \mathbf{W}\mathbf{A}^\top + \mathbf{E}$ where $\mathbf{E} = [[\epsilon_{n,t}]]$ is a matrix of "noise." How to model the noise?

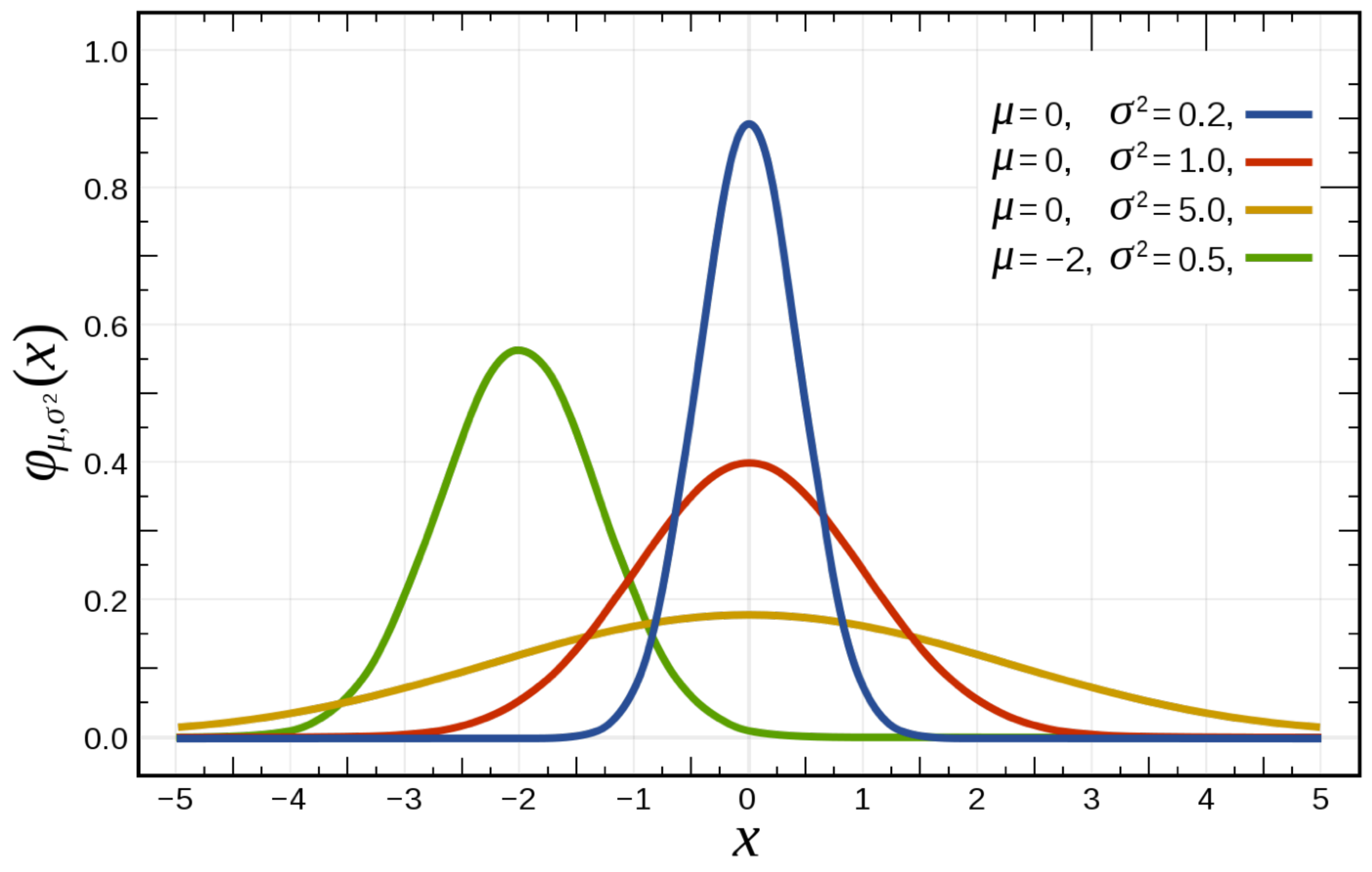
- Simple assumption: $\epsilon_{n,t} \sim \mathcal{N}(0,\sigma^2)$ where

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2}(x-\mu)^2 \right\}$$

  is the **Gaussian** or **normal distribution**.

