Optimization:

Adaptive Gradient Descent

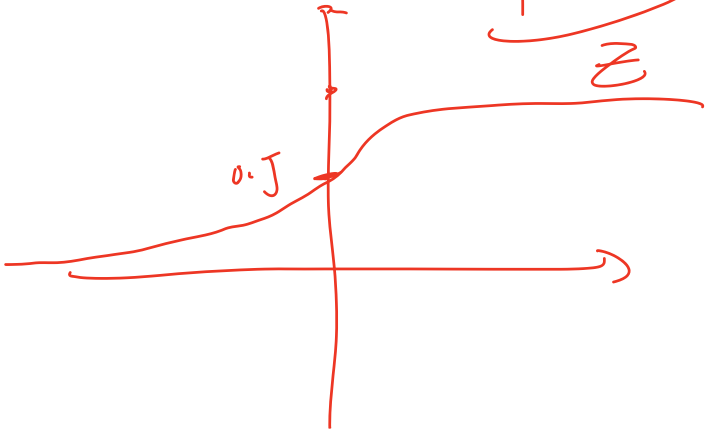
Announcements:

P~~2~~3 (NB) has been released;
HW3 is coming out this afternoon

Recap on Logistic Regression

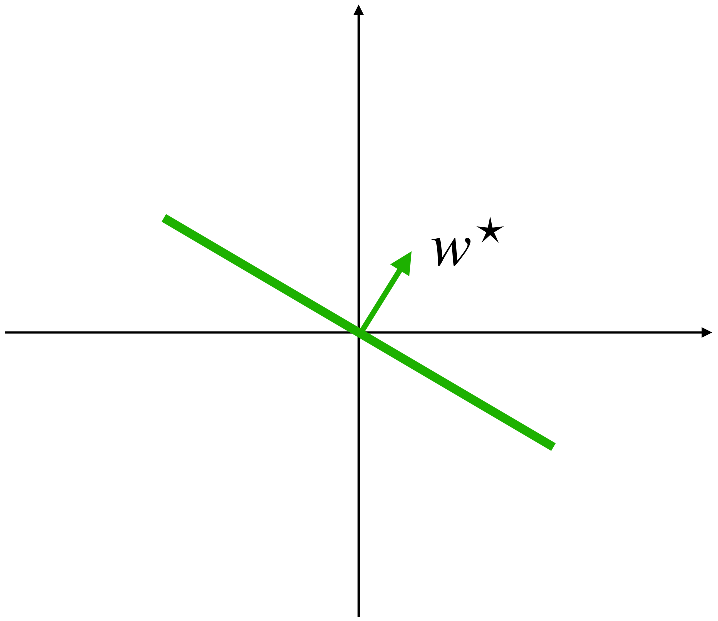
LR directly models the label generation process:

$$P(y \mid x) = 1/(1 + \exp(-\underbrace{y(x^\top w^\star)}_{z}))$$



Recap on Logistic Regression

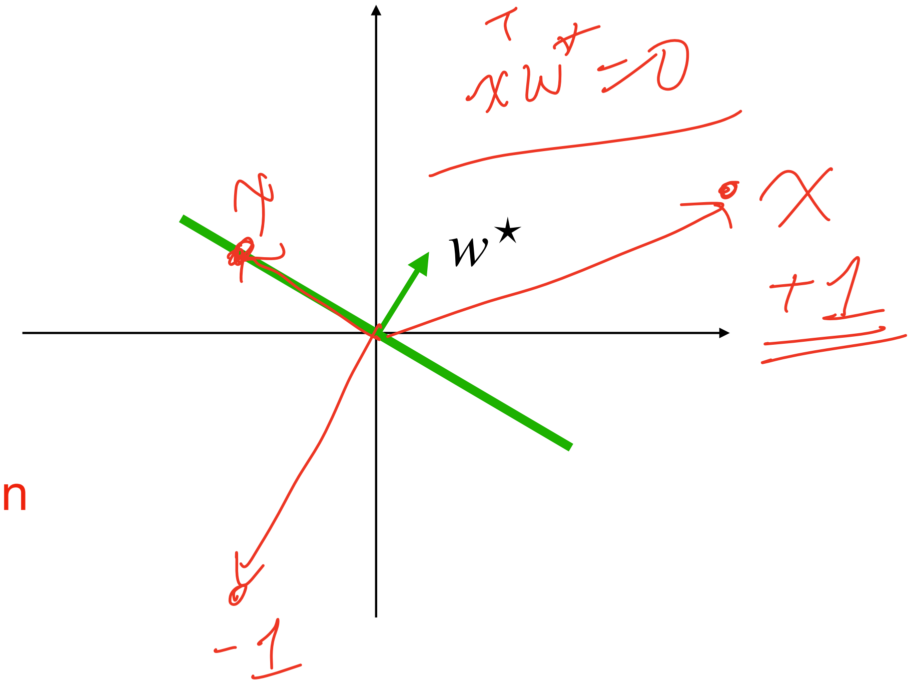
LR directly models the label generation process:
$P(y \mid x) = 1/(1 + \exp(-y(x^\top w^\star)))$



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:

NLL

prior

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

$\lambda > 0$

$\arg\min_w \ell(w)$

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

Gradient Descent

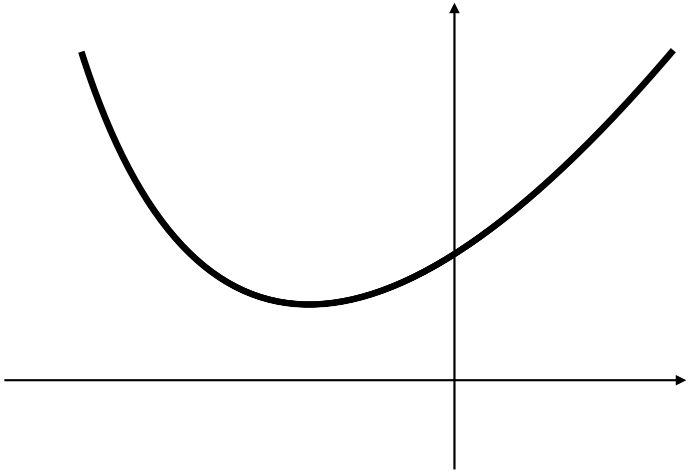
Gradient descent is a general technique that can minimize a function

