CS 383: Machine Learning

Prof Adam Poliak

Fall 2024

09/19/2024

Lecture 07

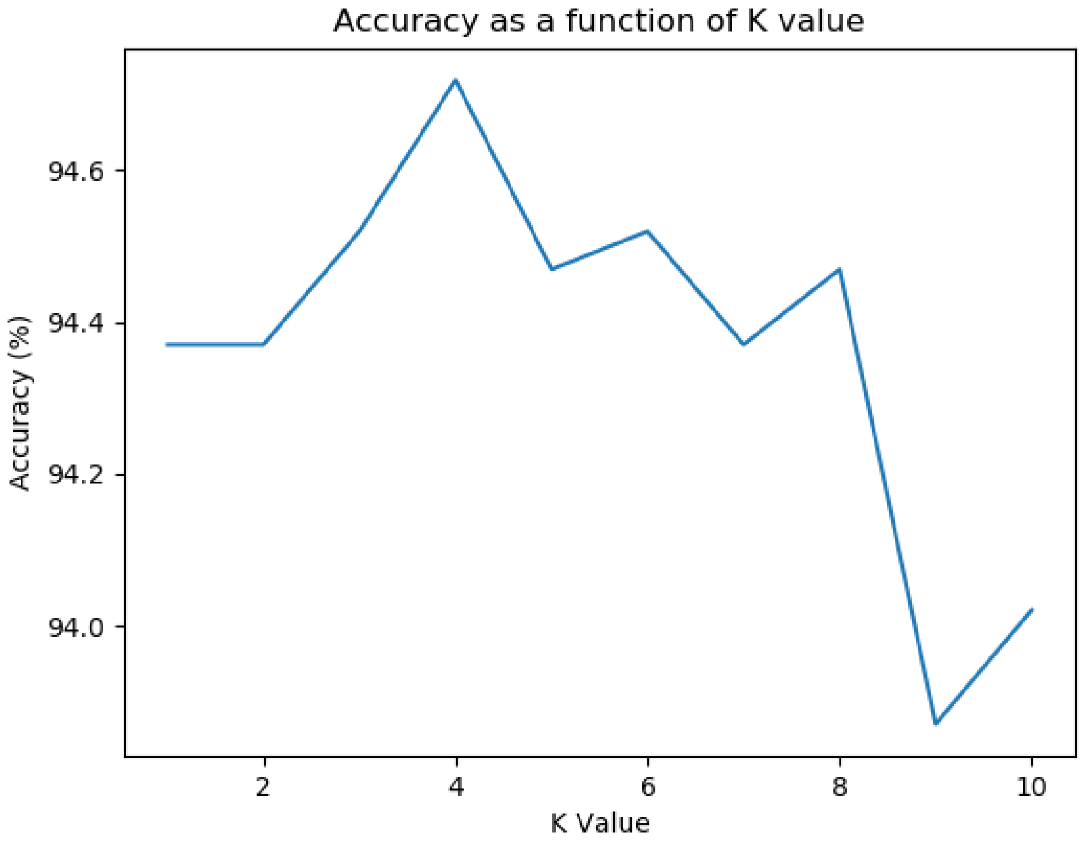
Announcements

HW02 is due Sunday night

- **Reading quiz: Thursday**
  - Duame 7.6 (2+ pages)
  - ISL 59-63 (4+ pages)

- Midterm 1: Thursday October 3rd

My accuracy


Accuracy as a function of K value
Accuracy (%)
94.6
94.4
94.2
94.0
2
4
6
8
10
K Value

Speeding up K-NN

- Runtime: exercise!
- Don't need to sort all distances – for small K, we can find the top K neighbors in linear time
- Save matrix of pair-wise distances across K
- Use less of the training data
- Put each training example in a "zone" or "cluster". For each test example, identify cluster and only consider neighbors within that cluster

Outline

Reading quiz #3

Simple linear regression

SGD (Stochastic Gradient Descent)

Normal equations solution

Outline

Reading quiz #3

Simple linear regression

SGD (Stochastic Gradient Descent)

Normal equations solution

Goals of Inference

1) Which of the features/explanatory variables/predictors (x) are associated with the response variable (y)?

2) What is the relationship between x and y?

3) Is a linear model enough?

4) Can we predict y given a new x?

Linear Regression so far

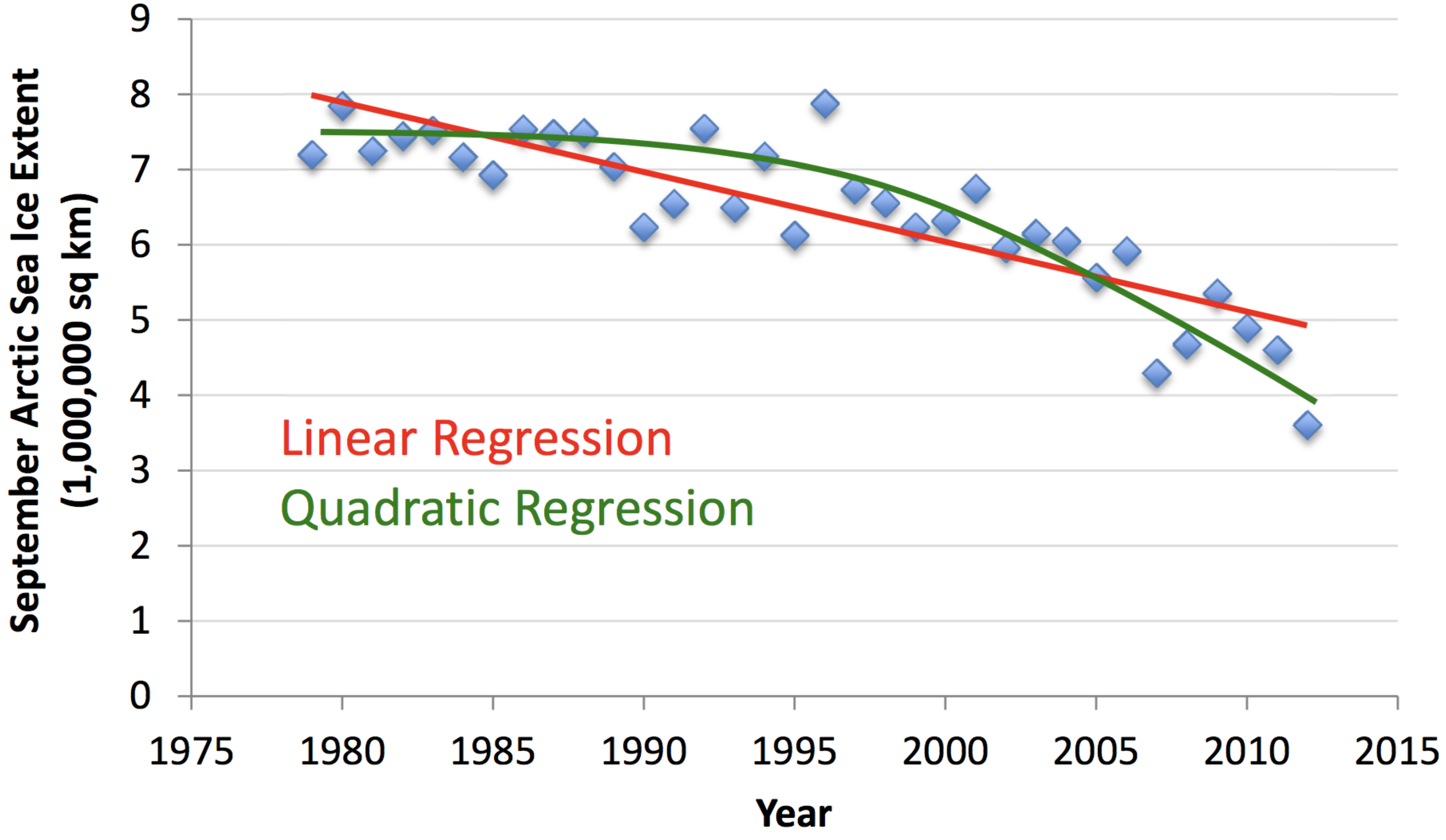
Output (*y*) is continuous, not a discrete label

Learned model: *linear function* mapping input to output (a *weight* for each feature + *bias*)