- Linear transformations of Gaussians are still Gaussian!

$$x \sim \mathcal{N}(\mu, \sigma^2) \Rightarrow ax + b \sim \mathcal{N}(a\mu + b, a^2\sigma^2)\,.$$



https://en.wikipedia.org/wiki/Normal_distribution

A simple probabilistic model

The joint distribution

$$
\begin{aligned}
p(\mathbf{X}, \mathbf{W}, \mathbf{A}) = p(\mathbf{X} \mid \mathbf{W}, \mathbf{A})\, p(\mathbf{W})\, p(\mathbf{A}) \\
= \left[\prod_{n=1}^{N} \prod_{t=1}^{T} \mathcal{N}\left(x_{n,t} \mid \sum_{k=1}^{K} w_{k,n} a_{k,t}, \sigma^2 \right) \right] \\
\times \left[\prod_{k=1}^{K} \mathrm{Unif}(\mathbf{w}_k; \mathbb{S}_{N-1}) \right] \times \left[\prod_{k=1}^{K} \prod_{t=1}^{T} \mathrm{Exp}(a_{k,t}; \lambda) \right].
\end{aligned}
$$

This is called **semi-nonnegative matrix factorization (semi-NMF).**

Fitting the model

MAP estimation by coordinate ascent

- repeat until convergence:
  - for $k = 1, \ldots, K$:
    - Set $\mathbf{w}_k =$ arg max $p(\mathbf{X}, \mathbf{W}, \mathbf{A})$ holding all else fixed
    - Set $\mathbf{a}_k =$ arg max $p(\mathbf{X}, \mathbf{W}, \mathbf{A})$ holding all else fixed

Fitting the model
Optimizing the waveforms

Maximizing the joint probability wrt $\mathbf{w}_k$ is equivalent to maximizing the log joint probability,

$$\begin{aligned}
\log p(\mathbf{X}, \mathbf{W}, \mathbf{A}) &= \sum_{n=1}^{N} \sum_{t=1}^{T} \log \mathcal{N}\left(x_{n,t} \,\middle|\, \sum_{j=1}^{K} w_{j,n} a_{j,t}, \sigma^2\right) \\
&= -\frac{1}{2\sigma^2} \sum_{n=1}^{N} \sum_{t=1}^{T} \left(x_{n,t} - \sum_{j=1}^{K} w_{j,n} a_{j,t}\right)^2 + c' \\
&= -\frac{1}{2\sigma^2} \sum_{n=1}^{N} \sum_{t=1}^{T} \left(r_{n,t} - w_{k,n} a_{k,t}\right)^2 + c'
\end{aligned}$$

where $r_{n,t} = x_{n,t} - \sum_{j \neq k} w_{j,n} a_{j,t}$ is the **residual**.

Fitting the model
Optimizing the waveforms

It's easier to solve in vector form. Let $\mathbf{r}_t = (r_{1,t}, \ldots, r_{N,t})$. Then,

$$\log p(\mathbf{X}, \mathbf{W}, \mathbf{A}) = -\frac{1}{2\sigma^2} \sum_{t=1}^{T} (\mathbf{r}_t - \mathbf{w}_k a_{k,t})^\top (\mathbf{r}_t - \mathbf{w}_k a_{k,t}) + c'$$

$$= \sum_{t=1}^{T} \mathcal{N}(\mathbf{r}_t; \mathbf{w}_k a_{k,t}, \sigma^2 \mathbf{I}) + c'$$