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

Gradient Descent

Gradient descent is a general technique that can minimize a function

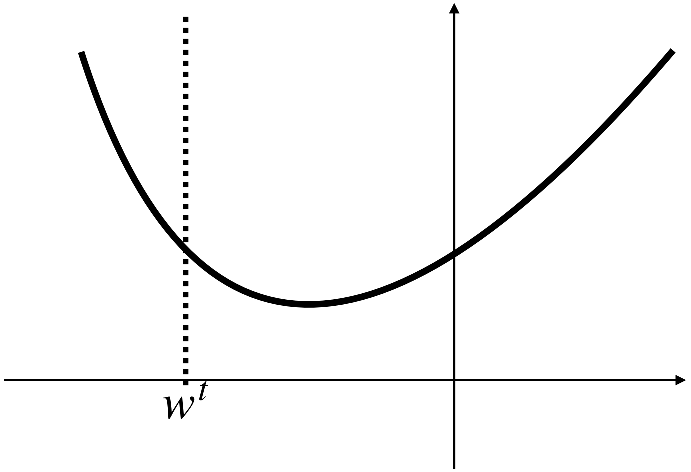
$$w^{t+1} = w^t - \eta \nabla \ell(w)\,\big|_{w=w_t}$$



Gradient Descent

Gradient descent is a general technique that can minimize a function

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



Gradient Descent

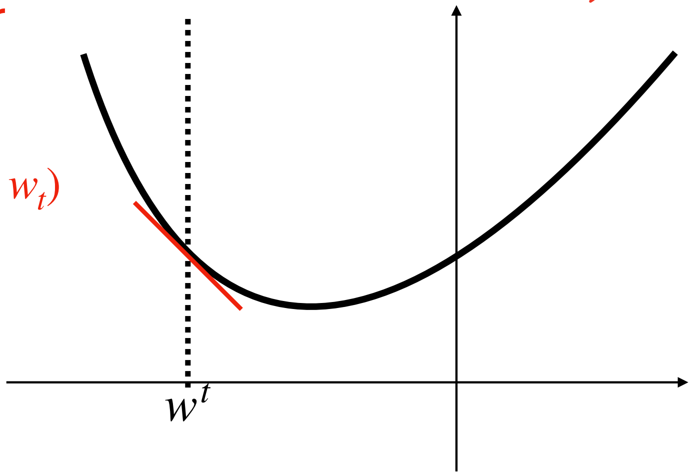
Gradient descent is a general technique that can minimize a function

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

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

First-order Taylor expansion at w_t:

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



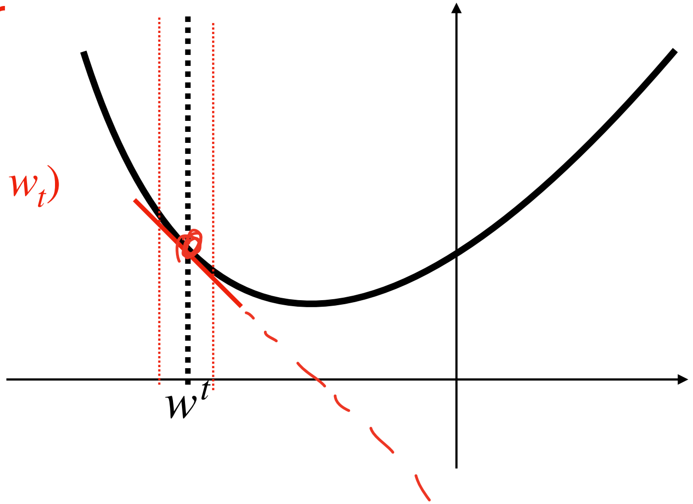
Gradient Descent

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:

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



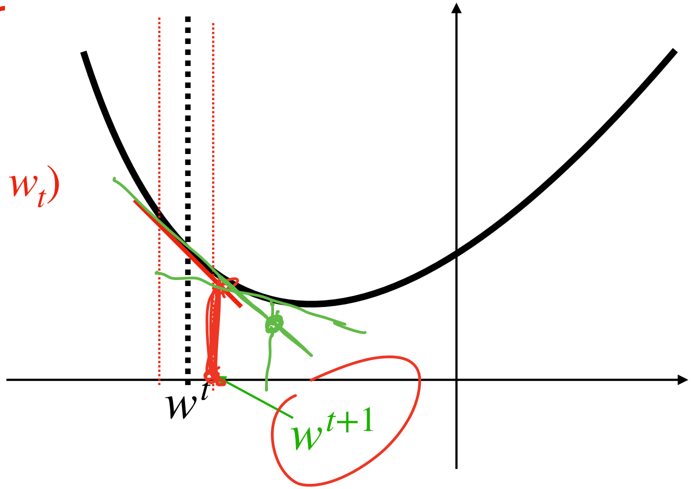
Gradient Descent

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

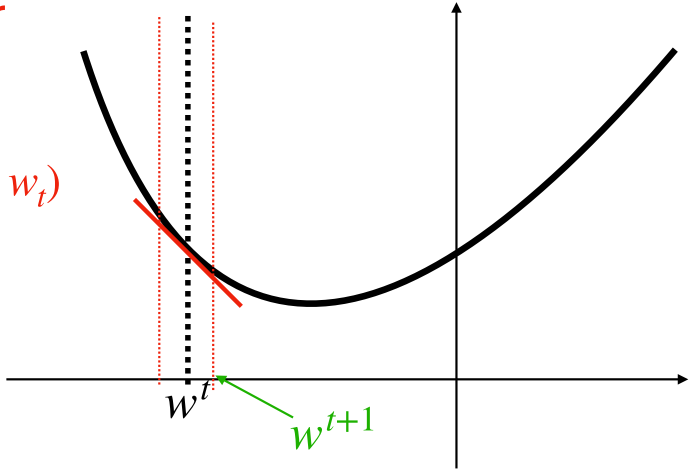
Gradient descent is a general technique that can minimize a function

