Title: Machine Learning Lecture - 230321

Speakers: Lauren Hayward

Collection: Machine Learning for Many-Body Physics (2022/2023)

Date: March 21, 2023 - 9:00 AM

URL: https://pirsa.org/23030034

Outline for today:

- dimensional reduction using t-distributed stochastic neighbour embedding (t-SNE)
- Kullback-Liebler (KL) divergence
- generative modelling, maximum likelihood estimation (MLE)

Recall the goal of dimensional reduction:

Dataset of M points

$\mathcal{D} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_M\}$

with $\vec{x}_i \in \mathbb{R}^N$

(often N is large)

dimensional reduction

$\mathcal{D}' = \{\vec{x}_1', \ldots, \vec{x}_M'\}$

with $\vec{x}_i' \in \mathbb{R}^{N'}$

(if $N' = 1, 2$ or 3 then we can visualize)

Last time: dim. reduction using PCA (linear method)

Dimensional reduction using t-SNE

In t-SNE, we use the Euclidean dist. between data points to define a prob. dist. called the "similarity"

$$p(j|i) = \frac{e^{-\|\vec{x}_i - \vec{x}_j\|/2\sigma_i^2}}{\sum_{k \neq i} e^{-\|\vec{x}_i - \vec{x}_k\|/2\sigma_i^2}} \quad (\text{for } i \neq j)$$

$p(i|i) = 0$

the prob. that $\vec{x}_i$ would choose $\vec{x}_j$ as its neighbour (if neighbours are chosen based on a Gaussian dist.)

the σ_i are free params

So $p(j|i)$ defines the "neighbourhood" of $\vec{x}_i$

Corresponding symmetric dist

$$p(i,j) = \frac{p(j|i) + p(i|j)}{2N}$$

(issues for outliers if the algorithm uses $p(j|i)$ instead of $p(i,j)$)

$p(i|i) = 0$

$$\sum_{k \neq i} e^{-\|x_i - x_j\|^2 / 2\sigma_i^2}$$

its neighbour (if neighbours are chosen based on a Gauss dist.)

Goal of t-SNE: find a lower-dim. repr. $\vec{x}' \in \mathbb{R}^{N'}$ that preserves the similarity dist.

For the low-dim. data, define the similarity as:

$$q(i,j) = \frac{f(\|\vec{x}_i' - \vec{x}_j'\|)}{\sum_{k \neq l} f(\|\vec{x}_l' - \vec{x}_k'\|)}$$

where $f(z) = \frac{1}{1+z^2}$ is the t-Student dist.

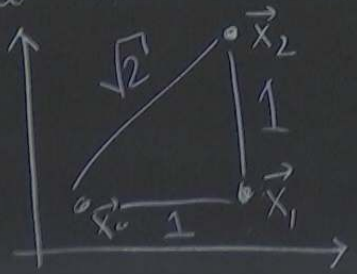
$q(i,i) = 0$

n a Gaussian

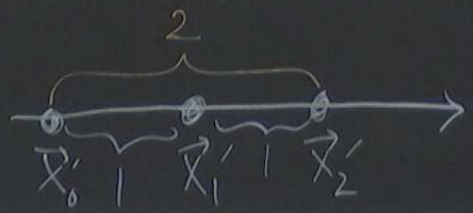
As we will see, t-SNE prioritizes preserving short-dist. information so that points close in the orig. space tend to still be close

Why use t-Student?