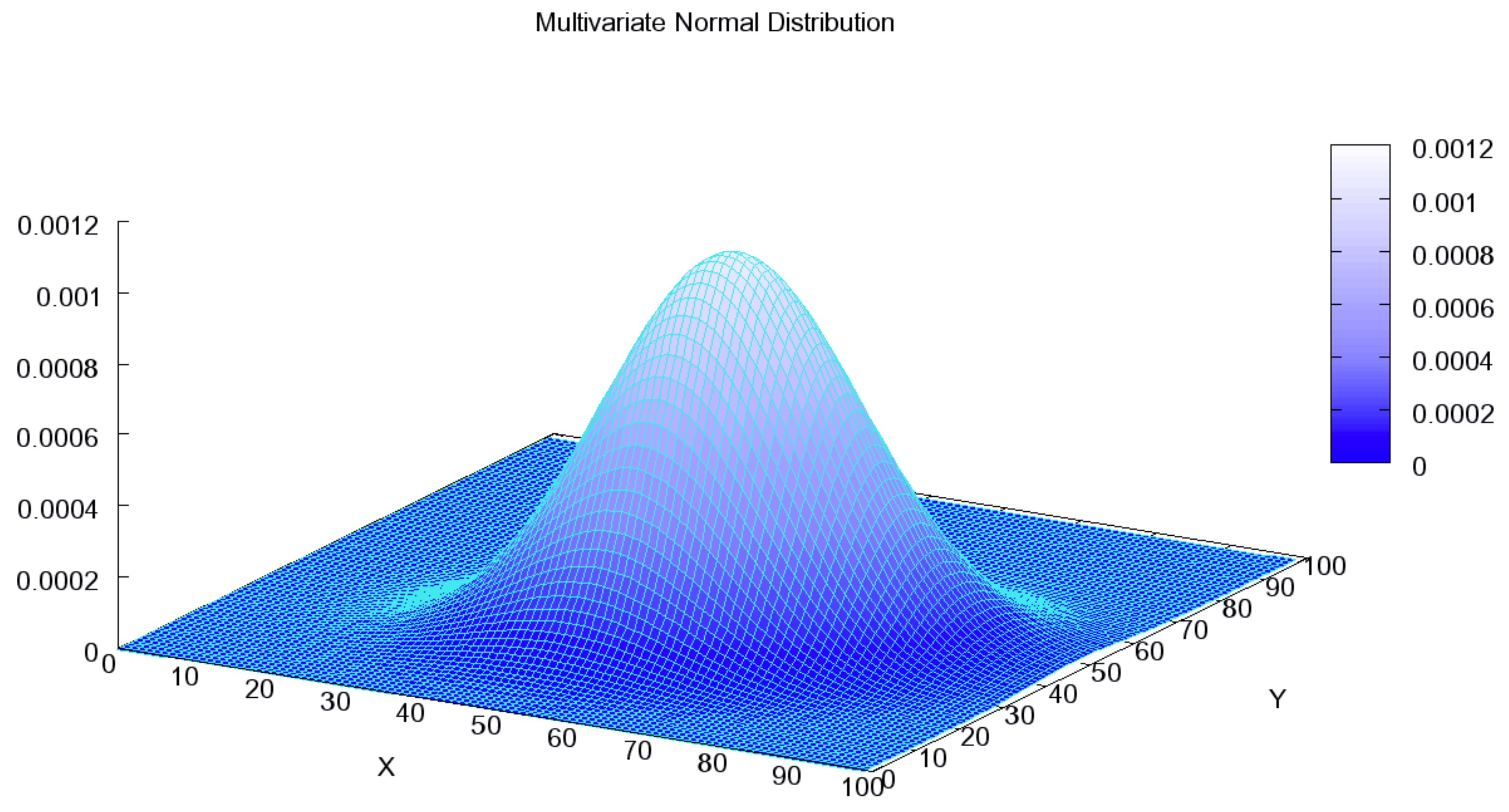
where $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the **multivariate normal distribution**.