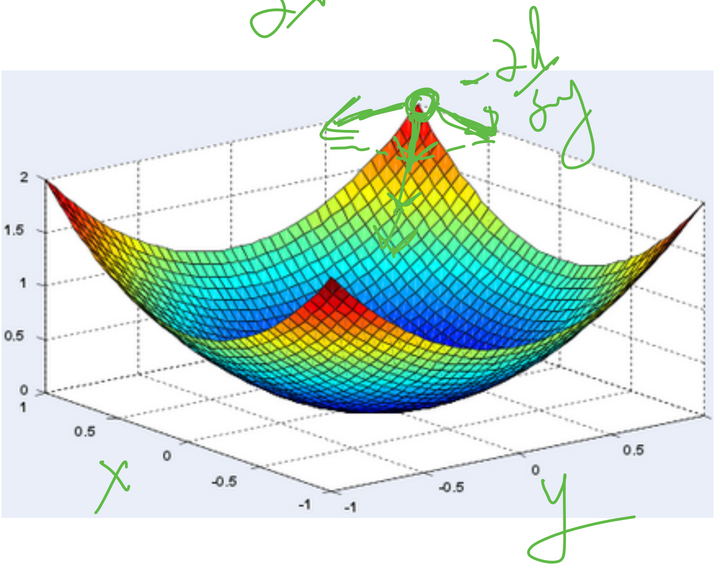
$$w^{t+1} = w^t - \eta \nabla \ell(w)\,\big|_{w=w_t}$$

$-\frac{\partial l}{\partial x}$

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

Gradient Descent

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

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

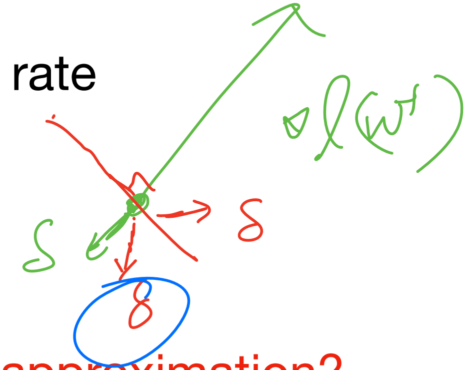
0

$\|\delta\| \approx 0$

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?

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:

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:

$$\ell(\underbrace{w^t - \eta \nabla \ell(w^t)}_{w^{t+1}}) \approx \ell(w^t) - \underbrace{\eta}_{>0}\underbrace{(\nabla \ell(w^t))^\top \nabla \ell(w^t)}_{>0} < \ell(w^t)$$

δ

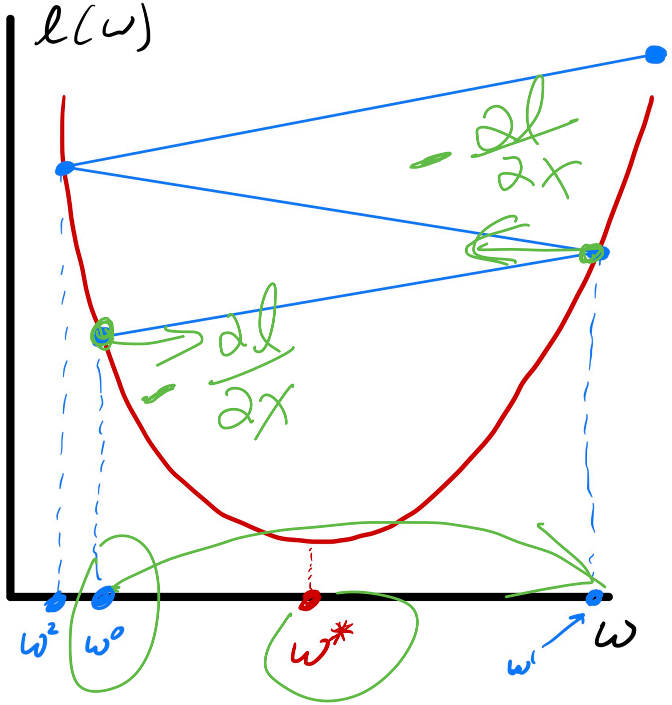
$\nabla f(w_t) \neq 0$

How to set learning rate η in practice?

Large η typically is bad and
can lead to diverge

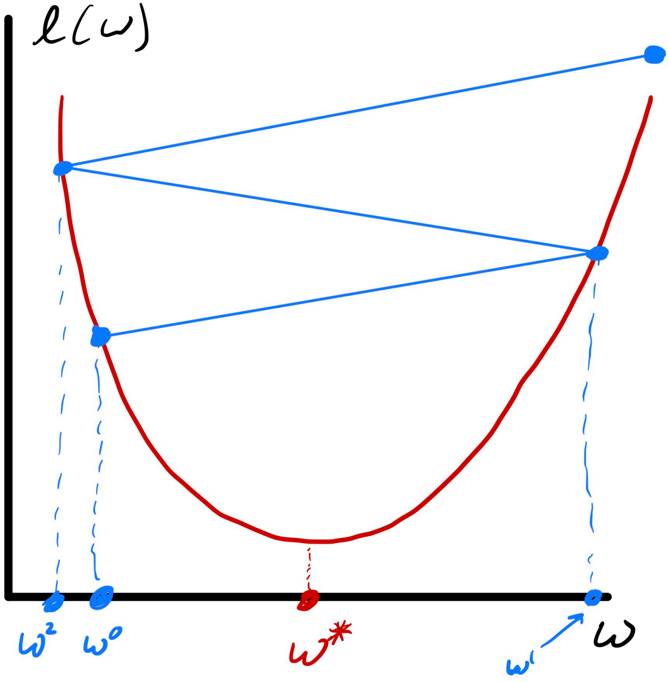
How to set learning rate η in practice?

