DSC 140A

Probabilistic Modeling & Machine Learning

Lecture 3 | Part 1

Recap

Empirical Risk

- Last time, we framed the problem of learning as **minimizing** the **empirical risk**.

$$R(H) = \frac{1}{n} \sum_{i=1}^{n} \ell(H(\vec{x}^{(i)}), y_i)$$

- In the case where H is linear::

$$R(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} \ell(\vec{w} \cdot \text{Aug}(\vec{x}^{(i)}), y_i)$$

Minimizing Empirical Risk

- Picking different loss functions changes the optimization problem.
- If we use **square loss**:

$$R(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(\vec{w}\cdot \text{Aug}(\vec{x}^{(i)}) - y_i)^2$$

- We can minimize by setting the gradient to zero.
- We get: $\vec{w} = (X^T X)^{-1} X^T \vec{y}$.

Minimizing Empirical Risk

- But sometimes we can't use this approach.
  - If R is not differentiable (absolute loss).
  - If computing $\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$ is too expensive.
  - ...

Today

- A general, very popular approach to optimization: **gradient descent**.
- Instead of solving for $\vec{w}^*$ "all at once", we'll iterate towards it.

DSC 140A

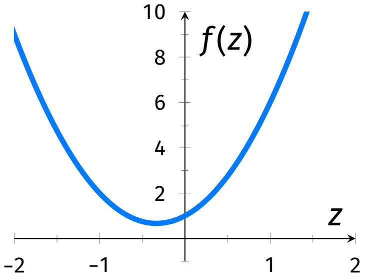
Probabilistic Modeling & Machine Learning

Lecture 3 | Part 2

What is the gradient?

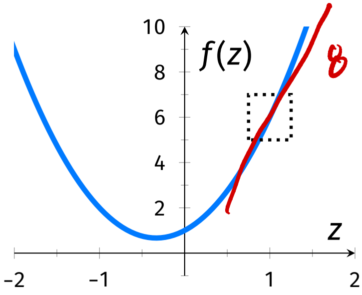
What is the derivative?

- Consider $f(z) = 3z^2 + 2z + 1$.
  - What is the **slope** of the curve at $z = 1$?



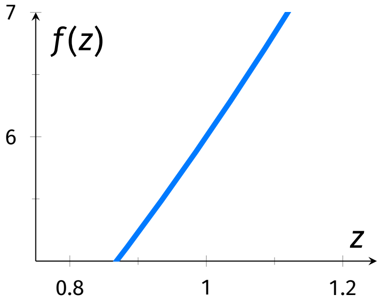
What is the derivative?

- Consider $f(z) = 3z^2 + 2z + 1$.
  - What is the **slope** of the curve at $z = 1$?



What is the derivative?

- Consider $f(z) = 3z^2 + 2z + 1$.
  - What is the **slope** of the curve at $z = 1$?



What is the derivative?

- The **derivative** gives the slope anywhere:

$$f(z) = 3z^2 + 2z + 1$$

$$\frac{df}{dz}(z) = 6z + 2$$

The slope of the curve at $z = 1$:

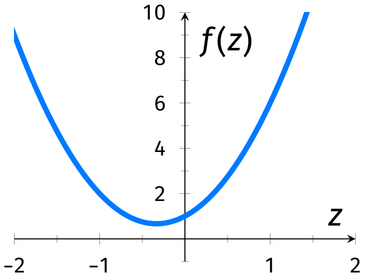
$$\frac{df}{dz}(1) = 6(1) + 2 = 8$$

What type of object?

- The derivative of $f : \mathbb{R} \to \mathbb{R}$ is a **function**:
  - Input: scalar.
  - Output: scalar.
  - Example: $\frac{df}{dz}(z) = 6z + 2$.
- The derivative **evaluated at a point** is a **scalar**:
  - Example: $\frac{df}{dz}(1) = 8$.

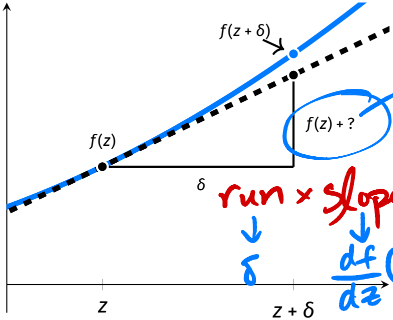
Sign of the Derivative

- If the derivative at a point is:
  - Positive: the function is **increasing**.
  - Negative: the function is **decreasing**.
  - Zero: the function is **flat**.



Exercise

What is the height of the dashed line at $z + \delta$?



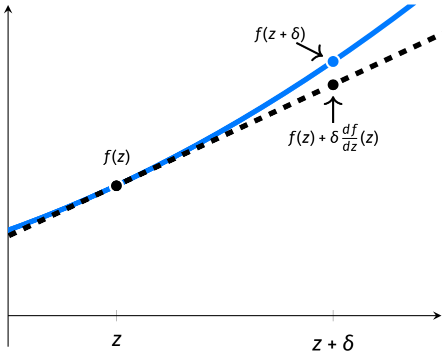
$$f(z) + \delta \frac{df}{dz}(z)$$

run × slope = rise

δ $\frac{df}{dz}(z)$

Derivatives and Change

- The derivative tells us **how much** the function changes with an infinitesimal increase in z.

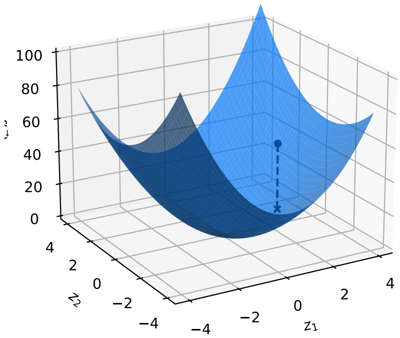


Increases and Decreases

- The sign of the derivative tells us if the function is increasing or decreasing.
  - Positive: f is increasing at z.
  - Negative: f is decreasing at z.

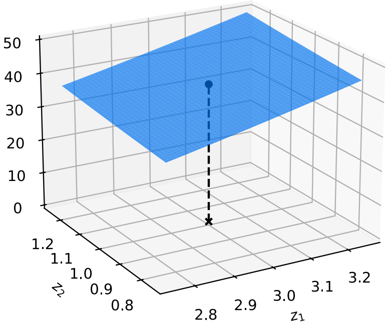
Multivariate Functions

- Now consider $f(\vec{z}) = f(z_1, z_2) = 4z_1^2 + 2z_2 + 2z_1 z_2$.
  - What is the **slope** of the surface at $(z_1, z_2) = (3, 1)$?



Multivariate Functions

- Now consider $f(\vec{z}) = f(z_1, z_2) = 4z_1^2 + 2z_2 + 2z_1 z_2$.
  - What is the **slope** of the surface at $(z_1, z_2) = (3, 1)$?



Partial Derivatives

- When f is a function of a vector $\vec{z} = (z_1, z_2)^T$, there are **two** slopes to talk about:

- $\frac{\partial f}{\partial z_1}$: slope in the z_1 direction.

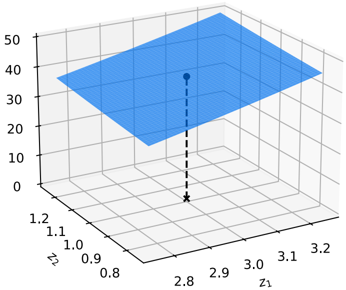
- $\frac{\partial f}{\partial z_2}$: slope in the z_2 direction.

Example

What is the slope of f at $(z_1, z_2) = (3, 1)$ in:

- The z_1 direction?
- The z_2 direction?

$$f(\vec{z}) = 4z_1^2 + 2z_2 + 2z_1z_2$$



- $\frac{\partial f}{\partial z_1}(z_1, z_2) = 8z_1 + 2z_2$
- $\frac{\partial f}{\partial z_1}(3, 1) = 8(3) + 2(1) = 26$
- $\frac{\partial f}{\partial z_2}(z_1, z_2) = 2 + 2z_1$
- $\frac{\partial f}{\partial z_2}(3, 1) = 2 + 2(3) = 8$

What is the gradient?

- We can package the partial derivatives into a single object: the **gradient**.

$$\frac{df}{d\vec{z}}(\vec{z}) = \begin{pmatrix} \frac{\partial f}{\partial z_1}(\vec{z}) \\ \frac{\partial f}{\partial z_2}(\vec{z}) \end{pmatrix}$$

What is the gradient?

- In general, if $f : \mathbb{R}^d \to \mathbb{R}$, then the gradient is:

$$\frac{df}{d\vec{z}}(\vec{z}) = \begin{pmatrix} \frac{\partial f}{\partial z_1}(\vec{z}) \\ \frac{\partial f}{\partial z_2}(\vec{z}) \\ \vdots \\ \frac{\partial f}{\partial z_d}(\vec{z}) \end{pmatrix}$$

What type of object?

$\frac{d}{d\vec{w}} \vec{w}^T A \vec{w}$

$\|\vec{w}\|^2$

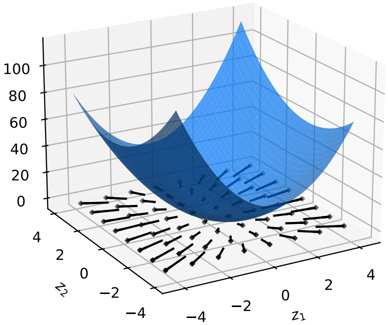
- The gradient of a function $f : \mathbb{R}^d \to \mathbb{R}$ is a **function**[1]:
  - Input: vector in $\mathbb{R}^d$.
  - Output: vector in $\mathbb{R}^d$.
  - Example: $\frac{df}{d\vec{z}}(\vec{z}) = (8z_1 + 2z_2, 2 + 2z_1)^T$.

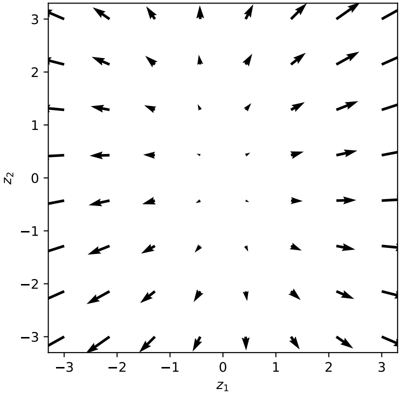
- The gradient of $f : \mathbb{R}^d \to \mathbb{R}$ **evaluated at a point** is a **vector** in $\mathbb{R}^d$:
  - Example: $\frac{df}{d\vec{z}}(2, 1) = (18, 6)^T$.

[1] Sometimes it is referred to as a **vector field**.

Gradient Fields

- The gradient can be viewed as a **vector field**:



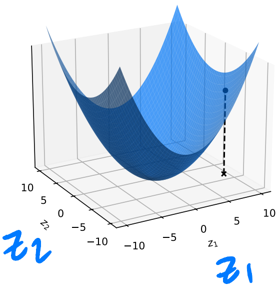


Meaning of Gradient Vector

- The gradient of a function $f : \mathbb{R}^d \to \mathbb{R}$ at a point $\vec{z}$ is a vector in $\mathbb{R}^d$.

- The ith component is the **slope** of f at $\vec{z}$ in the ith direction.

