CS 260 – Foundations of Data Science

Lecture 5 – Multiple Linear Regression, Gradient Descent

09/10/2026

Adam Poliak

Announcements

HW2 due Tuesday night

Complete pre-course survey

Office Hours:

Tuesdays: 4pm-4:30pm

Thursdays 2-3:30pm, Park 200D

Outline

Introduction to applied linear algebra

Multiple linear regression

Analytic solution to multiple linear regression

Matrix Multiplication

$$A = \begin{bmatrix} — & \mathbf{a}_1 & — \\ — & \mathbf{a}_2 & — \\ — & \mathbf{a}_3 & — \end{bmatrix}, \qquad B = \begin{bmatrix} | & | \\ \mathbf{b}_1 & \mathbf{b}_2 \\ | & | \end{bmatrix}$$

$$AB = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

Matrix Multiplication

$$A = \begin{bmatrix} — & \mathbf{a}_1 & — \\ — & \mathbf{a}_2 & — \\ — & \mathbf{a}_3 & — \end{bmatrix}, \qquad B = \begin{bmatrix} | & | \\ \mathbf{b}_1 & \mathbf{b}_2 \\ | & | \end{bmatrix}$$

$$AB = \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{b}_1 & \mathbf{a}_1 \cdot \mathbf{b}_2 \\ \mathbf{a}_2 \cdot \mathbf{b}_1 & \mathbf{a}_2 \cdot \mathbf{b}_2 \\ \mathbf{a}_3 \cdot \mathbf{b}_1 & \mathbf{a}_3 \cdot \mathbf{b}_2 \end{bmatrix}$$

Matrix Multiplication

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$AB = \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{b}_1 & \mathbf{a}_1 \cdot \mathbf{b}_2 \\ \mathbf{a}_2 \cdot \mathbf{b}_1 & \mathbf{a}_2 \cdot \mathbf{b}_2 \\ \mathbf{a}_3 \cdot \mathbf{b}_1 & \mathbf{a}_3 \cdot \mathbf{b}_2 \end{bmatrix}$$

Matrix Multiplication: Column View

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} | & | & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ | & | & | \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$AB = \begin{bmatrix} | & | \\ 1\mathbf{a}_1 + 0\mathbf{a}_2 + 1\mathbf{a}_3 & 0\mathbf{a}_1 + 1\mathbf{a}_2 + 1\mathbf{a}_3 \\ | & | \end{bmatrix}$$

Matrix Multiplication: Column View

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} | & | & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ | & | & | \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$AB = \begin{bmatrix} | & | \\ 1\mathbf{a}_1 + 0\mathbf{a}_2 + 1\mathbf{a}_3 & 0\mathbf{a}_1 + 1\mathbf{a}_2 + 1\mathbf{a}_3 \\ | & | \end{bmatrix} = \boxed{\begin{bmatrix} 4 & 5 \\ 4 & 5 \\ 1 & 1 \end{bmatrix}}$$

Matrix Multiplication: Row view

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} — & \mathbf{b}_1 & — \\ — & \mathbf{b}_2 & — \\ — & \mathbf{b}_3 & — \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$AB = \begin{bmatrix} — & 1\mathbf{b}_1 + 2\mathbf{b}_2 + 3\mathbf{b}_3 & — \\ — & 0\mathbf{b}_1 + 1\mathbf{b}_2 + 4\mathbf{b}_3 & — \\ — & 0\mathbf{b}_1 + 0\mathbf{b}_2 + 1\mathbf{b}_3 & — \end{bmatrix}$$

Matrix Multiplication: Row view

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} — & \mathbf{b}_1 & — \\ — & \mathbf{b}_2 & — \\ — & \mathbf{b}_3 & — \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$AB = \begin{bmatrix} — & 1\mathbf{b}_1 + 2\mathbf{b}_2 + 3\mathbf{b}_3 & — \\ — & 0\mathbf{b}_1 + 1\mathbf{b}_2 + 4\mathbf{b}_3 & — \\ — & 0\mathbf{b}_1 + 0\mathbf{b}_2 + 1\mathbf{b}_3 & — \end{bmatrix} = \begin{bmatrix} 4 & 5 \\ 4 & 5 \\ 1 & 1 \end{bmatrix}$$

Matrix Multiplication & Inverse

$$\mathbf{Ax} = \mathbf{y}$$

Matrix **A** is function that maps/transforms x into y

$$f(x) = y$$

Inverse function: a function that reverses what the original function does

$$f^{-1}(y) = x$$

Matrix Inverse

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad A^{-1} = \begin{bmatrix} p & q \\ r & s \end{bmatrix}$$

$$\boldsymbol{A A^{-1} = I}$$

Matrix Inverse

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$$

$$\boldsymbol{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \boldsymbol{A}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Matrix Transpose

$$\boldsymbol{A} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \qquad \boldsymbol{A}^T = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$$

Useful note: $(\boldsymbol{AB})^T = \boldsymbol{B}^T \boldsymbol{A}^T$

Outline

Introduction to applied linear algebra

***Multiple* linear regression**

Analytic solution to multiple linear regression

HW 3: USA Housing data

Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
79545.45857	5.682861322	7.009188143	4.09	23086.8005	1059033.558
79248.64245	6.002899808	6.730821019	3.09	40173.07217	1505890.915
61287.06718	5.86588984	8.51272743	5.13	36882.1594	1058987.988
63345.24005	7.188236095	5.586728665	3.26	34310.24283	1260616.807
59982.19723	5.040554523	7.839387785	4.23	26354.10947	630943.4893
80175.75416	4.988407758	6.104512439	4.04	26748.42842	1068138.074
64698.46343	6.025335907	8.147759585	3.41	60828.24909	1502055.817
78394.33928	6.989779748	6.620477995	2.42	36516.35897	1573936.564
59927.66081	5.36212557	6.393120981	2.3	29387.396	798869.5328
81885.92718	4.42367179	8.167688003	6.1	40149.96575	1545154.813
80527.47208	8.093512681	5.0427468	4.1	47224.35984	1707045.722
50593.6955	4.496512793	7.467627404	4.49	34343.99189	663732.3969
39033.80924	7.671755373	7.250029317	3.1	39220.36147	1042814.098
73163.66344	6.919534825	5.993187901	2.27	32326.12314	1291331.518
69391.38018	5.344776177	8.406417715	4.37	35521.29403	1402818.21
73091.86675	5.443156467	8.517512711	4.01	23929.52405	1306674.66
79706.96306	5.067889591	8.219771123	3.12	39717.81358	1556786.6

Multiple Linear Regression

$$\hat{y} = h_{\vec{w}}(\vec{x})$$
$$= w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_p x_p = \vec{w} \cdot \vec{x}$$

"fake" ones

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \cdots & x_{1p} \\ 1 & & & & & \\ \vdots & & & & & \\ 1 & & & & & \end{bmatrix}$$

$n \times (p+1)$

- Goal: minimize cost function

$$J(\vec{w}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \vec{w} \cdot \vec{x_i})^2$$

Multiple Linear Regression

$$\hat{y} = h_{\vec{w}}(X) = \vec{w}X$$

- Goal: minimize cost function

$$J(\vec{w}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = (\vec{y} - \vec{w}X)^2$$

Outline

Introduction to applied linear algebra

Multiple linear regression

Analytic solution to multiple linear regression

Gradient Descent

Analytic solution to multiple linear regression

SSE

Take the derivative

Set to 0

Divide by -2

Distribute

Rearrange

Multiply by inverse

Analytic solution to multiple linear regression

SSE

Take the derivative

Set to 0

Divide by -2

Distribute

Rearrange

Multiply by inverse

$$\hat{\mathbf{w}} = (X^T X)^{-1} X^T \mathbf{y}.$$

Variance of X

Covariance of X and $\vec{y}$

Analytic solut regression

SSE

Take the derivative

Set to 0

Divide by -2

Distribute

Rearrange

Multiply by inverse

$$L(\mathbf{w}) = \|\mathbf{y} - X\mathbf{w}\|^2.$$

Take the derivative:

$$\nabla_{\mathbf{w}} L = -2X^T(\mathbf{y} - X\mathbf{w}).$$

Set it equal to zero:

$$-2X^T(\mathbf{y} - X\mathbf{w}) = 0.$$

Divide by -2:

$$X^T(\mathbf{y} - X\mathbf{w}) = 0.$$

Distribute:

$$X^T\mathbf{y} - X^TX\mathbf{w} = 0.$$

Rearrange:

$$X^TX\mathbf{w} = X^T\mathbf{y}.$$

Multiply by $(X^TX)^{-1}$:

$$\boxed{\hat{\mathbf{w}} = (X^TX)^{-1}X^T\mathbf{y}}.$$

Analytic solution to multiple linear regression

$$J(\vec{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \vec{w}\cdot\vec{x_i})^2 = \frac{1}{2}(\vec{y} - X\vec{w})\cdot(\vec{y} - X\vec{w})$$

$$= \frac{1}{2}(\vec{y}\cdot\vec{y} - 2\vec{y}\cdot X\vec{w} + X\vec{w}\cdot X\vec{w})$$

- Take derivative and set to $\vec{0}$

$$\frac{\partial J}{\partial \vec{w}} = -X^T\vec{y} + (X^TX)\vec{w} = \vec{0}$$

$$\leftrightarrow (X^TX)\vec{w} = X^T\vec{y}$$

$$\leftrightarrow (X^TX)^{-1}(X^TX)\vec{w} = (X^TX)^{-1}X^T\vec{y}$$

$$\leftrightarrow \vec{w} = \underbrace{(X^TX)^{-1}}_{\text{Variance of X}}\underbrace{X^T\vec{y}}_{\text{Covariance of X and } \vec{y}}$$

(Keep this formula and its interpretation in mind!)