Large η typically is bad and
can lead to diverge



How to set learning rate η in practice?

Large η typically is bad and can lead to diverge

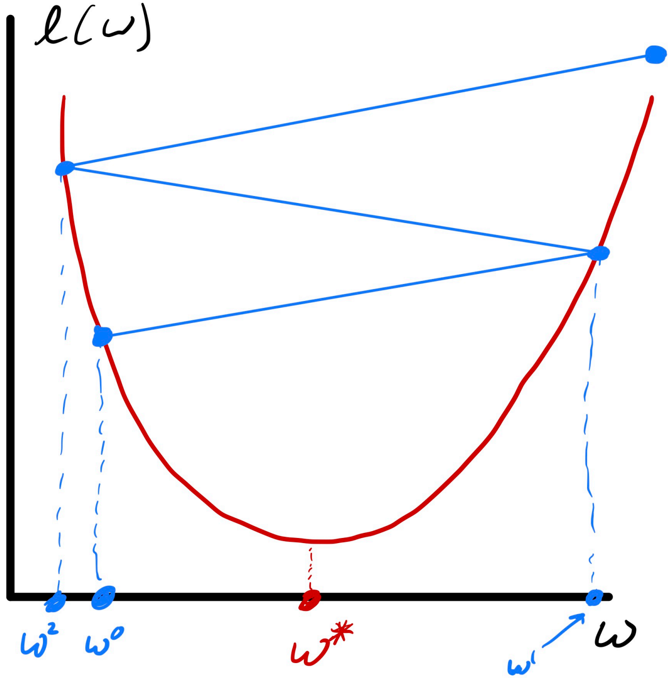


In theory, for convex loss, $\eta = c/\sqrt{k}$ guarantees convergence (1/k also works, but slower)

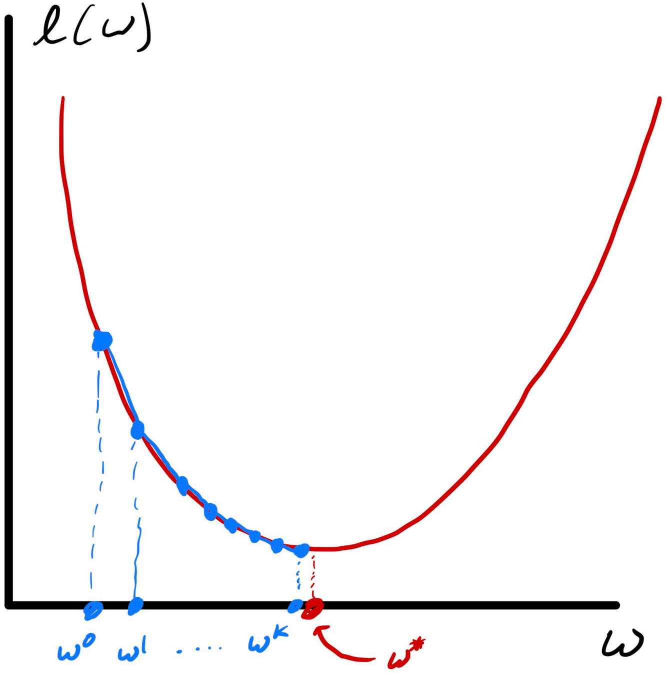
k is the current Iteration

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:

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

$\nabla \ell(w)$

$\nabla(x^T x) = 2x$

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

Iterate until convergence:

$$\sum_{i=1}^{n} \frac{\exp(-y_i\, w^T x_i)(-y_i x_i)}{1+\exp(-y_i(w^T x_i))} + 2\lambda \cdot w$$

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:

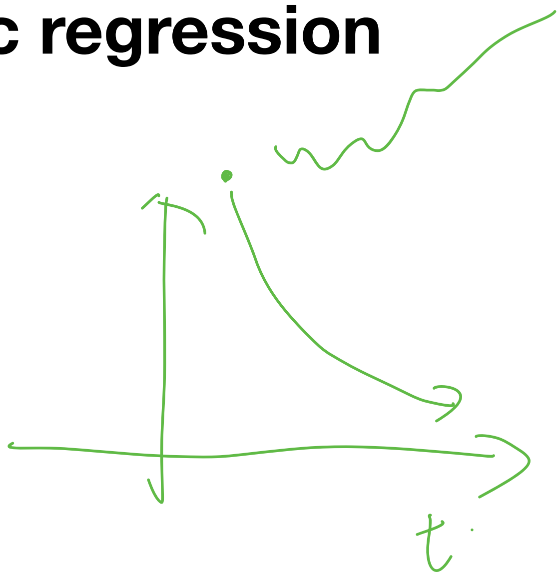
1. Compute gradient $g^t = \sum_i \frac{\exp(-y_i x_i^\top w^t)(-y_i x_i)}{1 + \exp(-y_i x_i^\top w^t)} + 2\lambda w^t$

Let's summarize by applying GD to logistic regression

Recall the objective for LR:

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

① $\|w^{t+1}-w^t\|_2 \le \delta$



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

Iterate until convergence:

② $\|g^t\|_2 \le \delta$

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): $w^{t+1} = w^t - \eta g^t$