Fitting the model

The multivariate normal distribution

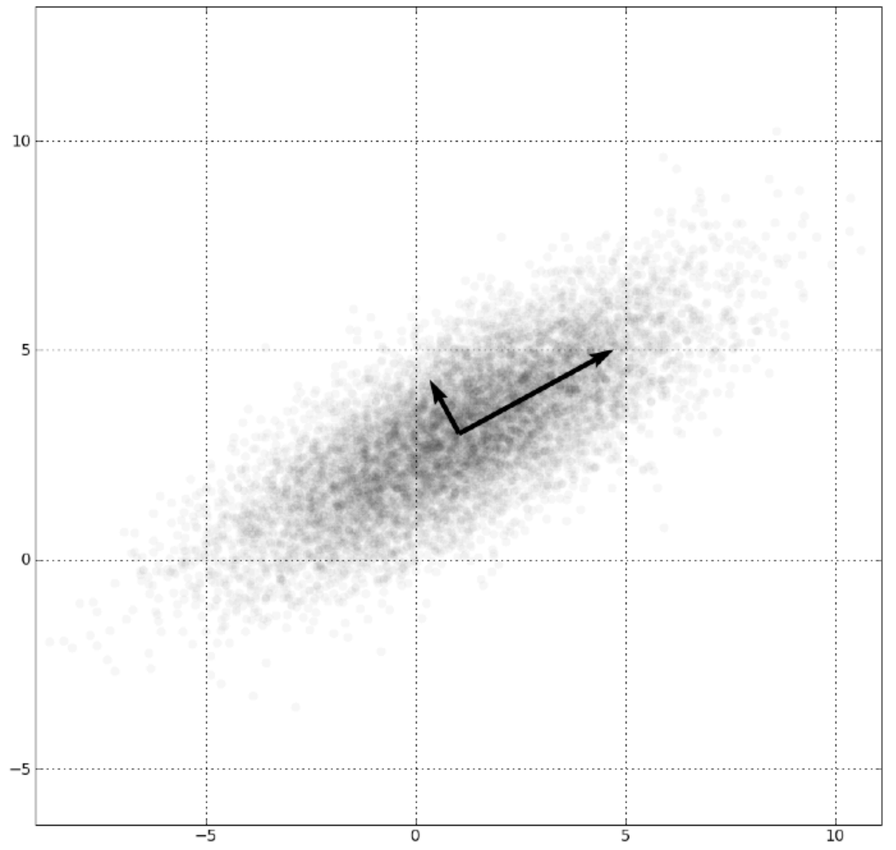
The multivariate normal density for $\mathbf{x} \in \mathbb{R}^D$ is,

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \right\}$$



where $\boldsymbol{\mu} \in \mathbb{R}^D$ is the **mean** and $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}_{\succeq 0}$ is the (positive definite) **covariance matrix.**

When $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$, we call it a **spherical Gaussian** distribution.



https://en.wikipedia.org/wiki/Multivariate_normal_distribution

Fitting the model

Optimizing the waveforms

Returning to the optimization

$$
\begin{aligned}
\log p(\mathbf{X}, \mathbf{W}, \mathbf{A}) &= \sum_{t=1}^{T} \mathcal{N}(\mathbf{r}_t; \mathbf{w}_k a_{k,t}, \sigma^2 \mathbf{I}) + c' \\
&= -\frac{1}{2\sigma^2} \sum_{t=1}^{T} (\mathbf{r}_t - \mathbf{w}_k a_{k,t})^\top (\mathbf{r}_t - \mathbf{w}_k a_{k,t}) + c' \\
&= \frac{1}{\sigma^2} \sum_{t=1}^{T} \left(\mathbf{r}_t^\top \mathbf{w}_k a_{k,t} - \frac{a_{k,t}^2}{2} \mathbf{w}_k^\top \mathbf{w}_k \right) + c''
\end{aligned}
$$

Note: $\mathbf{w}_k^\top \mathbf{w}_k = 1$ by the constraint $\mathbf{w}_k \in \mathbb{S}_{N-1}$.

