Title: Machine Learning (2021/2022)

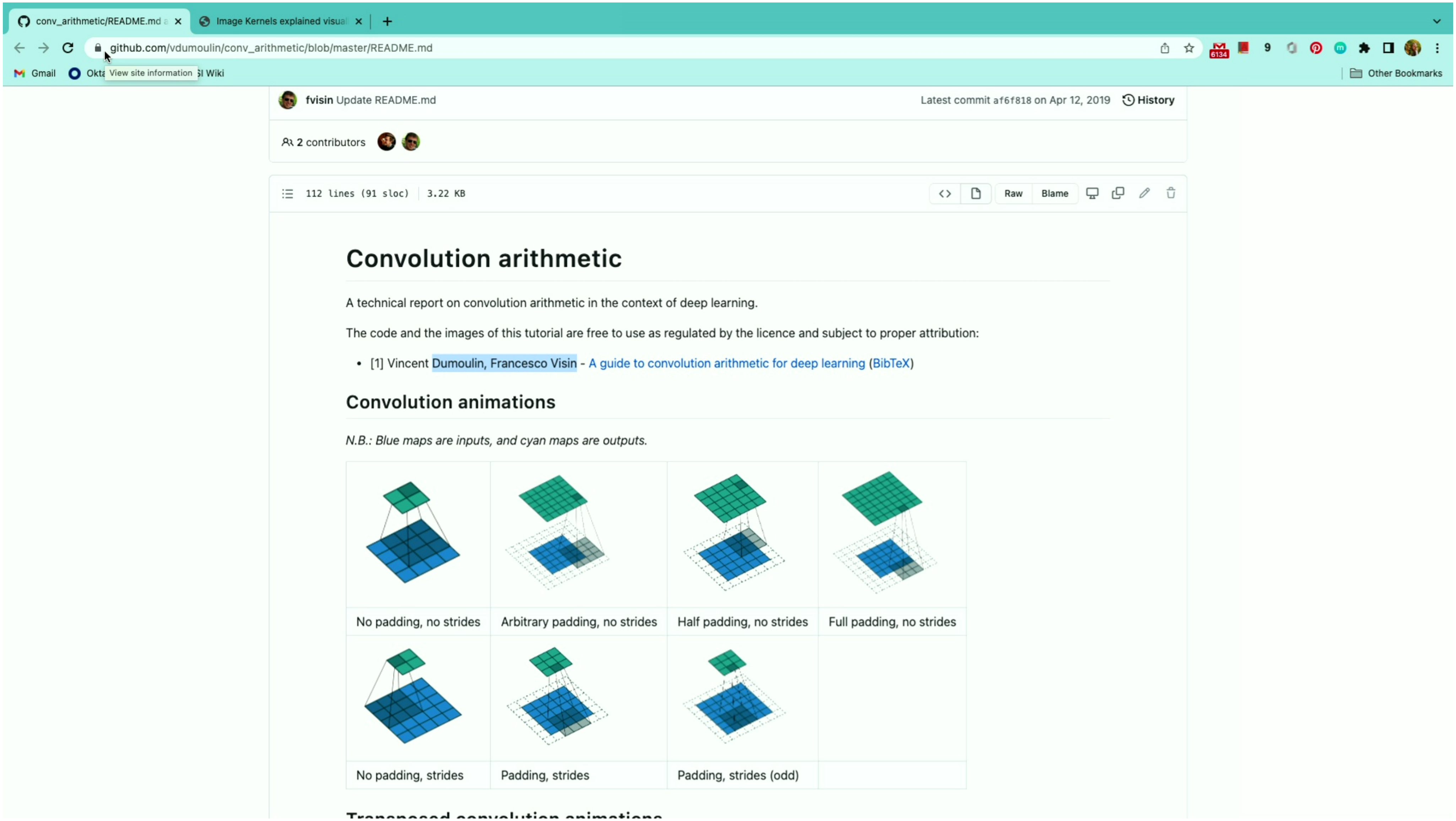
Speakers: Lauren Hayward

Collection: Machine Learning (2021/2022)

Date: April 21, 2022 - 11:30 AM

URL: https://pirsa.org/22040072

Abstract: This course is designed to introduce modern machine learning techniques for studying classical and quantum many-body problems encountered in condensed matter, quantum information, and related fields of physics. Lectures will focus on introducing machine learning algorithms and discussing how they can be applied to solve problem in statistical physics. Tutorials and homework assignments will concentrate on developing programming skills to study the problems presented in lecture.


conv_arithmetic/README.md
Image Kernels explained visual
github.com/vdumoulin/conv_arithmetic/blob/master/README.md
Gmail
Okta
View site information
Wiki
Other Bookmarks
fvisin Update README.md
Latest commit af6f818 on Apr 12, 2019
History
2 contributors
112 lines (91 sloc)
3.22 KB
Raw
Blame

Convolution arithmetic

A technical report on convolution arithmetic in the context of deep learning.

The code and the images of this tutorial are free to use as regulated by the licence and subject to proper attribution:

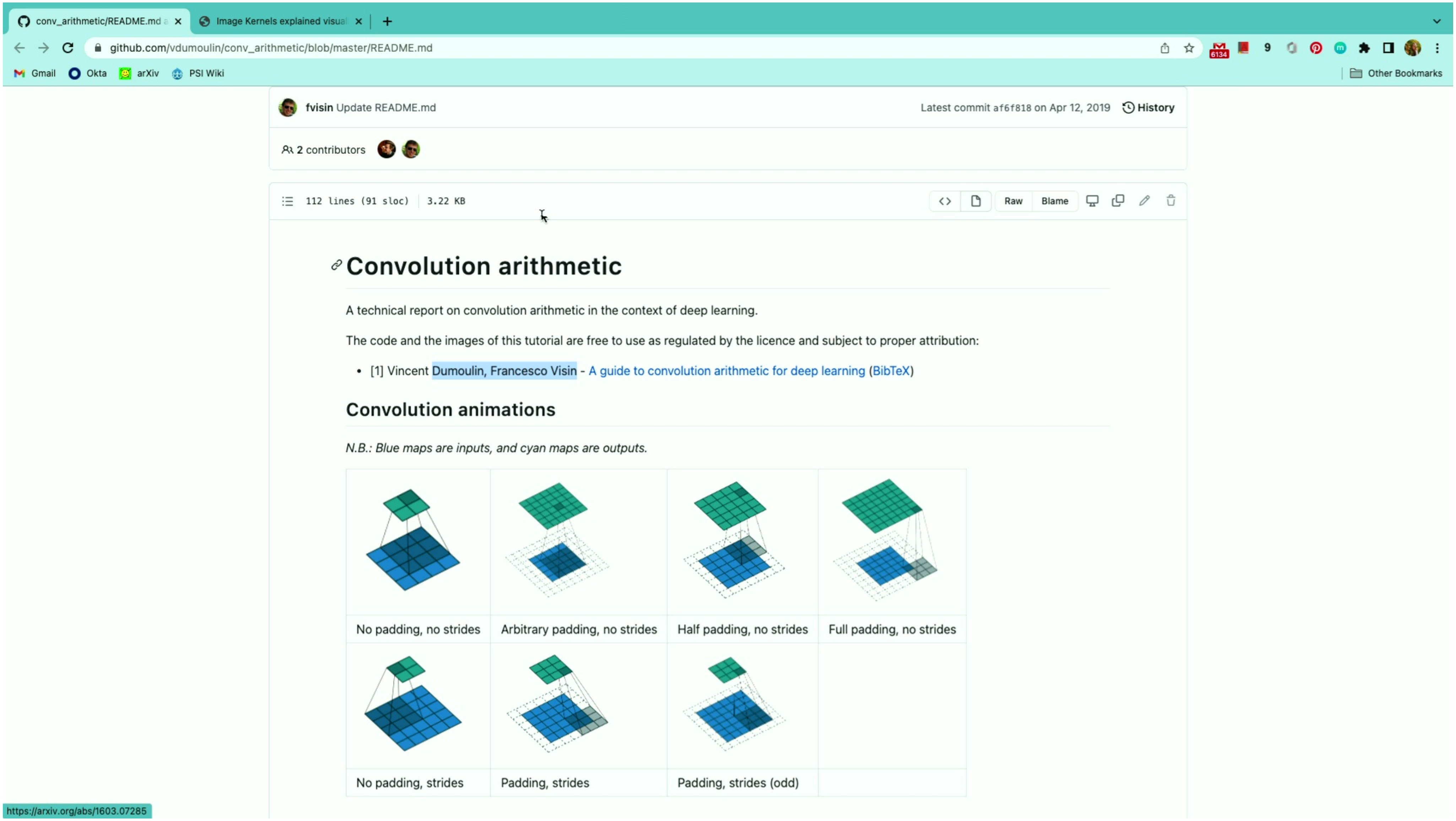
- [1] Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning (BibTeX)

Convolution animations

N.B.: Blue maps are inputs, and cyan maps are outputs.

No padding, no strides	Arbitrary padding, no strides	Half padding, no strides	Full padding, no strides
No padding, strides	Padding, strides	Padding, strides (odd)	

Transposed convolution animations


conv_arithmetic/README.md
Image Kernels explained visual
github.com/vdumoulin/conv_arithmetic/blob/master/README.md
Gmail
Okta
arXiv
PSI Wiki
Other Bookmarks
fvisin Update README.md
Latest commit af6f818 on Apr 12, 2019
History
2 contributors
112 lines (91 sloc) | 3.22 KB
Raw
Blame
https://arxiv.org/abs/1603.07285

Convolution arithmetic

A technical report on convolution arithmetic in the context of deep learning.

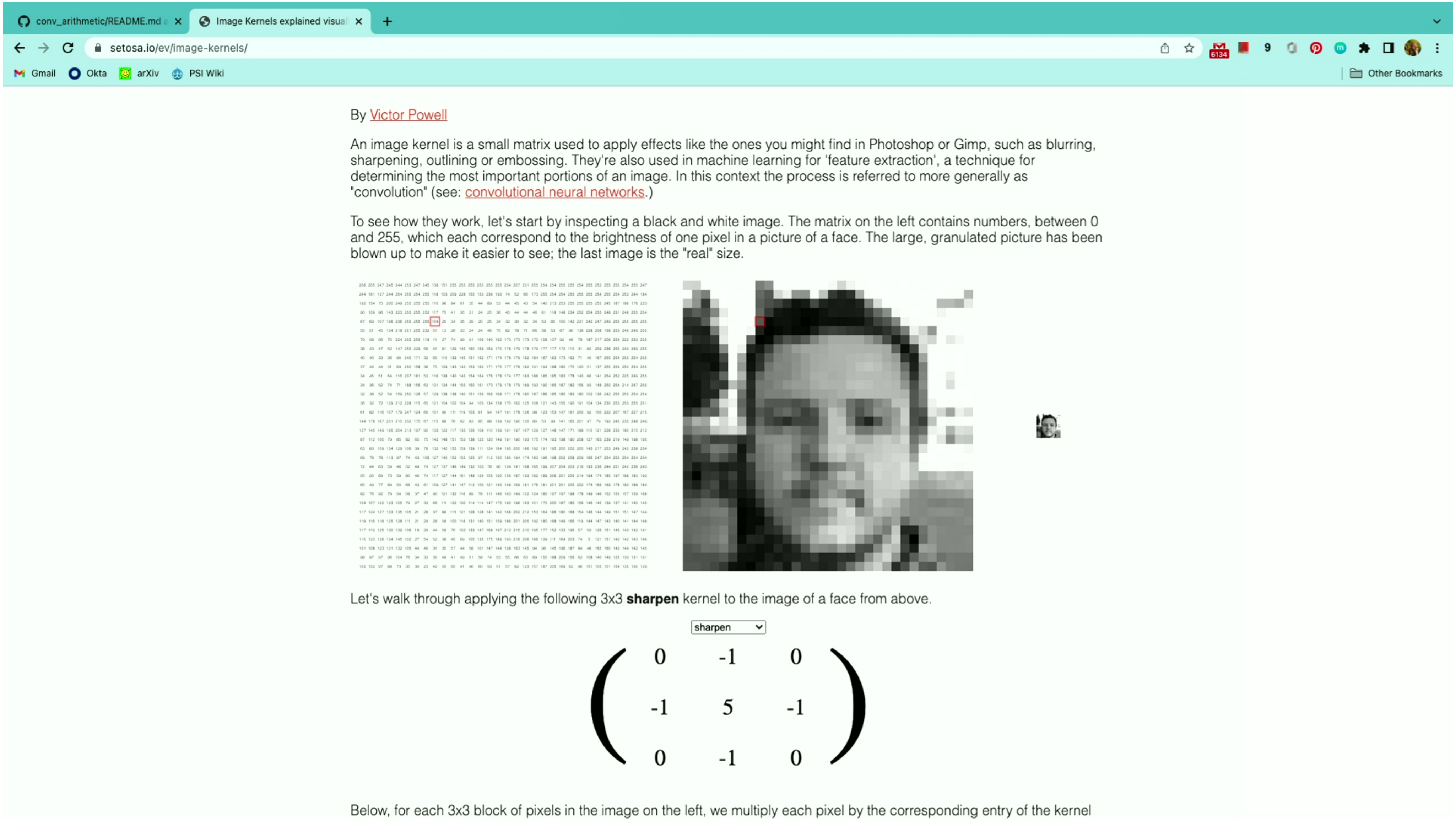
The code and the images of this tutorial are free to use as regulated by the licence and subject to proper attribution:

- [1] Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning (BibTeX)

Convolution animations

N.B.: Blue maps are inputs, and cyan maps are outputs.

No padding, no strides	Arbitrary padding, no strides	Half padding, no strides	Full padding, no strides
No padding, strides	Padding, strides	Padding, strides (odd)	



conv_arithmetic/README.md
Image Kernels explained visual
setosa.io/ev/image-kernels/
Gmail
Okta
arXiv
PSI Wiki
Other Bookmarks

By Victor Powell

An image kernel is a small matrix used to apply effects like the ones you might find in Photoshop or Gimp, such as blurring, sharpening, outlining or embossing. They're also used in machine learning for 'feature extraction', a technique for determining the most important portions of an image. In this context the process is referred to more generally as "convolution" (see: convolutional neural networks.)

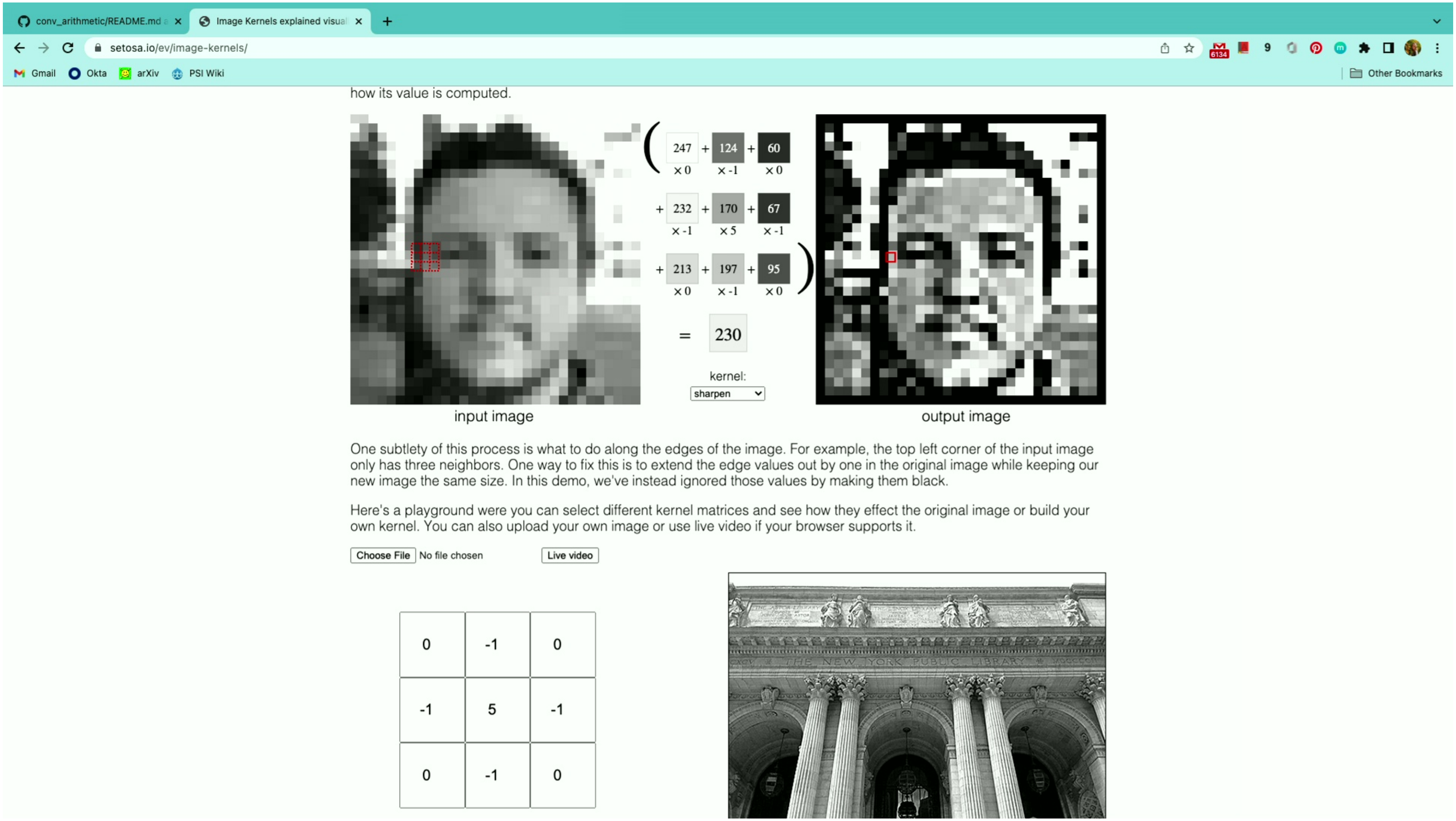
To see how they work, let's start by inspecting a black and white image. The matrix on the left contains numbers, between 0 and 255, which each correspond to the brightness of one pixel in a picture of a face. The large, granulated picture has been blown up to make it easier to see; the last image is the "real" size.

Let's walk through applying the following 3x3 **sharpen** kernel to the image of a face from above.

sharpen

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel


conv_arithmetic/README.md
Image Kernels explained visual
setosa.io/ev/image-kernels/
Gmail
Okta
arXiv
PSI Wiki
Other Bookmarks

how its value is computed.

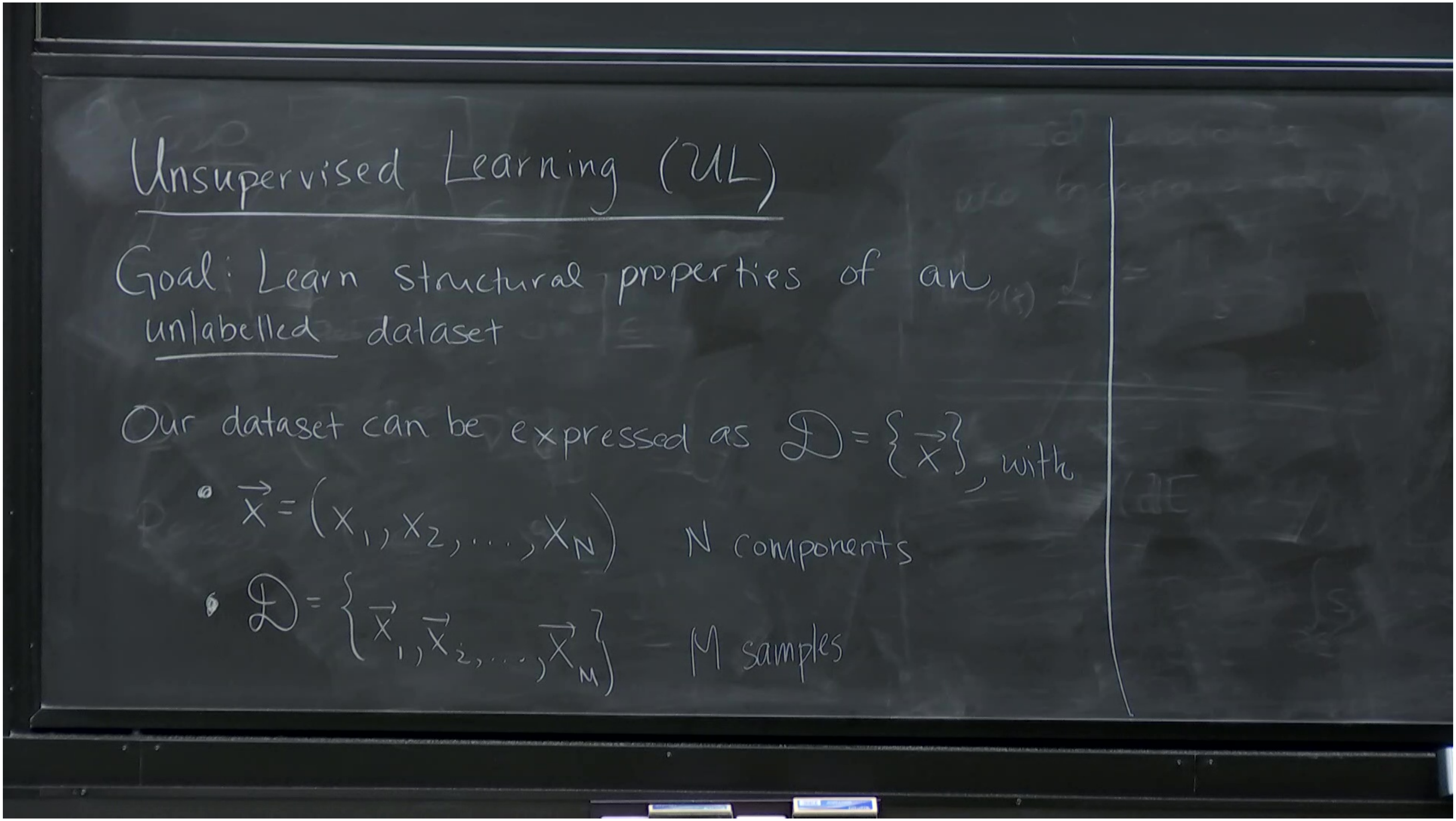
$$\left(\begin{array}{ccc} 247 \times 0 & + 124 \times -1 & + 60 \times 0 \\ + 232 \times -1 & + 170 \times 5 & + 67 \times -1 \\ + 213 \times 0 & + 197 \times -1 & + 95 \times 0 \end{array} \right) = 230$$

kernel: sharpen

One subtlety of this process is what to do along the edges of the image. For example, the top left corner of the input image only has three neighbors. One way to fix this is to extend the edge values out by one in the original image while keeping our new image the same size. In this demo, we've instead ignored those values by making them black.

Here's a playground were you can select different kernel matrices and see how they effect the original image or build your own kernel. You can also upload your own image or use live video if your browser supports it.

