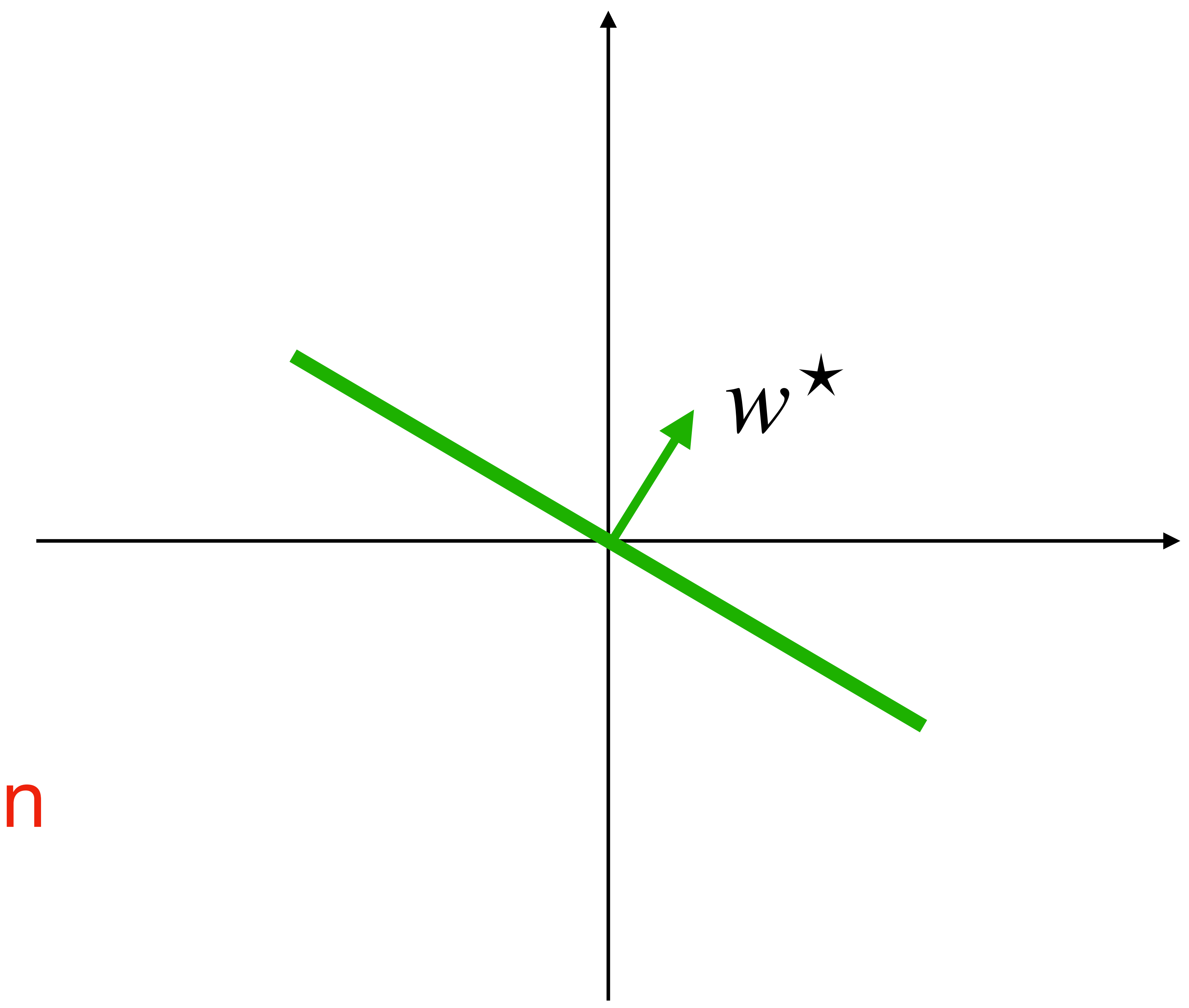
Optimization:
Adaptive Gradient Descent

Announcements:

P2 (NB) has been released;
HW3 is coming out this afternoon

Recap on Logistic Regression

LR directly models the label generation process:
$P(y \mid x) = 1/(1 + \exp(-y(x^\top w^\star)))$



Q: what the LR model will do for a point on the hyperplane?

Recap on Logistic Regression

Apply the MLE / MAP principles:

$$\hat{w} := \arg\min_{w} \underbrace{\sum_{i=1}^{n} \ln\left[1 + \exp\left(-y_i(w^\top x_i)\right)\right] + \lambda \|w\|_2^2}_{:=\ell(w)}$$

