Title: Machine Learning (2021/2022)

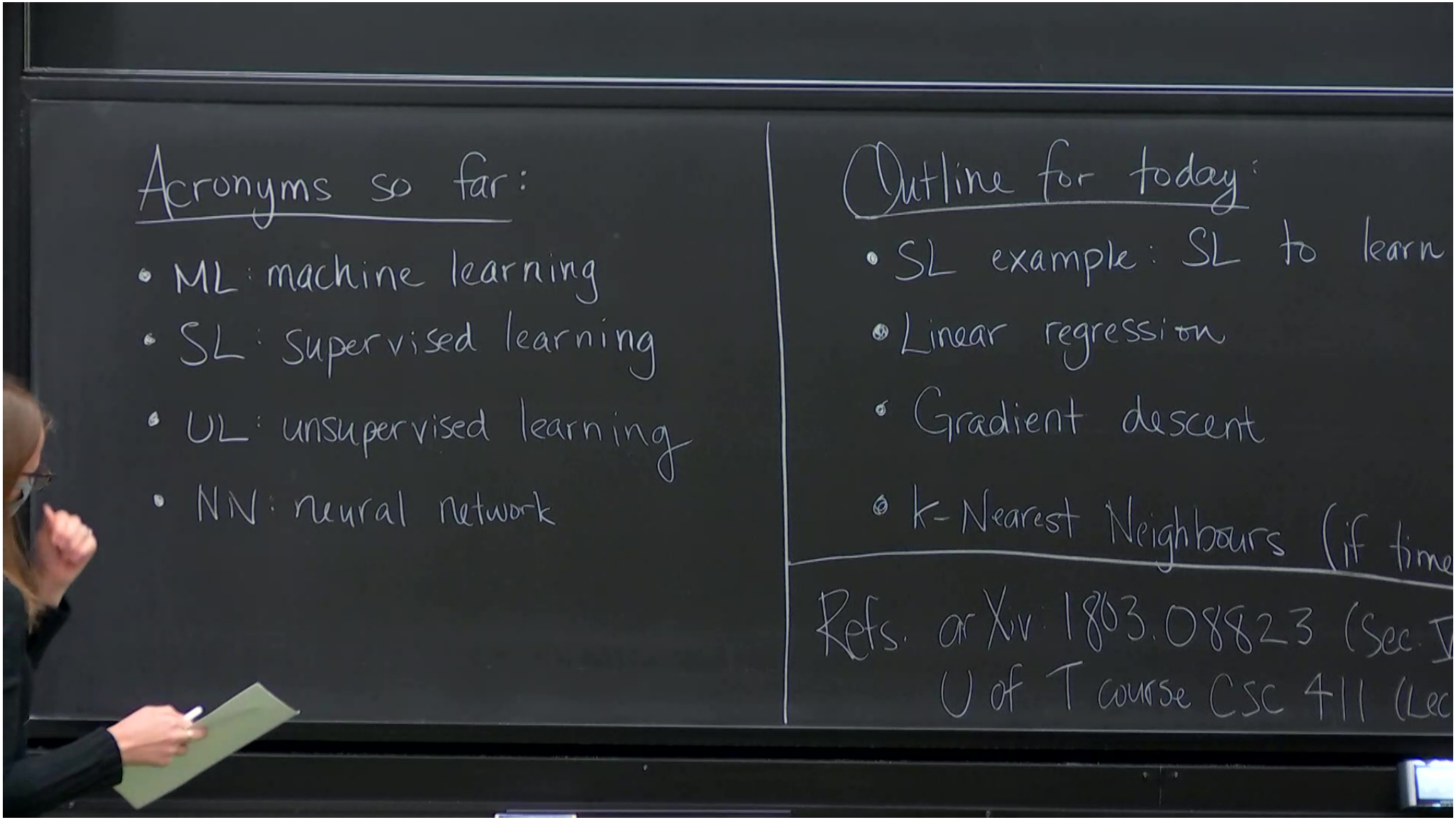
Speakers: Lauren Hayward

Collection: Machine Learning (2021/2022)

Date: April 07, 2022 - 9:00 AM

URL: https://pirsa.org/22040066

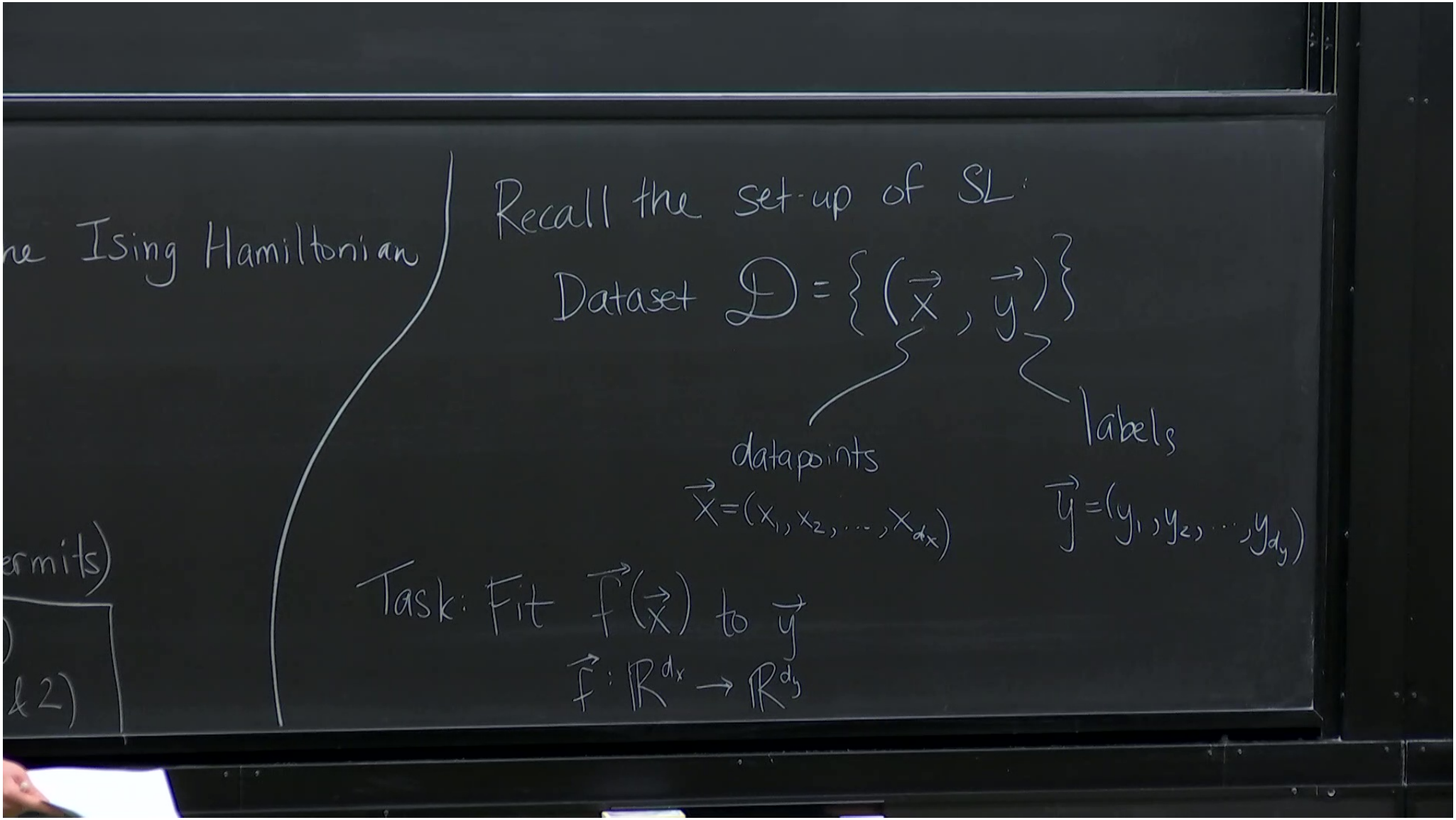
Abstract: This course is designed to introduce modern machine learning techniques for studying classical and quantum many-body problems encountered in condensed matter, quantum information, and related fields of physics. Lectures will focus on introducing machine learning algorithms and discussing how 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.