0	-1	0
-1	5	-1
0	-1	0

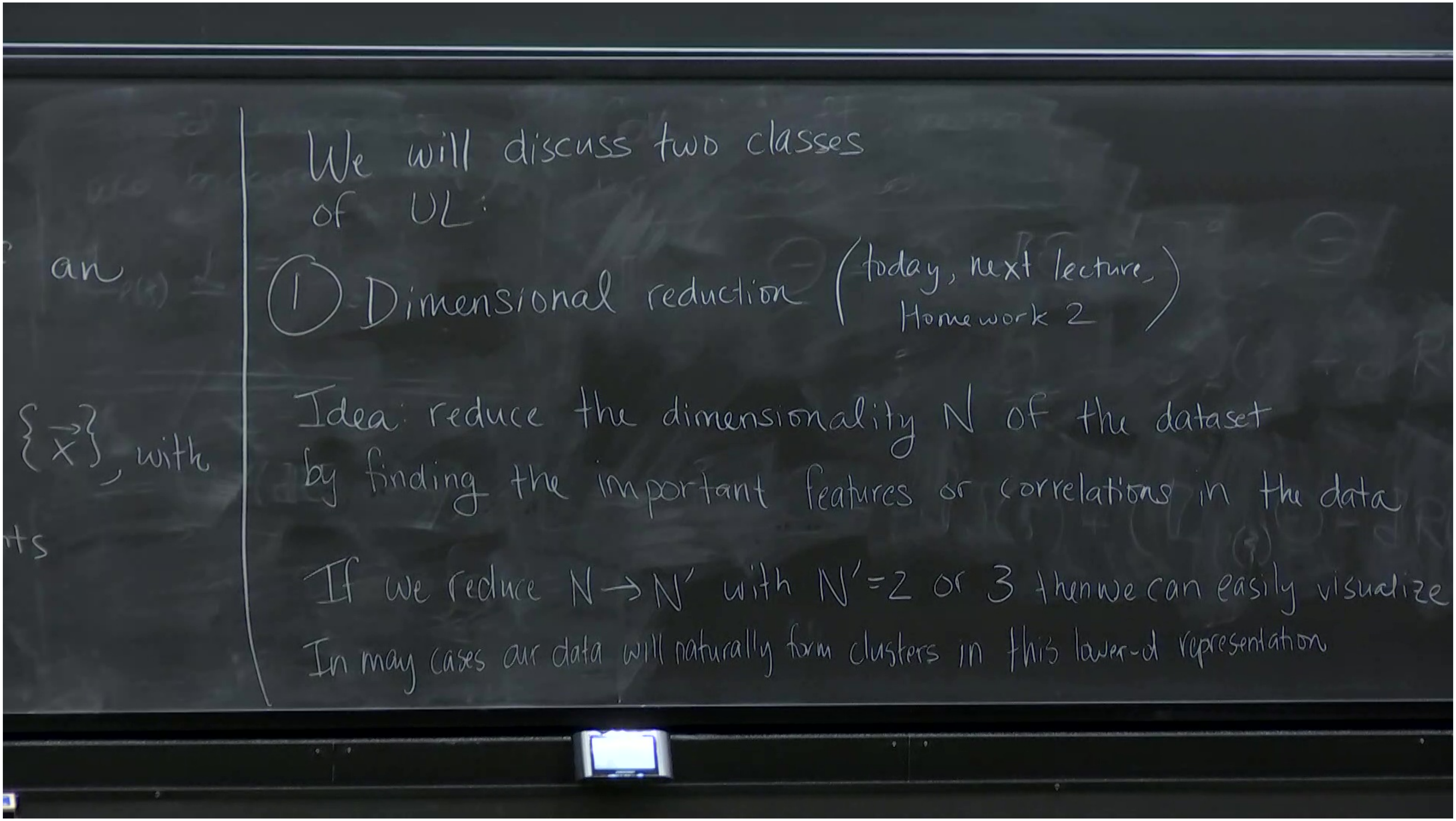


Unsupervised Learning (UL)

Goal: Learn structural properties of an unlabelled dataset

Our dataset can be expressed as $\mathcal{D} = \{\vec{x}\}$, with

- $\vec{x} = (x_1, x_2, \dots, x_N)$ N components
- $\mathcal{D} = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_M\}$ — M samples



an

$\{\vec{x}\}$, with

ts

We will discuss two classes of UL:

① Dimensional reduction (today, next lecture, Homework 2)

Idea: reduce the dimensionality N of the dataset by finding the important features or correlations in the data

If we reduce $N \to N'$ with $N' = 2$ or 3 then we can easily visualize

In may cases our data will naturally form clusters in this lower-d representation

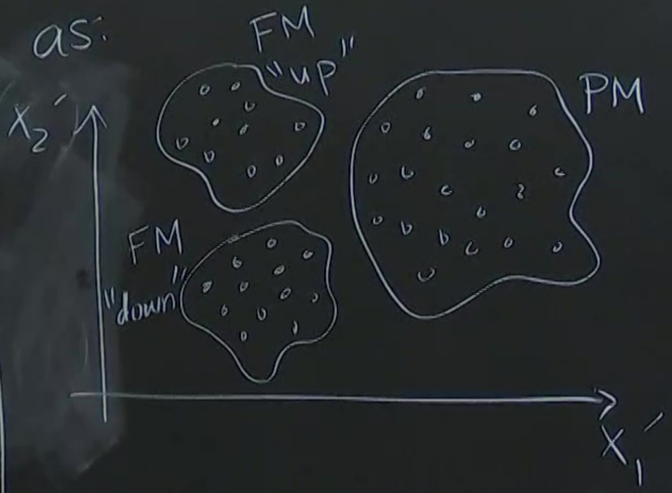
(today, next lecture
Homework 2)

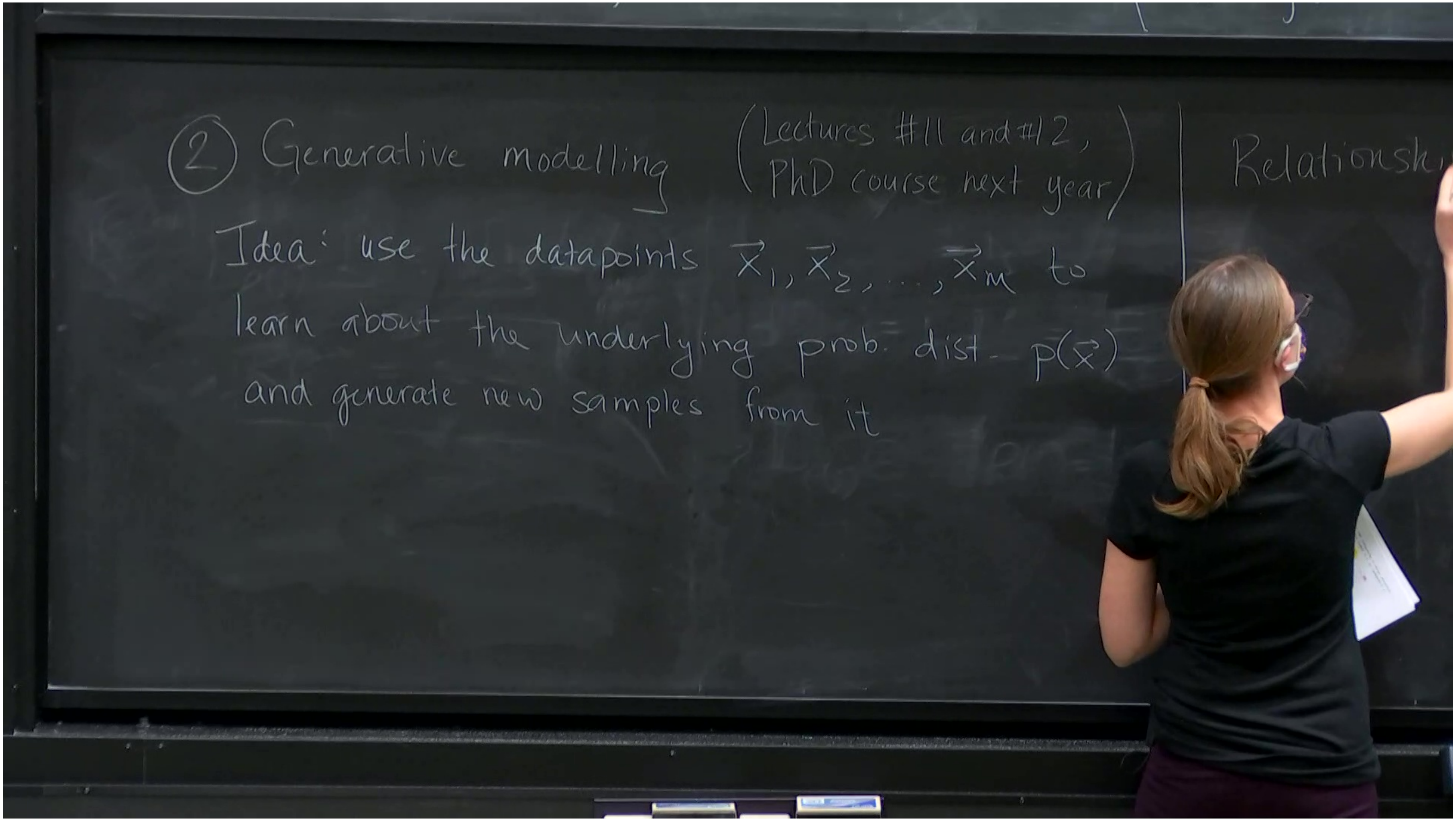
lity N of the dataset

ures or correlations in the data

=2 or 3 then we can easily visualize
clusters in this lower-d representation

eg) Ising model configs at various temperatures might form clusters in a 2D representation as:

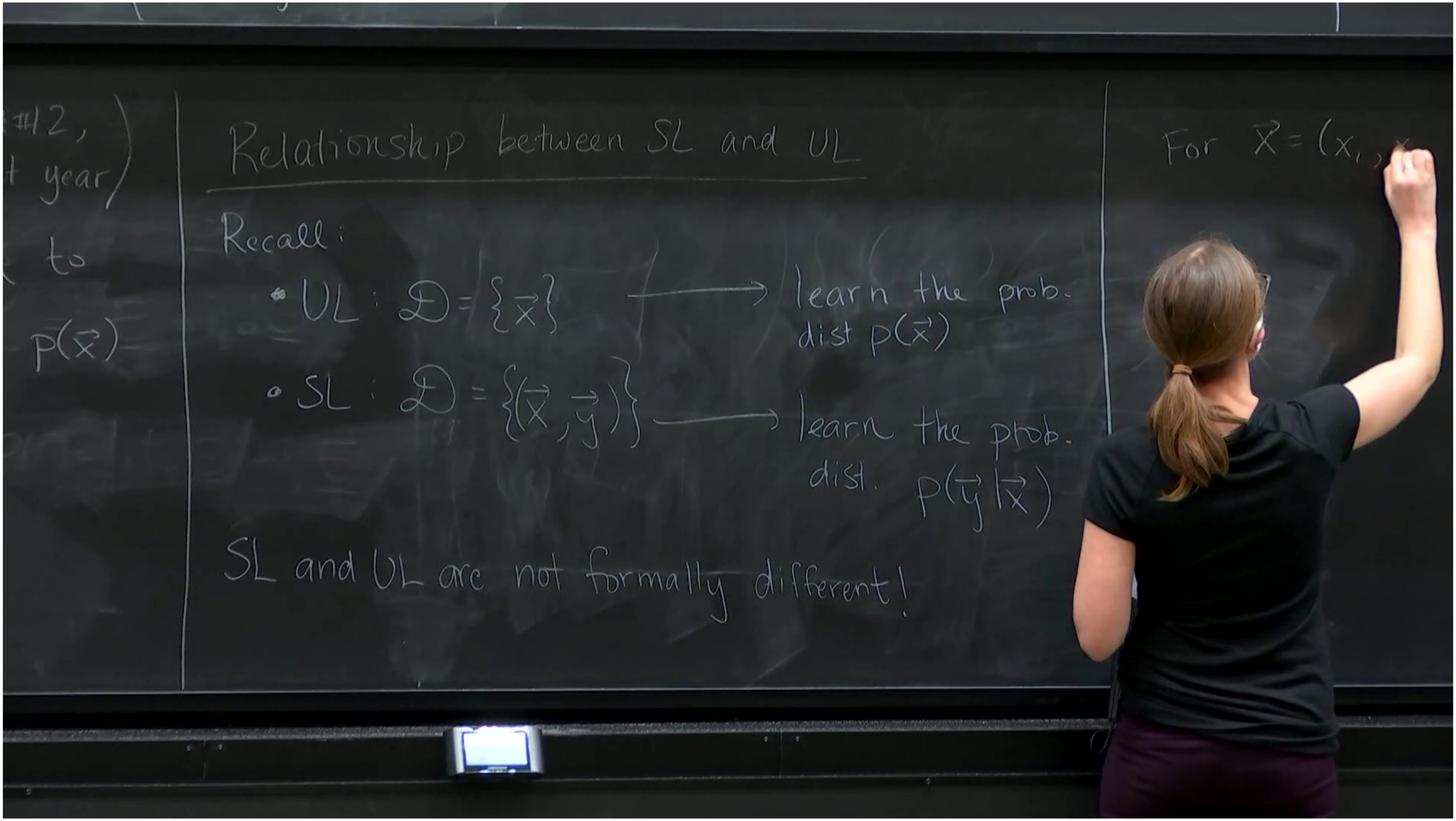




② Generative modelling (Lectures #11 and #12, PhD course next year)

Idea: use the datapoints $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_M$ to learn about the underlying prob. dist. $p(\vec{x})$ and generate new samples from it

Relationsh



#2,
t year)

to

$p(\vec{x})$

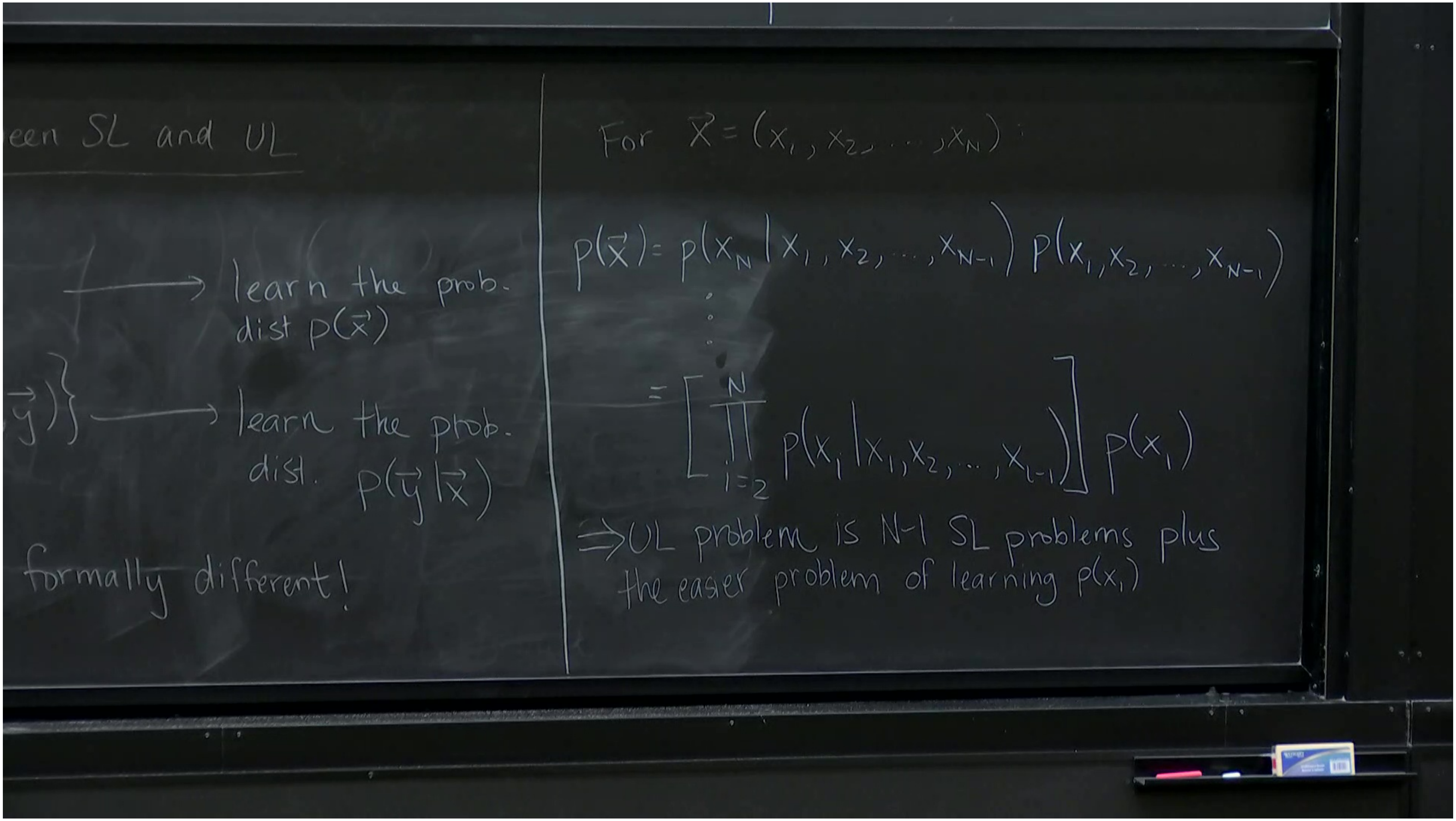
Relationship between SL and UL

Recall:

- UL: $\mathcal{D} = \{\vec{x}\}$ → learn the prob. dist $p(\vec{x})$
- SL: $\mathcal{D} = \{(\vec{x}, \vec{y})\}$ → learn the prob. dist. $p(\vec{y}|\vec{x})$

SL and UL are not formally different!

For $\vec{x} = (x_1, x$



een SL and UL

$\longrightarrow$ learn the prob. dist $p(\vec{x})$

$\vec{y})\} \longrightarrow$ learn the prob. dist. $p(\vec{y}|\vec{x})$

formally different!

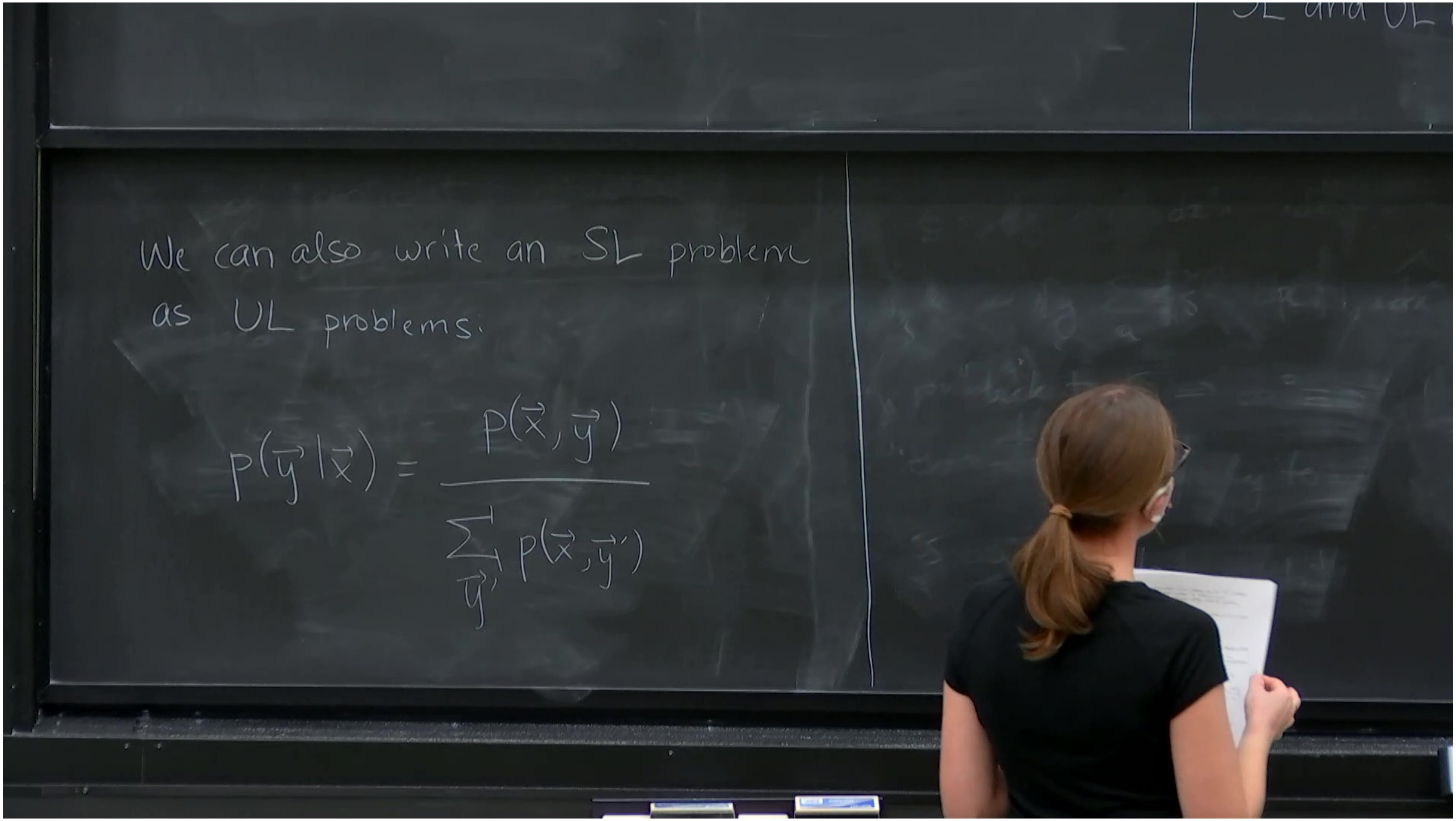
For $\vec{X} = (x_1, x_2, \ldots, x_N)$:

$$p(\vec{x}) = p(x_N | x_1, x_2, \ldots, x_{N-1})\, p(x_1, x_2, \ldots, x_{N-1})$$