Exercise



Which of these could possibly be the gradient at the point (9, −4)?

- A) ~~(0, 0)~~
- B) (4, −1)
- C) (−4, −1)
- D) (−4, 1)

(+, −1)

Gradients and Change

- Recall: $f(z+\delta) \approx f(z) + \delta \times \frac{df}{dz}(z)$.
- In multiple dimensions:

$$f(\vec{z}+\vec{\delta}) \approx f(\vec{z}) + \left(\delta_1 \times \frac{\partial f}{\partial z_1}(\vec{z})\right) + \left(\delta_2 \times \frac{\partial f}{\partial z_2}(\vec{z})\right) + \ldots$$

$$\approx f(\vec{z}) + \vec{\delta} \cdot \frac{df}{d\vec{z}}(\vec{z})$$

$$f(\vec{z}+\vec{\delta}) \approx f(\vec{z}) + (\text{change}^{0.1} \text{ in } z_1)\times(\text{slope}^{4} \text{ in } z_1)$$
$$\underset{7}{} + (\text{change in } z_2)_{0.1}\times(\text{slope in } z_2)_{-2}$$

Exercise

At a point $\vec{z} = (2,3)^T$, $f(\vec{z})$ is 7 and the gradient $\frac{df}{d\vec{z}}(\vec{z}) = (4,-2)^T$.

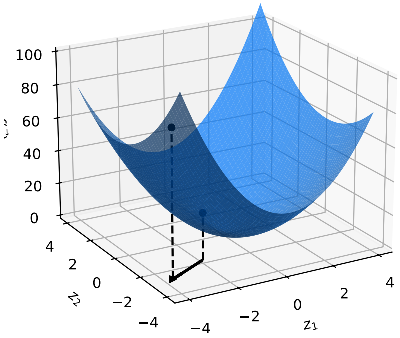
What is the approximate[a] value of $f(2.1, 3.1)$?

[a] Quality of approximation depends on second derivative.

$$7 + .4 - .2 = 7.2$$

Steepest Ascent

- **Key property**: the gradient vector points in the direction of **steepest ascent**.

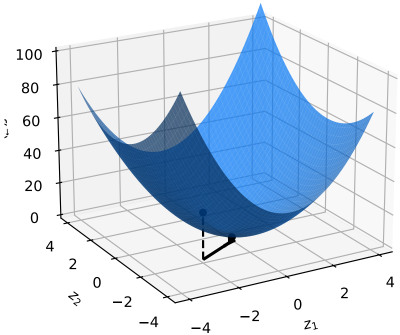


Proof

- Remember: $f(\vec{z}+\vec{\delta}) \approx f(\vec{z}) + \vec{\delta} \cdot \frac{df}{d\vec{z}}(\vec{z})$.
- So the total change is $\vec{\delta} \cdot \frac{df}{d\vec{z}}(\vec{z})$.
- Also remember: $\vec{\delta} \cdot \frac{df}{d\vec{z}}(\vec{z}) = \|\vec{\delta}\| \left\|\frac{df}{d\vec{z}}(\vec{z})\right\| \cos\theta$.
- So the increase in f is maximized when $\theta = 0$.
  - That is, when $\vec{\delta}$ points in the direction of $\frac{df}{d\vec{z}}(\vec{z})$.

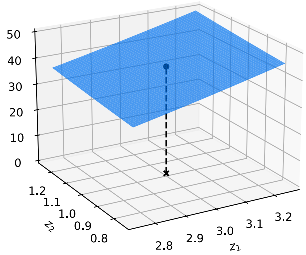
Steepest Descent

- The **negative** gradient points in the direction of **steepest descent**.

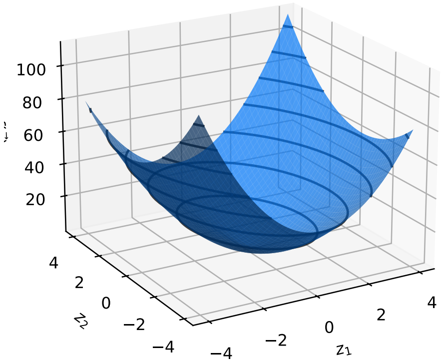


Why?

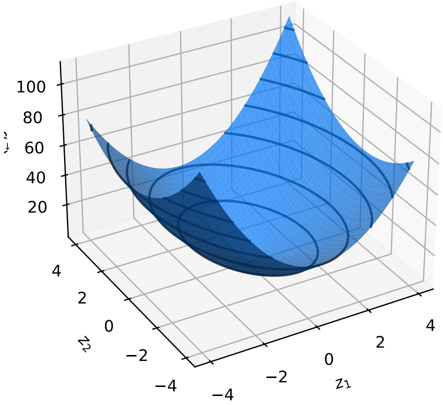
- The direction of steepest ascent is the **opposite** of the direction of steepest descent.
- Because, zoomed in, the function looks linear.



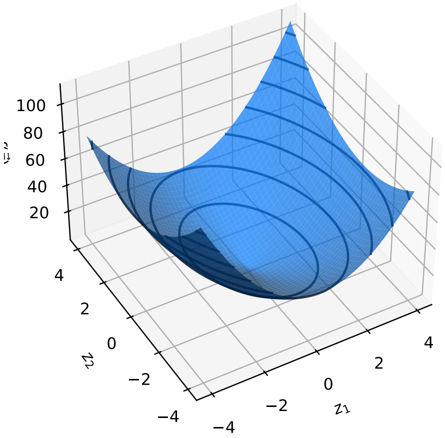
Contours



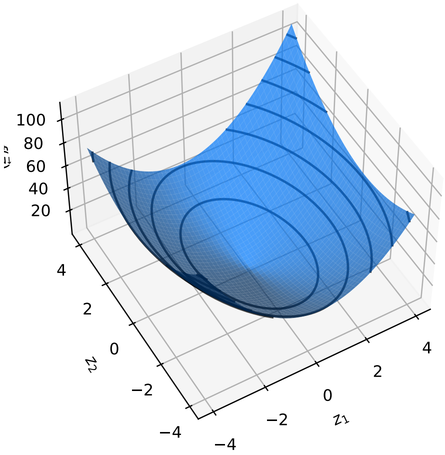
Contours



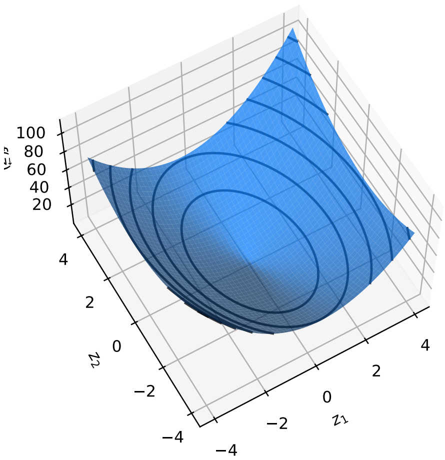
Contours



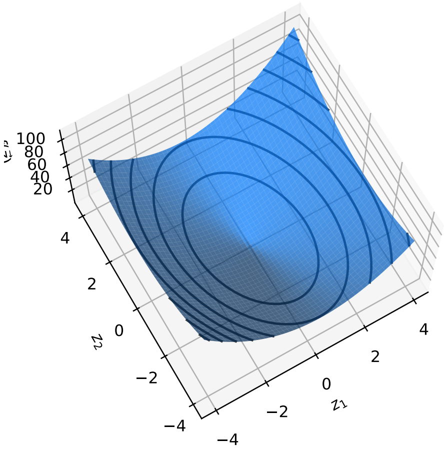
Contours



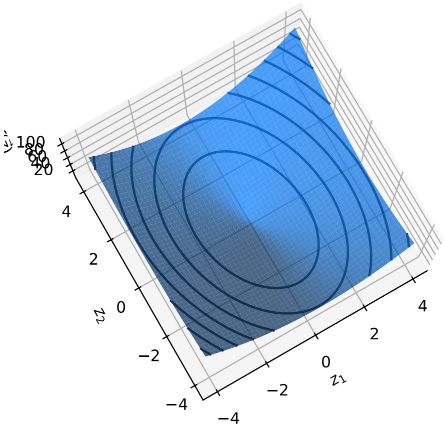
Contours



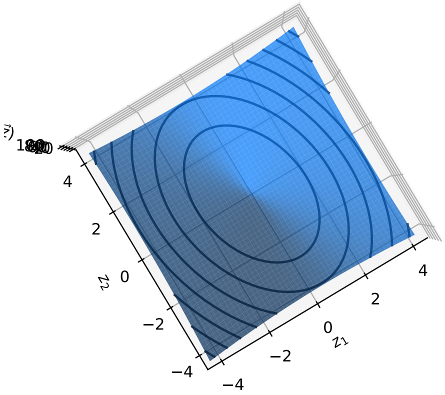
Contours



Contours

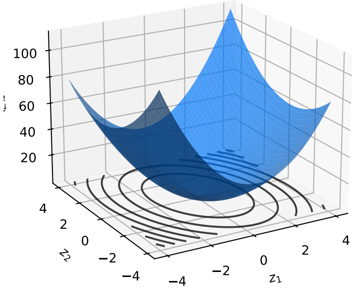


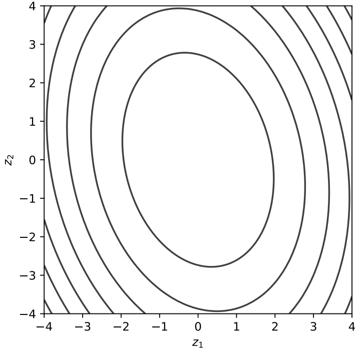
Contours



Contours

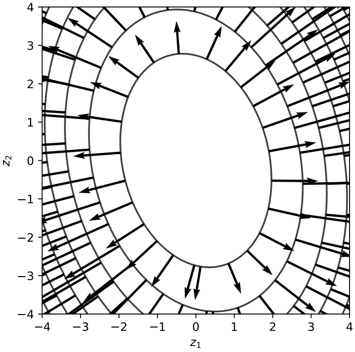
- The contours are the **level sets** of the function.





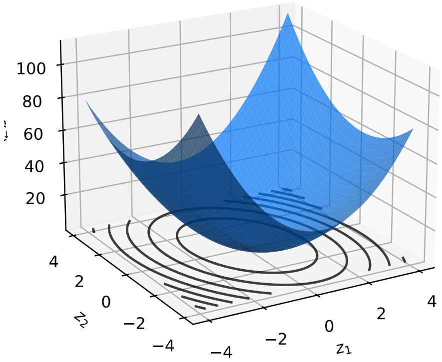
Contours and Gradients

- The gradient is **orthogonal** to the contours.



Optimization

- To find a **minimum** (or **maximum**), look for where the gradient is $\vec{0}$.



DSC 140A

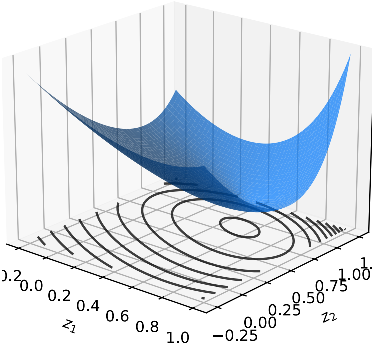
Probabilistic Modeling & Machine Learning

Lecture 3 | Part 3

Gradient Descent

Example

- **Goal:** minimize $f(\vec{z}) = e^{z_1^2 + z_2^2} + (z_1 - 2)^2 + (z_2 - 3)^2$.



