Week 4
Optimization

Nebius Academy

The objectives for today

1. How to optimize a loss function

2. Regularization

3. Dimension reduction

Gradient optimization

A quick reminder about loss functions

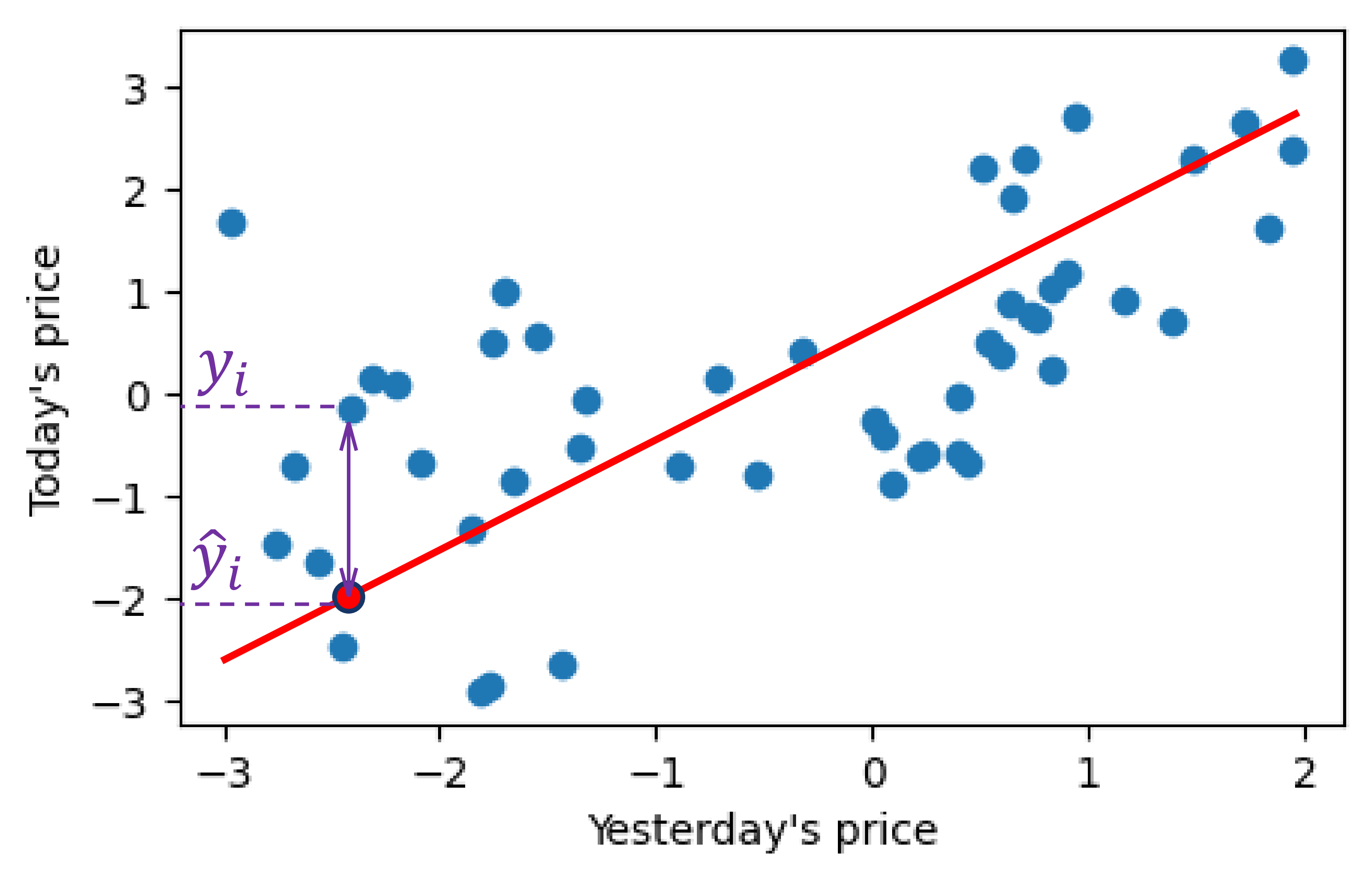
Regression loss

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y_i, \hat{y}_i)$$

y_i – true value
$\hat{y}_i$ – predicted value

MSE:

$$\mathcal{L}(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$



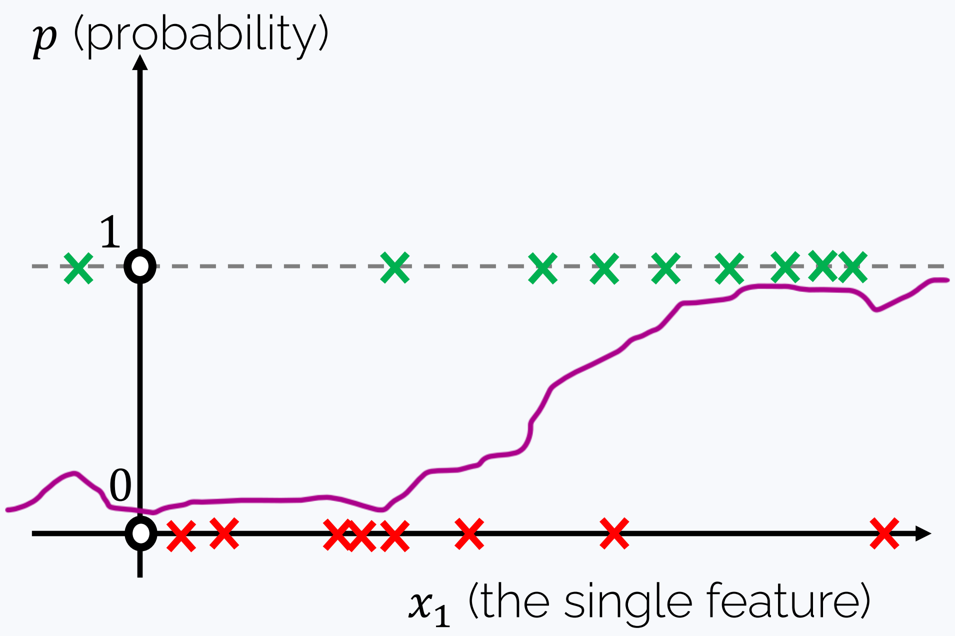
A quick reminder about loss functions

Classification: cross-entropy

$$\mathcal{L}(y_i, \hat{p}_i) = -y_i \log \hat{p}_i - (1 - y_i) \log(1 - \hat{p}_i)$$

$$l(y_i, \hat{p}_i) = \begin{cases} -\log(1 - \hat{p}_i), & if\ y_i = 0 \\ -\log \hat{p}_i, & if\ y_i = 1 \end{cases}$$

y_i – true class labels
$\hat{p}_i$ – predicted probabilities



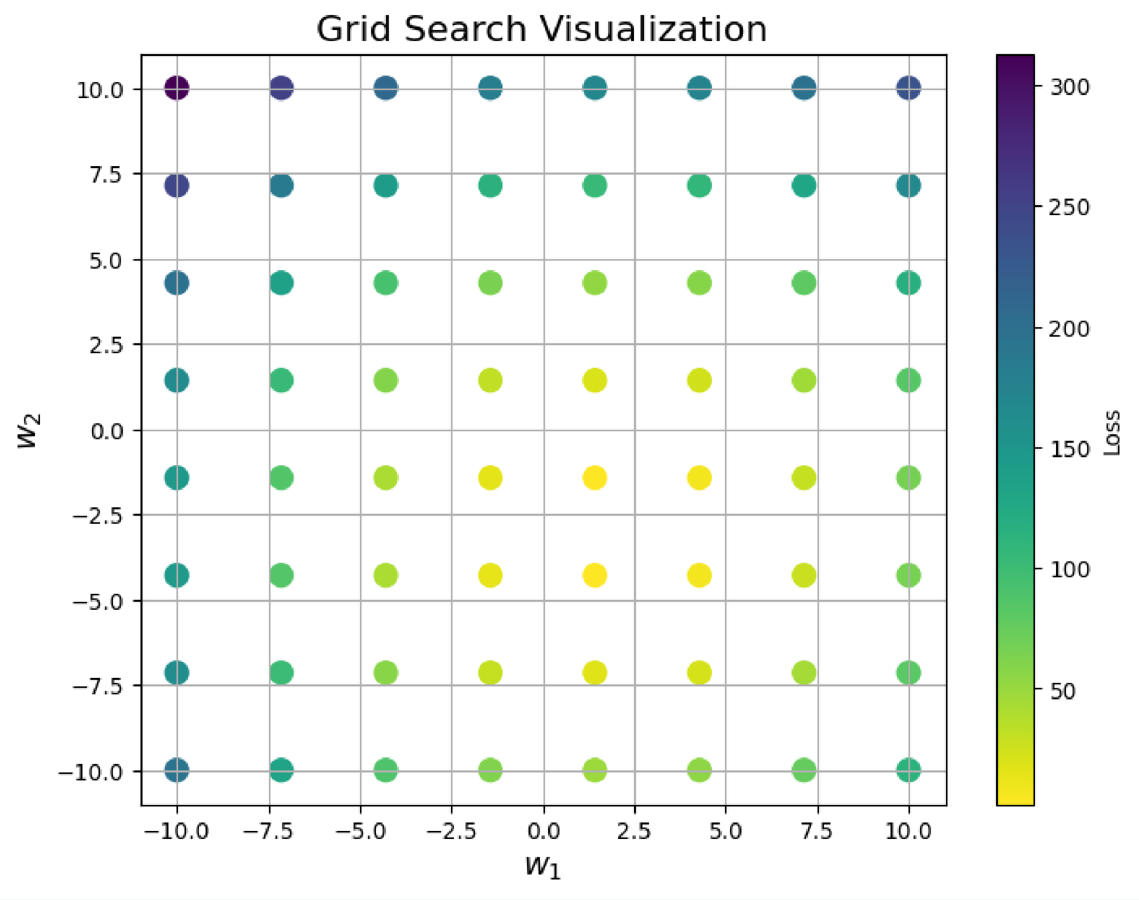
The next step – how to find the optimal w

We need to find the parameters w, which minimize the loss.

The next step – how to find the optimal w

We need to find the parameters w, which minimize the loss.

Why not grid search?



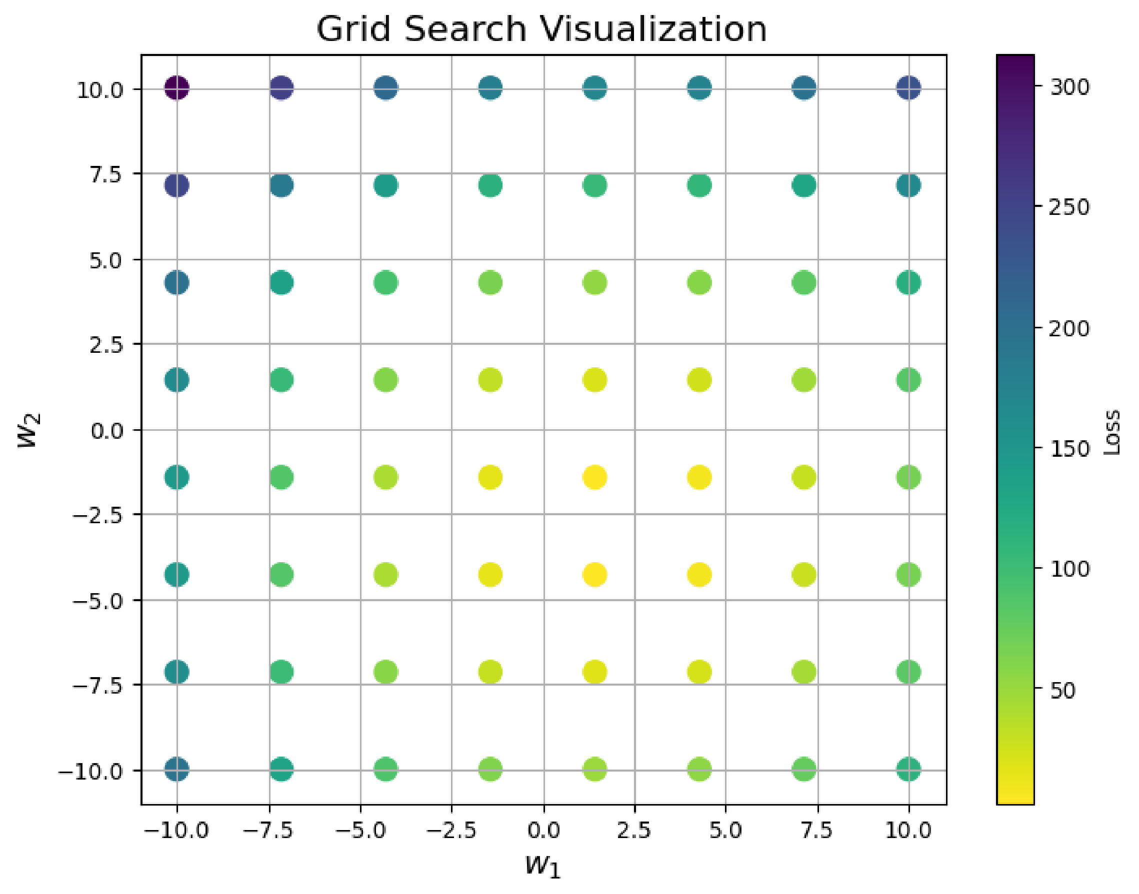
The next step – how to find the optimal w

We need to find the parameters w, which minimize the loss.

Why not grid search?

Imagine:
- 20 parameters
- -2..2 with 0.1 step (41 values)
- Training set size: 200
- Test set size: 100

We need to calculate $\mathcal{L}$ many times:
$41^{20} \cdot (200 + 100)$



Gradients

What is a gradient

Gradient of $f(w_1, \dots, w_D)$ at a point $\mathrm{w}_0 = (w_{01}, \dots, w_{0D})$ is the vector

$$[\nabla_w f](w_0) = \left(\frac{\partial f}{\partial w_1}, \dots, \frac{\partial f}{\partial w_D}\right)\Bigg|_{w=w_0}$$

Example:

$$f(w_1, \dots, w_D) = \sin(w_1) + \exp(w_1 w_2)$$

$$\frac{\partial f}{\partial w_1} = \cos(w_1) + w_2 \exp(w_1 w_2)$$

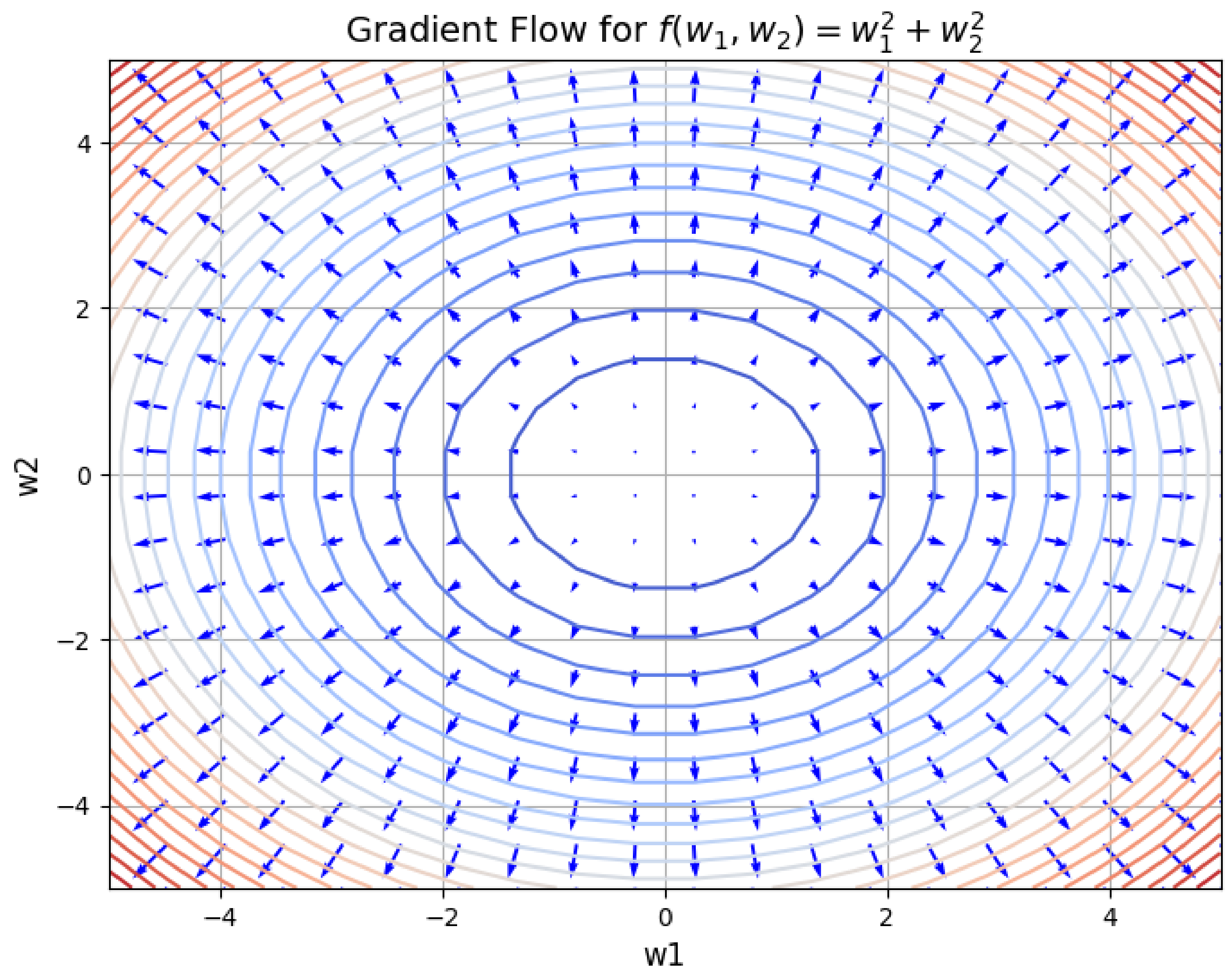
$$\frac{\partial f}{\partial w_2} = w_1 \exp(w_1 w_2)$$

$$[\nabla_w f](0, 1) = (2,0)$$

What is a gradient

Gradient of $f(w_1, \ldots, w_D)$ at a point $\mathrm{w}_0 = (w_{01}, \ldots, w_{0D})$ is the vector

$$[\nabla_w f](w_0) = \left.\left(\frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_D}\right)\right|_{w=w_0}$$

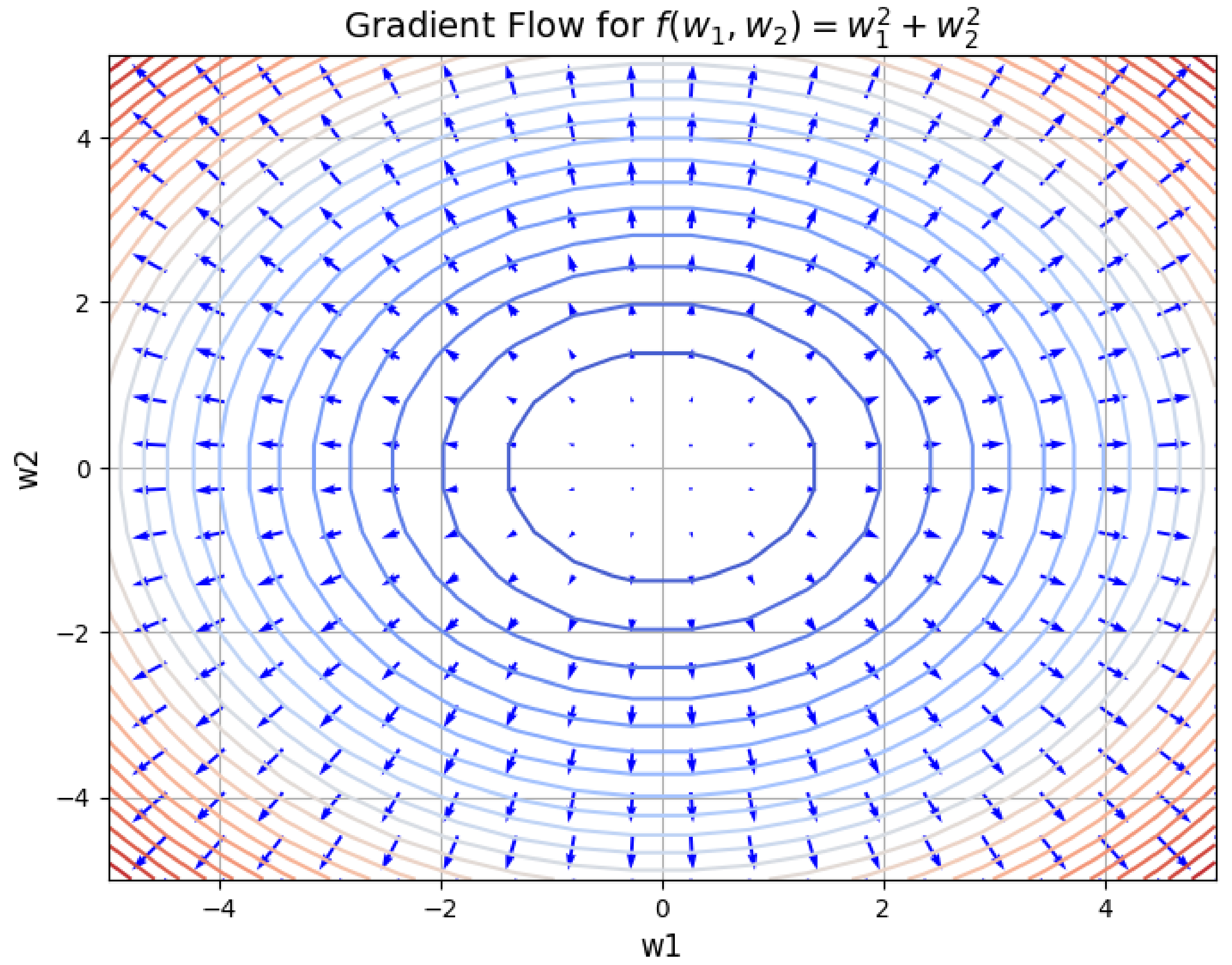


What is a gradient

Gradient of $f(w_1, \ldots, w_D)$ at a point $\mathrm{w}_0 = (w_{01}, \ldots, w_{0D})$ is the vector

$$[\nabla_w f](w_0) = \left(\frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_D}\right)\bigg|_{w=w_0}$$

It is the **direction in which the function increases the fastest**



What is a gradient

Gradient of $f(w_1, \dots, w_D)$ at a point $\mathrm{w}_0 = (w_{01}, \dots, w_{0D})$ is the vector

$$[\nabla_w f](w_0) = \left(\frac{\partial f}{\partial w_1}, \dots, \frac{\partial f}{\partial w_D}\right)\bigg|_{w=w_0}$$

It is the **direction in which the function increases the fastest**

$$f(w_0 + h) = f(w_0) + \langle \nabla_w f(w_0), h \rangle + \ \dots$$

If $\|h\| = d$:

$$\langle \nabla_w f(w_0), h \rangle = \|\nabla_w f(w_0)\| \cdot d \cdot \cos \angle(\nabla_w f(w_0), h)$$

The maximum is when h and $\nabla_w f(w_0)$ have the same direction.