Goal: minimize the *RSS* (residual sum of square) or *SSE* (sum of squared errors)

$$RSS = \sum_{i}^{n} (y_i - \hat{y}_i)^2$$

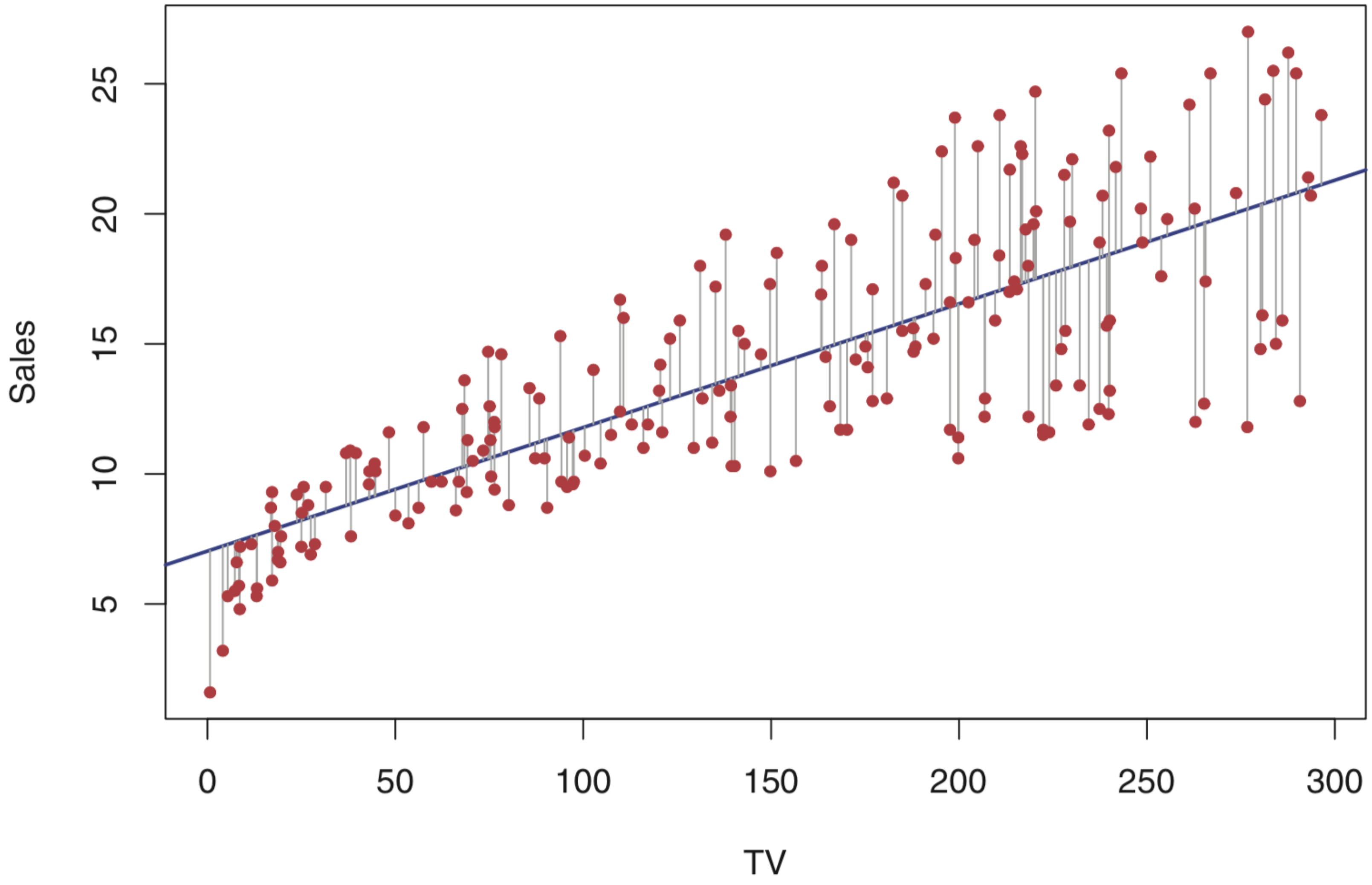
Regression Example



Slide: modified from Jessica Wu

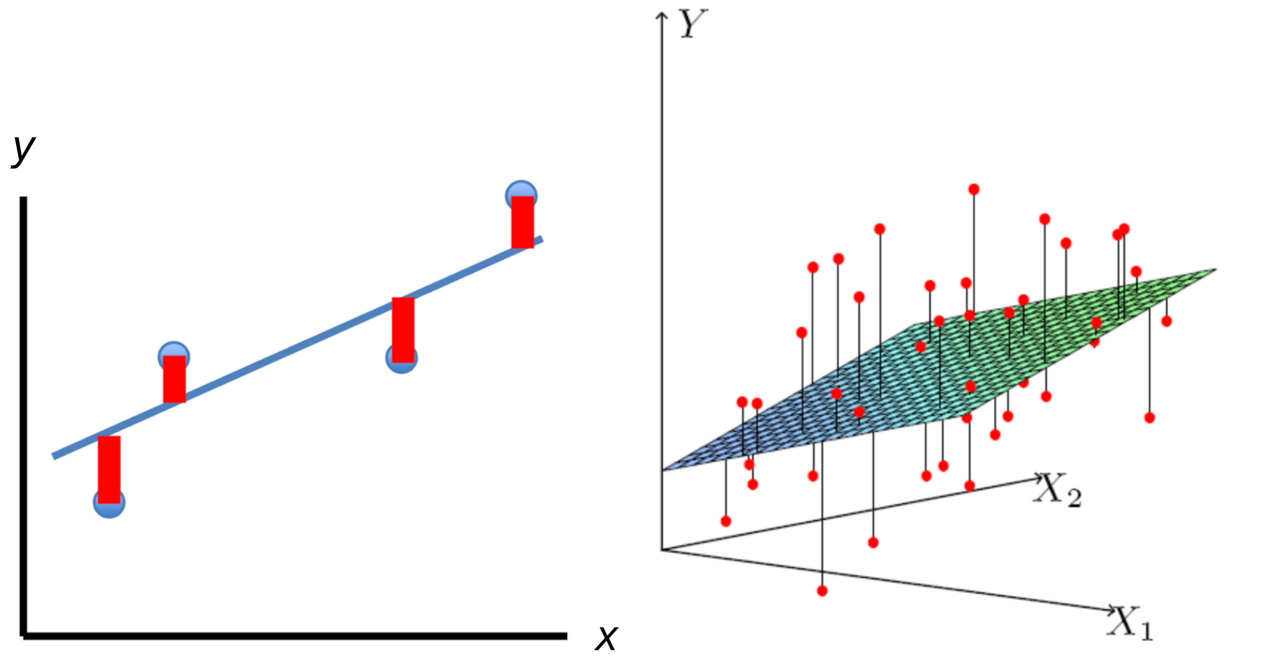
[Data from G. Witt. Journal of Statistics Education, Volume 21, Number 1 (2013)]

Example: predict sales from TV advertising budget



ISL: Figure 3.1

Cost Function: sum of squared errors



p=1 *p*=2

Outline

Reading quiz #3

Simple linear regression

SGD (Stochastic Gradient Descent)