$\frac{1}{\sqrt{t}}$

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

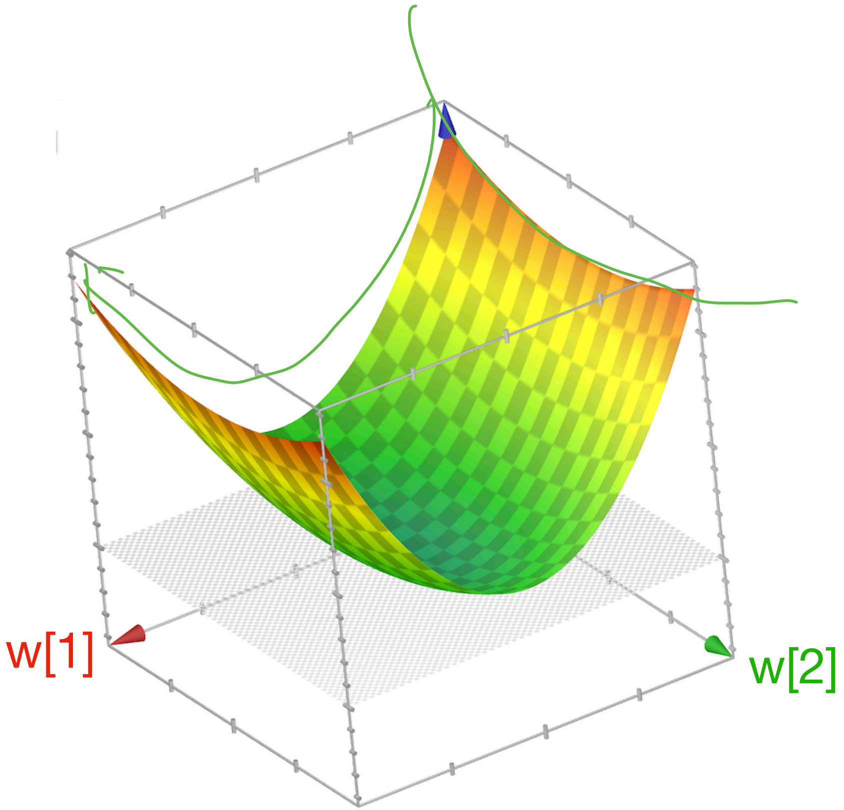
Potential Issue of Gradient Descent

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

It uses the same learning rate η for all dimension

Consider a function

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



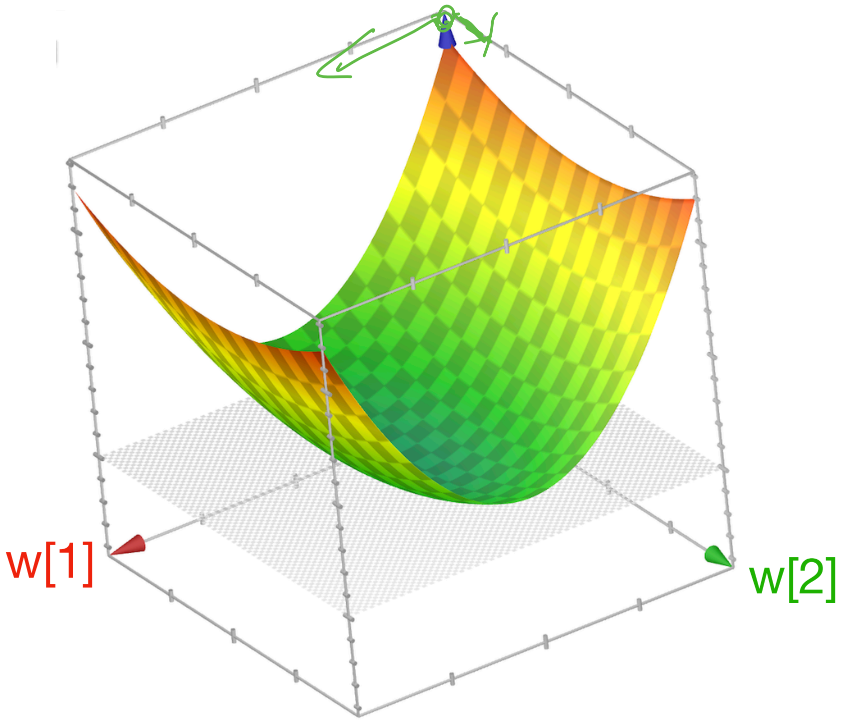
Potential Issue of Gradient Descent

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

It uses the same learning rate η for all dimension

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

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



Potential Issue of Gradient Descent

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

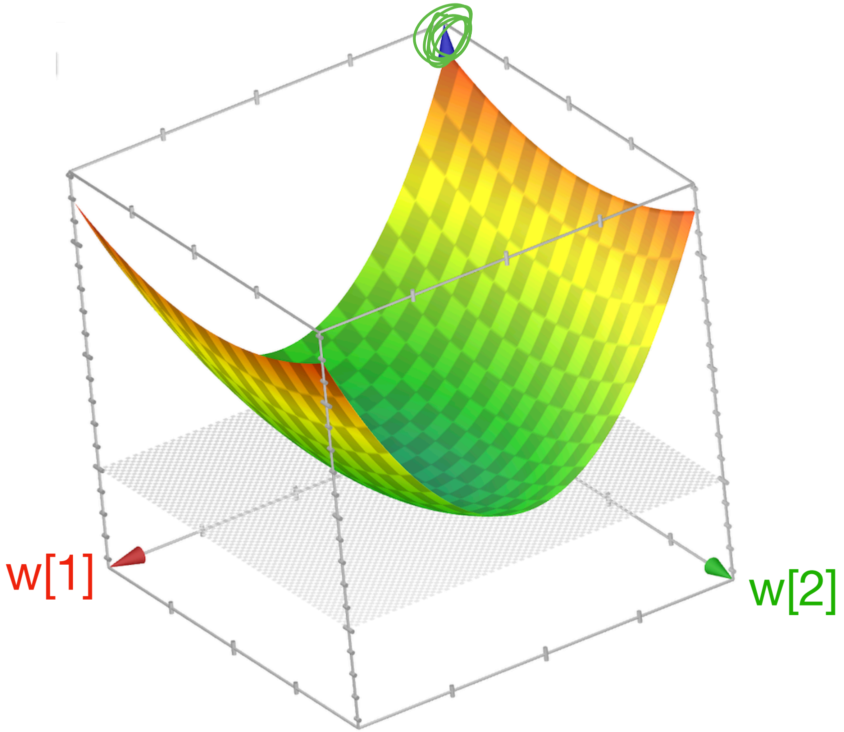
It uses the same learning rate η for all dimension

Consider a function

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

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

Q: what the GD path would look like?