Handout

Handout 5

2. $X = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad \vec{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

"fake" ones

3. $\vec{w} = (X^T X)^{-1} X^T \vec{y}$

$$= \left(\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{2-1} \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Same as before!

Normal Equation - RunTime

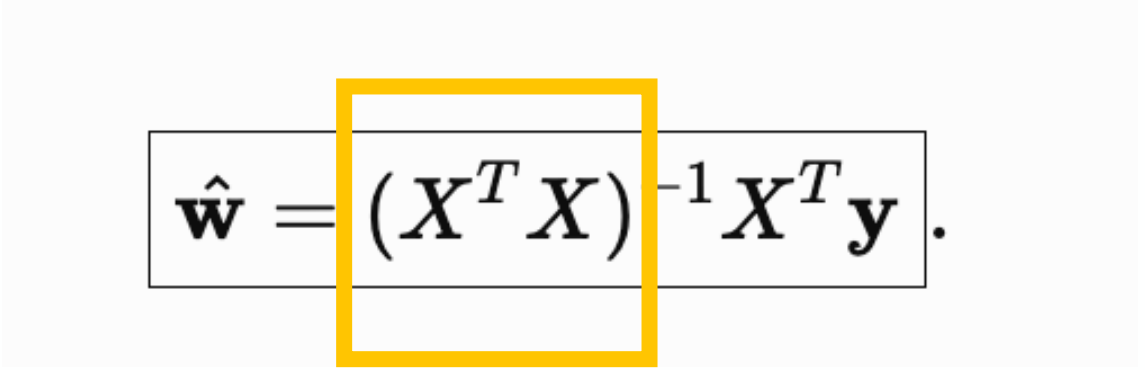
```
// Input: Matrix X of size n x p (n rows, p columns)
// Output: Matrix C of size p x p representing X^T * X

    n = number of rows in X
    p = number of columns in X

    Initialize matrix C of size p x p with all zeros

    For i from 1 to p:
        For j from 1 to p:
            For k from 1 to n:
                C[i][j] = C[i][j] + (X[k][i] * X[k][j])

    Return C
```



Outline

Introduction to applied linear algebra

Multiple linear regression

Analytic solution to multiple linear regression

Gradient Descent

Process Learning Weights

1. Randomly initialize weights

2. Make predictions $\hat{y}$

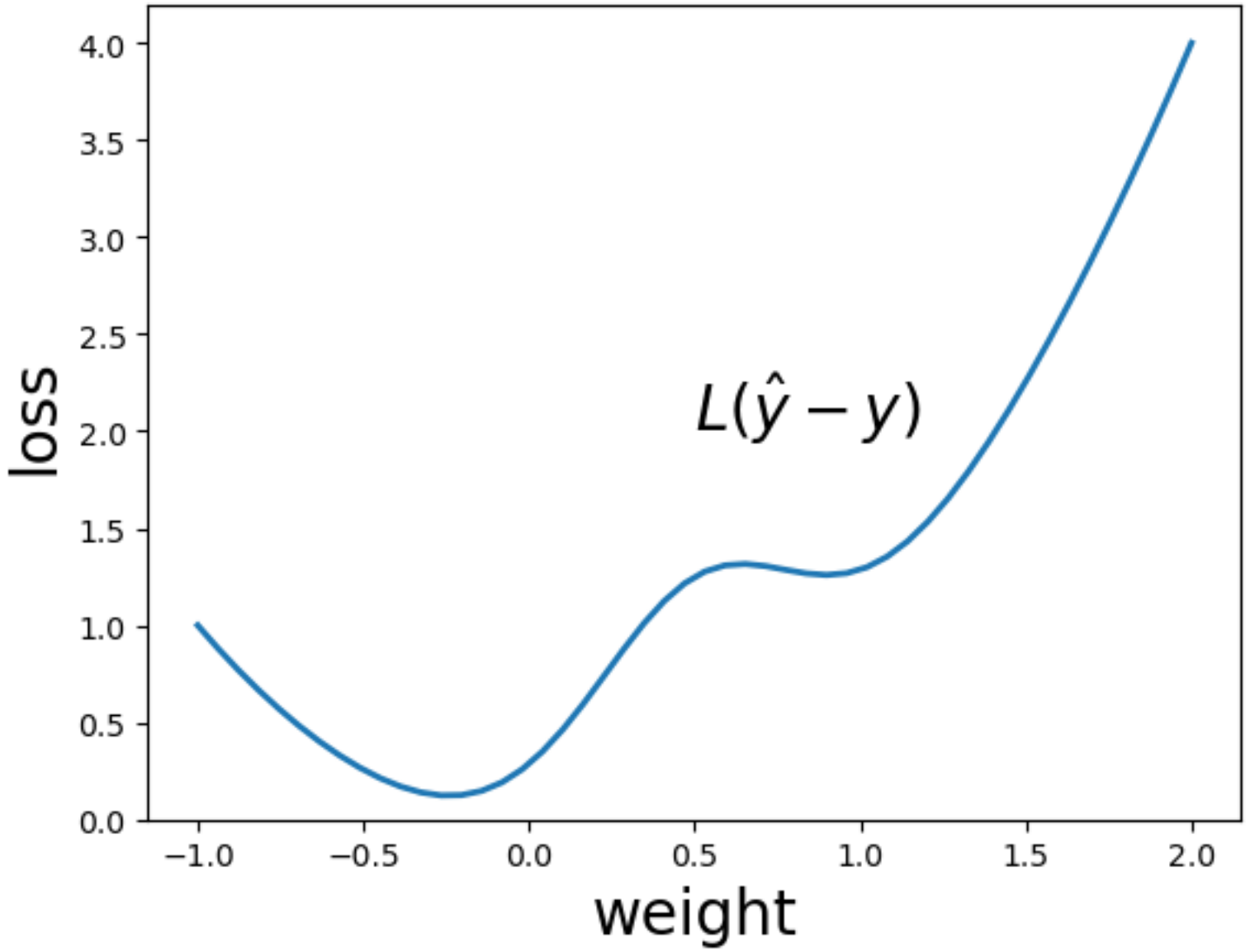
3. Compute loss function - quantify how close $\hat{y}$ *and* y are