Example

- Try solving $\frac{df}{d\vec{z}}(\vec{z}) = 0$.

- The gradient is:

$$\frac{df}{d\vec{z}}(\vec{z}) = \begin{pmatrix} 2z_1 e^{z_1^2+z_2^2} + 2(z_1 - 2) \\ 2z_2 e^{z_1^2+z_2^2} + 2(z_2 - 3) \end{pmatrix} = \begin{matrix} 0 \\ 0 \end{matrix}$$

- Can we solve the system?

$$2z_1 e^{z_1^2+z_2^2} + 2(z_1 - 2) = 0$$
$$2z_2 e^{z_1^2+z_2^2} + 2(z_2 - 3) = 0$$

Example

- Try solving $\frac{df}{d\vec{z}}(\vec{z}) = 0$.

- The gradient is:

$$\frac{df}{d\vec{z}}(\vec{z}) = \begin{pmatrix} 2z_1 e^{z_1^2+z_2^2} + 2(z_1 - 2) \\ 2z_2 e^{z_1^2+z_2^2} + 2(z_2 - 3) \end{pmatrix}$$

- Can we solve the system? **Not in closed form.**

$$2z_1 e^{z_1^2+z_2^2} + 2(z_1 - 2) = 0$$
$$2z_2 e^{z_1^2+z_2^2} + 2(z_2 - 3) = 0$$

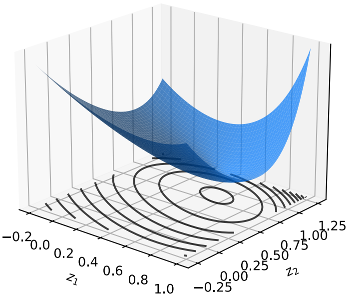
A Problem

- The function **is differentiable**[2].
- But we can't set gradient to zero and solve.
- **How do we find the minimum**?

[2] The gradient exists everywhere.

A Solution

- **Idea**: iterate towards a minimum, step by step.
- Start at an arbitrary location.
- At every step, move in direction of **steepest descent.**
  - i.e., the negative gradient.



Exercise

The gradient of a function $f(\vec{z})$ at $(1, 1)$ is $(2, 1)^T$.

If you're trying to minimize $f(\vec{z})$, which place should you go to next?

- A) $(1, 1)$
- B) $(.8, .9)$
- C) $(1.2, 1.1)$

Direction of Steepest Descent

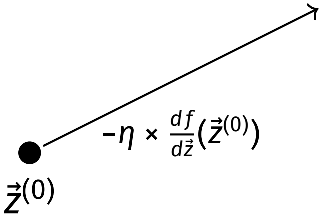
- If η is the **learning rate**, then the next step is:

$$\vec{z}^{(t+1)} = \vec{z}^{(t)} - \eta \times \frac{df}{d\vec{z}}(\vec{z}^{(t)})$$

Direction of Steepest Descent

- If η is the **learning rate**, then the next step is:

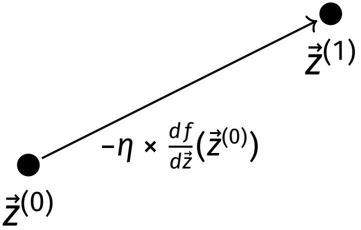
$$\vec{z}^{(t+1)} = \vec{z}^{(t)} - \eta \times \frac{df}{d\vec{z}}(\vec{z}^{(t)})$$



Direction of Steepest Descent

- If η is the **learning rate**, then the next step is:

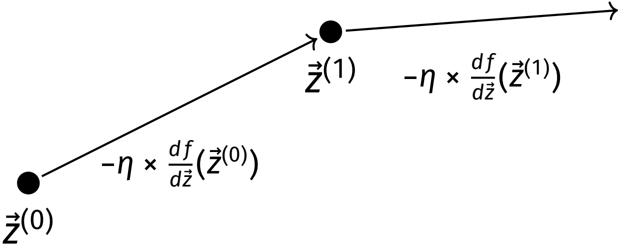
$$\vec{z}^{(t+1)} = \vec{z}^{(t)} - \eta \times \frac{df}{d\vec{z}}(\vec{z}^{(t)})$$



Direction of Steepest Descent

- If η is the **learning rate**, then the next step is:

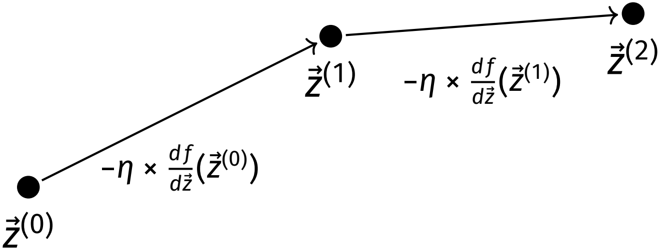
$$\vec{z}^{(t+1)} = \vec{z}^{(t)} - \eta \times \frac{df}{d\vec{z}}(\vec{z}^{(t)})$$



Direction of Steepest Descent

- If η is the **learning rate**, then the next step is:

$$\vec{z}^{(t+1)} = \vec{z}^{(t)} - \eta \times \frac{df}{d\vec{z}}(\vec{z}^{(t)})$$

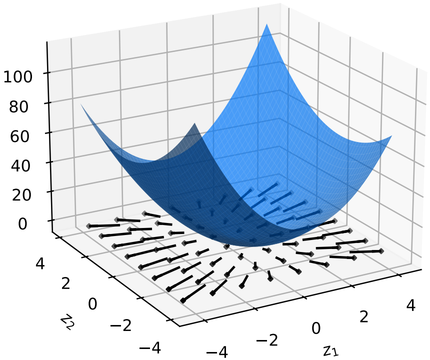


Gradient Descent

To minimize $f(\vec{z})$:

- Pick arbitrary starting point $\vec{z}^{(0)}$, **learning rate** $\eta > 0$
- Until convergence, repeat:
  - **Compute gradient:** $\frac{df}{d\vec{z}}(\vec{z}^{(t)})$ at $\vec{z}^{(t)}$.
  - **Update:** $\vec{z}^{(t+1)} = \vec{z}^{(t)} - \eta \times \frac{df}{d\vec{z}}(\vec{z}^{(t)})$.
- When converged, return $\vec{z}^{(t)}$.
  - It is (approximately) a local minimum.

Stopping Criterion



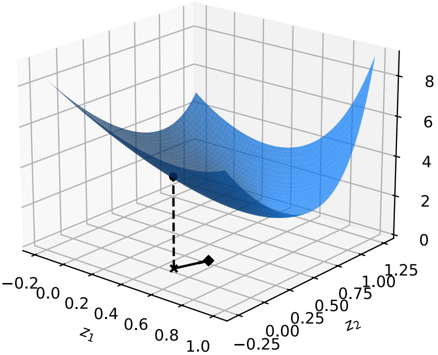
- Close to a minimum, gradient is small.
- **Idea:** stop when $\left\| \frac{df}{d\vec{z}}(\vec{z}^{(t)}) \right\|$ is small.
- **Alternative:** stop when $\| \vec{z}^{(t+1)} - \vec{z}^{(t)} \|$ is small.

```python
def gradient_descent(
    gradient, z_0, learning_rate, stop_threshold
):
    z = z_0
    while True:
        z_new = z - learning_rate * gradient(z)
        if np.linalg.norm(z_new - z) < stop_threshold:
            break
        z = z_new
    return z_new
```

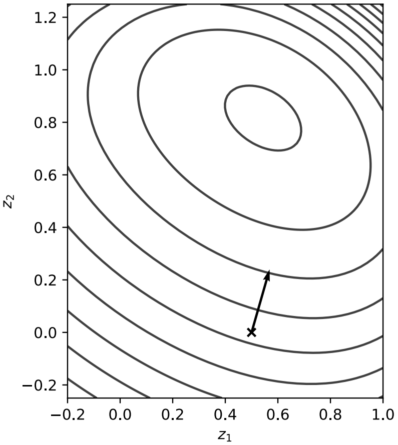
Picking Parameters

- The learning rate and stopping threshold are **parameters**.
- They need to be chosen carefully for each problem.
- If not, the algorithm **may not converge**.

Example

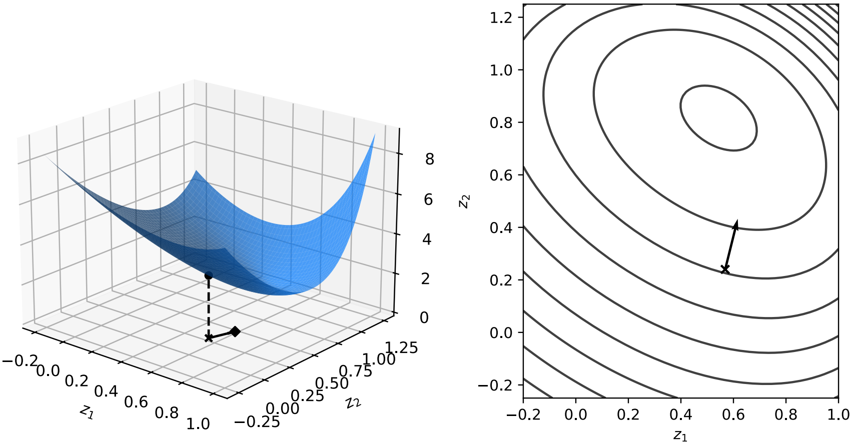


8
6
4
2
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
z_2



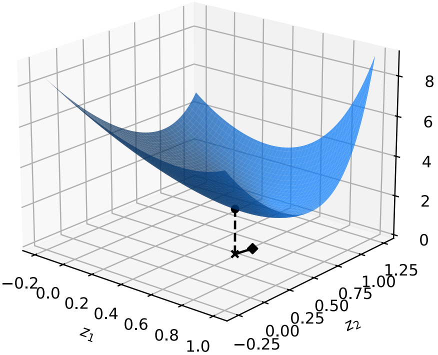
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
z_2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1

Example

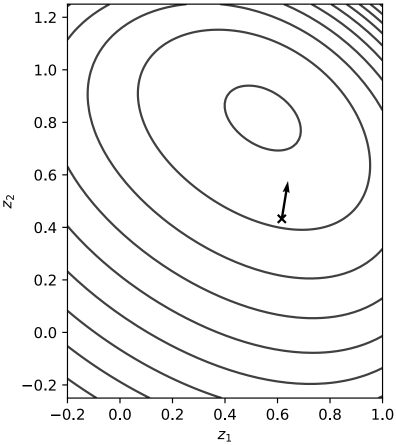


z_1
z_2
−0.2 0.0 0.2 0.4 0.6 0.8 1.0
−0.25 0.00 0.25 0.50 0.75 1.00 1.25
0 2 4 6 8
z_1
z_2
−0.2 0.0 0.2 0.4 0.6 0.8 1.0
−0.2 0.0 0.2 0.4 0.6 0.8 1.0 1.2

