Title: Lecture - Machine Learning, PHYS 777

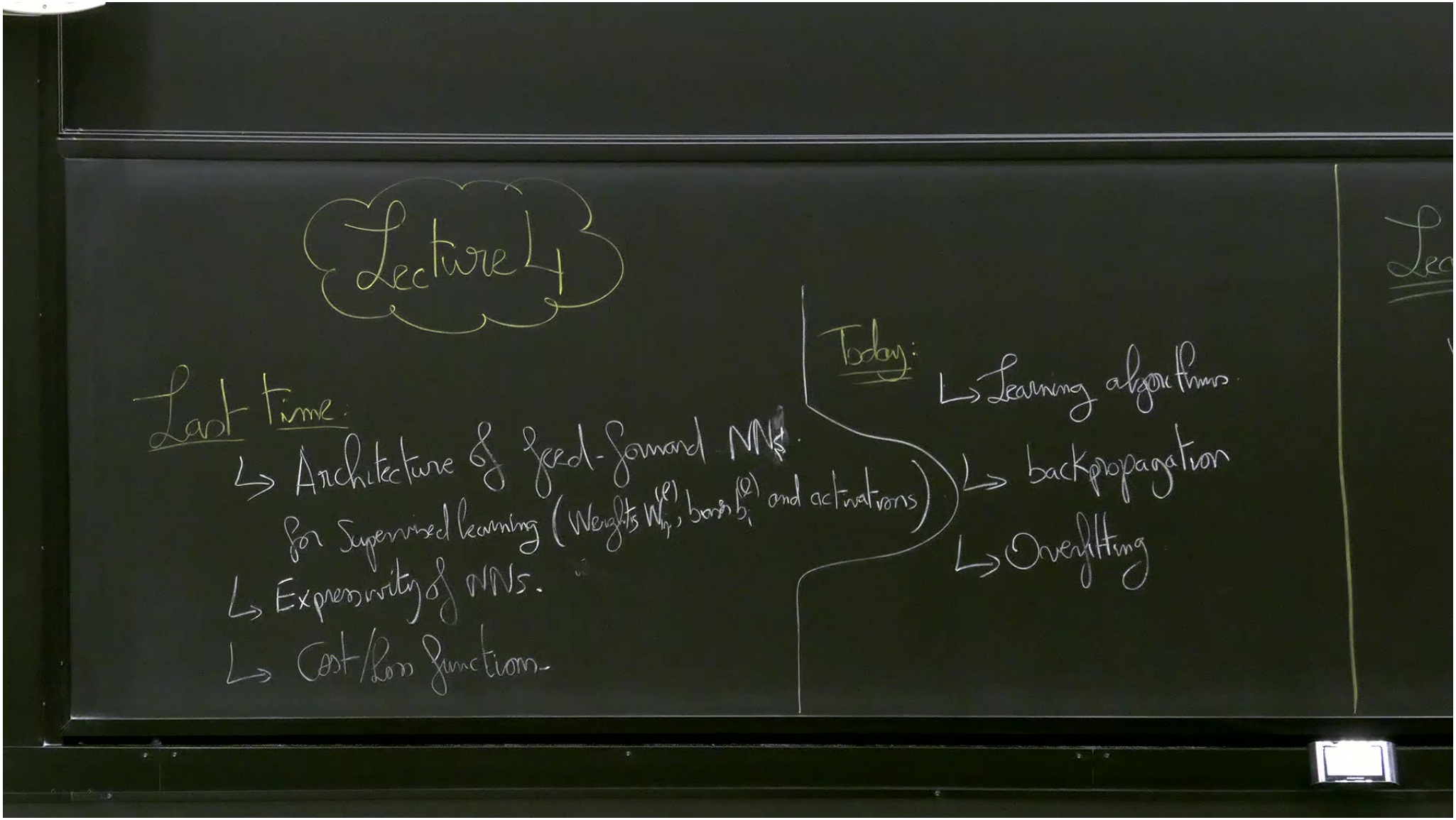
Speakers: Mohamed Hibat Allah

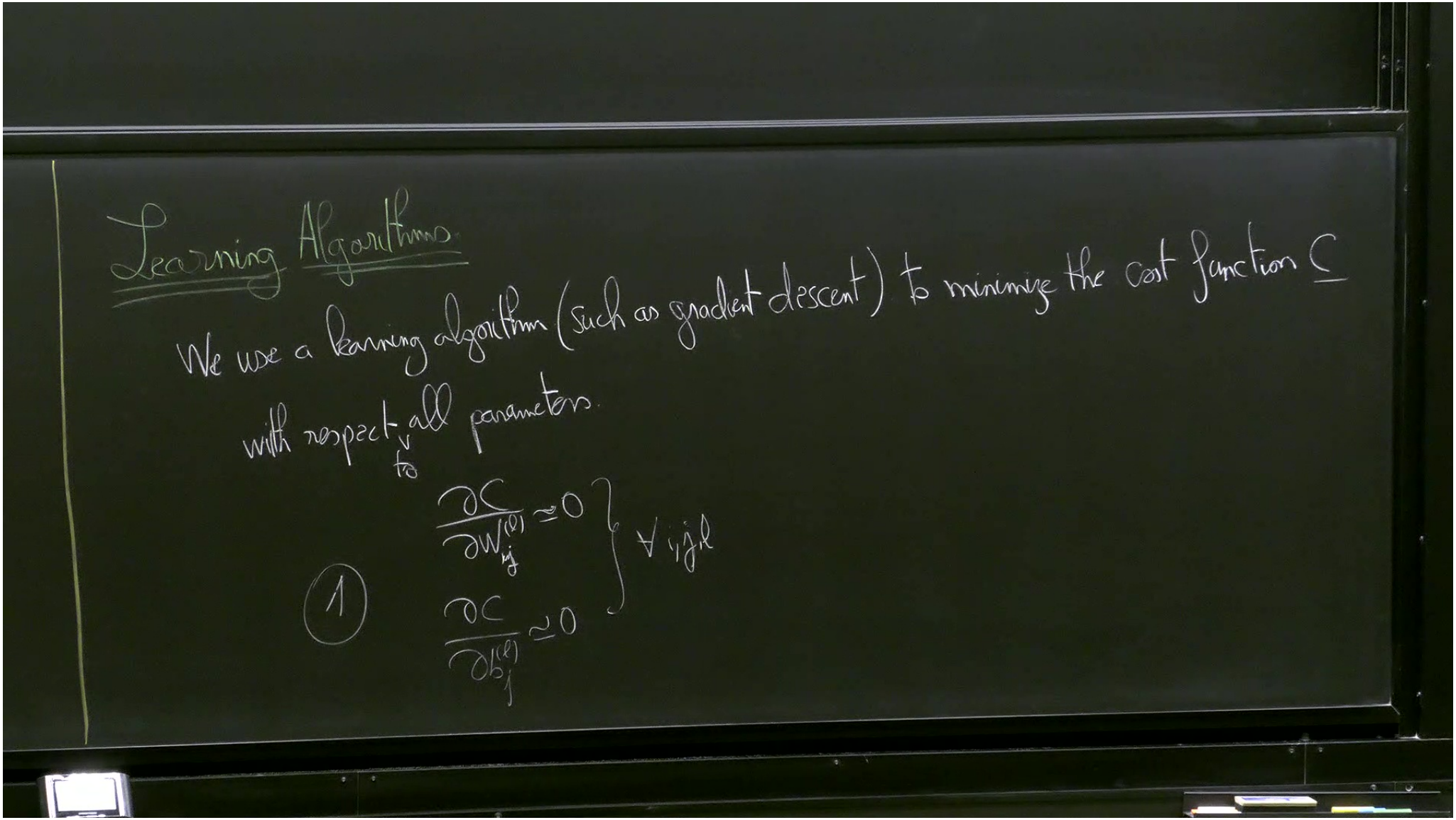
Collection/Series: Machine Learning (Elective), PHYS 777, February 24 - March 28, 2025

Subject: Condensed Matter, Other

Date: March 04, 2025 - 9:00 AM

URL: https://pirsa.org/25030035


Lecture 4
Last time:
↳ Architecture of feed-forward NNs for supervised learning (weights $W_{ij}^{(\ell)}$, biases $b_i^{(\ell)}$ and activations)
↳ Expressivity of NNs.
↳ Cost/Loss functions.
Today:
↳ Learning algorithms
↳ backpropagation
↳ Overfitting