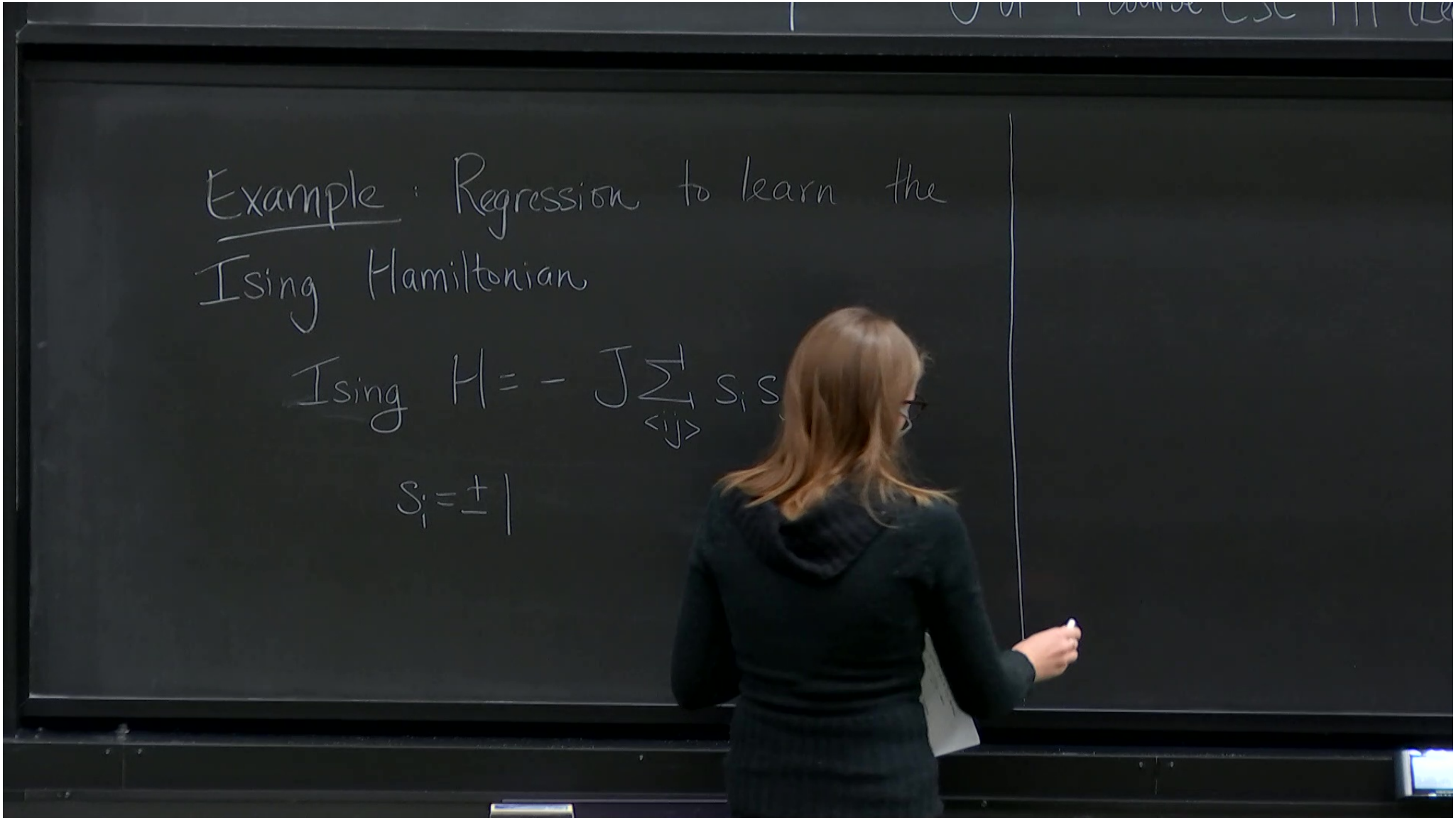

Acronyms so far:
• ML: machine learning
• SL: supervised learning
• UL: unsupervised learning
• NN: neural network
Outline for today:
• SL example: SL to learn
• Linear regression
• Gradient descent
• k-Nearest Neighbours (if time
Refs. arXiv: 1803.08823 (Sec. I
U of T course CSC 411 (Lec

Outline for today:

- SL example: SL to learn the Ising Hamiltonian
- Linear regression
- Gradient descent
- k-Nearest Neighbours (if time permits)

Refs. arXiv: 1803.08823 (Sec. VI)
U of T course CSC 411 (Lec. 6 & 2)


Recall the set-up of SL:
Dataset $\mathcal{D} = \{(\vec{x}, \vec{y})\}$
datapoints
$\vec{x} = (x_1, x_2, \dots, x_{d_x})$
labels
$\vec{y} = (y_1, y_2, \dots, y_{d_y})$
Task: Fit $\vec{f}(\vec{x})$ to $\vec{y}$
$\vec{f}: \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$


Example: Regression to learn the
Ising Hamiltonian
Ising H = - J Σ s_i s
<ij>
s_i = ±1

Dataset $\mathcal{D} = \{(\vec{x}, y)\}$

spin config

energy

$\vec{S} = (S_1, S_2, \ldots, S_N)$

(N = # of spins)

Assume we are given $\mathcal{D}$ but we don't know the Hamiltonian used to generate it.