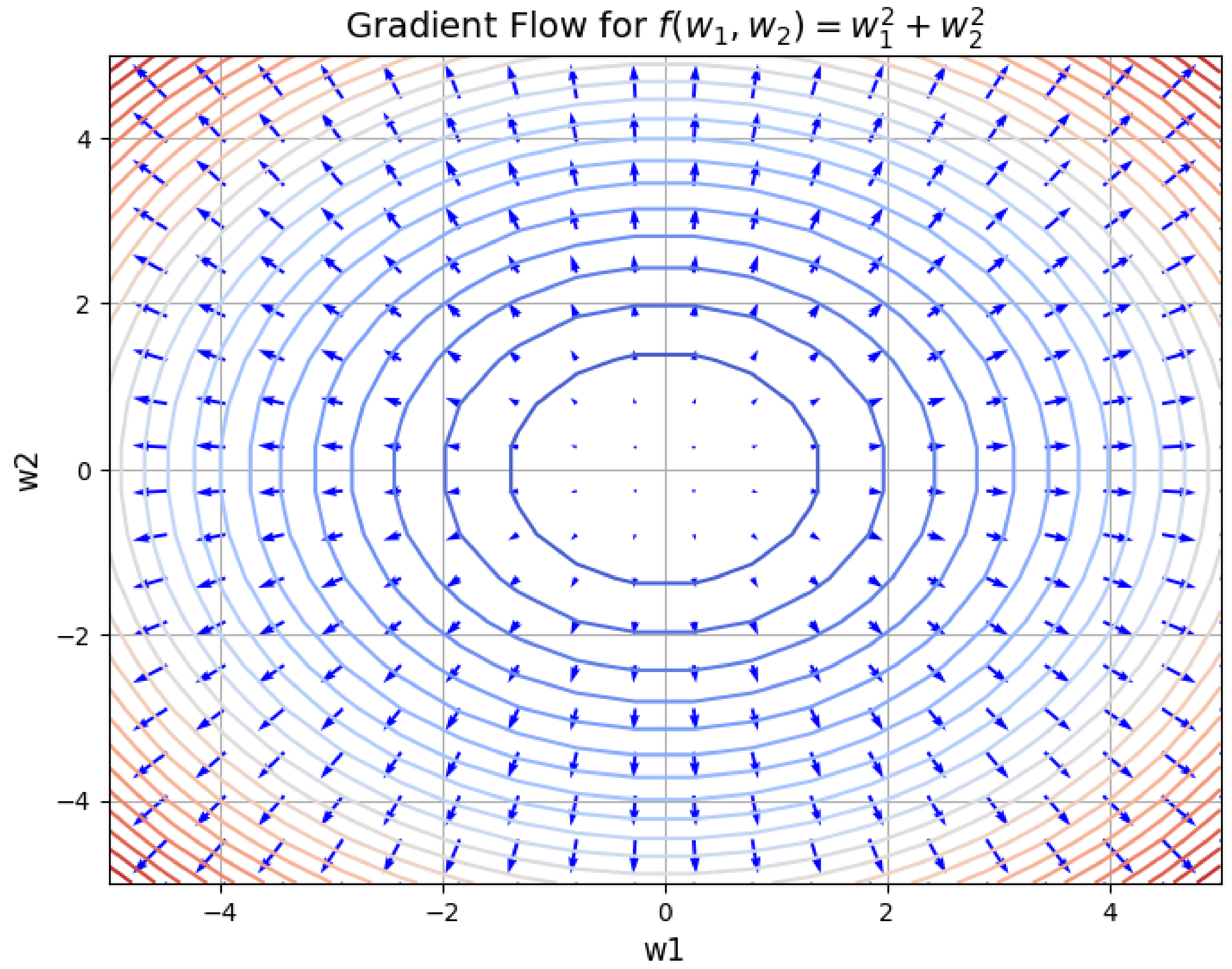
An obvious application

Loss minimum is where the gradient is zero

Solve

$$\frac{\partial f}{\partial w_1} = \cdots = \frac{\partial f}{\partial w_D} = 0,$$

get the answer.



How it works

Loss minimum is where the gradient is zero

Solve

$$\frac{\partial f}{\partial w_1} = \cdots = \frac{\partial f}{\partial w_D} = 0,$$

get the answer.

Linear regression

Solve

$$\mathcal{L}(X, w, y) = \sum_{i=1}^{N} (y_i - x_i w^T)^2$$

$$= (y - Xw^T)^T (y - Xw)$$

get the answer:

$$w^T = (X^T X)^{-1} X^T y$$

(see much later)

How it works – Linear Regression

$$\mathcal{L}(X, w, y) = \sum_{i=1}^{N} (y_i - x_i w^T)^2 = \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{D} x_{ij} w_j \right)^2$$

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^{N} 2(y_i - x_i w^T) \cdot (-x_{ij}) =$$

$$= -2 \sum_{i=1}^{N} (y - X w^T)_i \cdot x_{ij} = [-2(y - X w^T)^T X]_j$$

Linear regression

Good news

Loss minimum is where the gradient is zero

We can solve it!

$$-2(y - Xw^T)^{\boxed{T}}X = 0$$

$$-2(y^T - wX^T)X = 0$$

Multiply

$$-2y^TX + 2wX^TX = 0$$

$$wX^TX = y^TX$$

Transpose

$$X^TXw^T = X^Ty$$

$(X^TX)^{-1} \cdot$

$$w^T = (X^TX)^{-1}X^Ty$$

$$w = y^TX(X^TX)^{-1}$$

Bad news

Loss minimum is where the gradient is zero

Linear regression is almost the single ML task that has closed-form solution (((

(And to make things even worse, this formula is numerically unstable)

Linear regression

Solve

$$-2(y - Xw^T)^T X = 0$$

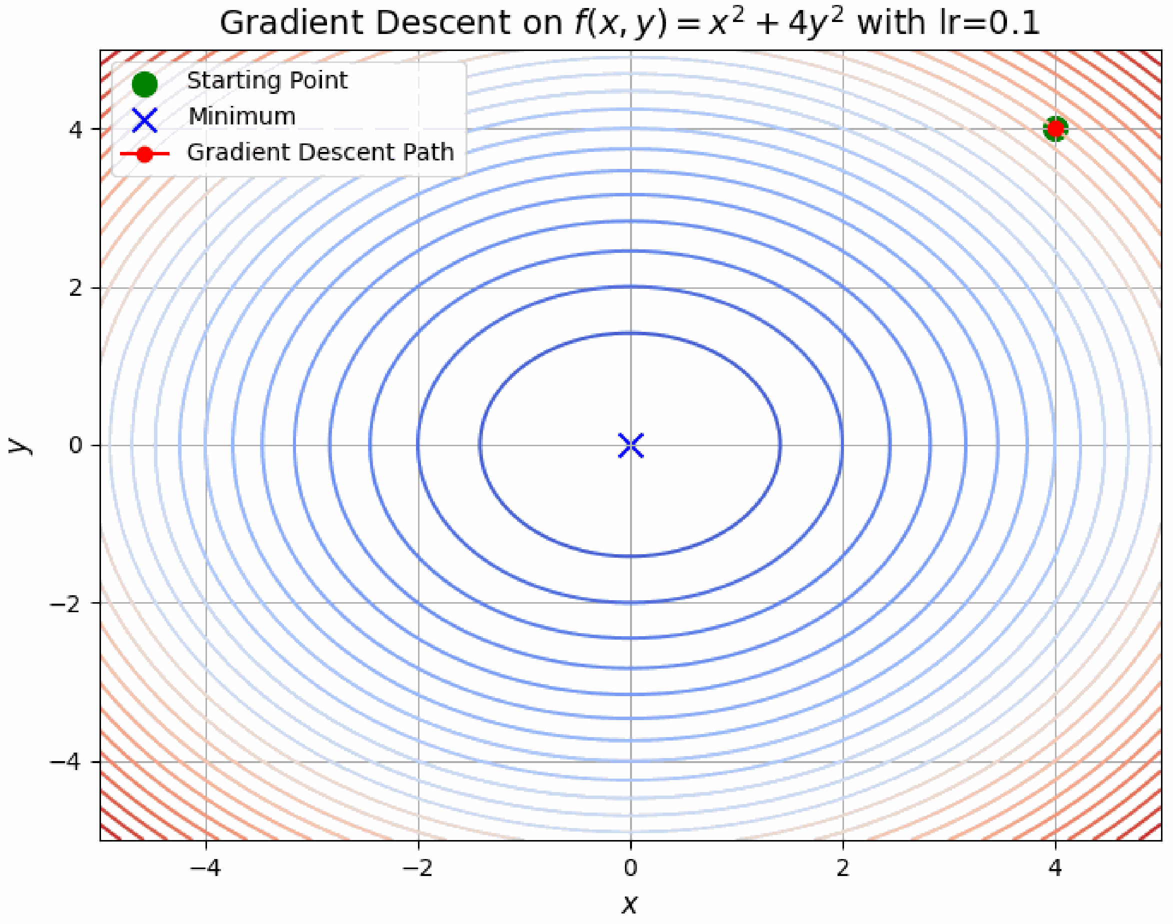
Get

$$w^T = (X^T X)^{-1} X^T y$$

Gradient descent

Gradient is the direction of the steepest increase ⇒ **Anti-gradient $-\nabla f$ is the direction of the steepest decrease**

Just go along the anti-gradient to reach the minimum!

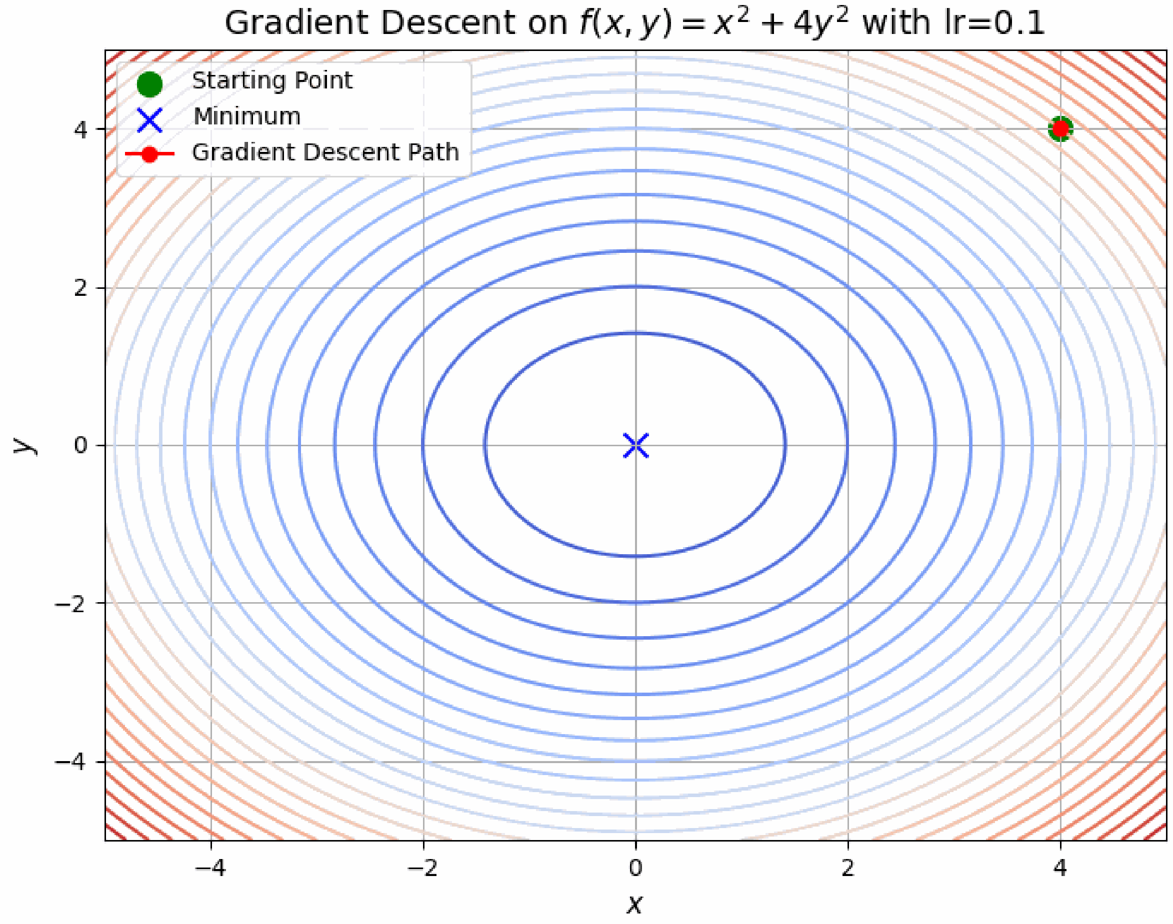


Gradient descent

Math formulation:

Given: f, ∇f, w_0, **max_steps**, α – learning rate

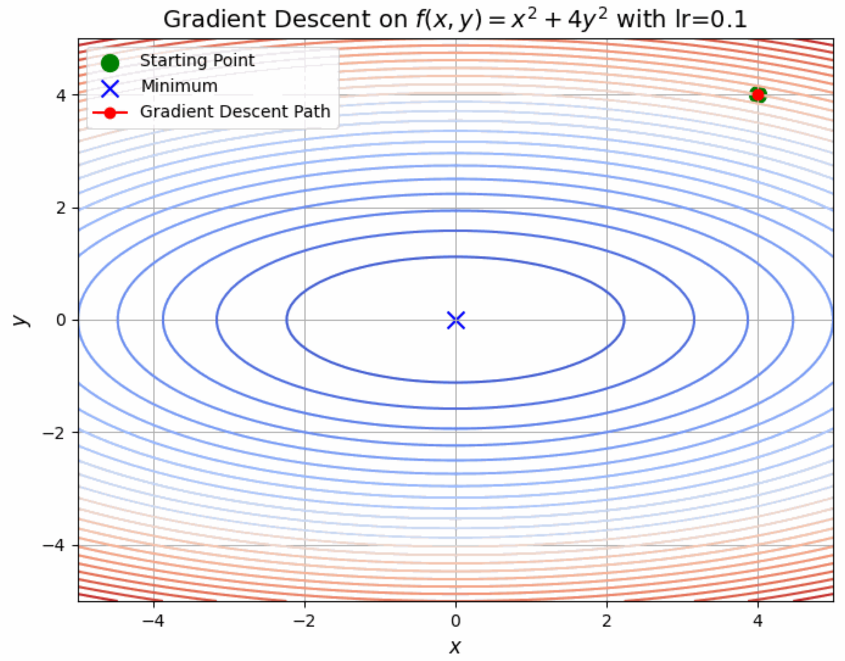
1. Initialize w with w_0
2. Perform several anti-gradient steps:
$$w_{m+1} = w_m - \alpha \cdot \nabla_w f(w_m)$$
3. Finish when reached **max_steps** or converged

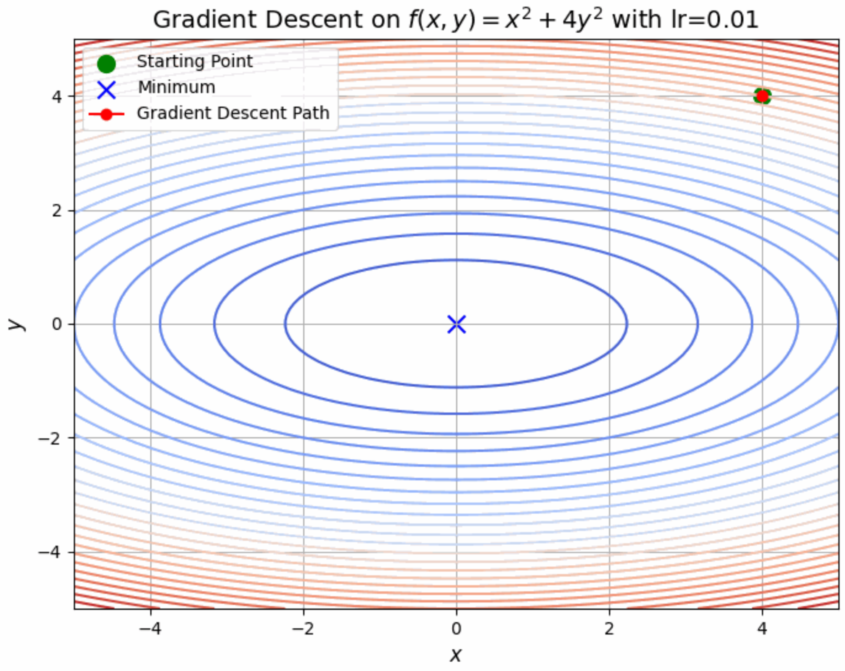


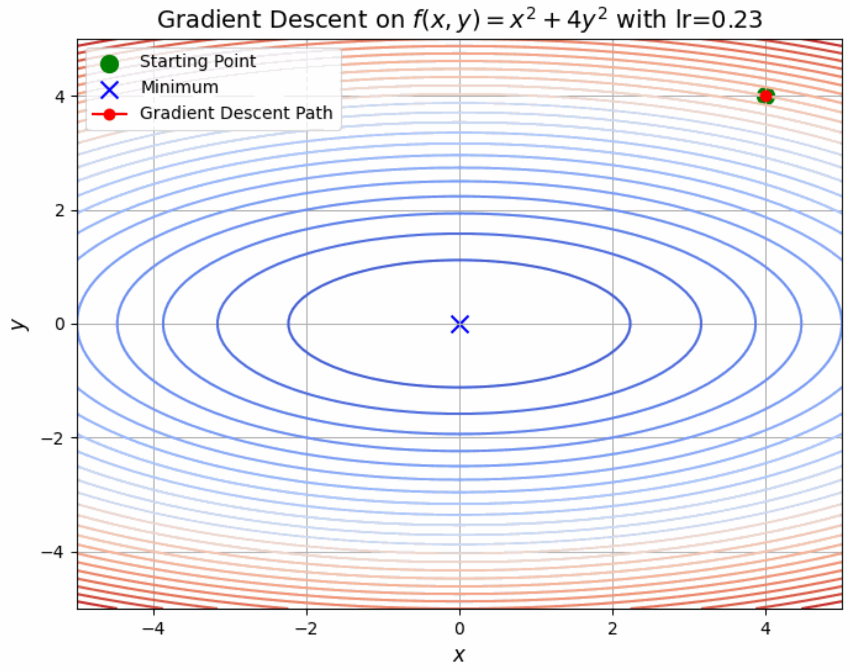
Learning rate is important

If it's too little, it won't converge;

If it's too big, it will diverge.







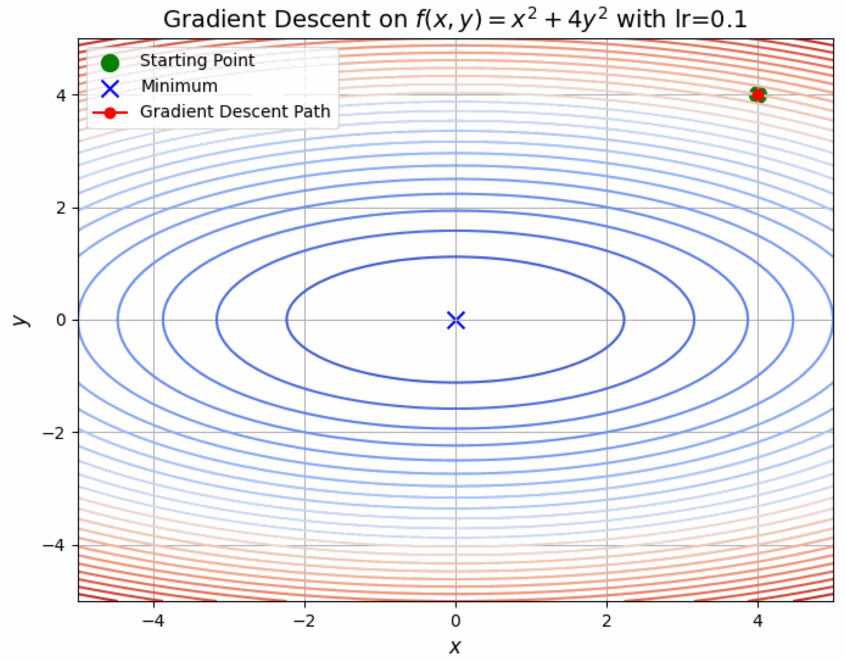
Learning rate is important

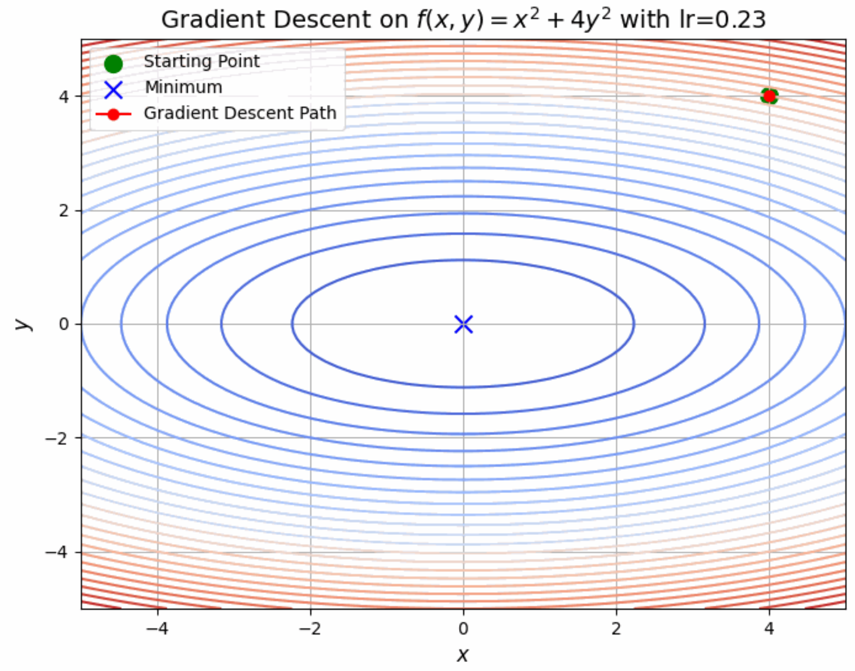
If it's too little, it won't converge;

If it's too big, it will diverge.