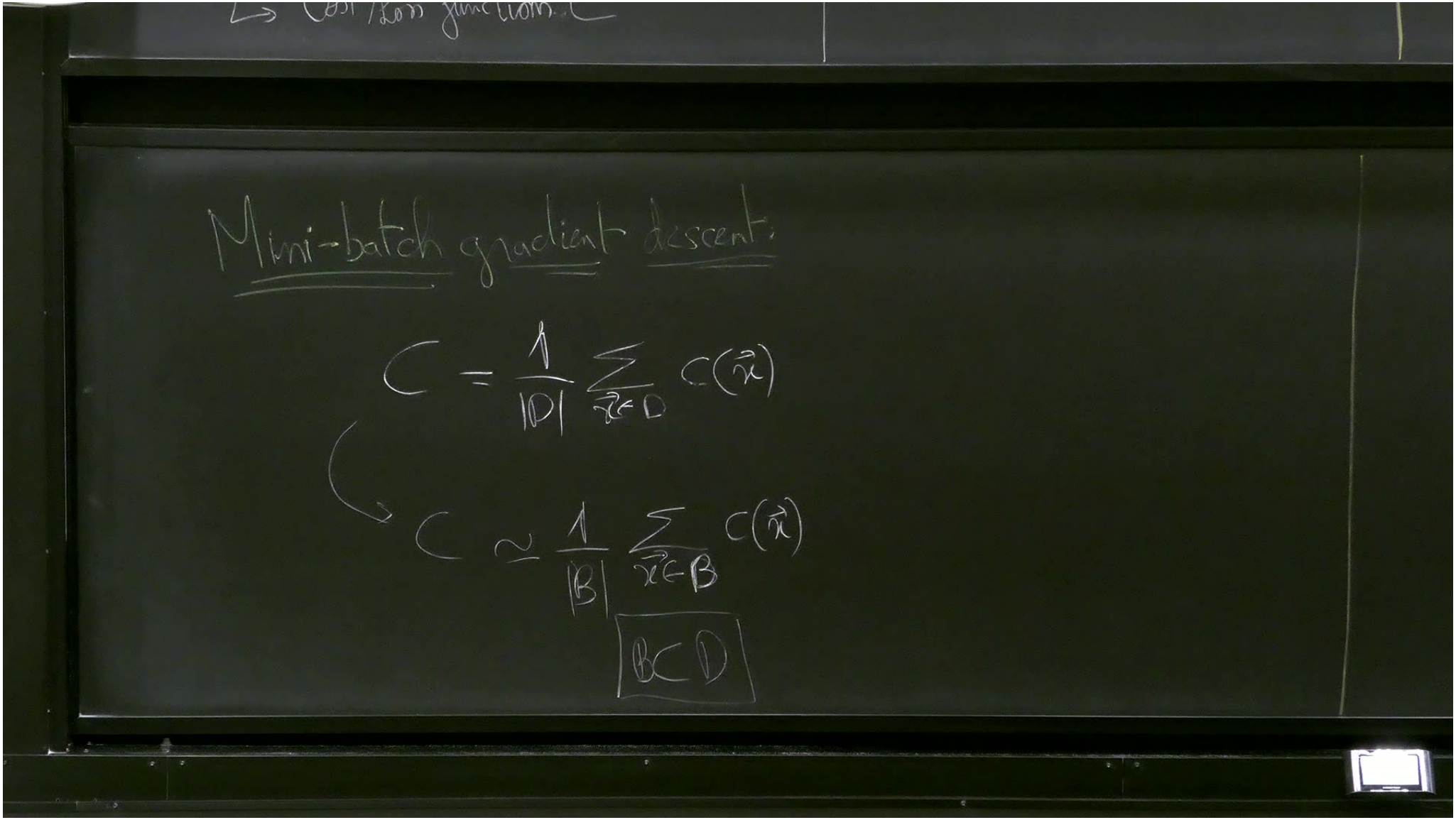

Learning Algorithms
We use a learning algorithm (such as gradient descent) to minimize the cost function C
with respect to all parameters.
$\frac{\partial C}{\partial w_{ij}^{(\ell)}} \simeq 0$
$\frac{\partial C}{\partial b_j^{(\ell)}} \simeq 0$
$\forall\, i,j,\ell$
1

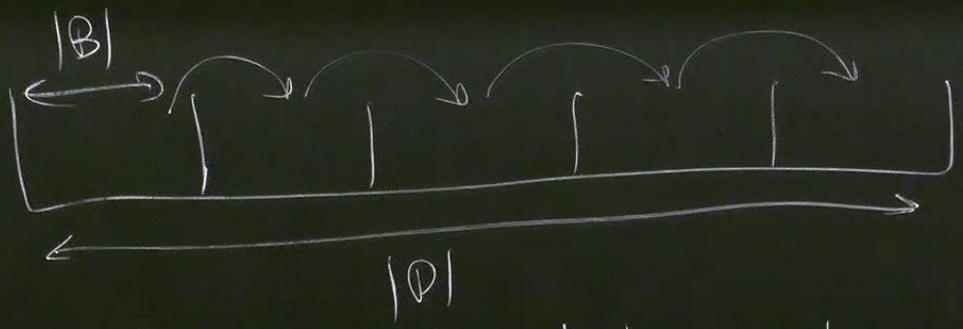
→ Unlike linear regression, as the # of parameters (weights & biases) increases, it quickly becomes impossible to solve (1) analytically.

→ Recall gradient descent (GD)

$$\vec{p} \leftarrow \vec{p} - \eta \nabla_{\vec{p}} C \quad (\vec{p} = \text{vector of parameters}).$$

→ Today we'll examine some alternative/improvements to GD.

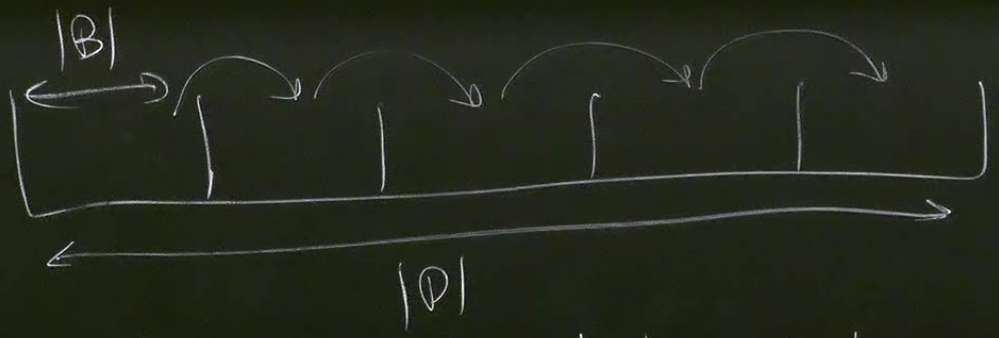

↳ Cost/Loss functions: C
Mini-batch gradient descent:
$C = \frac{1}{|D|} \sum_{\vec{x} \in D} C(\vec{x})$
$C \simeq \frac{1}{|B|} \sum_{\vec{x} \in B} C(\vec{x})$
BCD



Idea: approximate C and its gradient $\nabla_p C$ by summing over a.

randomly chosen mini-batch B

$\frac{\partial C}{\partial b_j^{(l)}} \approx 0$

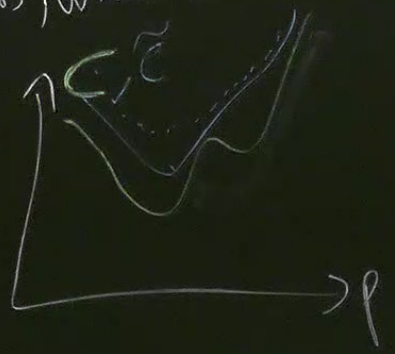


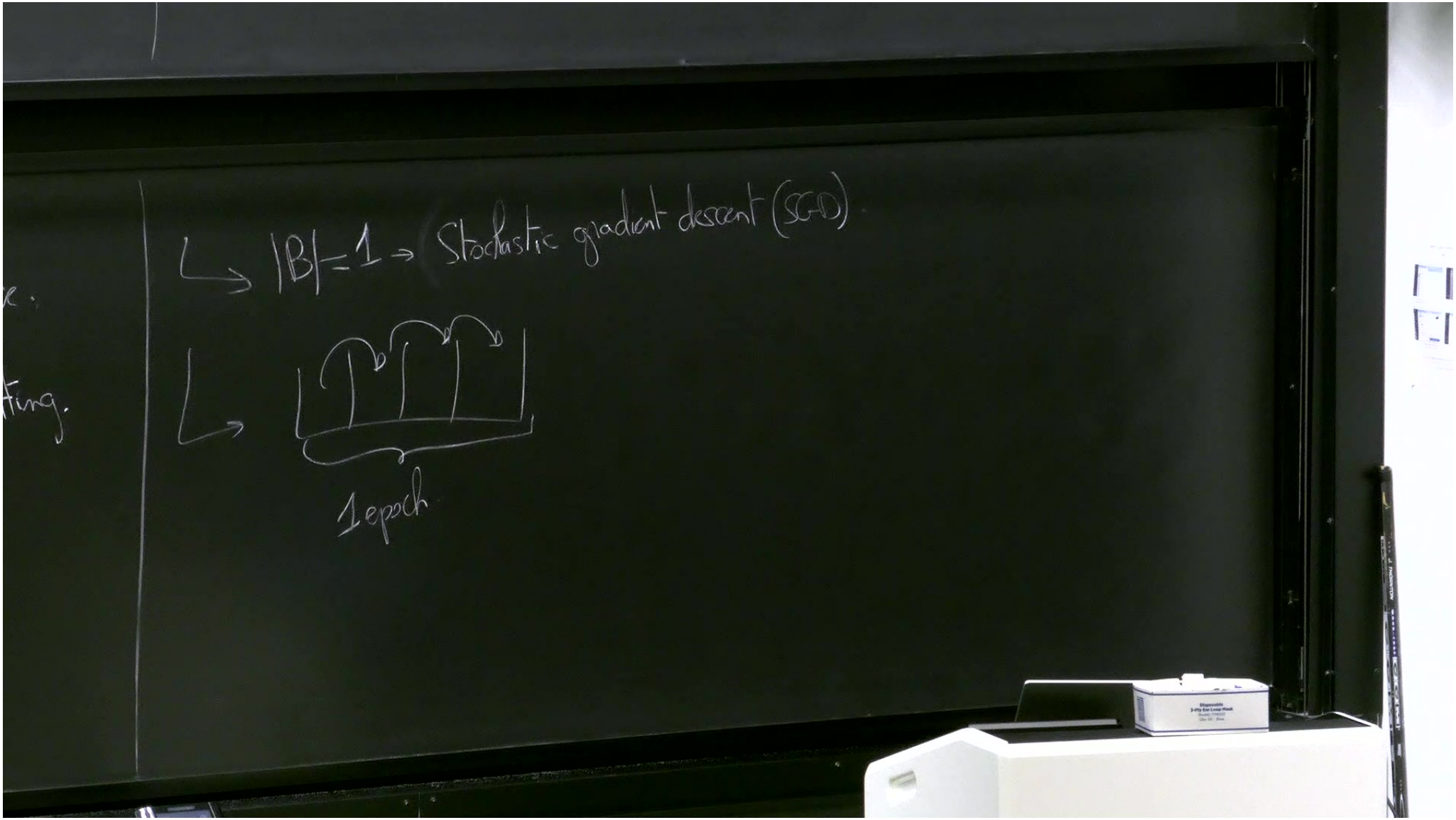
Idea: approximate C and its gradient $\nabla_p C$ by summing over a randomly chosen mini-batch B

Benefits:

↳ Speeds up the calculation of the gradient / optimization / convergence.