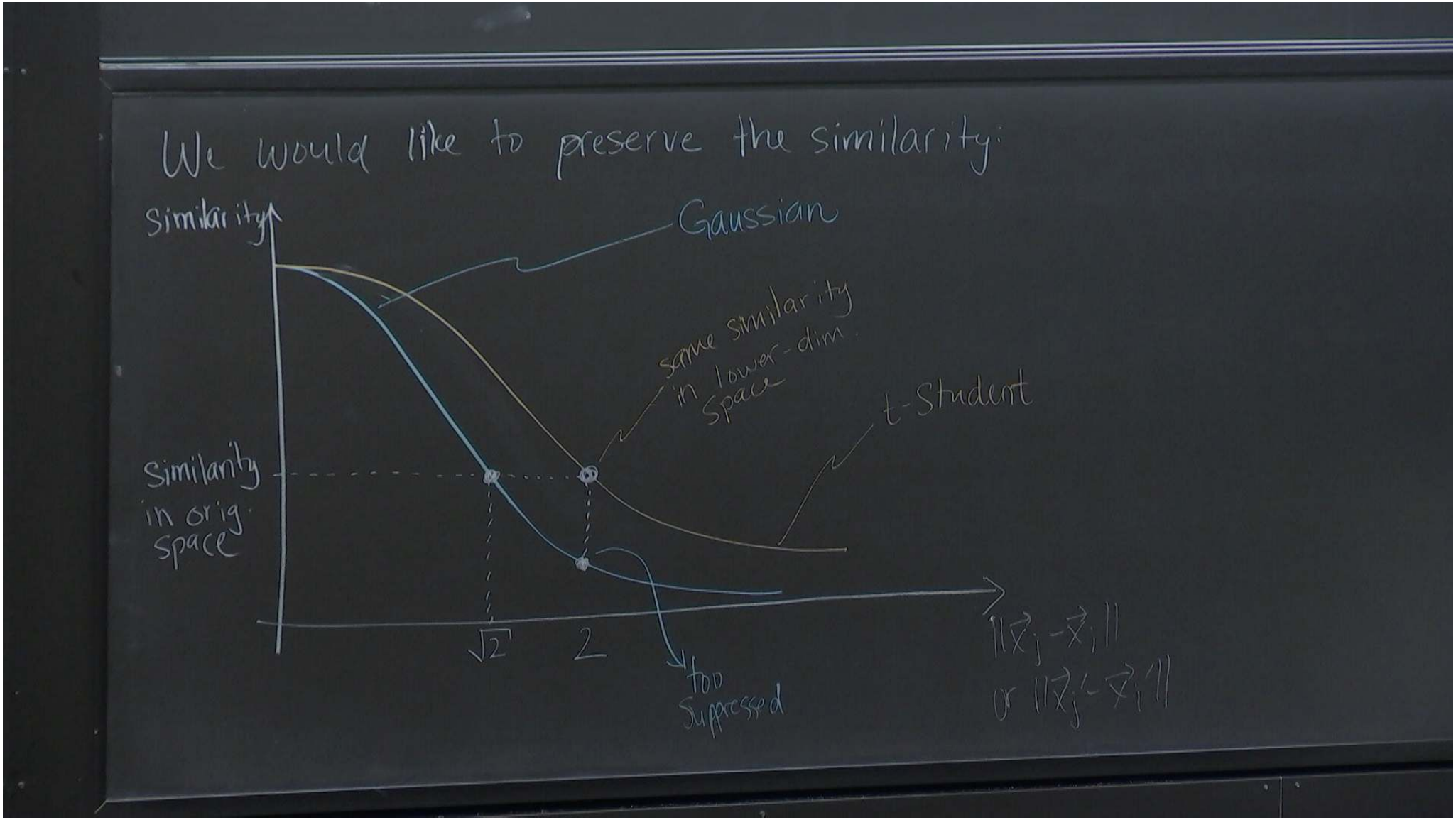
Consider dim. reduction from $N=2$ to $N'=1$



dim. reduction w short dist. preserved



Need to consider the "crowding problem"


We would like to preserve the similarity:
Similarity
Gaussian
same similarity in lower-dim space
t-Student
Similarity in orig. space
√2
2
too suppressed
$\|\vec{x}_j - \vec{x}_i\|$
or $\|\vec{x}_j' - \vec{x}_i'\|$

- If $q(i,j)$ used the same Gaussian dist. as $p(i,j)$, then $q(0,2)$ would be much smaller than $p(0,2)$

- Since the t-Student dist. has a longer tail, we can better preserve the similarity