Example

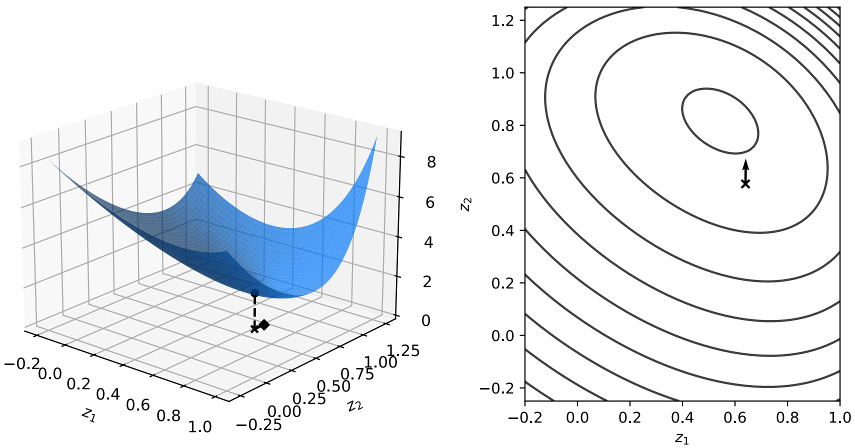


8
6
4
2
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
z_2



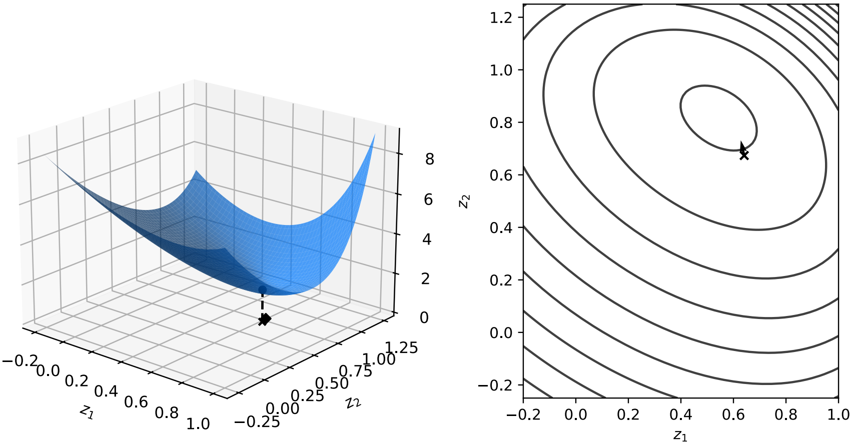
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
z_2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1

Example



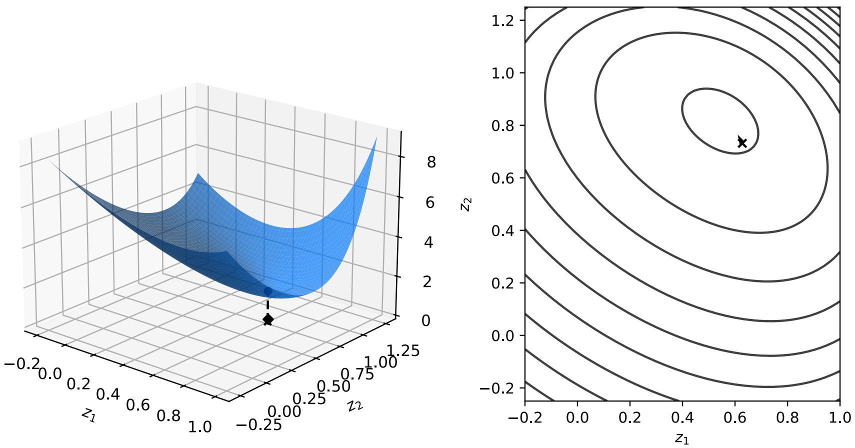
8
6
4
2
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
z_2
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
z_2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1

Example



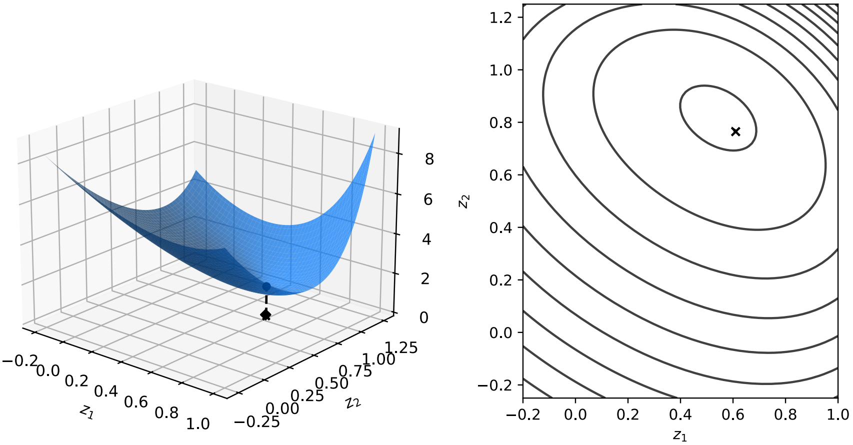
8
6
4
2
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
z_2
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
z_2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1

Example



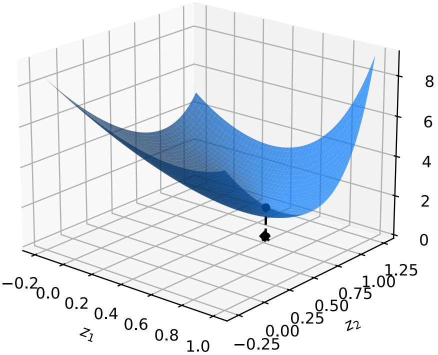
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
z_2
0
2
4
6
8
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
z_2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1

Example

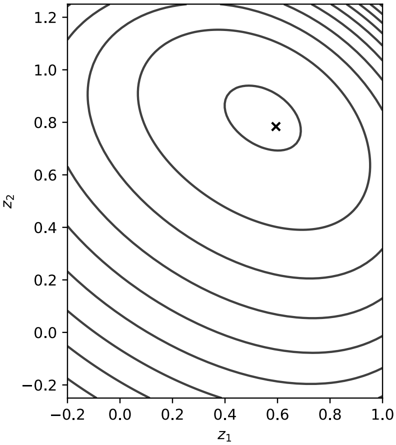


−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
z_2
0
2
4
6
8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
z_2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1

Example

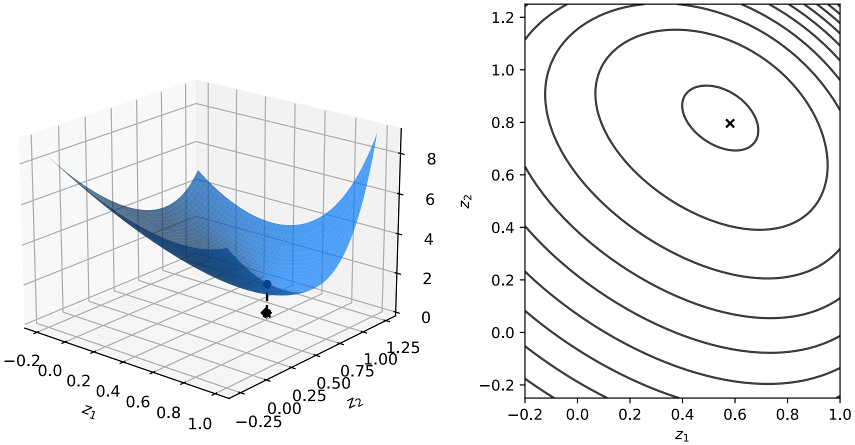


8
6
4
2
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
z_2



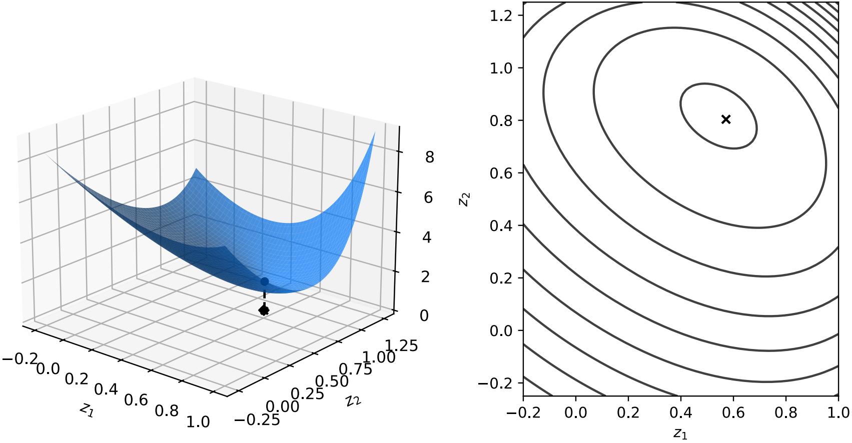
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
z_2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1

Example



8
6
4
2
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
z_2
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
z_2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1

Example



8
6
4
2
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
z_2
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
z_2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
z_1

$$\frac{d}{d\vec{z}} f(\vec{z}) = \begin{pmatrix} 4z_1^3 + z_2 \\ 6z_2 + z_1 \end{pmatrix}$$

$$\vec{z}^{(1)} = \vec{z}^{(1)} + \quad = \begin{pmatrix} .5 \\ .3 \end{pmatrix}$$

Exercise

Let $f(z_1, z_2) = z_1^4 + 3z_2^2 + z_1 z_2$.

Starting at $\vec{z}^{(0)} = (1, 1)$, what is the next point after one step of gradient descent with learning rate $\eta = 0.1$?

$$\frac{df}{d\vec{z}}(\vec{z}^{(0)}) = \begin{pmatrix} 4+1 \\ 6+1 \end{pmatrix} = \begin{pmatrix} 5 \\ 7 \end{pmatrix}$$

$$-\eta \cdot \frac{df}{d\vec{z}}(\vec{z}^{(0)}) = -0.1 \times \begin{pmatrix} 5 \\ 7 \end{pmatrix} = \begin{pmatrix} -.5 \\ -.7 \end{pmatrix}$$

DSC 140A

Probabilistic Modeling & Machine Learning

Lecture 3 | Part 4

Gradient Descent for ERM

Gradient Descent for ERM

- In ERM, our goal is to minimize **empirical risk**:[3]

$$R(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\text{Aug}(\vec{x}^{(i)}) \cdot \vec{w}, y_i)$$

- Often, we can minimize using **gradient descent**.

[3] We've assumed H is a linear prediction function.

The Gradient of the Risk

- The gradient of the empirical risk is:

$$\begin{aligned}\frac{dR}{d\vec{w}}(\vec{w}) &= \frac{d}{d\vec{w}}\left(\frac{1}{n}\sum_{i=1}^{n} \ell(\operatorname{Aug}(\vec{x}^{(i)}) \cdot \vec{w}, y_i)\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} \frac{d\ell}{d\vec{w}}(\operatorname{Aug}(\vec{x}^{(i)}) \cdot \vec{w}, y_i)\end{aligned}$$

- Gradient of risk is average gradient of loss.

- As far as we can go without knowing the loss.