↳ Introduces randomness, which can decrease the chance of getting stuck in a local min.

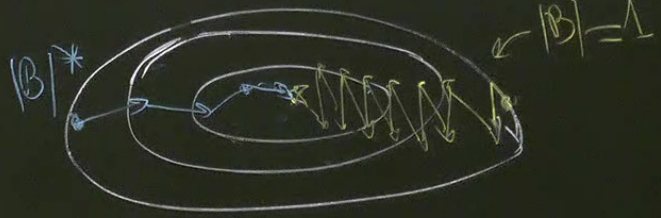


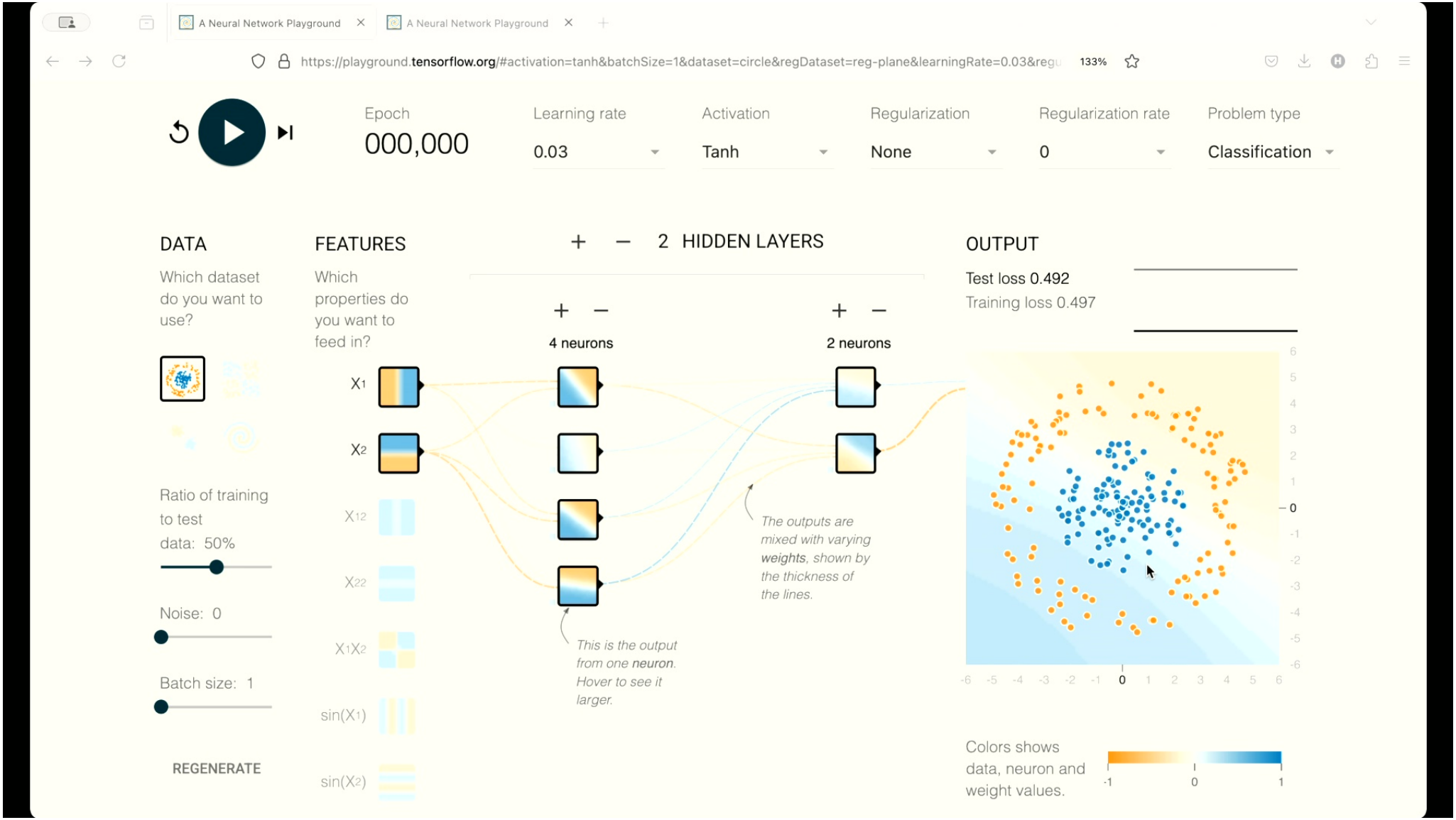

↳ |B| = 1 → Stochastic gradient descent (SGD)
1 epoch

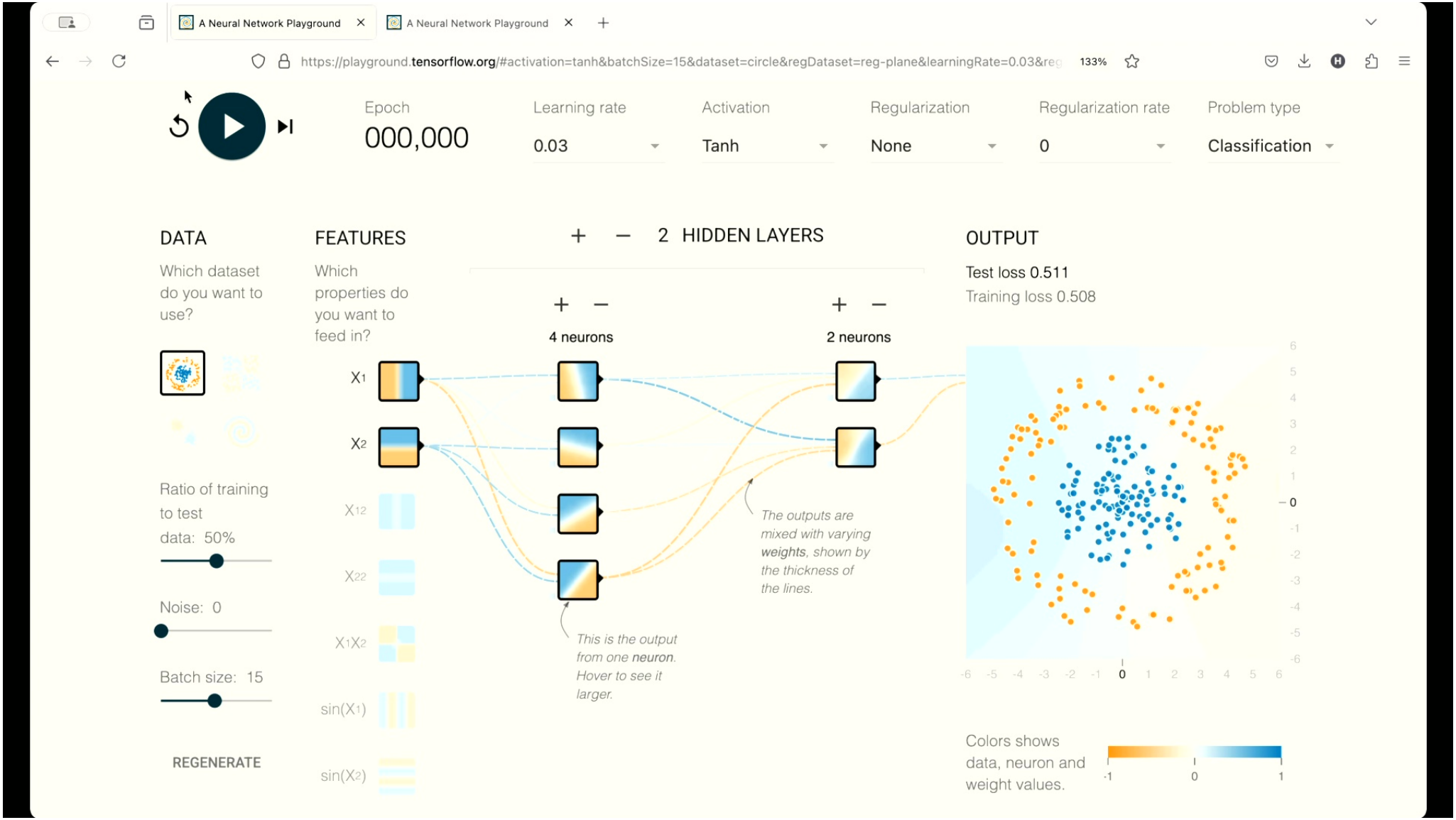
BCD

↳ $|B| = 1$ (a lot of noise).

↳ $|B|^*$ in the middle for optimal optimization.