Learning rate is a **hyperparameter** of optimization tuned on a logarithmic scale. The choice would be between: 1e-5, 1e-4, 1e-3, 1e-2

In practice, learning rate decay is often used.





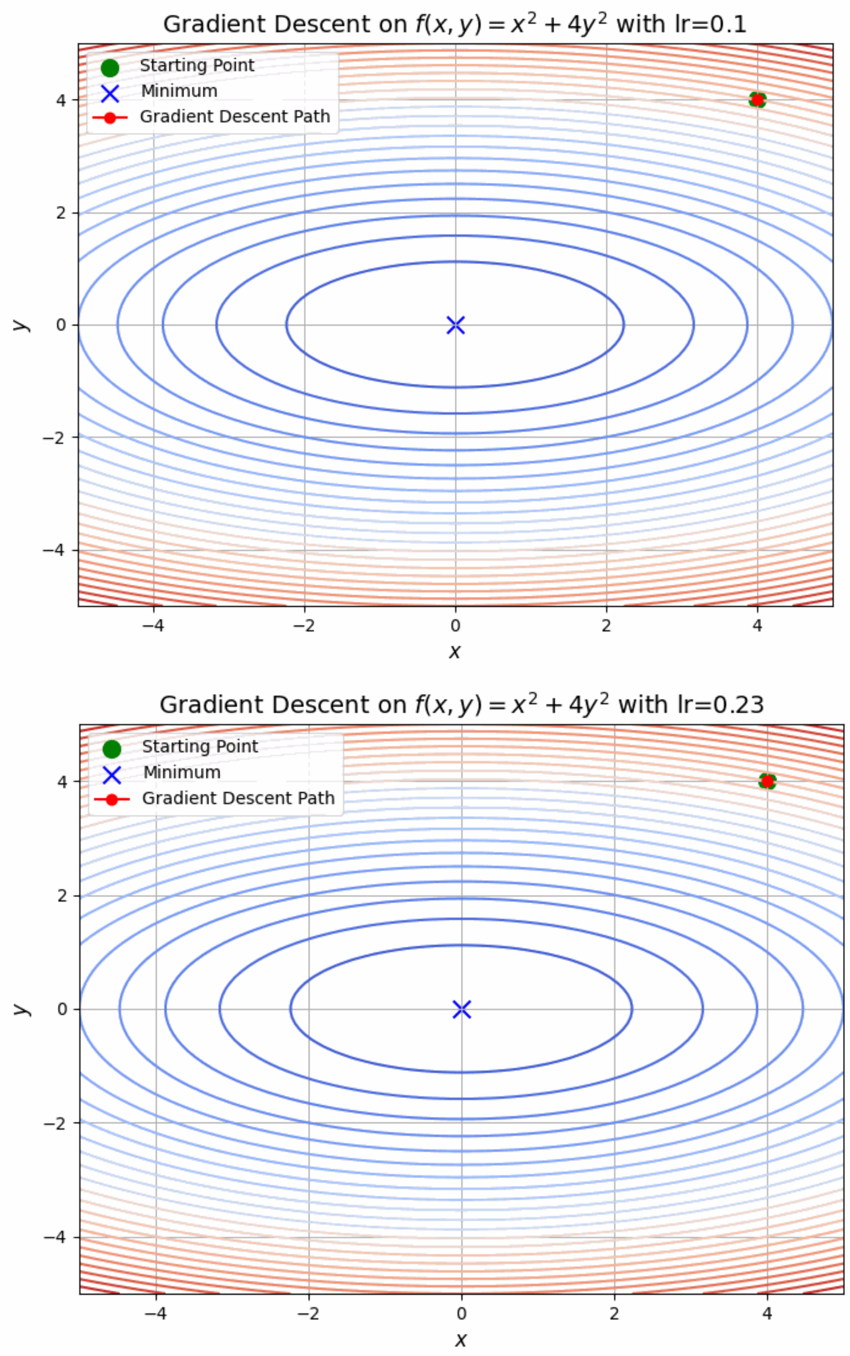
About elongated loss basin

$$\mathcal{L}(X, w, y) = \sum_{i=1}^{N} (y_i - x_i w^T)^2 =$$

$$= \|y - Xw^T\|^2 = (y - Xw^T)^T (y - Xw^T) =$$

$$= (y^T - wX^T)(y - Xw^T) =$$

$$= wX^T X w^T + linear\ terms$$



About elongated loss basin

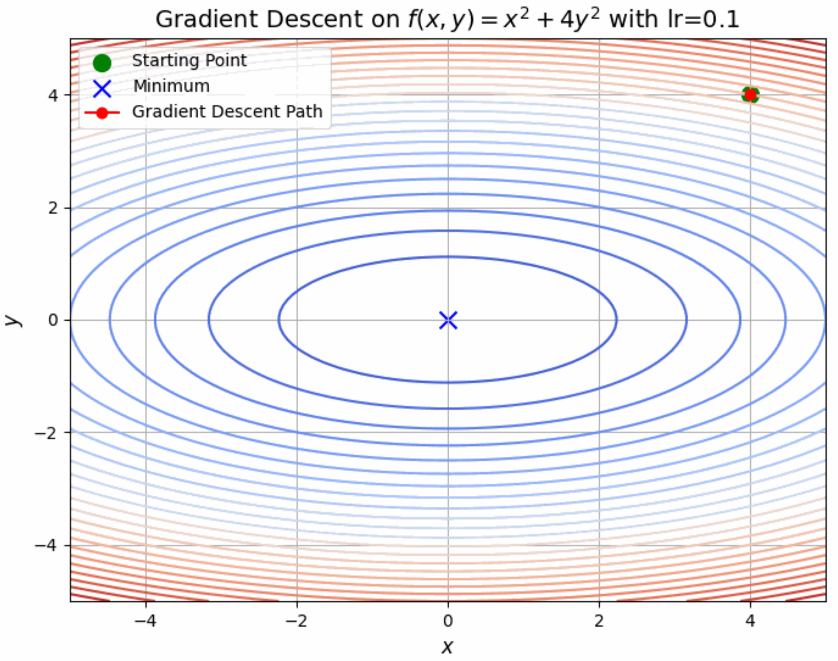
$$\mathcal{L}(X, w, y) = \sum_{i=1}^{N} (y_i - x_i w^T)^2 =$$

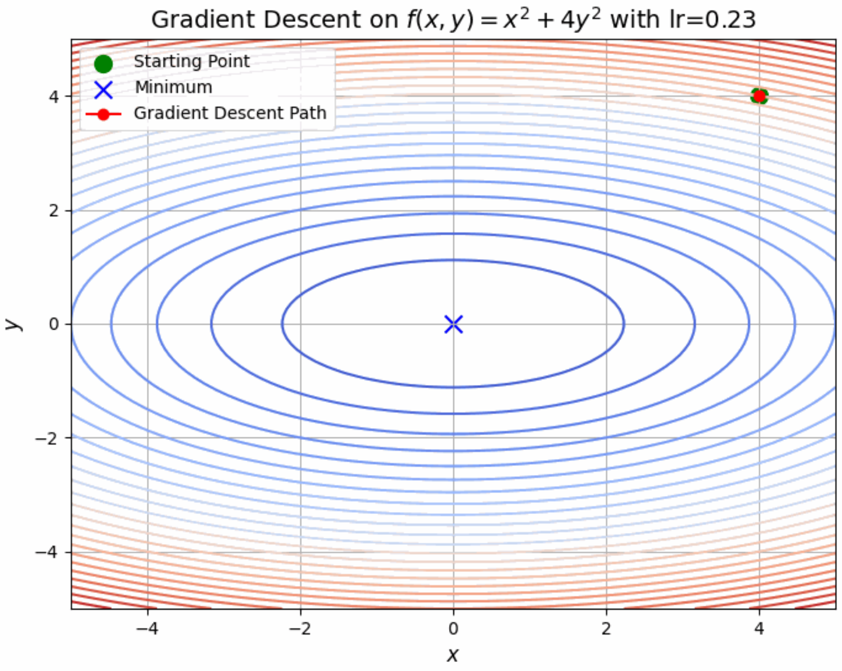
$$= wX^T X w^T + linear\ terms$$

If every feature has $\boldsymbol{mean = 0}$, then

$$(X^T X)_{ij} \sim Cov(feature_i, feature_j)$$

$$(X^T X)_{ii} \sim \mathbb{V}(feature_i)$$





About elongated loss basin

$$(X^T X)_{ij} =$$

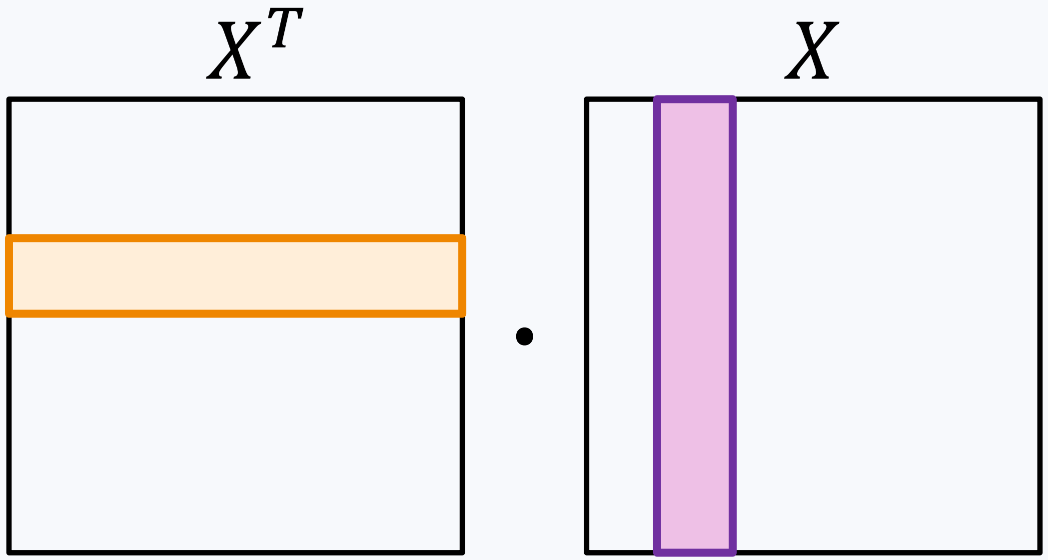
$$\mathcal{L}(X, w, y) = \sum_{i=1}^{N} (y_i - x_i w^T)^2 =$$

$$= wX^T X w^T + linear\ terms$$

If every feature has $\boldsymbol{mean} = \mathbf{0}$, then

$$(X^T X)_{ij} \sim Cov\big(feature_i, feature_j\big)$$

$$(X^T X)_{ii} \sim \mathbb{V}(feature_i)$$



i-th row of X^T
i-th column of X

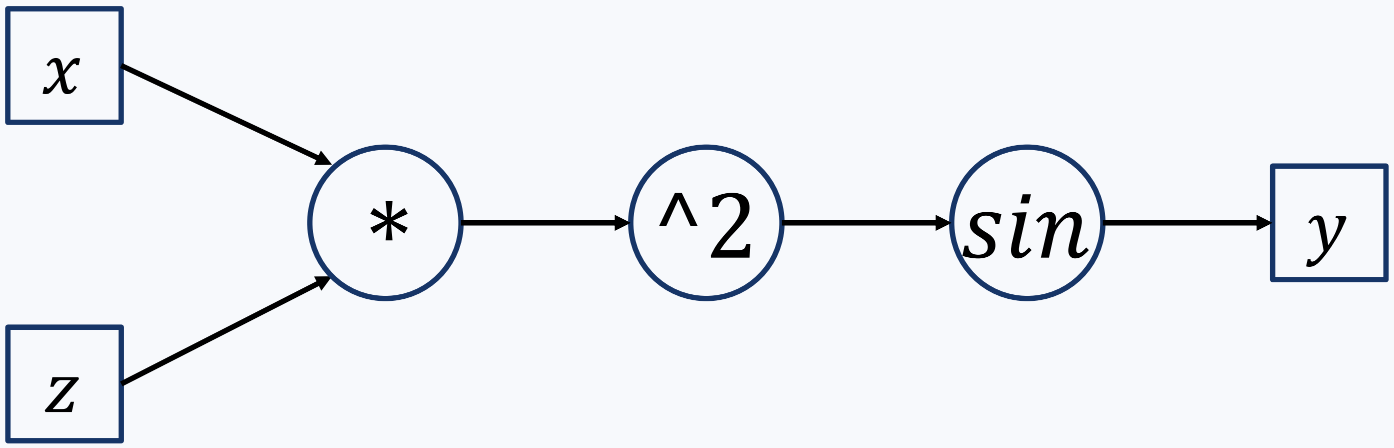
j-th column of X

Autodiff

Automating differentiation

Every expression is a computational graph

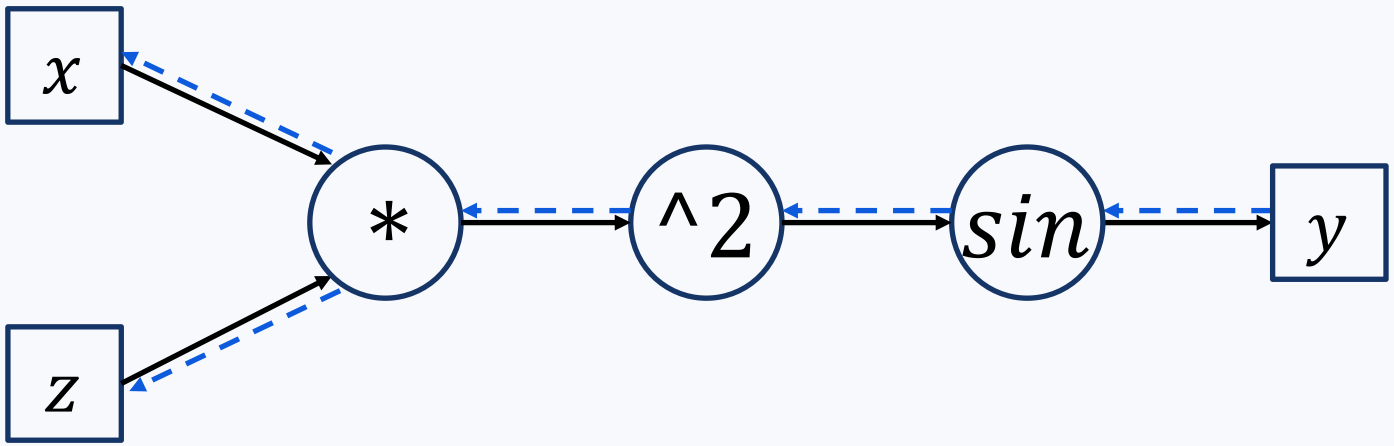
Example. $f(x, z) = \sin(x \cdot z)^2$



Automating differentiation

Every expression is a computational graph

Example. $f(x, z) = \sin(x \cdot z)^2$

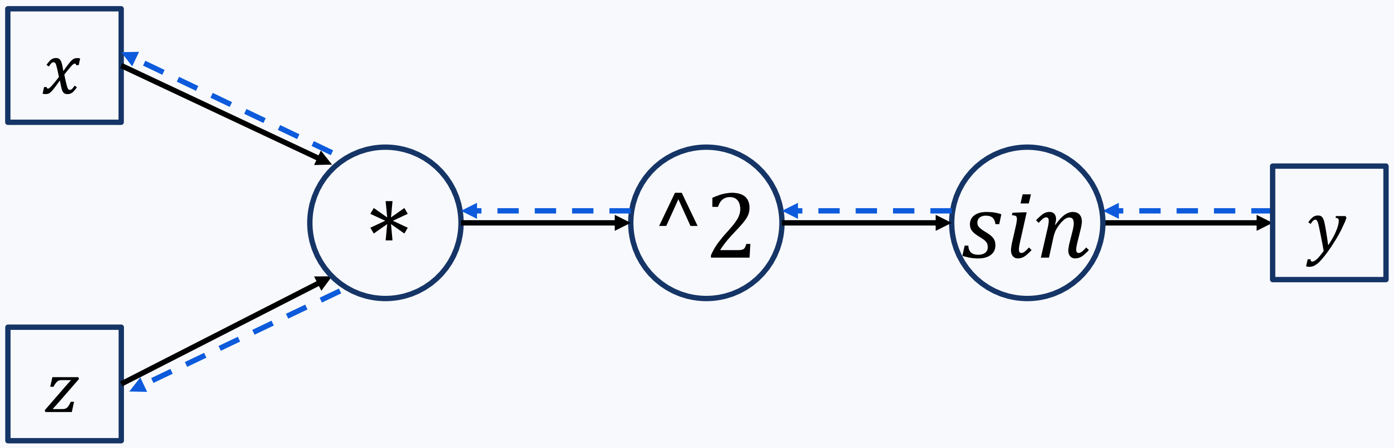


Derivatives flow backward

Automating differentiation

Every expression is a computational graph

Example. $f(x, z) = \sin(x \cdot z)^2$



Derivatives flow backward

$$\frac{\partial}{\partial x} \sin(x \cdot z)^2 \bigg|_{\substack{x=1 \\ z=2}}$$

$$= \cos\left((x \cdot z)^2 \Big|_{\substack{x=1 \\ z=2}} \right)$$

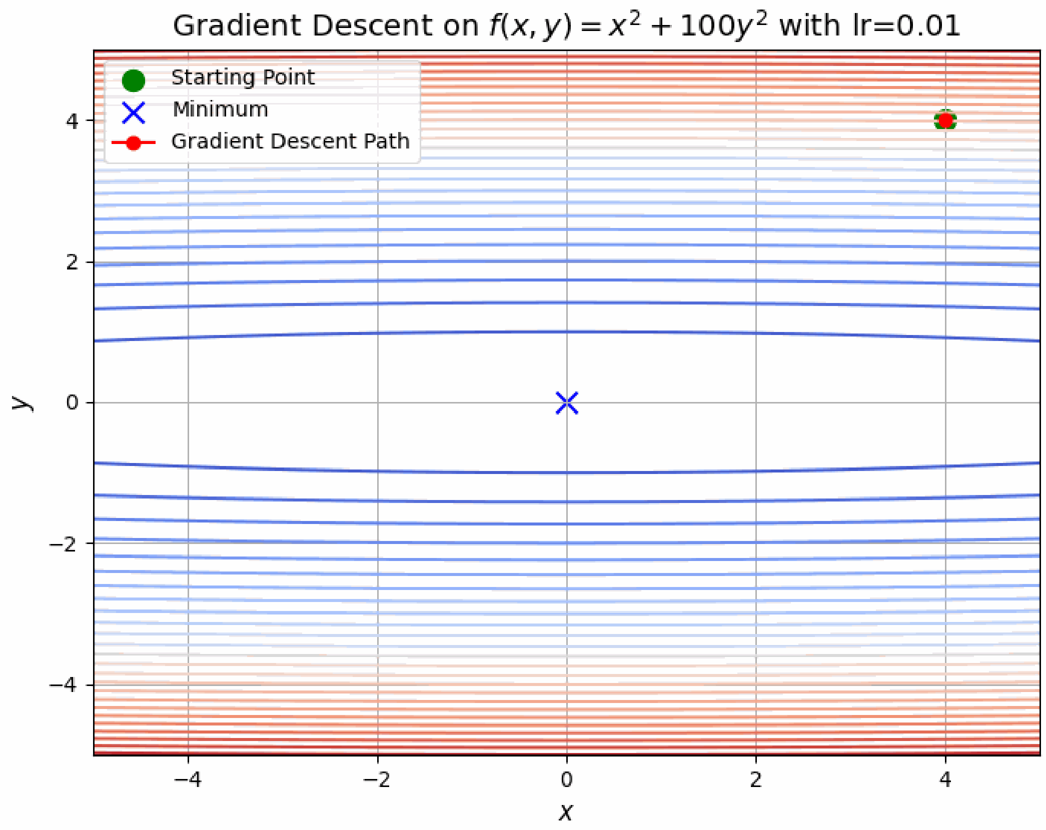
$$\cdot\, 2 \left((x \cdot z) \Big|_{\substack{x=1 \\ z=2}} \right)$$

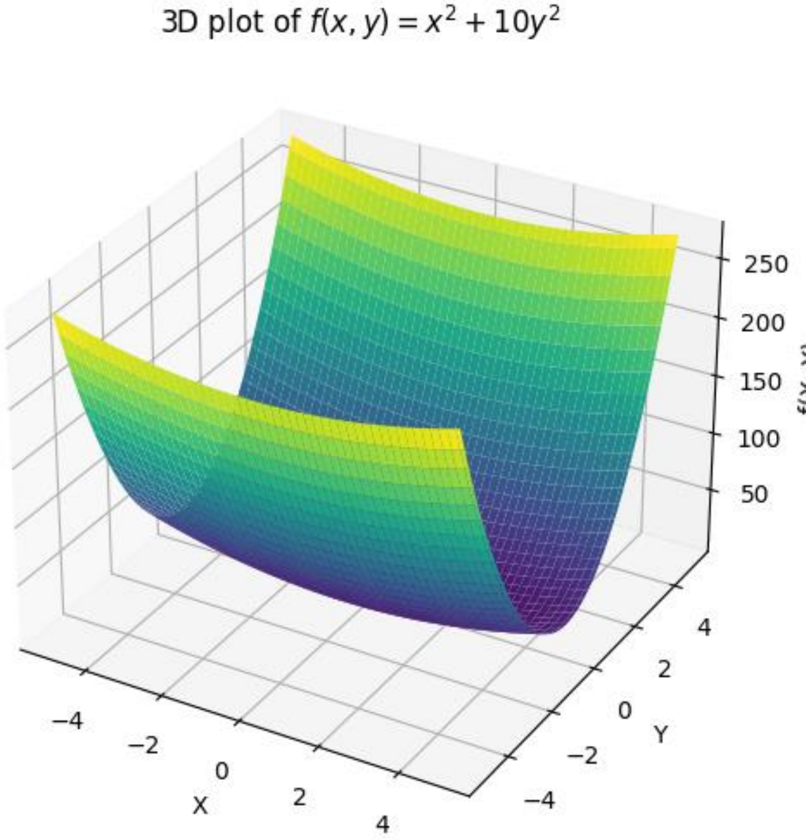
$$\cdot\, z$$

Why gradient optimization is tricky?