Fitting the model

Optimizing the waveforms

$$\mathbf{w}_k^\star = \arg \max_{\mathbf{w}_k \in \mathbb{S}_{N-1}} \left(\sum_{t=1}^{T} a_{k,t} \mathbf{r}_t \right)^\top \mathbf{w}_k$$
$$= \arg \max_{\mathbf{w}_k \in \mathbb{S}_{N-1}} \left\langle \sum_{t=1}^{T} a_{k,t} \mathbf{r}_t, \mathbf{w}_k \right\rangle$$
$$= \arg \max_{\mathbf{w}_k \in \mathbb{S}_{N-1}} \langle \mathbf{R}\mathbf{a}_k, \mathbf{w}_k \rangle$$
$$\propto \mathbf{R}\mathbf{a}_k \,.$$

where $\mathbf{R} \in \mathbb{R}^{N \times T}$ is the matrix of residuals with columns $[\mathbf{r}_1, \ldots, \mathbf{r}_T]$.

Fitting the model

Optimizing the amplitudes

As a function of $a_{k,t}$, the log joint probability is,

$$\log p(\mathbf{X}, \mathbf{W}, \mathbf{A}) = \frac{\mathbf{r}_t^\top \mathbf{w}_k a_{k,t}}{\sigma^2} - \frac{a_{k,t}^2}{2\sigma^2} - \lambda a_{k,t} + c'$$

This is a **quadratic optimization subject to a non-negativity constraint.**

Fitting the model

Generic solution

Assume $\alpha > 0$. Solve

$$\arg\max_{x \geq 0} \quad f(x) = -\frac{\alpha}{2}x^2 + \beta x + \gamma,$$

Fitting the model

Optimizing the amplitudes

By pattern matching to our problem, we have

$$a_{k,t}^{\star} = \max\left\{0, \sigma^2\left(\frac{\mathbf{r}_t^\top \mathbf{w}_k}{\sigma^2} - \lambda\right)\right\} = \max\left\{0, \mathbf{r}_t^\top \mathbf{w}_k - \lambda\sigma^2\right\}$$

$\mathbf{r}_t^\top \mathbf{w}_k$, is the **projection** of the residual onto the waveform for neuron k.

$\lambda\sigma^2$ the **threshold** that projection must exceed to designate a spike in amplitude.

The final algorithm

MAP estimation by coordinate ascent

- repeat until convergence:
  - for $k = 1, \ldots, K$:
    - Compute the residual $\mathbf{R} = \mathbf{X} - \sum_{j \neq k} \mathbf{w}_j \mathbf{a}_j^\top$
    - Set $\mathbf{w}_k \propto \mathbf{R}\mathbf{a}_k$
    - Set $\mathbf{a}_k = \max\{0, \mathbf{R}^\top \mathbf{w}_k - \lambda\sigma^2\}$

Note: You don't have to recompute the residual from scratch each iteration.

Conclusion

- We developed a basic spike sorting model that was good for building intuition, but not very practical.
- We derived a **coordinate ascent algorithm** for *maximum a posteriori* (MAP) inference, and that involved solving constrained optimization problems (over the unit sphere and the non-negative reals).
- **Next time**: you'll implement the algorithm in lab! You'll learn a bit of PyTorch for implementing the convolutions and cross-correlations, then test it out on the GPU.

Further reading

- **Simple Spike Sorting** and **Spike Sorting by Deconvolution** course notes.
- Convolution and cross-correlation:
    - Chapter 9 of *The Deep Learning Book* (deeplearningbook.org/contents/convnets.html)
    - Start reading up on PyTorch convolutions! https://pytorch.org/docs/stable/generated/torch.nn.functional.conv1d.html
- Spike sorting:
    - Pachitariu, Marius, Shashwat Sridhar, and Carsen Stringer. "Solving the spike sorting problem with Kilosort." bioRxiv (2023).
        - The model we presented is a slightly modified version of *Kilosort*