A Neural Network Playground
A Neural Network Playground
https://playground.tensorflow.org/#activation=tanh&batchSize=1&dataset=circle&regDataset=reg-plane&learningRate=0.03&regu
133%
Epoch
000,000
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 1
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
2 HIDDEN LAYERS
4 neurons
2 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.492
Training loss 0.497
Colors shows data, neuron and weight values.
-1
0
1


A Neural Network Playground
A Neural Network Playground
https://playground.tensorflow.org/#activation=tanh&batchSize=15&dataset=circle&regDataset=reg-plane&learningRate=0.03&reg
133%
Epoch
000,000
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 15
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
2 HIDDEN LAYERS
4 neurons
2 neurons
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.511
Training loss 0.508
Colors shows data, neuron and weight values.
-1
0
1

Randomly chosen mini-batch B

Adding momentum:

$$\frac{d\vec{p}}{dt} = -\eta \nabla_{\vec{p}} C$$

velocity.

Randomly chosen mini-batch B

Adding momentum:

$$\frac{d\vec{p}}{dt} = -\eta \nabla_{\vec{p}} C \longrightarrow \frac{d\vec{v}}{dt} = -\nabla_{\vec{p}} C - \lambda \vec{v}$$

$$\frac{d\vec{p}}{dt} = v$$

$\frac{d\vec{p}}{dt}$: Velocity.

$\Delta t = 1$

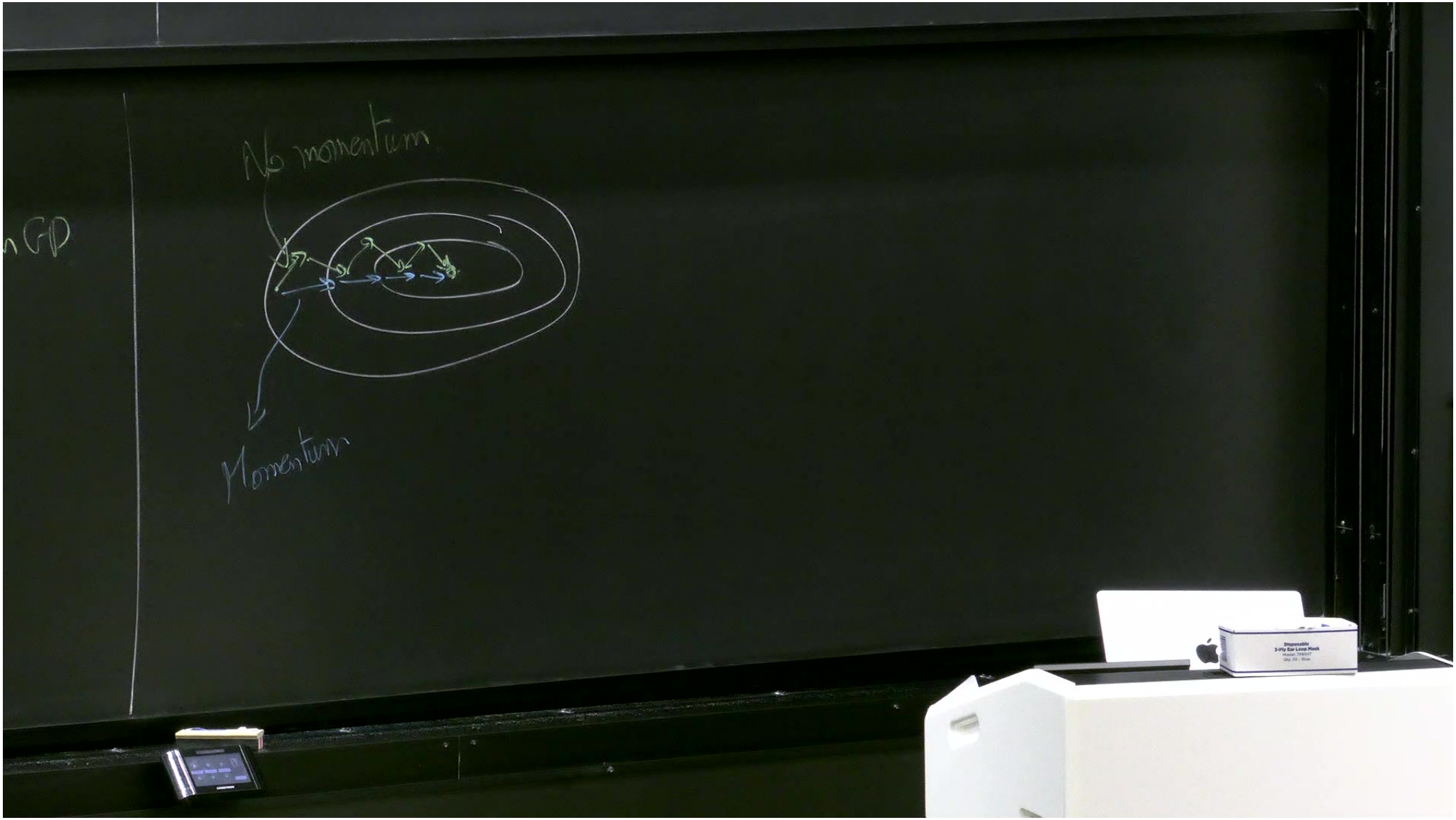
$$\vec{v}_{i+1} = \vec{v}_i + \nabla_{\vec{p}} C - \lambda \vec{v}$$

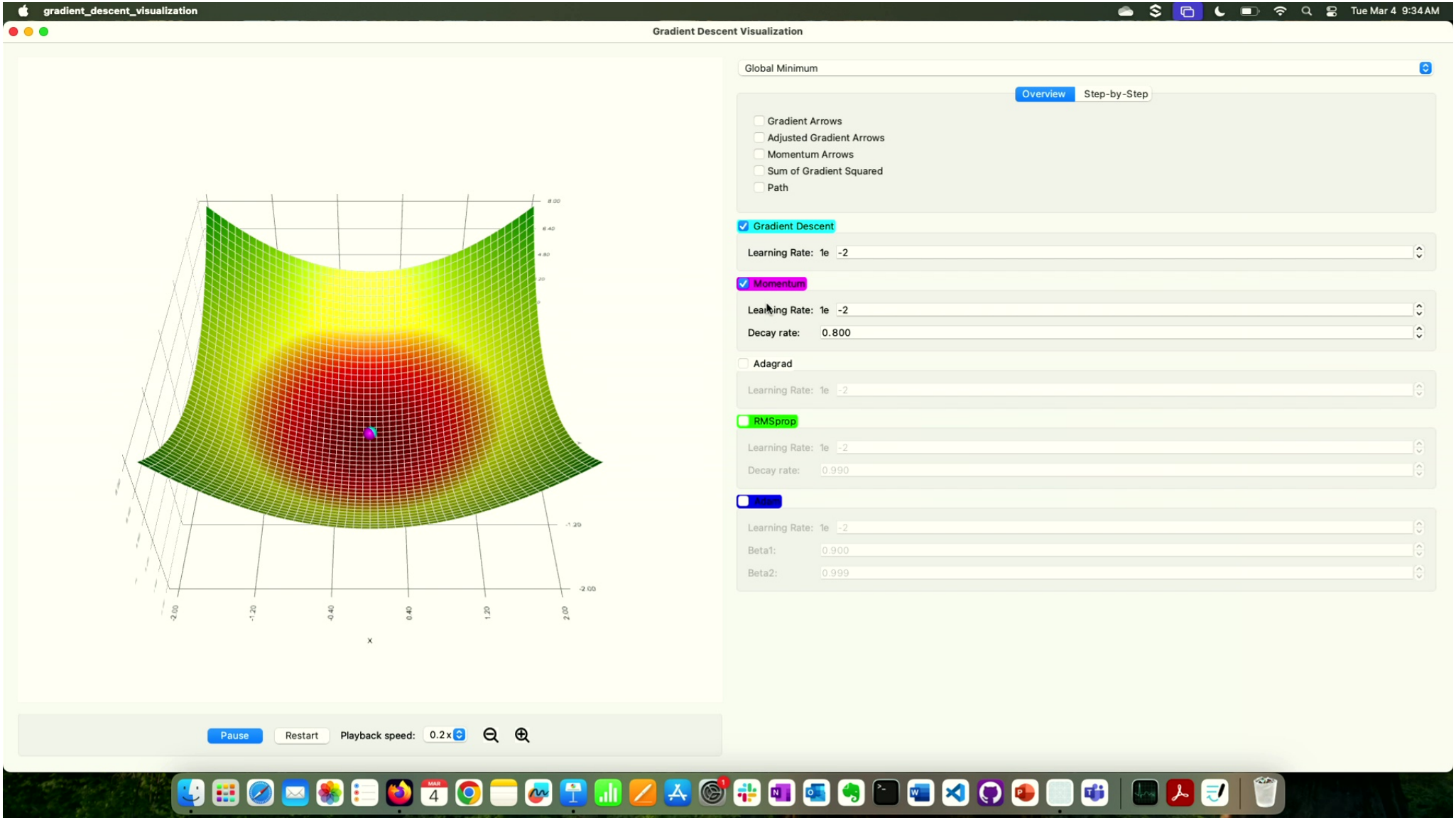
$$\vec{v}_{i+1} = \underbrace{(1-\lambda)}_{\beta} \vec{v}_i + \nabla_{\vec{p}} C$$