The Gradient of the MSE

- Recall: the **mean squared error** is the empirical risk with respect to the square loss:

$$R(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(\text{Aug}(\vec{x}^{(i)}) \cdot \vec{w} - y_i)^2$$

- The gradient is:

$$\frac{dR}{d\vec{w}}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}\frac{d}{d\vec{w}}(\text{Aug}(\vec{x}^{(i)}) \cdot \vec{w} - y_i)^2$$

$$\frac{d}{dw}(xw) = x$$

Exercise

Recall that the square loss for a linear predictor is: $(\text{Aug}(\vec{x}^{(i)}) \cdot \vec{w} - y_i)^2$.

What is the gradient of the square loss with respect to $\vec{w}$?

$$\text{Aug}(\vec{x}^{(i)}) \qquad 0$$

$$2\left(\text{Aug}(\vec{x}^{(i)}) \cdot \vec{w} - y_i\right) \times \frac{d}{d\vec{w}}\left(\cancel{\text{Aug}(\vec{x}^{(i)}) \cdot \vec{w}} - \cancel{y_i}\right)$$

The Gradient of the MSE

- The gradient of the mean squared error is:[4]

$$\frac{dR}{d\vec{w}}(\vec{w}) = \frac{2}{n}\sum_{i=1}^{n}(\text{Aug}(\vec{x}^{(i)}) \cdot \vec{w} - y_i)\,\text{Aug}(\vec{x}^{(i)})$$

- Each training point $\vec{x}^{(i)}$ contributes to the gradient.

[4] We saw before that $\frac{dR}{d\vec{w}}(\vec{w}) = 2X^TX\vec{w} - 2X^T\vec{y}$. These two are actually equal.

Exercise

What will be the gradient if every prediction is exactly correct?

$$\frac{dR}{d\vec{w}}(\vec{w}) = \frac{2}{n}\sum_{i=1}^{n}\left(\mathrm{Aug}(\vec{x}^{(i)}) \cdot \vec{w} - y_i\right)\mathrm{Aug}(\vec{x}^{(i)})$$

Gradient Descent for Least Squares

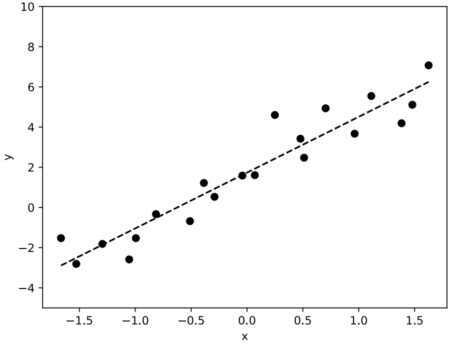
- We can perform least squares regression by solving the normal equations: $\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$.

- But we can find the **same solution** using **gradient descent**:

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta \times \frac{2}{n} \sum_{i=1}^{n} (\text{Aug}(\vec{x}^{(i)}) \cdot \vec{w}^{(t)} - y_i)\, \text{Aug}(\vec{x}^{(i)})$$

Example

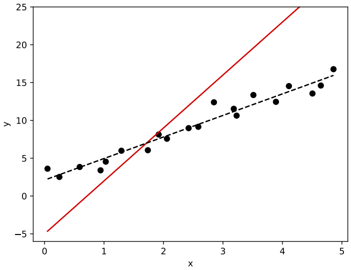
- We will run gradient descent to train a least squares regression model on the following data:



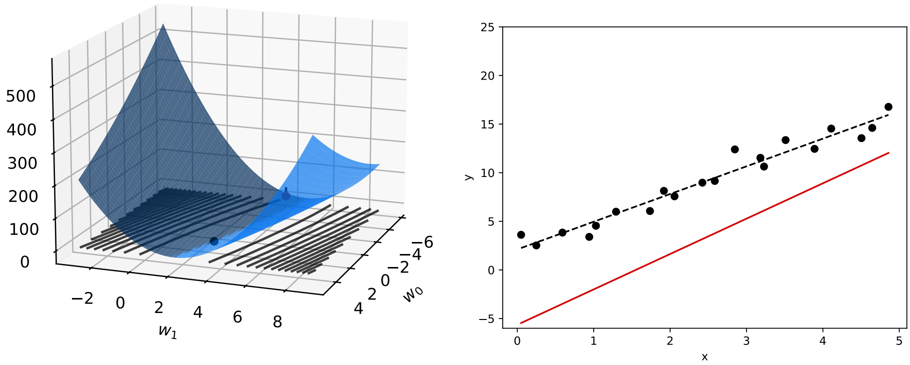
Exercise

The plot below shows a linear prediction function using weight vector $\vec{w}^{(0)}$.

What is the sign of the **second** entry of $\frac{dR}{d\vec{w}}(\vec{w}^{(0)})$?

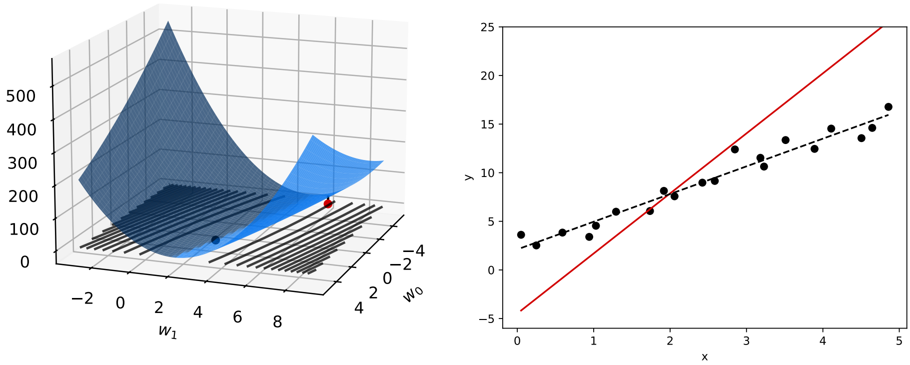


Iteration #1



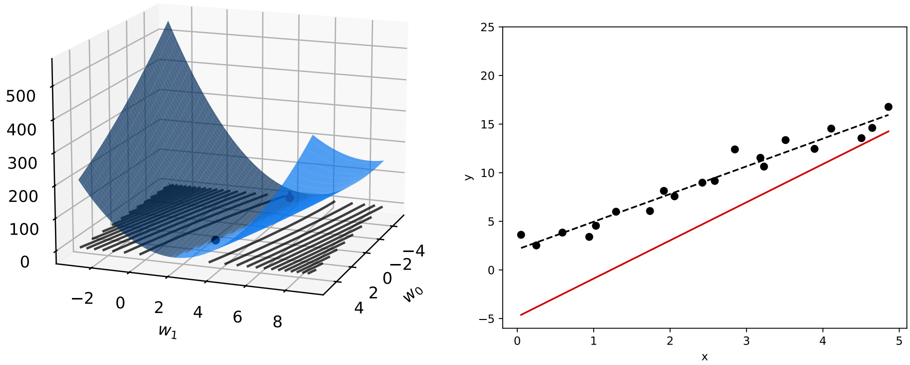
500
400
300
200
100
0
−2
0
2
4
6
8
w_1
−6
−4
−2
0
2
4
w_0
25
20
15
10
5
0
−5
y
0
1
2
3
4
5
x

Iteration #2



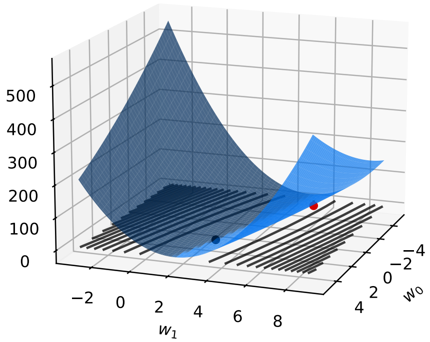
500
400
300
200
100
0
−2
0
2
4
6
8
w_1
−4
−2
0
2
4
w_0
25
20
15
10
5
0
−5
y
0
1
2
3
4
5
x

Iteration #3

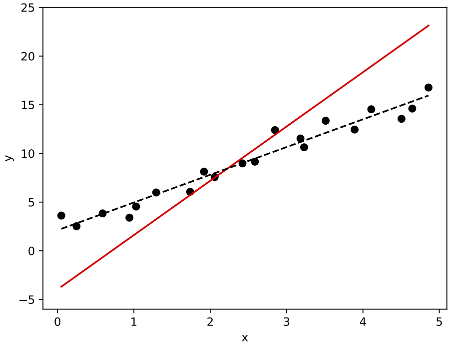


500
400
300
200
100
0
−2
0
2
4
6
8
w_1
−4
−2
0
2
4
w_0
25
20
15
10
5
0
−5
y
0
1
2
3
4
5
x

Iteration #4

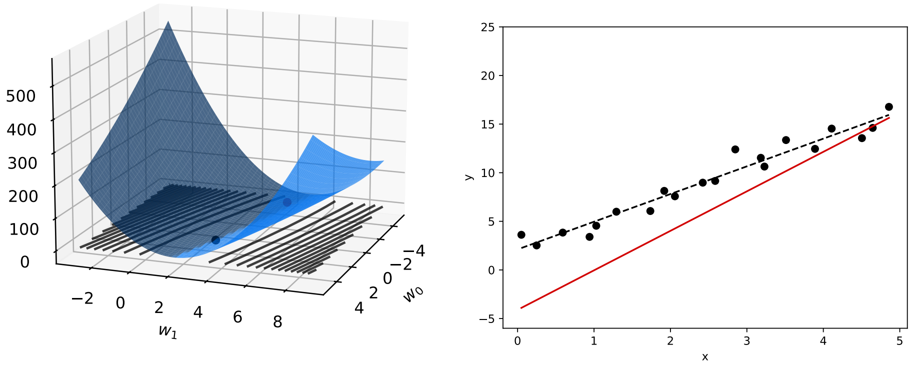


500
400
300
200
100
0
−2
0
2
4
6
8
w_1
−4
−2
0
2
4
w_0



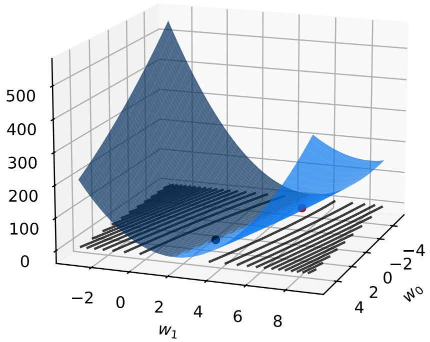
25
20
15
10
5
0
−5
y
0
1
2
3
4
5
x

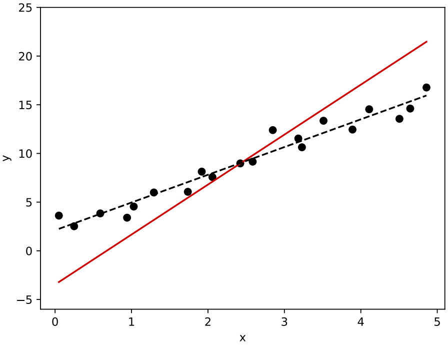
Iteration #5



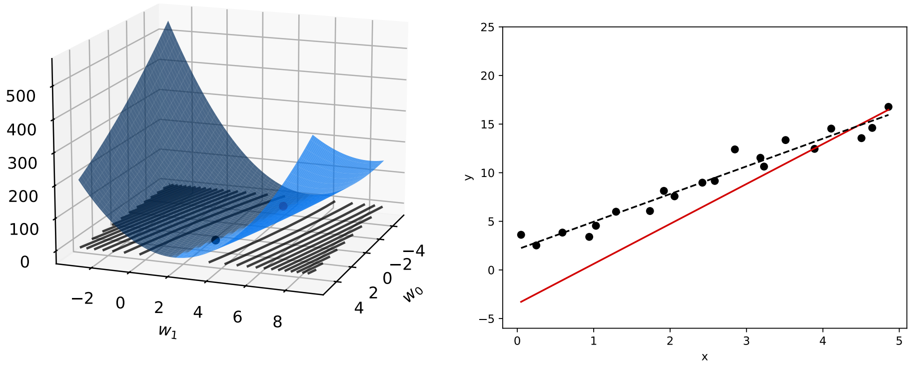
500
400
300
200
100
0
−2
0
2
4
6
8
w_1
−4
−2
0
2
4
w_0
25
20
15
10
5
0
−5
y
0
1
2
3
4
5
x