Badly conditioned problems

Very steep gradients are bad

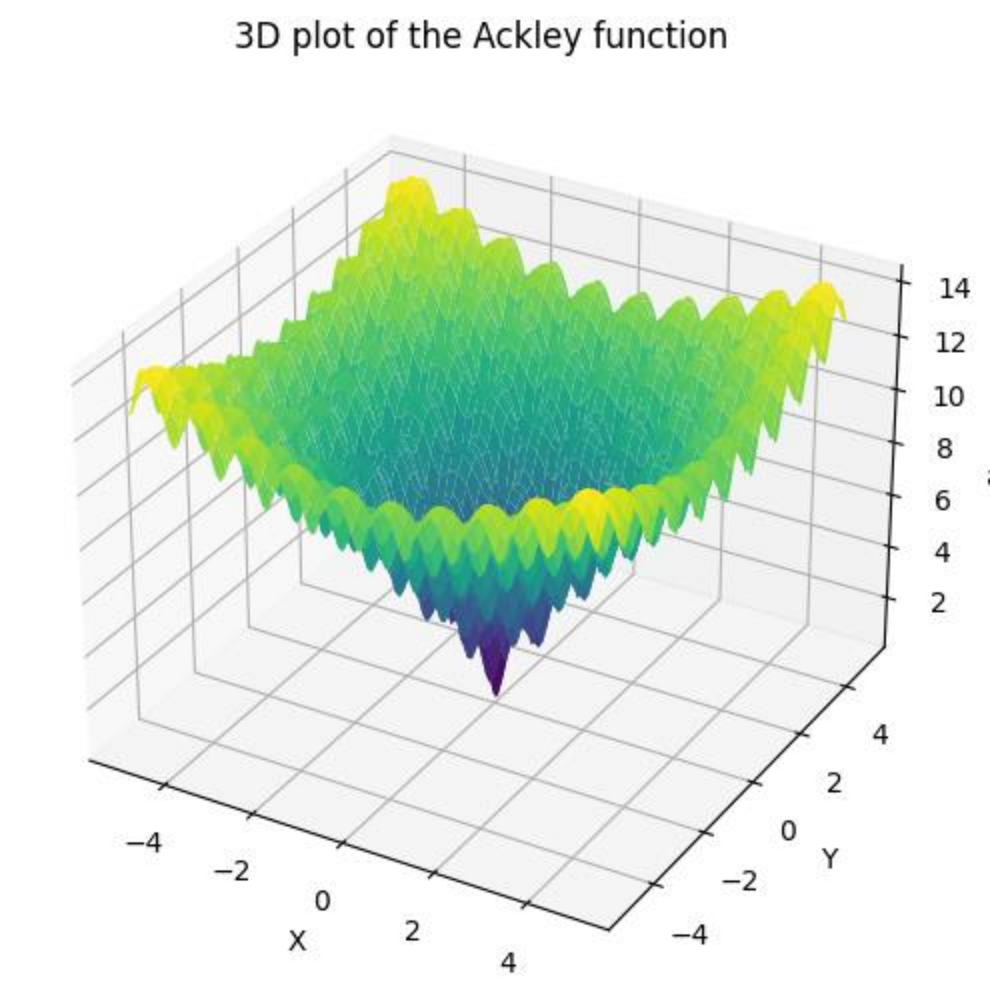


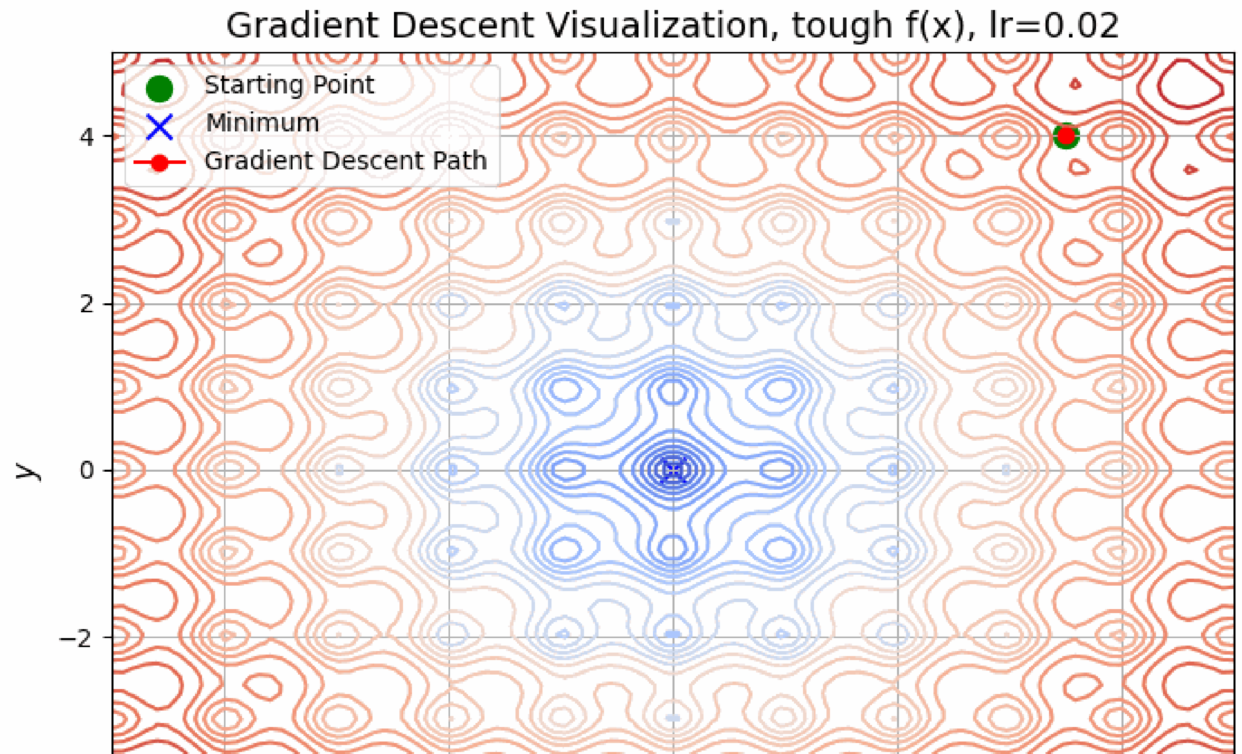


Having slopes of very different magnitude along different directions is bad

Local minima

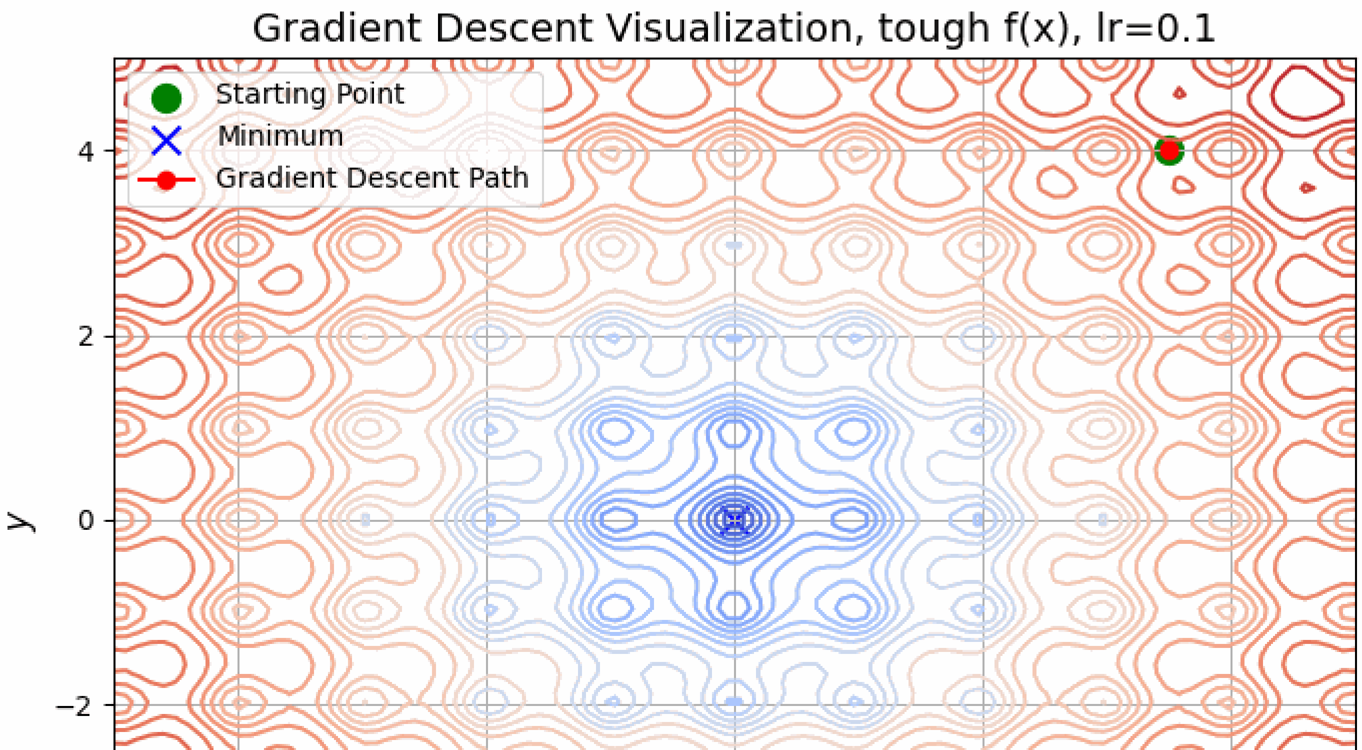
Gradient descent may get stuck in local optima

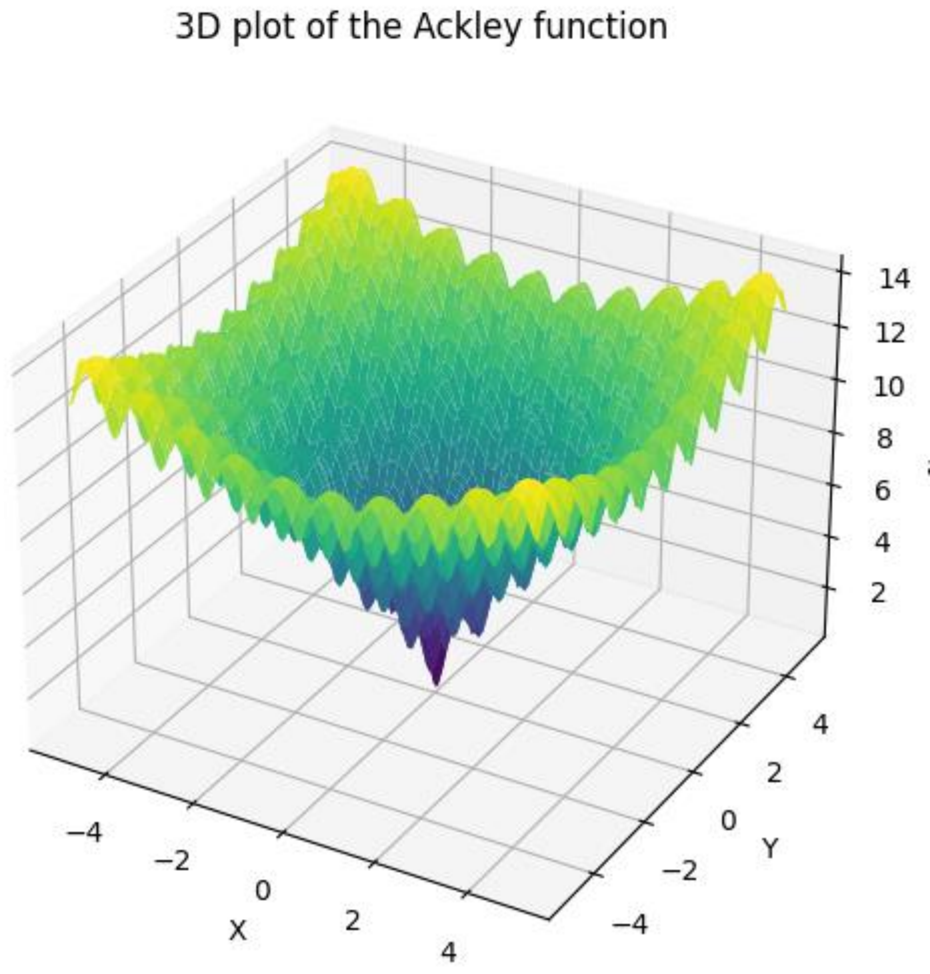




Local minima

Gradient descent may get stuck in local optima





Gradient descent in Machine Learning

$$\mathcal{L}(y, X, w) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y_i, x_i, w) \rightsquigarrow$$

$$\nabla_w \mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\nabla_w \mathcal{L}(y_i, x_i, w) = \frac{1}{N}(-2y^T X + 2wX^T X)$$

See any problems?

Gradient descent in Machine Learning

Why gradient descent isn't used:

- Avoid manifesting large matrices in the memory
- If your dataset is huge, we don't feedback from the system anytime soon.

Stochastic gradient descent

Stochastic gradient descent

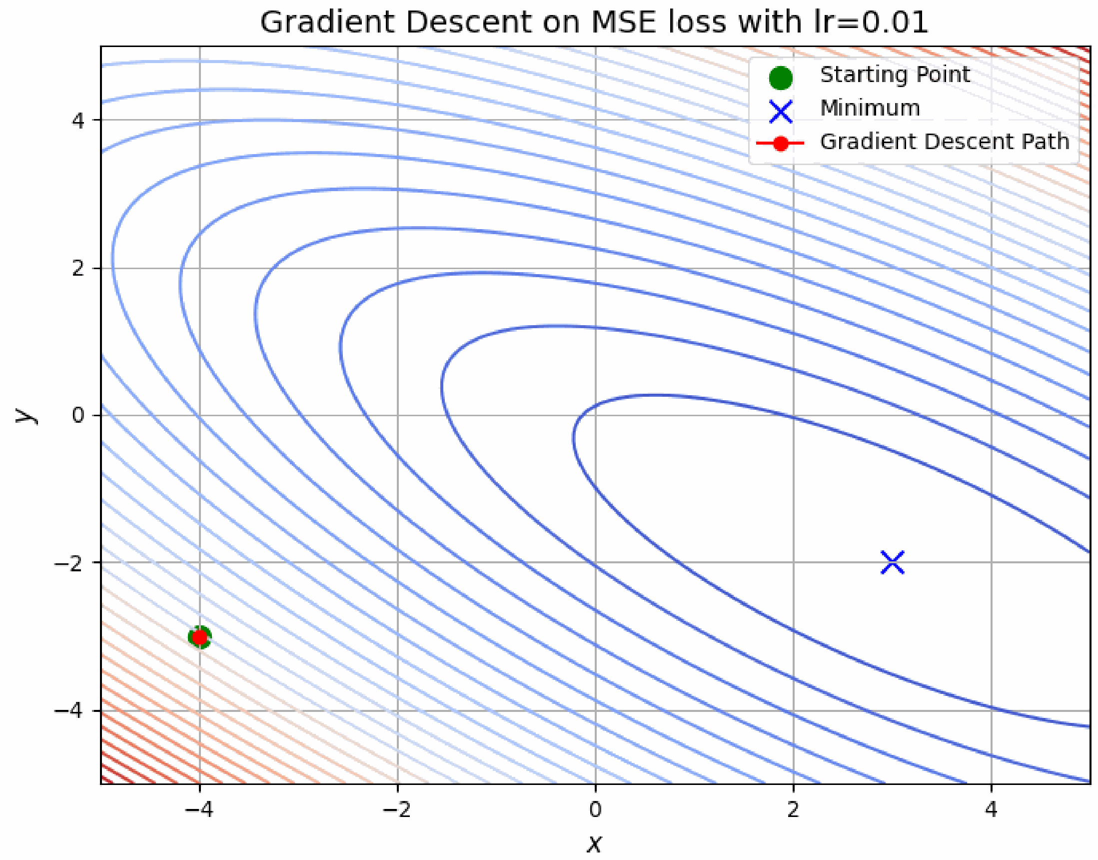
GD	SGD with batch size B
$$w_{k+1} = w_k - \alpha \sum_{i=1}^{N} \nabla_w \mathcal{L}(y_i, x_i, w_k)$$	Pick B **random** data points $(x_{i_s}, y_{i_s})_{s=1}^{B}$, $$w_{k+1} = w_k - \alpha \sum_{s=1}^{B} \nabla_w \mathcal{L}(y_{i_s}, x_{i_s}, w_k)$$

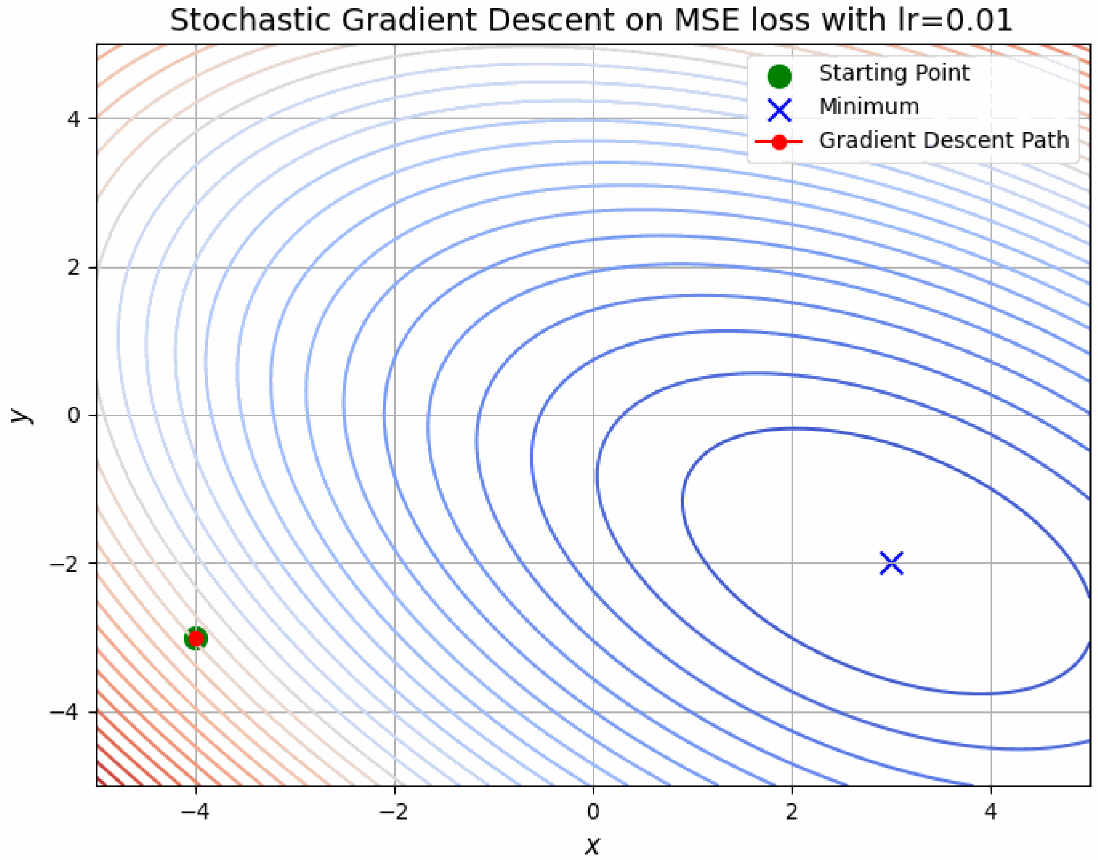
A typical implementation

GD	SGD with batch size B
$$w_{k+1} = w_k - \alpha \sum_{i=1}^{N} \nabla_w \mathcal{L}(y_i, x_i, w_k)$$	For several **epochs**: • Randomize data order • Now, for **t=0..(N/B-1)** $$w_{k+1} = w_k - \alpha \sum_{i=Bt}^{Bt+B-1} \nabla_w \mathcal{L}(y_i, x_i, w_k)$$

Comparing behaviour: GD vs SGD

SGD is erratic, and the smaller the batch size, the worse





A tricky question

GD	**SGD with batch size B**
$$w_{k+1} = w_k - \frac{\alpha}{N}\sum_{i=1}^{N} \nabla_w \mathcal{L}(y_i, x_i, w_k)$$	Pick B **random** data points $\left(x_{i_s}, y_{i_s}\right)_{s=1}^{B}$, $$w_{k+1} = w_k - \frac{\alpha}{???}\sum_{s=1}^{B} \nabla_w \mathcal{L}\left(y_{i_s}, x_{i_s}, w_k\right)$$

A tricky question

GD	**SGD with batch size B**
$w_{k+1} = w_k - \frac{\alpha}{N}\sum_{i=1}^{N} \nabla_w \mathcal{L}(y_i, x_i, w_k)$	Pick B **random** data points $\left(x_{i_s}, y_{i_s}\right)_{s=1}^{B}$, $w_{k+1} = w_k - \frac{\alpha}{???}\sum_{s=1}^{B} \nabla_w \mathcal{L}\left(y_{i_s}, x_{i_s}, w_k\right)$
$\nabla_w \mathcal{L}(y, X, w) = \frac{1}{N}\sum_{i=1}^{N} \nabla_w \mathcal{L}(y_i, x_i, w_k)$	$\nabla_w \mathcal{L}(y, X, w) \approx \frac{1}{B}\sum_{s=1}^{B} \nabla_w \mathcal{L}\left(y_{i_s}, x_{i_s}, w_k\right)$ A Monte Carlo estimate

A tricky question

GD	SGD with batch size B
$$w_{k+1} = w_k - \frac{\alpha}{N}\sum_{i=1}^{N} \nabla_w \mathcal{L}(y_i, x_i, w_k)$$	Pick B **random** data points $\left(x_{i_s}, y_{i_s}\right)_{s=1}^{B}$, $$w_{k+1} = w_k - \frac{\alpha}{\boldsymbol{B}}\sum_{s=1}^{B} \nabla_w \mathcal{L}\left(y_{i_s}, x_{i_s}, w_k\right)$$
$$\nabla_w \mathcal{L}(y, X, w) = \frac{1}{N}\sum_{i=1}^{N} \nabla_w \mathcal{L}(y_i, x_i, w_k)$$	$$\nabla_w \mathcal{L}(y, X, w) \approx \frac{1}{B}\sum_{s=1}^{B} \nabla_w \mathcal{L}\left(y_{i_s}, x_{i_s}, w_k\right)$$ A Monte Carlo estimate

A curious analogy: Natural Language Gradient

Example TextGrad

Loss: an LLM call to evaluate a solution based on pre-set evaluation instructions. Outputs criticism of a solution.