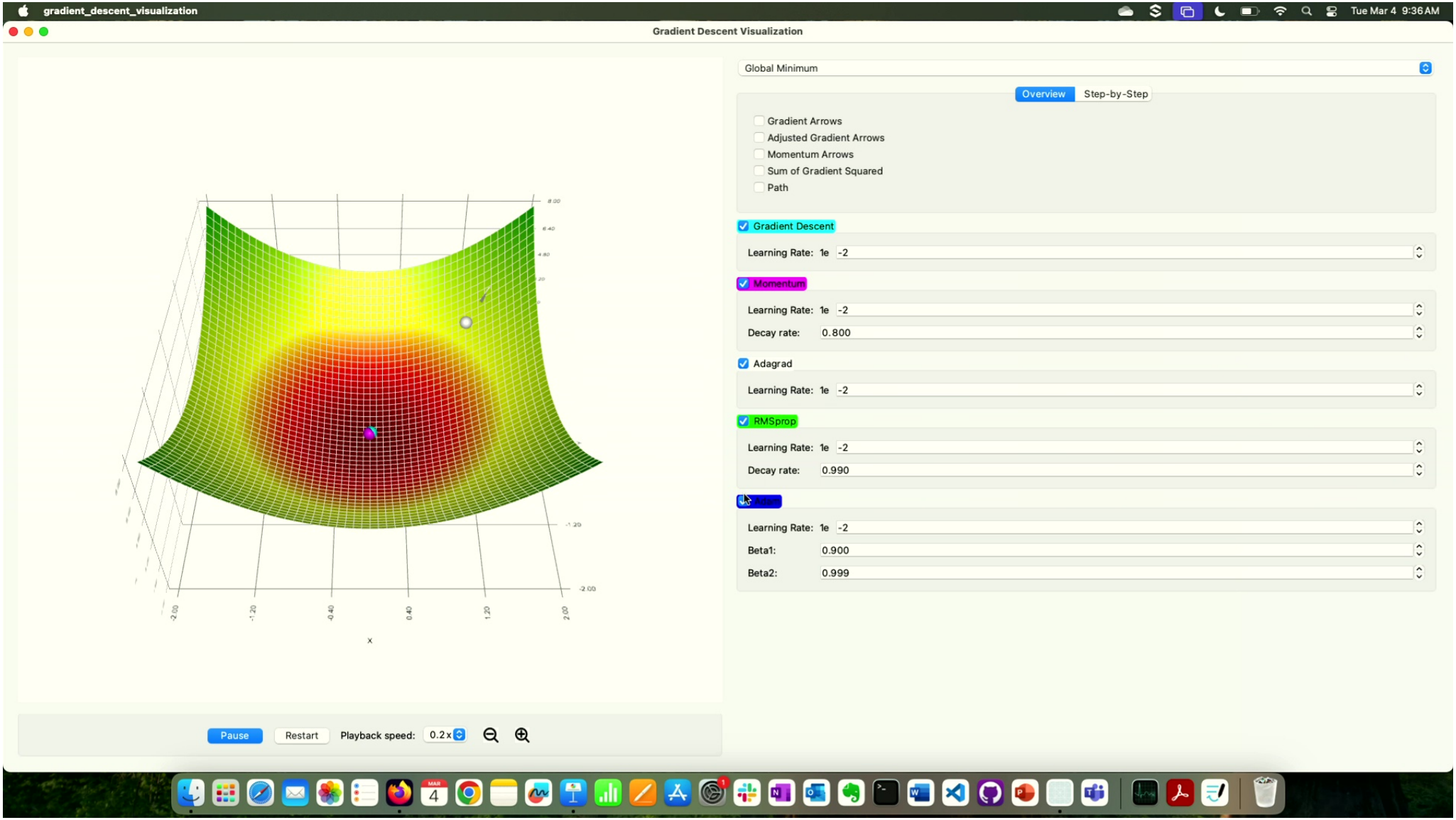
β "momentum"

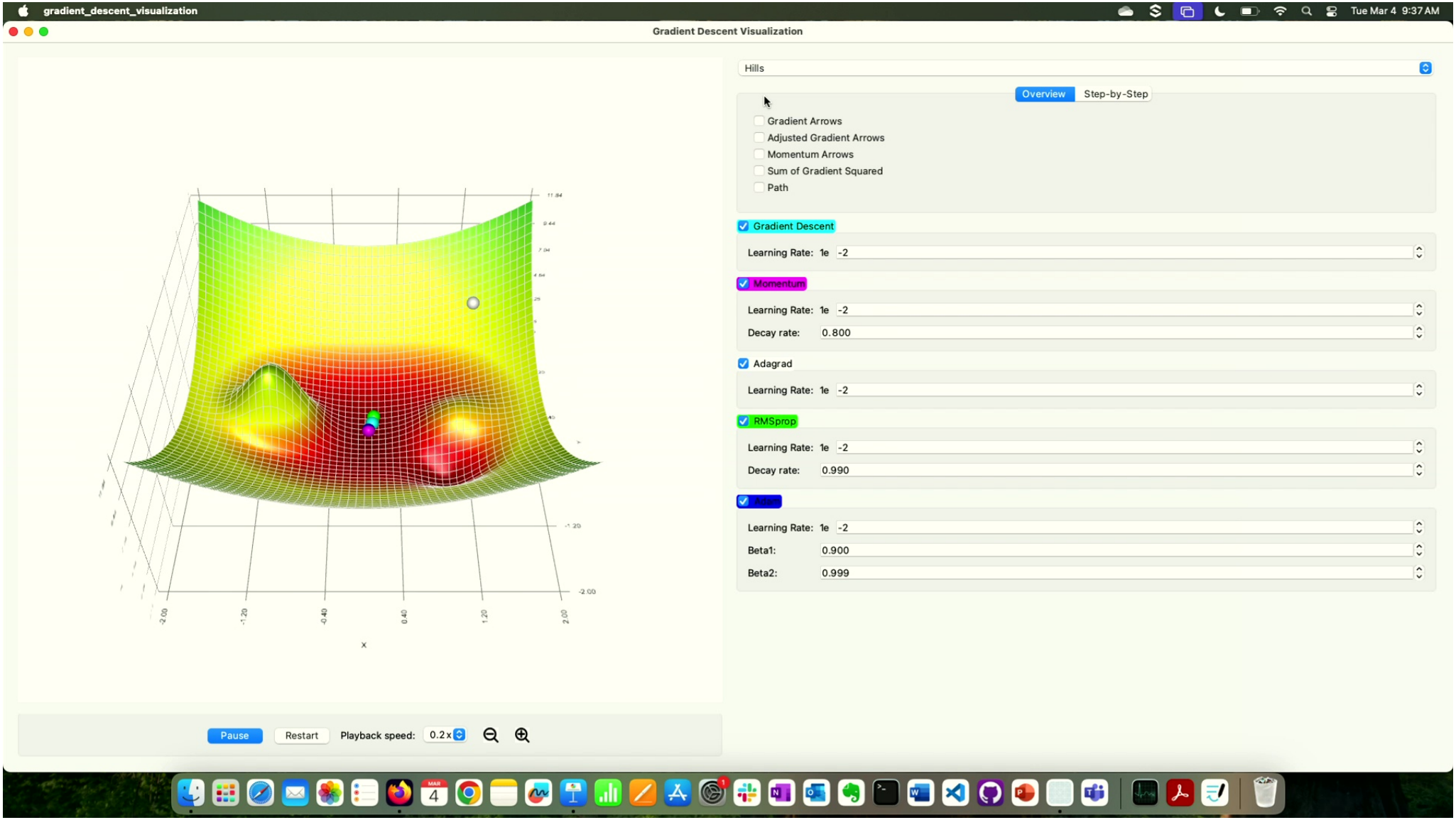
$$\vec{p}_{i+1} = \vec{p}_i - \eta \vec{v}_i$$