$\vec{x}, y)\}$

energy

$s_N)$

ns)

n $\mathcal{D}$ but we don't

used to generate it.

We could assume a more general model, such as

$$H_{model} = -\sum_{i=1}^{N} \sum_{j>i} J_{ij} S_i S_j$$

and then use SL to determine J_{ij}

(See Tutorial #1)

know the Hamiltonian

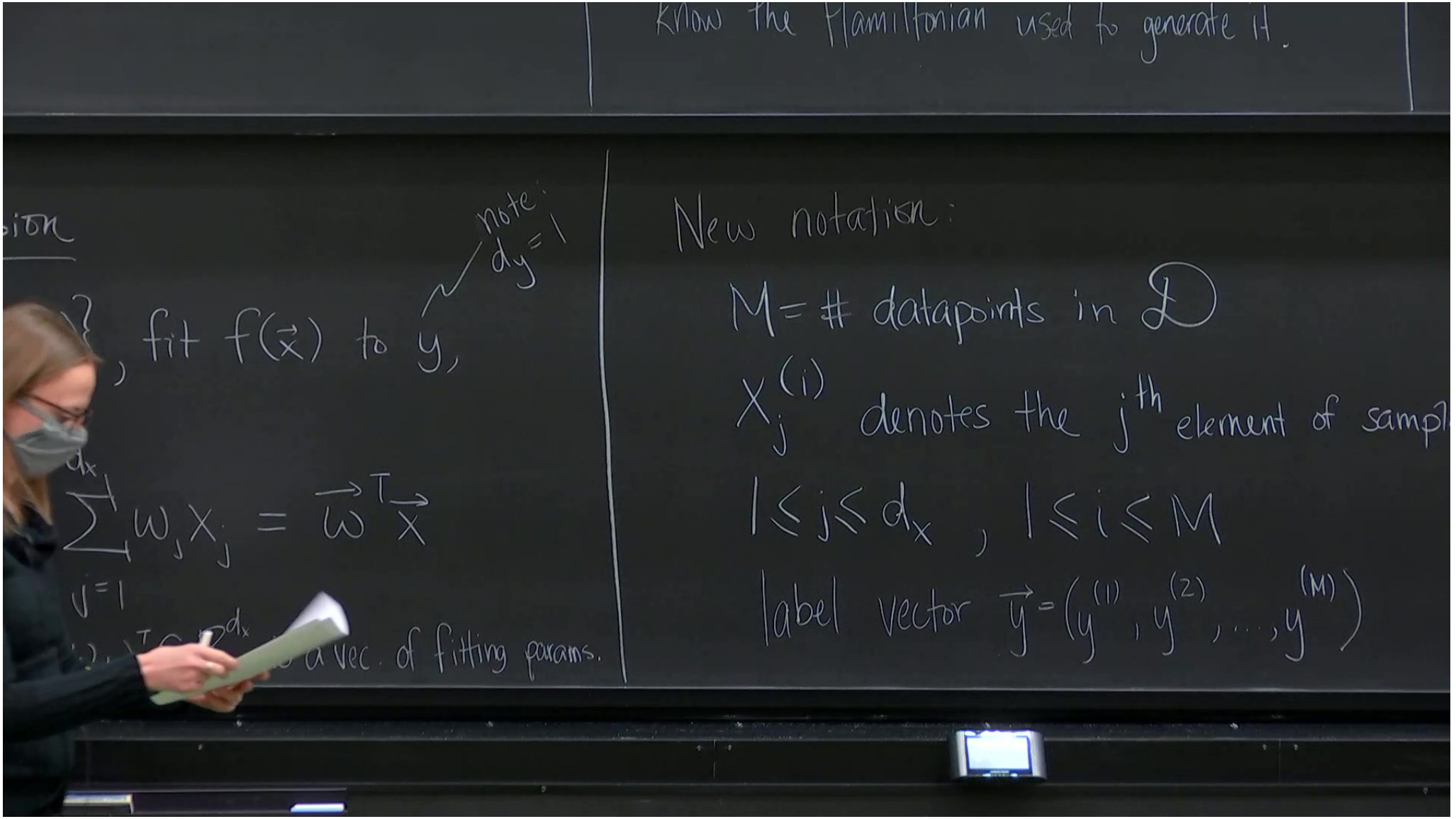
Linear Regression

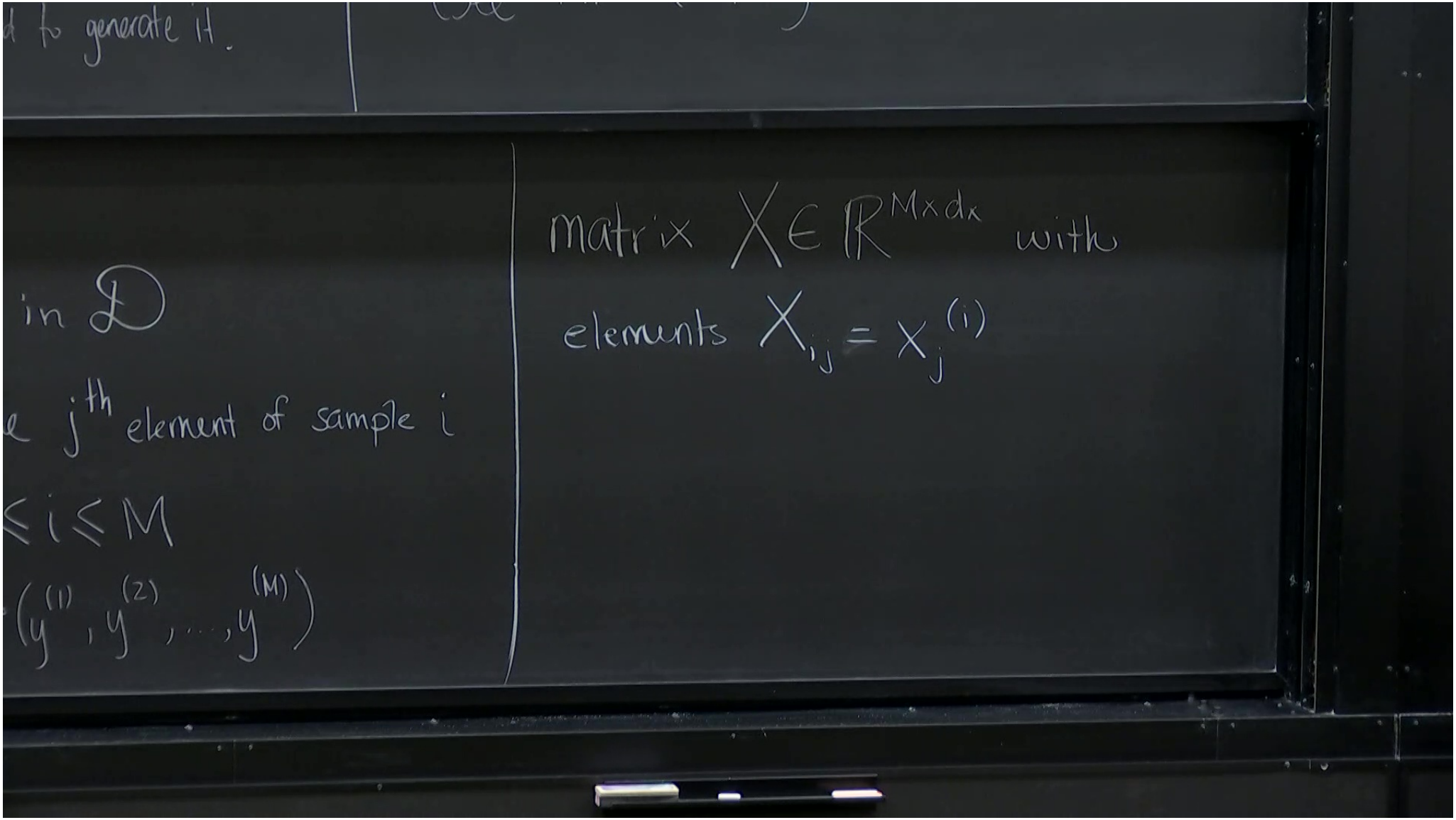
For $\mathcal{D} = \{(\vec{x}, y)\}$, fit $f(\vec{x})$ to y, where