Unfortunately, no closed-form solution, needs to use optimization techniques

Objective

Understand the State-of-art algorithms — adaptive gradient descent

Outline for Today

1. Gradient Descent (continued)

2. Adaptive Gradient Descent

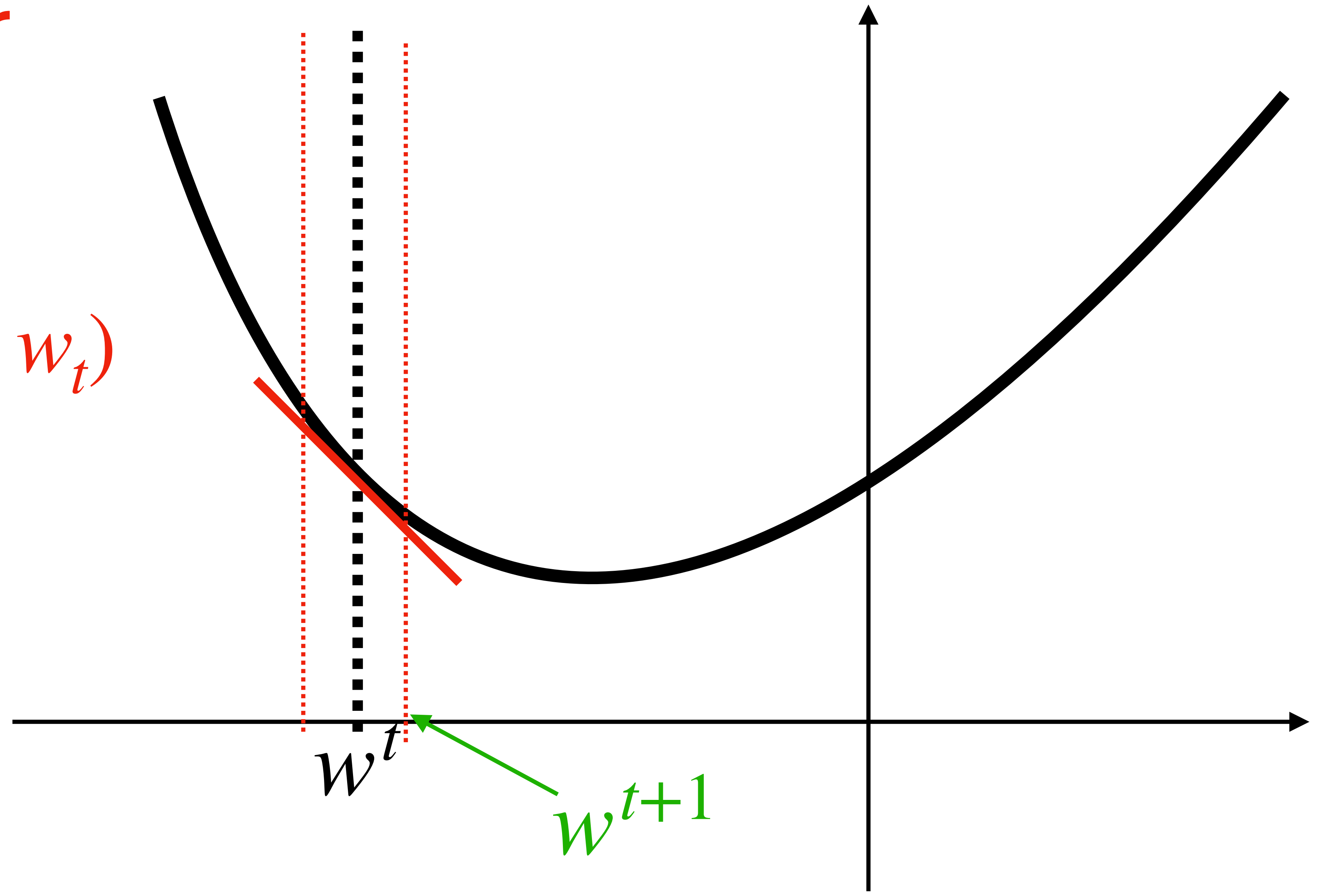
Gradient Descent

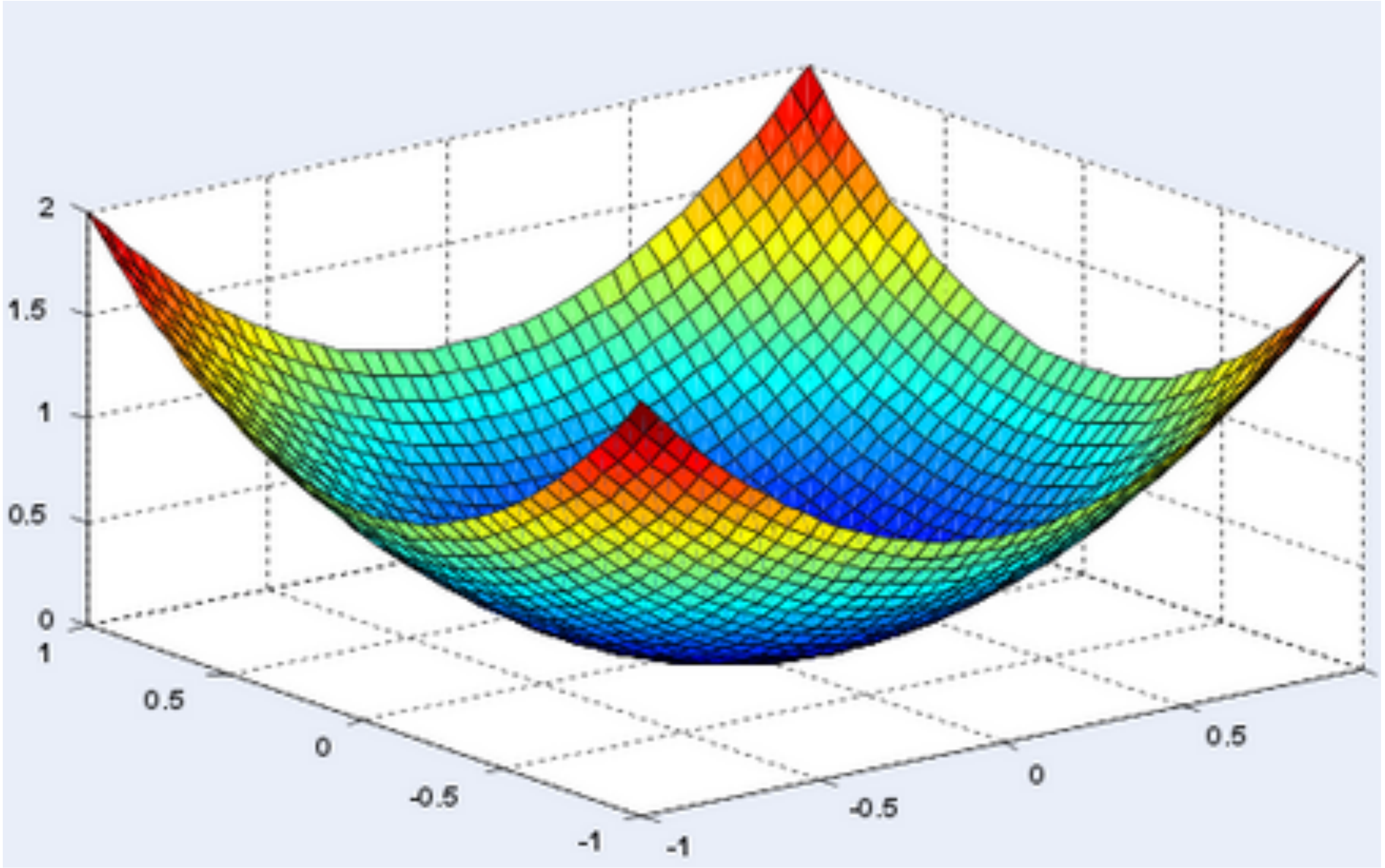
Gradient descent is a general technique that can minimize a function