$$\vdots$$

$$= \left[\prod_{i=2}^{N} p(x_i | x_1, x_2, \ldots, x_{i-1})\right] p(x_1)$$

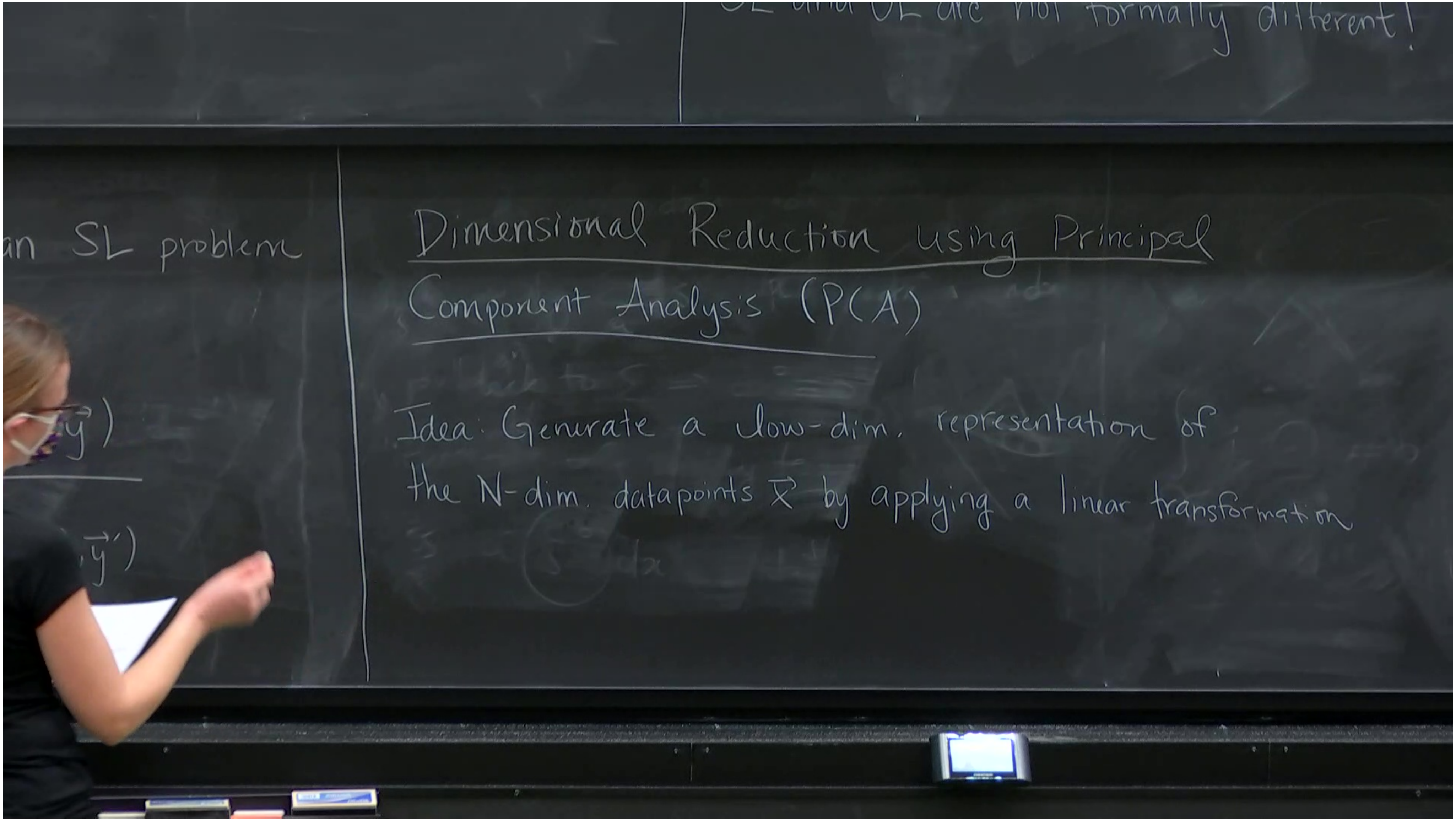
$\Rightarrow$ UL problem is N-1 SL problems plus the easier problem of learning $p(x_1)$



SL and UL

We can also write an SL problem as UL problems:

$$p(\vec{y}\,|\,\vec{x}) = \frac{p(\vec{x},\vec{y})}{\sum_{\vec{y}'} p(\vec{x},\vec{y}')}$$



SL and UL are not formally different!

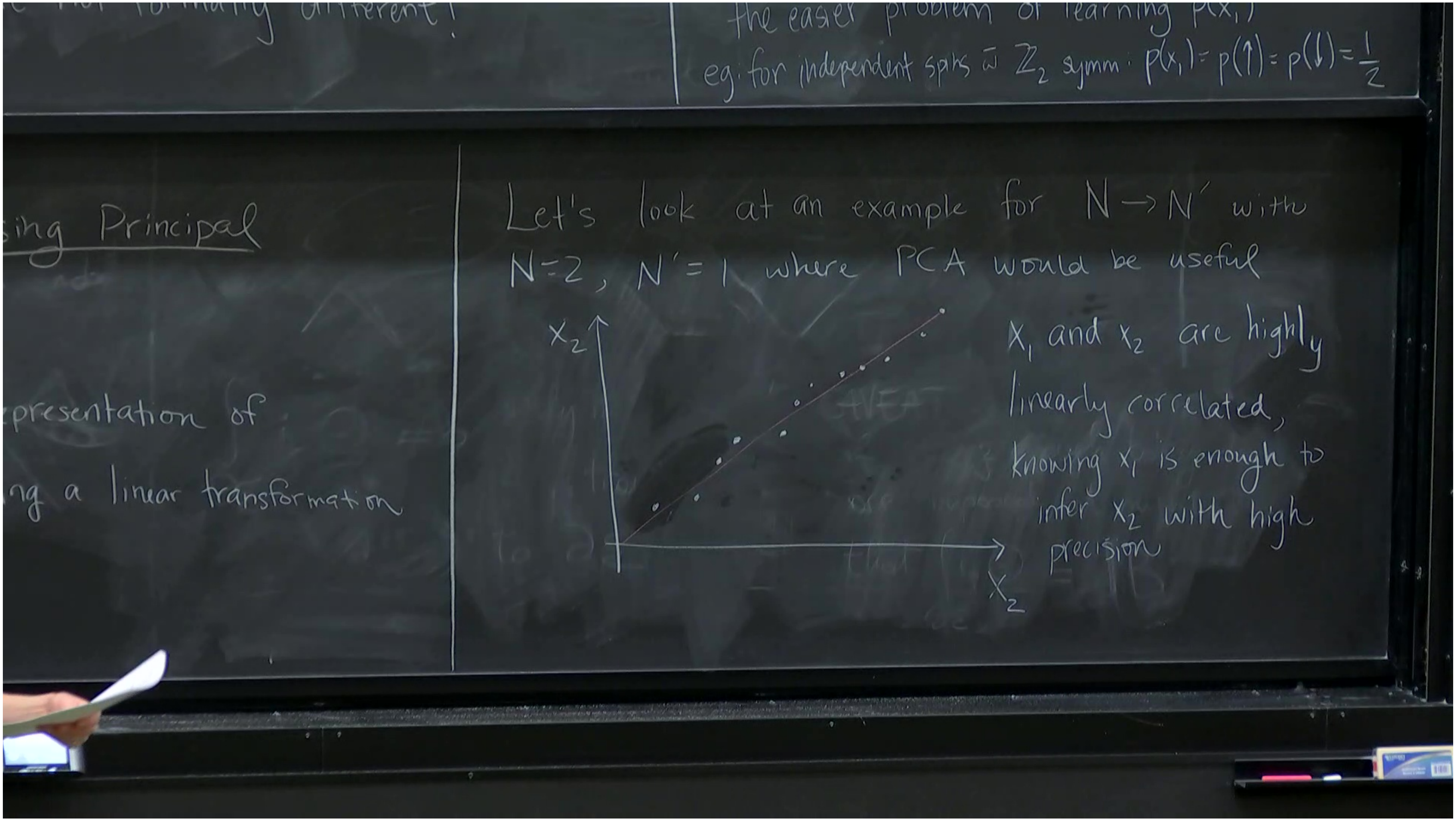
an SL problem

$\vec{y}$)

$\vec{y}'$)

Dimensional Reduction using Principal Component Analysis (PCA)

Idea: Generate a low-dim. representation of the N-dim. datapoints $\vec{x}$ by applying a linear transformation



e not formally different!

the easier problem of learning $p(x_1)$

eg: for independent spins $\bar{w}$ $\mathbb{Z}_2$ symm: $p(x_1) = p(\uparrow) = p(\downarrow) = \frac{1}{2}$

sing Principal

presentation of

ng a linear transformation

Let's look at an example for $N \to N'$ with $N=2$, $N'=1$ where PCA would be useful

x_1 and x_2 are highly linearly correlated, knowing x_1 is enough to infer x_2 with high precision e not formally different!

the easier problem of learning $p(x_1)$

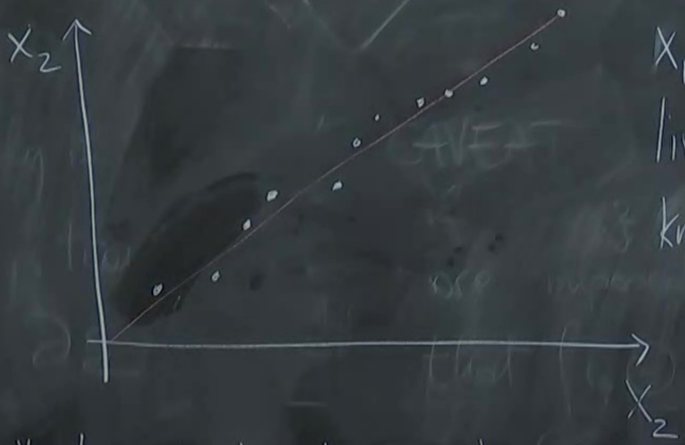
eg: for independent spins w/ $\mathbb{Z}_2$ symm: $p(x_1) = p(\uparrow) = p(\downarrow) = \frac{1}{2}$

sing Principal

presentation of

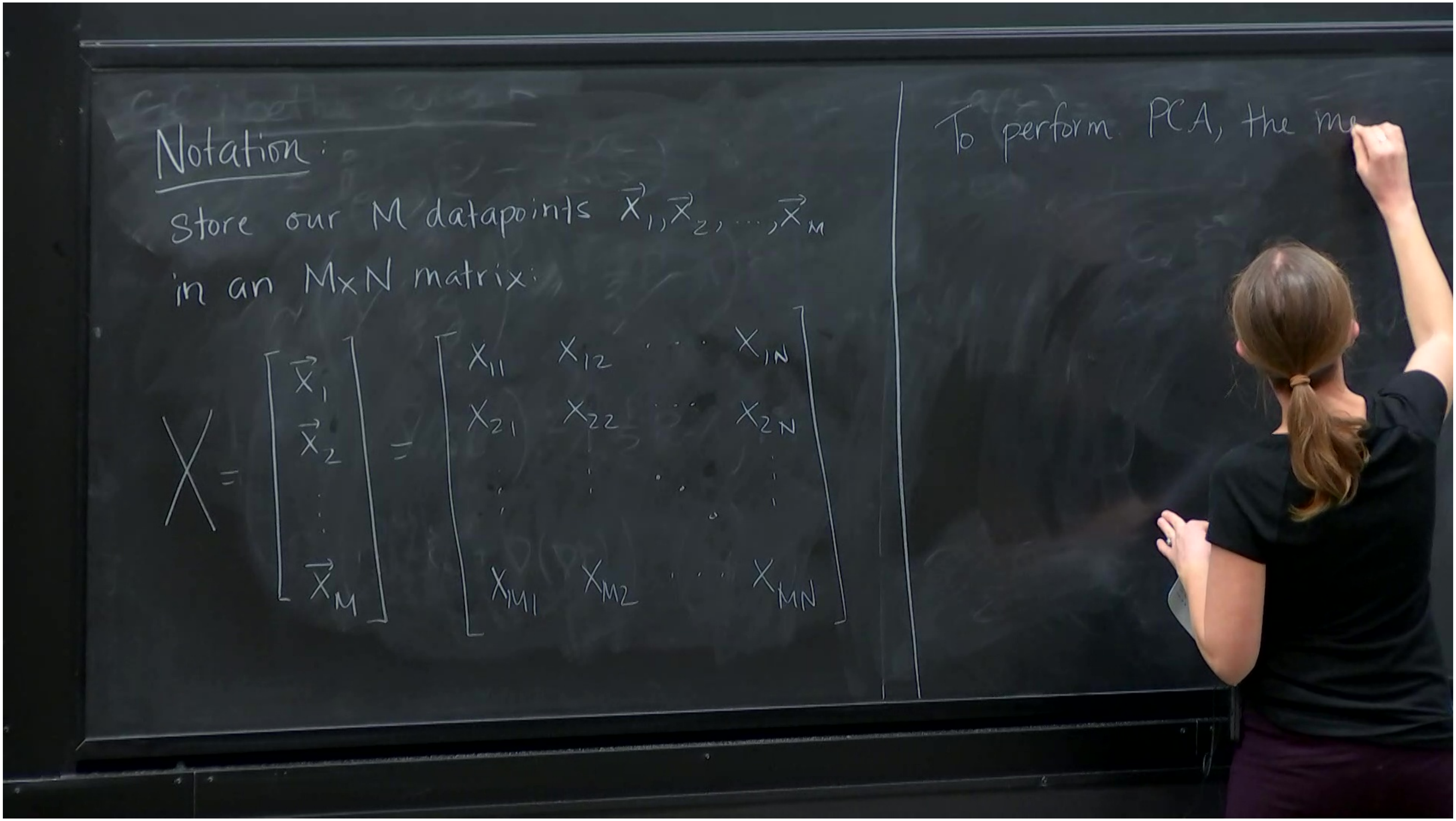
ng a linear transformation

Let's look at an example for $N \to N'$ with $N=2$, $N'=1$ where PCA would be useful



x_1 and x_2 are highly linearly correlated, knowing x_1 is enough to infer x_2 with high precision

(We will also encounter situations where PCA is not useful!)