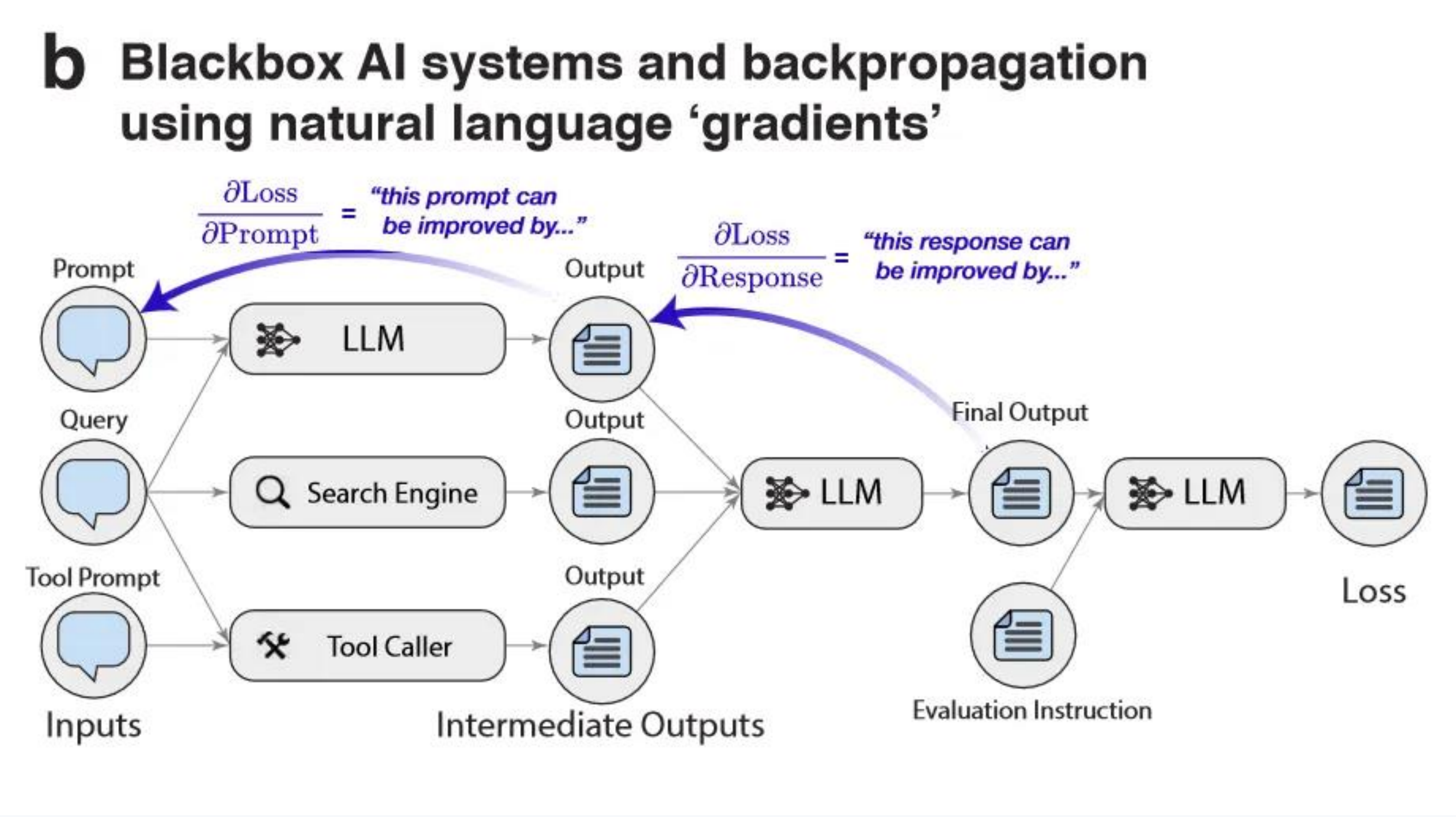
“Gradient operator":
Criticism of the prompt ← Criticism of the intermediate output ← Criticism of final output

"Textual Gradient Descent“: rewrites a prompt or an intermediate output based on the criticism.

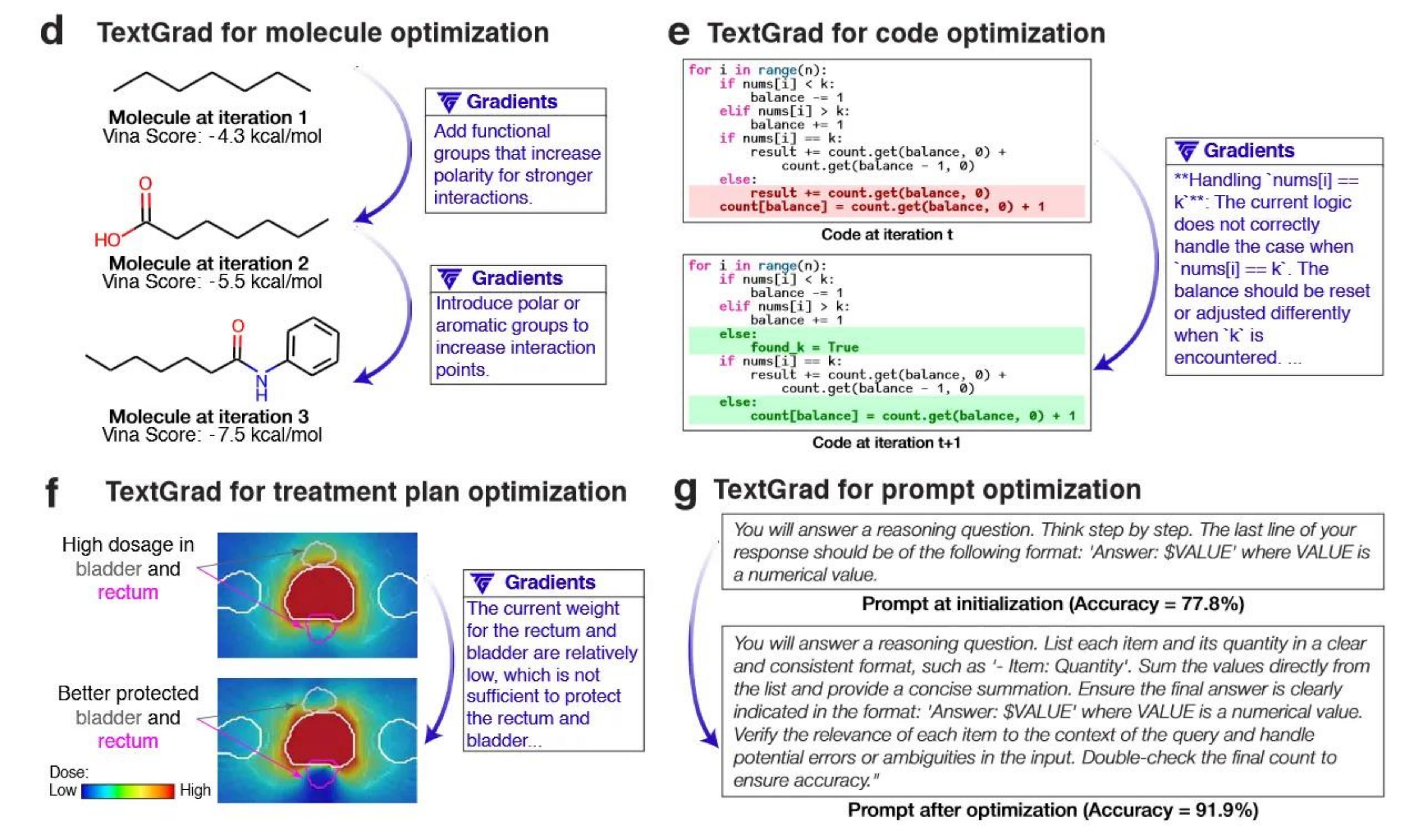
TextGrad: Automatic “Differentiation” via Text https://arxiv.org/pdf/2406.07496

Example

TextGrad



Example


d TextGrad for molecule optimization
Molecule at iteration 1
Vina Score: -4.3 kcal/mol
Gradients
Add functional groups that increase polarity for stronger interactions.
Molecule at iteration 2
Vina Score: -5.5 kcal/mol
Gradients
Introduce polar or aromatic groups to increase interaction points.
Molecule at iteration 3
Vina Score: -7.5 kcal/mol
e TextGrad for code optimization
for i in range(n):
    if nums[i] < k:
        balance -= 1
    elif nums[i] > k:
        balance += 1
    if nums[i] == k:
        result += count.get(balance, 0) +
            count.get(balance - 1, 0)
    else:
        result += count.get(balance, 0)
    count[balance] = count.get(balance, 0) + 1
Code at iteration t
for i in range(n):
    if nums[i] < k:
        balance -= 1
    elif nums[i] > k:
        balance += 1
    else:
        found_k = True
    if nums[i] == k:
        result += count.get(balance, 0) +
            count.get(balance - 1, 0)
    else:
        count[balance] = count.get(balance, 0) + 1
Code at iteration t+1
Gradients
Handling `nums[i] == k`: The current logic does not correctly handle the case when `nums[i] == k`. The balance should be reset or adjusted differently when `k` is encountered. ...
f TextGrad for treatment plan optimization
High dosage in bladder and rectum
Better protected bladder and rectum
Dose:
Low
High
Gradients
The current weight for the rectum and bladder are relatively low, which is not sufficient to protect the rectum and bladder...
g TextGrad for prompt optimization
You will answer a reasoning question. Think step by step. The last line of your response should be of the following format: 'Answer: $VALUE' where VALUE is a numerical value.
Prompt at initialization (Accuracy = 77.8%)
You will answer a reasoning question. List each item and its quantity in a clear and consistent format, such as '- Item: Quantity'. Sum the values directly from the list and provide a concise summation. Ensure the final answer is clearly indicated in the format: 'Answer: $VALUE' where VALUE is a numerical value. Verify the relevance of each item to the context of the query and handle potential errors or ambiguities in the input. Double-check the final count to ensure accuracy."
Prompt after optimization (Accuracy = 91.9%)

Regularization

What is regularization?

- Countering overfitting
- Stabilizing optimization
- Imposing conditions

Multicollinearity

The worst regression task:

$$X = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

$$\mathcal{L}(y, X, w) = \frac{1}{N} \sum_{i=1}^{N} (y_i - w_1)^2$$

Solution:

$$w_1 = \text{mean}(y_i), \qquad w_2 - \text{any!}$$

Multicollinearity

The worst regression task:

$$X = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

$$\mathcal{L}(y, X, w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - w_1)^2$$

Solution:

$$w_1 = \text{mean}(y_i), \qquad w_2 - \text{any!}$$

$$X^T X = \begin{pmatrix} N & 0 \\ 0 & 0 \end{pmatrix}$$

Not invertible!

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

Multicollinearity

A quite bad regression task:

$$X = \begin{pmatrix} 1 & \varepsilon \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

ε is a very small number.

$$X^T X = \begin{pmatrix} N & \varepsilon \\ \varepsilon & \varepsilon^2 \end{pmatrix}$$

is close to degenerate.

$$(X^T X)^{-1} = \frac{1}{N-1} \begin{pmatrix} 1 & -\frac{1}{\varepsilon} \\ -\frac{1}{\varepsilon} & \frac{N}{\varepsilon^2} \end{pmatrix}$$

$$(X^T X)^{-1} X^T = \frac{1}{N-1} \begin{pmatrix} 0 & 1 & \cdots \\ \frac{N-1}{\varepsilon} & -\frac{1}{\varepsilon} & \cdots \end{pmatrix}$$

$$\hat{w}^T \to (X^T X)^{-1} X^T (y + \delta)$$

A small change in y may lead to a catastrophic change in $\hat{w}$.

Multicollinearity

A quite bad regression task:

$$X = \begin{pmatrix} 1 & \varepsilon \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

$\varepsilon = 10^{-4}$,
$N = 100$

$$X^T X = \begin{pmatrix} 100 & 10^{-4} \\ 10^{-4} & 10^{-8} \end{pmatrix}$$

is close to degenerate.

$$(X^T X)^{-1} = \frac{1}{99} \begin{pmatrix} 1 & -10^4 \\ -10^4 & 10^{10} \end{pmatrix} \approx \begin{pmatrix} 10^{-2} & -10^2 \\ -10^2 & 10^8 \end{pmatrix}$$

$$(X^T X)^{-1} X^T \approx \begin{pmatrix} 0 & 10^{-2} & \cdots \\ 10^4 & 10^2 & \cdots \end{pmatrix}$$

$$\hat{w}^T \to \begin{pmatrix} 0 & 10^{-2} & \cdots \\ 10^4 & 10^2 & \cdots \end{pmatrix} (y + 10^{-2})$$

A small change in y may lead to a catastrophic change in $\hat{w}$.

Multicollinearity

A quite bad regression task:

$$X = \begin{pmatrix} 1 & \varepsilon \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

$\varepsilon = 10^{-4}$,
$N = 100$

$$X^T X + 10^{-4} I = \begin{pmatrix} 100 + 10^{-4} & 10^{-4} \\ 10^{-4} & 10^{-8} + 10^{-4} \end{pmatrix}$$

$$(X^T X)^{-1} X^T \approx \begin{pmatrix} 0.25 & 0.25 & \cdots \\ 0.75 & -0.25 & \cdots \end{pmatrix}$$

That's much better!

L2-regularization

How to fix an almost-degenerate matrix:

$$\hat{w}^T = (X^T X + \lambda I)^{-1} X^T y$$

We can prove that this corresponds to the following task:

$$\mathcal{L}(y, X, w) = \left[\sum_{i=1}^{N} (y_i - x_i w^T)^2\right] + \lambda \|w\|_2^2$$

where $\|w\|_2^2 = w_1^2 + \cdots + w_D^2$

L2-regularization in general case

$$\mathcal{L}_{reg}(y, X, w) = \mathcal{L}(y, X, w) + \lambda \|w\|_2^2$$

where $\|w\|_2^2 = w_1^2 + \cdots + w_D^2$

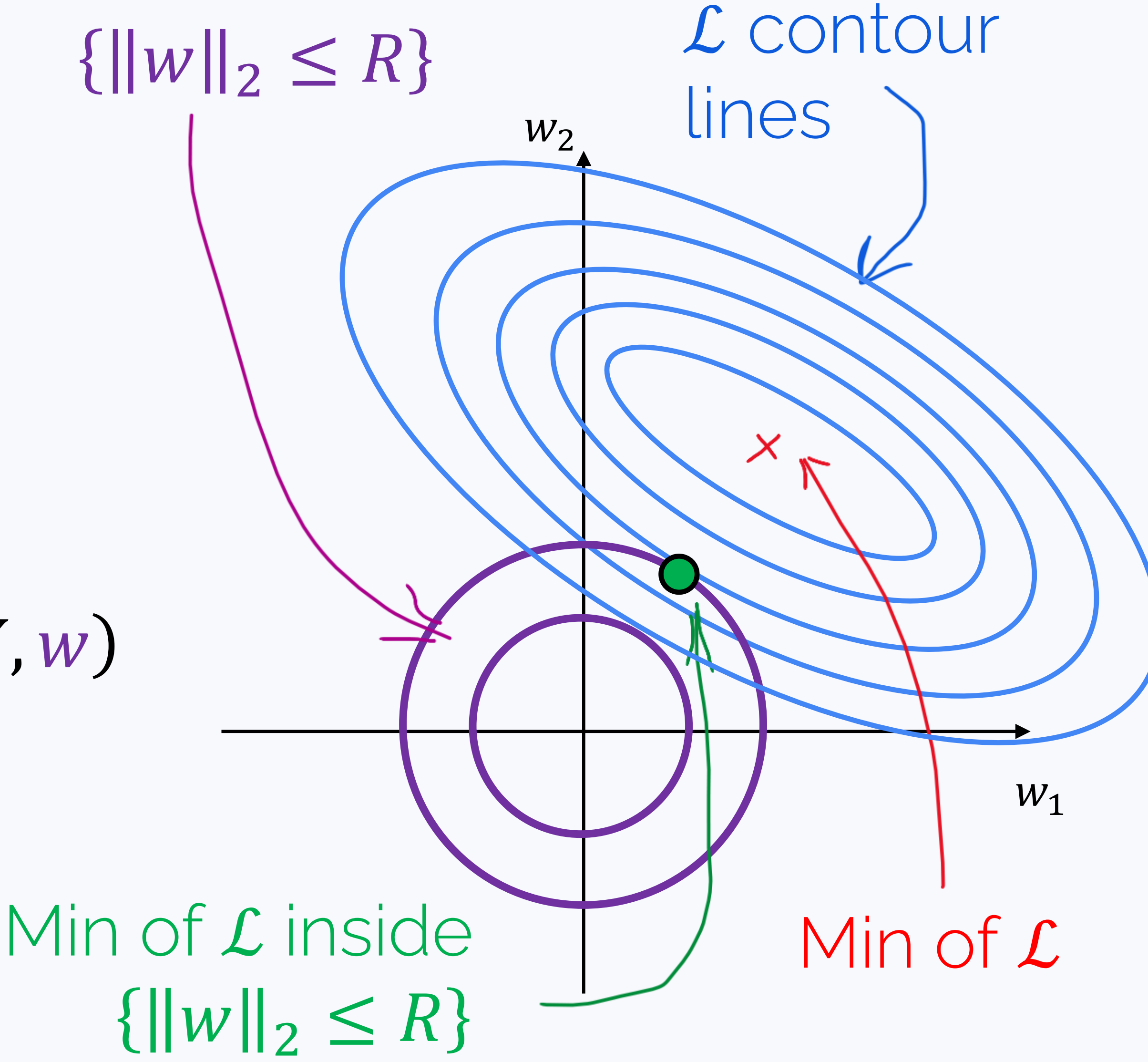
Theory says that optimizing this new loss is equivalent to optimizing $\mathcal{L}(y, X, w)$ inside some $\{\|w\|_2 \leq R\}$

L2-regularization in general case

$$\mathcal{L}_{reg}(y, X, w) = \mathcal{L}(y, X, w) + \lambda \|w\|_2^2$$

where $\|w\|_2^2 = w_1^2 + \cdots + w_D^2$

Theory says that optimizing this new loss is equivalent to optimizing $\mathcal{L}(y, X, w)$ inside some $\{\|w\|_2 \leq R\}$



L2-regularization in general case

$$\mathcal{L}_{reg}(y, X, w) = \mathcal{L}(y, X, w) + \lambda \|w\|_2^2$$

where $\|w\|_2^2 = w_1^2 + \dots + w_D^2$

Helps not to overfit on noisy features.

λ is a **hyperparameter**. It is tuned on a logarithmic scale.

L1-regularization in general case

$$\mathcal{L}_{reg}(y, X, w) = \mathcal{L}(y, X, w) + \lambda \|w\|_1$$

where $\|w\|_1 = |w_1| + \cdots + |w_D|$

Theory says that optimizing this new loss is equivalent to optimizing $\mathcal{L}(y, X, w)$ inside some $\{\|w\|_1 \leq R\}$