4. Update weights accordingly based on the loss function

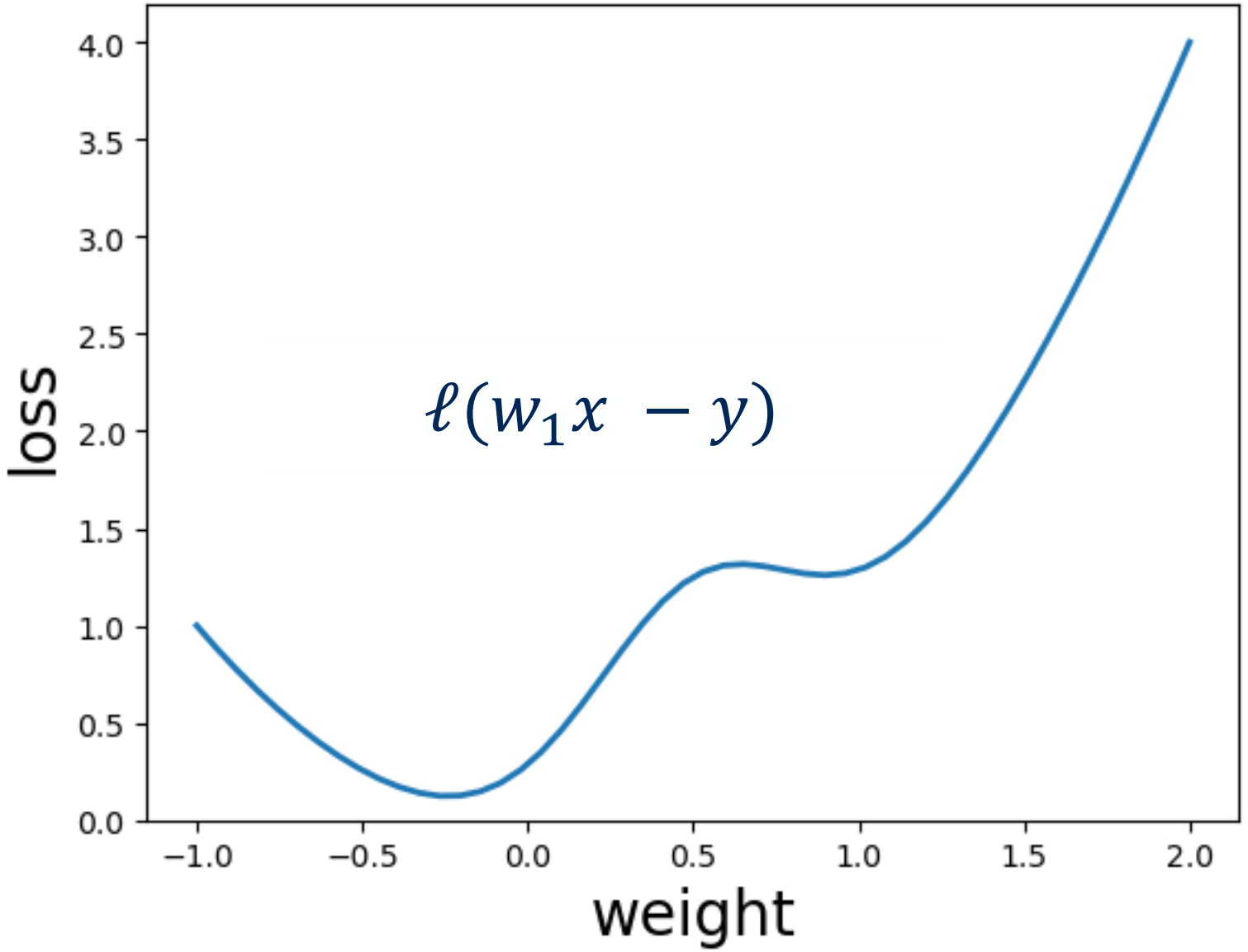
aka Optimization

5. Repeat 2-4

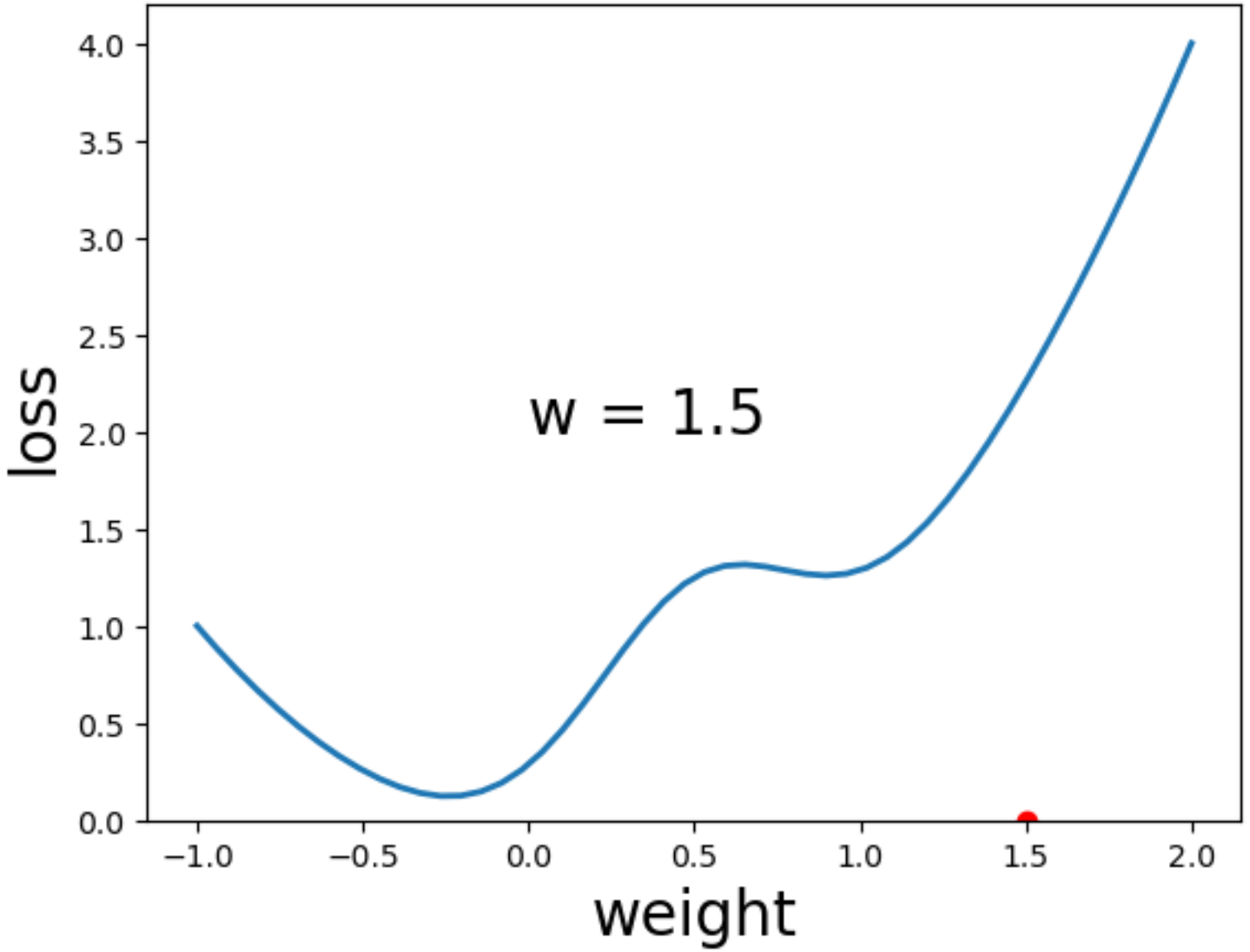
Find weights that minimize the loss



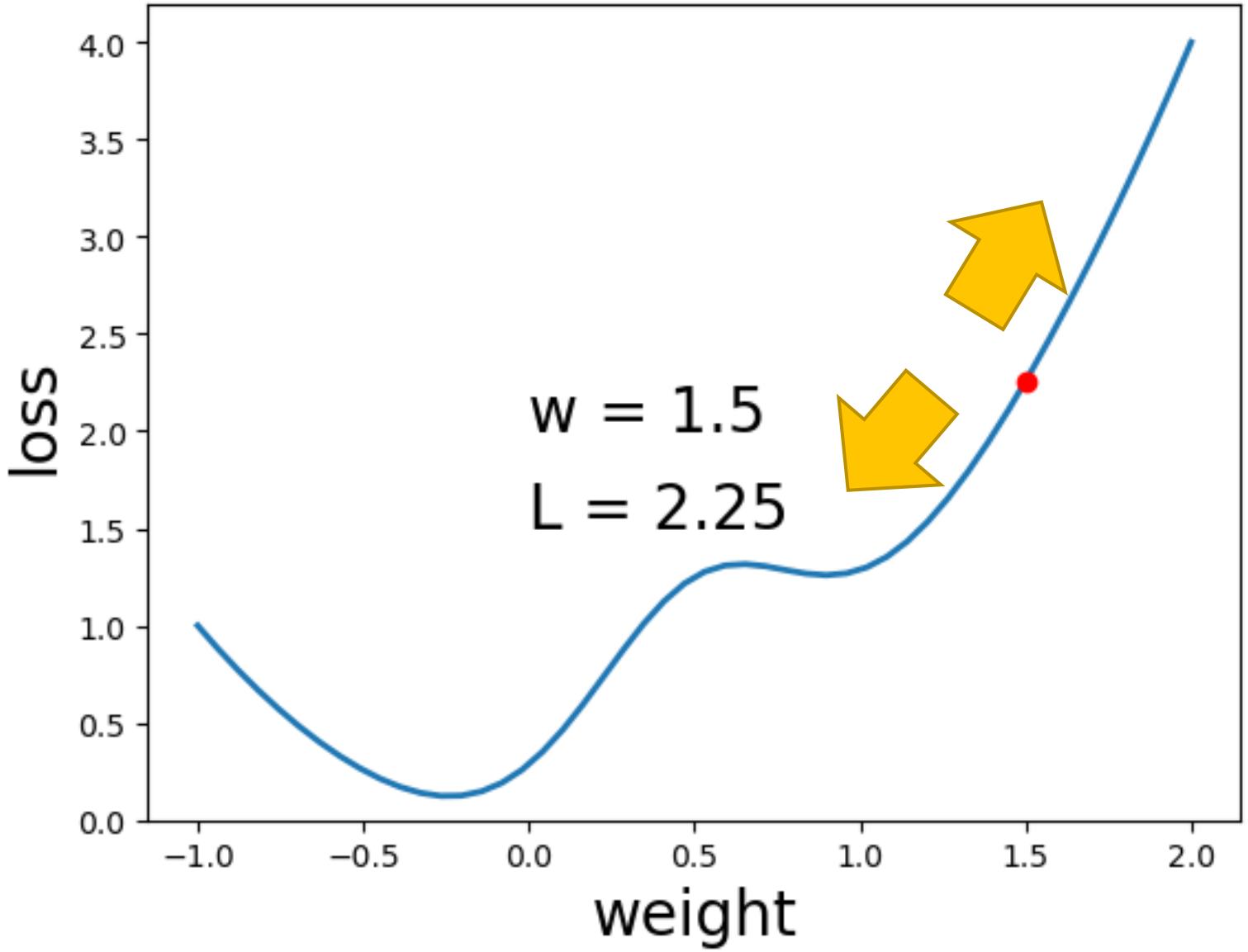
Find weights that minimize the loss