Potential Issue of Gradient Descent

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

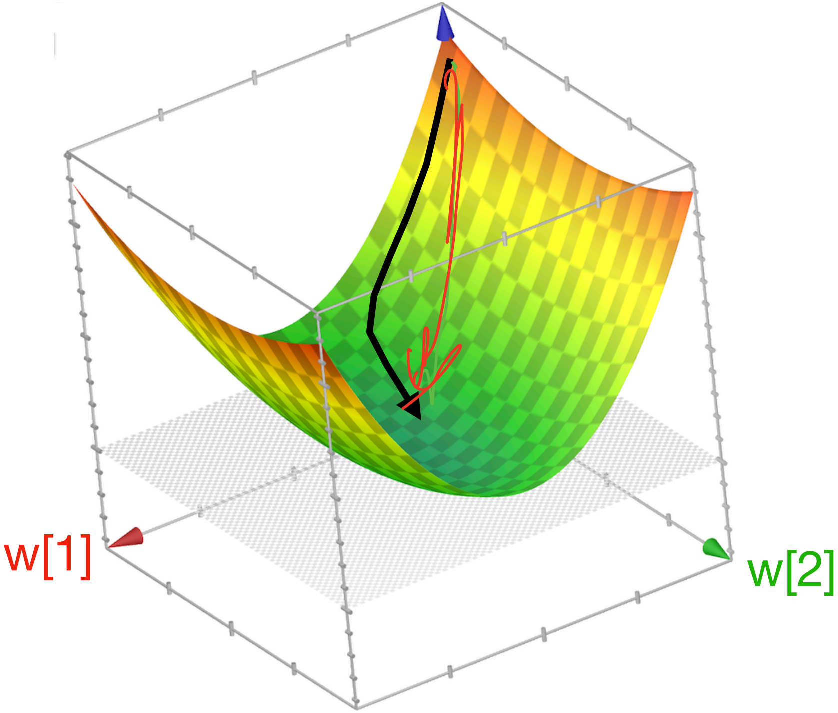
It uses the same learning rate η for all dimension

Q: what the GD path would look like?

Consider a function

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

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



Adaptive Gradient Descent (AdaGrad)

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

Adaptive Gradient Descent (AdaGrad)

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

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

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

$$g^t[j] = \frac{\partial \ell(w^t)}{\partial w^t[j]}$$

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:

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

1e-8

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

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

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

$$x * x = \begin{bmatrix} x[1]^2 \\ x[2]^2 \\ \vdots \\ x[d]^2 \end{bmatrix}$$

Adaptive Gradient Descent (AdaGrad)

Put everything together (vectorized form)

$$\frac{1}{\sqrt{x}} = \begin{bmatrix} \frac{1}{\sqrt{x(1)}} \\ \vdots \\ \frac{1}{\sqrt{x(d)}} \end{bmatrix}$$

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

$$\text{np.diag}(x) = \begin{bmatrix} x(1) & & 0 \\ & \ddots & \\ 0 & & x[d] \end{bmatrix}$$

Visualization of AdaGrad VS GD

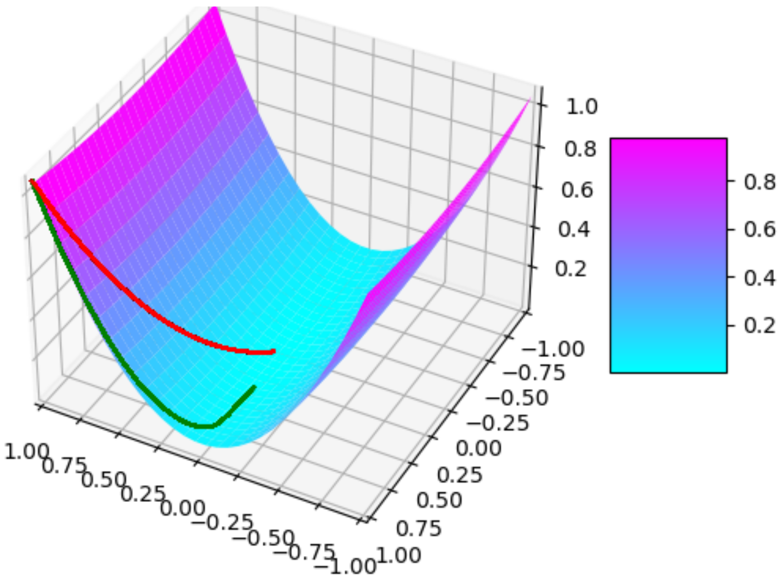
Demo:

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

Visualization of AdaGrad VS GD

Demo:

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

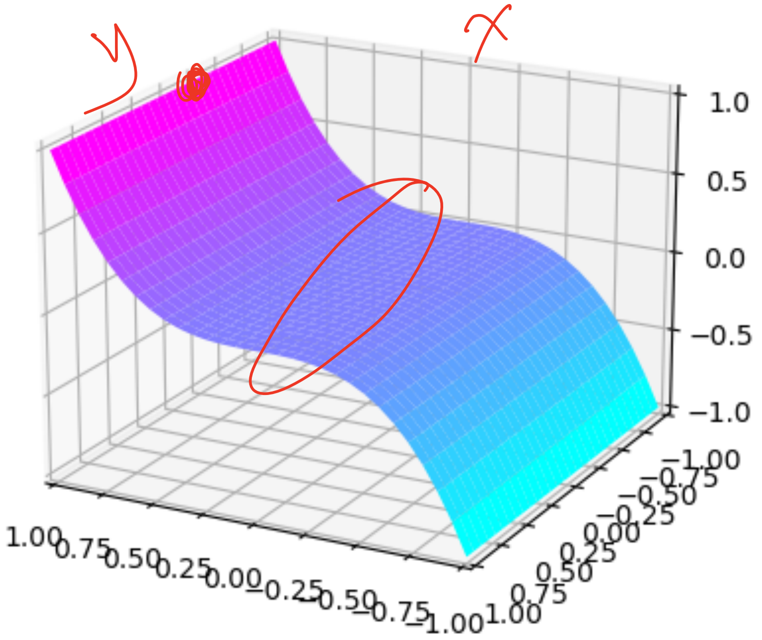


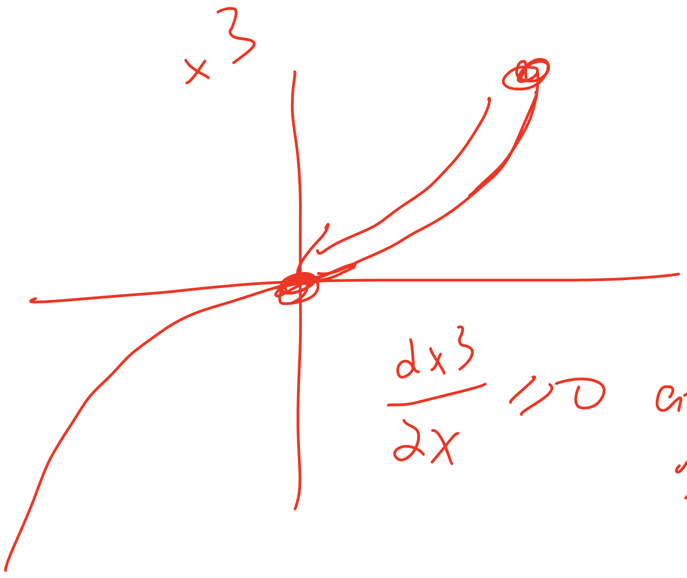
AdaGrad can make good progress on all axis

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)

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



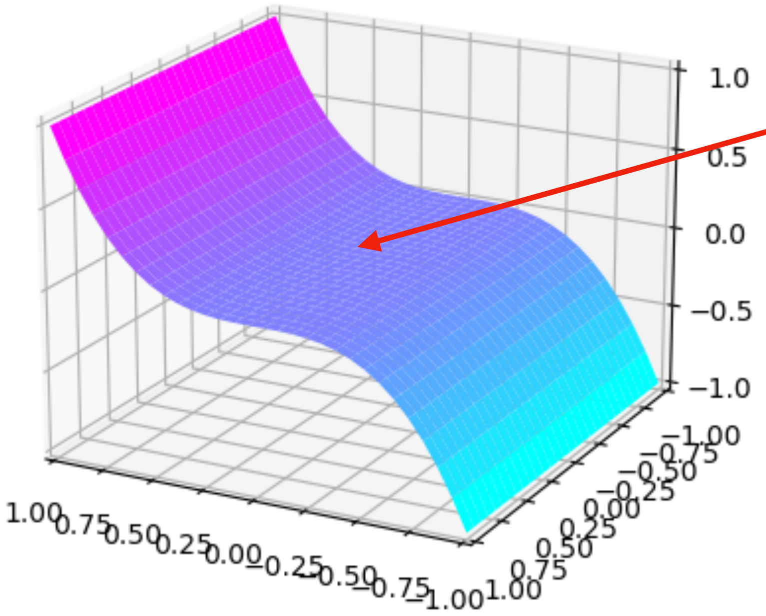


$\frac{dx^3}{dx} = 0$ at $x = 0$

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)

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



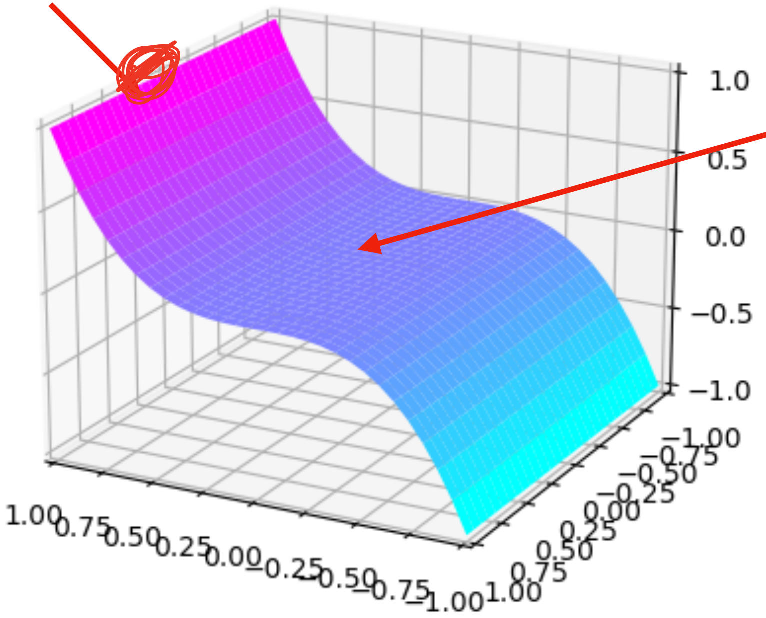
GD and Adagrad will get stuck here

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)

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

Q: what would happen if I drop a ball here



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)

Gradient Descent (GD) with Momentum

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

(The idea is motivated from physics)

$v^1 = (1-\alpha) g^1$

$v^2 = \alpha(1-\alpha) g^1 + (1-\alpha) g^2$

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

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

$\alpha \in [0, 1]$

$v^3 = \alpha^2 (1-\alpha) g^1 + \alpha(1-\alpha) g^2 + (1-\alpha) \cdot g^3$

current gradien

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:

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

$$(v^t = \underbrace{\alpha^{t-1}(1-\alpha)}_{w_1}g^1 + \underbrace{\alpha^{t-2}(1-\alpha)}_{w_2}g^2 + \ldots + \underbrace{\alpha(1-\alpha)}_{w_{t-1}}g^{t-1} + \underbrace{(1-\alpha)}_{w_t}g^t)$$

$$\text{Normalizer} = w_1 + w_2 + \cdots + w_t = (1-\alpha)\left[\alpha^{t-1} + \alpha^{t-2} \cdots 1\right] = (1-\alpha)\cdot\frac{1-\alpha^t}{1-\alpha} = 1-\alpha^t$$

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

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

$= 0$

Gradient Descent with Momentum

Putting things together:

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

For t = 1 ....