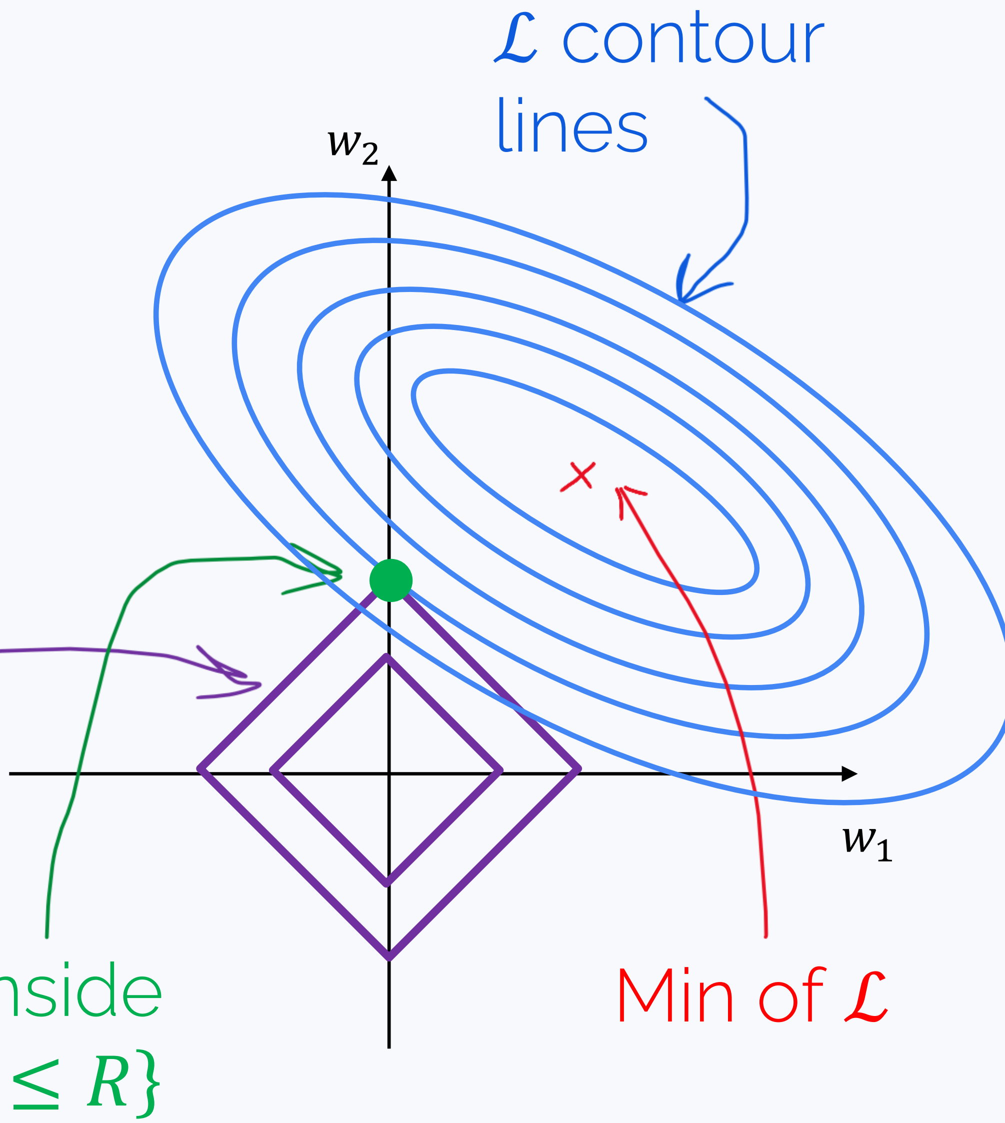
L1-regularization leads to sparsity

$$\mathcal{L}_{reg}(y, X, w) = \mathcal{L}(y, X, w) + \lambda \|w\|_1$$

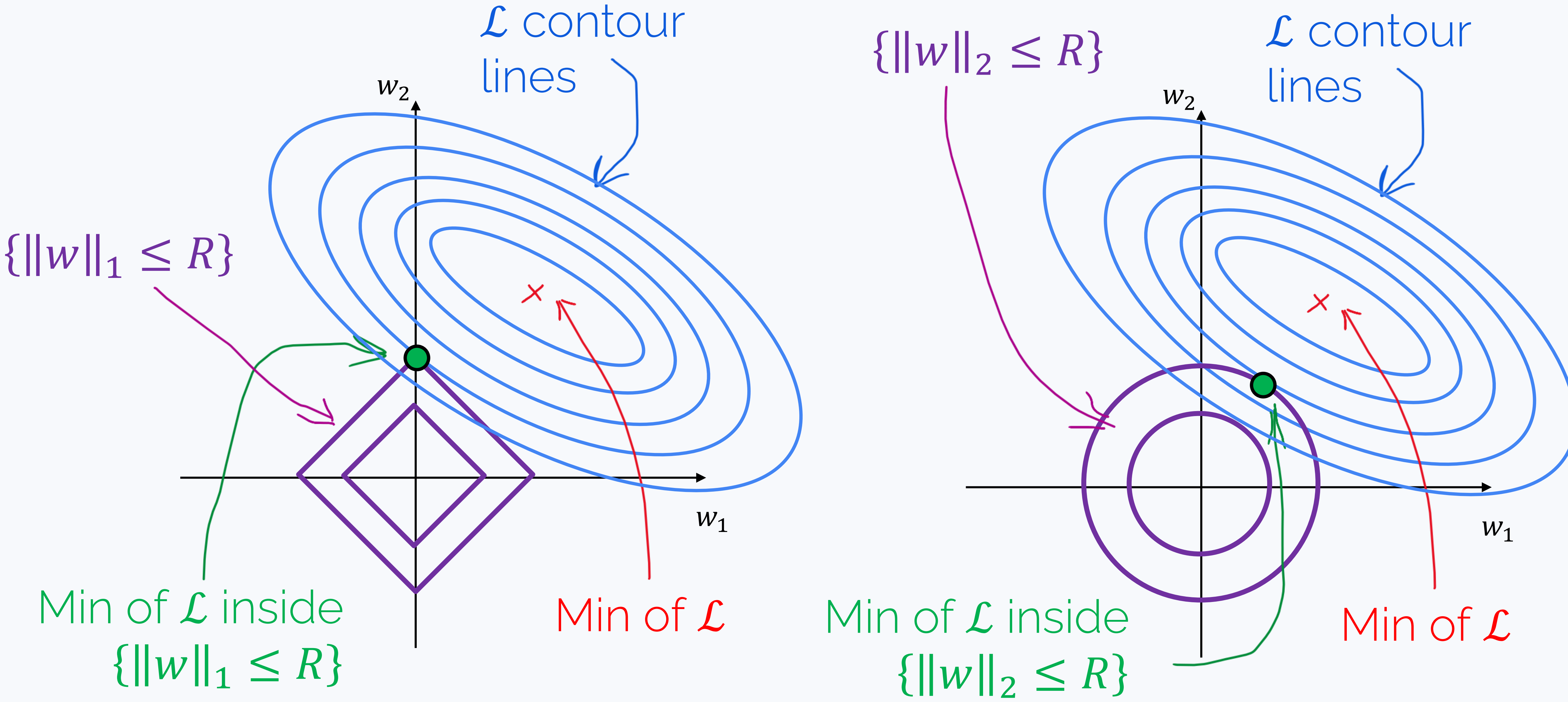
where $\|w\|_1 = |w_1| + \cdots + |w_D|$

Theory says that optimizing this new loss is equivalent to optimizing $\mathcal{L}(y, X, w)$ inside some $\{\|w\|_1 \leq R\}$

Many coordinates of the resulting $\hat{w}$ will likely be zero.
The larger λ, the more.



Compare: L1 vs L2



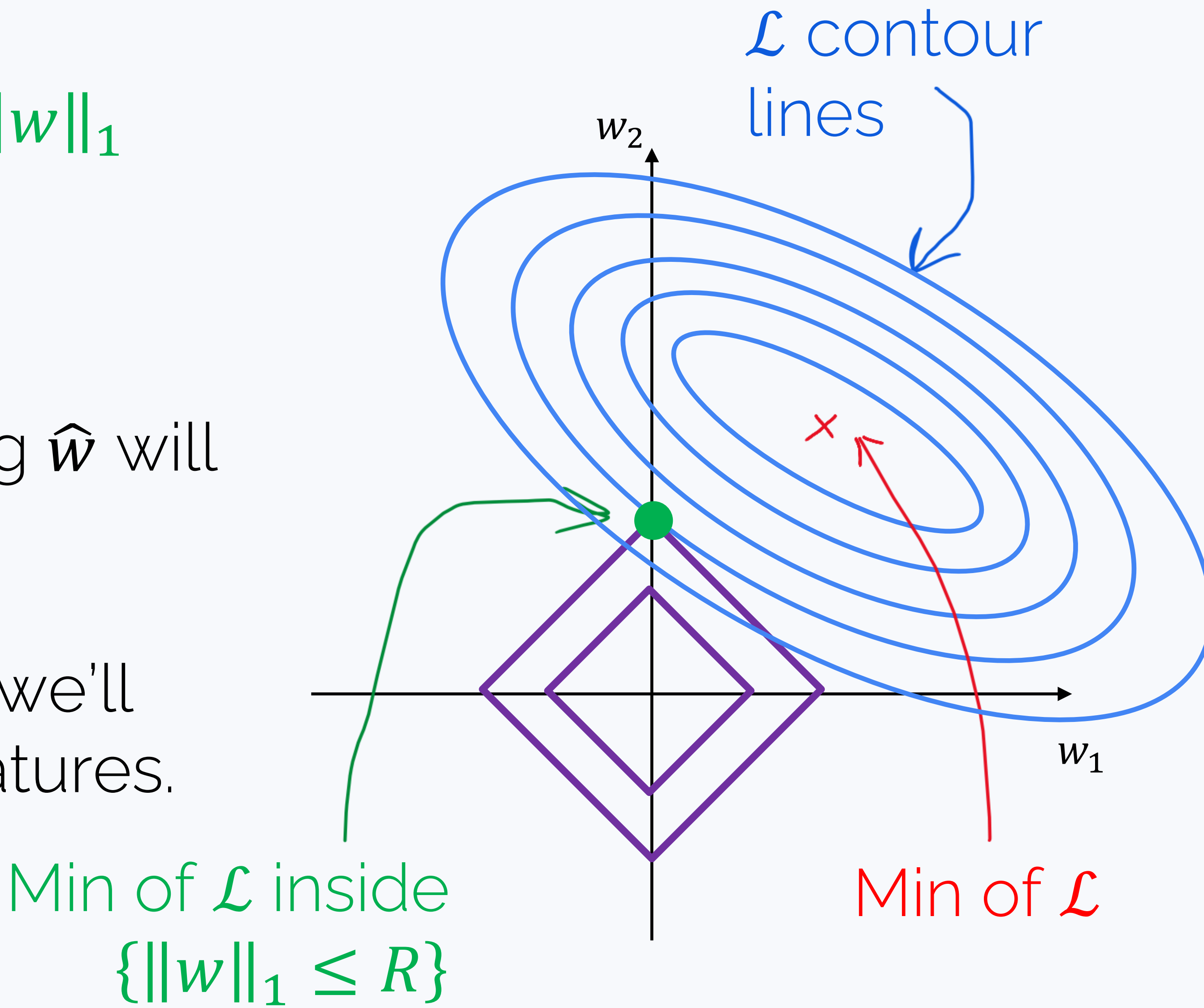
L1-regularization: why sparsity?

$$\mathcal{L}_{reg}(y, X, w) = \mathcal{L}(y, X, w) + \lambda\|w\|_1$$

where $\|w\|_1 = |w_1| + \cdots + |w_D|$

Many coordinates of the resulting $\hat{w}$ will likely be zero.

Why bother? There's hope that we'll get rid of noise and irrelevant features. Counters overfitting.



Dimension reduction

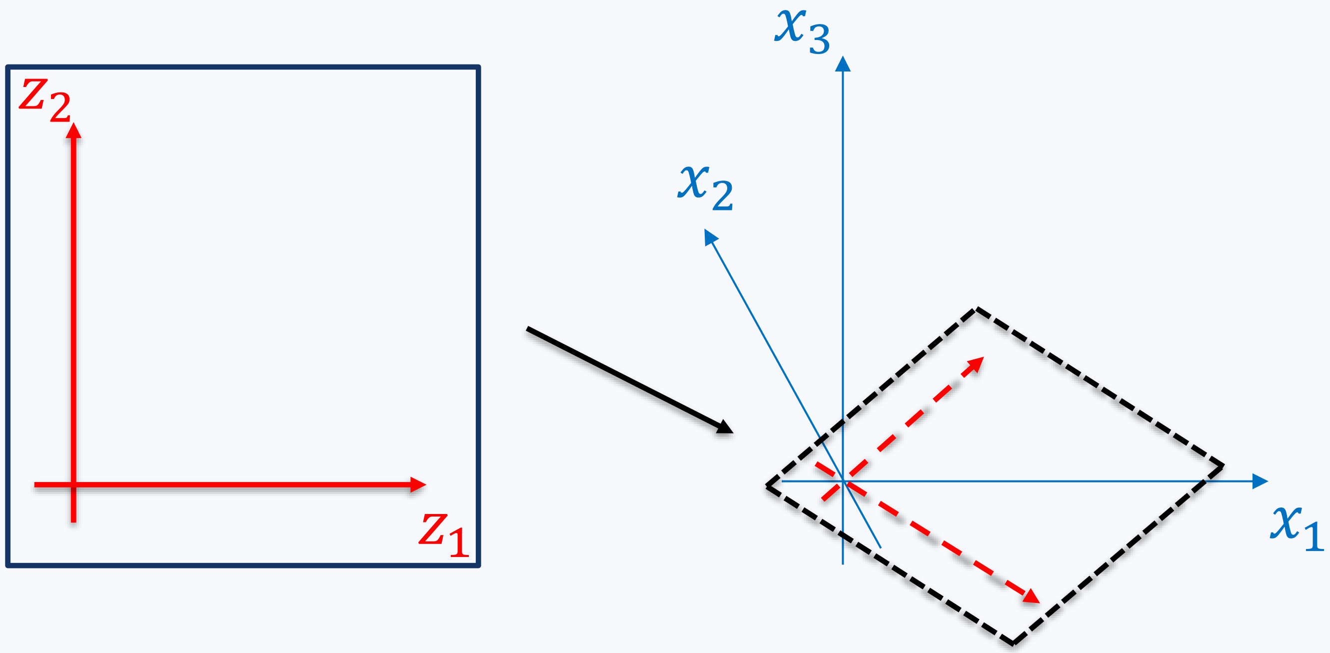
Principal component analysis

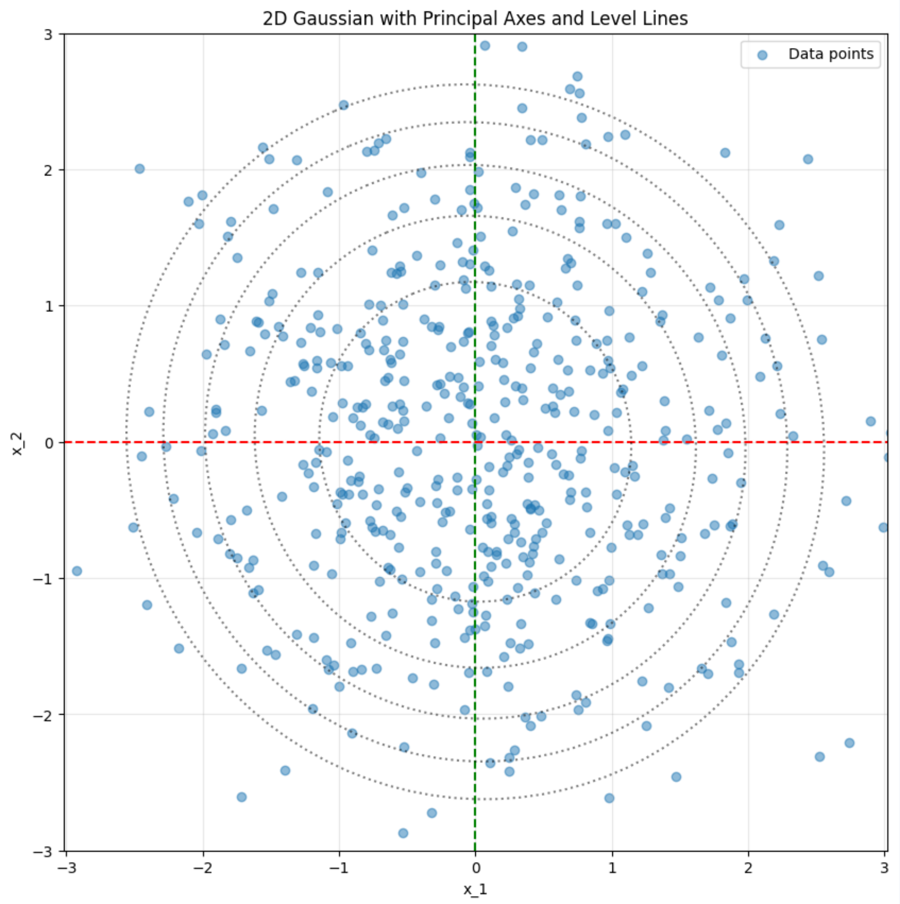
Establish

$$\overset{N \times D}{X} \approx \overset{N \times d}{Z} \cdot V^T$$

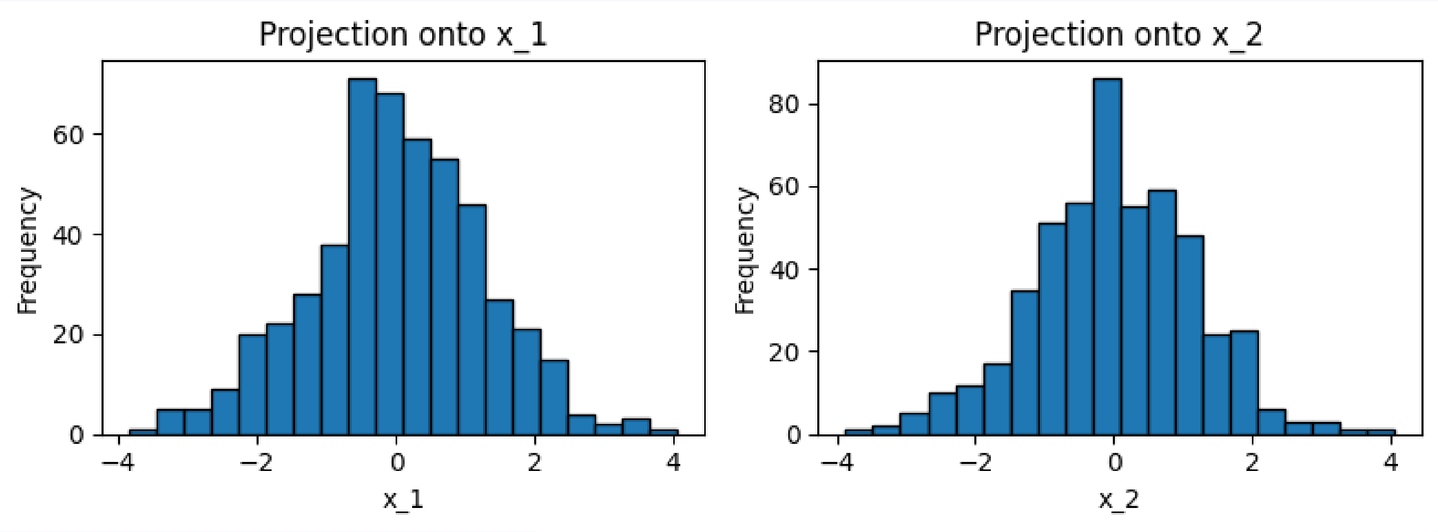
where z_i are new (**latent**) features of the i-th object and $\mathrm{d} < D$.

- $Z \to Z \cdot V^T$ is feature mixing
- $Z \to U \cdot Z$ is object mixing

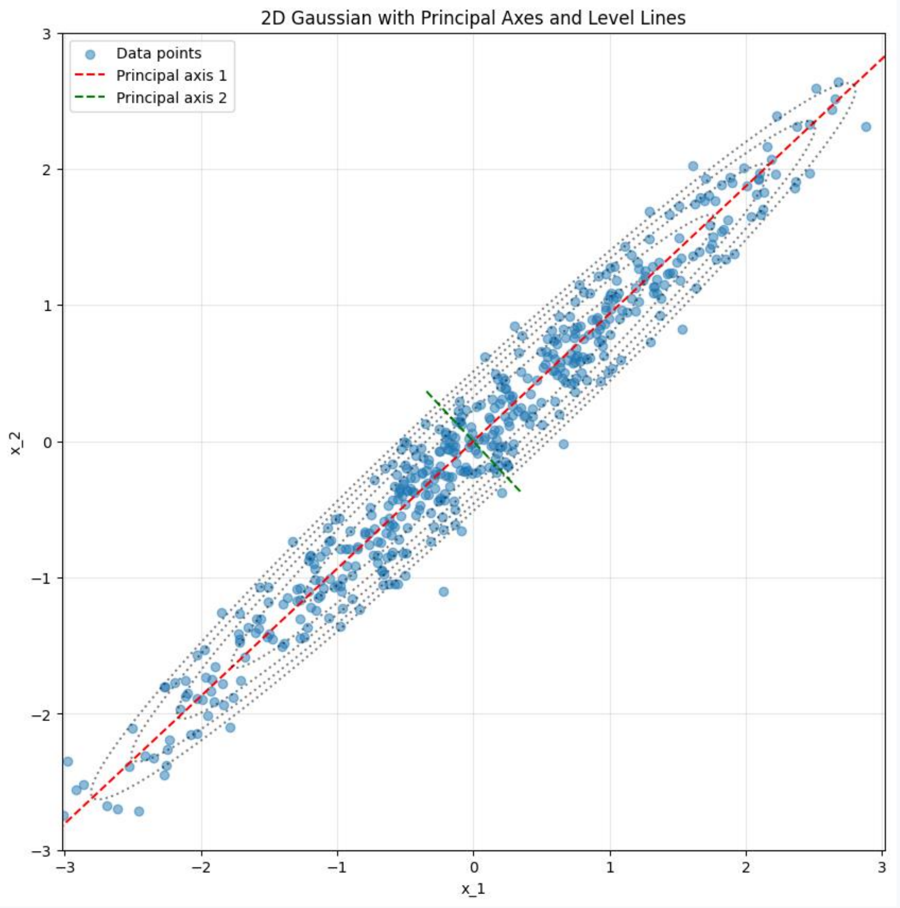




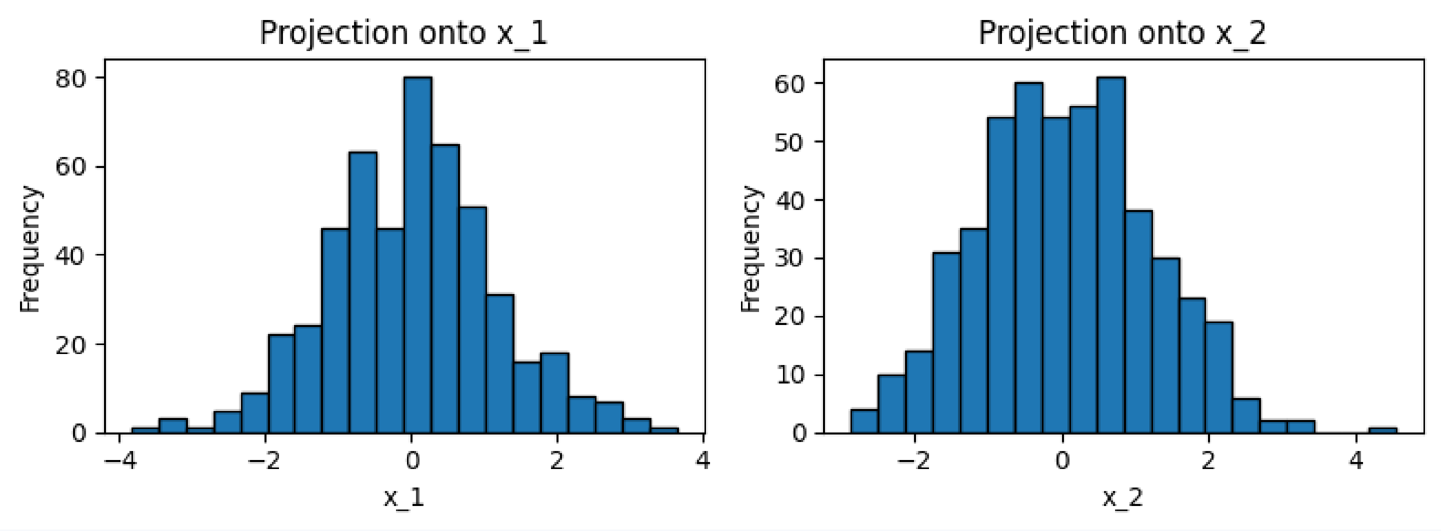
2D Gaussian with Principal Axes and Level Lines
Data points
x_1
x_2