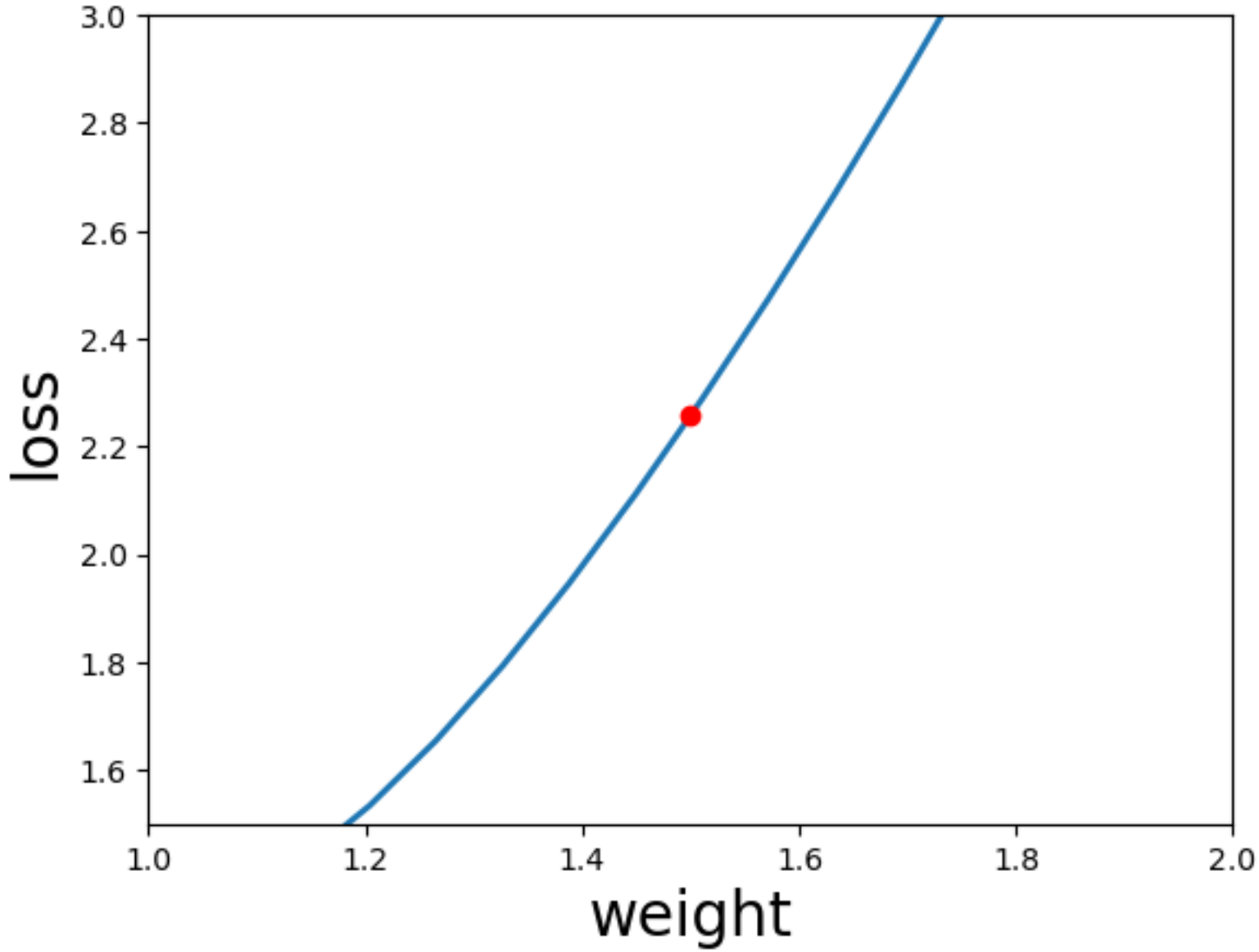
Find weights that minimize the loss



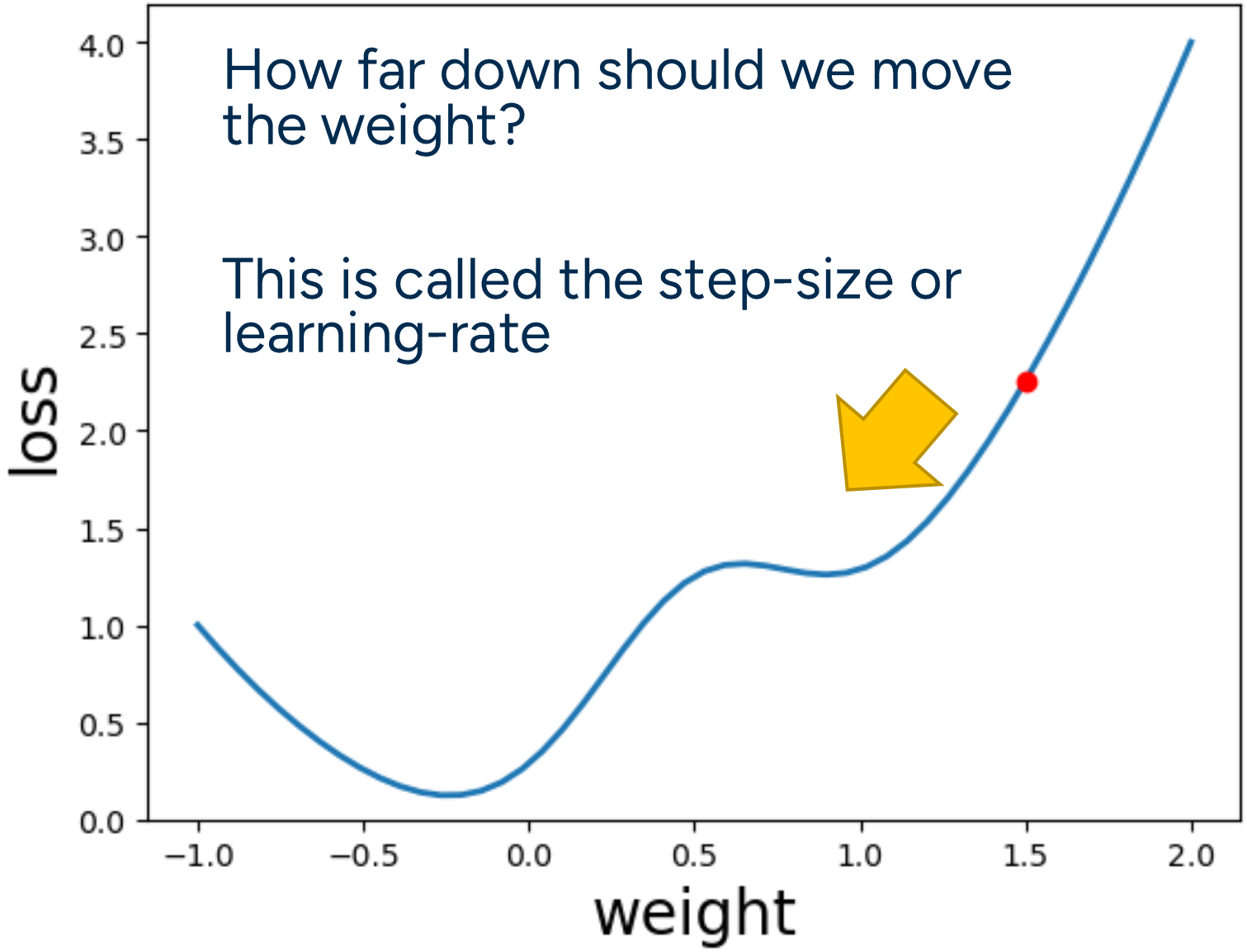
Find weights that minimize the loss



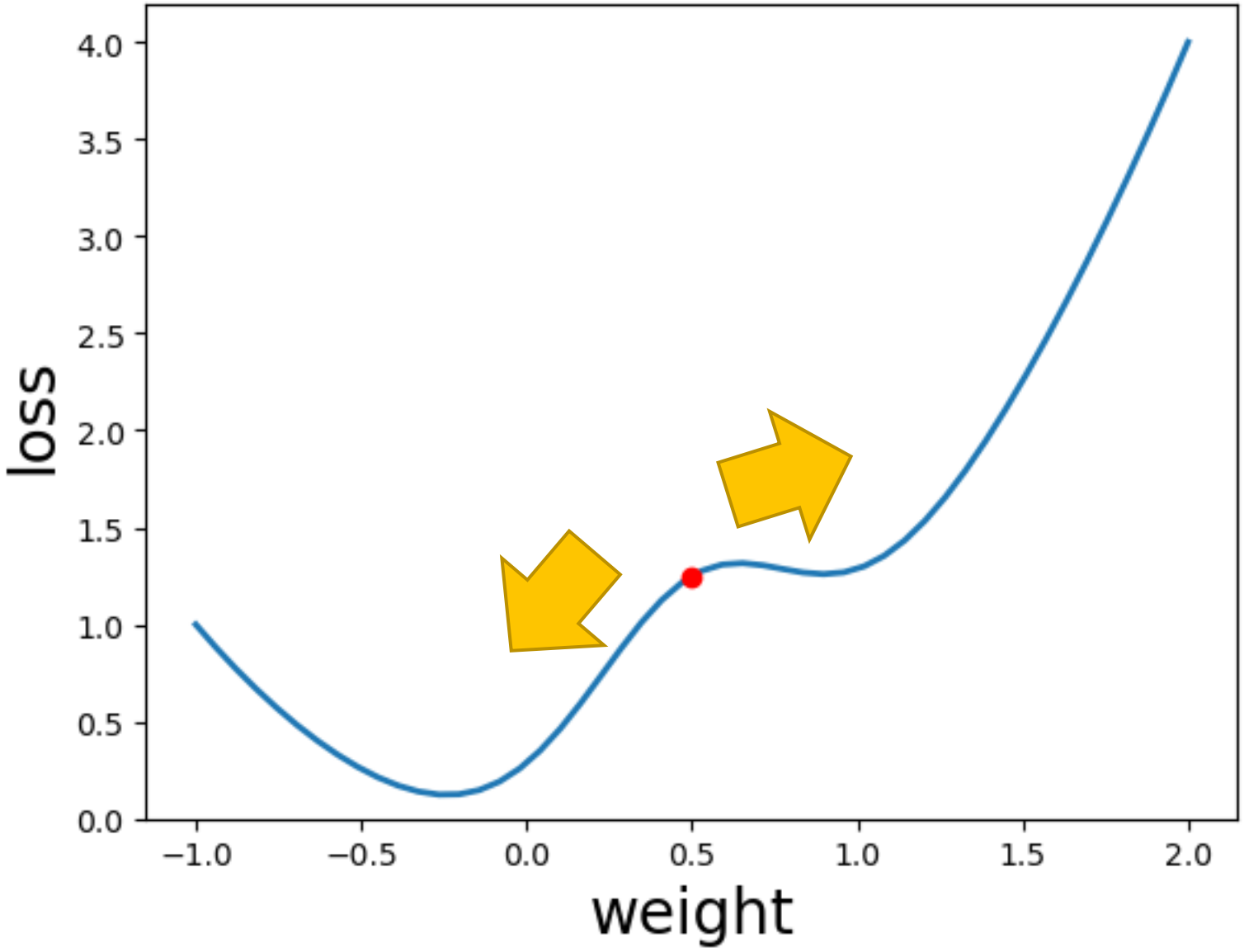
Find weights that minimize the loss



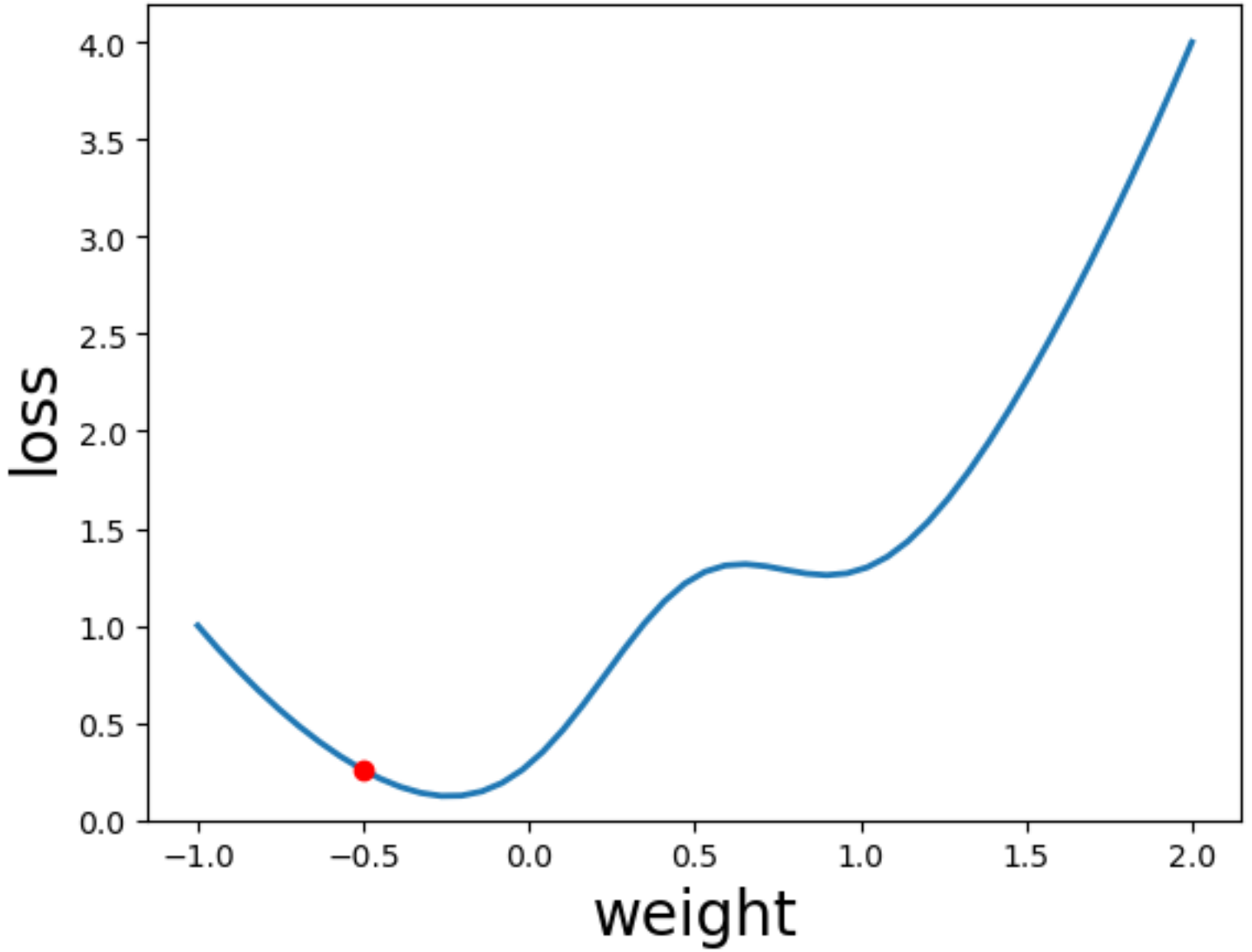
Find weights that minimize the loss