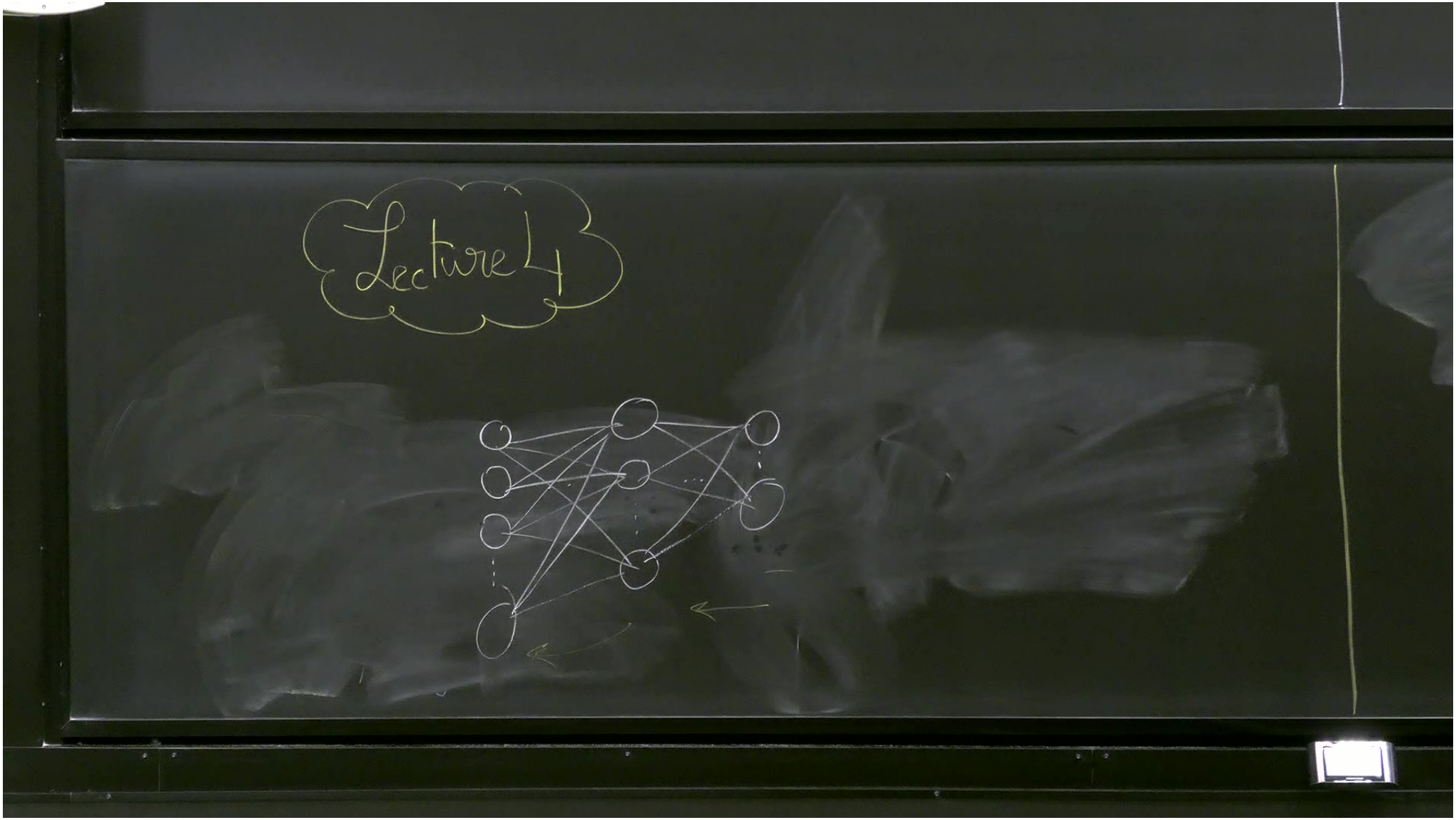
$\beta = 0$ back to regular GD


No momentum
Momentum


gradient_descent_visualization
Tue Mar 4 9:34 AM
Gradient Descent Visualization
Global Minimum
Overview
Step-by-Step
Gradient Arrows
Adjusted Gradient Arrows
Momentum Arrows
Sum of Gradient Squared
Path
Gradient Descent
Learning Rate: 1e -2
Momentum
Learning Rate: 1e -2
Decay rate: 0.800
Adagrad
Learning Rate: 1e -2
RMSprop
Learning Rate: 1e -2
Decay rate: 0.990
Adam
Learning Rate: 1e -2
Beta1: 0.900
Beta2: 0.999
Pause
Restart
Playback speed: 0.2x
x


gradient_descent_visualization
Tue Mar 4 9:36 AM
Gradient Descent Visualization
Global Minimum
Overview
Step-by-Step
Gradient Arrows
Adjusted Gradient Arrows
Momentum Arrows
Sum of Gradient Squared
Path
Gradient Descent
Learning Rate: 1e -2
Momentum
Learning Rate: 1e -2
Decay rate: 0.800
Adagrad
Learning Rate: 1e -2
RMSprop
Learning Rate: 1e -2
Decay rate: 0.990
Adam
Learning Rate: 1e -2
Beta1: 0.900
Beta2: 0.999
Pause
Restart
Playback speed: 0.2x
x

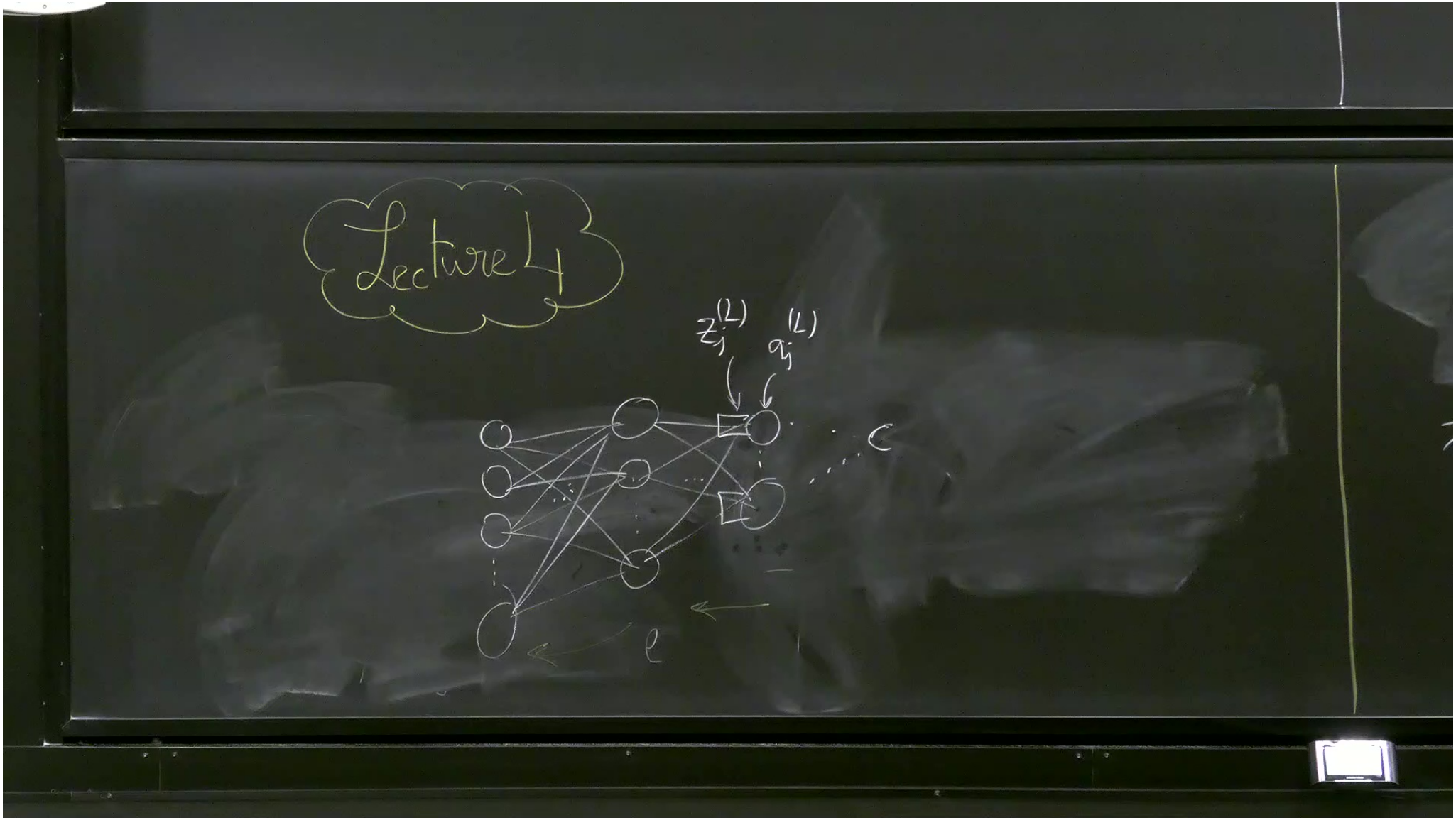

gradient_descent_visualization
Tue Mar 4 9:37 AM
Gradient Descent Visualization
Hills
Overview
Step-by-Step
Gradient Arrows
Adjusted Gradient Arrows
Momentum Arrows
Sum of Gradient Squared
Path
Gradient Descent
Learning Rate: 1e -2
Momentum
Learning Rate: 1e -2
Decay rate: 0.800
Adagrad
Learning Rate: 1e -2
RMSprop
Learning Rate: 1e -2
Decay rate: 0.990
Adam
Learning Rate: 1e -2
Beta1: 0.900
Beta2: 0.999
Pause
Restart
Playback speed: 0.2x
x