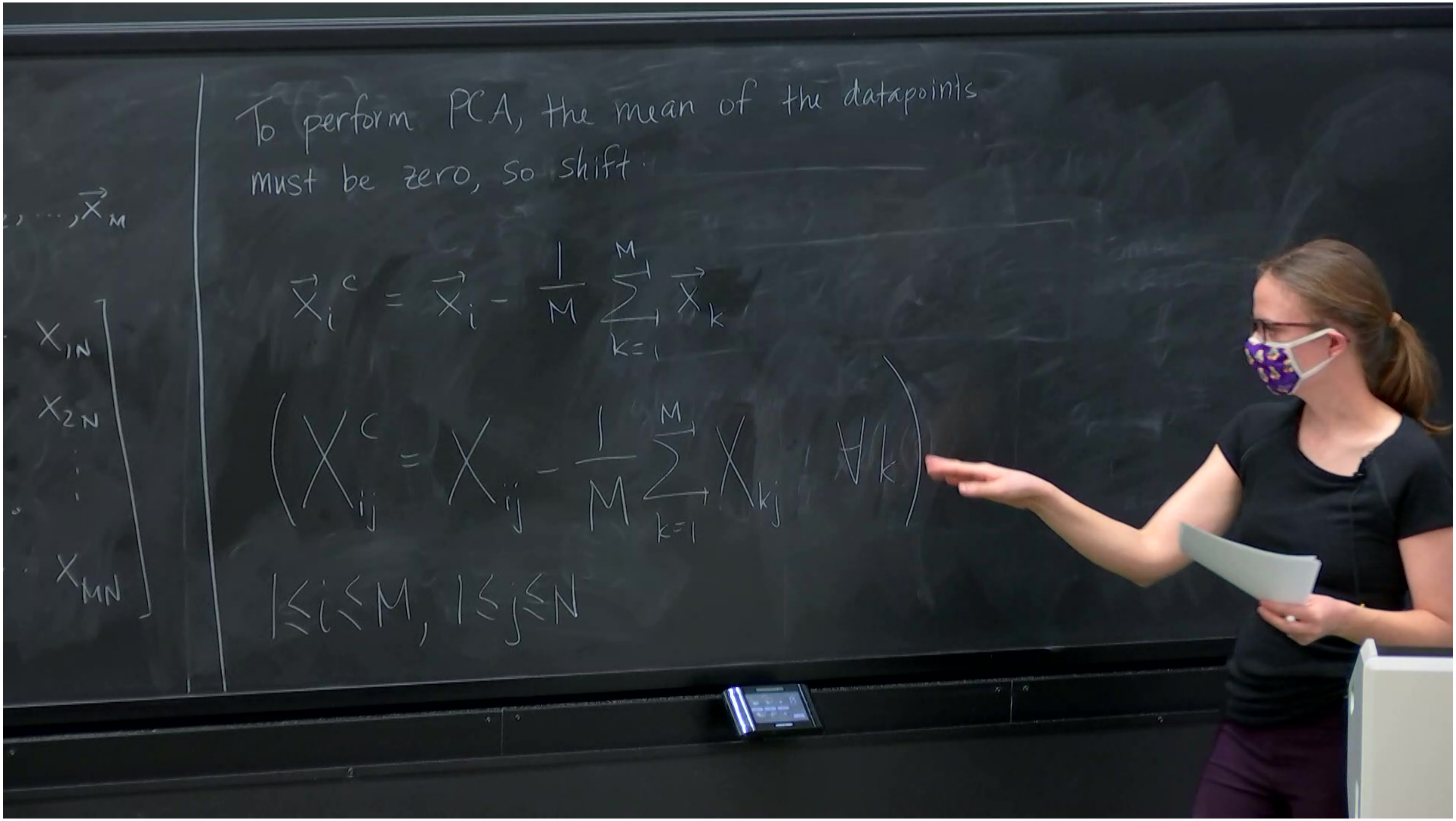


Notation:

Store our M datapoints $\vec{x}_1, \vec{x}_2, \dots, \vec{x}_M$ in an M×N matrix:

$$X = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_M \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix}$$

To perform PCA, the me



$\ldots, \vec{x}_M$

X_{1N}

X_{2N}

$\vdots$

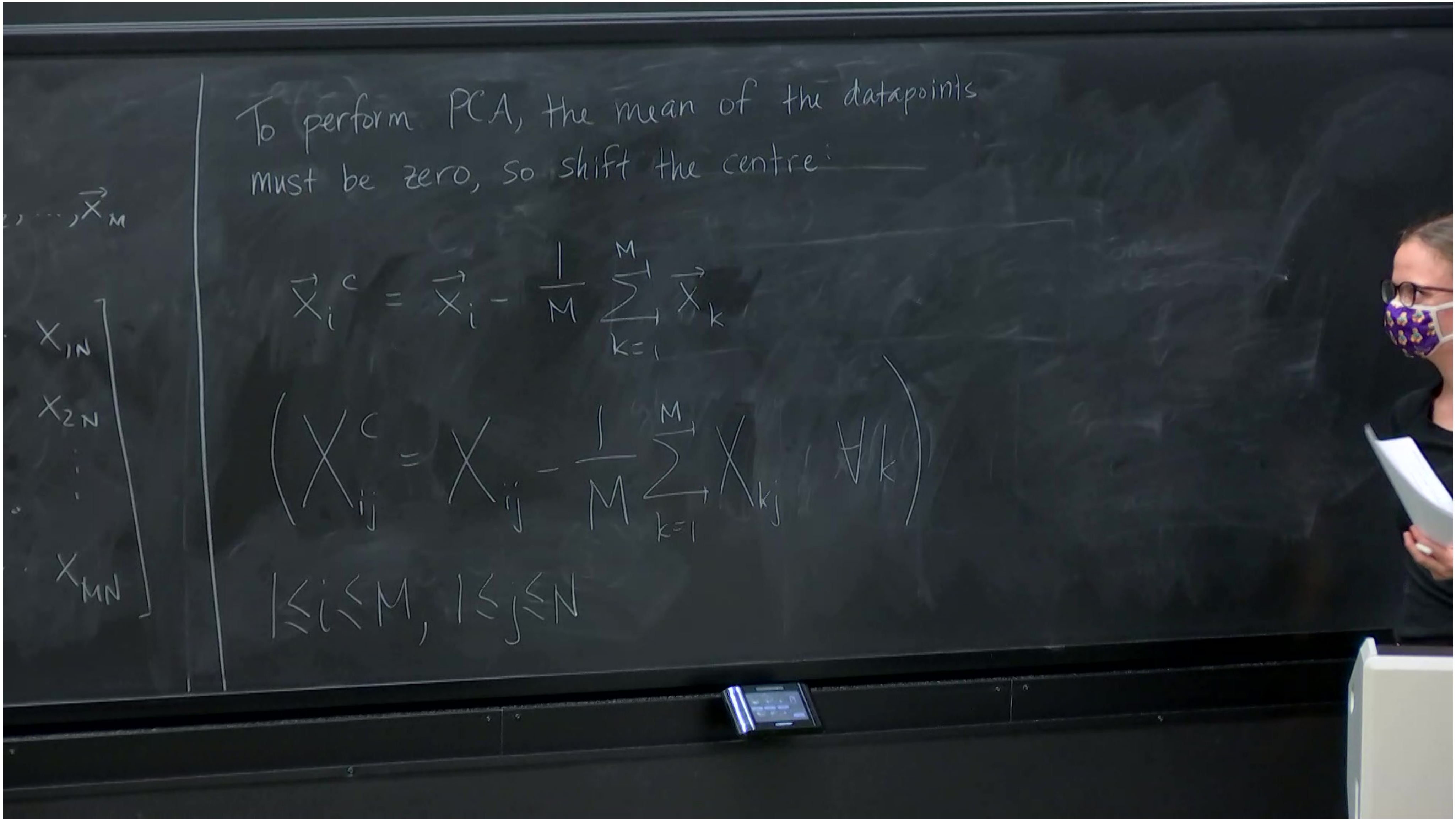
X_{MN}

To perform PCA, the mean of the datapoints must be zero, so shift:

$$\vec{x}_i^{\,c} = \vec{x}_i - \frac{1}{M}\sum_{k=1}^{M}\vec{x}_k$$

$$\left(X_{ij}^{c} = X_{ij} - \frac{1}{M}\sum_{k=1}^{M} X_{kj} \quad \forall k\right)$$