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

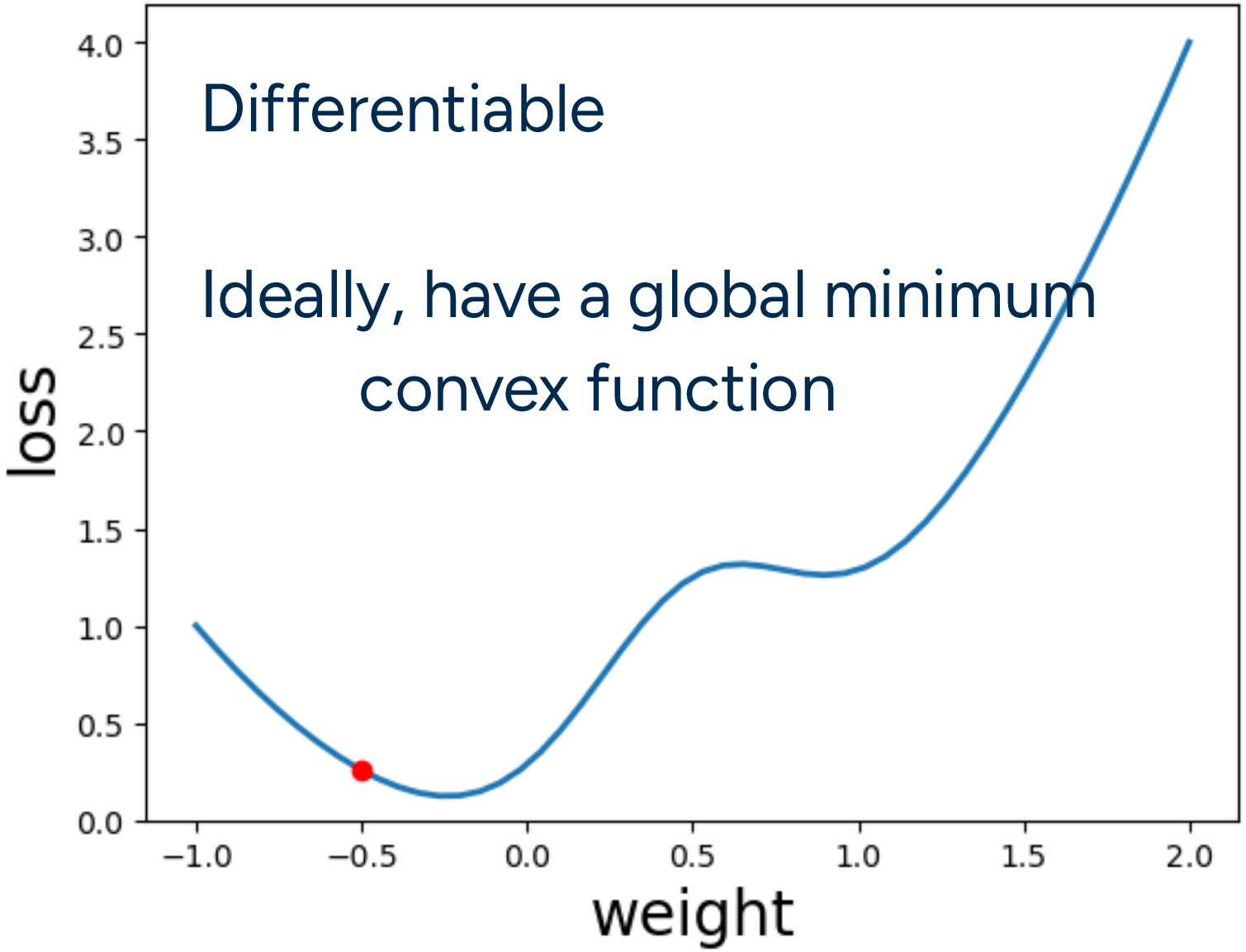
   aka gradient of the loss

2. Move the weight in that direction

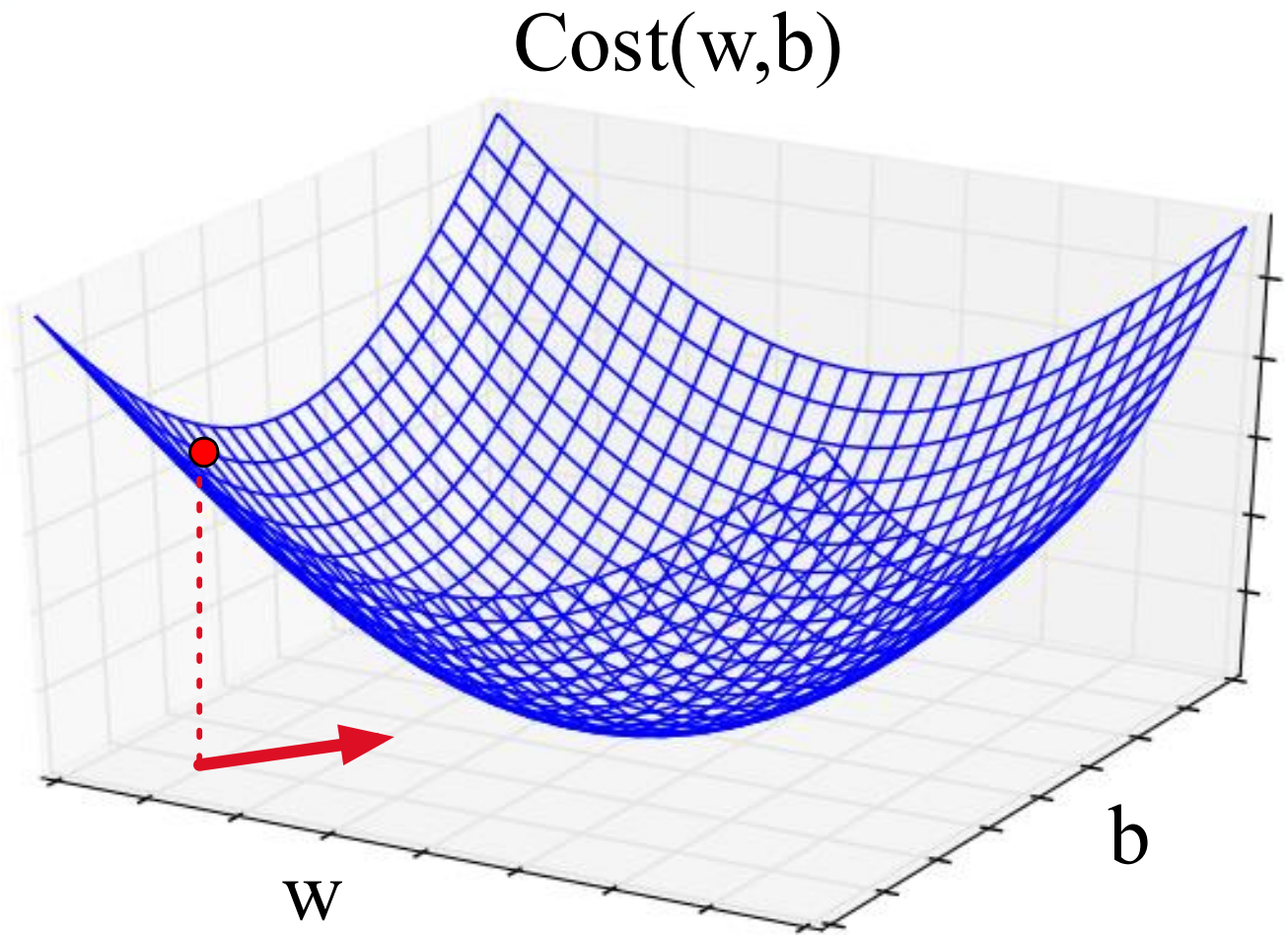
3. Then make a prediction on a new $x_{\{i\}}, y_{\{i\}}$ pair, and repeat 1 and 2