Normal equations solution

Process Learning Weights

1. Randomly initialize weights

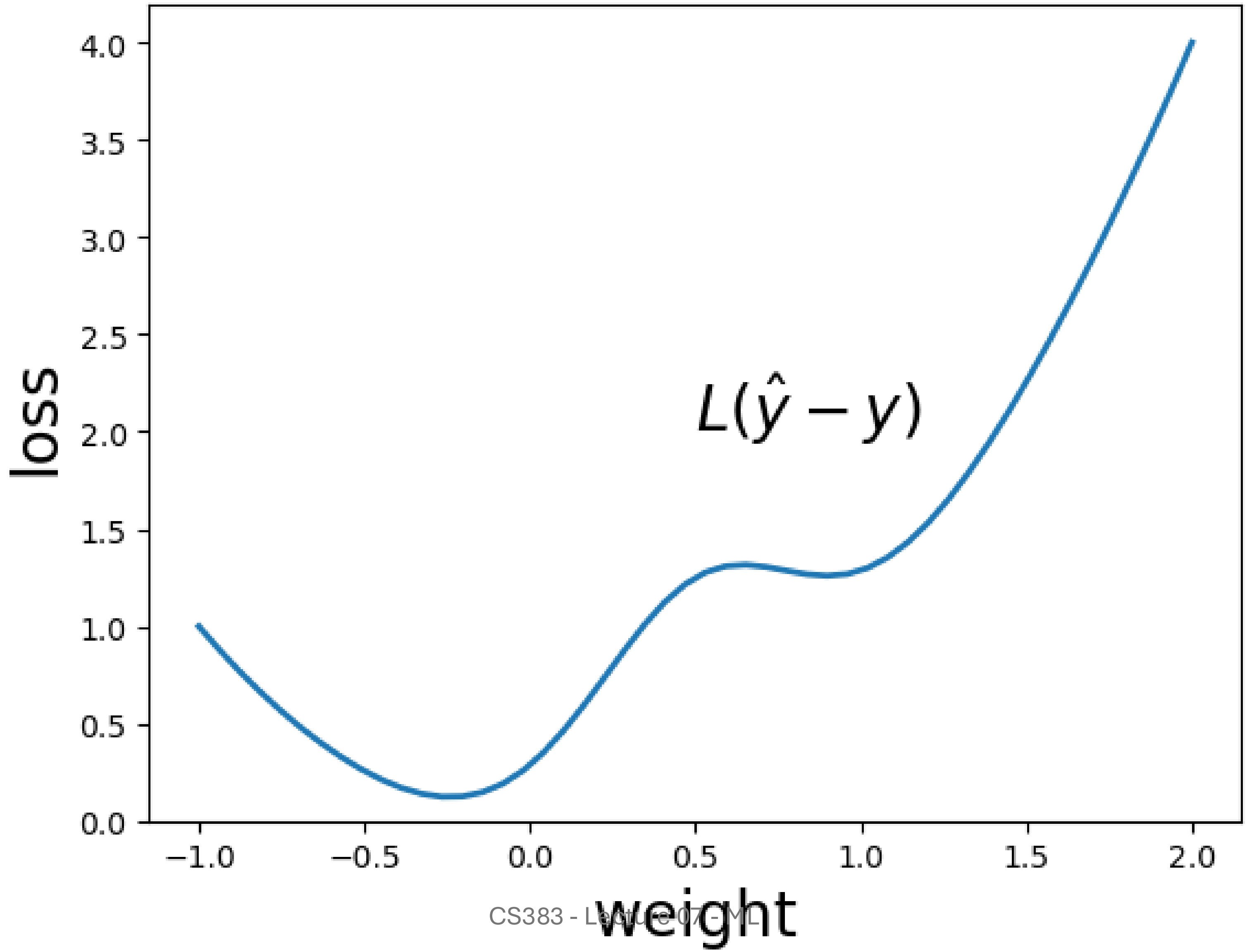
2. Make predictions $\hat{y}$

3. Compute loss function - quantify how close $\hat{y}$ *and* y are

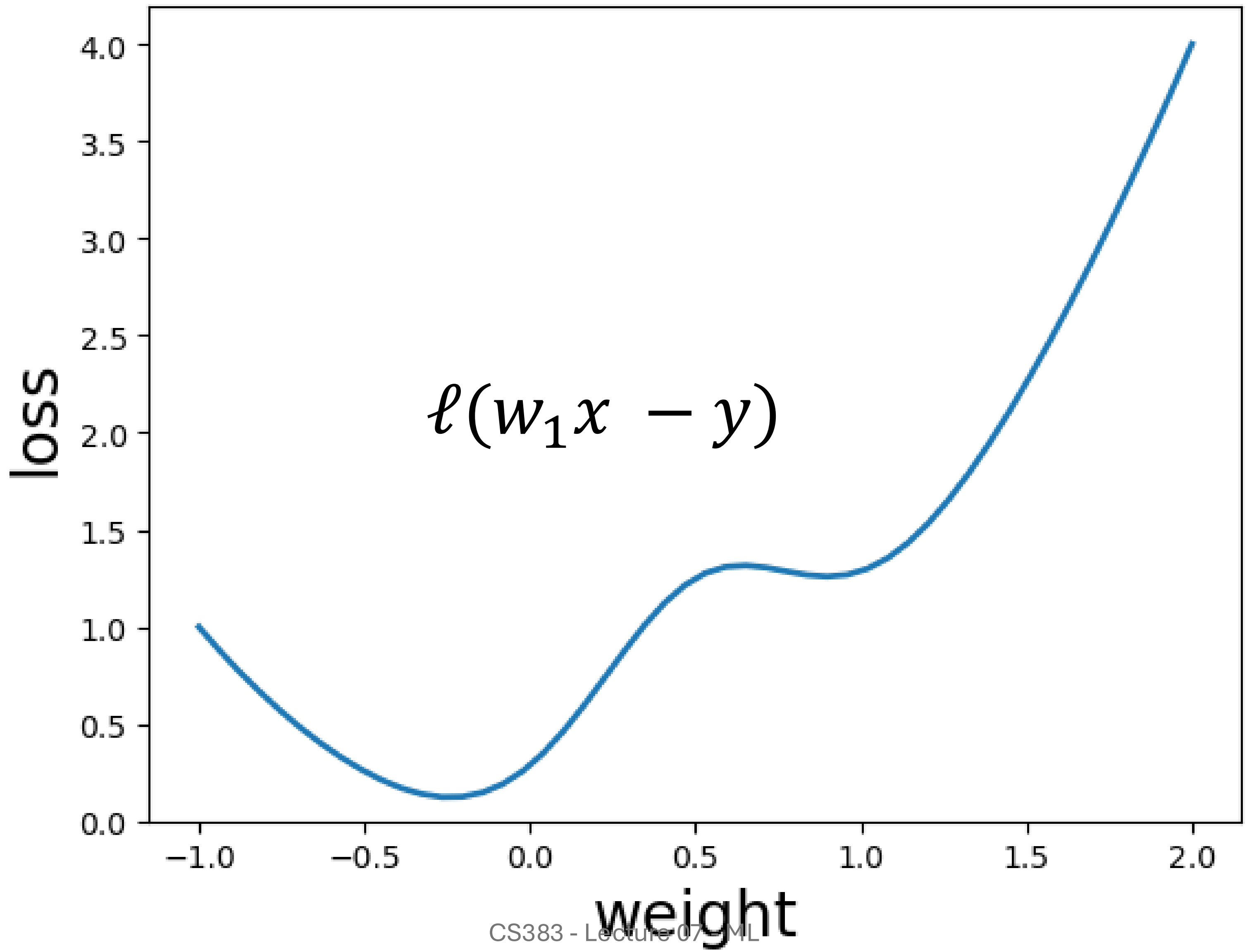
4. Update weights accordingly based on the loss function
   aka Optimization