$$w^{t+1} = w^t - \eta \nabla \ell(w)\,\big|_{w=w_t}$$

First-order Taylor expansion at w_t:

$\ell(w_t) + \nabla \ell(w_t)^\top (w - w_t)$





Gradient Descent

GD can decrease loss every time step w/ small learning rate

$$\ell(w^t + \delta) \approx \ell(w^t) + \nabla \ell(w^t)^\top \delta$$

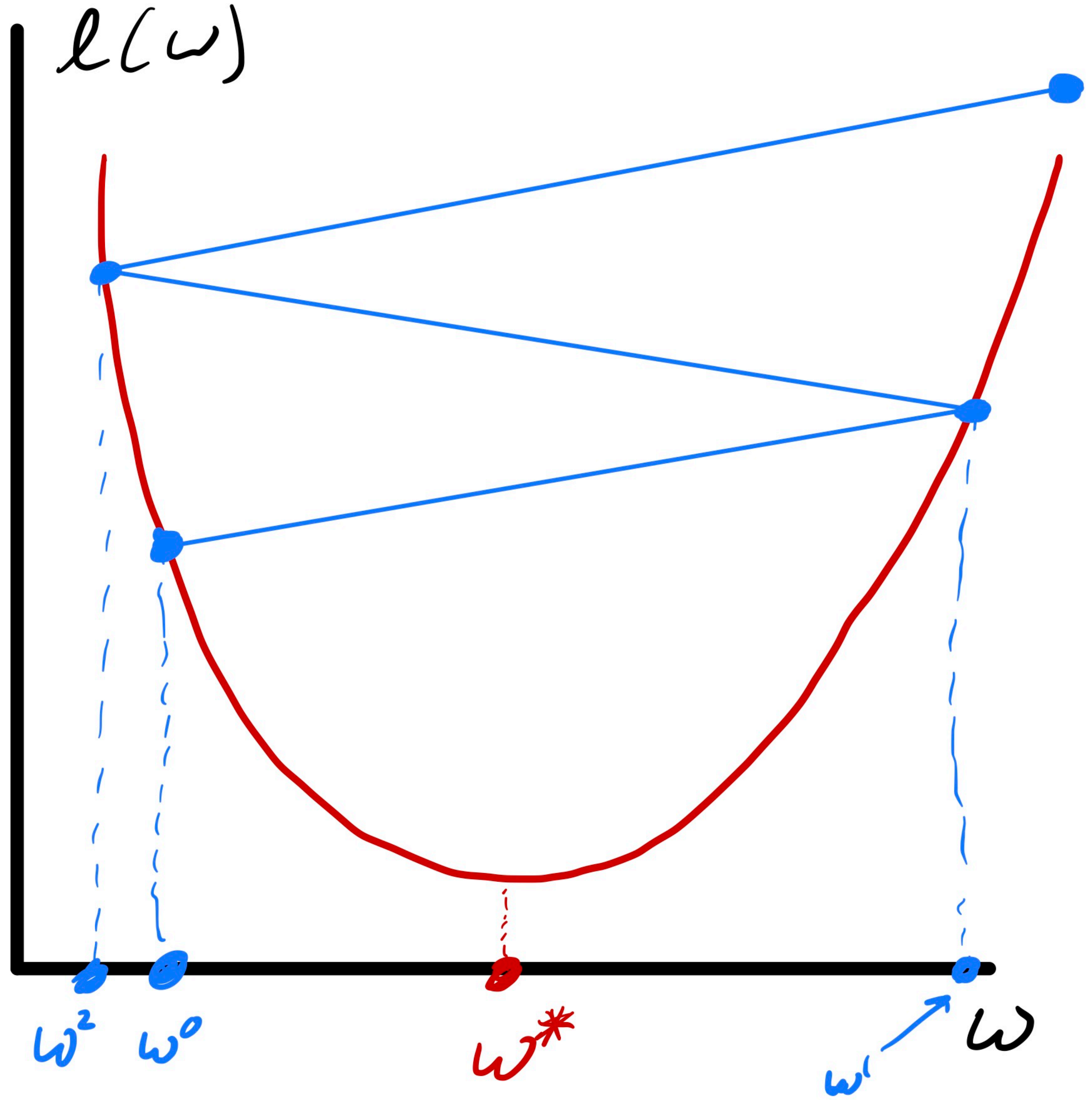
Q: Which direction δ should point to in order to minimize the linear approximation?