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

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

What can we do after we computed the gradients?

Updating weights based on gradients

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

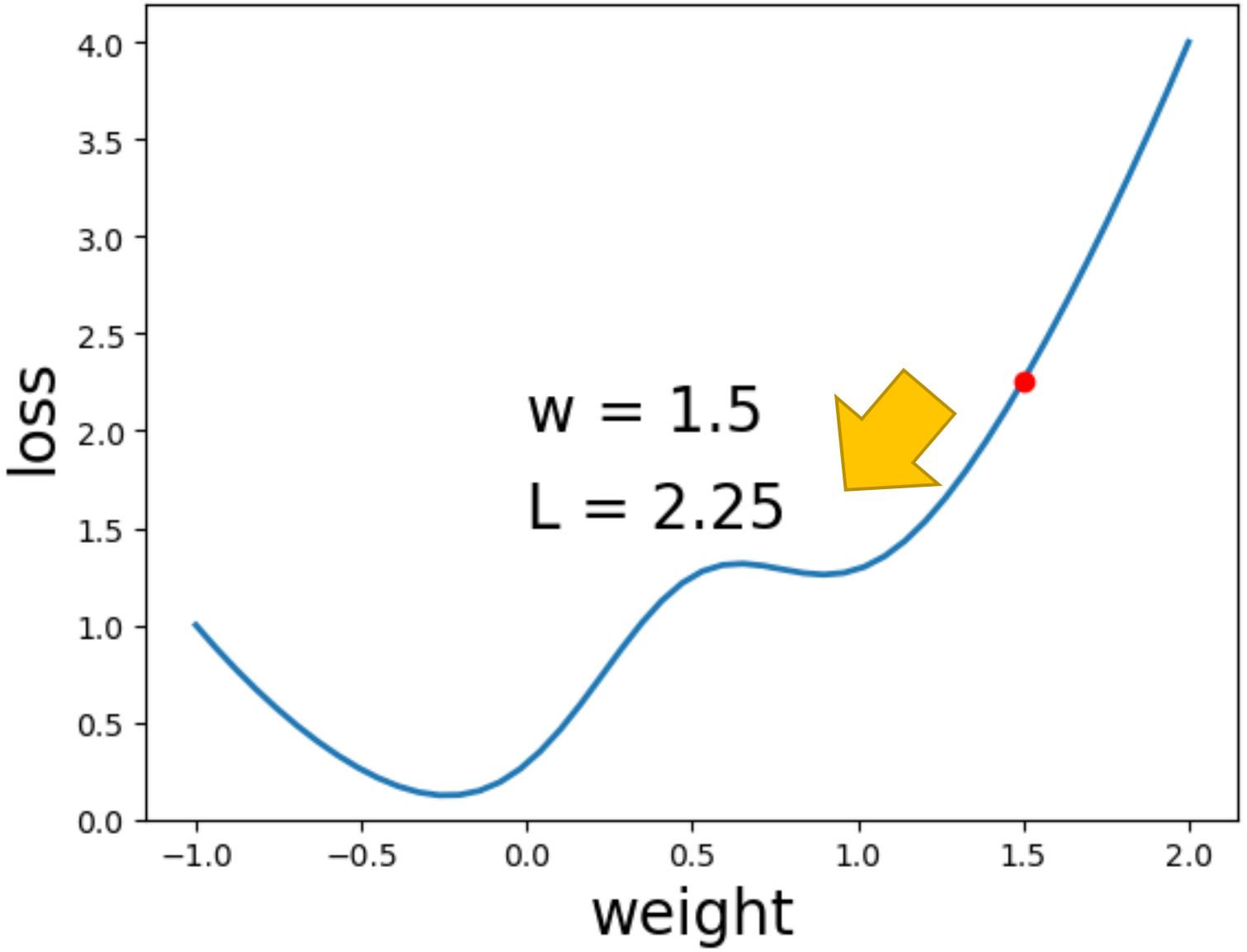
Update each individual weight:

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

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

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

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 …

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

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

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

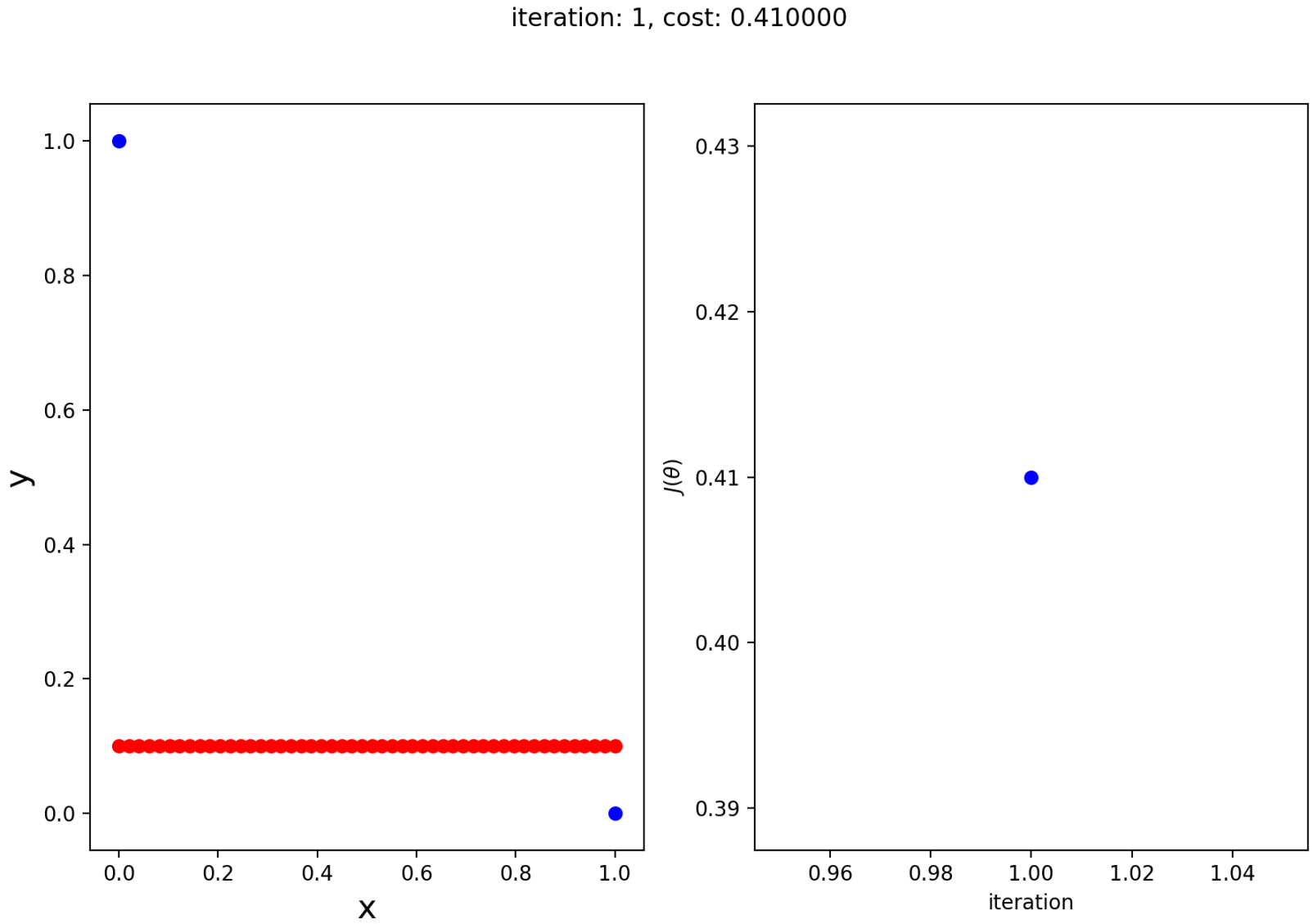
No need to choose η

Slow if p is large

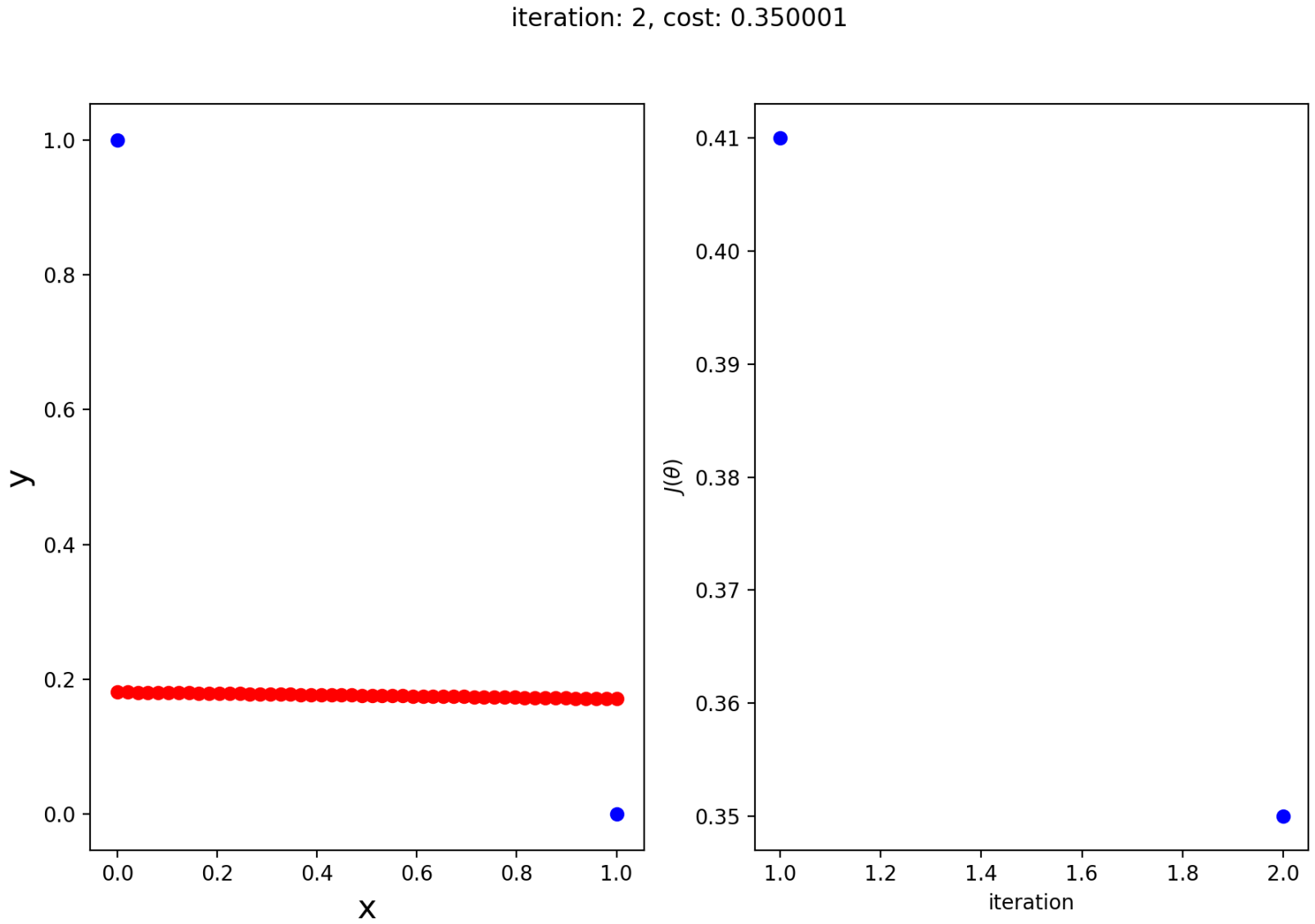
Matrix inversion is $O(p^3)$

SGD with our small dataset from the handouts

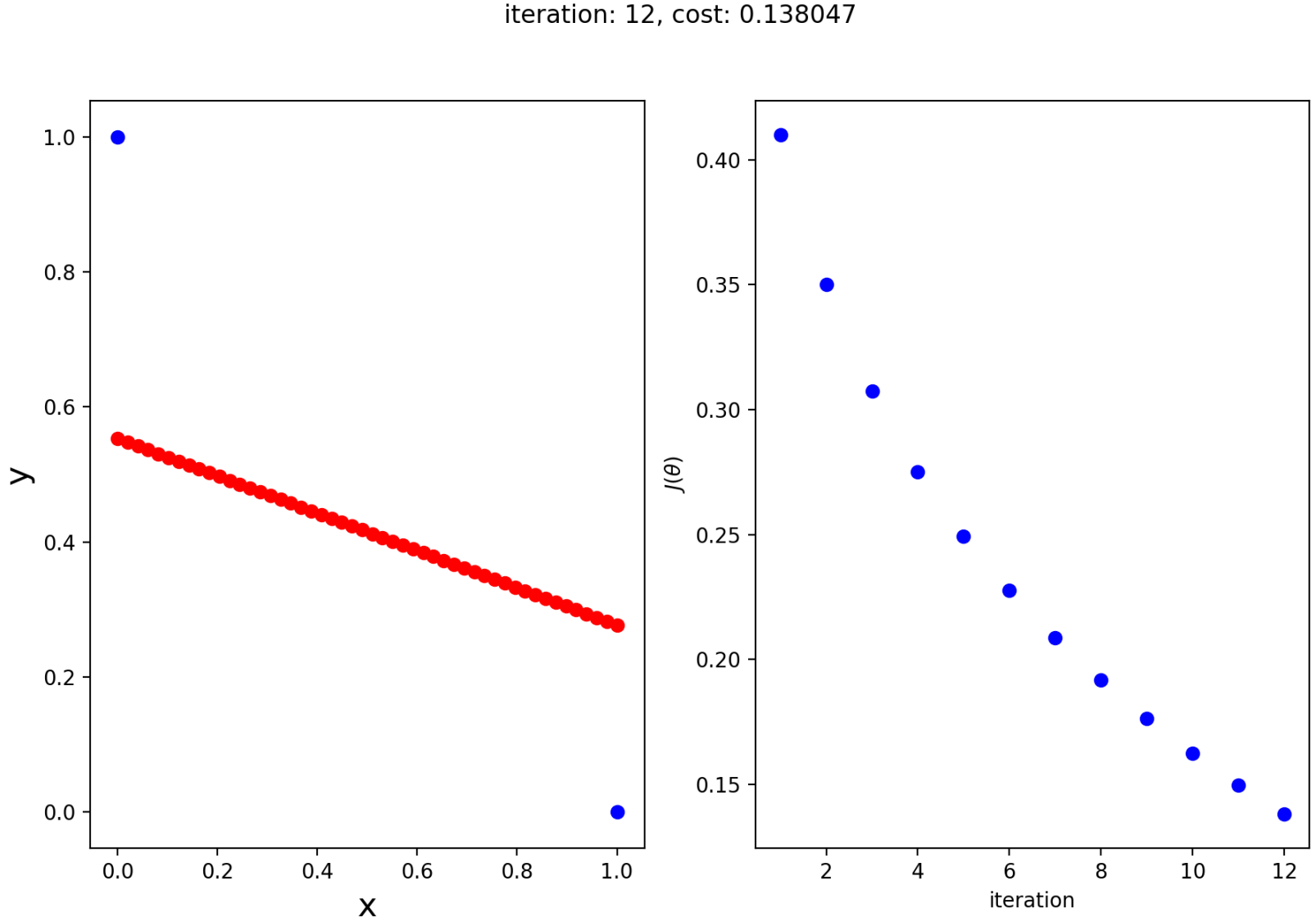
Toy example, iteration 1



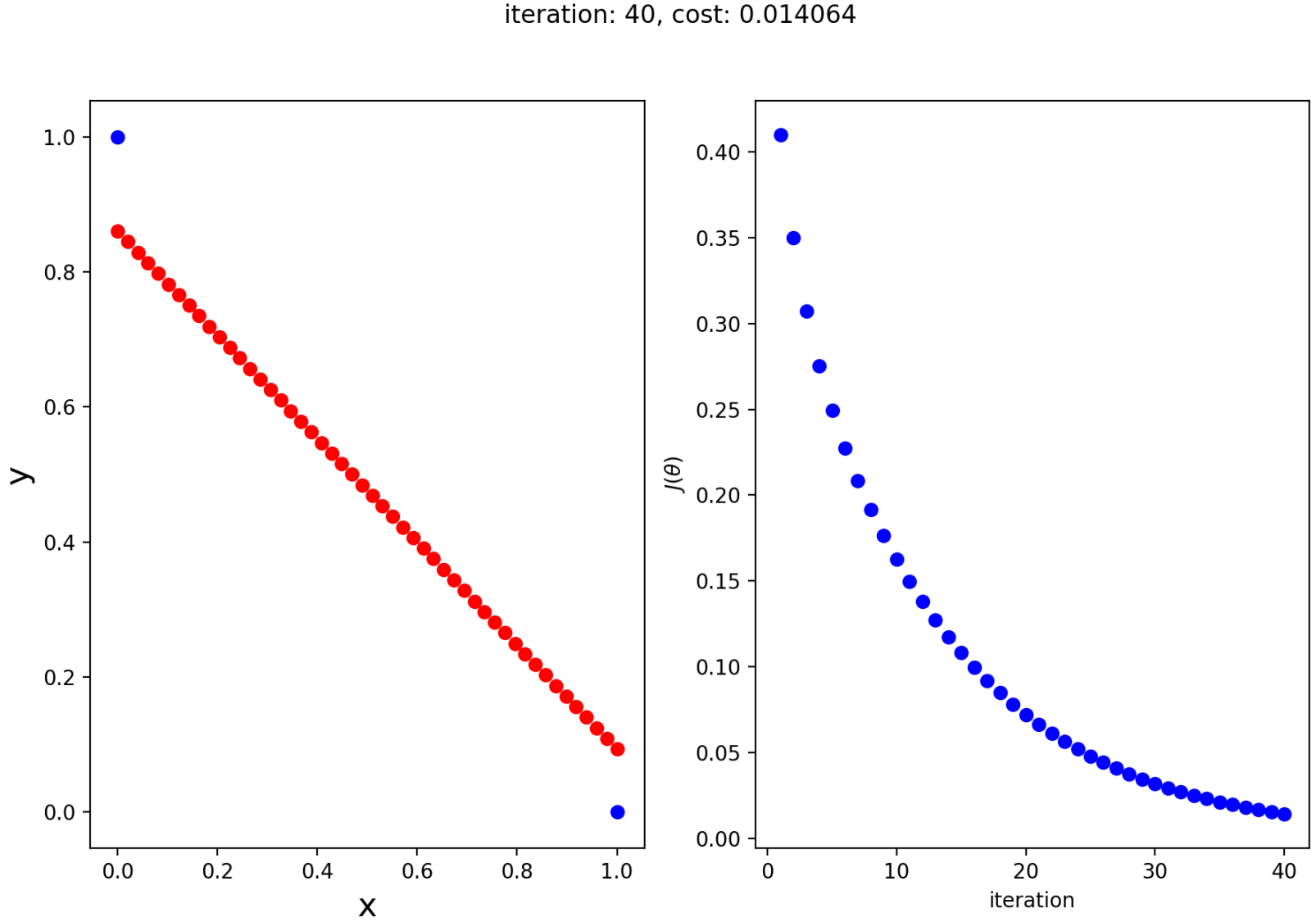