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

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

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

Normalier

Demo of GD w/ Momentum

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

Adam (Adaptive Momentum Estimation)

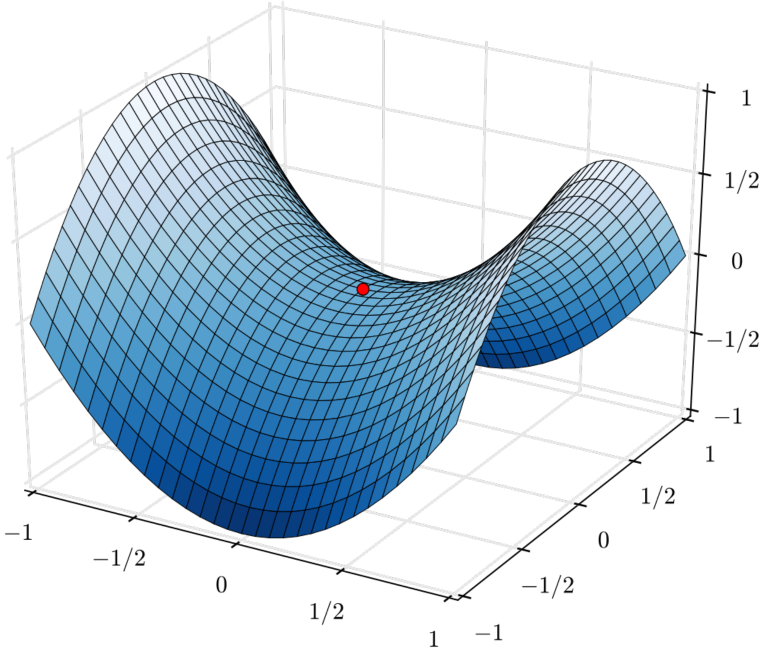
Adam = Momentum + AdaGrad

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

(The second paper reading quiz)

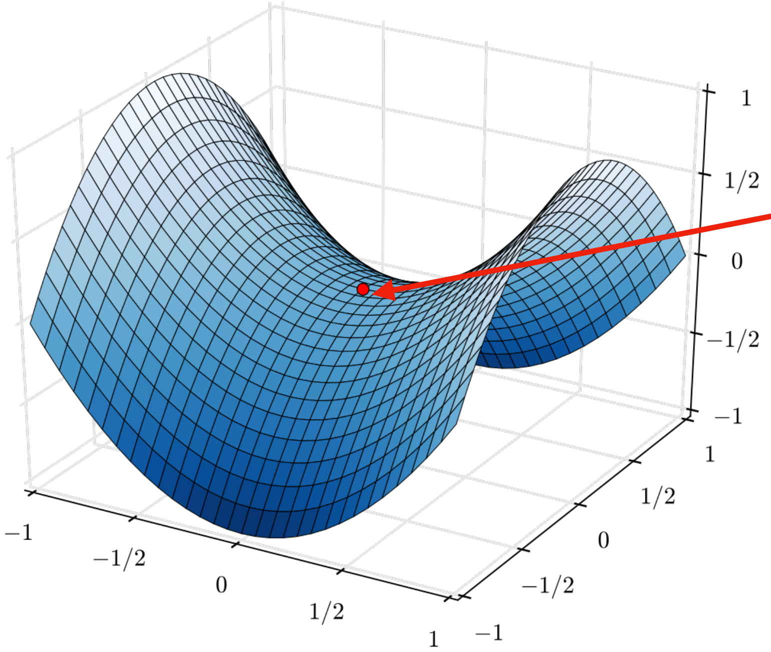
Even w/ AdaGrad + Momentum, we may still have issue

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



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

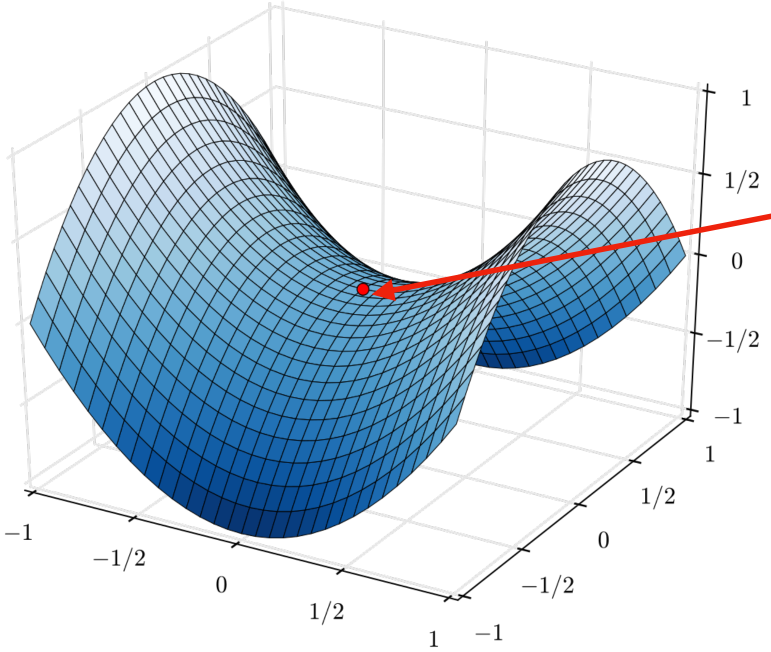
e.g., saddle point $x^2 - y^2$



Can stuck at the saddle point

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