Set $\delta = -\eta \nabla \ell(w^t)$ (w/ small η), we have:

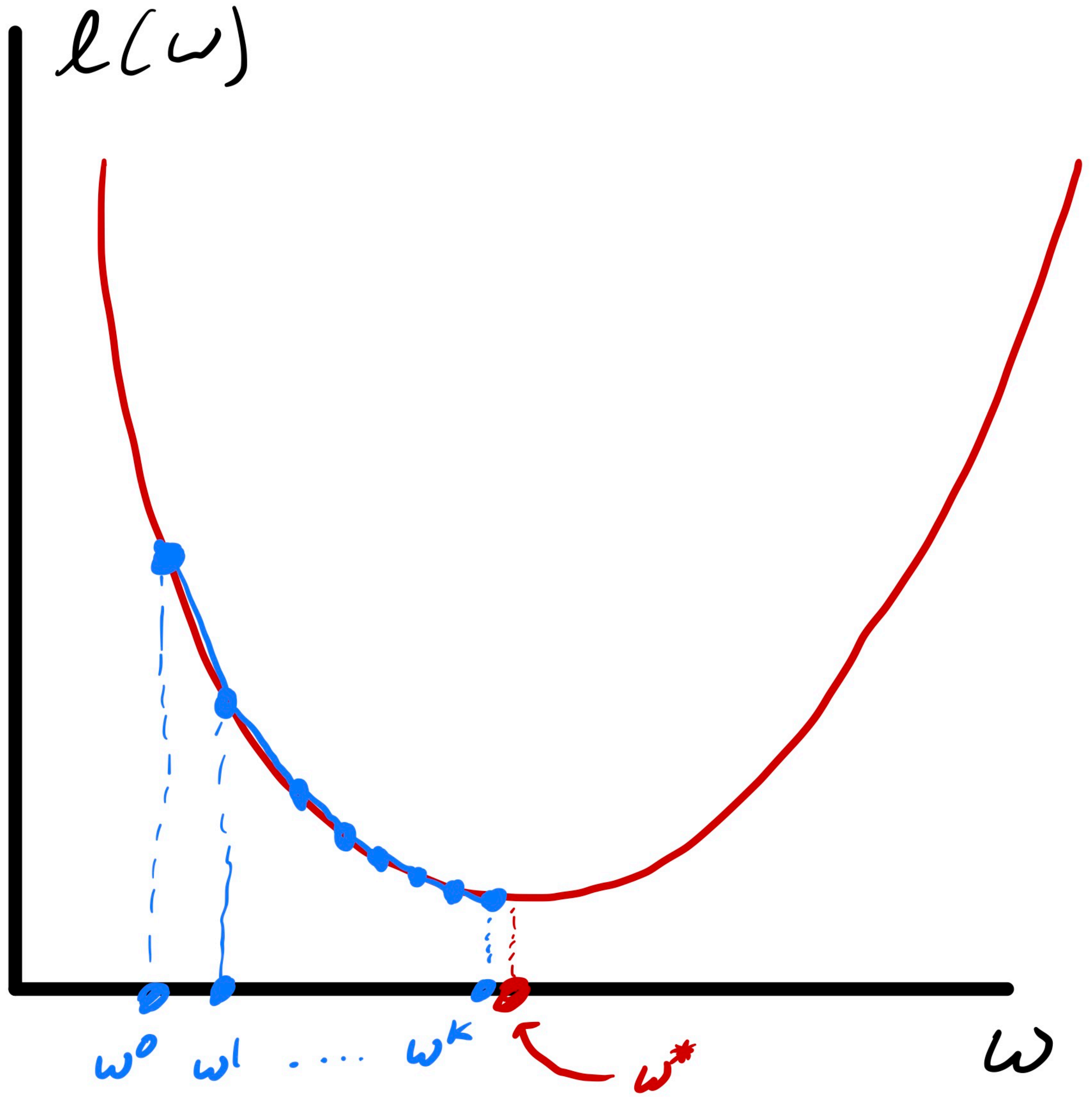
$$\ell(w^t - \eta \nabla \ell(w^t)) \approx \ell(w^t) - \eta(\nabla \ell(w^t))^\top \nabla \ell(w^t) < \ell(w^t)$$

How to set learning rate η in practice?