$$f(\vec{x}) = \sum_{j=1}^{d_x} w_j x_j = \vec{w}^T \vec{x}$$

where $\vec{w} = (w_1, w_2, \ldots, w_{d_x})^T \in \mathbb{R}^{d_x}$ is a vec. of fitting params.

know the Hamiltonian used to generate it.




to generate it.
in $\mathcal{D}$
j^{th} element of sample i
$\leq i \leq M$
$(y^{(1)}, y^{(2)}, \ldots, y^{(M)})$
matrix $X \in \mathbb{R}^{M \times d_x}$ with
elements $X_{i,j} = x_j^{(i)}$

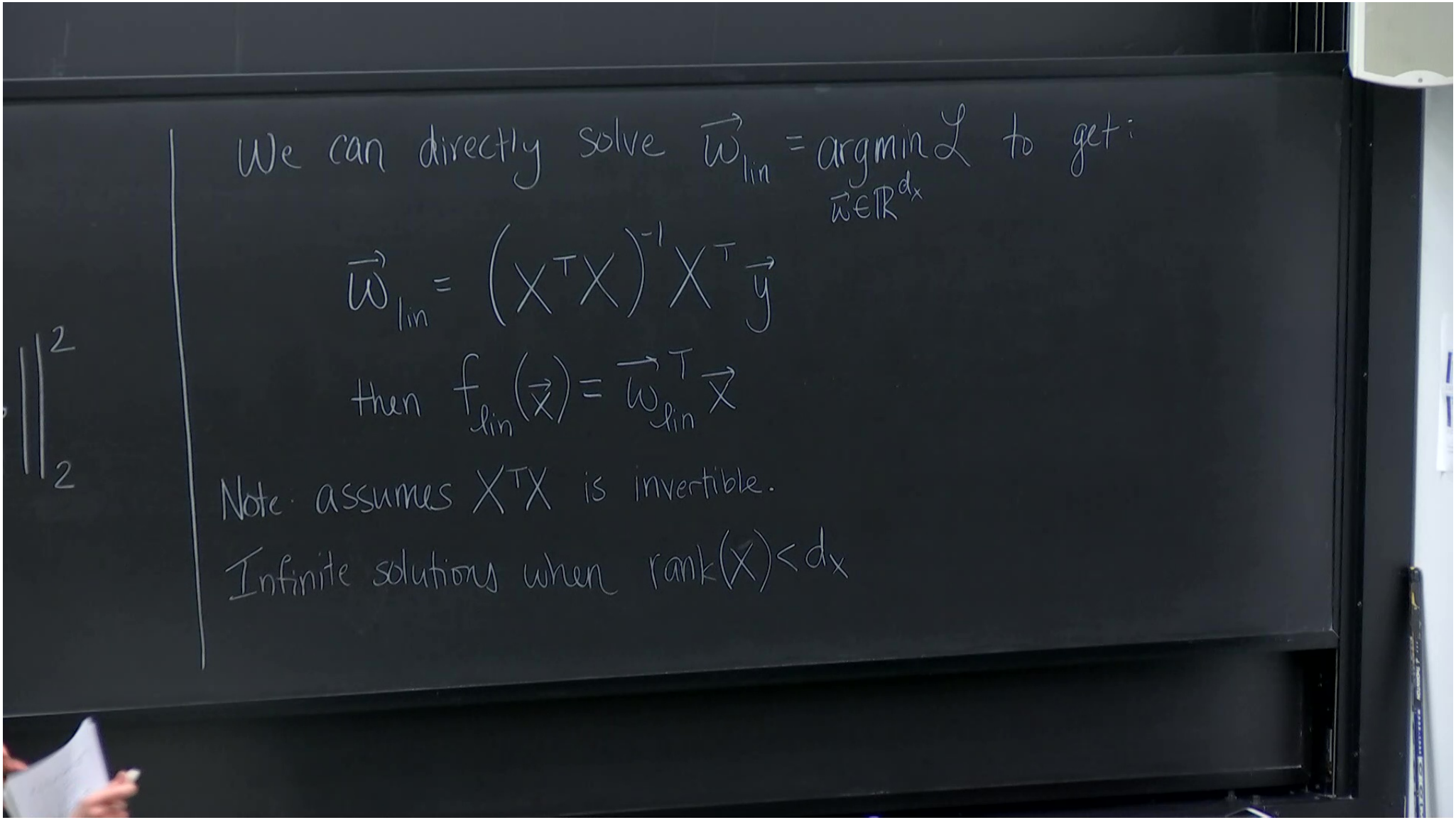
Determine the params w_j by minimizing $\mathcal{L}$, where

$$\mathcal{L} = \sum_{i=1}^{M} \left(\sum_{j=1}^{d_x+1} w_j x_j^{(i)} - y^{(i)} \right)^2 = \left\| X\vec{w} - \vec{y} \right\|_2^2$$