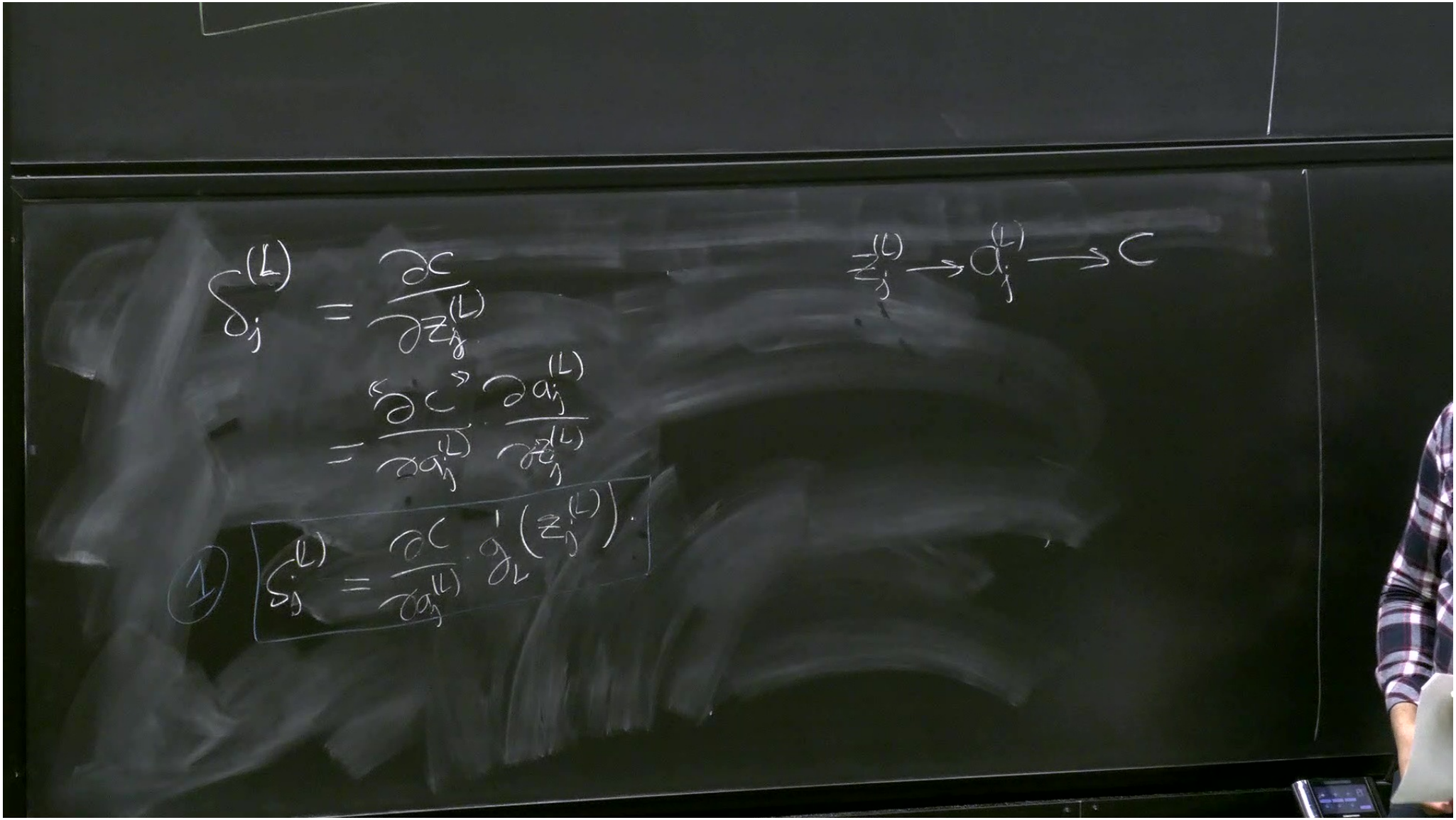

Lecture 4

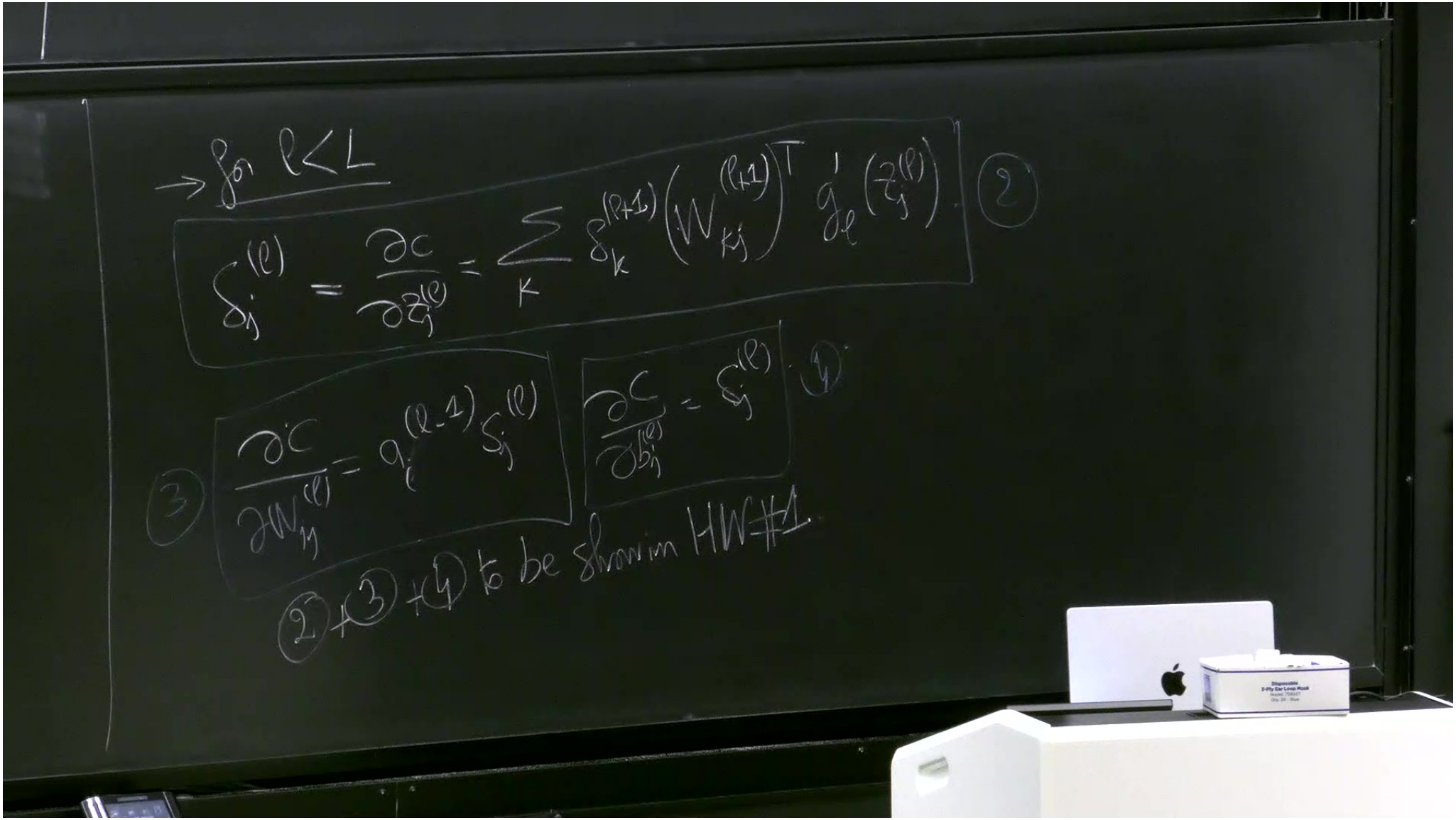
$$C = \frac{1}{|D|} \sum_{x \in D} c(\vec{x})$$

$$\frac{\partial c(\vec{x})}{\partial w_{ij}^{(\ell)}}, \quad \frac{\partial c(\vec{x})}{\partial b_j^{(\ell)}}$$

$$a_j^{(\ell)} = g_\ell\left(z_j^{(\ell)}\right) \qquad z_j^{(\ell)} = \sum_{i=1}^{n_{(\ell-1)}} a_i^{(\ell-1)} w_{ij}^{(\ell)} + b_j^{(\ell)} \; ; \; \delta_j^{(\ell)} = \frac{\partial c}{\partial z_j^{(\ell)}}$$


Lecture 4
$z_j^{(L)}$
$a_j^{(L)}$
C
L


$\delta_j^{(L)} = \frac{\partial C}{\partial z_j^{(L)}}$
$= \frac{\partial C}{\partial a_j^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}}$
(1) $\delta_j^{(L)} = \frac{\partial C}{\partial a_j^{(L)}} g'(z_j^{(L)})$
$z_j^{(L)} \rightarrow a_j^{(L)} \rightarrow C$


→ for l<L
$\delta_j^{(l)} = \frac{\partial C}{\partial z_j^{(l)}} = \sum_k \delta_k^{(l+1)} \left(W_{kj}^{(l+1)}\right)^T g_l'\left(z_j^{(l)}\right)$ (2)
(3) $\frac{\partial C}{\partial W_{ij}^{(l)}} = a_i^{(l-1)} \delta_j^{(l)}$
$\frac{\partial C}{\partial b_j^{(l)}} = \delta_j^{(l)}$ (4)
(2)+(3)+(4) to be shown in HW#1

Credit: Lauren Hayward

Overfitting

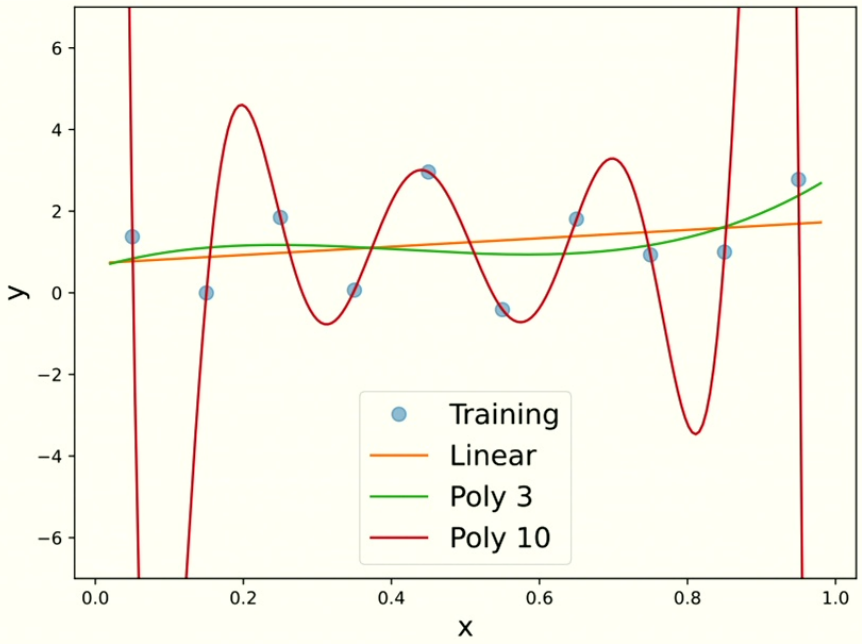
Training data:

```
N_train = 10
x = np.linspace(0.05,0.95,N_train)
s = np.random.randn(N_train)
y = 2*x+s
```

Models for fitting:

Polynomials of degree 1, 3, and 10

Question: Which model will give the best fit (lowest error) to the training data?