Projection onto x_1
Frequency
x_1
Projection onto x_2
Frequency
x_2



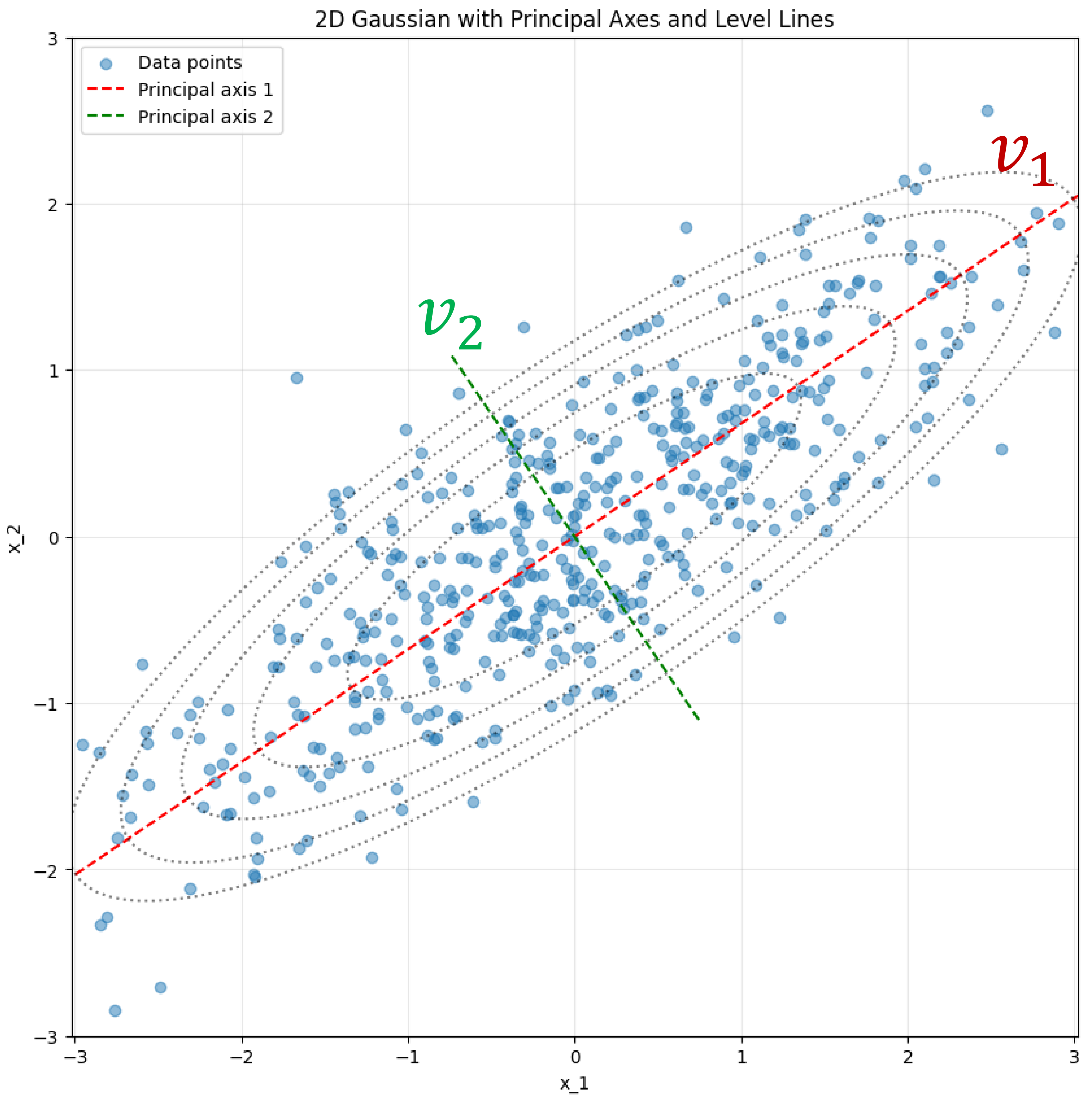
2D Gaussian with Principal Axes and Level Lines
Data points
Principal axis 1
Principal axis 2
x_1
x_2



Projection onto x_1
Frequency
x_1
Projection onto x_2
Frequency
x_2

Principal component analysis

- X should be centered
- Then, we take the data ellipse's principal axes as new features, from the largest to the smallest one.

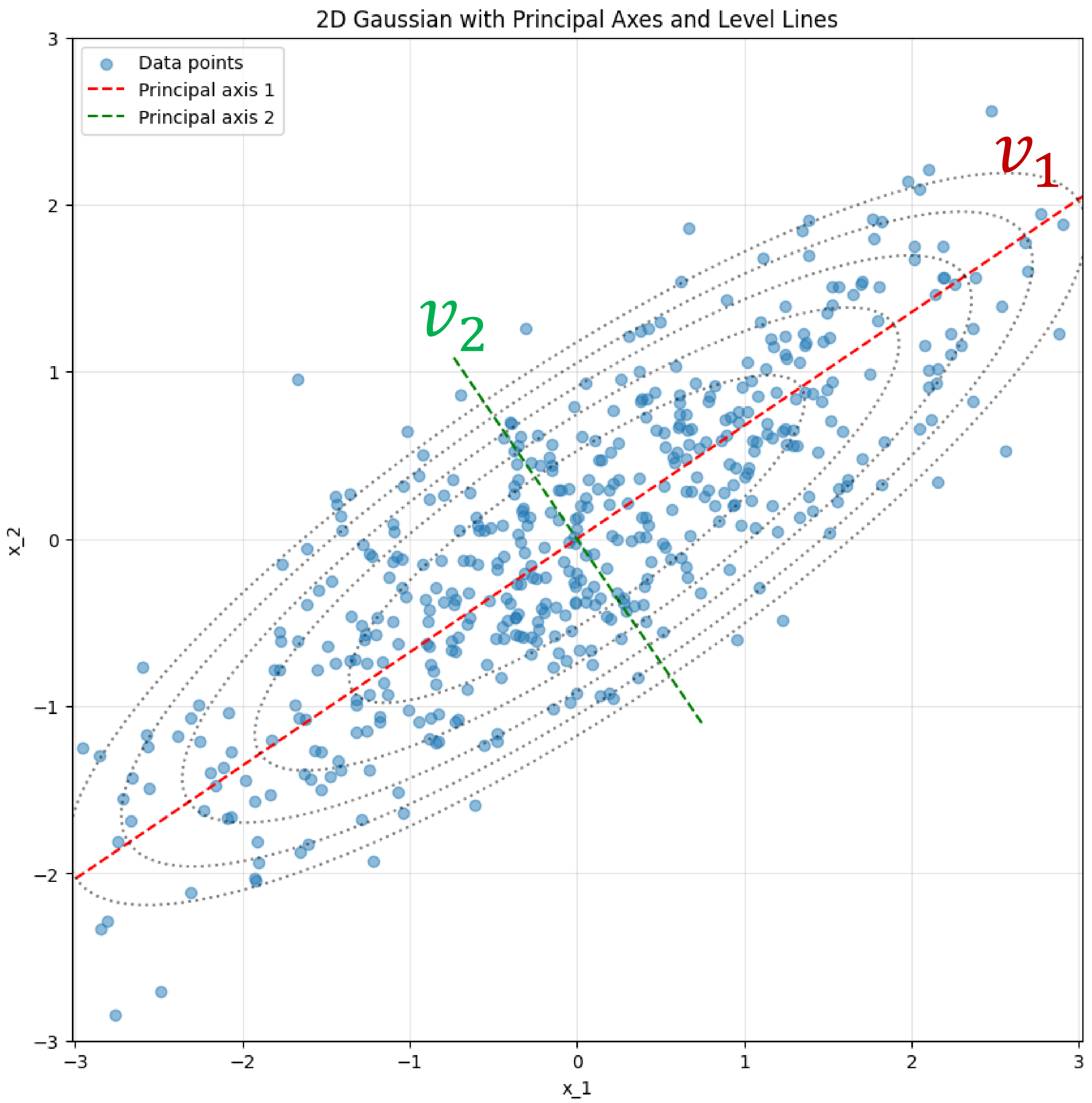


Principal component analysis

If $V = (v_1, \ldots, v_D)$, then

$$V^T X^T X V = \Sigma =$$
$$= diag(\sigma_1, \sigma_2, \ldots)$$

where $\sigma_1 \geq \sigma_2 \geq \cdots$



Singular value decomposition (SVD)

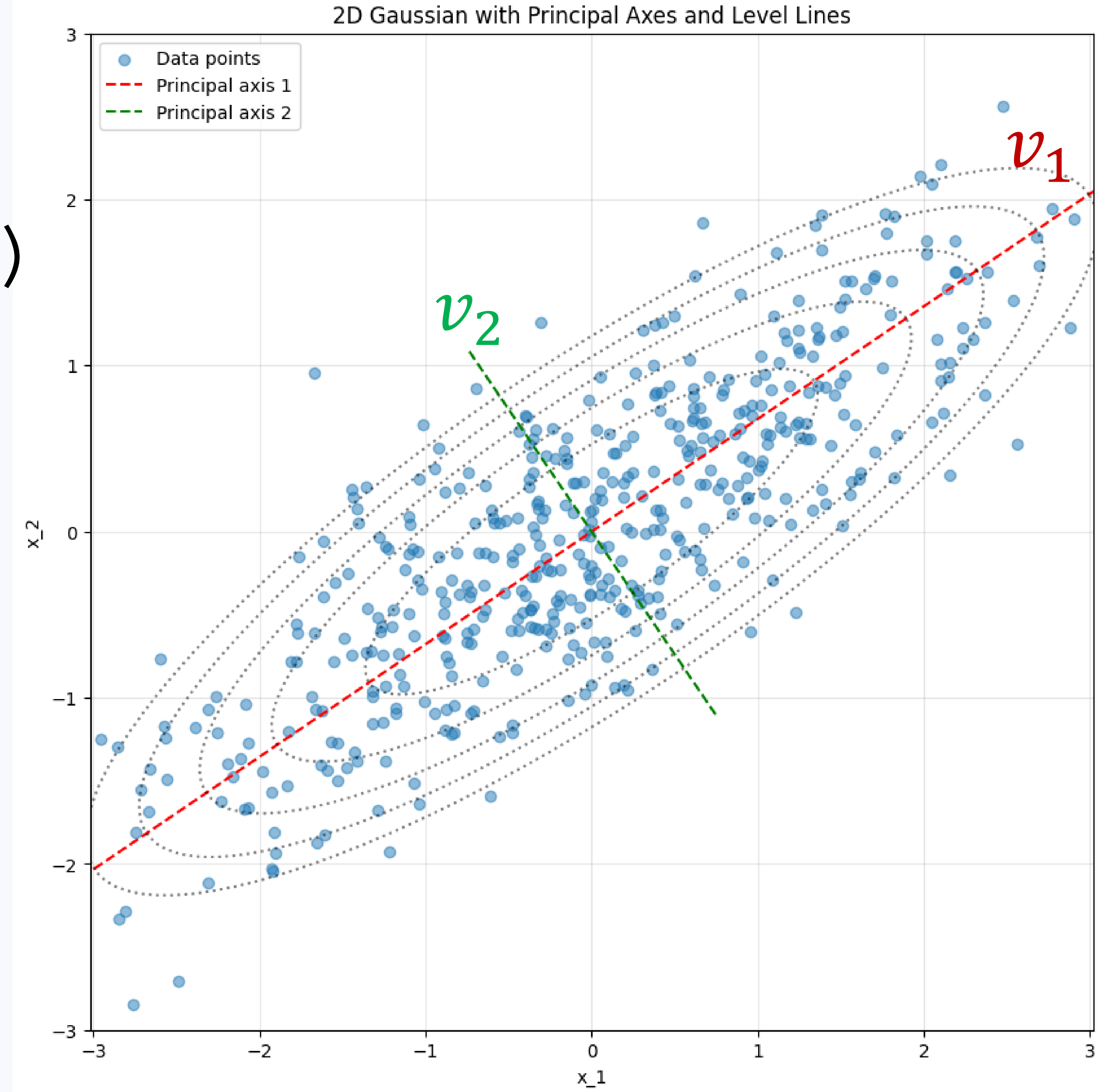
One may prove that

$$X = U\Sigma V^T$$

where

- U and V have orthogonal columns,
- $\Sigma = diag(\sigma_1, \sigma_2, \dots)$

where $\sigma_1 \geq \sigma_2 \geq \cdots$



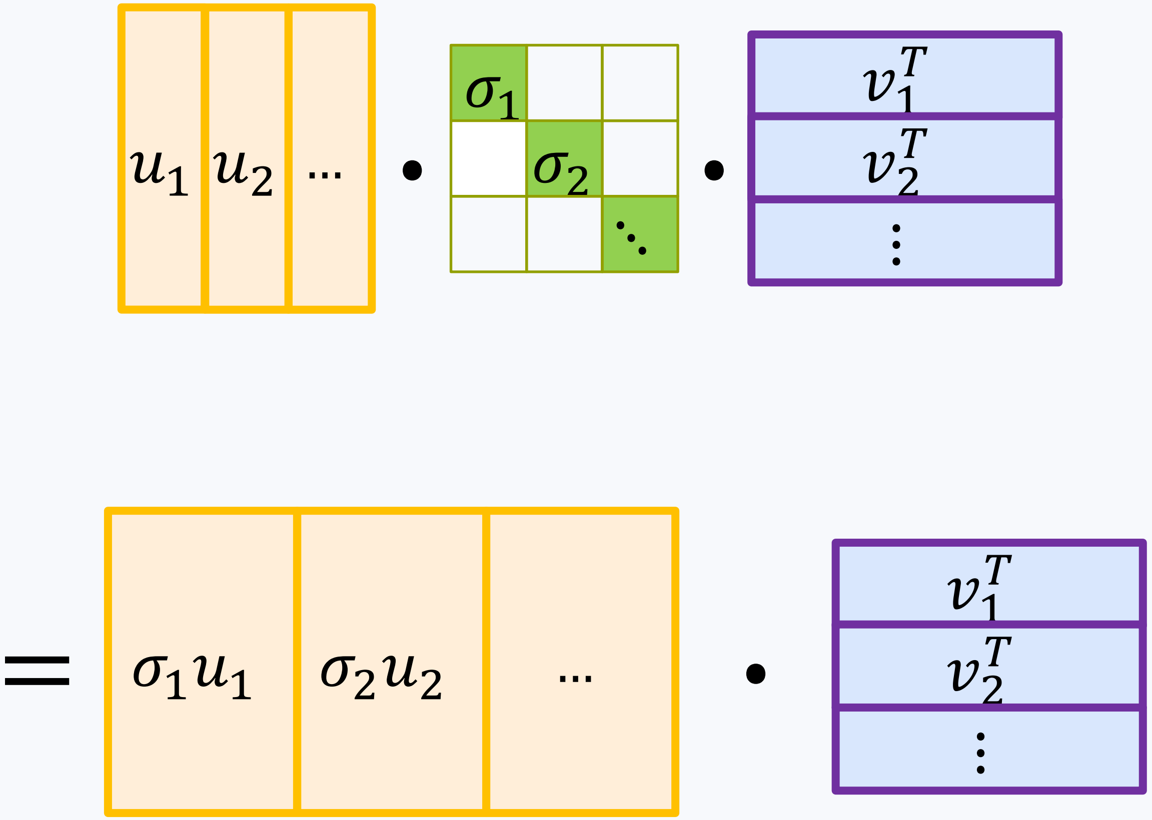
Principal component analysis

$$X = U\Sigma V^T$$

Here, V^T mixes features. We may take new feature description:

$$X \to U\Sigma$$

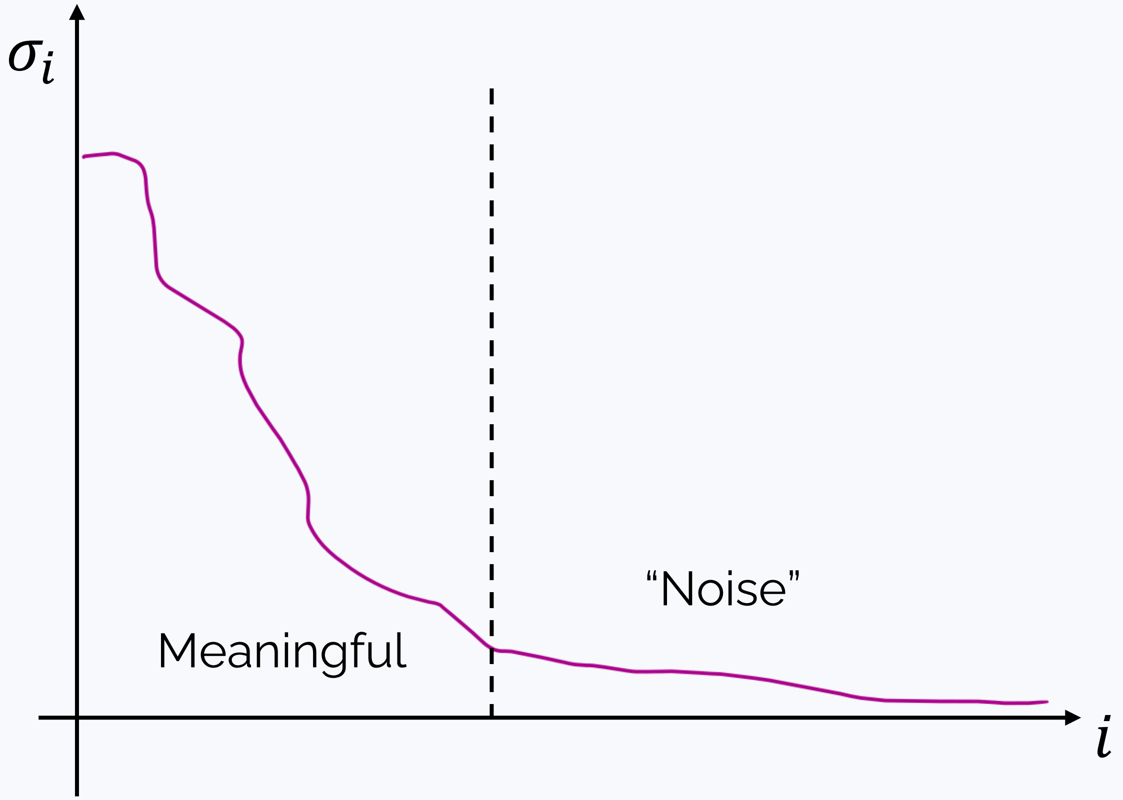
i-th object will be described by $(\sigma_i u_{i1}, \sigma_i u_{i2}, \dots)$

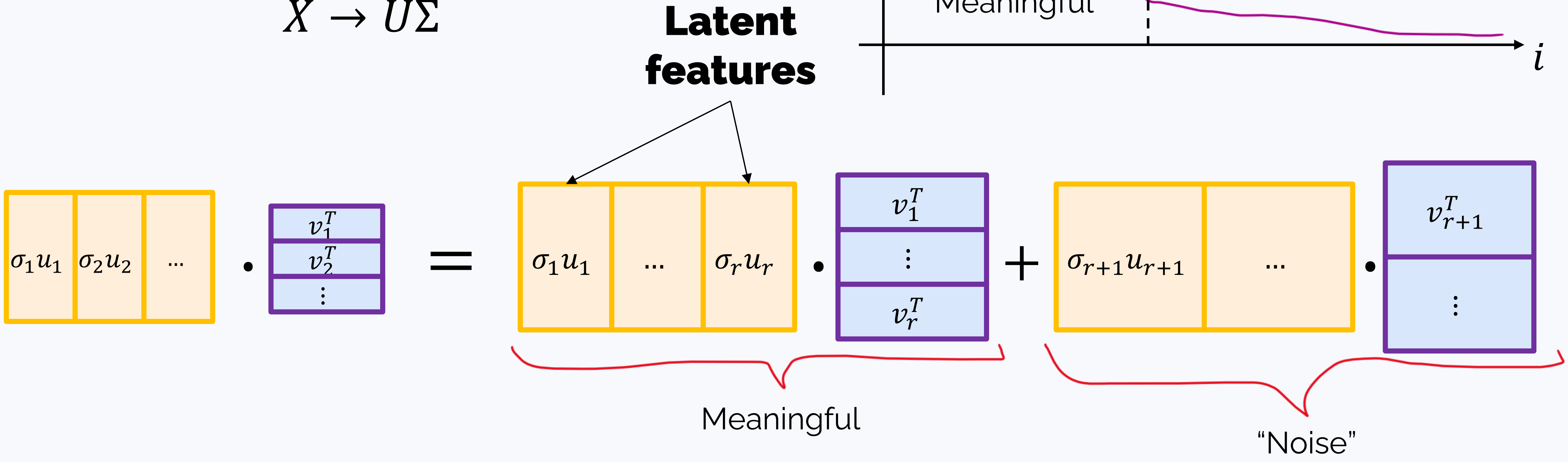


Principal component analysis

$$X = U\Sigma V^T$$

$$X \to U\Sigma$$





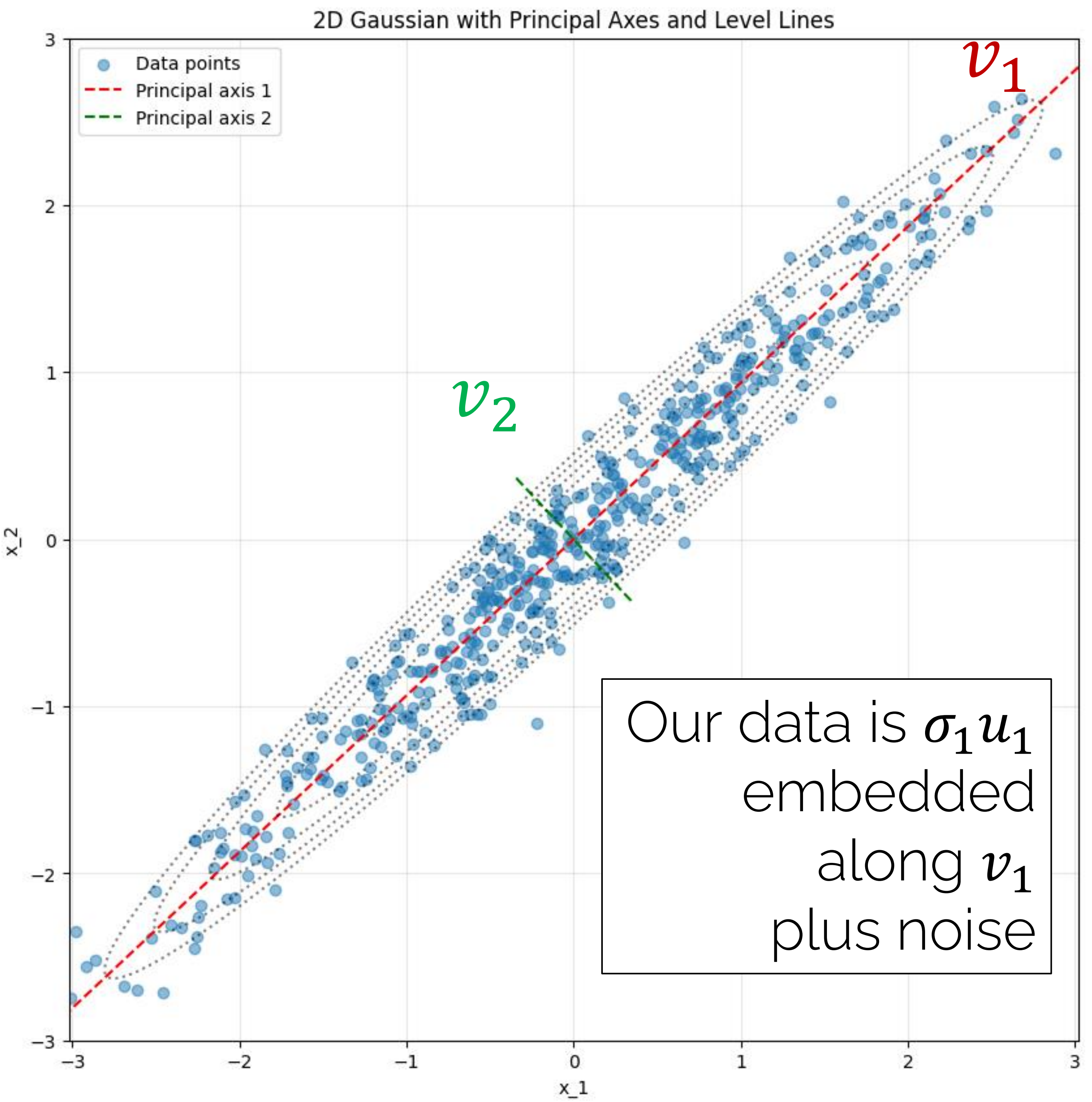
Principal component analysis

Now, we say:

- If v_2 is a noisy feature, let's just forget about it.