Toy example, iteration 2



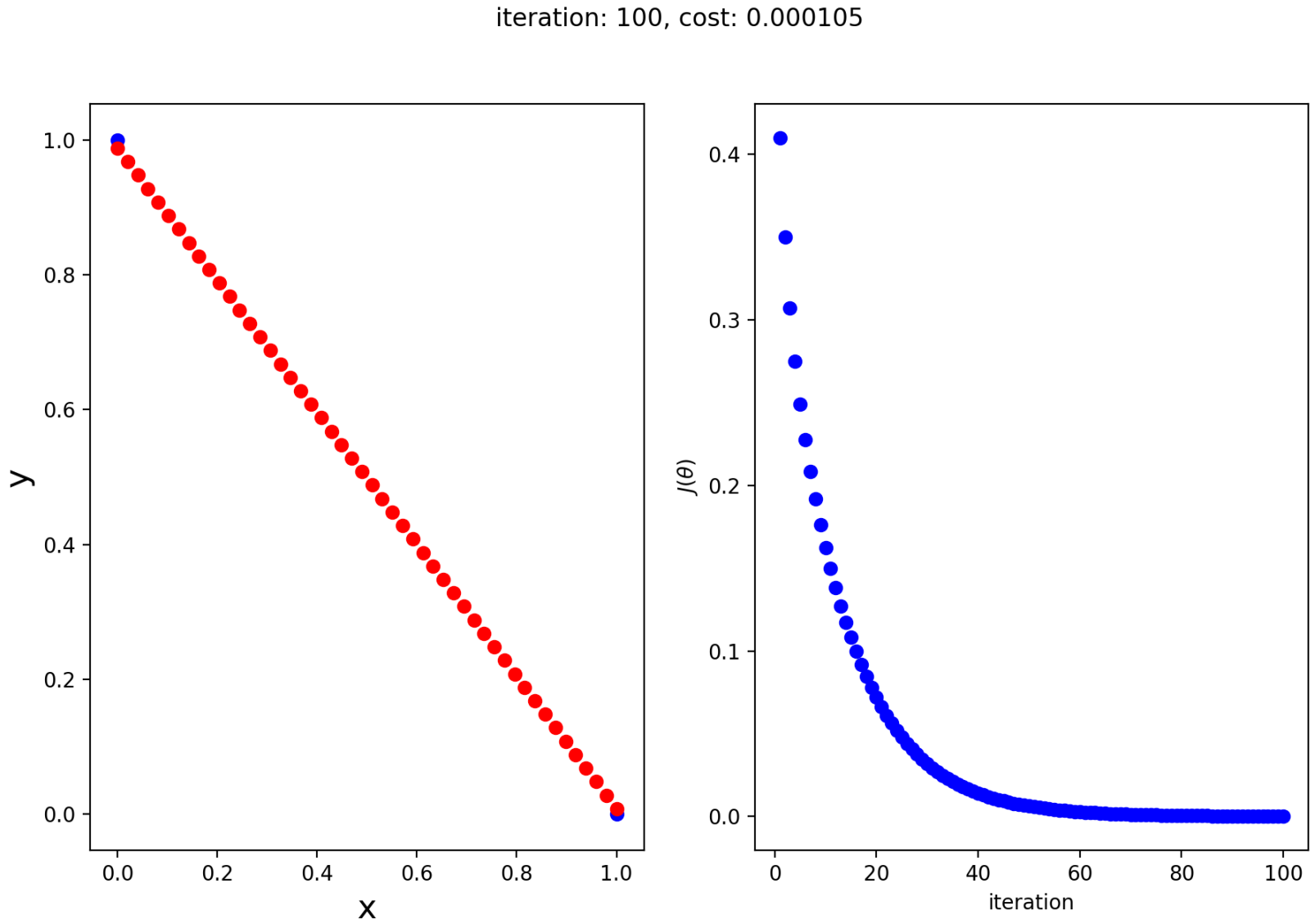
Toy example, iteration 12



Toy example, iteration 40



Toy example, iteration 100



Handout 5

Linear Regression Runtime

T = # iterations of SGD

n = # examples

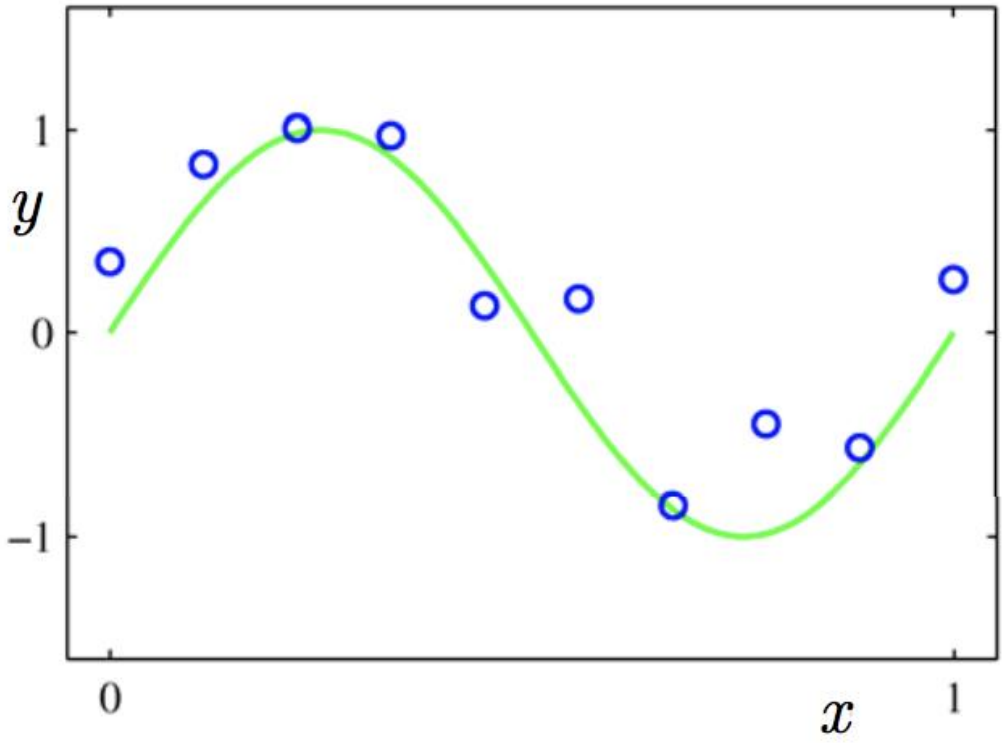
p = # features

1) What is the runtime of SGD?
2) What is the runtime of the analytic solution?

Challenge outside of class!

Polynomial Regression

Do not need to limit ourselves to a line



Plot: Jessica Wu

Polynomial Regression

$$\boldsymbol{X} = \begin{bmatrix} x_1^0 & x_1^1 & x_1^2 & \cdots & x_1^d \\ & & \vdots & & \\ x_n^0 & x_n^1 & x_n^2 & \cdots & x_n^d \end{bmatrix}$$