Iteration #6


500
400
300
200
100
0
−2
0
2
4
6
8
w_1
−4
−2
0
2
4
w_0

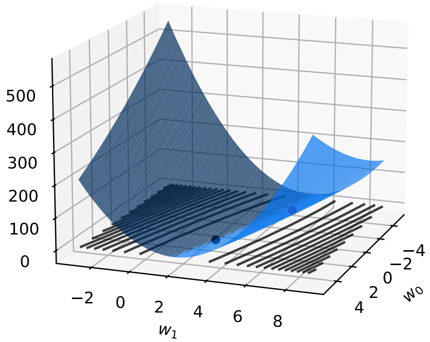

25
20
15
10
5
0
−5
y
0
1
2
3
4
5
x

Iteration #7

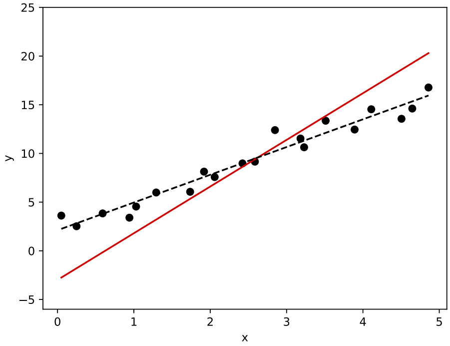


500
400
300
200
100
0
−2
0
2
4
6
8
w_1
−4
−2
0
2
4
w_0
25
20
15
10
5
0
−5
y
0
1
2
3
4
5
x