Large η typically is bad and can lead to diverge



In theory, for convex loss, $\eta = c/\sqrt{k}$ guarantees convergence (1/k also works, but slower)



Let's summarize by applying GD to logistic regression

Recall the objective for LR:

$$\min_{w} \sum_{i=1}^{n} \ln\left[1 + \exp\left(-y_i(w^\top x_i)\right)\right] + \lambda \|w\|_2^2$$

Initialize $w^0 \in \mathbb{R}^d$

Iterate until convergence:

1. Compute gradient $g^t = \sum_i \frac{\exp(-y_i x_i^\top w^t)(-y_i x_i)}{1 + \exp(-y_i x_i^\top w^t)} + 2\lambda w^t$
2. Update (GD): $w^{t+1} = w^t - \eta g^t$

Outline for Today

1. Gradient Descent (continued)

2. Adaptive Gradient Descent

Potential Issue of Gradient Descent

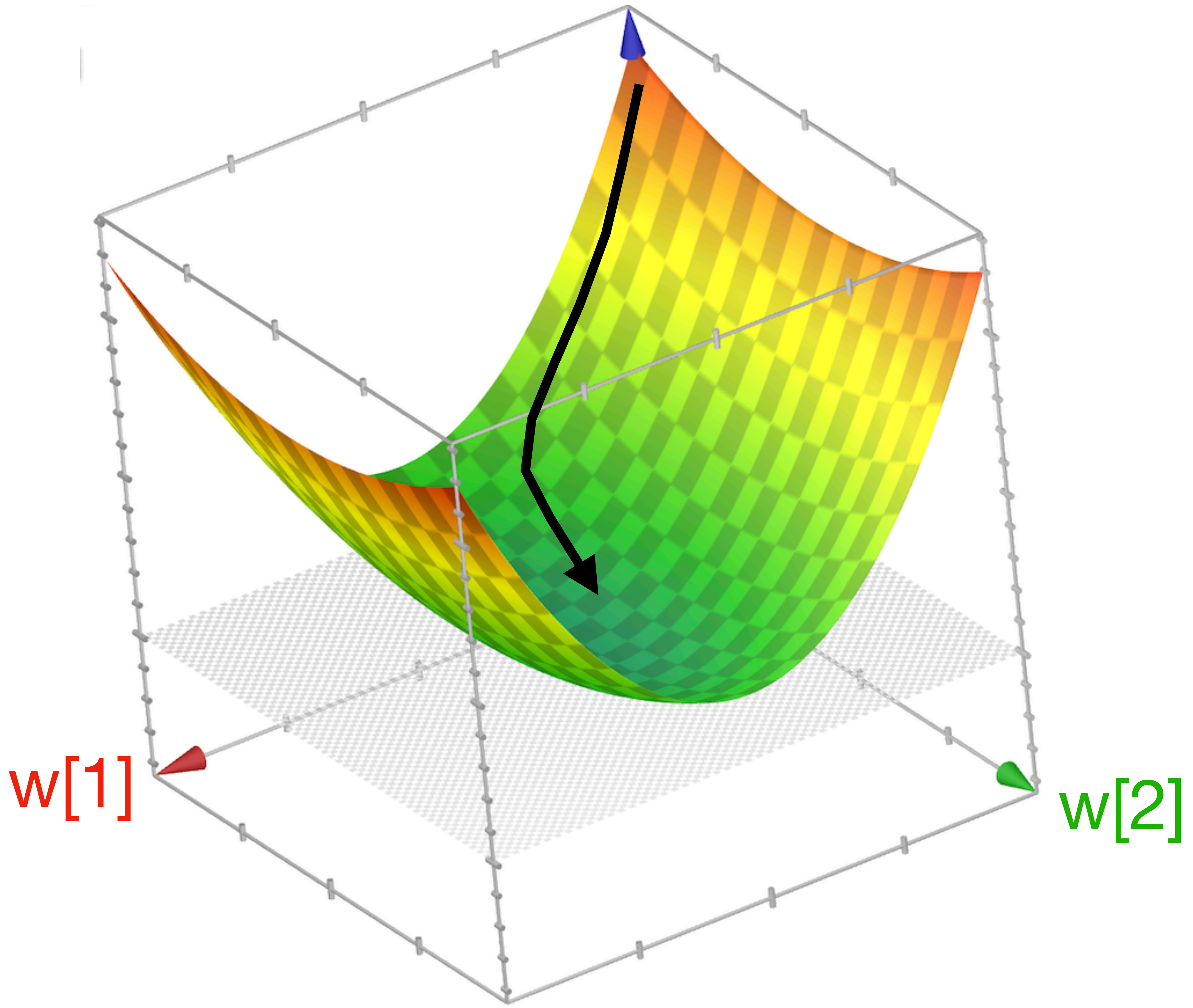
$$w^{t+1} = w^t - \eta \nabla \ell(w)\,|_{w=w^t}$$

It uses the same learning rate η for all dimension

Consider a function
$$\ell(w) = w[1]^2 + 0.1w[2]^2$$

$$\nabla \ell(w) = \begin{bmatrix} 2w[1] \\ 0.2w[2] \end{bmatrix}$$

Q: what the GD path would look like?