5. Repeat 2-4

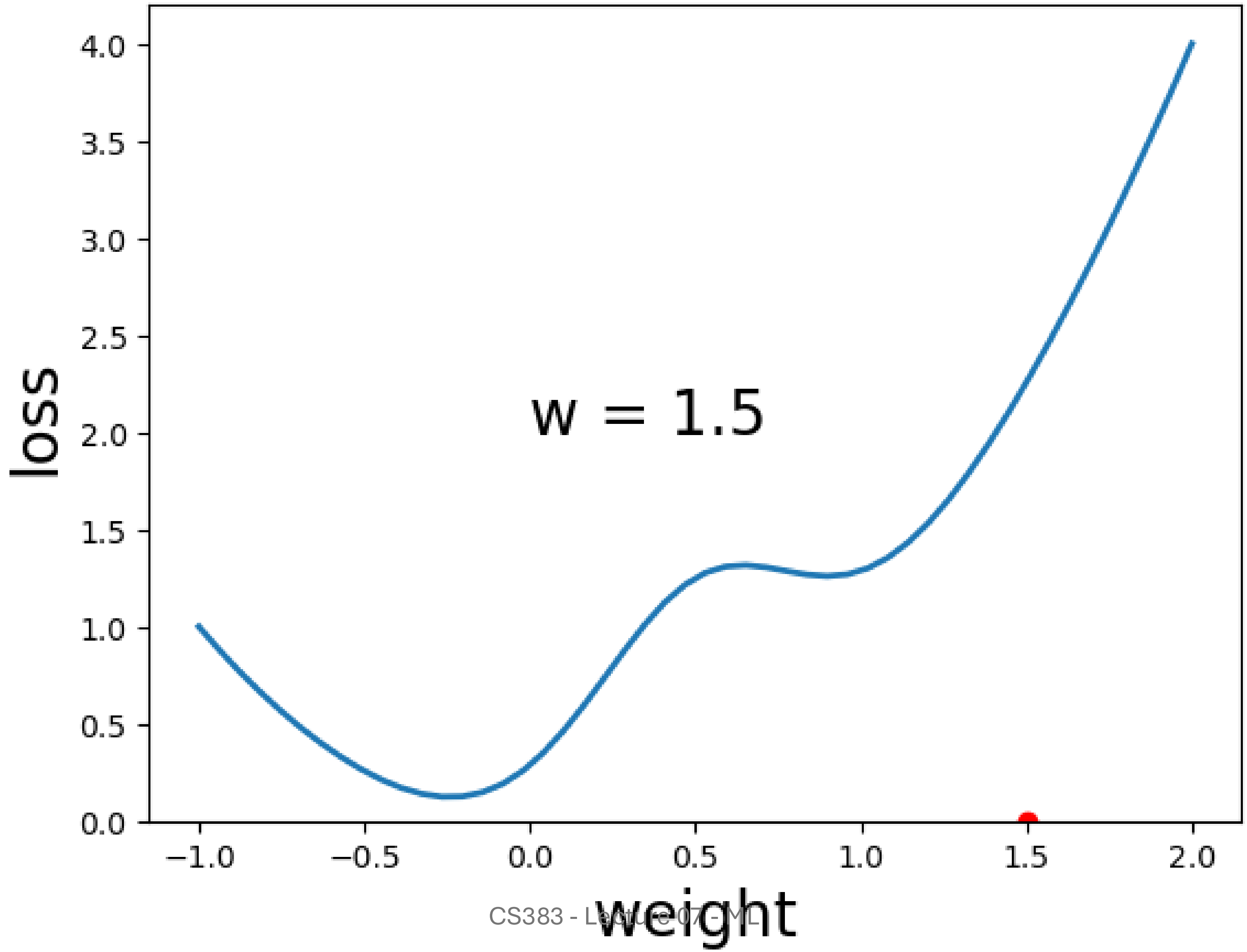
Find weights that minimize the loss



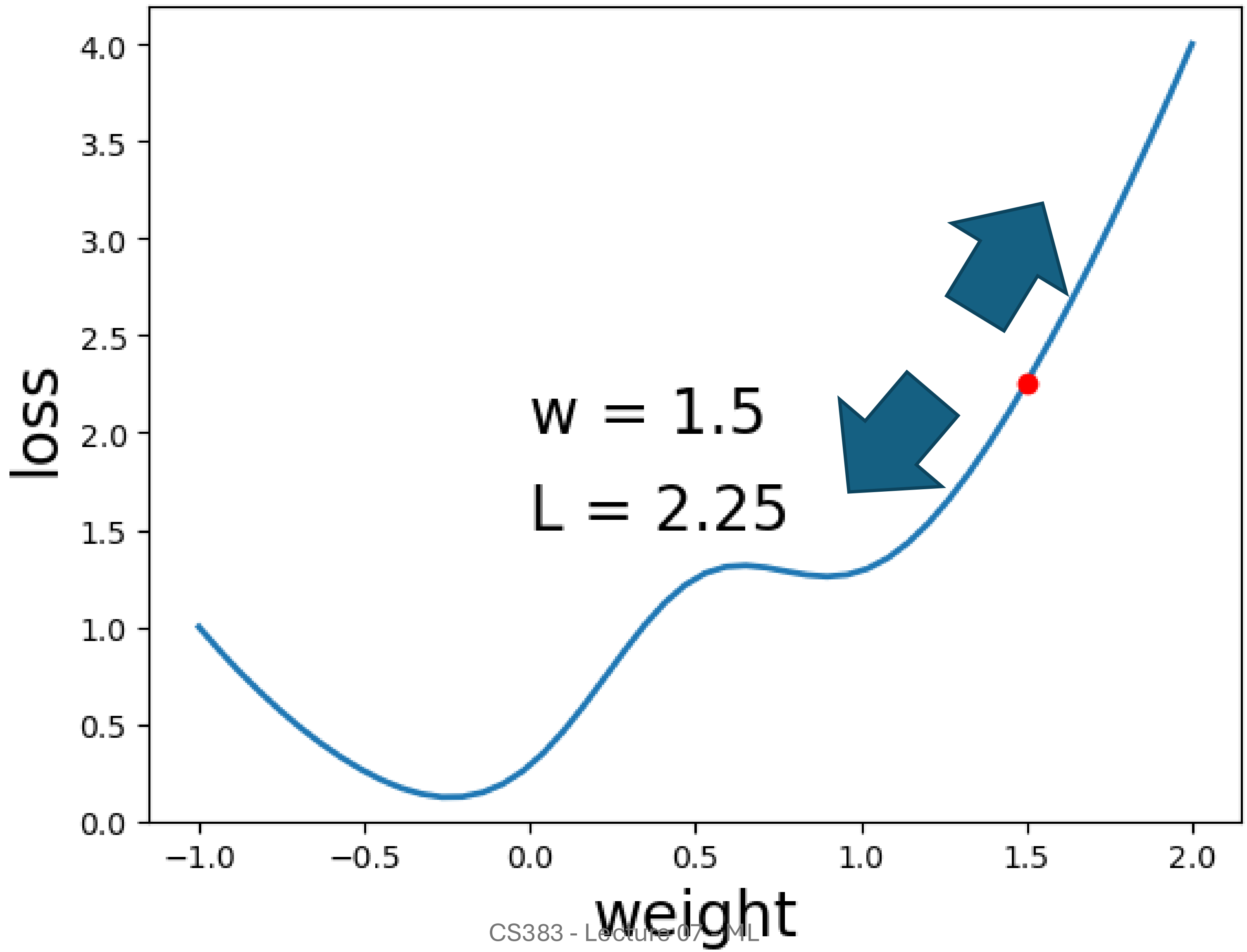
Find weights that minimize the loss



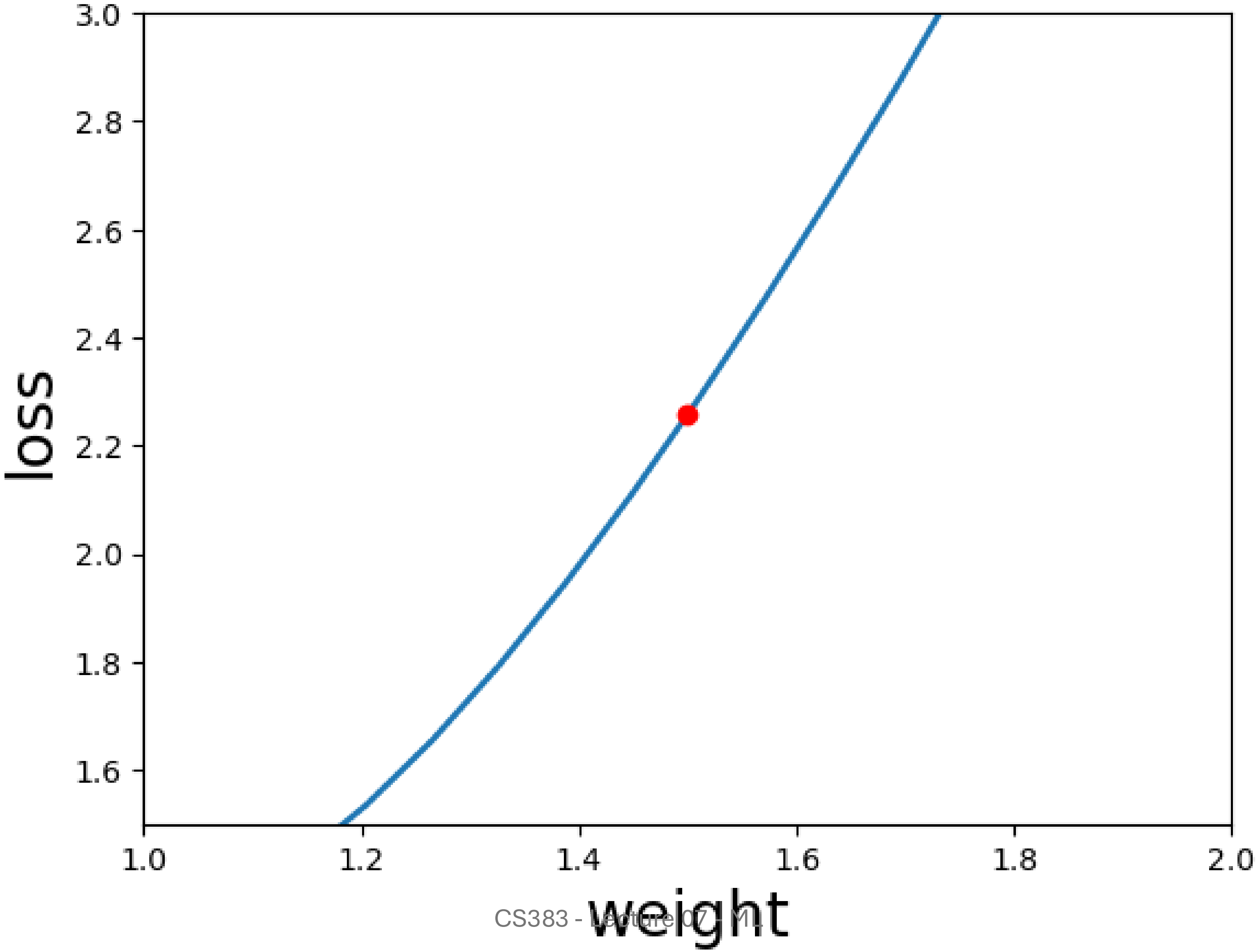
Find weights that minimize the loss



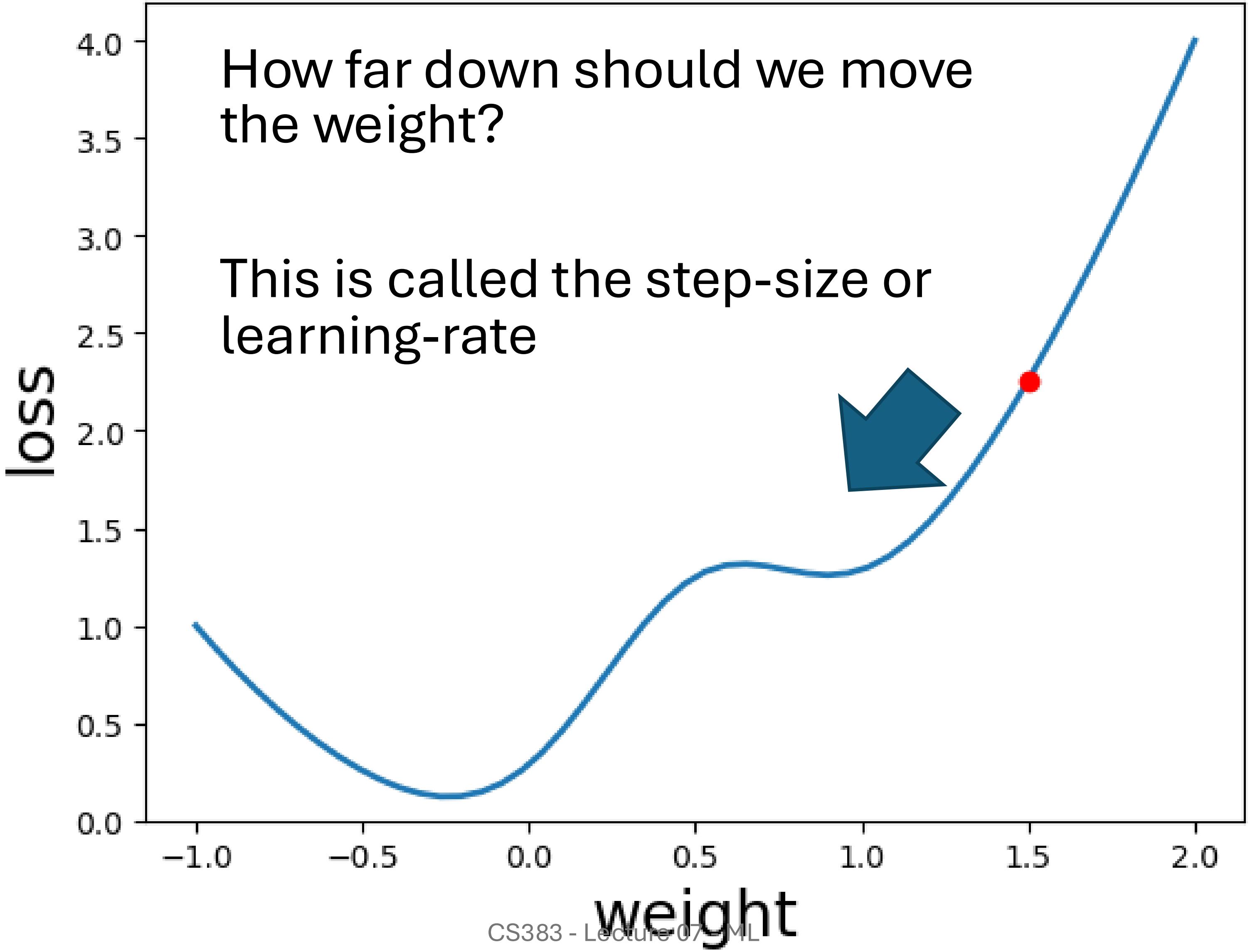
Find weights that minimize the loss



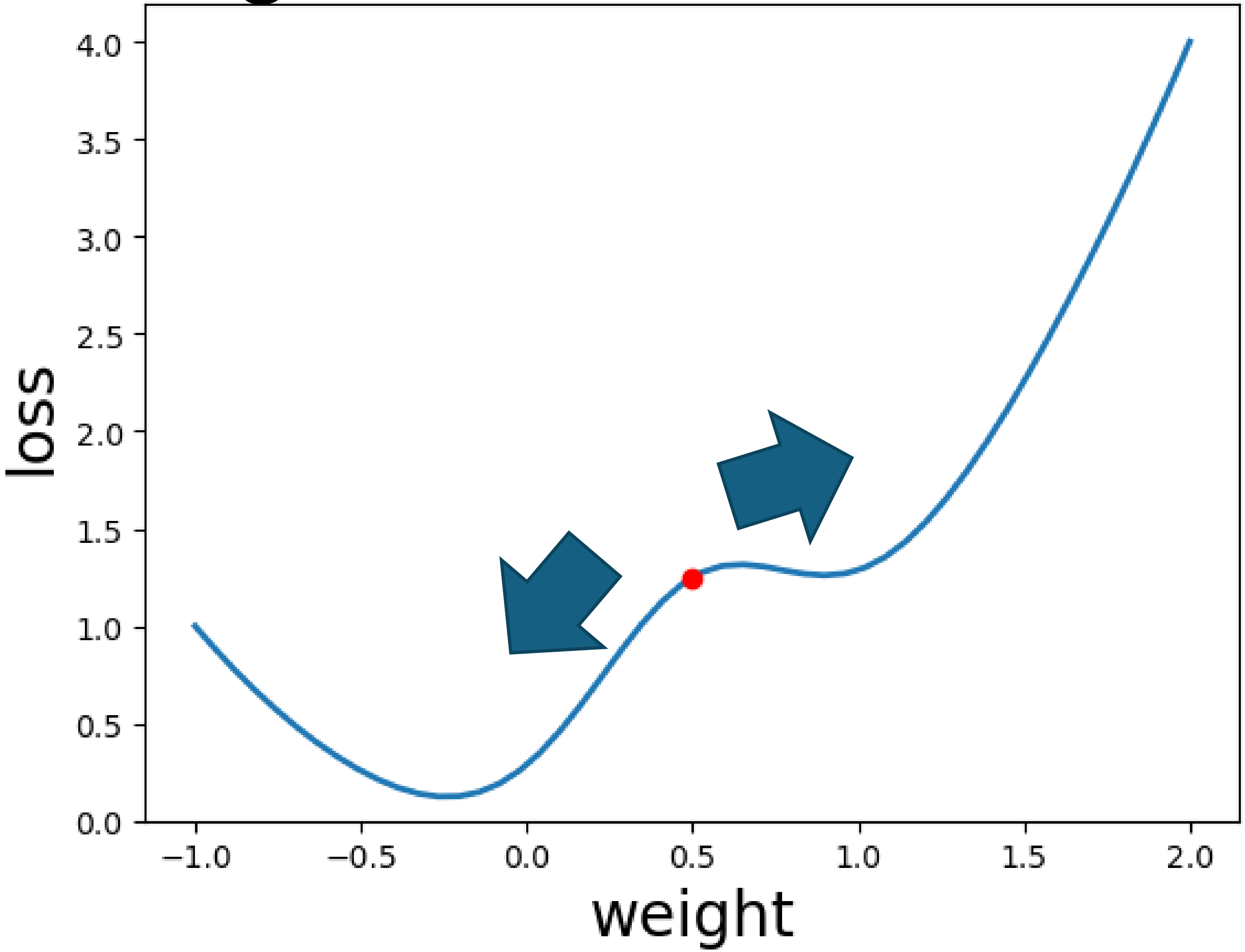
Find weights that minimize the loss



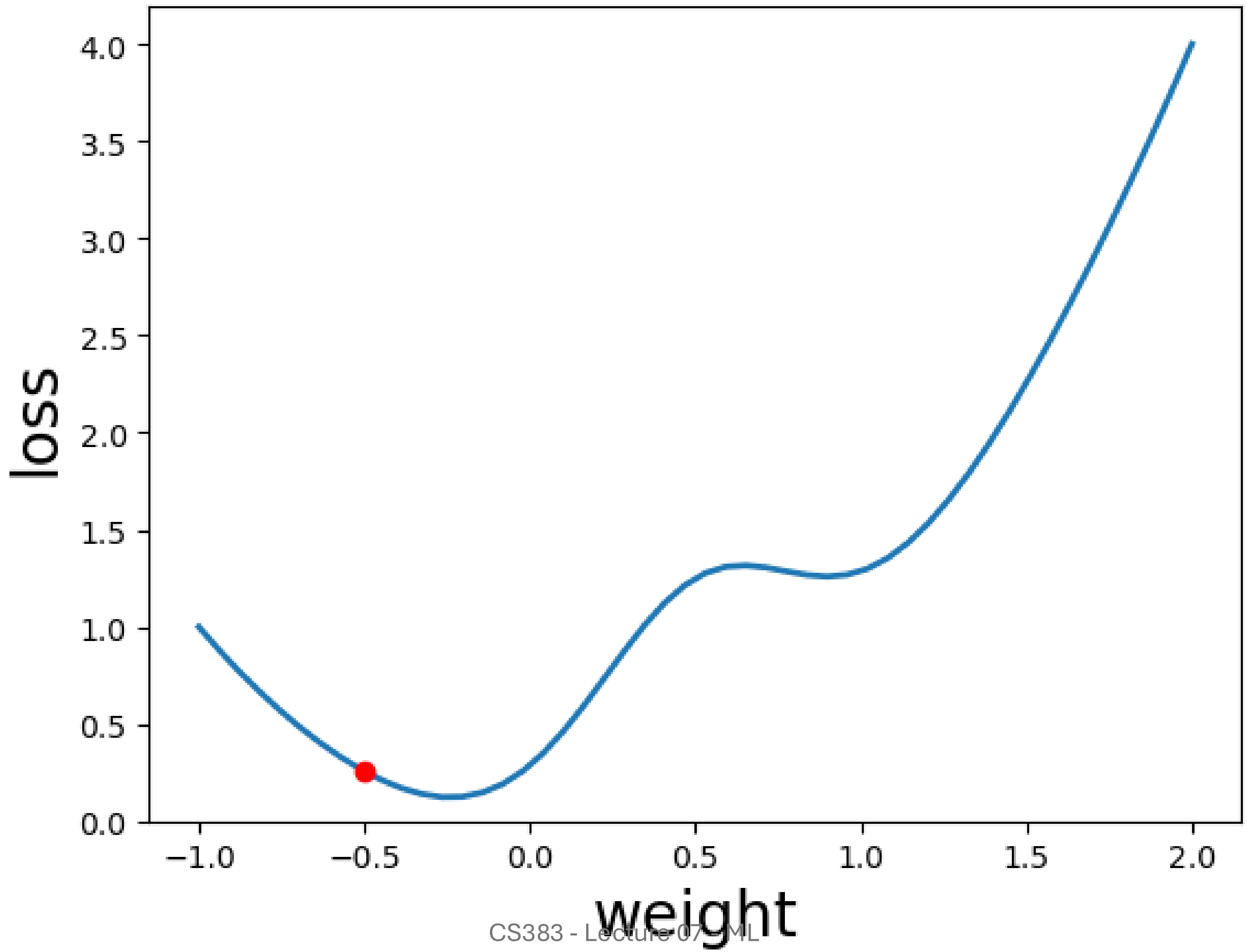
Find weights that minimize the loss