See Section II of arXiv:1803.08823 and the corresponding Notebook 1

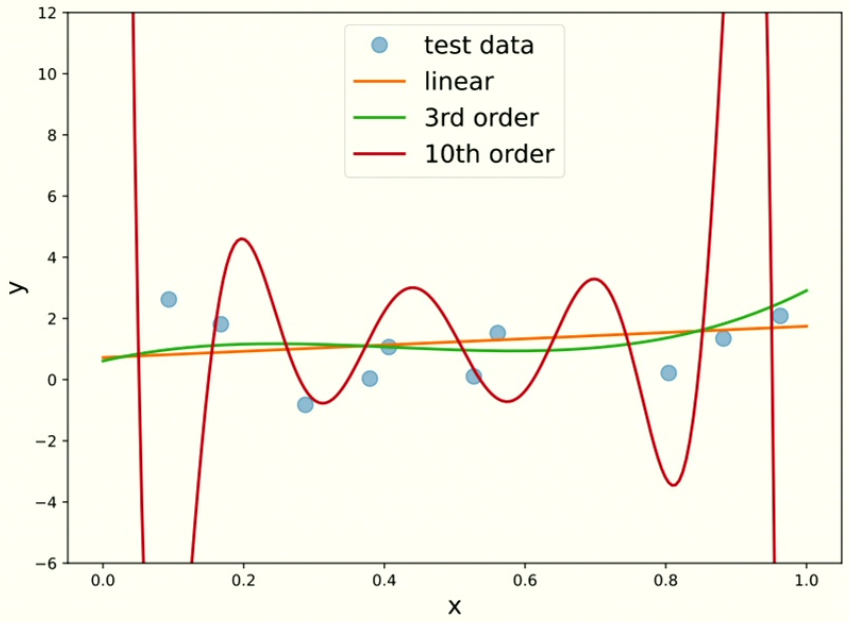
PERIMETER SCHOLARS INTERNATIONAL

Credit: Lauren Hayward

Overfitting

Testing data (from the same distribution as the training data):

```
N_test = 10
x_test = np.random.random(N_test)
s_test = np.random.randn(N_test)
y_test = 2*x_test+s_test
```

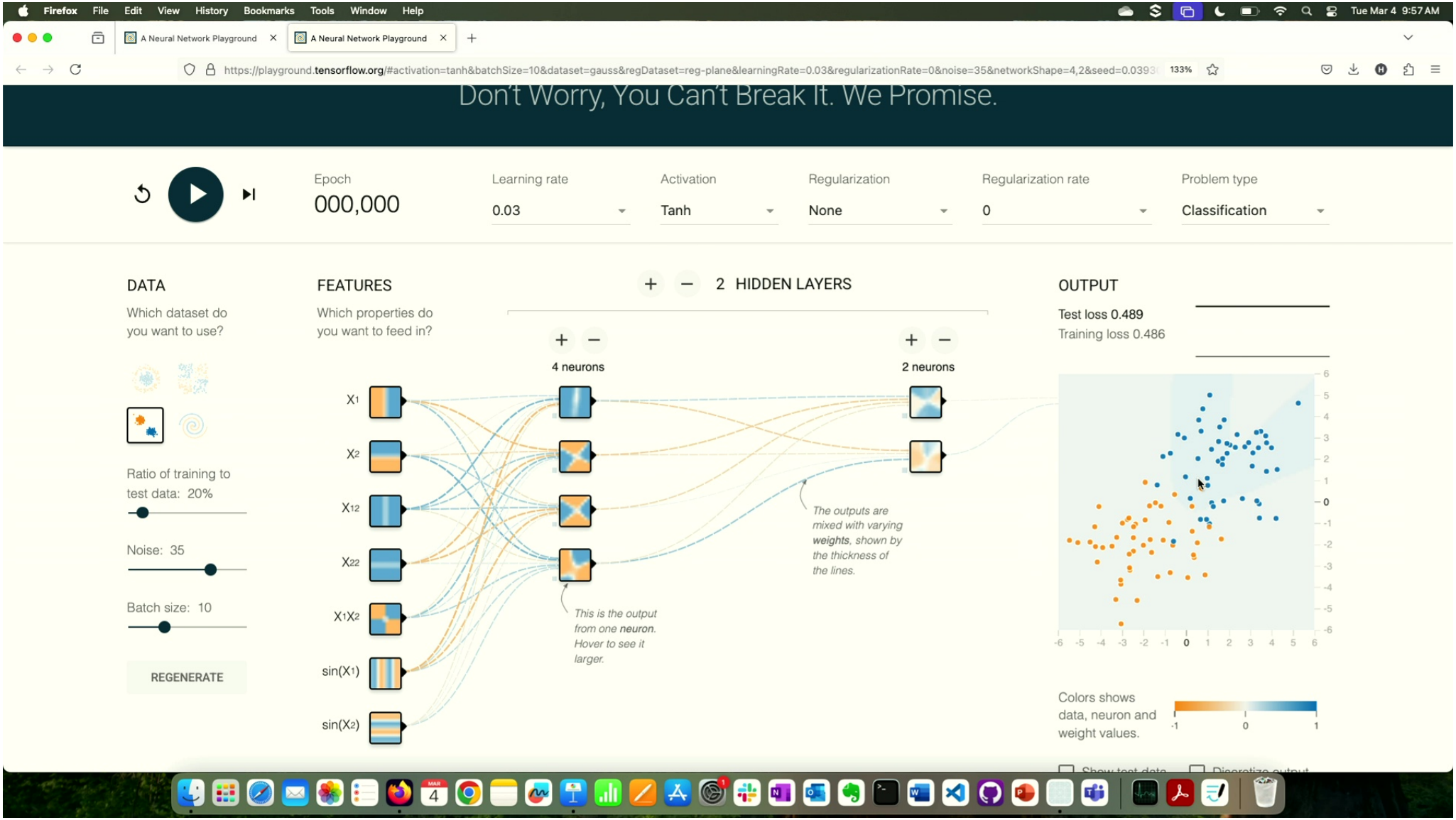


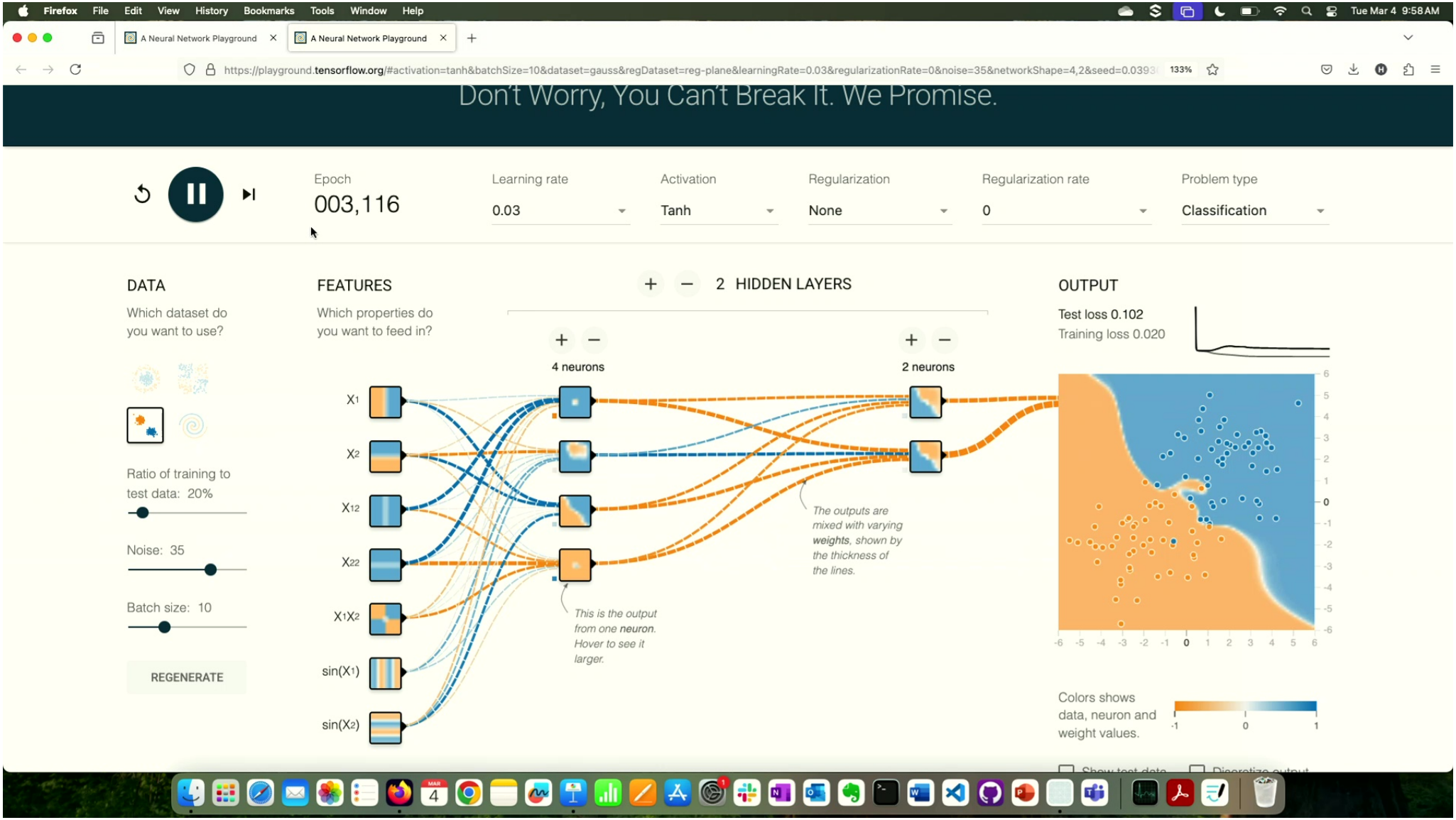
Questions:

- Which model from the previous slide will make the best predictions on the testing data?
- What would happen if we increased the number of data points in the training dataset?

See Section II of arXiv:1803.08823 and the corresponding Notebook 1

PERIMETER SCHOLARS INTERNATIONAL


Don't Worry, You Can't Break It. We Promise.
Epoch
000,000
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 20%
Noise: 35
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.489
Training loss 0.486
Colors shows data, neuron and weight values.
-1
0
1


Don't Worry, You Can't Break It. We Promise.
Epoch
003,116
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 20%
Noise: 35
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.102
Training loss 0.020
Colors shows data, neuron and weight values.