Adaptive Gradient Descent (AdaGrad)

Key idea: make learning rate dependent on dim, and update it during optimization

For each dim $j \in [d]$:

$$z^t[j] = \sum_{i=1}^{t} (g^t[j])^2$$

Update the j-th coordinate as follows:

$$w^{t+1}[j] = w^t[j] - \frac{\eta}{\sqrt{z^t[j] + \epsilon}} g^t[j]$$

A dim-dependent adaptive learning rate!

Adaptive Gradient Descent (AdaGrad)

Put everything together (vectorized form)

Initialize $w^0 \in \mathbb{R}^d$, $z^0 = 0$

While not converged:

Compute $g^t = \nabla \ell(w)\,|_{w=w^t}$

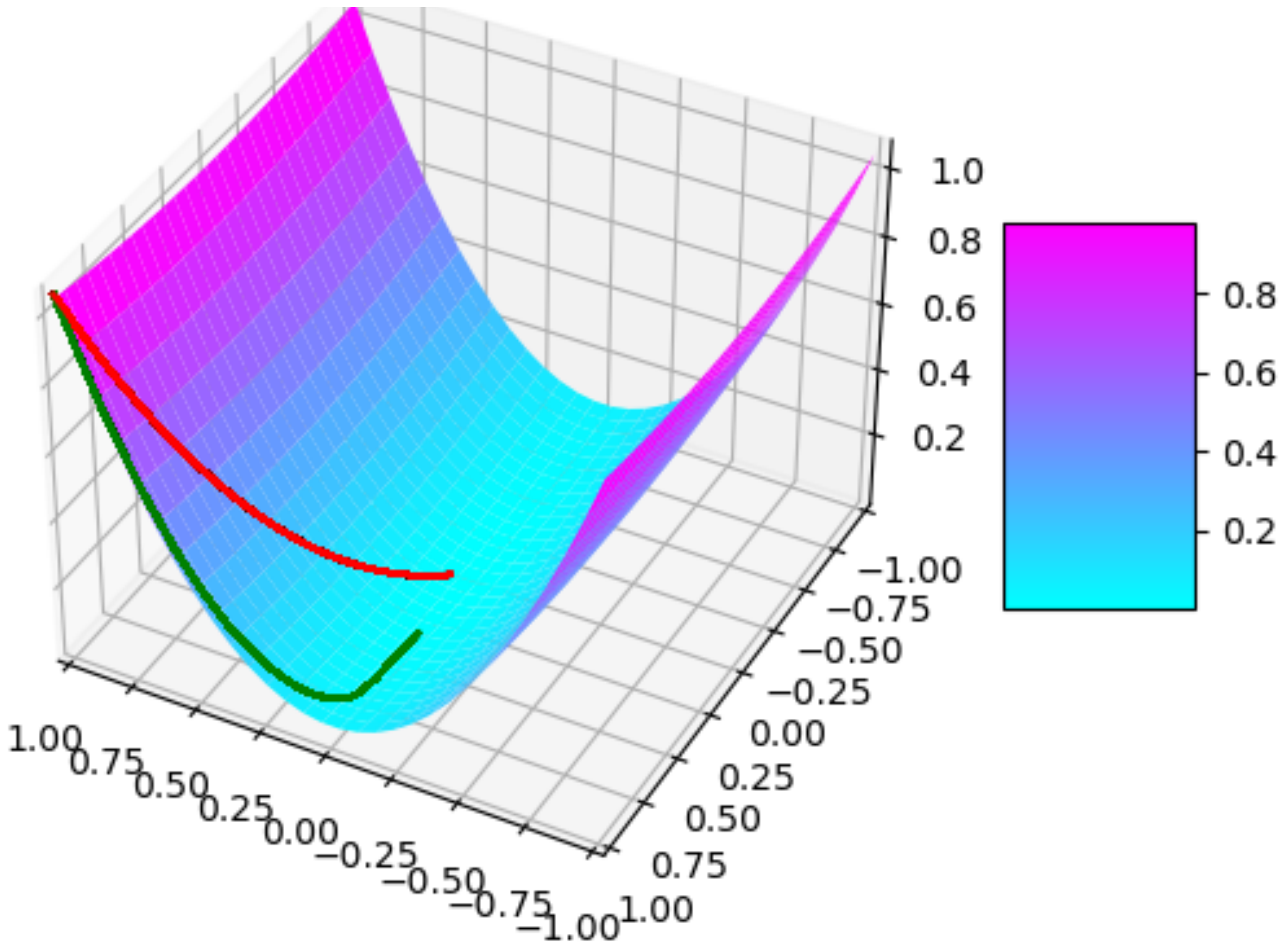
Compute $z^t = z^{t-1} + g^t * g^t$

Update $w^{t+1} = w^t - \eta \cdot \mathrm{diag}(1/\sqrt{z^t})g^t$

Visualization of AdaGrad VS GD

Demo:

$$\ell(w) = w[1]^2 + 0.01w[2]^2$$



AdaGrad can make good progress on all axis

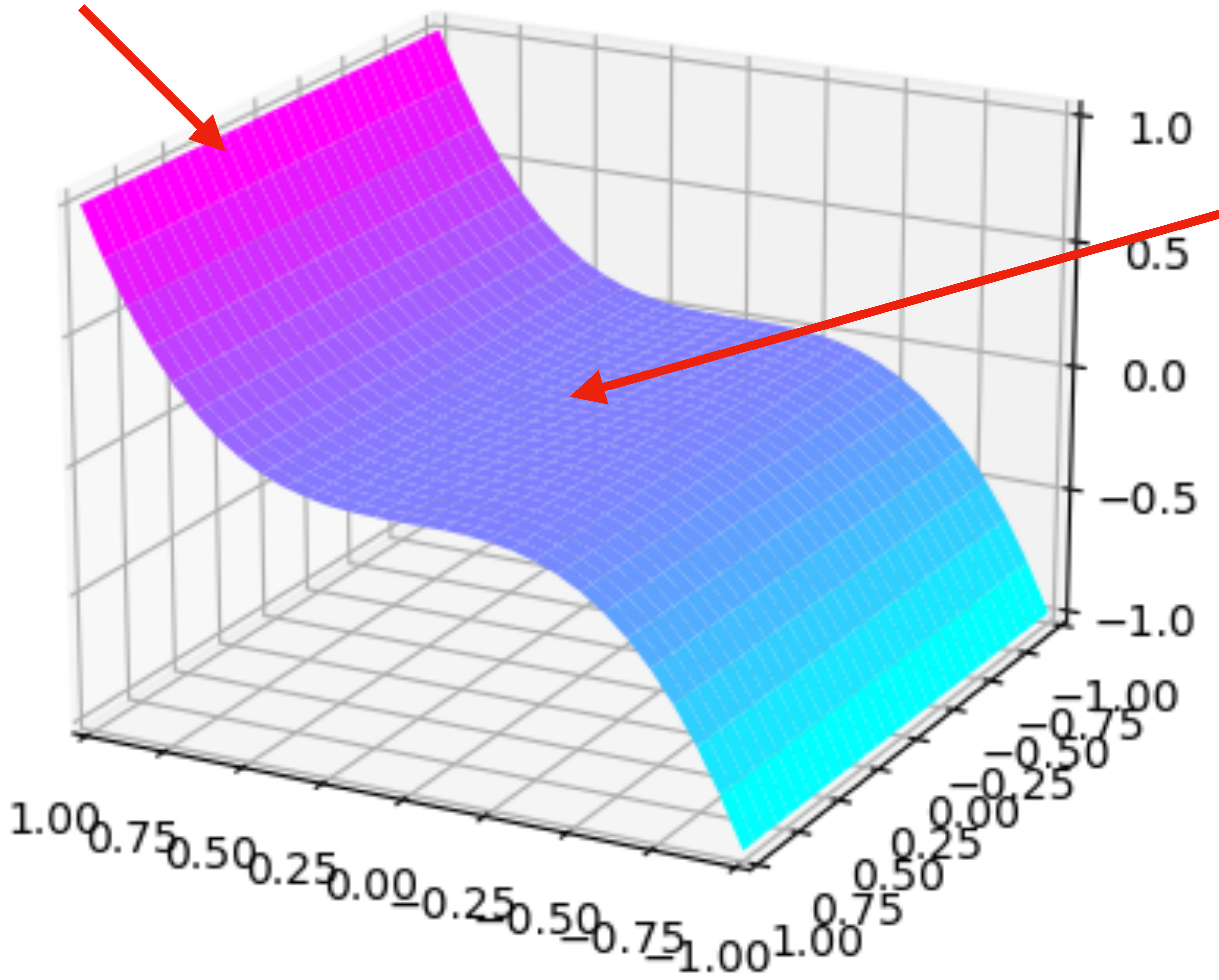
Issue of AdaGrad and GD

When the loss is non-convex, they both can get stuck at flat region (places where gradient is almost zero)

Q: what would happen if I drop a ball here

e.g., $x^3 + 0 \times y$

GD and Adagrad will get stuck here



Gradient Descent (GD) with Momentum

Possible solution to escape the flat gradient is to use ***momentum***

(The idea is motivated from physics)

Think about gradient g^t as "acceleration",
we estimate the "velocity" via:

$$v^t = \alpha v^{t-1} + (1-\alpha) g^t$$

$$(v^t = \alpha^{t-1}(1-\alpha)g^1 + \alpha^{t-2}(1-\alpha)g^2 + \ldots + \alpha(1-\alpha)g^{t-1} + (1-\alpha)g^t)$$

Exponential average

Gradient Descent with Momentum

Putting things together:

Initialize $w^1 \in \mathbb{R}^d$, $v^0 = 0$

For t = 1 ….