$1 \leq i \leq M, \ 1 \leq j \leq N$



$, \ldots, \vec{x}_M$

X_{1N}

X_{2N}

$\vdots$

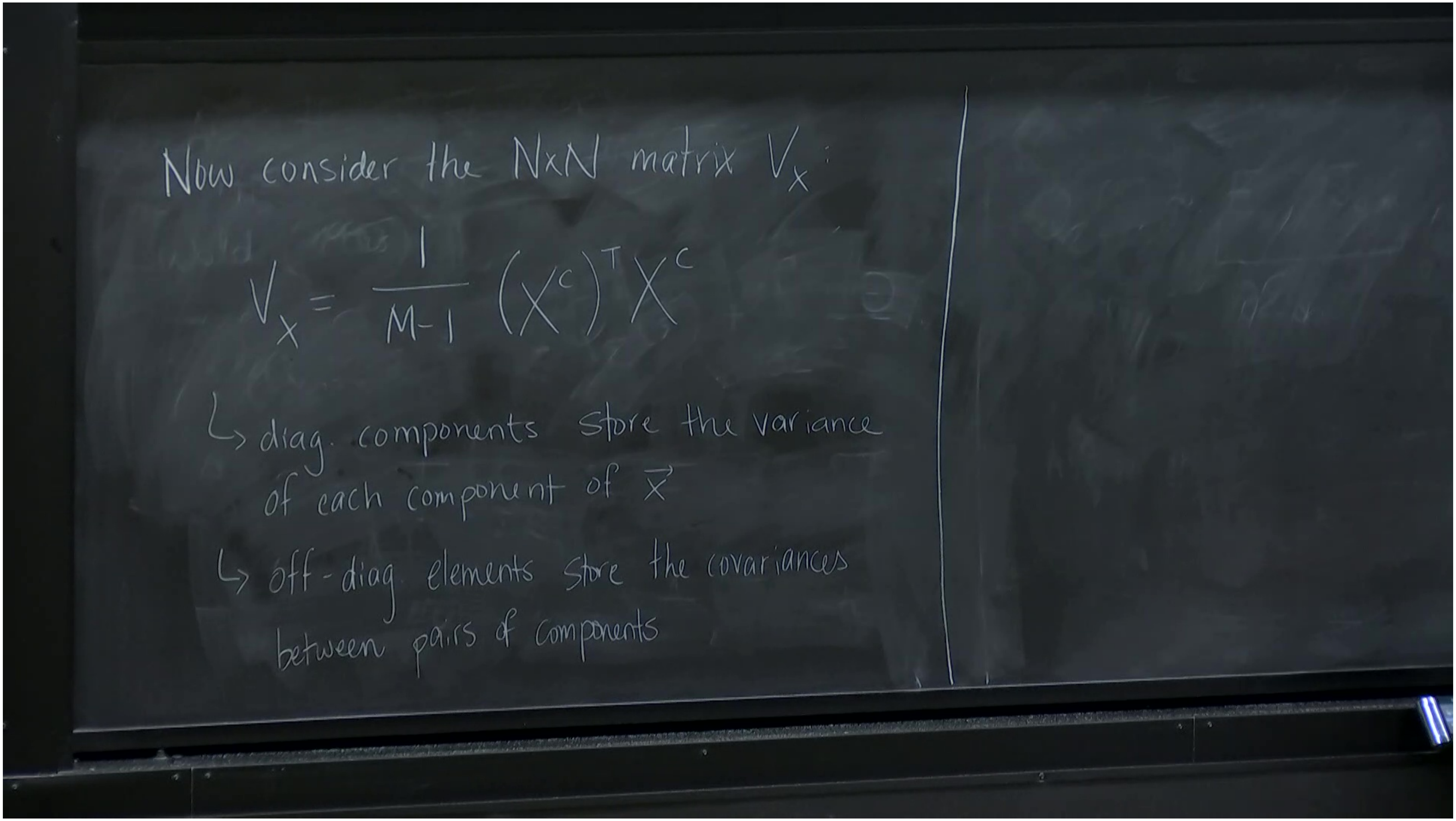
X_{MN}

To perform PCA, the mean of the datapoints must be zero, so shift the centre:

$$\vec{x}_i^{\,c} = \vec{x}_i - \frac{1}{M}\sum_{k=1}^{M}\vec{x}_k$$

$$\left(X^c_{ij} = X_{ij} - \frac{1}{M}\sum_{k=1}^{M} X_{kj} \quad \forall k \right)$$

$1 \le i \le M, \ 1 \le j \le N$

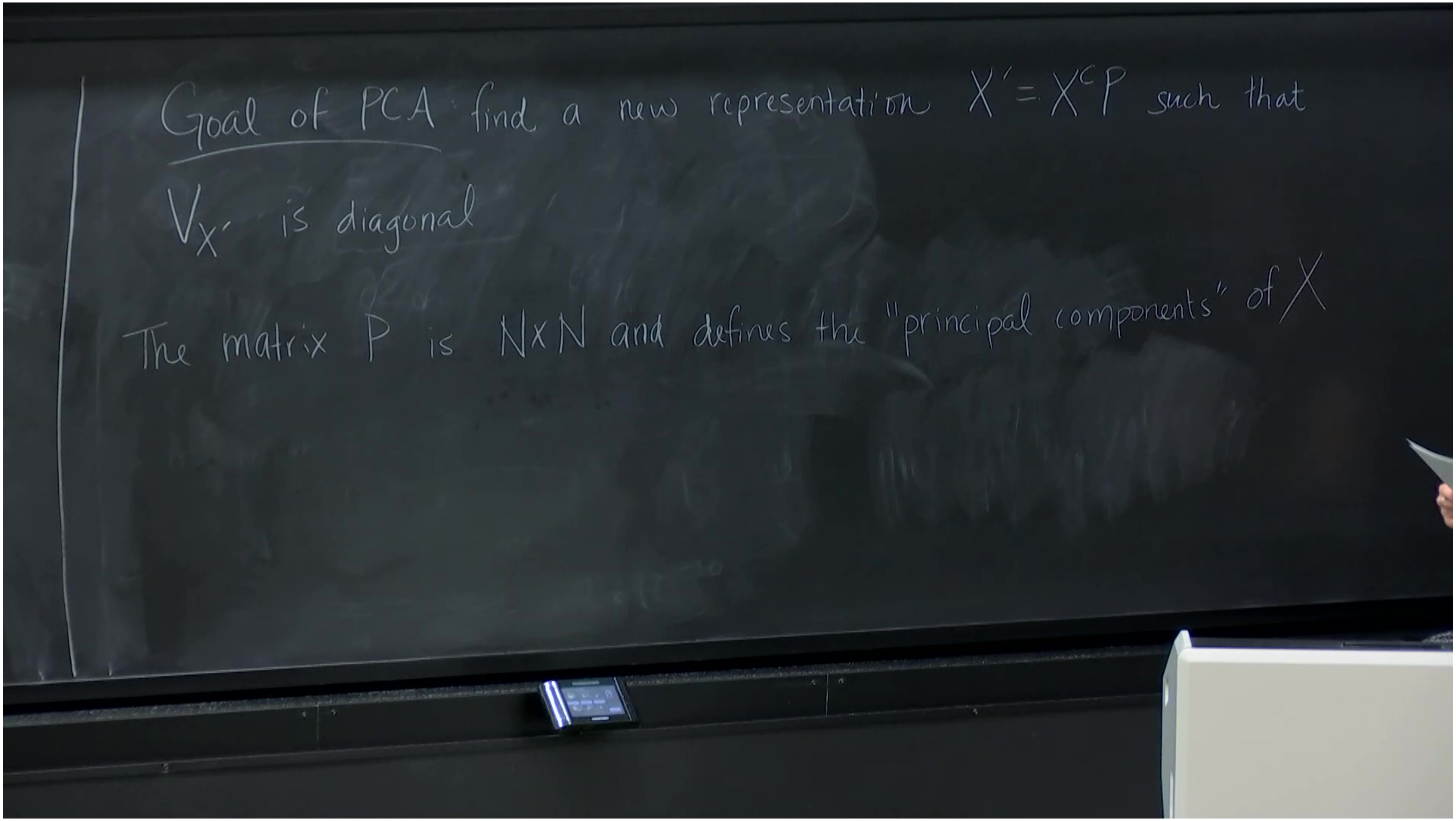


Now consider the N×N matrix V_x:

$$V_x = \frac{1}{M-1} (X^c)^T X^c$$

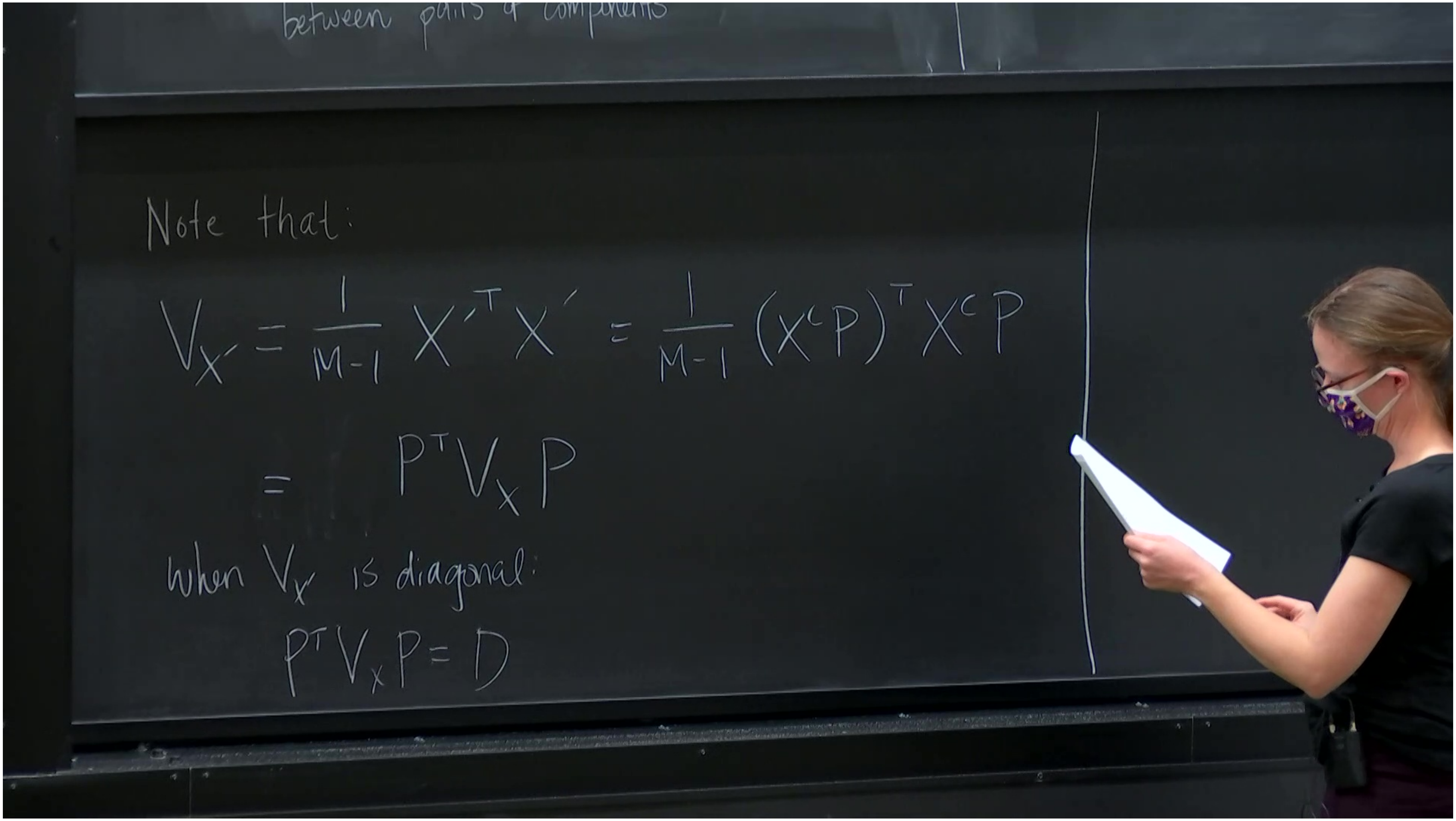
↳ diag. components store the variance of each component of $\vec{x}$

↳ off-diag. elements store the covariances between pairs of components



Goal of PCA: find a new representation $X' = X^c P$ such that $V_{X'}$ is diagonal