Iteration #8

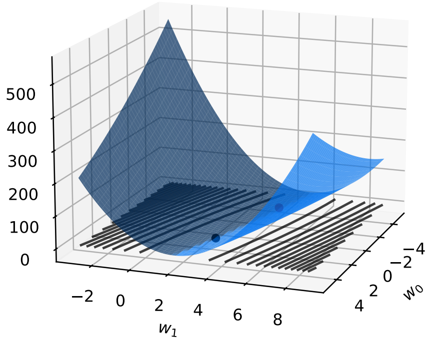


500
400
300
200
100
0
−2
0
2
4
6
8
w_1
−4
−2
0
2
4
w_0

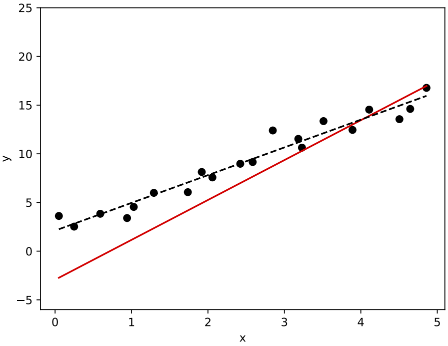


25
20
15
10
5
0
−5
y
0
1
2
3
4
5
x

Iteration #9

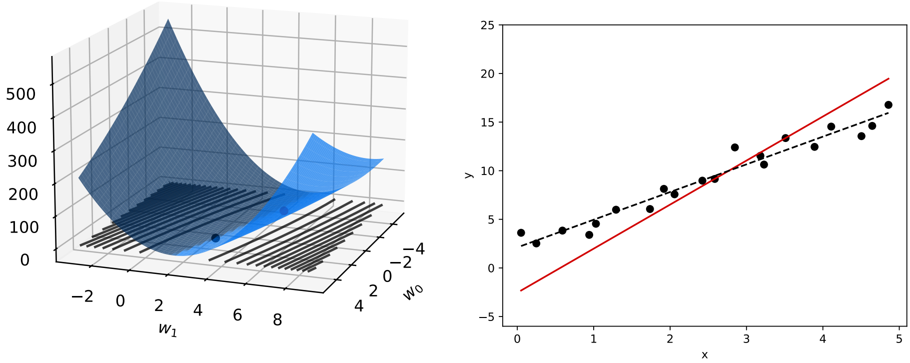


500
400
300
200
100
0
−2
0
2
4
6
8
w_1
−4
−2
0
2
4
w_0

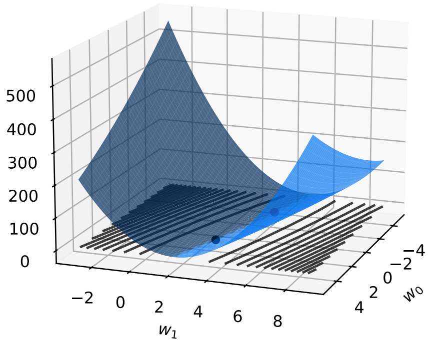


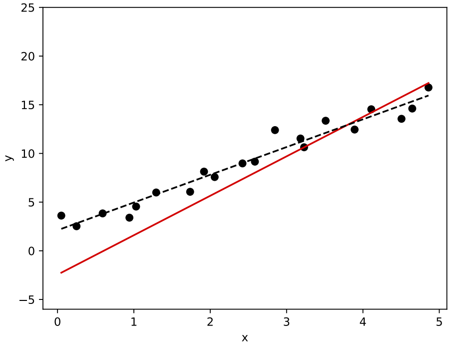
25
20
15
10
5
0
−5
y
0
1
2
3
4
5
x

Iteration #10

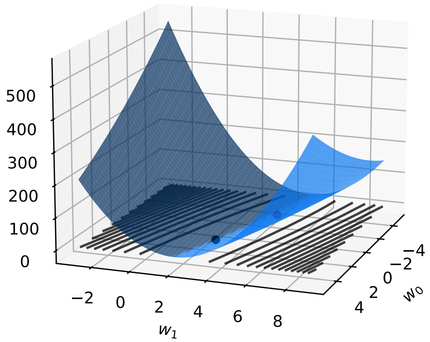


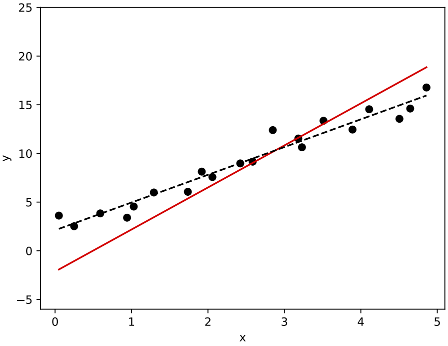
Iteration #11



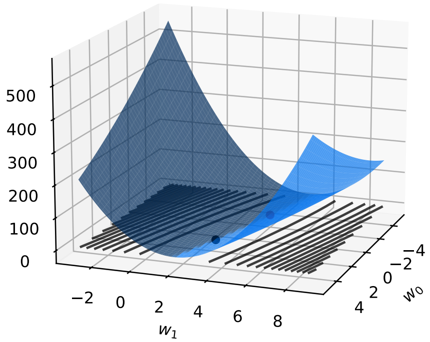


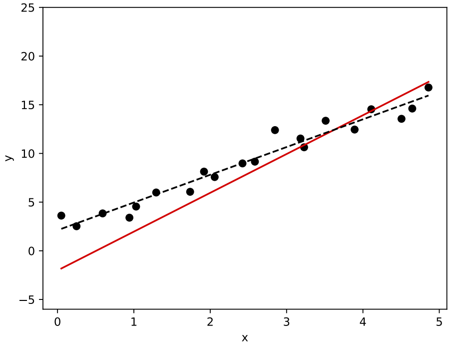
Iteration #12



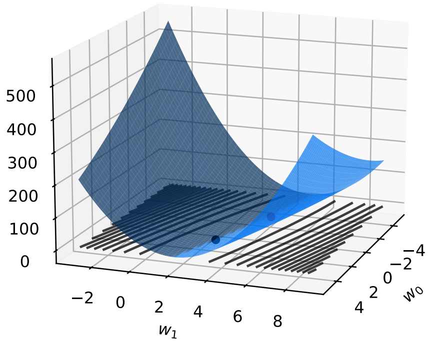


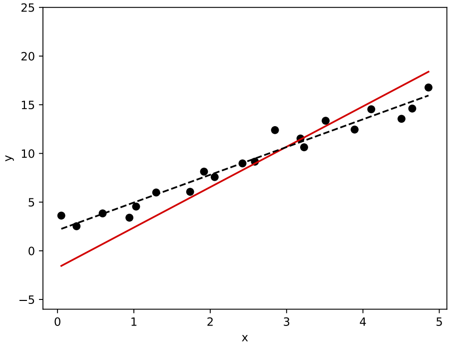
Iteration #13



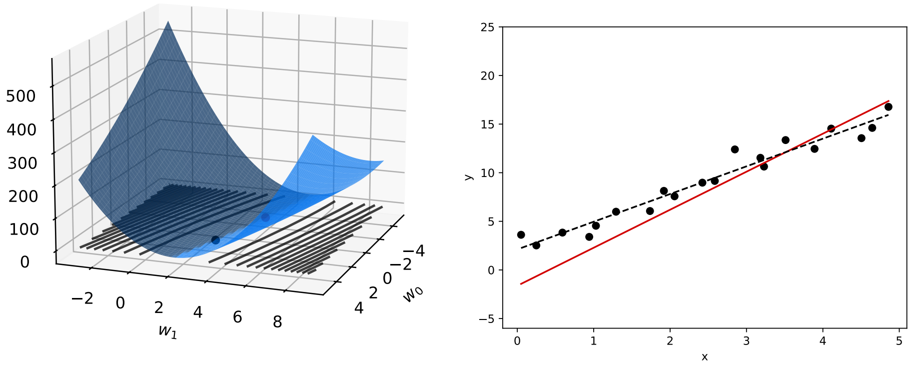


Iteration #14

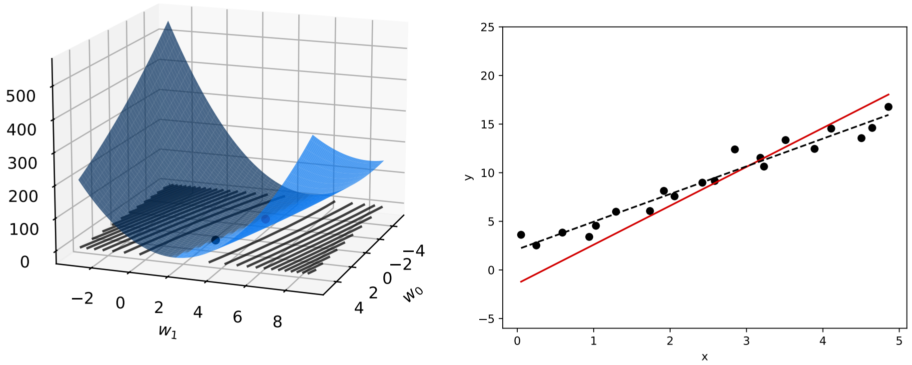




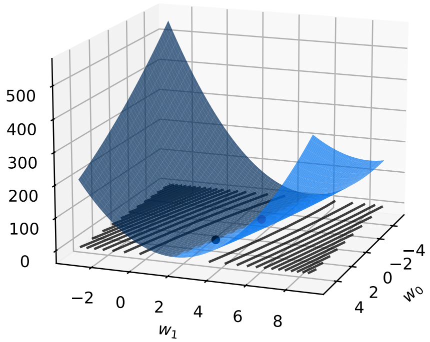
Iteration #15

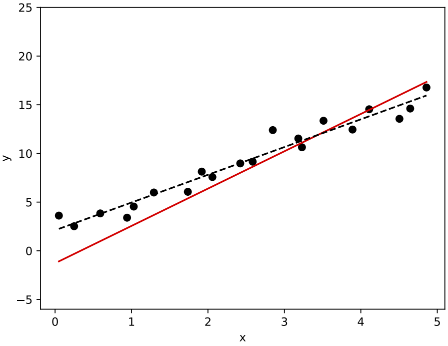


Iteration #16

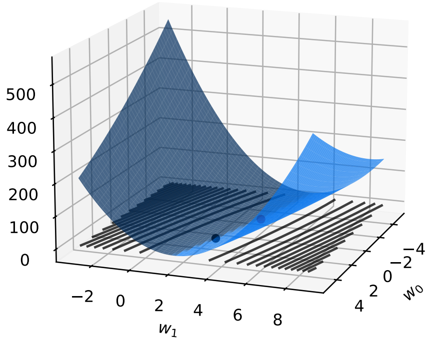


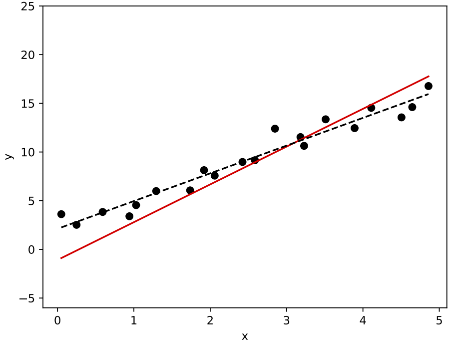
Iteration #17



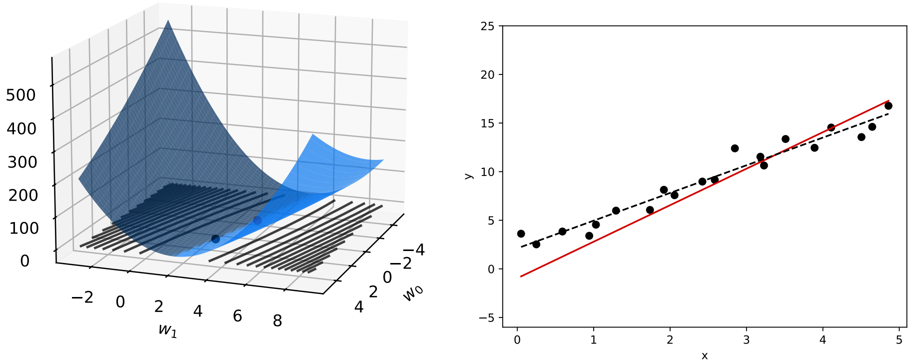


Iteration #18

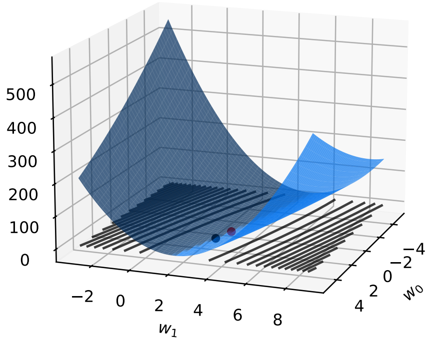


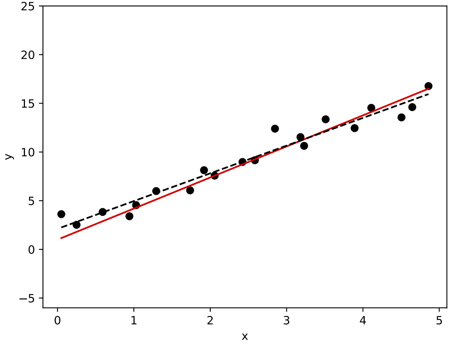


Iteration #19

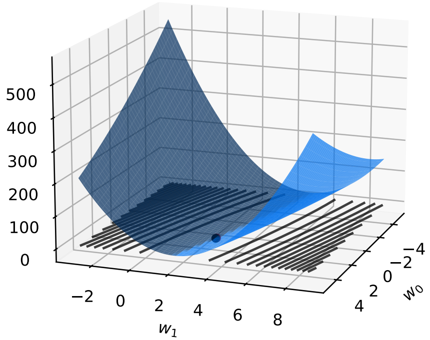


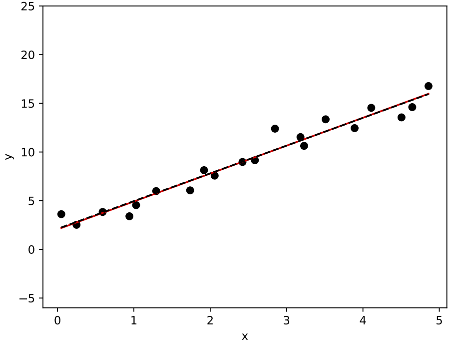
Iteration #40





Iteration #100





DSC 140A

Probabilistic Modeling & Machine Learning

Lecture 3 | Part 5

Stochastic Gradient Descent

Gradient Descent for Minimizing Risk

- In ML, we often want to minimize a **risk function**:

$$R(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(H(\vec{x}^{(i)}; \vec{w}), y_i)$$

Observation

- The gradient of the risk function is a sum of gradients:

$$\frac{d}{d\vec{w}}R(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}\frac{d}{d\vec{w}}\ell(H(\vec{x}^{(i)};\vec{w}), y_i)$$

- One term for each point in training data.

Problem

- In machine learning, the number of training points n can be **very large**.
- Computing the gradient can be **expensive** when n is large.
- Therefore, each step of gradient descent can be **expensive**.

Idea

- The (full) gradient of the risk uses all of the training data:

$$\frac{d}{d\vec{w}} R(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} \frac{d}{d\vec{w}} L(H(\vec{x}^{(i)}; \vec{w}), y_i)$$

- It is an average of n gradients.
- **Idea:** instead of using all n points, randomly choose $\ll n$.

Stochastic Gradient

- Choose a random subset (**mini-batch**) B of the training data.
- Compute a **stochastic gradient**:

$$\frac{d}{d\vec{w}} R(\vec{w}) \approx \sum_{i \in B} \frac{d}{d\vec{w}} L(H(\vec{x}^{(i)}; \vec{w}), y_i)$$

Stochastic Gradient

$$\frac{d}{d\vec{w}} R(\vec{w}) \approx \sum_{i \in B} \frac{d}{d\vec{w}} \ell(H(\vec{x}^{(i)}; \vec{w}), y_i)$$

- **Good:** if $|B| \ll n$, this is much faster to compute.
- **Bad:** it is a (random) approximation of the full gradient, noisy.

Stochastic Gradient Descent (SGD) for ERM

- Pick arbitrary starting point $\vec{x}^{(0)}$, **learning rate** parameter $\eta > 0$, batch size $m \ll n$.
- Until convergence, repeat:
  - Randomly sample a batch B of m training data points.
  - Compute stochastic gradient of f at $\vec{x}^{(i)}$:

$$\vec{g} = \sum_{i \in B} \frac{d}{d\vec{w}} \ell(H(\vec{x}^{(i)}; \vec{w}), y_i)$$

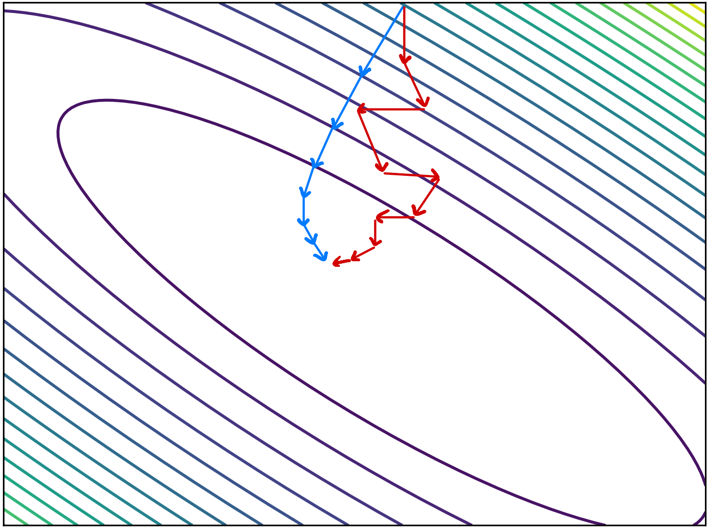
  - Update $\vec{w}^{(i+1)} = \vec{w}^{(i)} - \eta \vec{g}$

Idea

- In practice, a stochastic gradient often works well enough.
- It is better to take many noisy steps quickly than few exact steps slowly.

Batch Size

- Batch size m is a parameter of the algorithm.
- The larger m, the more reliable the stochastic gradient, but the more time it takes to compute.
- Extreme case when $m = 1$ will still work.



Usefulness of SGD

- SGD allows learning on **massive** data sets.
- Useful even when exact solutions available.
  - E.g., least squares regression / classification.

Example

- Trained on data set with $d = 20{,}000$ features and $n = 60{,}000$ examples.

- Solving the normal equations, $\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$:
  - about **3 minutes**
  - MSE: 6.7×10^{-7}

- Using SGD with $m = 16$ and $\eta = 0.0005$:
  - about **30 seconds**
  - MSE: 1.9×10^{-6}

DSC 140A

Probabilistic Modeling & Machine Learning

Lecture 3 | Part 6

From Theory to Practice

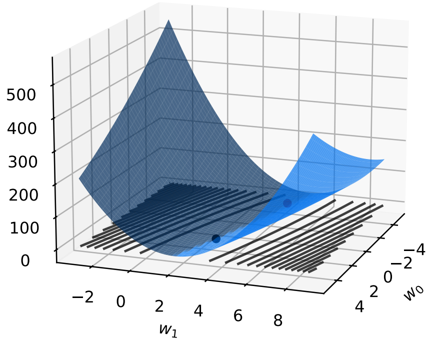
In Practice

- (~~S~~)GD is **heavily used** in machine learning.
- Can be used to solve many optimization problems.
- But it can be tricky to get working.