To find the lower-dim rep. $\mathcal{D}' = \{\vec{x}'_i\}$, iteratively update the points $\vec{x}'_i$ using an algorithm such as gradient descent, with cost func.

$$C(p,q) = D_{KL}(p|q)$$
$$= \sum_{i,j} p(i,j)\left[\log \frac{p(i,j)}{q(i,j)}\right]$$

} Kullback-Liebler (KL) divergence

Additional notes:

- $p(i,j)$ is fixed (given by dataset). We adjust the points $\vec{x}_i'$ so $q(i,j)$ is as close as possible to $p(i,j)$
- We prioritize preserving local structure since $p(i,j)$ acts as a "weight" in the sum $\sum_{i,j}'$
- The resulting lower-dim. rep. is harder to interpret than in PCA

Generative modelling

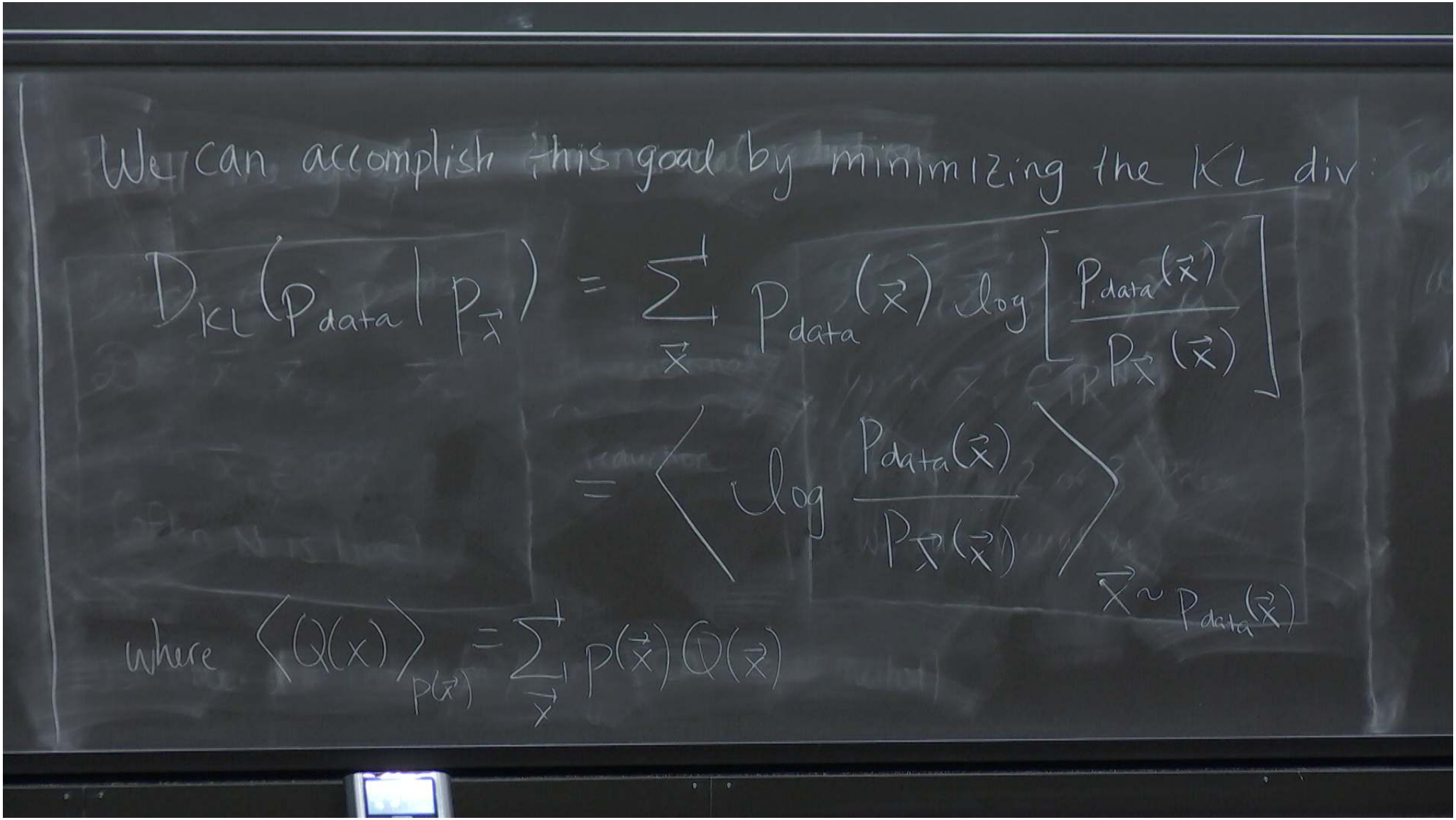
Given: datapoints $\vec{x}_1, \ldots, \vec{x}_M$ with underlying $P_{data}(\vec{x})$