The matrix P is $N \times N$ and defines the "principal components" of X



between pairs of components

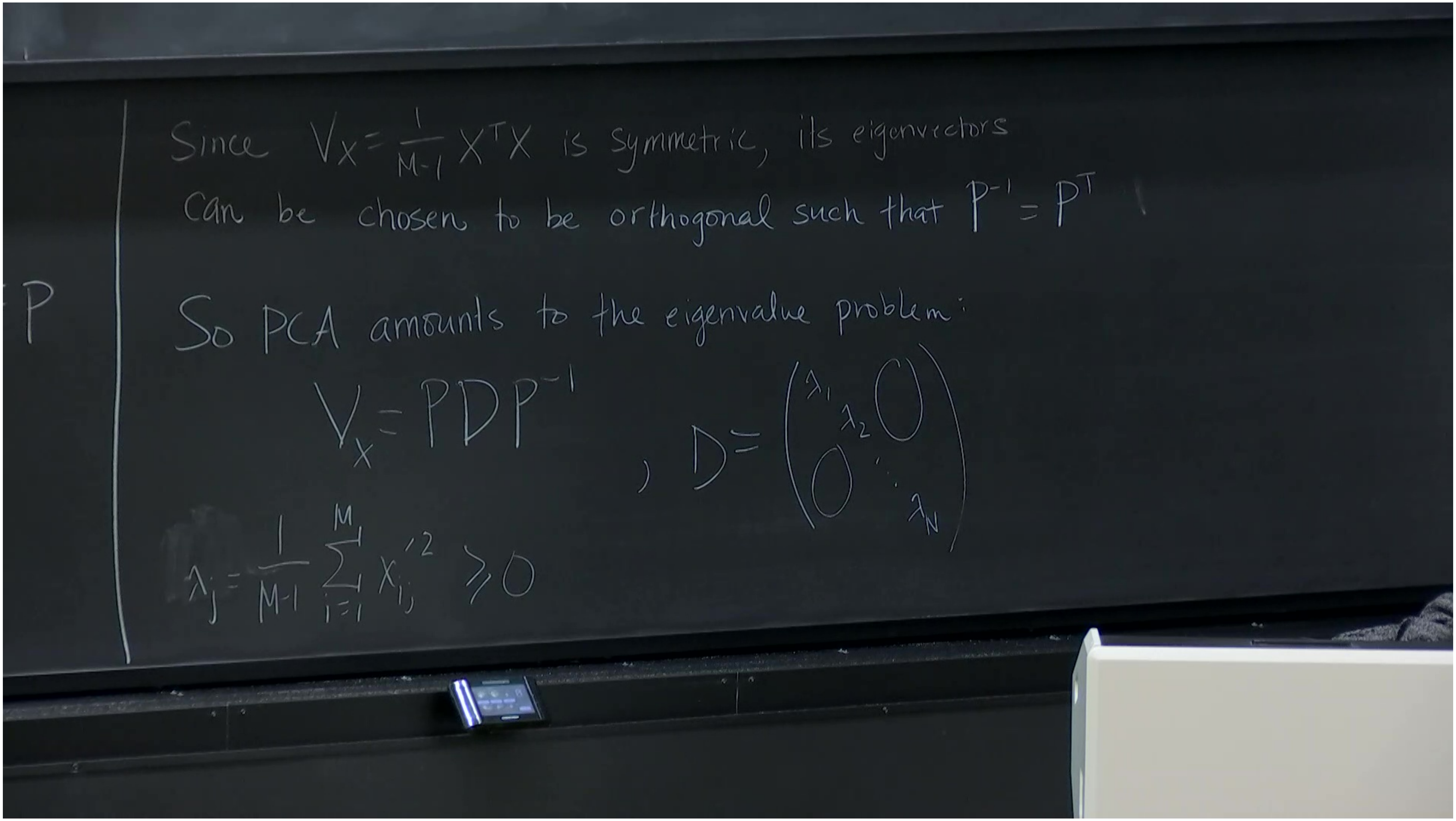
Note that:

$$V_{X'} = \frac{1}{M-1} X'^{T} X' = \frac{1}{M-1} (X^c P)^T X^c P$$

$$= P^T V_X P$$

when $V_{X'}$ is diagonal:

$$P^T V_X P = D$$



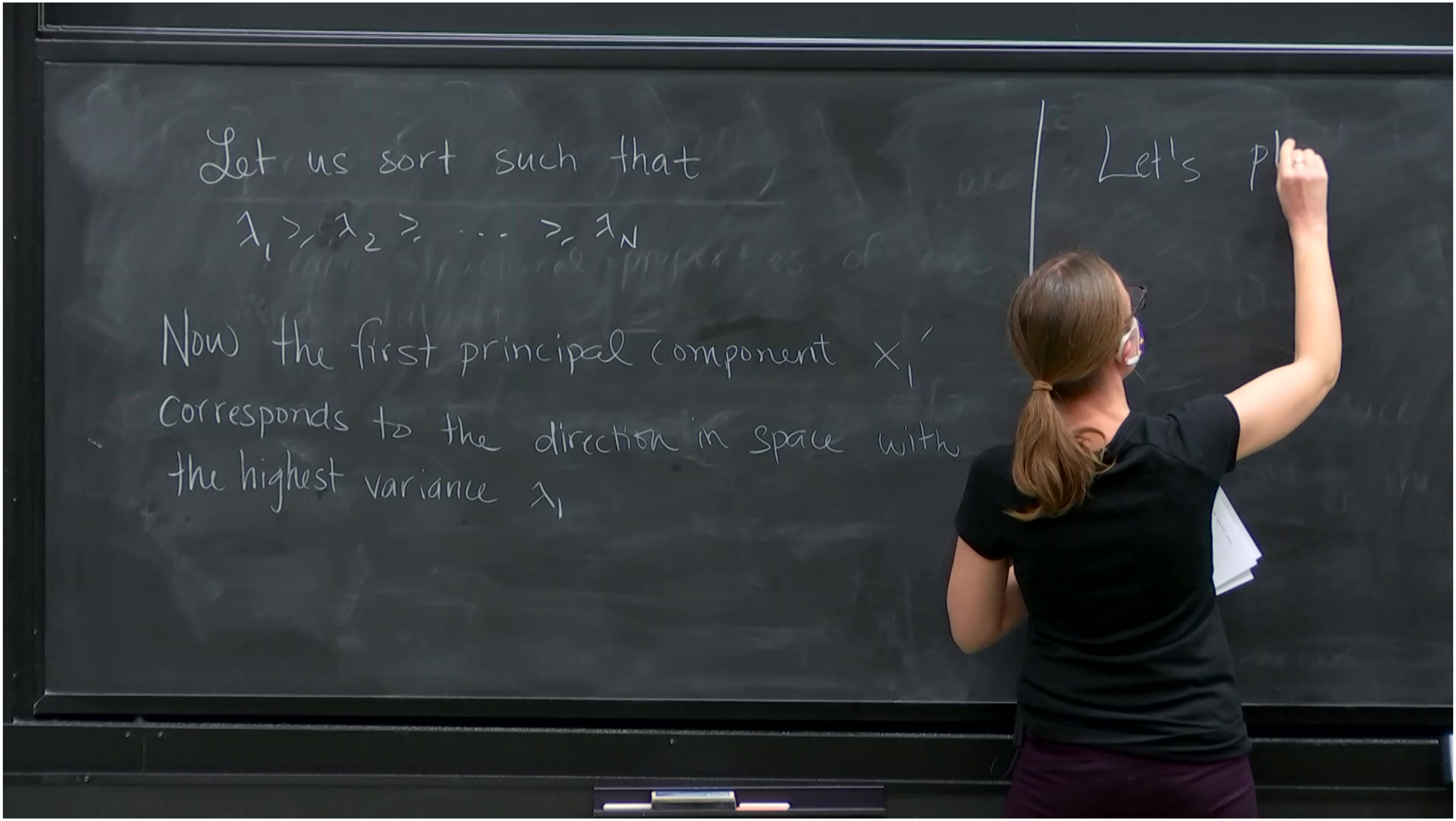
P

Since $V_X = \frac{1}{M-1} X^T X$ is symmetric, its eigenvectors can be chosen to be orthogonal such that $P^{-1} = P^T$

So PCA amounts to the eigenvalue problem:

$$V_X = PDP^{-1} \quad , \quad D = \begin{pmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_N \end{pmatrix}$$

$$\lambda_j = \frac{1}{M-1} \sum_{i=1}^{M} x'^2_{ij} \geq 0$$

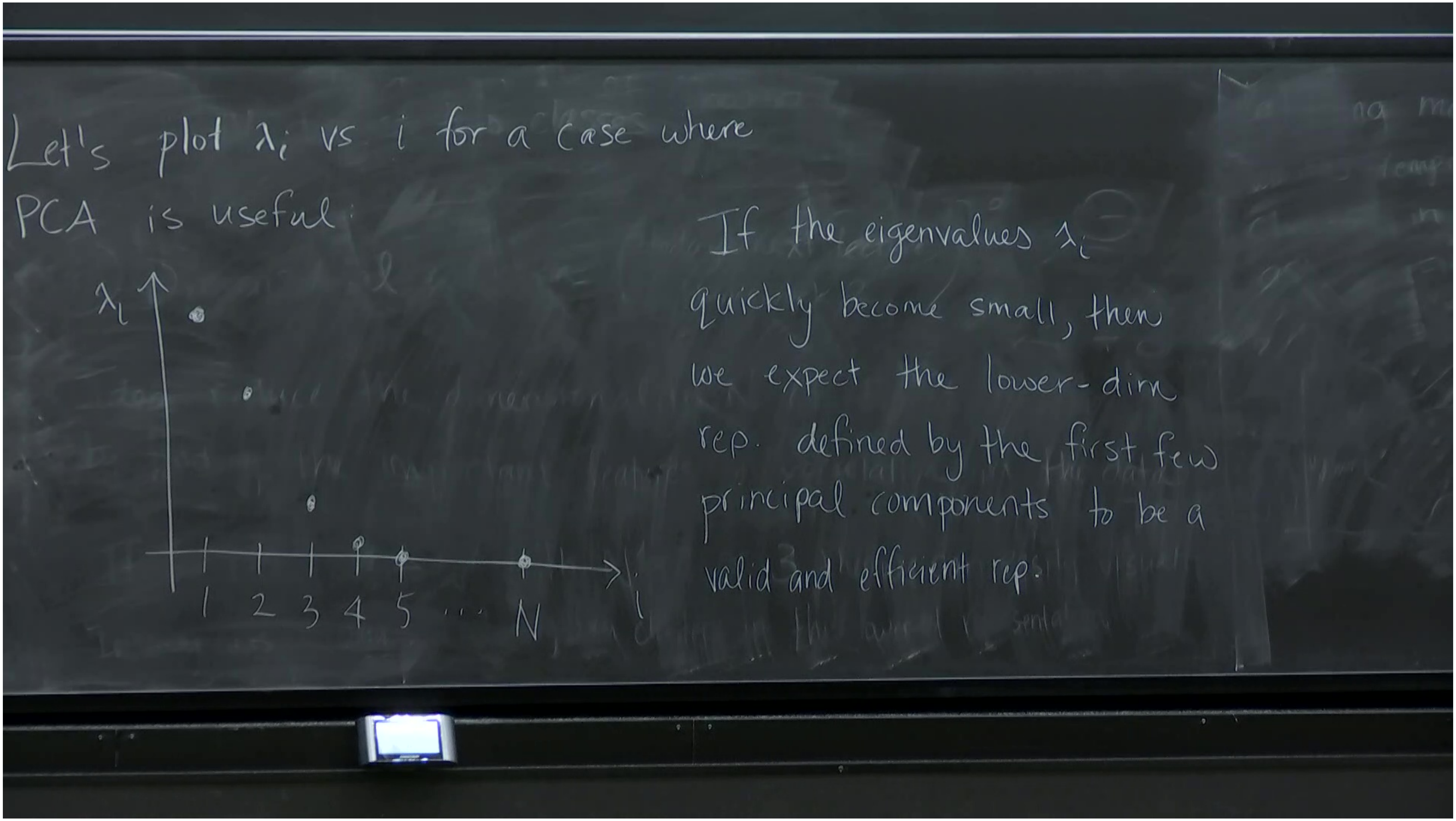


Let us sort such that

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N$$

Now the first principal component x_1' corresponds to the direction in space with the highest variance λ_1

Let's pl



Let's plot λ_i vs i for a case where PCA is useful:

If the eigenvalues λ_i quickly become small, then we expect the lower-dim rep. defined by the first few principal components to be a valid and efficient rep.