Find weights that minimize the loss



Find weights that minimize the loss



How to update the weights

1. Find the direction of the derivative of the loss function

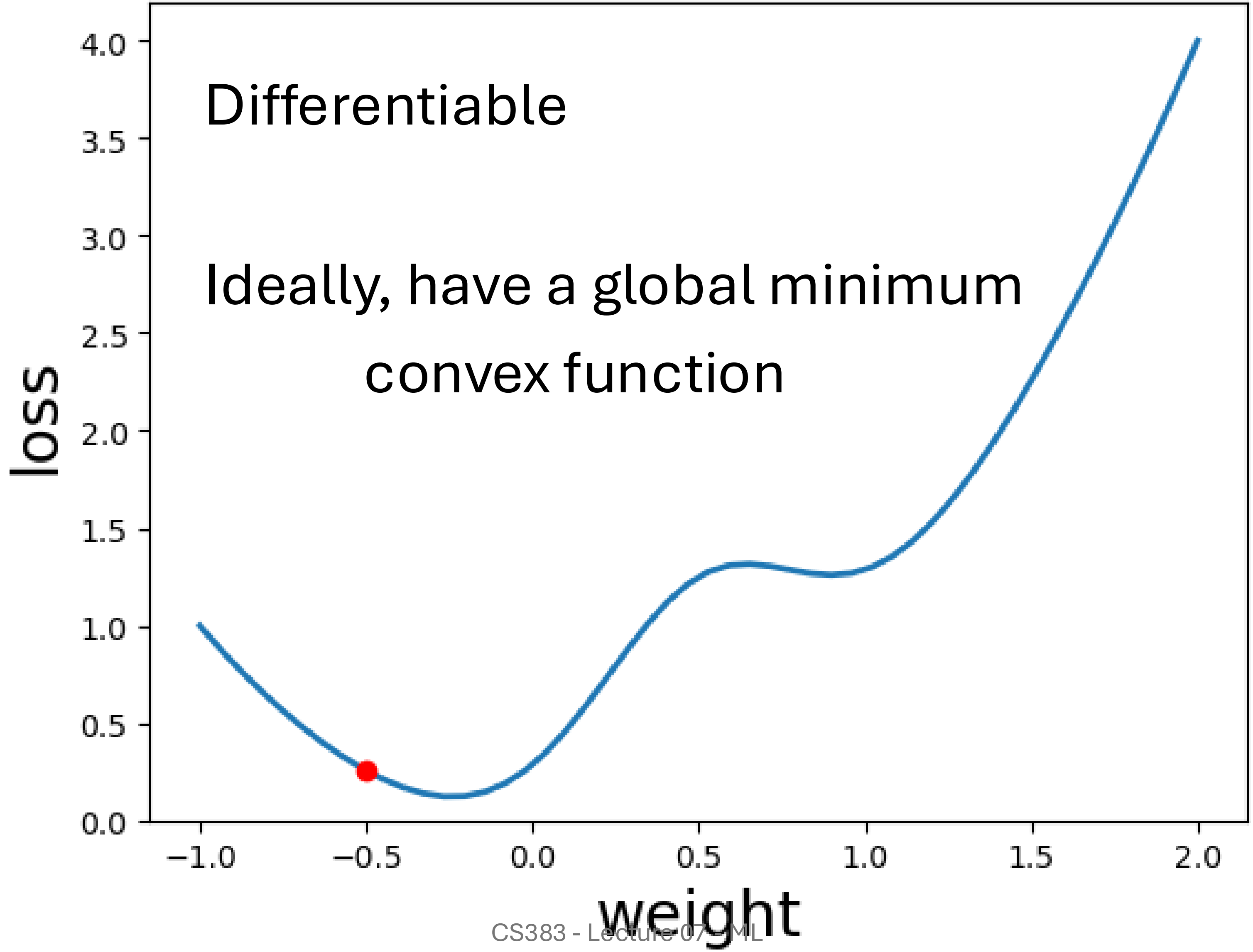
    aka gradient of the loss

2. Move the weight in that direction

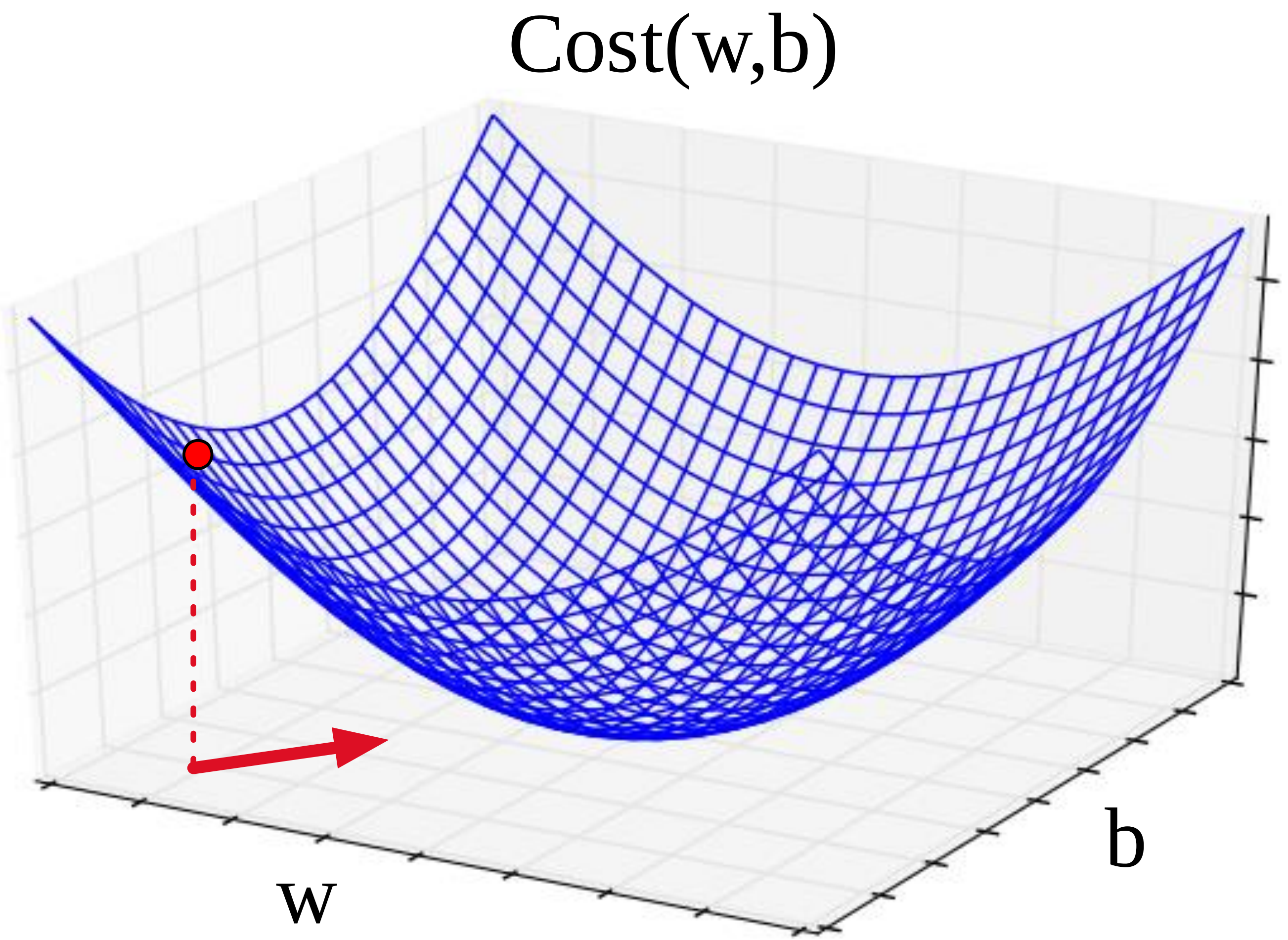
3. Then make a prediction on a new $x_{\{i\}}, y_{\{i\}}$ pair, and repeat 1 and 2

4. Repeat this for every example in our training set

Loss function properties