$P_{data}(\vec{x})$ ← usually unknown

Goal: tune the params. $\vec{\lambda}$ of a model such that $P_{\vec{\lambda}}(\vec{x})$ is close to $P_{data}(\vec{x})$

$P_{\vec{\lambda}}(\vec{x})$: model dist.

$P_{data}(\vec{x})$: underlying data dist.


We can accomplish this goal by minimizing the KL div:
$D_{KL}(P_{data} \| P_{\vec{x}}) = \sum_{\vec{x}} P_{data}(\vec{x}) \log \left[\frac{P_{data}(\vec{x})}{P_{\vec{x}}(\vec{x})} \right]$
$= \left\langle \log \frac{P_{data}(\vec{x})}{P_{\vec{x}}(\vec{x})} \right\rangle_{\vec{x} \sim P_{data}(\vec{x})}$
where $\langle Q(x) \rangle_{P(\vec{x})} = \sum_{\vec{x}} P(\vec{x}) Q(\vec{x})$

model dist. underlying data dist.

We can also state the problem of generative modelling in terms of

Maximum Likelihood Estimation (MLE)

Idea: choose the model's parameters $\vec{\lambda}$ such that we maximize $P_{\vec{\lambda}}(\vec{x})$ for the M given datapoints

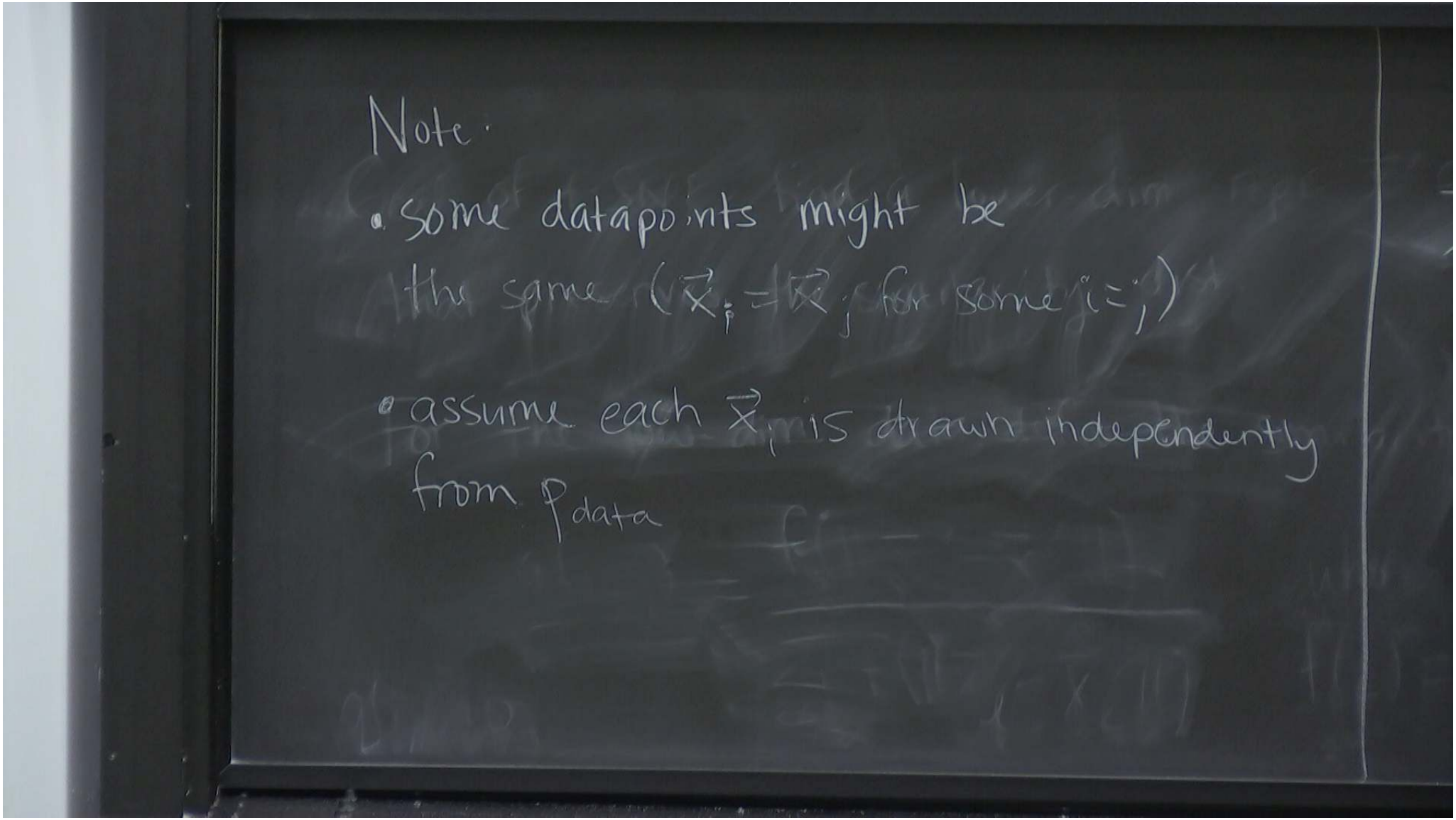
We could take:

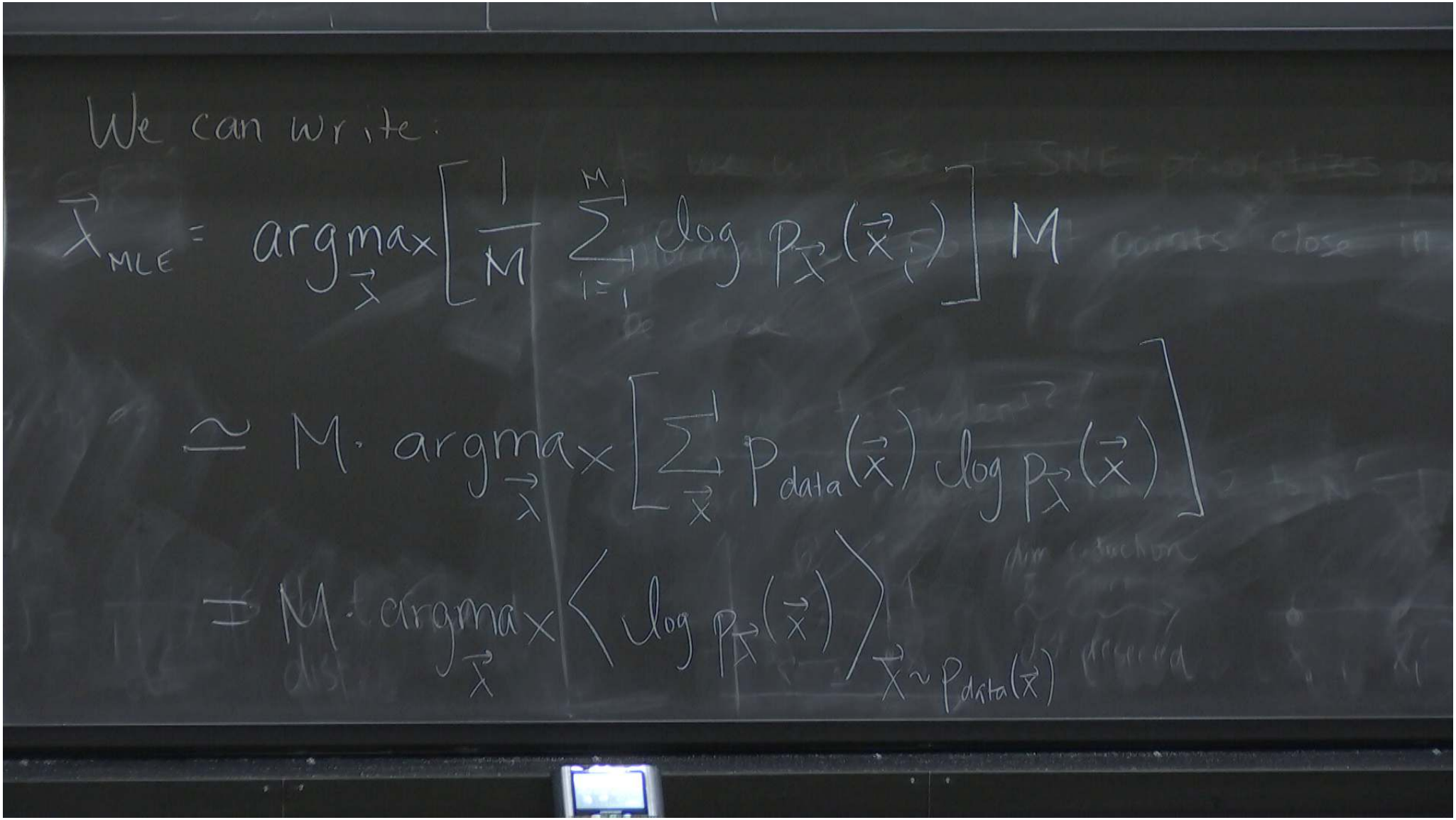
$$\vec{\lambda}_{MLE} = \underset{\vec{\lambda}}{\operatorname{argmax}} \left[\prod_{i=1}^{M} P_{\vec{\lambda}}(\vec{x}_i) \right]$$