Determine the params w_j by minimizing $\mathcal{L}$, where

$$\mathcal{L} \equiv \sum_{i=1}^{M} \left(\sum_{j=1}^{d_x} w_j x_j^{(i)} - y^{(i)} \right)^2 = \left\| X\vec{w} - \vec{y} \right\|_2^2$$

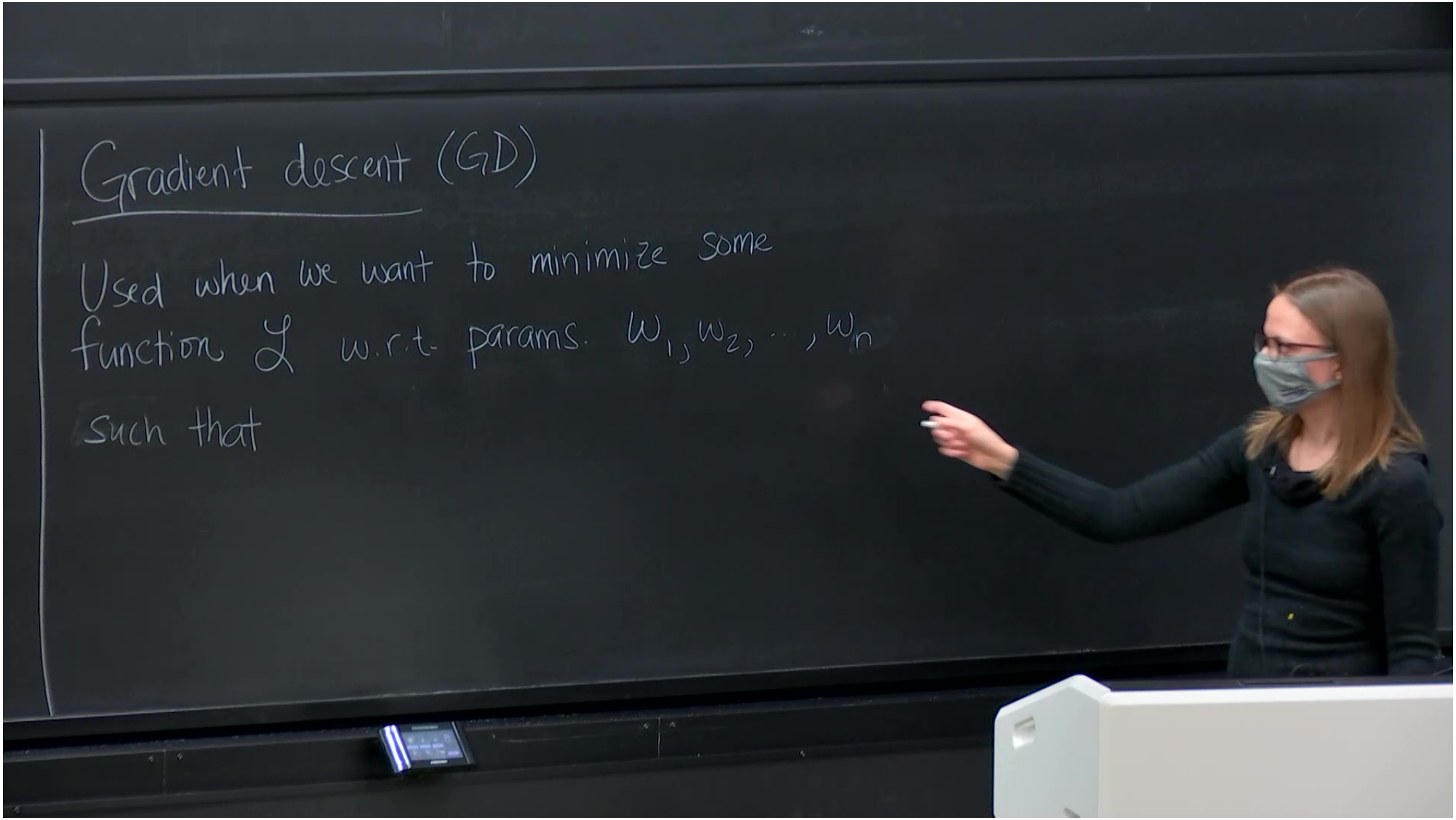
where $\|\vec{x}\|_2 = \left(\sum_{i=1}^{1} x_i^2 \right)^{1/2}$ is the "L_2 norm"