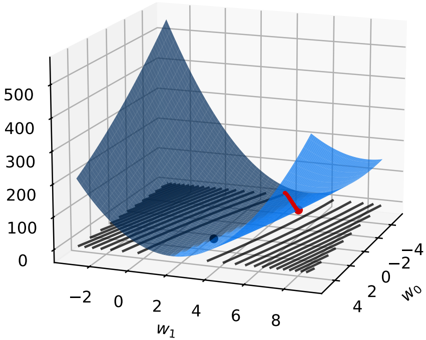
Learning Rate

- The learning rate has to be chosen carefully.
- If too large, the algorithm may **diverge**.
- If too small, the algorithm may **converge slowly**.

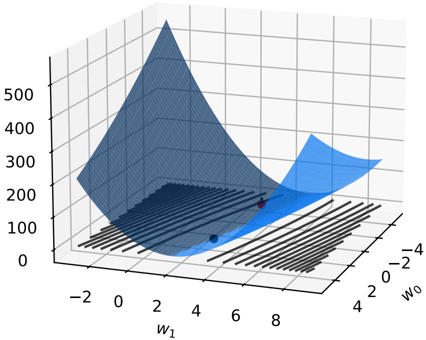
Diverging



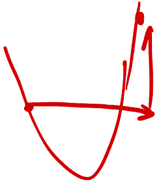
Diverging

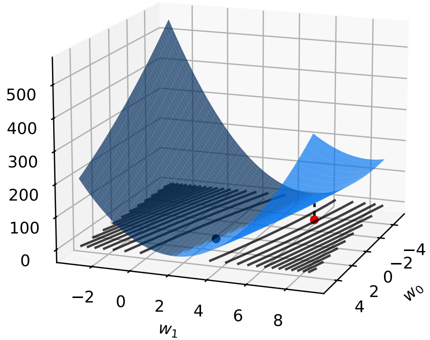


Diverging

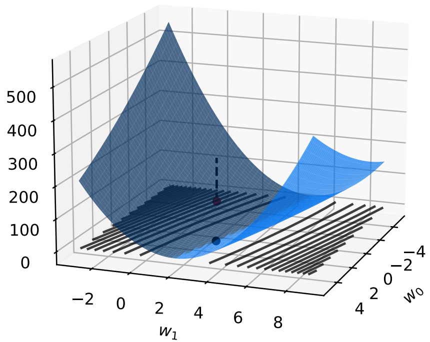


Diverging





Diverging

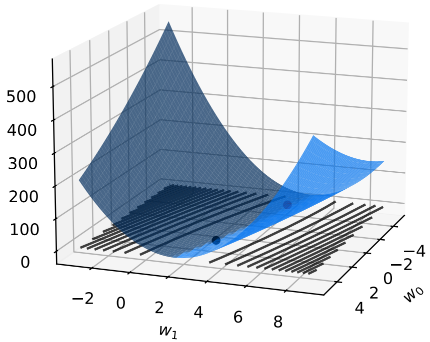


Diverging

- To diagnose, print $R(\vec{w})$ at each iteration.
- If it is increasing consistently, the algorithm is diverging.
- **Fix:** decrease the learning rate.
  - But not by too much! Then it may converge **too slowly**.

Problem

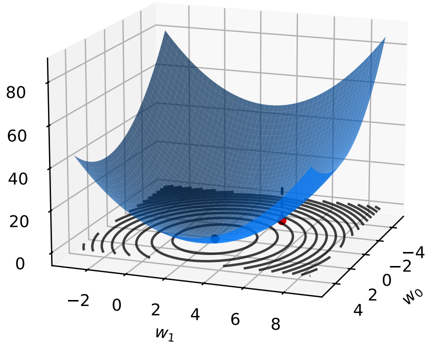
- When the contours are “long and skinny,” you will be forced to pick a very small learning rate.

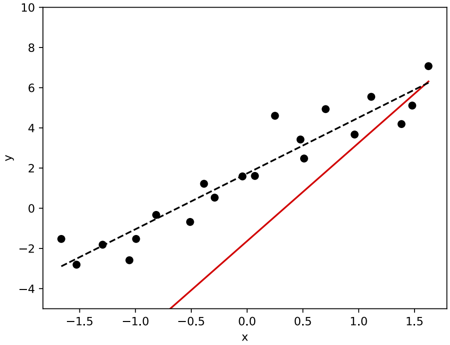


A **Fix**

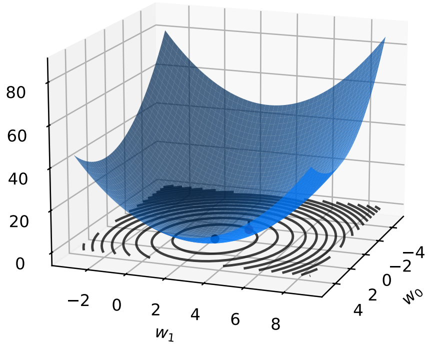
- Scaling (standardizing) the features can help.
- This makes the contours more circular.
- Doesn't change the prediction!

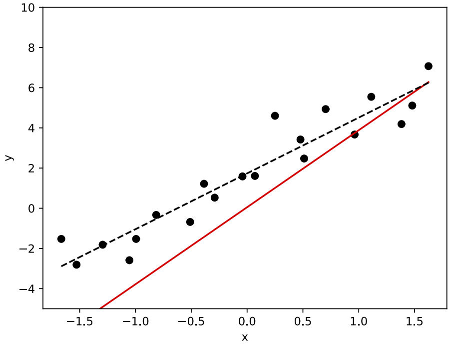
Iteration #1



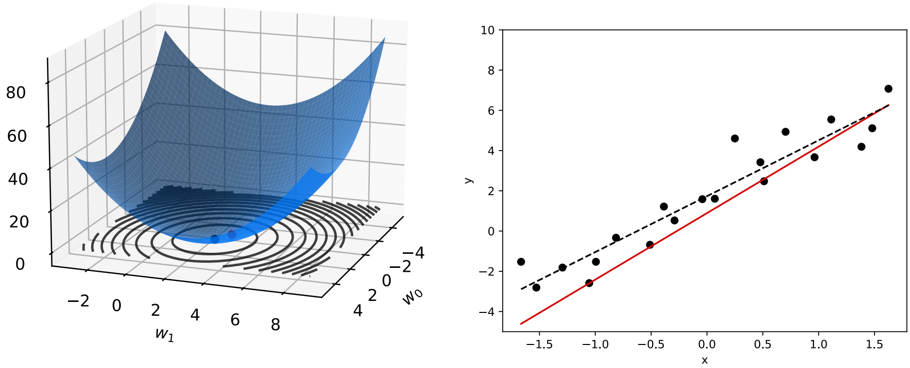


Iteration #2

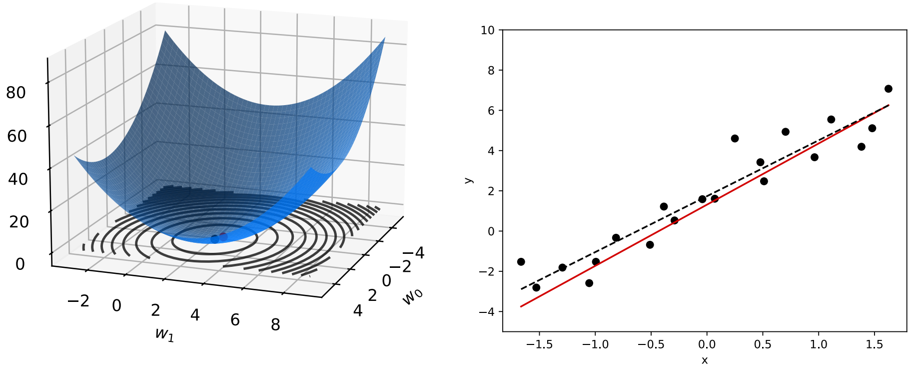




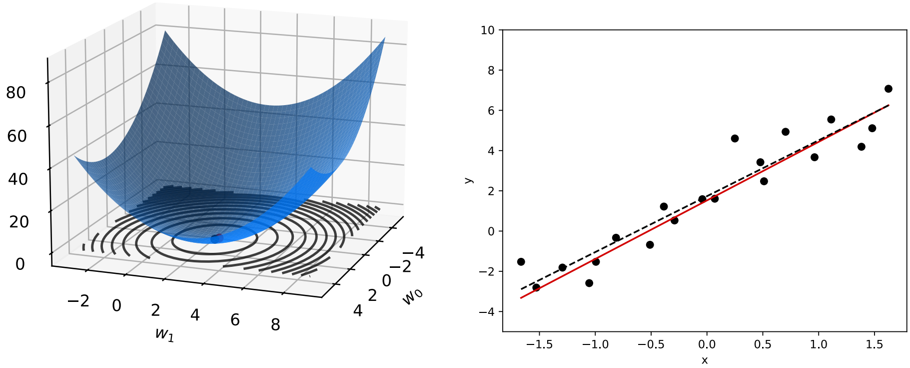
Iteration #3



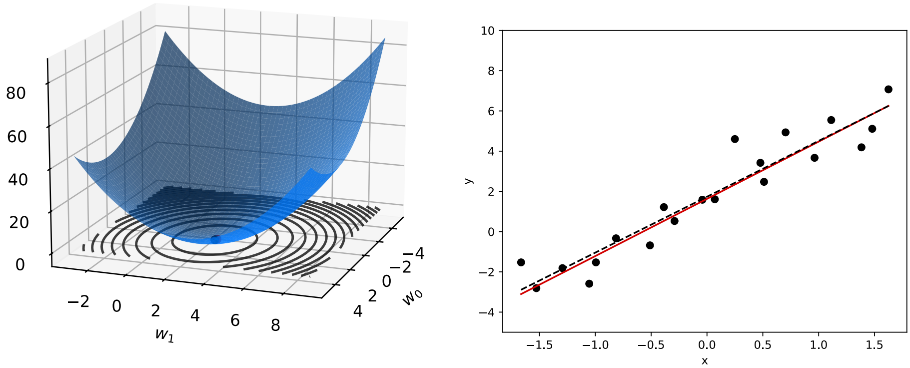
Iteration #4



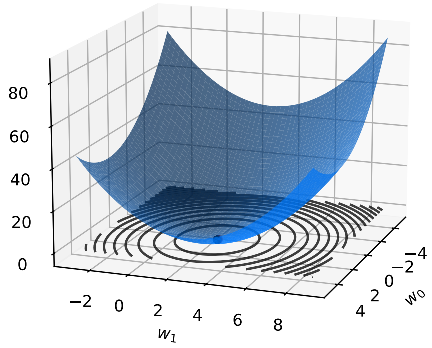
Iteration #5

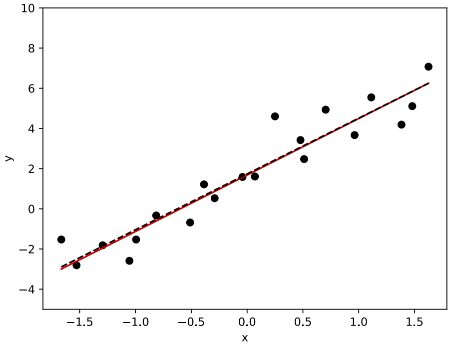


Iteration #6



Iteration #7





Next Time

- How do we minimize the risk with respect to absolute loss?
- When is gradient descent guaranteed to converge?