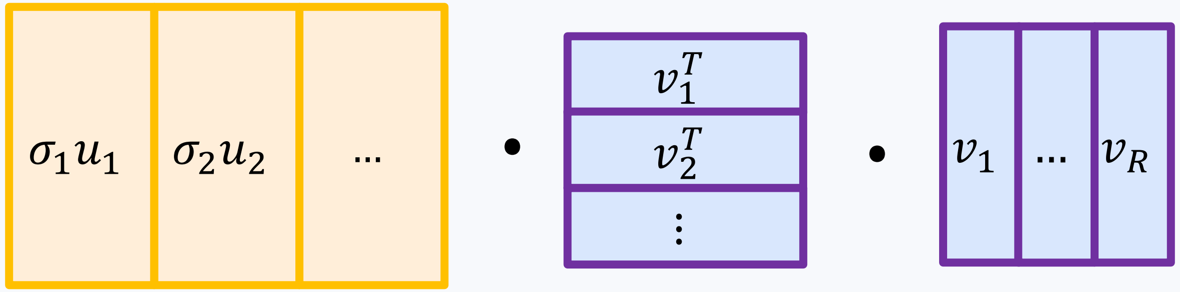
In general, we have new features

$\sigma_1 u_1$	...	$\sigma_r u_r$



Mapping to latent feature space and back

$$X = U\Sigma V^T \rightarrow U\Sigma \overset{= I}{\boxed{V^T \cdot V}} = U\Sigma$$

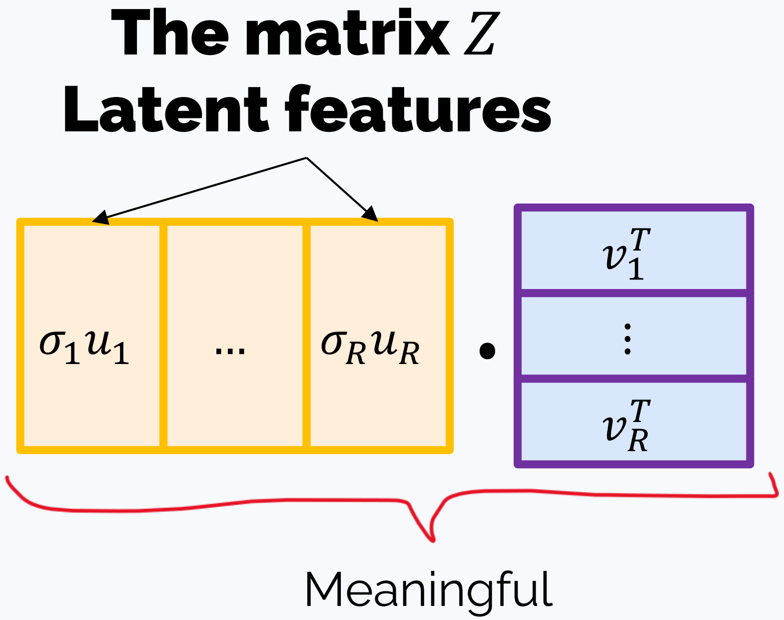


If we want to get R latent features:

$$U\Sigma[:,:R] = U\Sigma V^T \cdot V[:,:R]$$

$$X_{test} \rightarrow X_{test} \cdot V[:,:R]$$

Back: $Z \rightarrow Z \cdot V^T[:R,:]$



Pros and cons of PCA

Pros:

- Simple and effective
- Still encountered in papers

Cons:

- May not work well with non-ellipsoid (non-gaussian data).
- Small ≠ irrelevant
- Not so practical in the LLM era :(

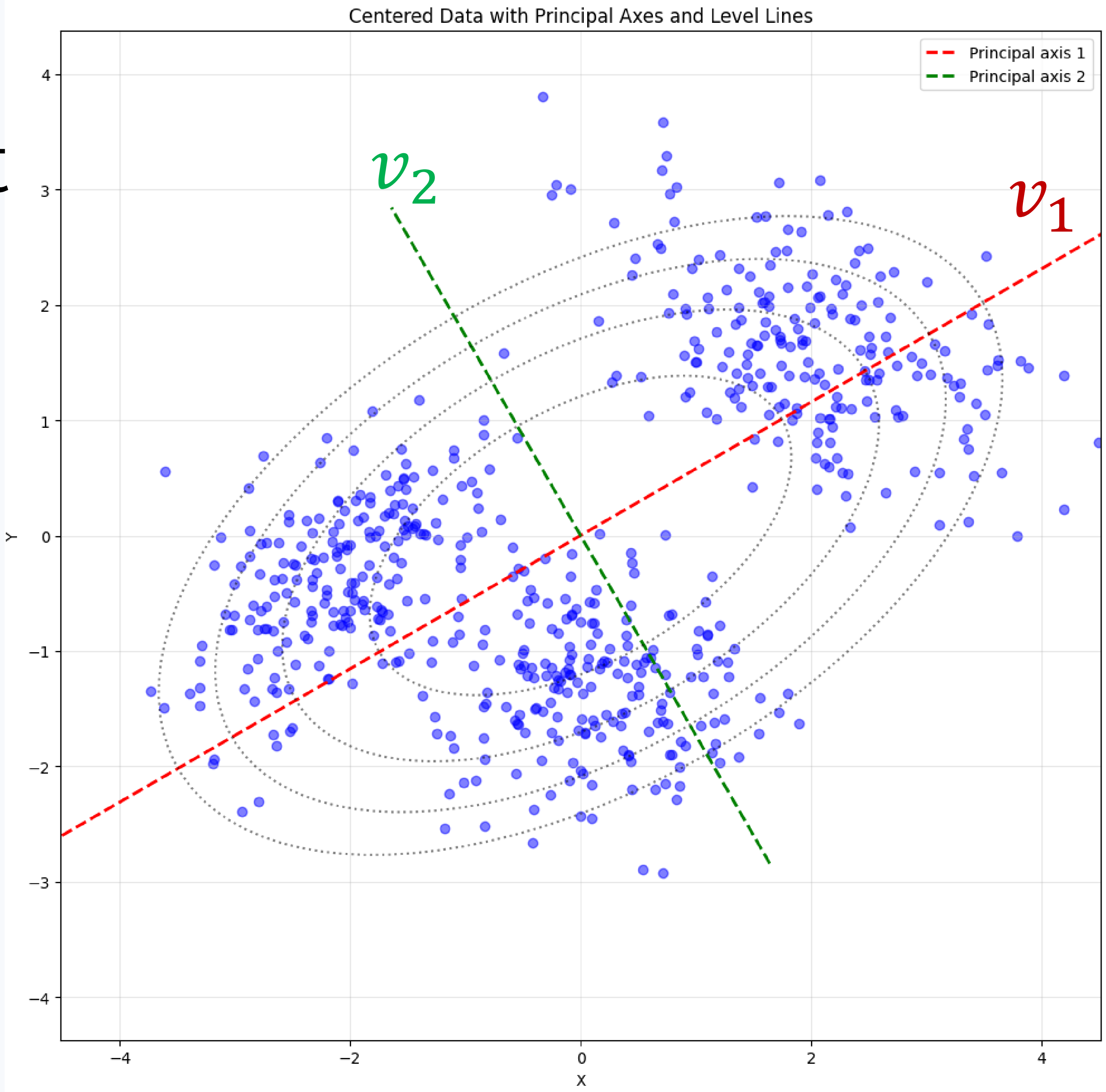
Principal component analysis

$$X = U\Sigma V^T$$

Here, V^T mixes features. We may take new feature description:

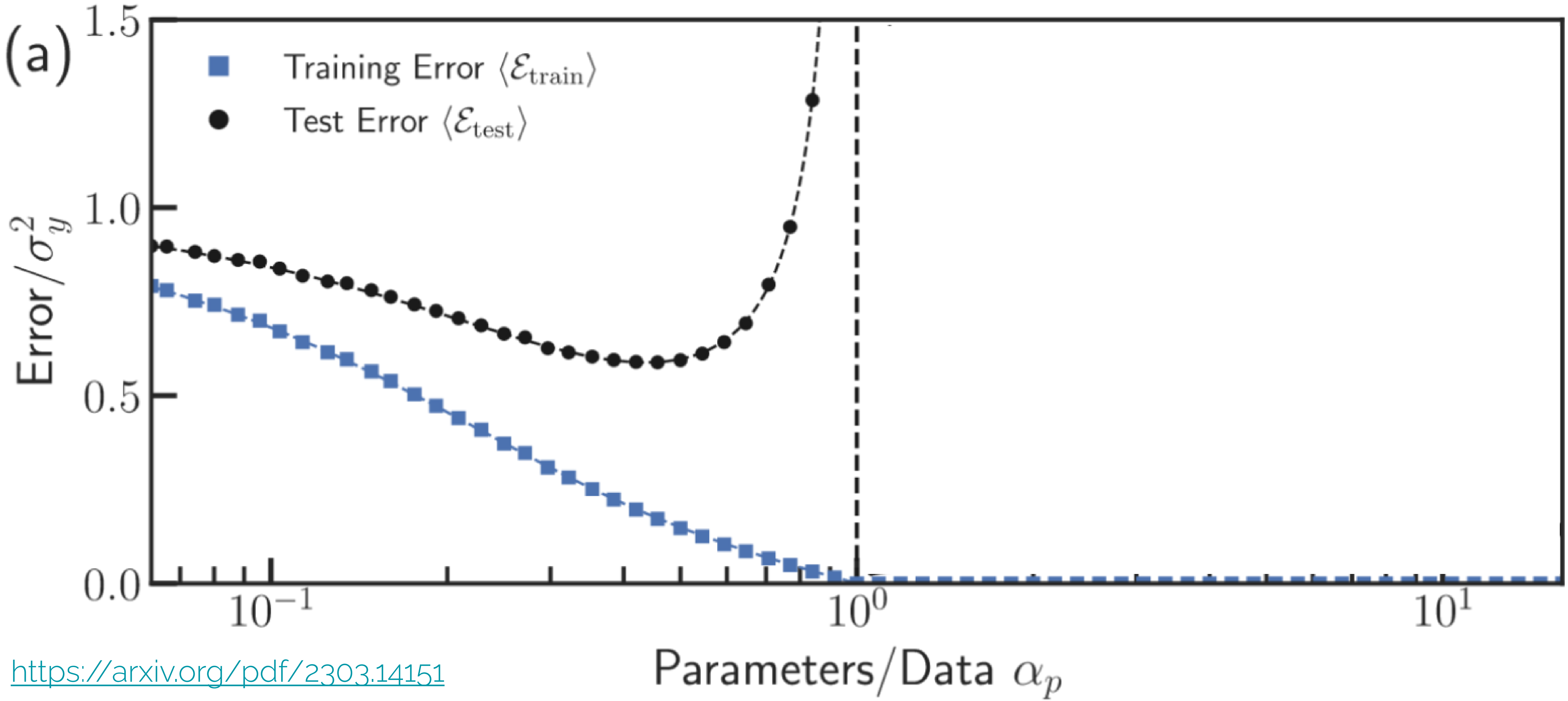
$$X \to U\Sigma$$

i-th object will be described by $(\sigma_i u_{i1}, \sigma_i u_{i2}, \dots)$



Complexity and overfitting

A classical view on complexity vs generalization



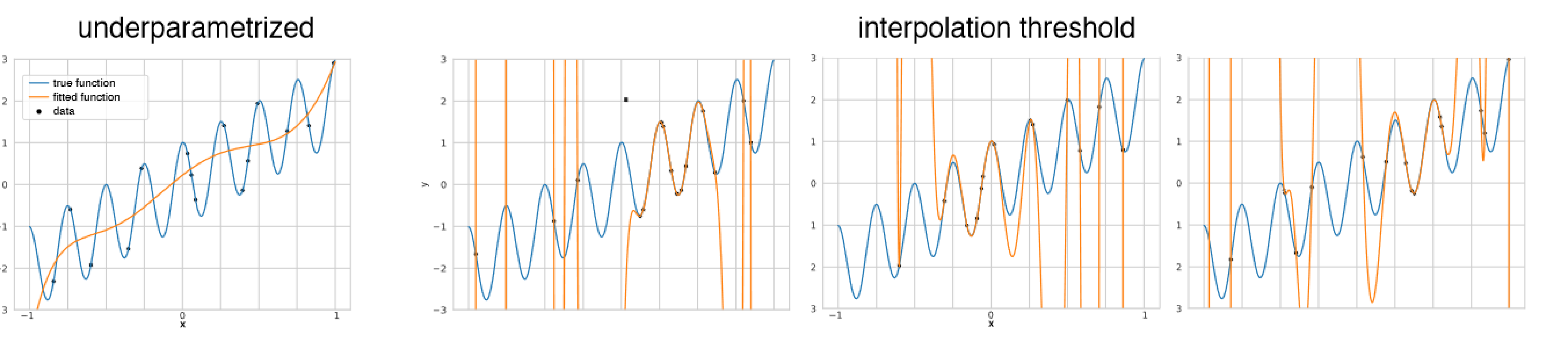
https://arxiv.org/pdf/2303.14151

For 1d linear regression

N points, D functions.
When $D \leq N$:

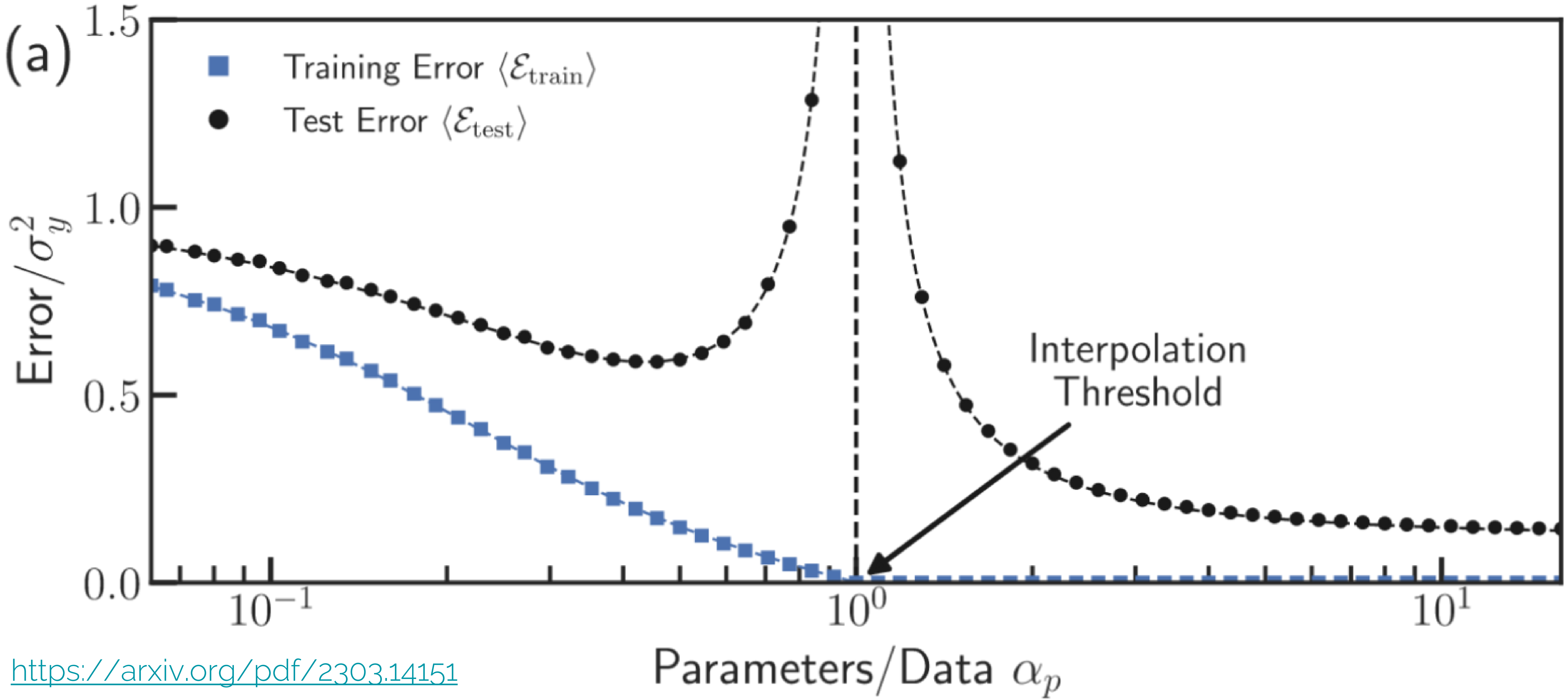
$$\hat{w}^T = (X^T X)^{-1} X^T y$$

At $D = N$, it's. **interpolation**.



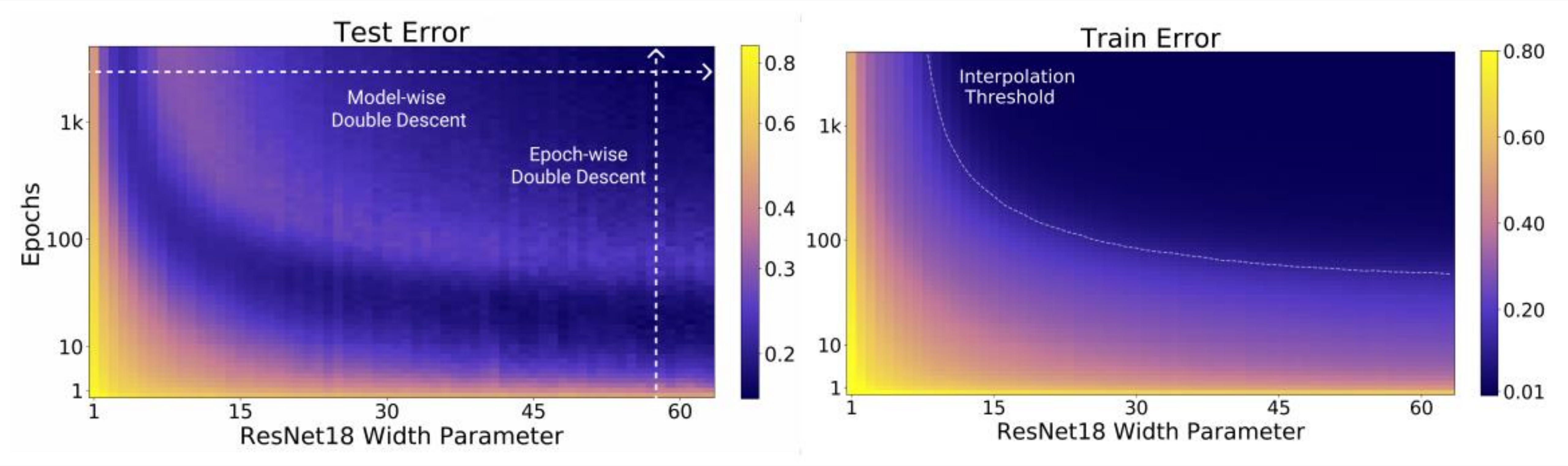
https://arxiv.org/pdf/2303.14151

Double descent



https://arxiv.org/pdf/2303.14151

Double descent



https://www.lesswrong.com/posts/FRv7ryoqtvSuqBxuT/understanding-deep-double-descent
https://openai.com/index/deep-double-descent/

Underparametrized vs overparametrized setting

N points, D functions.
When $D \leq N$:

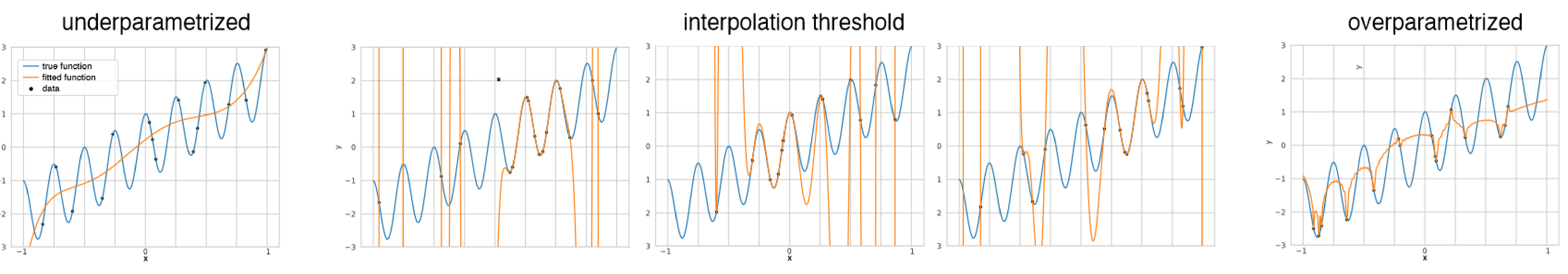
$$\hat{w}^T = (X^T X)^{-1} X^T y$$

At $D = N$, it's. **interpolation**.

When $D > N$:

$$\begin{cases} \|w\|_2^2 \to \min \\ Xw^T = y \end{cases}$$

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



https://arxiv.org/pdf/2303.14151