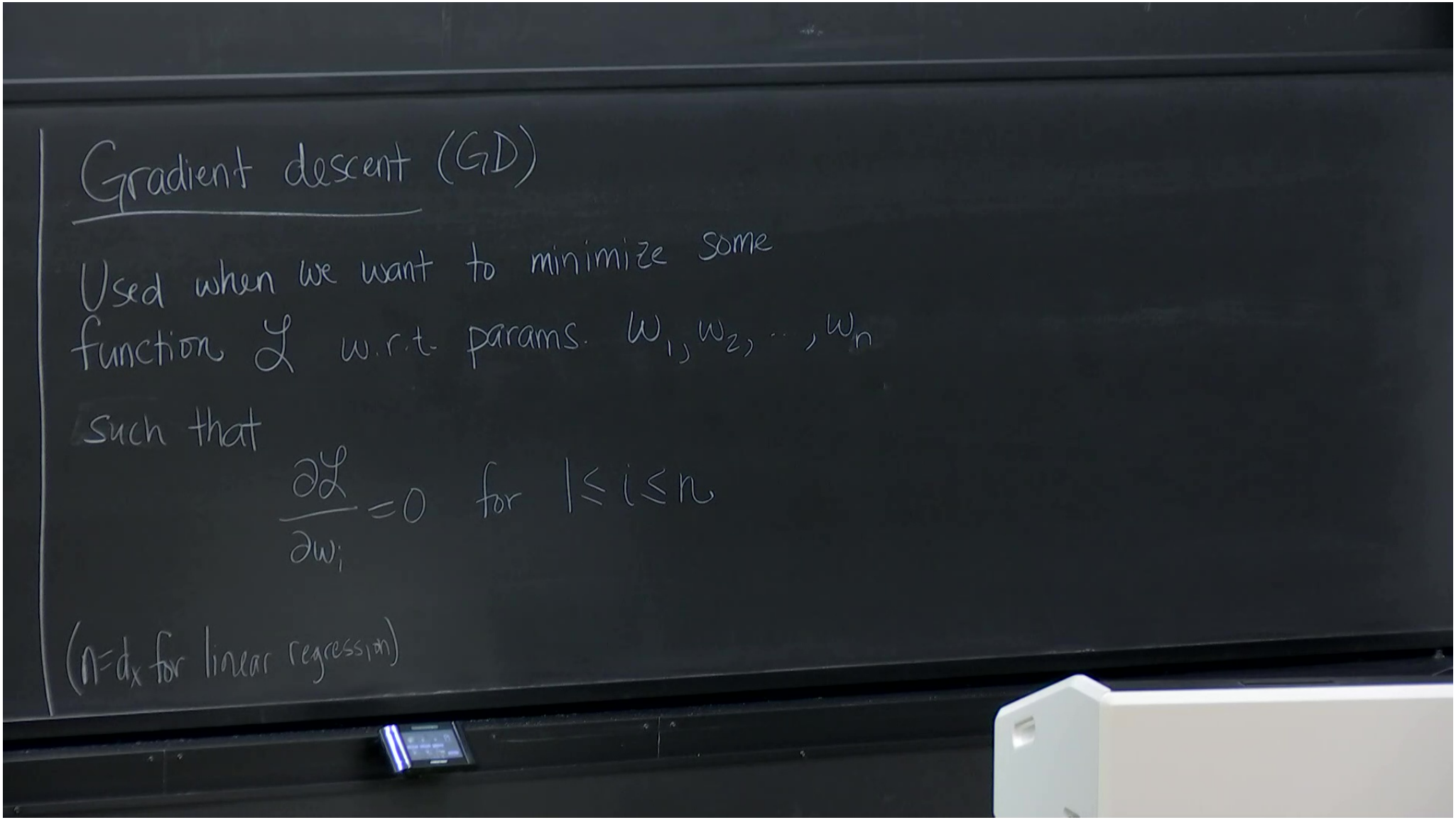

We can directly solve $\vec{w}_{lin} = \underset{\vec{w} \in \mathbb{R}^{d_x}}{\operatorname{argmin}} \mathcal{L}$ to get:
$\vec{w}_{lin} = (X^T X)^{-1} X^T \vec{y}$
then $f_{lin}(\vec{x}) = \vec{w}_{lin}^T \vec{x}$
Note: assumes $X^T X$ is invertible.
Infinite solutions when $\text{rank}(X) < d_x$
$\|\|_2^2$
$\|\|_2$

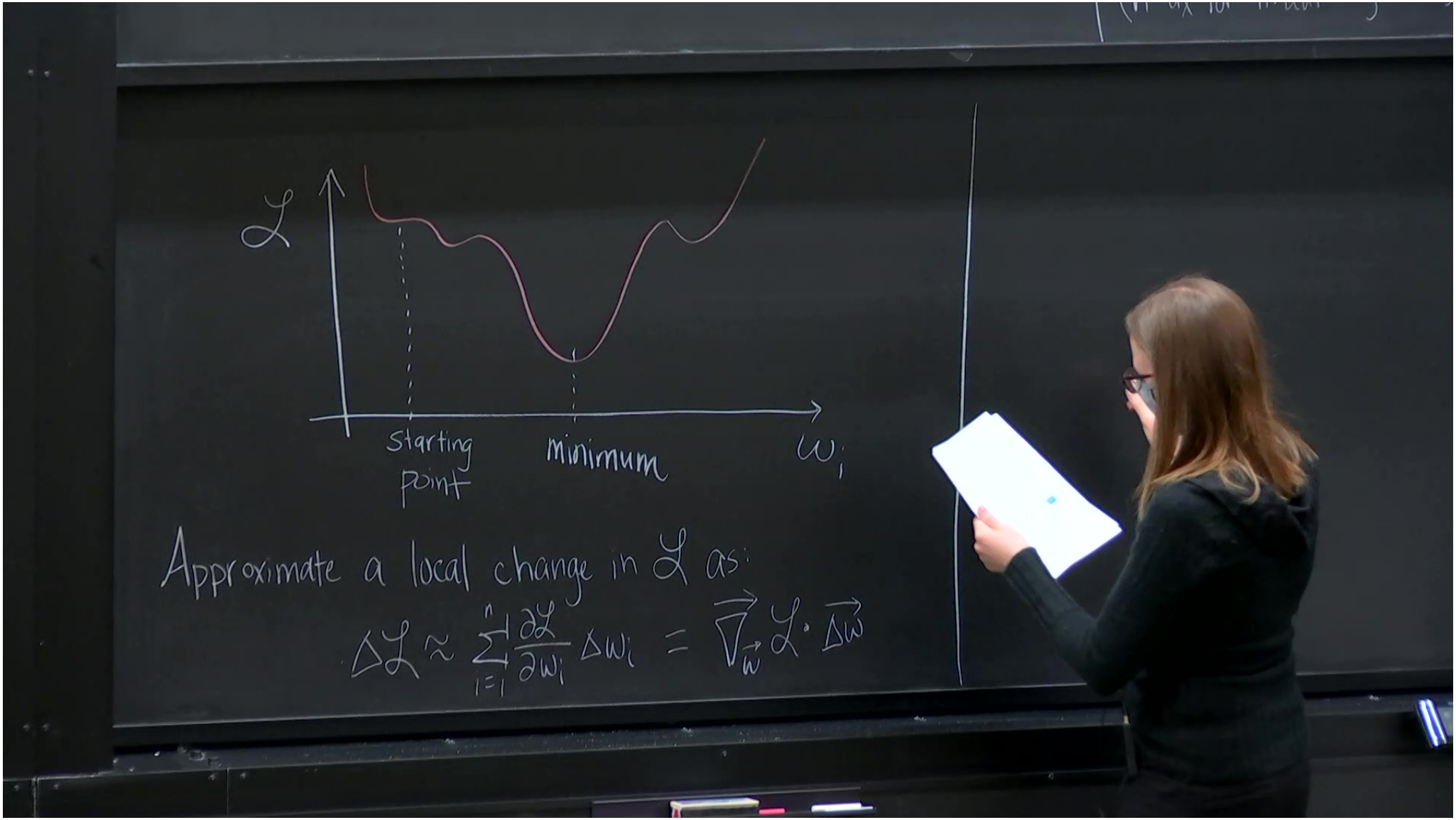
Alternatively, we can use an algorithm such as gradient descent to find w;