Moving to 2 weights



Computing the gradient of $\mathcal{L}$ partial derivatives

$$\nabla_{\vec{w}}\mathcal{L} = \begin{bmatrix} \frac{d\mathcal{L}}{dw_0} \\ \frac{d\mathcal{L}}{dw_1} \\ \cdots \\ \cdots \\ \frac{d\mathcal{L}}{dw_p} \end{bmatrix}$$

What can we do after we computed the gradients?

Updating weights based on gradients

$$\Delta_{\vec{w}}\mathcal{L} = \eta\nabla_{\vec{w}}\mathcal{L}\left(\vec{w}\right)$$

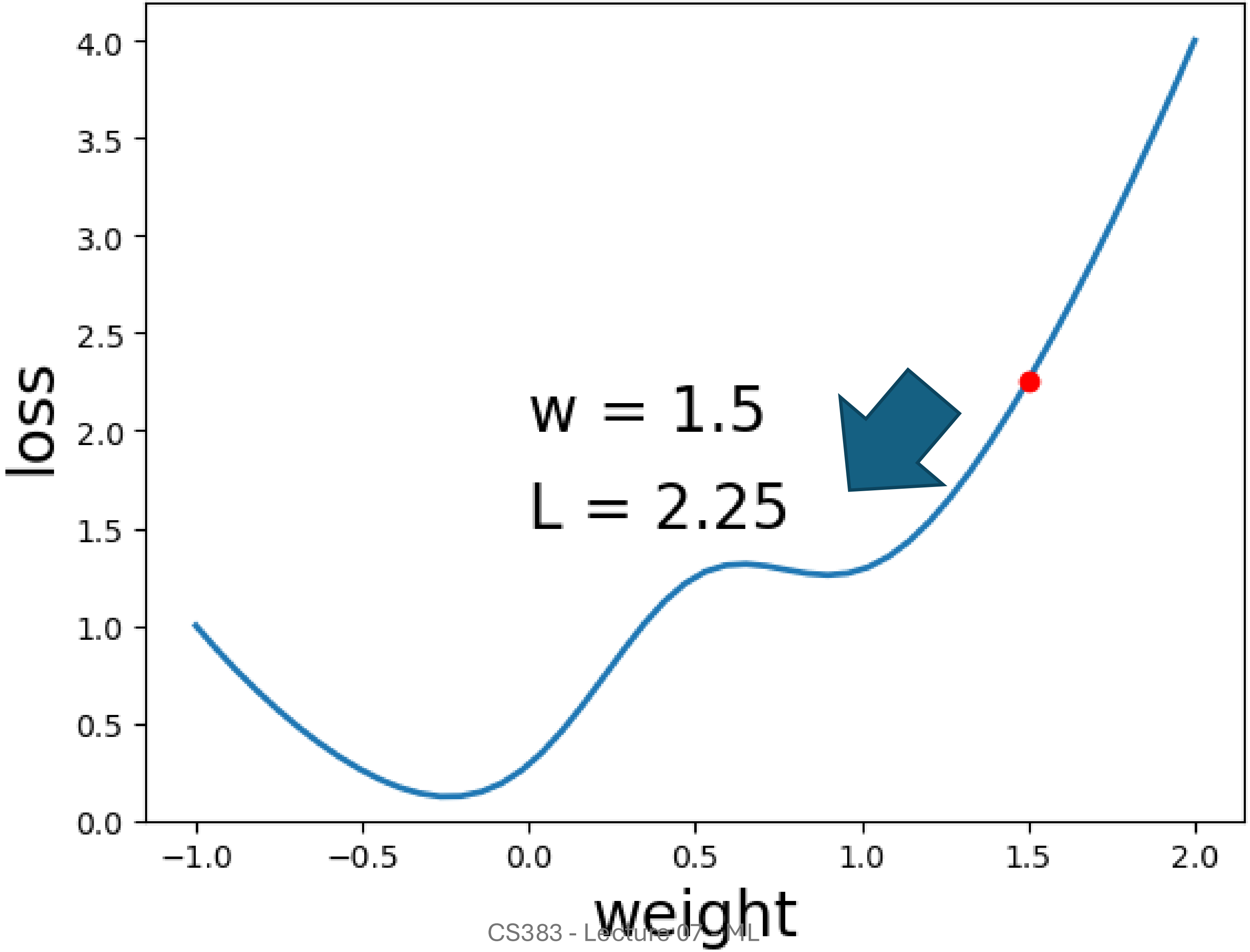
Update each individual weight:

$$w_i \leftarrow w_i - \eta\frac{d\mathcal{L}\left(\vec{w}\right)}{dw_i}$$

If we want to perform gradient ascent, we …

$$w_i \leftarrow w_i + \eta\frac{d\mathcal{L}\left(\vec{w}\right)}{dw_i}$$

Find weights that minimize the loss



Updating weights based on gradients

$$\Delta_{\vec{w}}\mathcal{L} = \eta \nabla_{\vec{w}}\mathcal{L}(\vec{w})$$

Update each individual weight:

$$w_i \leftarrow w_i - \eta \frac{d\mathcal{L}(\vec{w})}{dw_i}$$

If we want to perform gradient ascent, we ...

$$w_i \leftarrow w_i + \eta \frac{d\mathcal{L}(\vec{w})}{dw_i}$$

Step size

Gradient Descent

1. Randomly initialize $\underset{w}{\rightarrow}$

2. For every $\{x_i, y_i\}$ pair in our training set:

    Compute the gradient of the loss

    $$\frac{d\mathcal{L}\left(\underset{w}{\rightarrow}\right)}{dw_i}$$

    Update each weights based on the gradients

    $$w_i \leftarrow w_i - \eta \frac{d\mathcal{L}(\boldsymbol{w})}{dw_i}$$

3. Repeat 2 until convergance (or max epochs)
4. return β_i

Stochastic Gradient Descent

1. Randomly initialize $\underset{w}{\rightarrow}$

2. Randomly choose a $\{x_i, y_i\}$ pair in our training set without replacement until all pairs are used:

Compute the gradient of the loss

$$\frac{d\mathcal{L}\left(\underset{w}{\rightarrow}\right)}{dw_i}$$

Update each weights based on the gradients

$$w_i \leftarrow w_i - \eta \frac{d\mathcal{L}(\boldsymbol{w})}{dw_i}$$

3. Repeat 2 until convergence (or max epochs)
4. return β_i

Outline

Reading quiz #3

Simple linear regression

SGD (Stochastic Gradient Descent)

Normal equations solution – see handout posted on slack with derivation

Pros and Cons

Gradient Descent

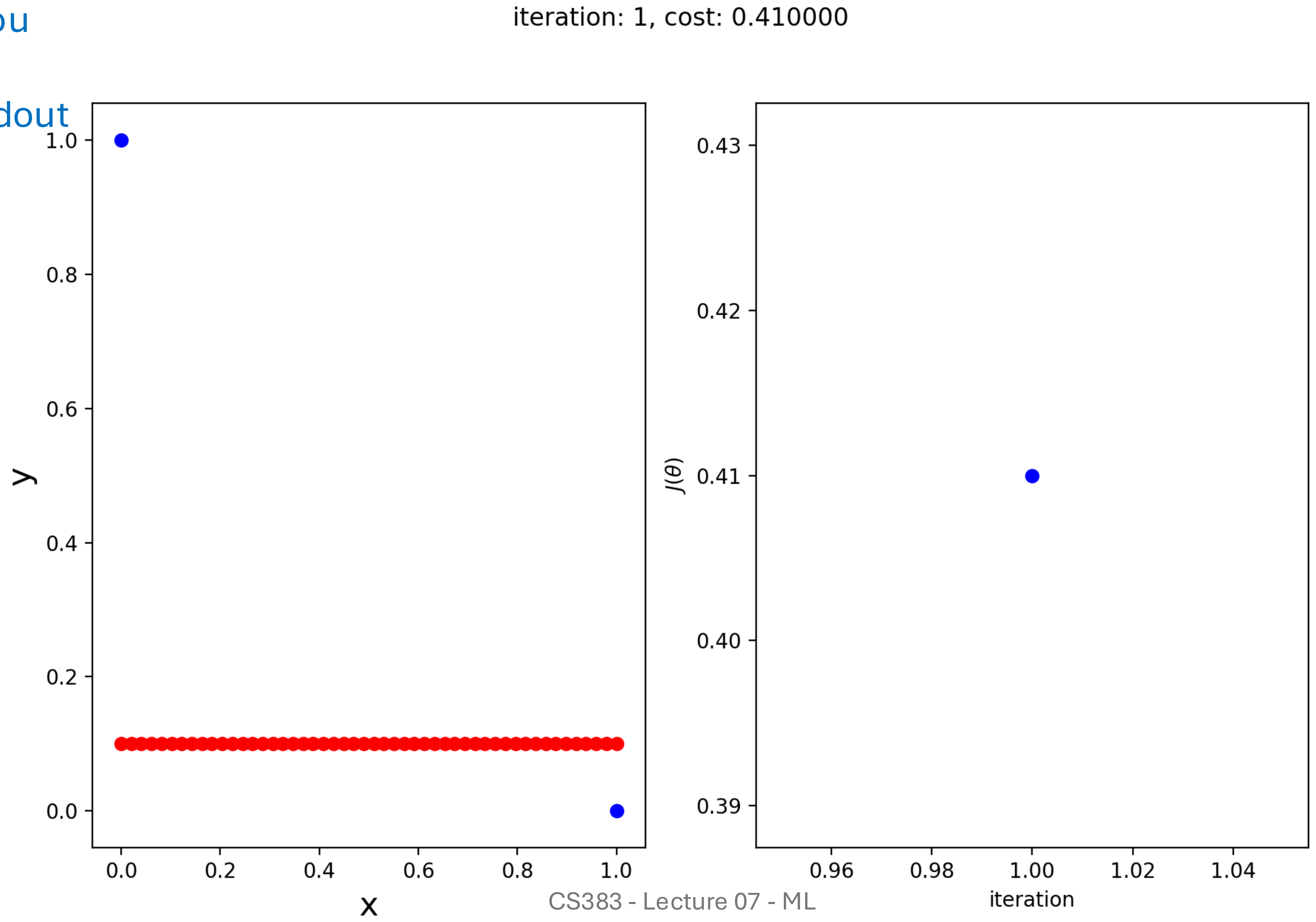
- Requires multiple iterations
- Need to choose η
- Works well when *n* is large
- Can support online learning

Normal Equation

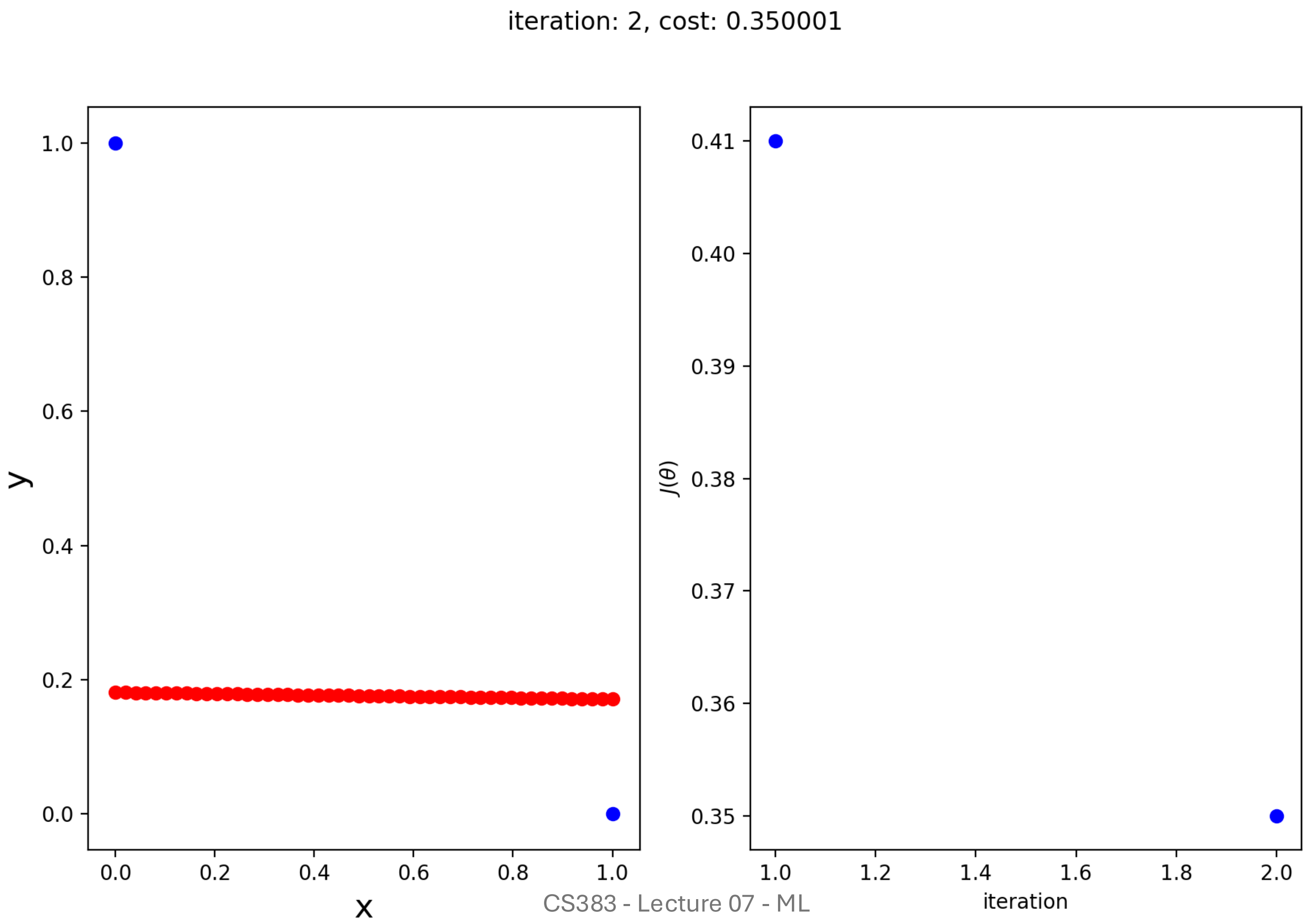
- Non-iterative
- No need to choose η
- Slow if *p* is large
  - Matrix inversion is $O(p^3)$