Compute $g^t = \nabla \ell(w)\,|_{w=w^t}$

Compute momentum $v^t = \alpha v^{t-1} + (1-\alpha)g^t$

Update $w^{t+1} = w^t - \eta v^t \cdot \dfrac{1}{1-\alpha^t}$

Adam (Adaptive Momentum Estimation)

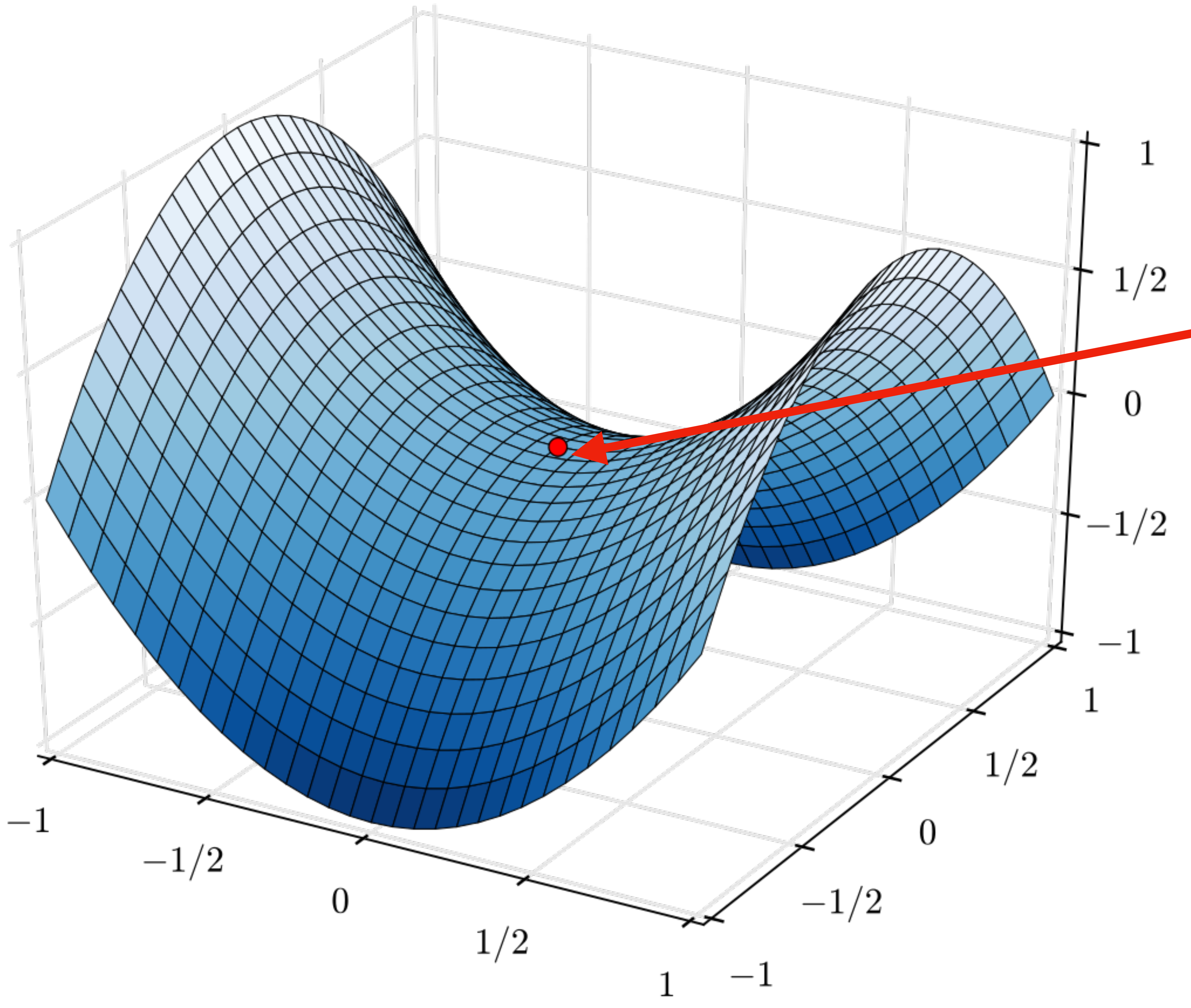
Adam is the most widely used optimizer for training neural network today!

Adam = Momentum + AdaGrad

(The second paper reading quiz)

Even w/ AdaGrad + Momentum, we may still have issue

e.g., saddle point $x^2 - y^2$



Can stuck at the saddle point

We will revisit this next Tuesday

Summary

Gradient-based optimization methods:

GD: simply follow the negative of the gradient

AdaGrad — each dim has its own learning rate, adapted based on the cumulation of the past squared derivatives — help make progress along all axises.

GD w/ momentum: think about gradient as “acceleration”, “velocity” is the exponential average of “acceleration” — help power through very flat region

Adam: Momentum + AdaGrad