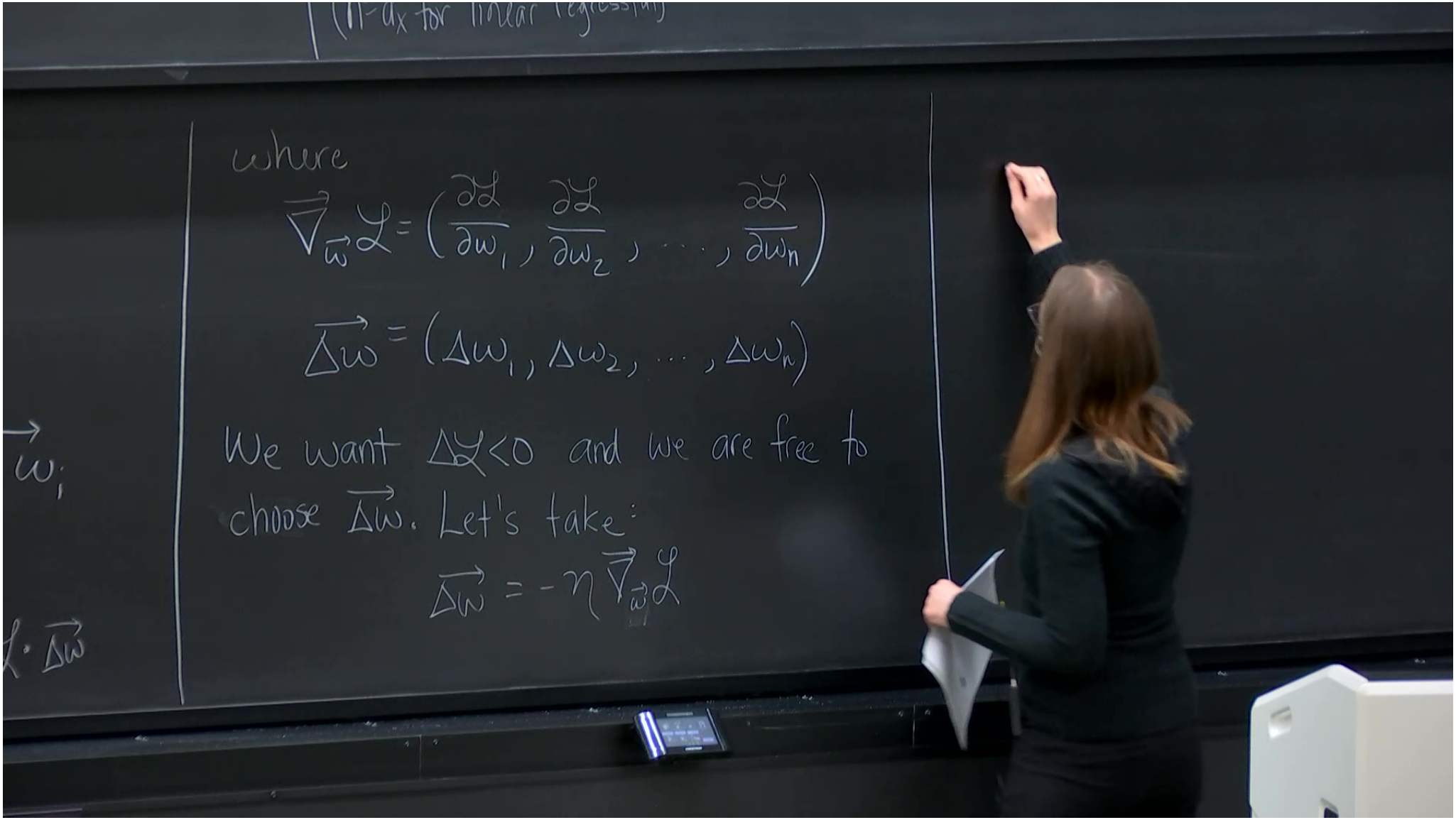
↳ can be more efficient

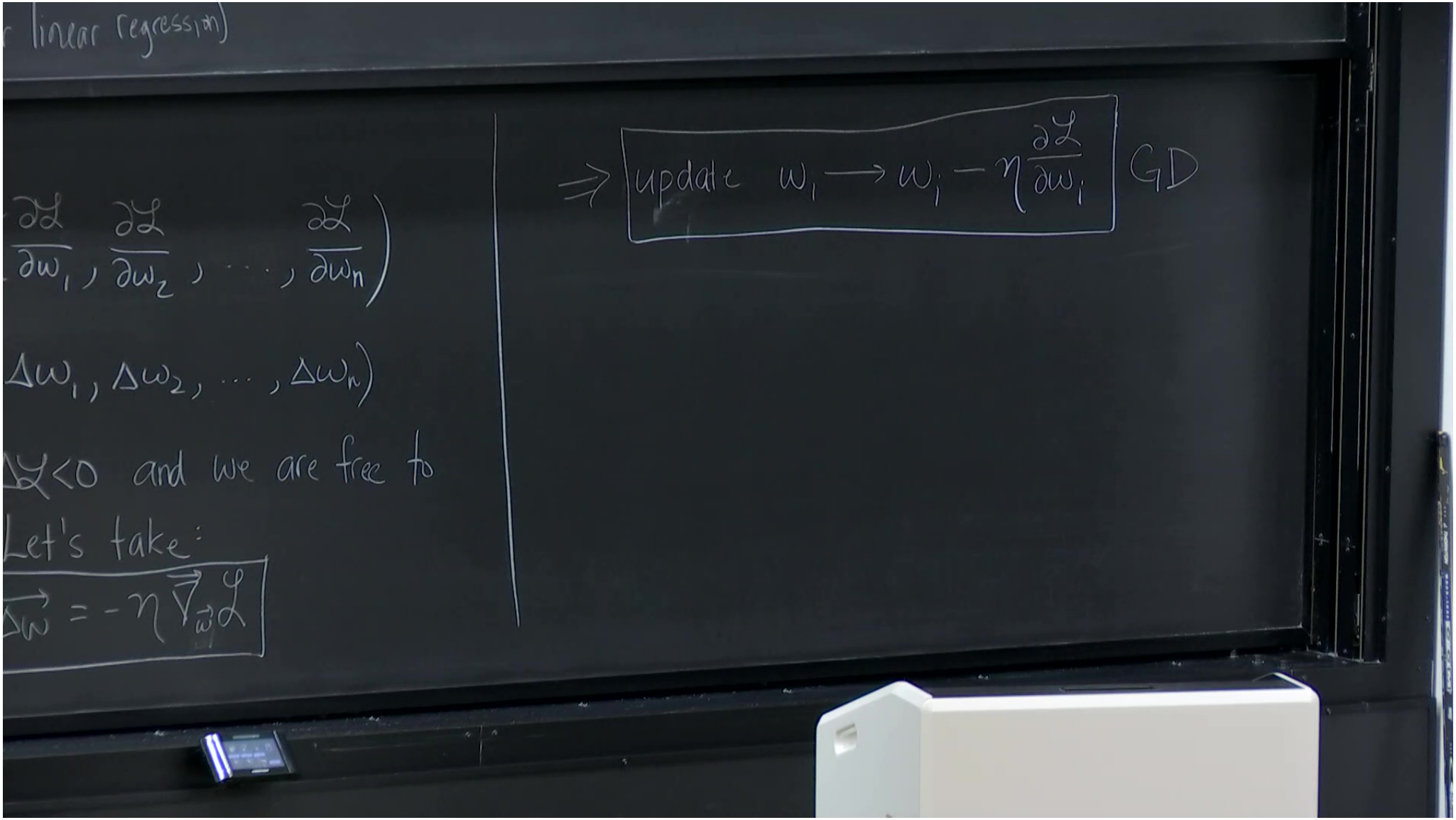
↳ broadly used in many ML algorithms


Gradient descent (GD)
Used when we want to minimize some function $\mathcal{L}$ w.r.t. params. $w_1, w_2, \ldots, w_n$
such that


Gradient descent (GD)
Used when we want to minimize some function $\mathcal{L}$ w.r.t. params. $w_1, w_2, \ldots, w_n$
such that
$\frac{\partial \mathcal{L}}{\partial w_i} = 0$ for $1 \le i \le n$
($n = d_x$ for linear regression)


$\mathcal{L}$
starting point
minimum
w_i
Approximate a local change in $\mathcal{L}$ as:
$\Delta\mathcal{L} \approx \sum_{i=1}^{n} \frac{\partial \mathcal{L}}{\partial w_i} \Delta w_i = \vec{\nabla}_{\vec{w}} \mathcal{L} \cdot \vec{\Delta w}$

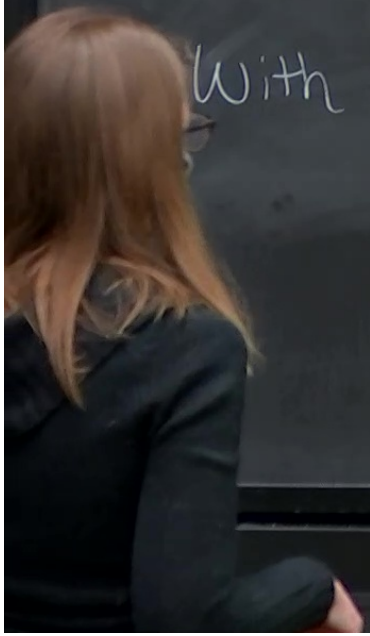

(1-dx for linear regression)
where
$\vec{\nabla}_{\vec{w}}\mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \cdots, \frac{\partial \mathcal{L}}{\partial w_n}\right)$
$\vec{\Delta w} = (\Delta w_1, \Delta w_2, \cdots, \Delta w_n)$
We want $\Delta \mathcal{L} < 0$ and we are free to choose $\vec{\Delta w}$. Let's take:
$\vec{\Delta w} = -\eta \vec{\nabla}_{\vec{w}}\mathcal{L}$


linear regression
$\left(\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \cdots, \frac{\partial \mathcal{L}}{\partial w_n}\right)$
$(\Delta w_1, \Delta w_2, \cdots, \Delta w_n)$
$\Delta \mathcal{L} < 0$ and we are free to
Let's take:
$\vec{\Delta w} = -\eta \vec{\nabla}_{\vec{w}} \mathcal{L}$
$\Rightarrow$ update $w_i \longrightarrow w_i - \eta \frac{\partial \mathcal{L}}{\partial w_i}$ GD

$\eta > 0$ is a param. called the learning rate that we can choose (a "hyperparameter")

With this choice of $\vec{\Delta w}$:

$$\Delta \mathcal{L} \approx -\eta \| \vec{\nabla}_{\vec{w}} \mathcal{L} \|_2^2 < 0$$



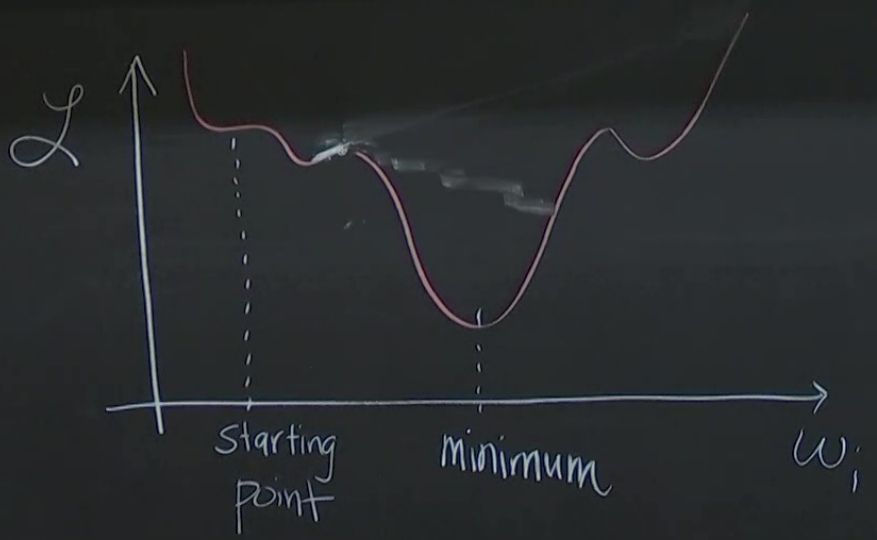
Notes about η:

- needs to be small enough that the first-order approx. is valid (otherwise $\mathcal{L}$ could increase, miss the minimum)
- for sufficiently small (finite) η, we will eventually converge to a local minimum

Notes about η:

- needs to be small enough that the first-order approx. is valid (otherwise $\mathcal{L}$ could increase, miss the minimum)
- for sufficiently small (finite) η, we will eventually converge to a local minimum

We can get stuck in a local min.



Approximate a local change in $\mathcal{L}$ as:

$$\Delta\mathcal{L} \approx \sum_{i=1}^{n} \frac{\partial \mathcal{L}}{\partial w_i} \Delta w_i = \vec{\nabla}_{\vec{w}} \mathcal{L} \cdot \vec{\Delta w}$$

where

$$\vec{\nabla}_{\vec{w}} \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \dots \right)$$

$$\vec{\Delta w} = (\Delta w_1, \Delta w_2, \dots$$

We want $\Delta\mathcal{L} < 0$ and we choose $\vec{\Delta w}$. Let's take:

$$\boxed{\vec{\Delta w} = -\eta \vec{\nabla}_{\vec{w}} \mathcal{L}}$$