Toy example, iteration 1

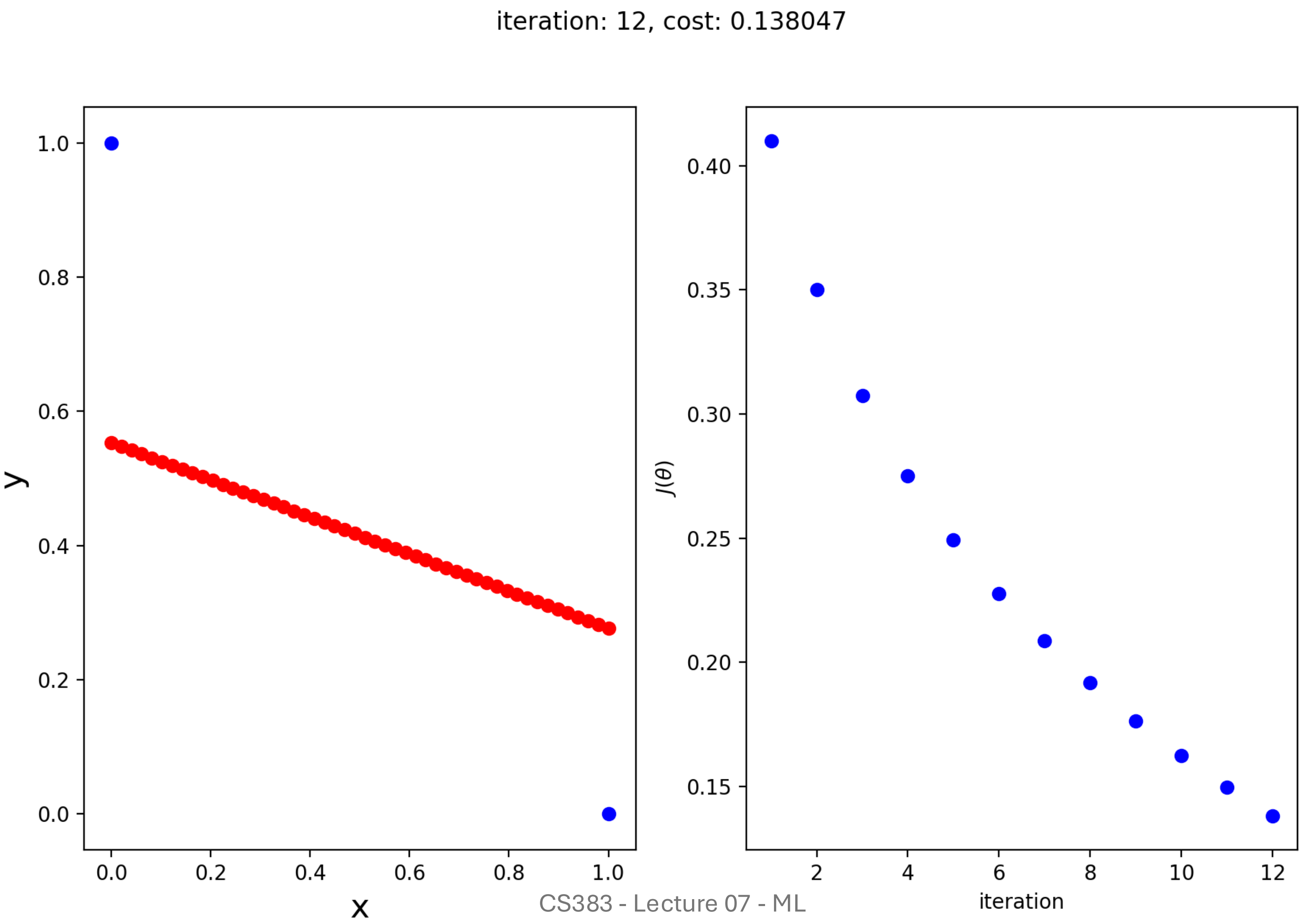
This is what you should have obtained in Handout 7!



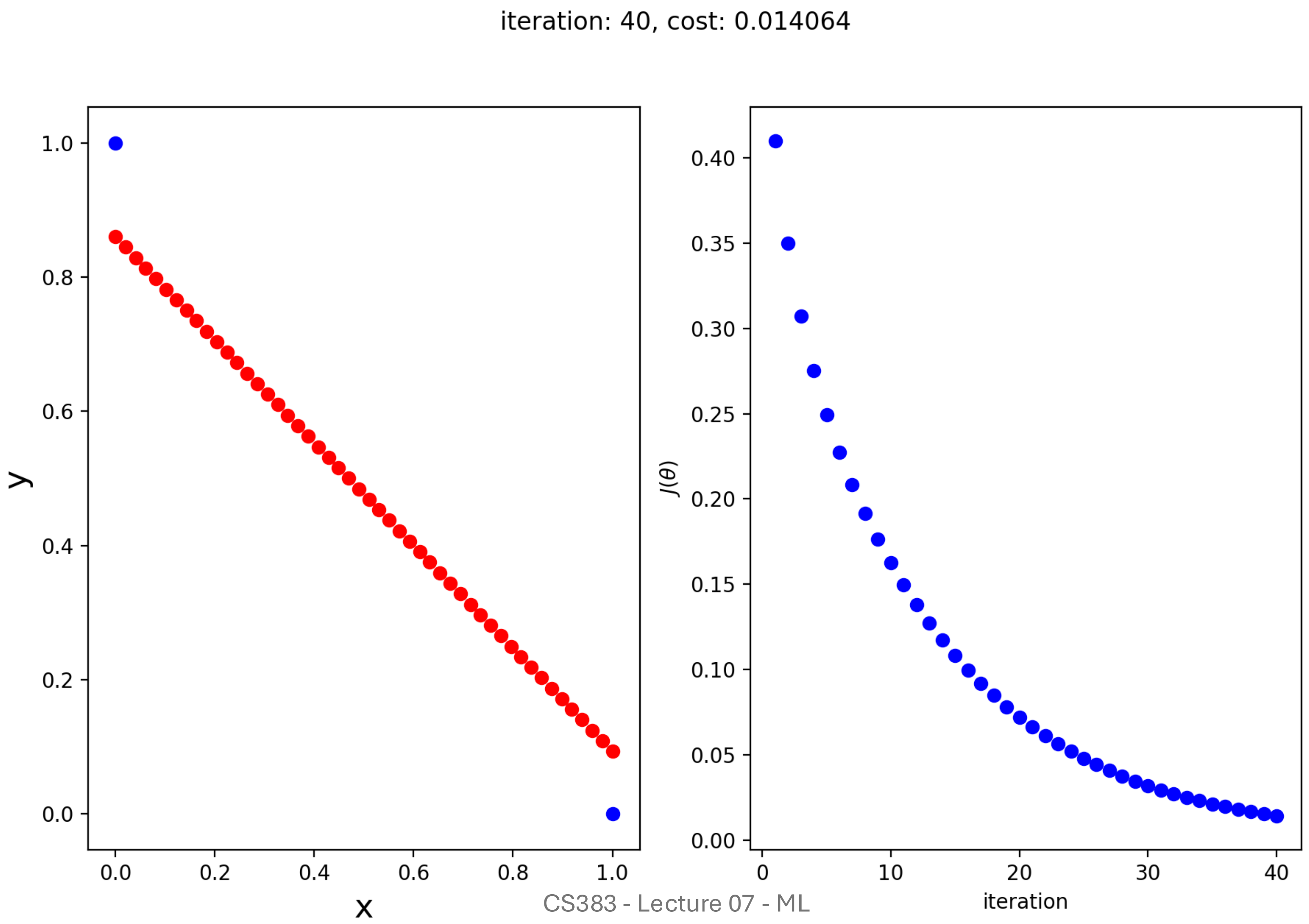
Toy example, iteration 2



Toy example, iteration 12



Toy example, iteration 40



Toy example, iteration 100