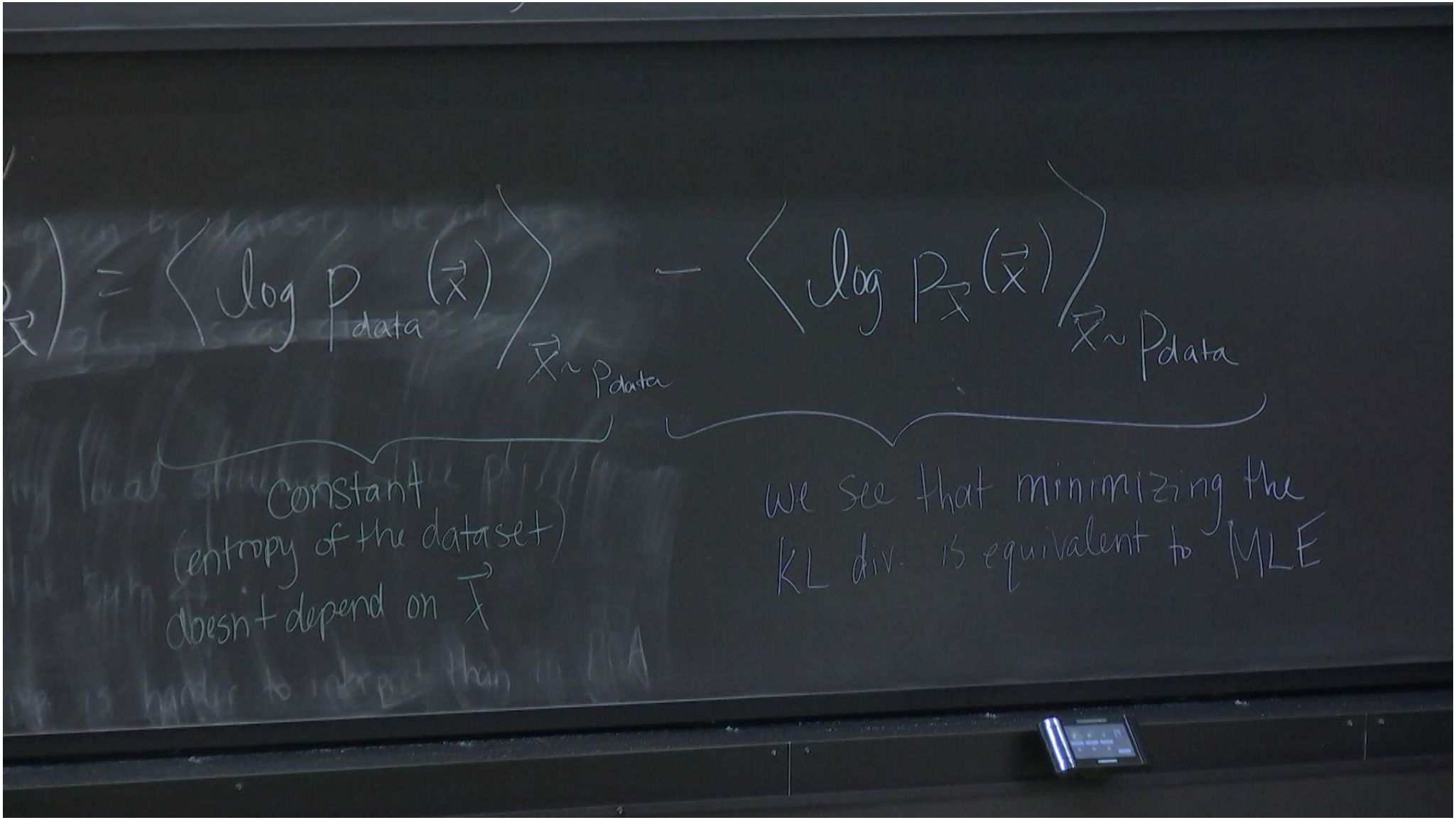
multiplying many probabilities (all ≤ 1) can be numerically unstable

Instead let's use:

$$\vec{\lambda}_{MLE} = \underset{\vec{\lambda}}{\operatorname{argmax}} \left[\log \left(\prod_{i=1}^{M} P_{\vec{\lambda}}(\vec{x}_i) \right) \right] = \underset{\vec{\lambda}}{\operatorname{argmax}} \left[\sum_{i=1}^{M} \log P_{\vec{\lambda}}(\vec{x}) \right]$$


Note:
• some datapoints might be the same ($\vec{x}_i = \vec{x}_j$ for some $i \neq j$)
• assume each $\vec{x}_i$ is drawn independently from p_{data}


We can write:
$\vec{x}_{MLE} = \underset{\vec{x}}{\text{argmax}} \left[\frac{1}{M} \sum_{i=1}^{M} \log P_{\vec{x}}(\vec{x}_i) \right] M$
$\simeq M \cdot \underset{\vec{x}}{\text{argmax}} \left[\sum_{\vec{x}} P_{data}(\vec{x}) \log P_{\vec{x}}(\vec{x}) \right]$
$= M \cdot \underset{\vec{x}}{\text{argmax}} \left\langle \log P_{\vec{x}}(\vec{x}) \right\rangle_{\vec{x} \sim P_{data}(\vec{x})}$


$\vec{x}) = \langle \log p_{data}(\vec{x}) \rangle_{\vec{x} \sim p_{data}} - \langle \log p_{\vec{x}}(\vec{x}) \rangle_{\vec{x} \sim p_{data}}$
constant
(entropy of the dataset)
doesn't depend on $\vec{x}$
we see that minimizing the
KL div. is equivalent to MLE